

Modicon M221 Logic Controller

Advanced Functions Library Guide

12/2016

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2016 Schneider Electric. All Rights Reserved.

Table of Contents



	Safety Information	7
	About the Book	11
Part I	Introduction to Advanced Functions	17
Chapter 1	Generalities	19
	Expert I/O	21
	Embedded Expert I/O Mapping	23
	General Information on Function Block Management	26
Part II	Advanced Expert Input Functions	27
Chapter 2	Fast Counter (%FC)	29
	Description	30
	Configuration	32
	Programming Example	35
Chapter 3	High Speed Counter (%HSC)	37
	Description	38
	High Speed Counter in Counting Modes	43
	High Speed Counter in Frequency Meter Mode	50
Part III	Advanced Expert Output Functions	53
Chapter 4	Pulse (%PLS)	55
	Description	56
	Function Block Configuration	58
	Programming Example	62
Chapter 5	Pulse Width Modulation (%PWM)	63
	Description	64
	Function Block Configuration	65
	Programming Example	69
Chapter 6	Drive (%DRV)	71
	Description	72
	Drive and Logic Controller States	74
	Adding a Drive Function Block	76
	Function Block Configuration	78
	MC_Power_ATV: Enable/Disable Power Stage	79
	MC_Jog_ATV: Start Jog Mode	81
	MC_MoveVel_ATV: Move at Specified Velocity	84
	MC_Stop_ATV: Stop Movement	87

	MC_ReadStatus_ATV: Read Device Status	89
	MC_ReadMotionState_ATV: Read Motion State	92
	MC_Reset_ATV: Acknowledge and Reset Error	95
	Error Codes	97
Chapter 7	Pulse Train Output (%PTO)	101
7.1	Description	102
	Pulse Train Output (PTO)	103
	Pulse Output Modes	106
	Acceleration / Deceleration Ramp	108
	Probe Event	111
	Backlash Compensation	114
	Positioning Limits	116
7.2	Configuration	119
	PTO Configuration	120
	Motion Task Table	121
7.3	Programming	129
	Adding / Removing a Function Block	130
	PTO Function Blocks	132
7.4	Home Modes	134
	Homing Modes	135
	Position Setting	138
	Long Reference	139
	Short Reference No Reversal	141
	Short Reference Reversal	143
	Home Offset	145
7.5	Data Parameters	146
	Function Block Object Codes	146
7.6	Operation Modes	151
	Motion State Diagram	152
	Buffer Mode	154
7.7	Motion Function Blocks	156
	MC_MotionTask_PTO Function Block	157
	MC_Power_PTO Function Block	161
	MC_MoveVel_PTO Function Block	164
	MC_MoveRel_PTO Function Block	169
	MC_MoveAbs_PTO Function Block	174

	MC_Home_PTO Function Block	178
	MC_SetPos_PTO Function Block	181
	MC_Stop_PTO Function Block	183
	MC_Halt_PTO Function Block	186
7.8	Administrative Function Blocks	189
	MC_ReadActVel_PTO Function Block.....	190
	MC_ReadActPos_PTO Function Block.....	192
	MC_ReadSts_PTO Function Block	194
	MC_ReadMotionState_PTO Function Block.....	196
	MC_ReadAxisError_PTO Function Block.....	198
	MC_Reset_PTO Function Block	200
	MC_TouchProbe_PTO Function Block.....	202
	MC_AbortTrigger_PTO Function Block.....	204
	MC_ReadPar_PTO Function Block	206
	MC_WritePar_PTO Function Block	208
Chapter 8	Frequency Generator (%FREQGEN)	211
	Description.....	212
	Configuration.....	214
Part IV	Advanced Software Functions	217
Chapter 9	PID Function.....	219
9.1	PID Operating Modes	220
	PID Operating Modes	220
9.2	PID Auto-Tuning Configuration	222
	PID Auto-Tuning Configuration	222
9.3	PID Standard Configuration.....	226
	PID Word Address Configuration	227
	PID Tuning with Auto-Tuning (AT).....	230
	Manual Mode.....	233
	Determining the Sampling Period (Ts)	235
9.4	PID Assistant.....	238
	Access the PID Assistant	239
	General Tab.....	241
	Input Tab	244
	PID Tab	245
	AT Tab.....	247
	Output Tab.....	249

9.5	PID Programming	251
	Description	252
	Programming and Configuring	254
	PID States and Detected Error Codes	255
Appendices	259
Appendix A	PID Parameters	261
	Role and Influence of PID Parameters	262
	PID Parameter Adjustment Method	264
Glossary	267
Index	271

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book



At a Glance

Document Scope

This document provides descriptions of the SoMachine Basic advanced functions and their relation to the M221 Logic Controller expert I/O and PID support. Here you can find descriptions of the functionalities, characteristics and performances of the M221 Logic Controller advanced functions.

Validity Note

The information in this manual is applicable **only** for SoMachine Basic products.

This document has been updated for the release of SoMachine Basic V1.5.

The technical characteristics of the devices described in this document also appear online. To access this information online:

Step	Action
1	Go to the Schneider Electric home page www.schneider-electric.com .
2	In the Search box type the reference of a product or the name of a product range. <ul style="list-style-type: none">• Do not include blank spaces in the reference or product range.• To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the Product Datasheets search results and click on the reference that interests you. If you entered the name of a product range, go to the Product Ranges search results and click on the product range that interests you.
4	If more than one reference appears in the Products search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click Download XXX product datasheet .

The characteristics that are presented in this manual should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the manual and online information, use the online information as your reference.

Related Documents

Title of Documentation	Reference Number
SoMachine Basic Operating Guide	<i>EIO0000001354 (ENG)</i> <i>EIO0000001355 (FRA)</i> <i>EIO0000001356 (GER)</i> <i>EIO0000001357 (SPA)</i> <i>EIO0000001358 (ITA)</i> <i>EIO0000001359 (CHS)</i> <i>EIO0000001366 (POR)</i> <i>EIO0000001367 (TUR)</i>
SoMachine Basic Generic Functions - Library Guide	<i>EIO0000001474 (ENG)</i> <i>EIO0000001475 (FRA)</i> <i>EIO0000001476 (GER)</i> <i>EIO0000001477 (SPA)</i> <i>EIO0000001478 (ITA)</i> <i>EIO0000001479 (CHS)</i> <i>EIO0000001480 (POR)</i> <i>EIO0000001481 (TUR)</i>
Modicon M221 Logic Controller - Programming Guide	<i>EIO0000001360 (ENG)</i> <i>EIO0000001361 (FRE)</i> <i>EIO0000001362 (GER)</i> <i>EIO0000001363 (SPA)</i> <i>EIO0000001364 (ITA)</i> <i>EIO0000001365 (CHS)</i> <i>EIO0000001369 (TUR)</i> <i>EIO0000001368 (POR)</i>
Modicon M221 Logic Controller - Hardware Guide	<i>EIO0000001384 (ENG)</i> <i>EIO0000001385 (FRA)</i> <i>EIO0000001386 (GER)</i> <i>EIO0000001387 (SPA)</i> <i>EIO0000001388 (ITA)</i> <i>EIO0000001389 (CHS)</i> <i>EIO0000001370 (POR)</i> <i>EIO0000001371 (TUR)</i>

You can download these technical publications and other technical information from our website at <http://www.schneider-electric.com/ww/en/download>

Product Related Information

WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.¹
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

¹ For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
EN 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2008	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
EN 1088:2008 ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2006	Safety of machinery - Emergency stop - Principles for design
EN/IEC 62061:2005	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2008	Digital data communication for measurement and control: Functional safety field buses.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive (2006/42/EC)* and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Part I

Introduction to Advanced Functions

Chapter 1

Generalities

Overview

This document provides descriptions of the SoMachine Basic advanced functions and their relation to the M221 expert I/O and PID support. Here you can find descriptions of the functionalities, characteristics and performances of the Fast Counter (%FC), High Speed Counter (%HSC), Pulse (%PLS), Pulse Width Modulation (%PWM), and Pulse Train Output (%PTO) inputs and outputs. In addition, you can find a complete description of the PID advanced software functionality.

The functions provide simple yet powerful solutions for your application. However, the use and application of the information contained herein require expertise in the design and programming of automated control systems.

Only you, the user, machine builder or integrator, can be aware of all the conditions and factors present during installation and setup, operation, and maintenance of the machine or related processes, and can therefore determine the automation and associated equipment and the related safeties and interlocks which can be effectively and properly used. When selecting automation and control equipment, and any other related equipment or software, for a particular application, you must also consider any applicable local, regional or national standards and/or regulations.

 WARNING
REGULATORY INCOMPATIBILITY
Ensure that all equipment applied and systems designed comply with all applicable local, regional, and national regulations and standards.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

The functionality provided by the SoMachine Basic advanced functions for the M221 controllers was conceived and designed assuming that you incorporate the necessary safety hardware into your application architecture, such as, but not limited to, appropriate limit switches and emergency stop hardware and controlling circuitry. It is implicitly assumed that functional safety measures are present in your machine design to prevent undesirable machine behavior such as over-travel or other forms of uncontrolled movement. Further, it is assumed that you have performed a functional safety analysis and risk assessment appropriate to your machine or process.

 **WARNING**

UNINTENDED EQUIPMENT OPERATION

Ensure that a risk assessment is conducted and respected according to EN/ISO 12100 during the design of your machine.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Expert I/O	21
Embedded Expert I/O Mapping	23
General Information on Function Block Management	26

Expert I/O

Introduction

The M221 logic controller provides:

- Four fast inputs (%I0.0, %I0.1, %I0.6 and %I0.7)
- Two fast outputs on controller references that contain transistor outputs (%Q0.0 and %Q0.1)
- Four fast outputs on controller references TM221C40U and TM221CE40U (%Q0.0, %Q0.1, %Q0.2, and %Q0.3)

NOTE: No fast output functions are supported on controller references that contain relay outputs.

The M221 logic controller supports the following expert I/O functions (depending on the reference):

Functions		Description
Counters	Fast Counter (see page 29)	The FC function can execute fast counts of pulses from sensors, switches, and so on.
	High Speed Counter (see page 37)	The HSC function can execute fast counts of pulses from sensors, switches, and so on, that are connected to the fast inputs.
Pulse Generators	Pulse (see page 55)	The PLS function generates a square wave pulse signal on dedicated output channels.
	Pulse Width Modulation (see page 63)	The PWM function generates a modulated wave signal on dedicated output channels with a variable duty cycle.
	Pulse Train Output (see page 101)	The PTO function generates a pulse train output to control a linear single-axis stepper or servo drive in open loop mode.
	Frequency Generator (see page 211)	The FREQGEN function generates a square wave signal on a dedicated output channel with programmable frequency and duty cycle of 50%.

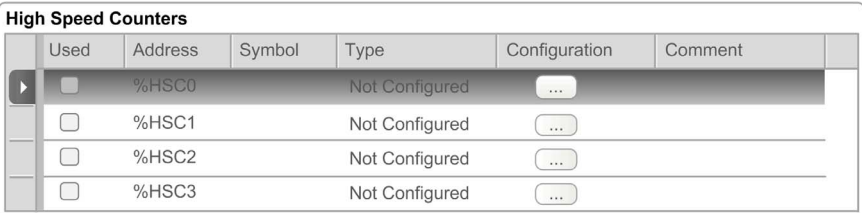
NOTE:

- When an input is used as Run/Stop, it cannot be used by an expert function.
- When an output is used as Alarm, it cannot be used by an expert function.

For more details, refer to Embedded Input/Output Configuration (see *Modicon M221, Logic Controller, Programming Guide*).

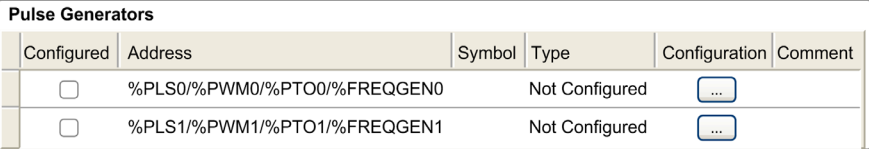
Configuring an Expert Input Function

To configure an expert input function, proceed as follows:

Step	Description																														
1	<p>Click the High Speed Counters node in the hardware tree. Result: The High Speed Counters list is displayed:</p>  <table border="1" data-bbox="278 342 1142 553"> <thead> <tr> <th>Used</th> <th>Address</th> <th>Symbol</th> <th>Type</th> <th>Configuration</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>%HSC0</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%HSC1</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%HSC2</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%HSC3</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> </tbody> </table>	Used	Address	Symbol	Type	Configuration	Comment	<input type="checkbox"/>	%HSC0		Not Configured	...		<input type="checkbox"/>	%HSC1		Not Configured	...		<input type="checkbox"/>	%HSC2		Not Configured	...		<input type="checkbox"/>	%HSC3		Not Configured	...	
Used	Address	Symbol	Type	Configuration	Comment																										
<input type="checkbox"/>	%HSC0		Not Configured	...																											
<input type="checkbox"/>	%HSC1		Not Configured	...																											
<input type="checkbox"/>	%HSC2		Not Configured	...																											
<input type="checkbox"/>	%HSC3		Not Configured	...																											
2	<p>Click ... in the Configuration column to select the type of high speed counter and to display the High Speed Counter Assistant window.</p>																														

Configuring an Expert Output Function

To configure an expert output function, proceed as follows:

Step	Description																		
1	<p>Click the Pulse Generators node in the hardware tree. Result: The Pulse Generators list is displayed:</p>  <table border="1" data-bbox="278 893 1149 1040"> <thead> <tr> <th>Configured</th> <th>Address</th> <th>Symbol</th> <th>Type</th> <th>Configuration</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>%PLS0/%PWM0/%PTO0/%FREQGEN0</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%PLS1/%PWM1/%PTO1/%FREQGEN1</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> </tbody> </table>	Configured	Address	Symbol	Type	Configuration	Comment	<input type="checkbox"/>	%PLS0/%PWM0/%PTO0/%FREQGEN0		Not Configured	...		<input type="checkbox"/>	%PLS1/%PWM1/%PTO1/%FREQGEN1		Not Configured	...	
Configured	Address	Symbol	Type	Configuration	Comment														
<input type="checkbox"/>	%PLS0/%PWM0/%PTO0/%FREQGEN0		Not Configured	...															
<input type="checkbox"/>	%PLS1/%PWM1/%PTO1/%FREQGEN1		Not Configured	...															
2	<p>Click [...] in the Configuration column to select the type of pulse generator and to display the Pulse Train Output Assistant window.</p>																		

Expert I/O Function Configuration Characteristics

- Inputs can be read through standard memory variables even if configured in association with expert I/O functions.
- Short-circuit management still applies on all expert outputs.
- All I/O that are not used by expert I/O functions can be used as regular I/O.
- Outputs used by the **Pulse**, **Pulse Train Output**, **Pulse Width Modulation**, and **High Speed Counters** can only be accessed through the expert I/O function block. They cannot be read or written directly within the application.

Embedded Expert I/O Mapping

Input Mapping for Expert Functions on M221 Logic Controller

Embedded digital inputs can be assigned to functions (Run/Stop, Latch, Event, Fast Counter, HSC, PTO). Inputs not assigned to functions are used as regular inputs. The following table presents the possible assignments of the embedded M221 Logic Controller digital inputs:

Function		Simple Input Function			Advanced Input Function		
		Run/Stop	Latch	Event	Fast Counter	HSC	PTO ⁽³⁾
Fast Input	%I0.0	X	–	–	–	%HSC0	–
	%I0.1	X	–	–	–	%HSC0 or %HSC2 ⁽¹⁾	–
Regular Input	%I0.2	X	X	X	%FC0	Preset for %HSC0	Ref or probe for %PTO0 to %PTO3
	%I0.3	X	X	X	%FC1	Catch for %HSC0	
	%I0.4	X	X	X	%FC2	Catch for %HSC1	
	%I0.5	X	X	X	%FC3	Preset for %HSC1	
Fast Input	%I0.6	X	–	–	–	%HSC1	–
	%I0.7	X	–	–	–	%HSC1 or %HSC3 ⁽²⁾	–
X Yes – No (1) %HSC2 is available when %HSC0 is configured as Single Phase or Not Configured. (2) %HSC3 is available when %HSC1 is configured as Single Phase or Not Configured. (3) PTO function is available on controller references that contain transistor outputs.							

Function		Simple Input Function			Advanced Input Function		
		Run/Stop	Latch	Event	Fast Counter	HSC	PTO ⁽³⁾
Regular Input (depending on the controller reference)	%I0.8	X	-	-	-	-	Ref or probe for %PTO0 to %PTO3 on TM221C40U and TM221CE40U controllers
	%I0.9	X	-	-	-	-	
	%I0.10	X	-	-	-	-	-
	%I0.11	X	-	-	-	-	-
	%I0.12	X	-	-	-	-	-
	%I0.13	X	-	-	-	-	-
	%I0.14	X	-	-	-	-	-
	%I0.15	X	-	-	-	-	-
	%I0.16	X	-	-	-	-	-
	%I0.17	X	-	-	-	-	-
	%I0.18	X	-	-	-	-	-
	%I0.19	X	-	-	-	-	-
	%I0.20	X	-	-	-	-	-
	%I0.21	X	-	-	-	-	-
	%I0.22	X	-	-	-	-	-
%I0.23	X	-	-	-	-	-	

X Yes
- No
(1) %HSC2 is available when %HSC0 is configured as Single Phase or Not Configured.
(2) %HSC3 is available when %HSC1 is configured as Single Phase or Not Configured.
(3) PTO function is available on controller references that contain transistor outputs.

Output Mapping for Expert Functions on M221 Logic Controller

The information below refers to regular and fast transistor outputs on M221 Logic Controller:

Function		Alarm Output	HSC	PLS / PWM / PTO / FREQGEN
Fast Output ⁽¹⁾	%Q0.0	X	–	<ul style="list-style-type: none"> • %PLS0 • %PWM0 • %PTO0 • %FREQGEN0
	%Q0.1	X	–	<ul style="list-style-type: none"> • %PLS1 • %PWM1 • %PTO⁽²⁾ • %FREQGEN1
Regular Output ⁽³⁾ (depending on the controller reference)	%Q0.2	X	Reflex output 0 for %HSC0 or %HSC2	<ul style="list-style-type: none"> • %PTO⁽⁴⁾ • %FREQGEN2
	%Q0.3	X	Reflex output 1 for %HSC0 or %HSC2	<ul style="list-style-type: none"> • %PTO⁽⁵⁾ • %FREQGEN3
	%Q0.4	X	Reflex output 0 for %HSC1 or %HSC3	%PTOx direction
	%Q0.5	X	Reflex output 1 for %HSC1 or %HSC3	%PTOx direction
	%Q0.6	X	–	%PTOx direction
	%Q0.7	X	–	%PTOx direction
	%Q0.8	–	–	%PTOx direction
	%Q0.9	–	–	%PTOx direction
	%Q0.10	–	–	%PTOx direction
	%Q0.11	–	–	%PTOx direction
	%Q0.12	–	–	%PTOx direction
	%Q0.13	–	–	%PTOx direction
	%Q0.14	–	–	%PTOx direction
	%Q0.15	–	–	%PTOx direction

(1) Fast output functions are only available on controller references that contain transistor outputs.

(2) %PTO0 direction in CW/CCW output mode, or %PTO1 (not available when %PTO0 is configured in CW/CCW output mode), or %PTOx direction in other cases.

(3) %Q0.2 and %Q0.3 are fast outputs on TM221C40U and TM221CE40U controllers

(4) %PTO2 on TM221C40U and TM221CE40U controllers, or %PTOx direction in other cases.

(5) %PTO2 direction in CW/CCW output mode on TM221C40U and TM221CE40U controllers, or %PTO3 (not available when %PTO2 is configured in CW/CCW output mode) on TM221C40U and TM221CE40U controllers, or %PTOx direction in other cases.

General Information on Function Block Management

Management of Function Block Inputs and Input Objects

The variables (function block inputs and input objects) are used with the rising edge of the `Execute` input. To modify any variable, it is necessary to change the input variables and to trigger the function block again. However, there are some function blocks that do provide a continuous update option.

Management of Function Block Outputs and Output Objects

The `Done`, `Error`, `Busy`, and `CmdAborted` outputs are mutually exclusive: only one of them can be TRUE on one function block. When the `Execute` input is TRUE, one of these outputs is TRUE.

At the rising edge of the `Execute` input, the `Busy` output is set to TRUE. It remains TRUE during the execution of the function block and is reset at the rising edge of one of the other outputs (`Done`, `Error` and `CmdAborted`).

The `Done` output is TRUE when the execution of the function block has completed successfully.

If an error is detected, the function block terminates by setting the `Error` output to TRUE, and the error code is contained within the `ErrId` output.

The `Done`, `Error`, and `CmdAborted` outputs are set to TRUE or FALSE with the falling edge of the `Execute` input, according to the following conditions:

- set for one task cycle if the function block execution is finished and the `Execute` input is FALSE and then reset to their default values.
- retain their value if the function block execution is finished and the `Execute` input is TRUE.

When an instance of a function block receives a new `Execute` before it is finished (as a series of commands on the same instance), the function block does not return any feedback, like `Done`, for the previous action. However, the new command is started on the function block (status is `Busy`).

Error Handling

All blocks have two outputs that can report errors detected during the execution of the function block:

- `Error`= The rising edge of this output indicates that an error was detected.
- `ErrID`= The error code of the error detected.

Part II

Advanced Expert Input Functions

Overview

This part describes the advanced expert input functions.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
2	Fast Counter (%FC)	29
3	High Speed Counter (%HSC)	37

Chapter 2

Fast Counter (%FC)

Using Fast Counter Function Blocks

This chapter provides descriptions and programming guidelines for using `Fast Counter` function blocks.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	30
Configuration	32
Programming Example	35

Description

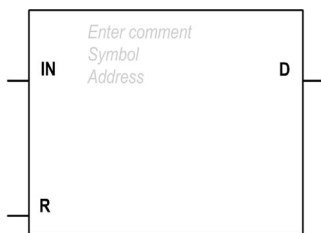
Introduction

The `Fast Counter` function block **1123** serves as either an up-counter or a down-counter. It can count the rising edge of digital inputs up to frequencies of 5 kHz in single word or double word computational mode. Because `Fast Counter` function blocks are managed by specific hardware interrupts, maintaining maximum frequency sampling rates may vary depending on your specific application and hardware configuration.

The `Fast Counter` function blocks `%FC0`, `%FC1`, `%FC2`, and `%FC3` use dedicated inputs `%I0.2`, `%I0.3`, `%I0.4` and `%I0.5` respectively. These bits are not reserved for their exclusive use. Their allocation must be considered with the use of other function blocks for these dedicated resources.

Illustration

This illustration is a `Fast Counter` function block in single-word mode:



Inputs

The `Fast Counter` function block has the following inputs:

Label	Description	Value
IN	Enable	At state 1, the current value is updated according to the pulses applied to the physical input. At state 0, the current value is held at its last value.
R	Reset (optional)	Used to initialize the block. At state 1, the current value is reset to 0 if configured as an up-counter, or set to <code>%FC.P</code> or <code>%FC.PD</code> if configured as a down-counter. The Done bit <code>%FC.D</code> is set back to its default value.

Outputs

The `Fast Counter` function block has the following output:

Label	Description	Value
D	Done (%FCi.D)	<p>This bit is set to 1 when:</p> <ul style="list-style-type: none">• %FCi.V or %FCi.VD reaches the preset value %FCi.P or %FCi.PD configured as an up-counter.• or when %FCi.V or %FCi.VD reaches 0 when configured as a down-counter. <p>This read-only bit is reset only by setting %FCi.R to 1.</p>

Configuration

Parameters

To configure parameters, follow the Configuring a Function Block procedure (*see SoMachine Basic, Generic Functions Library Guide*) and read the description of Memory Allocation Modes in the SoMachine Basic Operating Guide (*see SoMachine Basic, Operating Guide*).

The `Fast Counter` function block has the following parameters:

Parameter	Description	Value
Used	Address used	If selected, this address is currently in use in a program.
Address	<code>%FCi.Fast Counter address</code>	The instance identifier, where it is from 0 to the number of objects available on this logic controller. Refer to Maximum Number of Objects table (<i>see Modicon M221, Logic Controller, Programming Guide</i>) for the maximum number of <code>Fast Counters</code> .
Input	<code>%IO.i</code>	The dedicated input associated with this function block instance. <code>%IO.2...%IO.5</code>
Symbol	Symbol	The symbol associated with this object. Refer to the SoMachine Basic Operating Guide (Defining and Using Symbols) for details.
Configured	Whether to count up or down	Set to one of: <ul style="list-style-type: none"> ● Not used ● Up Counter ● Down Counter
Preset	Preset value (<code>%FCi.P</code> or <code>%FCi.PD</code>)	Initial value may be set: <ul style="list-style-type: none"> ● Using associated object <code>%FCi.P</code> from 1 to 65535 in single word mode, ● Using associated object <code>%FCi.PD</code> from 1 to 4294967295 in double word mode.
Double Word	Double word mode	If selected, use double word mode. Otherwise, use single-word mode.
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Objects

The `Fast Counter` function block is associated with the following objects:

Object	Description	Value
%FCi.V %FCi.VD	Current value	The current value increments or decrements according the up or down counting function selected. For up-counting, the current counting value is updated and can reach 65535 in single word mode (%FCi.V) and 4294967295 in double word mode (%FCi.VD). For down-counting, the current value is the preset value %FC.P or %FC.PD and can count down to 0.
%FCi.P %FCi.PD	Preset value	See description in Parameters table above.
%FCi.D	Done	See description in Outputs table above.

Special Note

The application can change the preset value %FCi.P or %FCi.PD and the current value %FCi.V or %FCi.VD at any time. A new value is taken into account only if the R input is active or at the rising edge of the D output %FC.D. This allows for successive different counts without the loss of a single pulse.

Operation

This table describes the main stages of `Fast Counter` function block operations:

Operation	Action	Result
Count up	A rising edge appears at the Count up input.	The current value %FCi.V is incremented by 1 unit.
	When the preset value %FCi.P or %FCi.PD is reached.	The Done output bit %FCi.D is set to 1.
Count down	A rising edge appears at the down-counting input.	The current value %FCi.V is decremented by 1 unit.
	When the value is 0.	The Done output bit %FCi.D is set to 1.

Special Cases

This table contains a list of special operating cases for the `Fast Counter` function block:

Special Case	Description
Effect of cold restart (%S0=1)	Resets the <code>Fast Counter</code> attributes with the values configured or user application.
Effect of warm restart (%S1=1)	No effect.
Effect of controller stops	The <code>Fast Counter</code> stops counting when the controller is set to <code>STOPPED</code> state and resumes counting when it returns to <code>RUNNING</code> state. The counter resumes counting from the last value before entering the <code>STOPPED</code> state.

Programming Example

Introduction

In this example, the application counts a number of items up to 5000 while %I0.1 is set to 1. The input for %FC1 is the dedicated input %I0.3. When the preset value is reached, %FC1.D is set to 1 and retains the same value until %FC1.R is commanded by the result of AND on %I0.2 and %M0.

Programming

This example is a Fast Counter function block:

Rung	Instruction
0	BLK %FC1 LD %I0.1 IN LD %I0.2 AND %M0 R OUT_BLK LD D ST %Q0.0 END_BLK

NOTE: Refer to the reversibility procedure (*see SoMachine Basic, Generic Functions Library Guide*) to obtain the equivalent Ladder Diagram.

Chapter 3

High Speed Counter (%HSC)

Using High Speed Counter Function Blocks

This chapter provides descriptions and programming guidelines for using High Speed Counter function blocks.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	38
High Speed Counter in Counting Modes	43
High Speed Counter in Frequency Meter Mode	50

Description

Introduction

The High Speed Counter function block **11123** can be configured by SoMachine Basic to perform any one of the following functions:

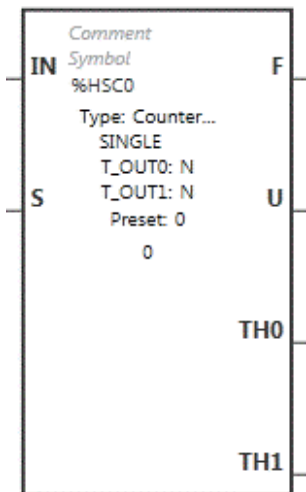
- Dual Phase [Pulse / Direction]
- Dual Phase [Quadrature X1]
- Dual Phase [Quadrature X2]
- Dual Phase [Quadrature X4]
- Single Phase
- Frequency Meter

The High Speed Counter function block works at a maximum frequency of 100 kHz for all counting modes with a range of 0 to 65535 in single word and 0 to 4294967295 in double word.

The High Speed Counter function block uses dedicated inputs and auxiliary inputs and outputs. Refer to the M221 Logic Controller - Hardware Guide for more information on inputs and outputs.

You must initialize the High Speed Counter function in the **Configuration** tab using the **High Speed Counter Assistant** before using an instance of the function block. Refer to Configuring High Speed Counters (*see Modicon M221, Logic Controller, Programming Guide*).

Graphical Representation



Inputs

The High Speed Counter function block has the following inputs:

Label	Description	Value
IN	Enable (required) At state 1, the counting function or frequency measurement is enabled. At state 0, the present value is held at its last value.	0 or 1
S	Preset input. At state 1: <ul style="list-style-type: none"> ● initializes the value with the preset value for: <ul style="list-style-type: none"> ○ Dual Phase [Quadrature X1], ○ Dual Phase [Quadrature X2], ○ Dual Phase [Quadrature X4], or ○ Dual Phase [Pulse / Direction] with down function in progress ● resets the value to 0 for: <ul style="list-style-type: none"> ○ Single Phase, or ○ Dual Phase [Pulse / Direction] with up function in progress <p>In addition, this also initializes the operation of the threshold outputs and takes into account any user modifications to the threshold values set in the properties window or the program.</p>	0 or 1

The High Speed Counter function block is associated with the following input objects:

Object	Type	Description	Value
%HSCi.P %HSCi.PD	WORD DOUBLE WORD	Preset value	Refer to Auxiliary Inputs (<i>see page 44</i>).
%HSCi.S0 %HSCi.S0D	WORD DOUBLE WORD	Threshold 0	Refer to Output Threshold in Counting Modes (<i>see page 43</i>).
%HSCi.S1 %HSCi.S1D	WORD DOUBLE WORD	Threshold 1	Refer to Output Threshold in Counting Modes (<i>see page 43</i>).
%HSCi.T	WORD	Time base	Refer to High Speed Counter in Frequency Meter Mode (<i>see page 50</i>).
%HSCi.R	BOOL	Enable reflex output 0	At state 1 enables the reflex output 0.
%HSCi.S	BOOL	Enable reflex output 1	At state 1 enables the reflex output 1.

NOTE: The %HSCi.R and %HSCi.S bits respectively enable or disable the reflex outputs only if the HSC function block is enabled, that is, if %HSCi.IN is set to 1.

Outputs

The High Speed Counter function block has the following outputs:

Label	Description	Value
F	Overflow Set to 1 if an arithmetic overflow occurs.	0 or 1
U	Counting direction Set by the system, this bit is used by the Dual Phase counting functions to indicate the direction of counting.	0: Down counting 1: Up counting
TH0	Threshold bit 0 Set to 1 when the present value is greater than or equal to the threshold value S0 (%HSCi.S0). Test this bit only once in the program because it is updated in real time. The user application is responsible for the validity of the value at its time of use.	0 or 1
TH1	Threshold bit 1 Set to 1 when the present value is greater than or equal to the threshold value S1 (%HSCi.S1). Test this bit only once in the program because it is updated in real time.	0 or 1

The High Speed Counter function block is associated with the following output objects:

Object	Type	Description	Value
%HSCi.V %HSCi.VD	WORD DOUBLE WORD	Present value	Refer to High Speed Counter in Counting Modes (<i>see page 43</i>) and to High Speed Counter in Frequency Meter Mode (<i>see page 50</i>).
%HSCi.C %HSCi.CD	WORD DOUBLE WORD	Capture value	Refer to Auxiliary Inputs (<i>see page 44</i>).
%HSCi.U	BOOL	Counting direction	0: Down counting 1: Up counting
%HSCi.F	BOOL	Overflow	0: No overflow 1: Counter overflow

Properties

The High Speed Counter function block has the following properties:

Property	Value	Description
Used	Activated / deactivated checkbox	Indicates whether the address is in use.
Address	%HSCi, where i is from 0 to 3, depending on the type(s) of counters configured	i is the instance identifier. For the maximum number of %HSC objects, refer to the table Maximum Number of Objects (see <i>Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	User-defined text	The symbol that uniquely identifies this object. For details, refer to <i>Defining and Using Symbols</i> (see <i>SoMachine Basic, Operating Guide</i>).
Preset	<ul style="list-style-type: none"> from 0 to 65535 for %HSCi.P from 0 to 4294967295 for %HSCi.PD 	Preset value to initialize the HSC present value (%HSCi.P, %HSCi.PD). Not valid for the Frequency Meter.
S0	<ul style="list-style-type: none"> from 1 to 65535 for %HSCi.S0 from 1 to 4294967295 for %HSCi.S0D 	Threshold 0 value is used as a comparator with the present value. The value of S0 must be less than S1 (%HSCi.S1).
S1	<ul style="list-style-type: none"> from 2 to 65535 for %HSCi.S1 from 2 to 4294967295 for %HSCi.S1D 	Threshold 1 value is used as a comparator with the present value. The value of S1 must be greater than S0 (%HSCi.S0).
Time Base	100 ms or 1 s for %HSCi.T	Frequency measurement time base.
Comment	User-defined text	A comment to associate with this object.

Special Cases

This table shows a list of special operating of the High Speed Counter function block:

Special Case	Description
Effect of cold restart (%S0=1)	Resets the High Speed Counter attributes with the values configured by the program.
Effect of warm restart (%S1=1)	Has no effect.

Special Case	Description
Effect of controller stop	<p>The High Speed Counter stops its function and the outputs stay in their present state.</p> <p>NOTE: When the controller stops, the reflex outputs are set to 0 if Maintain values is selected for the outputs. Otherwise, if Maintain values is not selected, the reflex outputs take the fallback values. For more information on configuring fallback behavior, refer to Fallback Behavior (<i>see SoMachine Basic, Operating Guide</i>).</p>

High Speed Counter in Counting Modes

Introduction

The High Speed Counter function block works at a maximum frequency of 100 kHz for all counting modes with a range of 0 to 65535 in single word and 0 to 4294967295 in double word.

The pulses to be counted are applied in the following way:

Function	Description	Input type	%HSC0	%HSC1	%HSC2	%HSC3
Dual Phase [Pulse / Direction]	The pulses are applied to the physical input associated to Pulse Input .	Pulse Input	%I0.0	%I0.6	–	–
	The present operation (upcount/downcount) is given by the state of the Direction Input : <ul style="list-style-type: none"> ● 0 = up counting ● 1 = down counting 	Direction Input	%I0.1	%I0.7	–	–
Dual Phase [Quadrature X1], Dual Phase [Quadrature X2], or Dual Phase [Quadrature X4]	The 2 phases of the encoder are applied to physical inputs associated to Pulse Input Phase A and Pulse Input Phase B .	Pulse Input Phase A	%I0.0	%I0.6	–	–
		Pulse Input Phase B	%I0.1	%I0.7	–	–
Single Phase	The pulses are applied to the physical input associated to Pulse Input .	Pulse Input	%I0.0	%I0.6	%I0.1	%I0.7

NOTE: I/O assignment is different between the Twido platform and M221 Logic Controller range. On the M221 Logic Controller, the main pulse input is %I0.0 for %HSC0 and %I0.6 for %HSC1. On the Twido platform, the main pulse input is %I0.1 for %HSC0 and %I0.7 for %HSC1.

Output Thresholds

During counting, the current value is compared to two thresholds: %HSCi.S0 or %HSCi.S0D and %HSCi.S1 or %HSCi.S1D. Changes to these threshold values are only taken into account if **Preset input** is set to 1 (function block input or auxiliary input).

According to the result of the comparisons, the bit objects, %HSCi.TH0 and %HSCi.TH1, are:

- set to 1 if the current value is greater than or equal to the corresponding threshold
- reset to 0 if the current value is less than the corresponding threshold.

Physical reflex outputs can be configured to respond differentially within the context of the compare results of the threshold values and the current value of the counters.

NOTE: None, 1 or 2 reflex outputs can be configured.

For more information on the configuration of reflex outputs, refer to *Configuring Dual Phase and Single Phase Counters (see Modicon M221, Logic Controller, Programming Guide)*.

`%HSCi.U` is an output of the function block; it gives the direction of the associated counter variation (1 for UP, 0 for DOWN).

Auxiliary Inputs

Counting operations are made on the rising edge of pulses, and only if the counting function block is enabled (**IN** input at state 1).

There are two optional inputs used in counting mode: **Catch Input** and **Preset Input**:

- A rising edge of the **Catch Input** is used to capture the current value (`%HSCi.V` or `%HSCi.VD`) and store it in `%HSCi.C` or `%HSCi.CD`. The catch inputs are specified as `%I0.3` for `%HSC0` and `%I0.4` for `%HSC1` if available.
- A rising edge of the **Preset Input** initializes `%HSCi.V` or `%HSCi.VD` value with the preset value for:
 - Dual Phase [Quadrature X1]
 - Dual Phase [Quadrature X2]
 - Dual Phase [Quadrature X4]
 - Dual Phase [Pulse / Direction] with down function in progress

The **Preset Input** resets the value to 0 for:

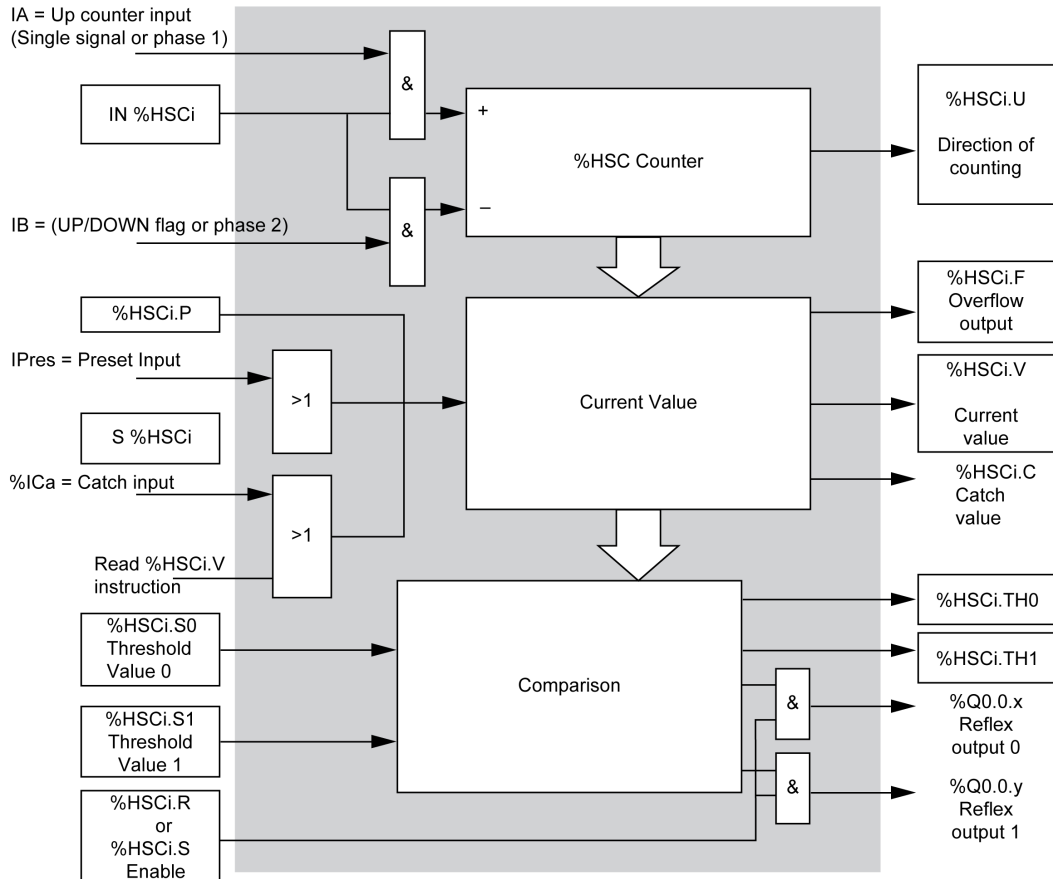
- Single Phase
- Dual Phase [Pulse / Direction] with up function in progress

If the auxiliary **Preset Input** is set to 1 with the input **IN** at 0 (the function is inhibited), the outputs are not monitored and maintain their values.

NOTE: `%HSCi.F` is also set to 0. The **Preset Input** is specified as `%I0.2` for `%HSC0` and/or `%I0.5` for `%HSC1`.

Operation

This illustration is the operation diagram of the counting mode in single word mode (in double word mode, use the double word function variables):



NOTE: Reflex outputs are managed independently from the controller cycle time.

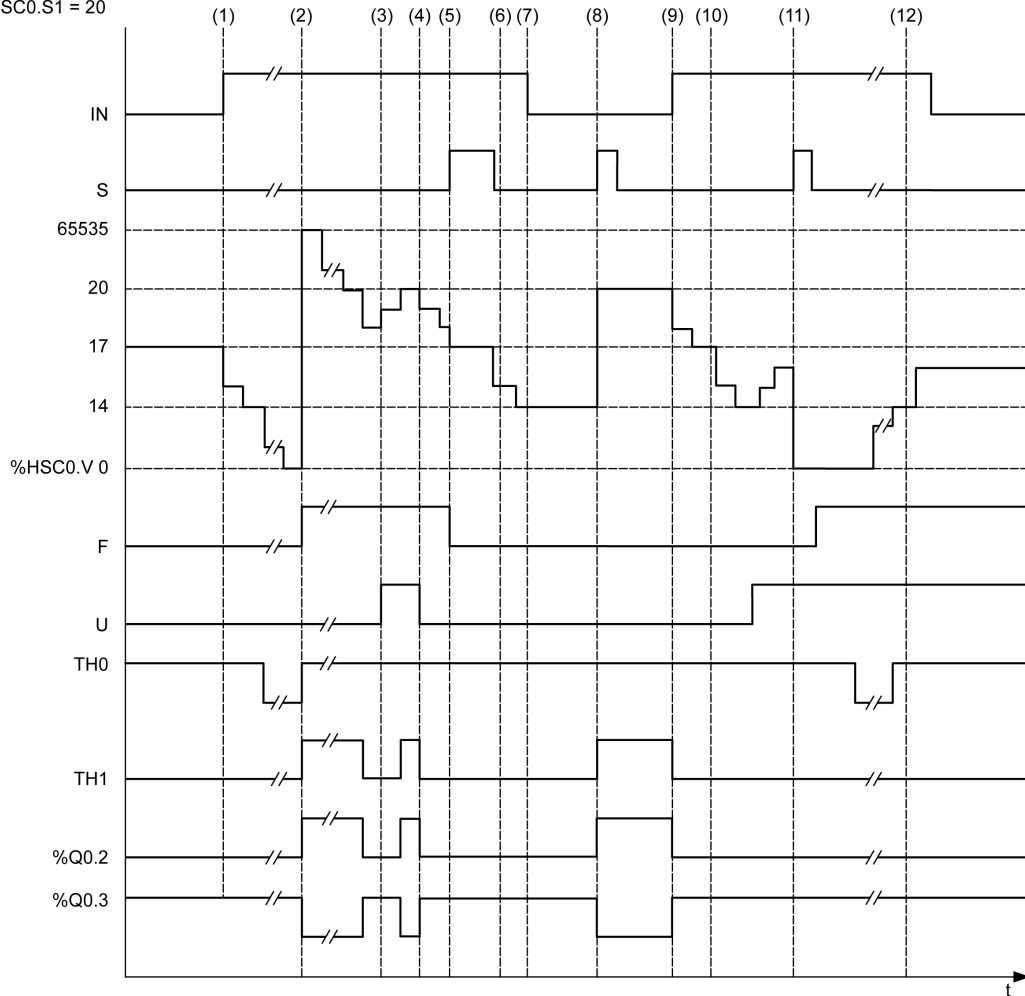
Dual Phase [Pulse / Direction] Timing Diagram

Reflex output configuration example:

Reflex Output	Value < %HSC0.S0	%HSC0.S0 <= Value < %HSC0.S1	Value >= %HSC0.S1
%Q0.2	0	0	1
%Q0.3	1	1	0

Timing diagram:

%HSC0.P = 17
 %HSC0.S0 = 14
 %HSC0.S1 = 20



- (1) Input IN is set to 1 so down-counting mode starts (%HSC0.U = 0 that is, IB = 1)
- (2) The current value reaches 0 so F output flag is set to 1 and %HSC0.V is set to 65535 at the next count
- (3) Change at the IB input, the counter is now in up counting mode and %HSC0.U = 1
- (4) IB input is set to 1 so the counter is in down counting mode and %HSC0.U is set to 0
- (5) Input S is set to 1 while down counting is in progress, so %HSC0.V is initialized to the Preset value
 %HSC0.P = 17

- (6) S is reset to 0 and the preset value %HSC0.P is changed to 20
- (7) The input IN is set to 0 so the function is inhibited, %HSC0.V is held
- (8) S is set to 1 so the new preset value (%HSC0.P = 20) is taken into account and the reflex outputs are updated. **Note:** If an auxiliary preset input is used instead of S, the reflex outputs are not updated in accordance with the Twido family of controllers.
- (9) IN input is set to 1 and the function restarts in down counting mode
- (10) The threshold value %HSC0.S1 is set to 17
- (11) S input active makes threshold S1 new value to be granted at the next count and resets %HSC0.V to 0
- (12) A catch of the current value %HSC0.V is made so %HSC0.C = 14

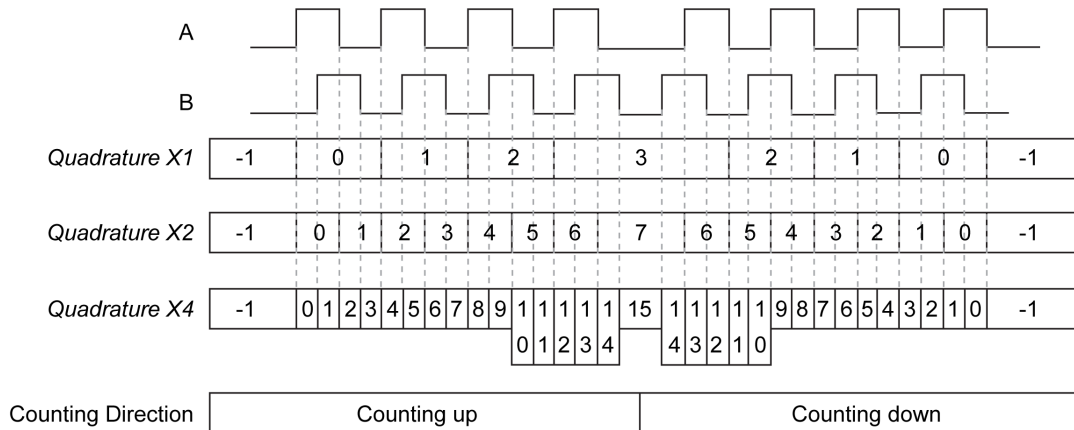
NOTE: %HSC0.R and %HSC0.S must be set to TRUE to have the configured reflex outputs active.

Dual Phase [Quadrature X1], Dual Phase [Quadrature X2], Dual Phase [Quadrature X4] Timing Diagram

A physical encoder provides two 90° shifted signals that allow the counter to count pulses and detect direction:

- X1** 1 count for each encoder cycle
- X2** 2 counts for each encoder cycle
- X4** 4 counts for each encoder cycle

Timing diagram:



- Quadrature X1** When channel A leads channel B, the counter increments on the rising edge of channel A. When channel B leads channel A, the counter decrements on the falling edge of channel A.
- Quadrature X2** Counter increments or decrements on each edge of channel A, depending on which channel leads the other. Each cycle results in two increments or decrements.
- Quadrature X4** The counter increments or decrements on each edge of channels A and B. Whether the counter increments or decrements depends on which channel leads the other. Each cycle results in 4 increments or decrements.

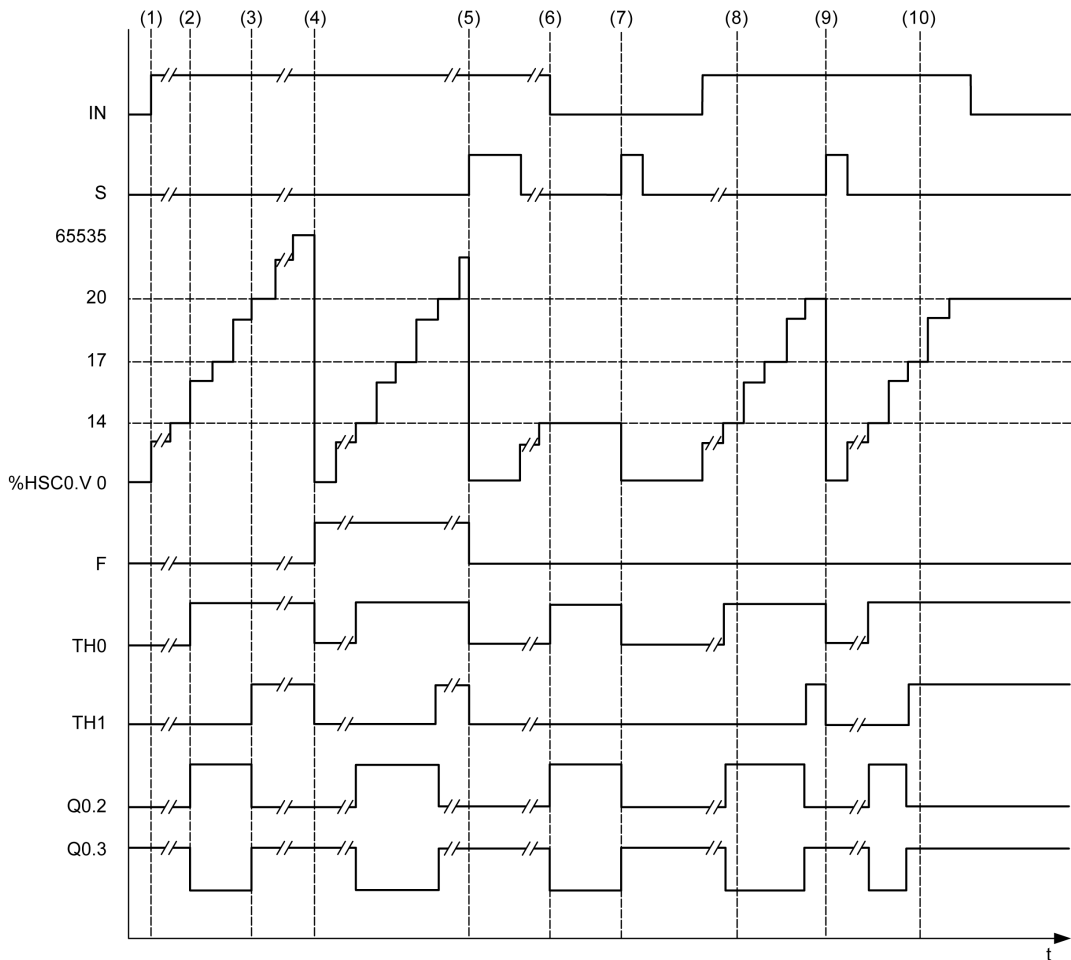
Single Phase Timing Diagram

Reflex output configuration example:

Reflex Output	Value < %HSC0.S0	%HSC0.S0 <= Value < %HSC0.S1	Value >= %HSC0.S1
%Q0.2	0	1	0
%Q0.3	1	0	1

Timing diagram:

%HSC0.P = 17
 %HSC0.S0 = 14
 %HSC0.S1 = 20



- (1) IN is set to 1: the counting function is activated ($\%HSC0.U = 1$ because %HSC0 is an up-counter)
- (2) %Q0.2 (Reflex Output) and TH0 are set to 1
- (3) TH1 is set to 1
- (4) The maximum value is reached so on the next count %HSC0.V is reset to 0 and F is set to 1
- (5) S is set to 1, the current value, %HSC0.V, is set to 0
- (6) The current function is inhibited while IN is set to 0
- (7) While the function is inhibited, S is set to 1 so the current value is reset to 0
- (8) Change of threshold value S1 to 17
- (9) S is set to 1 so the new value of S1 will be granted at the next count
- (10) Catch input is set to 1 so %HSC0.C = 17

High Speed Counter in Frequency Meter Mode

Introduction

The frequency meter mode of an High Speed Counter is used to measure the frequency of a periodic signal in Hz on input IA (pulse input phase A).

The frequency range which can be measured is 1 Hz to 100 kHz with a range of 0 to 4294967295 in double word mode.

It is possible to choose between 2 time bases, the choice being made by the object %HSC.T (Time base):

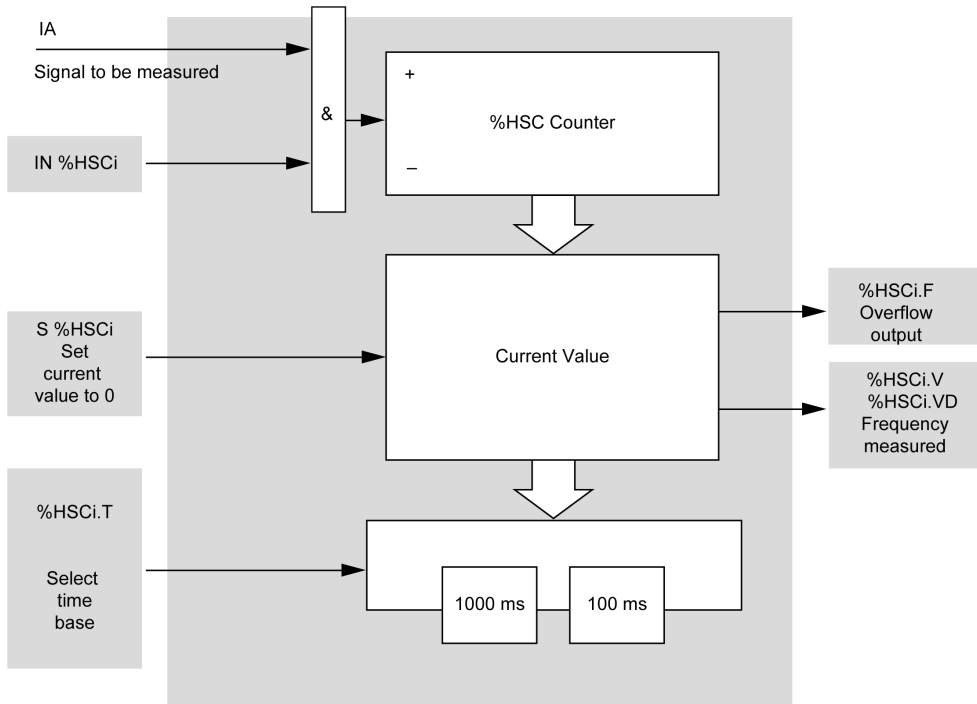
Time Base	Accuracy	Update
100 ms	0.01% for 100 kHz 10% for 100 Hz	10 times per second
1 s	0.001% for 100 kHz 10% for 10 Hz	Once per second

Accuracy Measurement

$$Accuracy(\%) = \frac{1}{f[Hz]} \times \frac{1}{TB[s]} \times 100$$

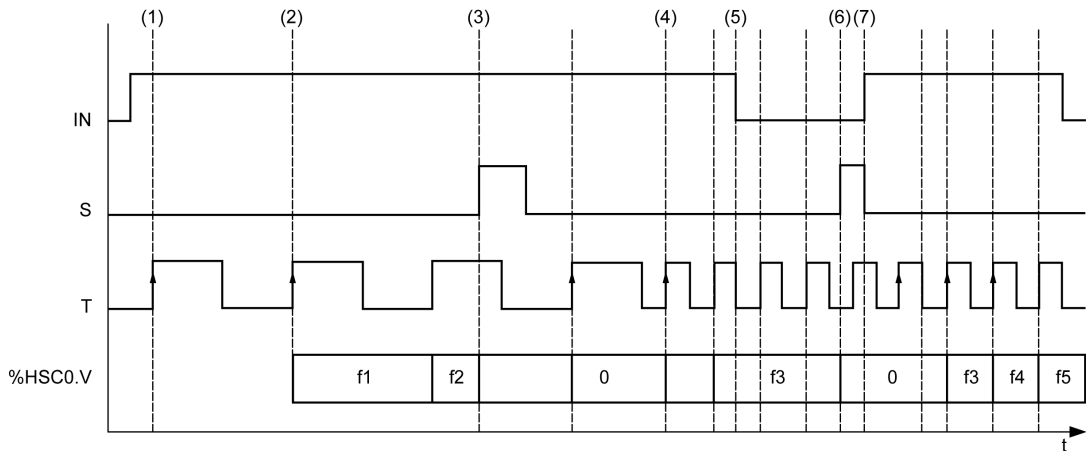
Operation

This illustration is the operation diagram of the frequency meter mode:



Timing Diagram

This timing diagram is an example of using a High Speed Counter in frequency meter mode:



- (1) The first frequency measurement starts at a rising edge of the TB signal
- (2) %HSC0.V (or %HSC0.VD) is updated after one period of the TB
- (3) Input IN and input S are set to 1 so %HSC0.V (or %HSC0.VD) is set to 0
- (4) %HSC0.T is set to 100 ms, so the measurement is canceled and a new one starts
- (5) Input IN is set to 0, so the frequency measurement function is inhibited and %HSC0.V (or %HSC0.VD) is held
- (6) S is set to 1, so the value %HSC0.V (or %HSC0.VD) is set to 0
- (7) S is set to 0 and IN is set to 1, so the measurement will start at the next rising edge of the TB signal

Part III

Advanced Expert Output Functions

Overview

This part describes the advanced expert output functions.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	Pulse (%PLS)	55
5	Pulse Width Modulation (%PWM)	63
6	Drive (%DRV)	71
7	Pulse Train Output (%PTO)	101
8	Frequency Generator (%FREQGEN)	211

Chapter 4

Pulse (%PLS)

Using Pulse Function Blocks

This chapter provides descriptions and programming guidelines for using `Pulse` function blocks.


What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	56
Function Block Configuration	58
Programming Example	62

Description

Introduction

The `Pulse` function block  is used to generate square wave signals.

Two `Pulse` function blocks are available on the dedicated output channel `%Q0.0` or `%Q0.1`. Logic controllers with relay outputs for these two channels do not support the `Pulse` function block. Refer to the M221 Logic Controller - Hardware Guide for more information on inputs and outputs.

The `Pulse` function block allows only a single signal width, or duty cycle, of 50%.

You can choose to limit either the number of pulses or the period when the pulse train is executed. These factors can be determined at the time of configuration and/or updated by the program.

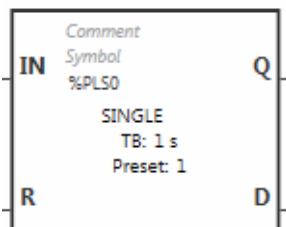
You must configure the `Pulse` function block in the **Configuration → Pulse Generators** before using an instance of the function block, refer to *Configuring Pulse Generators (see Modicon M221, Logic Controller, Programming Guide)*.

The PLS function has the following characteristics:

Characteristic	Value
Number of channels	2
Minimum frequency	1 Hz
Maximum frequency	10000 Hz
Accuracy on frequency	1 %

Illustration

This illustration is a `Pulse` function block:



Inputs

The `Pulse` function block has the following inputs:

Label	Object	Description	Value
IN	%PLSi.IN	Enable	At state 1, the pulse is produced at the dedicated output channel. At state 0, the output channel is set to 0.
R	%PLSi.R	Reset to 0 (optional)	At state 1, outputs %PLSi.Q and %PLSi.D are set to 0. The number of pulses generated in period T is set to 0.

Outputs

The `Pulse` function block has the following outputs:

Label	Object	Description	Value
Q	%PLSi.Q	Generation in progress	At state 1, indicates that the <code>Pulse</code> signal is generated at the dedicated output channel configured.
D	%PLSi.D	Generation complete (optional)	At state 1, signal generation is complete. The number of desired pulses has been reached.

Function Block Configuration

Overview

To configure the `Pulse Generator` resource, refer to *Configuring Pulse Generators (see Modicon M221, Logic Controller, Programming Guide)*.

To configure the `Pulse Generator` resource as a PLS, refer to *Configuring Pulse (see Modicon M221, Logic Controller, Programming Guide)*.

Parameters

The `Pulse` function block has the following parameters:

Parameter	Description	Value
Used	Address used	If selected, this address is currently in use in a program.
Address	%PLSi Pulse address	The instance identifier, where <i>i</i> is from 0 to the number of objects available on this logic controller. For the maximum number of <code>Pulse</code> objects, refer to the table <i>Maximum Number of Objects (see Modicon M221, Logic Controller, Programming Guide)</i> .
Symbol	Symbol	The symbol associated with this object. For details, refer to <i>Defining and Using Symbols (see SoMachine Basic, Operating Guide)</i> .
Preset	Preselection of the period (%PLSi.P)	<ul style="list-style-type: none"> • Time Base = 1 s, %PLSi.P=1 or 2 • Time Base = 10 ms, 1<=%PLSi.P<=200 • Time Base = 1 ms, 1<=%PLSi.P<=2000 • Time Base = 0.1 ms, 1<=%PLSi.P<=20000
Num. Pulse	Number of pulses (%PLSi.N, %PLSi.ND)	To produce an unlimited number of pulses, set %PLS.N or %PLS.ND to 0.
Current	Current output (%PLSi.Q)	0 or 1.
Done	Done pulse (%PLSi.D)	At state 1, signal generation is complete. The number of desired pulses has been reached. It is reset by either setting the IN or the R inputs to 1.
Duty Cycle	%PLSi.R	<p>This value gives the percentage of the signal in state 1 in a period. The width T_p is thus equal to:</p> $T_p = T \times (\%PLSi.R:100)$ <p>The user application writes the value for %PLSi.R. It is this word which controls the duty cycle of the period.</p> <p>The default value is 0 and values greater than 100 are considered to be equal to 100.</p>
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Objects

The Pulse function block is associated with the following objects:

Object	Description	Size (bit)	Default Value	Range	
%PLSi.P	Preset value	16	Preset (set on Configuration → Pulse Generators)	Preset %PLSi.P	Time Base
				1...20000	0.1 ms
				1...2000	1 ms
				1...200	10 ms
			1 or 2	1 s (default)	
%PLSi.N	Number of pulses	16	0	0...32767	
%PLSi.ND		32	0	0...2147483647	

Rules of Use

The output signal period T is set with **Preset** and the **Time Base** parameters such as

$$T = \%PLSi.P \times \text{Time Base.}$$

This table shows the range of available periods:

Time Base	Frequency
0.1 ms	0.5 Hz...10000 Hz
1 ms	0.5 Hz...1000 Hz
10 ms	0.5 Hz...100 Hz
1 s	0.5 Hz...1 Hz

The **Time Base** is set on the **Configuration → Pulse Generators** and cannot be modified. For more details, refer to *Configuring Pulse Generators (see Modicon M221, Logic Controller, Programming Guide)*.

If %PLSi.P is:

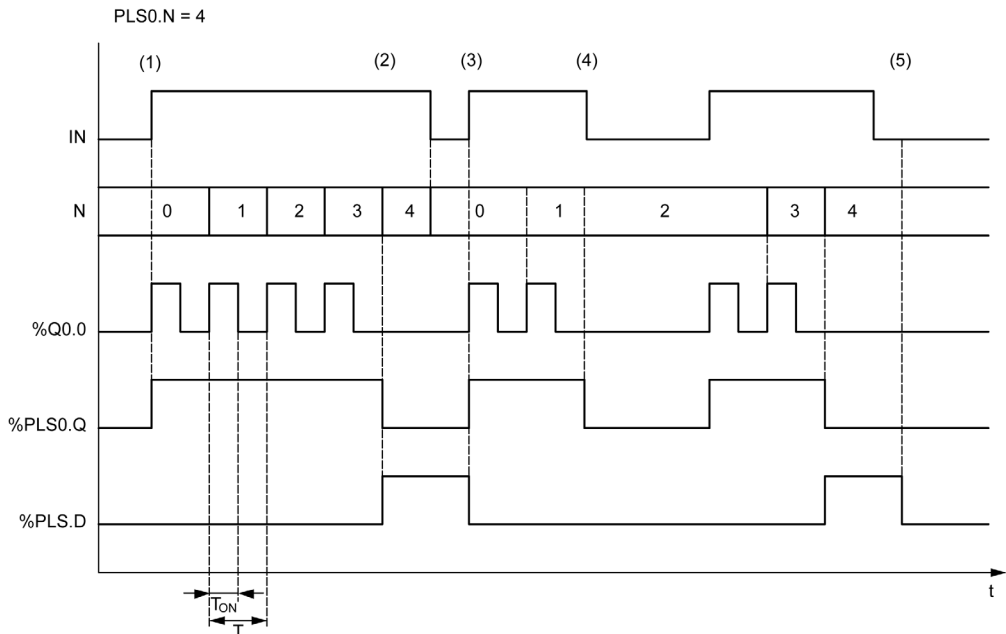
- changed, the output signal period is changed at the end of the current period.
- set to 0, the pulse generation function is stopped.
- out of range, the parameter is forced to 0 and the pulse generation function is stopped.

If %PLSi.N (or %PLSi.ND in **Double Word** mode) is:

- changed, the number of pulse to be generated is used at the next execution of the pulse generation function (%PLSi.D = 1 or after %PLSi.R = 1).
- set to 0, unlimited number of pulse are generated.
- out of range, the parameter is forced to 0.

Timing Diagram

This diagram displays the timing for Pulse function block:



- (1) IN input is set to 1, the pulse signal is generated at the dedicated output (%Q0.0) so %PLSi.Q is set to 1
- (2) The number of pulses reaches %PLS0.N (=4) so the Done flag output (%PLS0.D) is set to 1 and the pulse generation is stopped (%PLS0.Q = 0)
- (3) IN input is set to 1 so %PLS0.D is reset to 0
- (4) IN input is set to 0 so the output channel is set to 0 and %PLS0.Q = 0 indicates that the signal generation is not active
- (5) %PLS0.D is set to 0 by setting R input to 1

Special Cases

Special Case	Description
Effect of cold restart (%S0=TRUE)	<ul style="list-style-type: none"> ● Pulse generation is stopped. ● During the controller initialization, output is reset to 0. ● If after the controller initialization: <ul style="list-style-type: none"> ○ the controller enters the <code>STOPPED</code> state, the configured fallback strategy is applied to the output. ○ the controller enters the <code>RUNNING</code> state, the configuration parameters are restored.
Effect of warm restart (%S1=TRUE)	<ul style="list-style-type: none"> ● Pulse generation is stopped. ● During the controller initialization, output is reset to 0. ● If after the controller initialization: <ul style="list-style-type: none"> ○ the controller enters the <code>STOPPED</code> state, the configured fallback strategy is applied to the output. ○ the controller enters the <code>RUNNING</code> state, the configuration parameters are restored; however, the number of pulses that may have already been sent is reset to 0.⁽¹⁾
Effect at controller stop	<ul style="list-style-type: none"> ● Pulse generation is stopped. ● The fallback behavior depends on the configured fallback strategy: <ul style="list-style-type: none"> ○ Maintain value: the outputs are reset to 0. ○ Fallback value: the outputs are set to the fallback configured values (<i>see SoMachine Basic, Operating Guide</i>).
Effect of online modification	None
Effect of a short-circuit or over-current on an output addressed by the <code>Pulse</code> function block while generating a limited number of pulses	<ul style="list-style-type: none"> ● Pulse generation is stopped. ● Once the short-circuit or over-current has been corrected, pulse generation resumes the sequence from where it was stopped.
<p>(1) If there is an ongoing pulse output instruction affective at the time of the warm restart, the pulse generation, upon controller restart, will not take into account the number of pulses sent prior to the warm restart.</p>	

WARNING

UNINTENDED EQUIPMENT OPERATION

- Avoid issuing a warm restart command (%S1=TRUE) while an ongoing PLS command is active.
- If a warm restart is unavoidable, you must take into account any pulses that were sent prior to the warm restart.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Programming Example

Introduction

The `Pulse` function block can be configured as in this programming example.

Programming

This example is a `Pulse` function block:

Rung	Instruction
0	BLK %PLS0 LD %M1 IN LD %M0 R OUT_BLK LD Q ST %Q0.5 LD D ST %M10 END_BLK

NOTE: Refer to the reversibility procedure (*see SoMachine Basic, Generic Functions Library Guide*) to obtain the equivalent Ladder Diagram.

Chapter 5

Pulse Width Modulation (%PWM)

Using Pulse Width Modulation Function Blocks

This chapter provides descriptions and programming guidelines for using `Pulse Width Modulation` function blocks.


What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	64
Function Block Configuration	65
Programming Example	69

Description

Introduction

The Pulse Width Modulation function block  generates a variable wave signal on a dedicated output channel, %Q0.0 or %Q0.1, with variable width and, therefore, duty cycle.

Controllers with relay outputs for these two channels do not support this function.

%PWM0 uses dedicated output %Q0.0 and %PWM1 uses dedicated output %Q0.1. The Pulse function blocks %PLS can also be configured to use these same dedicated outputs. You can configure one or the other of these two functions, but not both, for any given dedicated output.

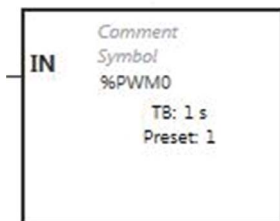
You must configure the Pulse Width Modulation function block in the **Configuration → Pulse Generators** before using an instance of the function block. Refer to *Configuring Pulse Generators (see Modicon M221, Logic Controller, Programming Guide)*.

The PWM function has the following characteristics:

Characteristic	Value
Number of channels	2
Minimum frequency	1 Hz
Maximum frequency	10000 Hz
Accuracy on frequency	1 %

Illustration

This illustration presents the Pulse Width Modulation function block:



Inputs

The Pulse Width Modulation function block has the following input:

Label	Object	Description	Value
IN	%PWMi.IN	Enable	At state 1, the Pulse Width Modulation signal is generated at the output channel. At state 0, the output channel is set to 0.

Function Block Configuration

Overview

To configure the `Pulse Generator` resource, refer to *Configuring Pulse Generators (see Modicon M221, Logic Controller, Programming Guide)*.

To configure the `Pulse Generator` resource as a PWM, refer to *Configuring Pulse Width Modulation (see Modicon M221, Logic Controller, Programming Guide)*.

Properties

The `Pulse Width Modulation` function block has the following properties:

Property	Value	Description
Used	Activated / deactivated checkbox	Indicates whether the address is in use.
Address	%PWWi where i is 0 or 1	i is the instance identifier. For the maximum number of PWM objects, refer to the table <i>Maximum Number of Objects (see Modicon M221, Logic Controller, Programming Guide)</i> .
Symbol	User-defined text	The symbol associated with this object. For details, refer to <i>Defining and Using Symbols (see SoMachine Basic, Operating Guide)</i> .
Preset	<ul style="list-style-type: none"> ● %PWWi.P=1 if Time Base=1 s ● 1<=%PWWi.P<=100 if Time Base=10 ms ● 1<=%PWWi.P<=1000 if Time Base=1 ms ● 1<=%PWWi.P<=10000 if Time Base=0.1 ms 	Preselection of the period
Duty cycle	From 0 to 100 NOTE: Values greater than 100 are considered to be equal to 100.	The Duty cycle is controlled by the object %PWWi.R, and is the percentage of the signal in state 1 within the period. The width of state 1 (Tp) is thus equal to: $T_P = T \times (\%PWWi.R/100)$ The user application writes the value for %PWWi.R.
Comment	User-defined text	A comment to associate with this object.

NOTE: The **Num.Pulse**, **Current** and **Done** properties that appear in the **Pulse Generators properties** table under the **Programming** tab do not apply to the PWM function.

Objects

The Pulse Width Modulation function block is associated with the following objects:

Object	Description	Size (bit)	Default Value	Range	
%PwMi.P	Preset value	16	Preset (set on Configuration → Pulse Generators)	Preset %PwMi.P	Time Base
				1...10000	0.1 ms
				1...1000	1 ms
				1...100	10 ms
				1	1 s (default)
%PwMi.R	Duty cycle (Ratio)	16	0	0...100	

If %PwMi.P is:

- modified, the output signal period is affected at the end of the ongoing period.
- set to 0, the pulse generation function is stopped.
- out of range, the parameter is forced to 0 and the pulse generation function is stopped.

If %PwMi.R is:

- set to 0, the pulse generation function is stopped (output set to 0).
- set to 100, the output signal is set to 1
- changed, the output signal ratio is changed at the end of the current period.
- out of range, the parameter is forced to 0.

Time Base

The **Time Base** is set in the menu **Configuration → Pulse Generators** and can only be modified under the **Configuration** tab. For more details, refer to *Configuring Pulse Generators (see Modicon M221, Logic Controller, Programming Guide)*.

The output signal period T is set with **Preset** and the **Time Base** parameters such that

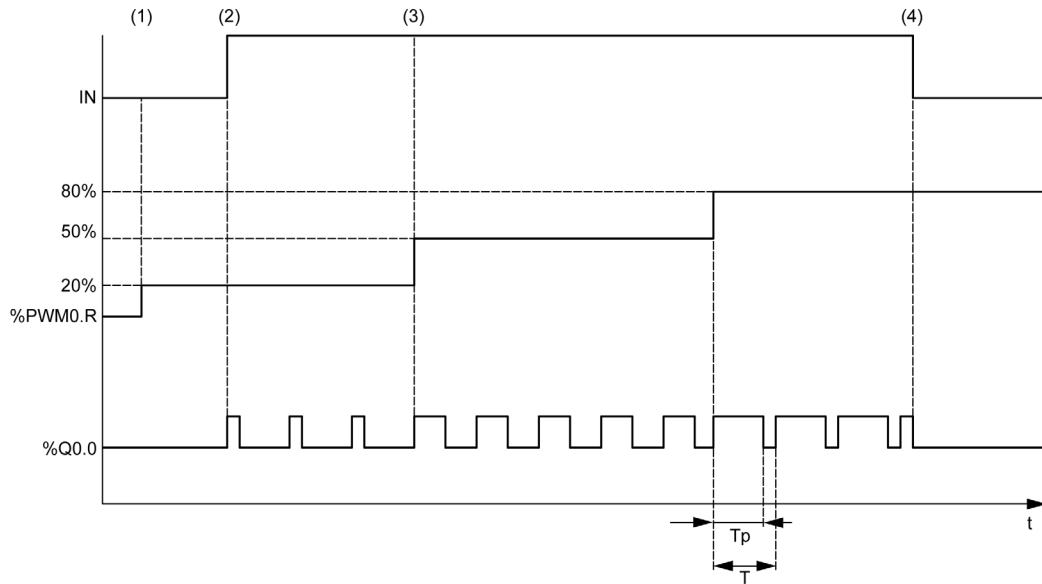
$$T = \%PwMi.P \times \text{Time Base.}$$

This table presents the range of available periods:

Time Base	Frequency Range
0.1 ms	1 Hz...10000 Hz
1 ms	1 Hz...1000 Hz
10 ms	1 Hz...100 Hz
1 s	1 Hz...1 Hz

Timing Diagram

This diagram presents the timing for the Pulse Width Modulation function block:



- (1) The PWM ratio (%PWMi.R) is set to 20%, IN = 0 so the pulse generation is not active
- (2) IN is set to 1 so PWM output is activated
- (3) The programmable width (T_p) changes with %PWM.R
- (4) IN is set to 0 so the PWM function is inhibited

Special Cases

Special Case	Description
Effect of cold restart (%S0=TRUE)	<ul style="list-style-type: none"> ● Pulse generation is stopped. ● During the controller initialization, output is reset to 0. ● If after the controller initialization: <ul style="list-style-type: none"> ○ the controller enters the STOPPED state, the configured fallback strategy is applied to the output. ○ the controller enters the RUNNING state, the configuration parameters are restored.
Effect of warm restart (%S1=TRUE)	<ul style="list-style-type: none"> ● Pulse generation is stopped. ● During the controller initialization, output is reset to 0. ● If after controller initialization, it enters the STOPPED state, the configured fallback strategy is applied to the output.

Special Case	Description
Effect at controller stop	<ul style="list-style-type: none">● Pulse generation is stopped.● The fallback behavior depends on the configured fallback strategy:<ul style="list-style-type: none">○ Maintain value: the outputs are reset to 0.○ Fallback value: the outputs are set to the fallback configured values (<i>see SoMachine Basic, Operating Guide</i>).
Effect of online modification	None

Programming Example

Introduction

The Pulse Width Modulation function block can be configured as in this programming example.

Programming Example

In this example:

- The signal width is modified by the program according to the state of controller input %I0.0 and %I0.1.
- The time base is set to 10 ms.
- The preset value %PWM0.P is set to 50 so the ratio step is equal to 2%.
- The configurable period T is equal to 500 ms.

The result is:

- If %I0.0 and %I0.1 are set to 0, the %PWM0.R ratio is set at 20%, the duration of the signal at state 1 is then: $20\% \times 500 \text{ ms} = 100 \text{ ms}$.
- If %I0.0 is set to 1 and %I0.1 is set to 0, the %PWM0.R ratio is set at 50% (duration 250 ms).
- If %I0.0 and %I0.1 are set to 1, the %PWM0.R ratio is set at 80% (duration 400 ms).

Examples of Pulse Width Modulation instructions:

Rung	Instruction
0	LDN %I0.0 ANDN %I0.1 [%PWM0.R:=20]
1	LD %I0.0 ANDN %I0.1 [%PWM0.R:=50]
2	LD %I0.0 AND %I0.1 [%PWM0.R:=80]
3	BLK %PWM0 LD %I0.2 IN END_BLK

NOTE: Refer to the reversibility procedure (*see SoMachine Basic, Generic Functions Library Guide*) to obtain the equivalent Ladder Diagram.

Chapter 6

Drive (%DRV)

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	72
Drive and Logic Controller States	74
Adding a Drive Function Block	76
Function Block Configuration	78
MC_Power_ATV: Enable/Disable Power Stage	79
MC_Jog_ATV: Start Jog Mode	81
MC_MoveVel_ATV: Move at Specified Velocity	84
MC_Stop_ATV: Stop Movement	87
MC_ReadStatus_ATV: Read Device Status	89
MC_ReadMotionState_ATV: Read Motion State	92
MC_Reset_ATV: Acknowledge and Reset Error	95
Error Codes	97

Description

Presentation

Drive function blocks **DRV** allow drive devices such as Altivar Speed Drives to be controlled by an M221 Logic Controller. For example:

- Control the speed of a motor managed by an ATV drive and update it continuously
- Monitor the status of the ATV drive and motor
- Manage errors detected in the ATV drive.

Communications take place over one of the serial lines of the logic controller, configured as a Modbus Serial I/O Scanner (*see Modicon M221, Logic Controller, Programming Guide*) using the Modbus RTU protocol.

In SoMachine Basic, first add targeted ATV drive types to the Modbus Serial I/O Scanner. This sets up predefined channels and initialization requests allowing data to be read from and written to specific registers on the ATV drive, including for example:

- **ETA** Status Word
- **ETI** Extended Status Word
- **RFRD** Output Velocity (RPM)
- **DP0** Error Code on Last Error
- **CMD** Control Word

Data transfer is carried out using Modbus request type **FC23 - Read/Write Multiple Registers**. This allows the program, for example, to read from the **ETA**, **ETI**, and **DP0** registers and write to the **CMD** register with a single Modbus request.

The following single-axis Drive function blocks are available in the **Programming** tab of SoMachine Basic:

Function Block	Description
MC_Power_ATV (<i>see page 79</i>)	Enables or disables the power stage of a device.
MC_Jog_ATV (<i>see page 81</i>)	Starts the Jog operating mode on a device.
MC_MoveVel_ATV (<i>see page 84</i>)	Specifies a target velocity for a device.
MC_Stop_ATV (<i>see page 87</i>)	Stops the current movement on a device.
MC_ReadStatus_ATV (<i>see page 89</i>)	Returns status information about a device.
MC_ReadMotionState_ATV (<i>see page 92</i>)	Returns status information on the current movement of a device.
MC_Reset_ATV (<i>see page 95</i>)	Reset device error regarding the drive state (<i>see page 74</i>) and acknowledge MC_Power_ATV (<i>see page 79</i>) errors.

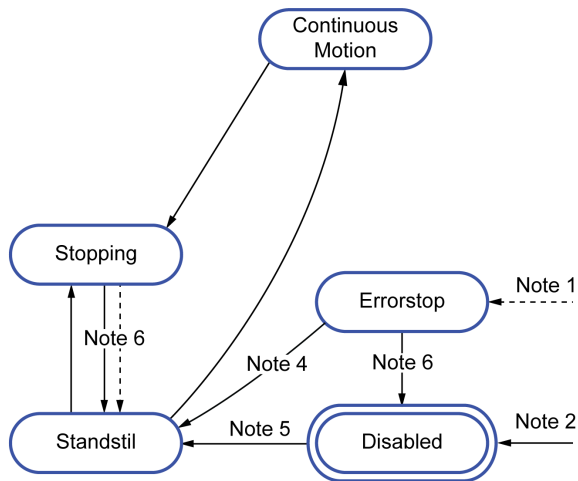
A maximum of 16 instances of each Drive function block can be used in a program at any one time.

When a device is added to the Modbus Serial IOScanner, SoMachine Basic allocates an axis for the device using a `%DRVn` object, where *n* is the number of the ATV drive. Each time you add a Drive function block to your program, you must associate it with an axis, creating a link between the function block, the axis, and the target device defined in the Modbus Serial IOScanner.

Drive and Logic Controller States

Drive State Diagram

The drive is always in one of the states defined in the diagram below. When a Drive function block is executed or an error occurs, this may cause a state transition:



Note 1 From any state if an error occurs.

Note 2 From any state (if no ErrorAxis) when %MC_Power_ATV.status is 0.

Note 3 Transition from ERRORSTOP to DISABLED state only if %MC_Reset_ATV.Done = 1 and %MC_Power_ATV.status = 0.

Note 4 Transition from ERRORSTOP to STANDSTILL state only if %MC_Reset_ATV.Done = 1 and %MC_Power_ATV.Enable = 1 and %MC_Power_ATV.Status = 1.

Note 5 Transition from DISABLED to STANDSTILL state only if %MC_Power_ATV.Enable = 1 and %MC_Power_ATV.Status = 1.

Note 6 Transition from STOPPING to STANDSTILL state only if %MC_Stop_ATV.Done = 1 and %MC_Stop_ATV.Execute = 0.

This table describes the drive states:

State	Description
DISABLED	Initial state. The drive is not in an operational status or in an error status.
STANDSTILL	The drive is in an operational status (ETA = 16#xx37) and Velocity = 0 (RFRD = 0).
ERRORSTOP	The drive is in an error status (ETA = 16#xxx8)
CONTINUOUS MOTION	The drive is in an operational status (ETA = 16#xx37) and Velocity ≠ 0 (RFRD ≠ 0).
STOPPING	MC_Stop_ATV function block is executing.

The function block MC_ReadStatus_ATV (*see page 89*) can be used to read the status of the ATV drive.

Logic Controller State Transitions

The following table describes how the Drive function blocks are affected by changes in the logic controller state:

Logic Controller State	Impact on Drive Function Blocks
RUNNING	Drive function blocks are executed normally according to the user logic.
STOPPED	The configured drive axes are stopped when the controller goes into the STOPPED state. If the Fallback Behavior option is set to Fallback values , the command 0x00 is sent to the ATV drive, which leads to a Switch on Disabled (NST) status. Otherwise, if Fallback Behavior is set to Maintain values , no action is taken (the command is not changed).
HALTED	The configured drive axes are stopped when the controller goes into the HALTED state. If the Fallback Behavior option is set to Fallback values , the command 0x00 is sent to the ATV drive, which leads to a Switch on Disabled (NST) status. Otherwise, if Fallback Behavior is set to Maintain values , no action is taken (the command is not changed).
POWERLESS, EMPTY	Drive function blocks are not executed (the Modbus Serial IOScanner is stopped). This is also the case when the application in the controller is updated.

NOTE: In the case of the controller state of HALTED or STOPPED, and you have selected to **Maintain values**, the drive is not given any further commands by the controller. Therefore, the drive must determine the appropriate state to assume. If you chose to **Maintain values** for the drive, you must include this in your hazard and risk analysis for any consequential and possibly hazardous events.

WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that a risk assessment is conducted and respected according to EN/ISO 12100 during the design of your machine.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Adding a Drive Function Block

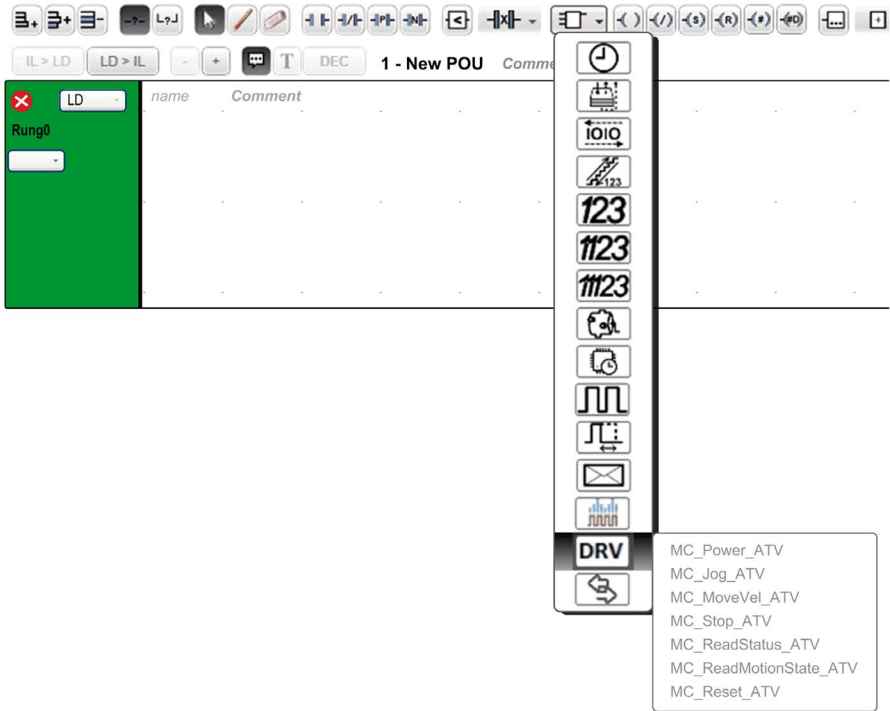
Prerequisites

Prerequisites to add a Drive function block:

- The serial line of the M221 Logic Controller must be configured as a Modbus Serial IOScanner.
- The ATV drives to be controlled must be added and configured (*see Modicon M221, Logic Controller, Programming Guide*) on the Modbus Serial IOScanner.

Adding a Drive Function Block

Follow these steps to add an instance of a Drive function block:

Step	Action
1	Select the Programming tab.
2	Select Function Blocks → Drive as shown in the following graphic: 
3	Click into the rung to place the selected function block.
4	Associate the inputs/outputs of the function block.

Removing a Function Block

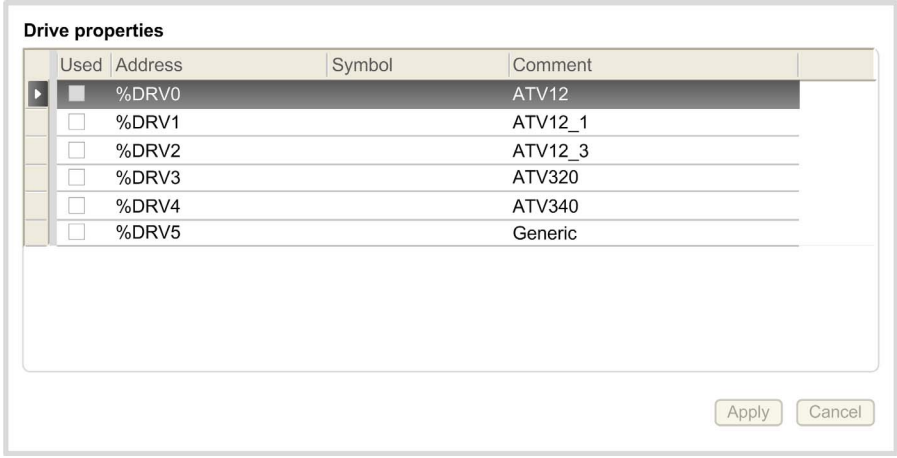
Follow these steps to remove an instance of a Drive function block:

Step	Action
1	In the Programming tab, click the instance of the function block.
2	Press Delete to remove the selected function block.

Function Block Configuration

Configuring Drive Objects

Each Drive function block is associated with a Drive (%DRV) object. To display a list of configured Drive objects:

Step	Action
1	<p>Select the Programming → Tools tab and click Drive objects → Drive to display Drive object properties.</p> 
2	Update the properties as required and click Apply

Drive function blocks have the following properties:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the Drive object is in use in the program.
Address	No	%DRVn	%DRVn	The address of the Drive object, where n is the object number.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the Drive object. Double-click the cell to define or edit a symbol.
Comment	Yes	–	–	Allows you to specify a comment to associate with the Drive object. Double-click the cell to define or edit a comment.

MC_Power_ATV: Enable/Disable Power Stage

Description

This function block enables or disables the drive power stage.

A rising edge of the input `Enable` enables the power stage. When the power stage is enabled, the output `Status` is set to 1.

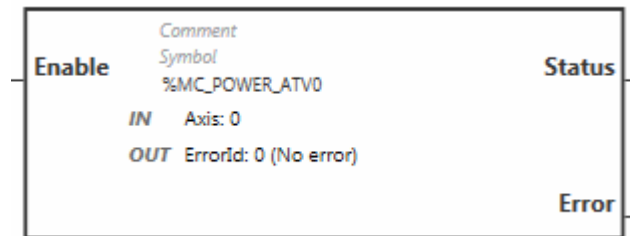
A falling edge of the input `Enable` disables the power stage (Shutdown command without `Error`). When the power stage is disabled, the output `Status` is reset to 0.

If the internal status register ETA of the ATV drive has not reached an operational status before the expiration of the timeout value, a `Timeout Error` is generated. The timeout is calculated as the channel cycle time multiplied by 4, or 200 ms, whichever is greater. A minimum of 200 ms is required to allow for drive reaction time.

If errors are detected during execution of the function block, the output `Error` is set to 1. This leads to a Shutdown command (CMD = 16#0006) to disable the ATV drive (Ready to switch on status, ETA = 16#xx21).

If an error occurs, only a successful execution of `MC_Reset_ATV` (*see page 95*) function block can restore the power stage.

Graphical Representation



Inputs

This table describes the inputs of the function block:

Label	Object	Initial value	Description
Enable	-	0	Set to 1 to start execution of the function block and enable the power stage. Set to 0 to stop execution of the function block and disable the power stage.
Axis	<code>%MC_POWER_ATVi.AXIS</code> where <code>i</code> is 0...15	-	Identifier of the axis (<code>%DRV0...%DRV15</code>) for which the function block is to be executed.

Outputs

This table describes the outputs of the function block:

Label	Object	Initial value	Value
Status	%MC_POWER_ATVi.STATUS	0	Default value: 0 <ul style="list-style-type: none"> ● 0: Power stage is disabled. ● 1: Power stage is enabled. Set to 1 when the ATV drive reaches an operational status (ETA = 16#xx37)
Error	%MC_POWER_ATVi.ERROR	0	Set to 0 when no error is detected. Set to 1 if an error occurs during execution. Function block execution is finished. The ErrorId output object indicates the cause of the error.
ErrorId	%MC_POWER_ATVi.ERRORID	0 (No error)	Error code returned by the function block when the Error output is set to 1. For details on the errors, refer to Error Codes (<i>see page 97</i>). Range: 0...65535

Parameters

Double-click the function block to display the function block parameters.

The MC_Power_ATV function block has the following parameters:

Parameter	Value	Description
Used	Address used	If selected, this address is currently in use in a program.
Address	%MC_Power_ATVi	The instance identifier, where i is from 0 to the number of objects available on this logic controller. For the maximum number of Drive objects, refer to the table Maximum Number of Objects (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	Symbol	The symbol associated with this object. For details, refer to Defining and Using Symbols (<i>see SoMachine Basic, Operating Guide</i>).
Axis	%DRV <i>i</i> , where i is 0...15 None	Select the axis (Drive object instance) for which the function block is to be executed. The Drive object must have been previously configured on the Modbus Serial IOScanner (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Update the parameters as required and click **Apply**.

MC_Jog_ATV: Start Jog Mode

Description

This function block starts the Jog operating mode. A Jog operation commands a device to move forwards or backwards at a specified velocity.

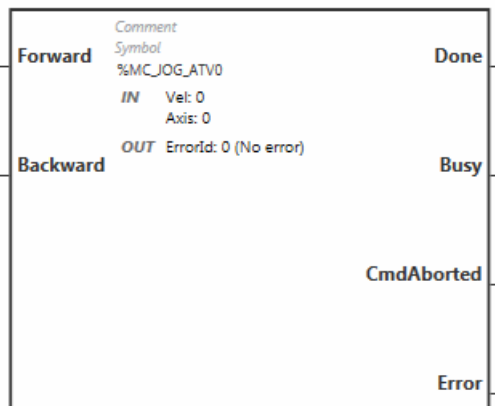
If either of the function blocks MC_MoveVel_ATV ([see page 84](#)) or MC_Stop_ATV ([see page 87](#)) is enabled while this function block is executing (`Busy` output set to 1), the MC_Jog_ATV function block commands the movement. The `Busy` output is reset to 0 and the `CmdAborted` output is set to 1.

When a Jog operation is in progress, a change of velocity value (`Vel`) is only applied on detection of a falling/rising edge of the `Forward` or `Backward` inputs.

If either of the `Error` or `CmdAborted` outputs is set to 1, the `Forward` and `Backward` inputs must first be reset to 0 and then a new rising edge applied to the `Forward` and/or `Backward` inputs to restart the movement.

Starting a Jog operation while the MC_Stop_ATV ([see page 87](#)) function block is executing causes a Stop Active Error. Starting a Jog operation when the drive is not in an operational status (ETA # 16#xx37) causes a Not Run Error.

Graphical Representation



Inputs

This table describes the inputs of the function block:

Input	Object	Initial Value	Description
Forward	-	0	Setting either the <code>Forward</code> input or the <code>Backward</code> input to 1 starts the jog movement. If the <code>Forward</code> and <code>Backward</code> inputs are both set to 1, the operating mode remains active, the jog movement is stopped, and the <code>Busy</code> output remains set to 1. If the <code>Forward</code> and <code>Backward</code> inputs are both set to 0, the operating mode is terminated and the <code>Done</code> output is set to 1 for one cycle.
Backward	-	0	
Vel	<code>%MC_JOG_ATVi.VEL</code>	0	Target velocity for the Jog operating mode. During jog movement, a change in the velocity value <code>Vel</code> is only applied upon detection of a falling/rising edge of the <code>Forward</code> or <code>Backward</code> input. Range: -32768...32767
Axis	<code>%MC_JOG_ATVi.AXIS</code> where <code>i</code> is 0...15	-	Identifier of the axis (<code>%DRV0...%DRV15</code>) for which the function block is to be executed. The axis must first be declared in the Configuration tab.

Outputs

This table describes the outputs of the function block:

Output	Output Object	Initial Value	Description
Done	<code>%MC_JOG_ATVi.DONE</code>	0	Set to 1 for one cycle when both the <code>Forward</code> and <code>Backward</code> inputs are set to 0. Set to 1 to indicate that the Jog operating mode is terminated.
Busy	<code>%MC_JOG_ATVi.BUSY</code>	0	Set to 1 when: <ul style="list-style-type: none"> • Jog is in progress (<code>Forward</code> = 1 or <code>Backward</code> = 1) • Both the <code>Forward</code> and <code>Backward</code> inputs are set to 1, indicating that the Jog operating mode remains active and the jog movement is stopped.
CmdAborted	<code>%MC_JOG_ATVi.CMDABORTED</code>	0	Set to 1 if function block execution terminates due to another command being executed.
Error	<code>%MC_JOG_ATVi.ERROR</code>	0	Set to 0 when no error is detected. Set to 1 if an error occurs during execution. Function block execution is finished. The <code>ErrorId</code> output object indicates the cause of the error.

Output	Output Object	Initial Value	Description
ErrorId	%MC_JOG_ATVi.ERRORID	0 (No error)	Error code returned by the function block when the <code>Error</code> output is set to 1. For details on the errors, refer to Error Codes (<i>see page 97</i>). Range: 0...65535

Parameters

Double-click the function block to display the function block parameters.

The `MC_Jog_ATV` function block has the following parameters:

Parameter	Value	Description
Used	Address used	If selected, this address is currently in use in a program.
Address	%MC_Jog_ATVi	The instance identifier, where <i>i</i> is from 0 to the number of objects available on this logic controller. For the maximum number of Drive objects, refer to the table Maximum Number of Objects (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	Symbol	The symbol associated with this object. For details, refer to Defining and Using Symbols (<i>see SoMachine Basic, Operating Guide</i>).
Axis	%DRV <i>i</i> , where <i>i</i> is 0...15 None	Select the axis (Drive object instance) for which the function block is to be executed. The Drive object must have been previously configured on the Modbus Serial IOScanner (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Vel	Target velocity	Enter the target velocity for the Jog operating mode and press Enter. Default value: 0 Range: -32768...32767
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Update the parameters as required and click **Apply**.

MC_MoveVel_ATV: Move at Specified Velocity

Description

This function block starts the Profile Velocity operating mode with a specified velocity. When the target velocity is reached, the `InVel` output is set to 1.

If the `MC_Jog_ATV` (see page 81) or `MC_Stop_ATV` (see page 87) function blocks are enabled while this function block is executing (`Busy` output set to 1), `MC_MoveVel_ATV` commands the movement. In this case, the `Busy` output is reset to 0 and the `CmdAborted` output is set to 1.

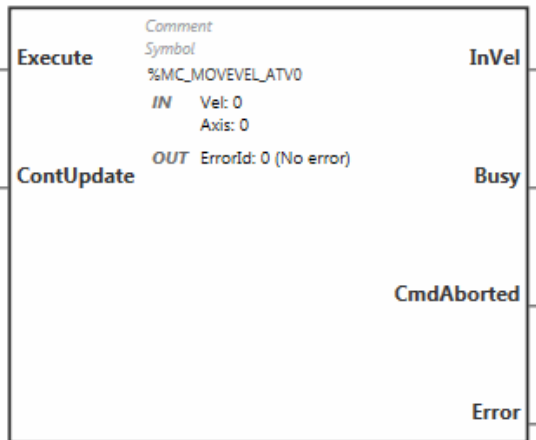
The `ContUpdate` and `Vel` input values are applied on a rising edge of the `Execute` input.

If either of the `Error` or `CmdAborted` outputs of `MC_MoveVel_ATV` is set to 1, a new rising edge of `Execute` is necessary to resume the movement.

Starting this function block while the `MC_Stop_ATV` (see page 87) function block is executing leads to a Stop Active Error.

Starting this function block when the drive is not in an operational status (ETA ≠ 16#xx37), leads to a Not Run Error.

Graphical Representation



Inputs

This table describes the inputs of the function block:

Input	Object	Initial Value	Description
Execute	-	0	Set to 1 to start execution of the function block.
ContUpdate	-	0	Set to 1 before executing the function block to enable continuous updating of the <code>Vel</code> parameter value.
Vel	<code>%MC_MOVEVEL_ATVi.VEL</code>	0	Target velocity for the operating mode. Range: -32 768...32 767. A negative value forces movement in the opposite direction.
Axis	<code>%MC_MOVEVEL_ATVi.AXIS</code> where <code>i</code> is 0...15	-	Identifier of the axis (<code>%DRV0...%DRV15</code>) for which the function block is to be executed. The axis must first be declared in the Configuration tab.

Outputs

This table describes the outputs of the function block:

Output	Object	Initial Value	Description
InVel	<code>%MC_MOVEVEL_ATVi.INVEL</code>	0	0 indicates that the target velocity (<code>Vel</code>) has not been reached. Set to 1 when the target velocity (<code>Vel</code>) is reached.
Busy	<code>%MC_MOVEVEL_ATVi.BUSY</code>	0	Set to 1 when the function block is executed. Remains at 1 even after the target velocity is reached. Reset to 0 when the function block is stopped or aborted.
CmdAborted	<code>%MC_MOVEVEL_ATVi.CMDABORTED</code>	0	Set to 1 if function block execution is terminated due to another command being executed.
Error	<code>%MC_MOVEVEL_ATVi.ERROR</code>	0	Set to 0 when no error is detected. Set to 1 if an error occurs during execution. Function block execution is finished. The <code>ErrorId</code> output object indicates the cause of the error.
ErrorId	<code>%MC_MOVEVEL_ATVi.ERRORID</code>	0 (No error)	Error code returned by the function block when the <code>Error</code> output is set to 1. For details on the errors, refer to Error Codes (see page 97). Range: 0...65535

NOTE: When the speed command of the ATV drive is low (< 10), the `InVel` and `ConstantVel` parameters may be invalid because the speed range of the ATV drive itself may be inaccurate.

Parameters

Double-click the function block to display the function block parameters.

The MC_Move1Vel_ATV function block has the following parameters:

Parameter	Value	Description
Used	Address used	If selected, this address is currently in use in a program.
Address	%MC_Move1Vel_ATVi	The instance identifier, where i is from 0 to the number of objects available on this logic controller. For the maximum number of Drive objects, refer to the table Maximum Number of Objects (see <i>Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	Symbol	The symbol associated with this object. For details, refer to Defining and Using Symbols (see <i>SoMachine Basic, Operating Guide</i>).
Axis	%DRV <i>i</i> , where i is 0...15 None	Select the axis (Drive object instance) for which the function block is to be executed. The Drive object must have been previously configured on the Modbus Serial IOScanner (see <i>Modicon M221, Logic Controller, Programming Guide</i>).
Vel	Target velocity	Enter the target velocity for the operating mode and press Enter. Default value: 0 Range: -32768...32767. A negative value forces movement in the opposite direction.
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Update the parameters as required and click **Apply**.

MC_Stop_ATV: Stop Movement

Description

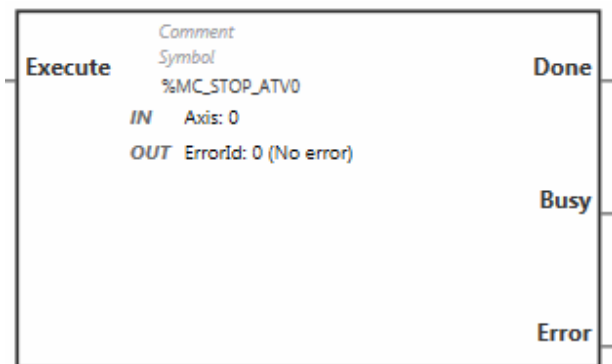
This function block stops the ongoing movement of the specified drive.

Drive-specific stop parameters, for example deceleration, are provided by the configuration of the drive.

Once started by a rising edge on the `Execute` input, any further activity on the `Execute` input is ignored until `Done` is set to TRUE. Executing another Drive function block while `MC_Stop_ATV` is busy does not abort the stop procedure—the function block `MC_Stop_ATV` remains busy and the other function block ends in an error.

The stop procedure can only be interrupted by disabling the power stage or if an error occurs (for example, ATV Not Run error or Modbus Serial IOScanner error).

Graphical Representation



Inputs

This table describes the inputs of the function block:

Input	Object	Initial Value	Description
Execute	-	0	Set to 1 to start execution of the function block. The execution of other motion function block is not possible when the <code>Busy</code> output is set to 1. In this case, the other function block returns an error.
Axis	%MC_STOP_ATV <i>i</i> . <code>AXIS</code> where <i>i</i> is 0...15	-	Identifier of the axis (%DRV0...%DRV15) for which the function block is to be executed.

Outputs

This table describes the outputs of the function block:

Output	Output Object	Initial Value	Description
Done	%MC_STOP_ATVi.DONE	0	Set to 1 to indicate the function block execution is complete.
Busy	%MC_STOP_ATVi.BUSY	0	Set to 1 when function block execution begins.
Error	%MC_STOP_ATVi.ERROR	0	Set to 0 when no error is detected. Set to 1 if an error occurs during execution. Function block execution is finished. The <code>ErrorId</code> output object indicates the cause of the error.
ErrorId	%MC_STOP_ATVi.ERRORID	0 (No error)	Error code returned by the function block when the <code>Error</code> output is set to 1. For details on the errors, refer to Error Codes (<i>see page 97</i>). Range: 0..65535

Parameters

Double-click the function block to display the function block parameters.

The `MC_Stop_ATV` function block has the following parameters:

Parameter	Value	Description
Used	Address used	If selected, this address is currently in use in a program.
Address	%MC_Stop_ATVi	The instance identifier, where <i>i</i> is from 0 to the number of objects available on this logic controller. For the maximum number of Drive objects, refer to the table Maximum Number of Objects (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	Symbol	The symbol associated with this object. For details, refer to Defining and Using Symbols (<i>see SoMachine Basic, Operating Guide</i>).
Axis	%DRV <i>i</i> , where <i>i</i> is 0...15 None	Select the axis (Drive object instance) for which the function block is to be executed. The Drive object must have been previously configured on the Modbus Serial IOScanner (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Update the parameters as required and click **Apply**.

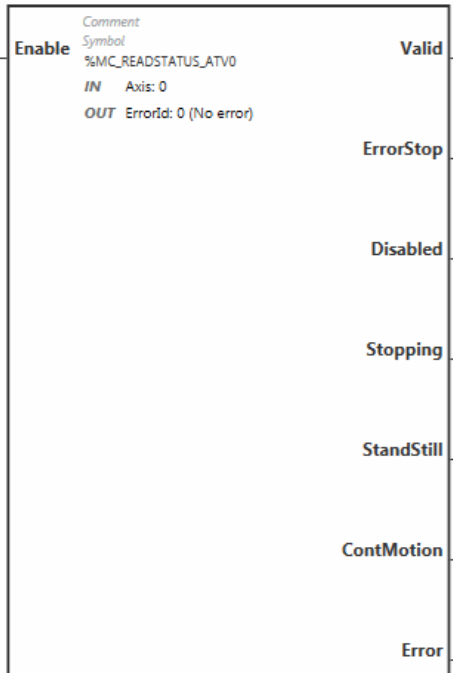
MC_ReadStatus_ATV: Read Device Status

Description

The function block reads the status of the ATV drive.

Refer to Drive State Diagram (*see page 74*) for details on the states.

Graphical Representation



Inputs

This table describes the inputs of the function block:

Label	Object	Initial value	Description
Enable	-	0	Set to 1 to enable the function block.
Axis	%MC_READSTATUS_ATV i.AXIS where i is 0...15	-	Identifier of the axis (%DRV0...%DRV15) for which the function block is to be executed.

Outputs

This table describes the outputs of the function block:

Label	Object	Initial value	Description
Valid	%MC_READSTATUS_ATVi.VALID	0	Set to 1 while the function block is running without errors.
ErrorStop	%MC_READSTATUS_ATVi.ERRORSTOP	0	Set to 1 if the ATV drive is in an error status (ETA = 16#xxx8).
Disabled	%MC_READSTATUS_ATVi.DISABLED	0	Set to 1 if the ATV drive is not in an operational status and not in an error status.
Stopping	%MC_READSTATUS_ATVi.STOPPING	0	Set to 1 if the MC_Stop_ATV function block is being executed, or the movement is being stopped.
Standstill	%MC_READSTATUS_ATVi.STANDSTILL	0	Set to 1 if the ATV drive is in an operational status and the velocity is 0 (ETA = 16#xx37 and RFRD = 0).
ContMotion	%MC_READSTATUS_ATVi.CONTMOTION	0	Set to 1 if the ATV drive is in an operational status and the velocity is not equal to 0 (ETA = 16#xx37 and RFRD ≠ 0).
Error	%MC_READSTATUS_ATVi.ERROR	0	Set to 0 when no error is detected. Set to 1 if an error occurs during execution. Function block execution is finished. The ErrorId output object indicates the cause of the error.
ErrorId	%MC_READSTATUS_ATVi.ERRORID	0 (No error)	Error code returned by the function block when the Error output is set to 1. For details on the errors, refer to Error Codes (<i>see page 97</i>). Range: 0...65535

Parameters

Double-click the function block to display the function block parameters.

The MC_ReadStatus_ATV function block has the following parameters:

Parameter	Value	Description
Used	Address used	If selected, this address is currently in use in a program.
Address	%MC_ReadStatus_ATVi	The instance identifier, where i is from 0 to the number of objects available on this logic controller. For the maximum number of Drive objects, refer to the table Maximum Number of Objects (see <i>Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	Symbol	The symbol associated with this object. For details, refer to Defining and Using Symbols (see <i>SoMachine Basic, Operating Guide</i>).
Axis	%DRV <i>i</i> , where i is 0...15 None	Select the axis (Drive object instance) for which the function block is to be executed. The Drive object must have been previously configured on the Modbus Serial IOScanner (see <i>Modicon M221, Logic Controller, Programming Guide</i>).
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

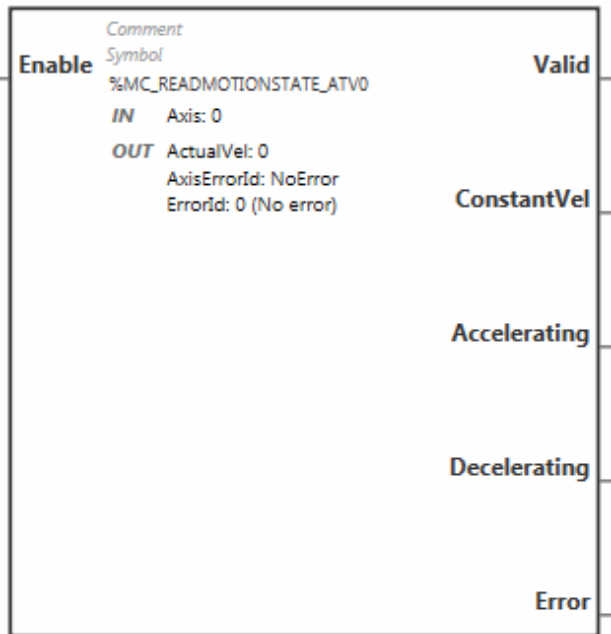
Update the parameters as required and click **Apply**.

MC_ReadMotionState_ATV: Read Motion State

Description

This function block outputs status information on the movement read from the ATV drive.

Graphical Representation



Inputs

This table describes the inputs of the function block:

Input	Object	Initial Value	Description
Enable	-	0	Set to 1 to start execution of the function block.
Axis	%MC_READMOTIONSTATE_ATVi.AXIS where i is 0...15	-	Identifier of the axis (%DRV0...%DRV15) for which the function block is to be executed.

Outputs

This table describes the outputs of the function block:

Output	Object	Initial Value	Description
Valid	%MC_READMOTIONSTATE_ATVi.VALID	0	Set to 1 while the function block is running without errors.
ConstantVel	%MC_READMOTIONSTATE_ATVi.CONSTANTVEL	0	Set to 1 when a movement at constant velocity is being performed (ETA register).
Accelerating	%MC_READMOTIONSTATE_ATVi.ACCELERATING	0	Set to 1 when the motor is accelerating (ETI register).
Decelerating	%MC_READMOTIONSTATE_ATVi.DECELERATING	0	Set to 1 when the motor is decelerating (ETI register).
Error	%MC_READMOTIONSTATE_ATVi.ERROR	0	Set to 0 when no error is detected. Set to 1 if an error occurs during execution. Function block execution is finished. The <code>ErrorId</code> output object indicates the cause of the error.
ActualVel	%MC_READMOTIONSTATE_ATVi.ACTUALVEL	0	Velocity returned by the ATV drive (RFRD register). Range: -32768...32767
AxisErrorId	%MC_READMOTIONSTATE_ATVi.AXISERRORID	0	Axis error identifier returned by the ATV drive (DP0 register). There is an axis error when the drive is in an error status. Set to 0 if the drive is not in an error status (ETA register ≠ 16#xxx8) For details on axis errors, refer to AxisErrorId Error Codes (see page 97) . Range: -32768...32767
ErrorId	%MC_READMOTIONSTATE_ATVi.ERRORID	No error (nOF)	Error code returned by the function block when the <code>Error</code> output is set to 1. For details on the errors, refer to Error Codes (see page 97) . Range: 0...65535

NOTE: When the speed command of the ATV drive is low (< 10), the `InVel` and `ConstantVel` parameters may be invalid because the speed range of the ATV drive itself may be inaccurate.

Parameters

Double-click the function block to display the function block parameters.

The MC_ReadMotionState_ATV function block has the following parameters:

Parameter	Value	Description
Used	Address used	If selected, this address is currently in use in a program.
Address	%MC_ReadMotionState_ATVi	The instance identifier, where i is from 0 to the number of objects available on this logic controller. For the maximum number of Drive objects, refer to the table Maximum Number of Objects (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	Symbol	The symbol associated with this object. For details, refer to Defining and Using Symbols (<i>see SoMachine Basic, Operating Guide</i>).
Axis	%DRV <i>i</i> , where i is 0...15 None	Select the axis (Drive object instance) for which the function block is to be executed. The Drive object must have been previously configured on the Modbus Serial IOScanner (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

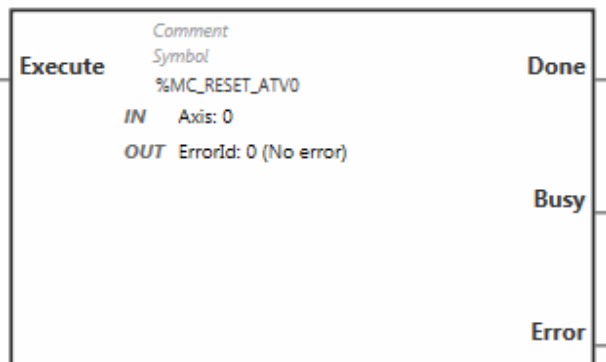
Update the parameters as required and click **Apply**.

MC_Reset_ATV: Acknowledge and Reset Error

Description

This function block is used to acknowledge an error and re-initialize the error condition on the drive. For more information, refer to Drive State Diagram (*see page 74*).

Graphical Representation



Inputs

This table describes the inputs of the function block:

Label	Object	Initial value	Description
Execute	-	0	Set to 1 to start execution of the function block.
Axis	%MC_RESET_ATVi.AXIS where i is 0...15	-	Identifier of the axis (%DRV0...%DRV15) for which the function block is to be executed.

Outputs

This table describes the outputs of the function block:

Output	Output Object	Initial Value	Description
Done	%MC_RESET_ATVi.DONE	0	Set to 1 when Reset has ended without error.
Busy	%MC_RESET_ATVi.BUSY	0	Set to 1 when the function block begins execution.

Output	Output Object	Initial Value	Description
Error	%MC_RESET_ATVi.ERROR	0	Set to 1 if the device remains in an error status after timeout expiration. The timeout is calculated as the channel cycle time multiplied by 4, or 200 ms, whichever is greater. A minimum of 200 ms is required to allow for drive reaction time. Refer to <i>Configuring Channels (see Modicon M221, Logic Controller, Programming Guide)</i> for information on the configuring the channel cycle time.
ErrorId	%MC_RESET_ATVi.ERRORID	0 (No error)	Error code returned by the function block when the Error output is set to 1. For details on the errors, refer to <i>Error Codes (see page 97)</i> . Range: 0...65535

Parameters

Double-click the function block to display the function block parameters.

The `MC_Reset_ATV` function block has the following parameters:

Parameter	Value	Description
Used	Address used	If selected, this address is currently in use in a program.
Address	%MC_Reset_ATVi	The instance identifier, where <i>i</i> is from 0 to the number of objects available on this logic controller. For the maximum number of Drive objects, refer to the table <i>Maximum Number of Objects (see Modicon M221, Logic Controller, Programming Guide)</i> .
Symbol	Symbol	The symbol associated with this object. For details, refer to <i>Defining and Using Symbols (see SoMachine Basic, Operating Guide)</i> .
Axis	%DRV <i>i</i> , where <i>i</i> is 0...15 None	Select the axis (Drive object instance) for which the function block is to be executed. The Drive object must have been previously configured on the Modbus Serial I/O Scanner (<i>see Modicon M221, Logic Controller, Programming Guide</i>).
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Update the parameters as required and click **Apply**.

Error Codes

ErrorId Error Codes

This table lists the possible function block error codes:

Value	Name	Description
0	No error	No error detected.
1	IOScanner error	Error detected on the serial line, device, or channel.
2	ATV is in an error status	The ATV drive is in an error status (ETA = 16#xxx8).
3	Timeout error	Timeout expired before the MC_Power_ATV function block has received the correct status from the drive.
4	Invalid ATV status	The ATV drive has an invalid ETA value.
5	Reset error	The MC_Reset_ATV function block is requested while the ATV drive is in an error status.
6	Stop Active error	The MC_Jog_ATV or MV_MoveVelocity_ATV function block is requested while MC_Stop is active.
7	ATV Not Run error	The MC_Jog_ATV or MV_MoveVelocity_ATV function block is requested while the ATV drive is not operational.
8	Invalid AxisRef error	AxisRef input %DRV of the function block is invalid (not present in the Modbus Serial IOScanner configuration (see <i>Modicon M221, Logic Controller, Programming Guide</i>)).
9	Internal error	A firmware error occurred.

AxisErrorId Error Codes

This table lists the possible function block axis error codes returned by the MC_ReadMotion-Status function block:

Value	Name
0	No error (nOF)
2	EEPROM control (EEF1)
3	Incorrect configuration (CFF)
4	Invalid Configuration (CFI)
5	Modbus Comm Interruption (SLF1)
6	Internal Link Error (ILF)
7	Fieldbus Com Interrupt (CnF)
8	External Error (EPF1)
9	Overcurrent (OCF)

Value	Name
10	Precharge Capacitor (CrF)
13	AI2 4-20 mA loss (LFF2)
15	Input Overheating (IHF)
16	Drive Overheating (OHF)
17	Motor Overload (OLF)
18	DC Bus Overvoltage (ObF)
19	Supply Mains Overervoltage (OSF)
20	Single Output Phase Loss (OPF1)
21	Input phase loss (PHF)
22	Supply Mains Undervoltage (USF)
23	Motor Short Circuit (SCF1)
24	Motor Overspeed (SOF)
25	Autotuning Error
26	Internal Error 1 (InF1)
27	Internal Error 2 (InF2)
28	Internal Error 3 (InF3)
29	Internal Error 4 (InF4)
30	EEPROM ROM Power (EEF2)
32	Ground Short Circuit (SCF3)
33	Output Phase Loss (OPF2)
37	Internal Error (InF7)
38	Fieldbus Error (EPF2)
40	Internal Error 8 (InF8)
42	PC Com Interruption (SLF2)
45	HMI Com Interruption (SLF3)
51	Internal Error 9 (InF9)
52	Internal Error 10 (InFA)
53	Internal Error 11 (InFb)
54	IGBT Overheating (tJF)
55	IGBT Short Circuit (SCF4)
56	Motor Short Circuit (SCF5)
60	Internal Error 12 (InFC)
64	Input Contactor (LCF)
68	Internal Error 6 (InF6)

Value	Name
69	Internal Error 14 (InFE)
71	AI3 4-20mA Loss (LFF3)
72	AI4 4-20mA Loss (LFF4)
73	Boards Compatibility (HCF)
77	Conf Transfer Error (CFI2)
79	AI5 4-20mA Loss (LFF5)
99	Channel Switch Error (CSF)
100	Process Underload (ULF)
101	Process Overload (OLC)
105	Angle Error (ASF)
106	AI1 4-20mA Loss (LFF1)
107	Safety Function Error (SAFF)
110	AI2 Th Detected Error (tH2F)
111	AI2 Thermal Sensor Error (t2CF)
112	AI3 Th Detected Error (tH3F)
113	AI3 Thermal Sensor Error (t3CF)
114	Pump Cycle Start Error (PCPF)
119	Pump Low Flow Error (PLFF)
120	AI4 Th Detected Error (tH4F)
121	AI4 Thermal Sensor Error (t4CF)
122	AI5 Th Detected Error (tH5F)
123	AI5 Thermal Sensor Error (t5CF)
126	Dry Run Error (drYF)
127	PID Feedback Error (PFMF)
128	Program Loading Error (PGLF)
129	Program Running Error (PGrF)
130	Lead Pump Error (MPLF)
131	Low Level Error (LCLF)
132	High Level Error (LCHF)
142	Internal Error 16 (InFG)
143	Internal Error 17 (InFH)
144	Internal Error 0 (InF0)
146	Internal Error 13 (InFd)
149	Internal Error 21 (InFL)

Value	Name
151	Internal Error 15 (InFF)
152	Firmware Update Error (FEr)
153	Internal Error 22 (InFM)
154	Internal Error 25 (InFP)
155	Internal Error 20 (InF)
157	Internal Error 27 (InFr)

Chapter 7

Pulse Train Output (%PTO)

Using Pulse Train Output Function Blocks

This chapter provides descriptions and programming guidelines for using Pulse Train Output function blocks.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	Description	102
7.2	Configuration	119
7.3	Programming	129
7.4	Home Modes	134
7.5	Data Parameters	146
7.6	Operation Modes	151
7.7	Motion Function Blocks	156
7.8	Administrative Function Blocks	189

Section 7.1

Description

Overview

This section describes the Pulse Train Output function.

What Is in This Section?

This section contains the following topics:

Topic	Page
Pulse Train Output (PTO)	103
Pulse Output Modes	106
Acceleration / Deceleration Ramp	108
Probe Event	111
Backlash Compensation	114
Positioning Limits	116

Pulse Train Output (PTO)

Introduction

The M221 PTO function provides pulse train output channels for a specified number of pulses and a specified velocity (frequency). The PTO function is used to control the positioning or speed of independent linear single-axis stepper or servo drives in open loop mode. The PTO function does not have any position feedback information from the process. Therefore, position information must be integrated in the drive.

The **PLS** (pulse), **PWM** (pulse width modulation), **PTO** (pulse train output), and **FREQGEN** (frequency generator) functions use the same dedicated outputs. Only one of these four functions can be used on the same channel.

A PTO channel can use optional interface signals for homing (Ref), event (Probe), limits (LimP, LimN), or drive interface (DriveReady, DriveEnable).

Automatic origin offset and backlash compensation are also managed to improve positioning accuracy. Diagnostics are available for status monitoring.

Supported Functions

The PTO channels support the following functions:

- two output modes (two channels for Pulse and Direction or one channel for CW/CCW)
- single axis moves (velocity and position)
- relative and absolute positioning, with automatic direction management
- trapezoidal and S-curve acceleration and deceleration
- homing (four modes with offset compensation)
- dynamic acceleration, deceleration, velocity, and position modification
- switch from speed to position mode
- move queuing (buffer of one move)
- position capture and move trigger on event (using probe input)
- backlash compensation
- limits (hardware and software)
- diagnostics

NOTE: Motion function blocks (*see page 156*) and administrative function blocks (*see page 189*) help you to program these functions.

PTO Characteristics

There are up to five physical inputs for a PTO channel:

- Two are assigned to the PTO function through configuration and are taken into account upon a rising edge on the input:
 - Ref input
 - Probe input
- Three are assigned to the MC_Power_PTO (*see page 161*) function block. They have no fixed assignment (they are not configured in the configuration screen), and are read with all other inputs:
 - DriveReady input
 - Limit positive input
 - Limit negative input

NOTE: These inputs are managed like any other regular input, but are used by the PTO function when assigned to MC_Power_PTO (*see page 161*) function block.

NOTE: The positive and negative limit inputs are required to help prevent over-travel.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Ensure that controller hardware limit switches are integrated in the design and logic of your application.
- Mount the controller hardware limit switches in a position that allows for an adequate braking distance.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

There are up to three physical outputs for a PTO channel:

- Two outputs are mandatory to manage the output mode of the PTO function. They have a fixed assignment and must be enabled by configuration:
 - CW / CCW
 - Pulse / Direction
- The other output, DriveEnable, is associated with the MC_Power_PTO (*see page 161*) function block. It has no fixed assignment and is written at the end of the MAST cycle as regular outputs.

The PTO function has the following characteristics:

Characteristic	Value
Number of channels	2 or 4 depending on the module
Number of axis	1 per channel
Position range	-2,147,483,648...2,147,483,647 (32 bits)
Minimum velocity	0 Hz

Characteristic	Value
Maximum velocity	100 kHz (for a 40/60 duty cycle and max. 200 mA)
Minimum step	1 Hz
Accuracy on velocity	1 %
Acceleration / deceleration (min)	1 Hz/ms
Acceleration / deceleration (max)	100 kHz/ms
Origin offset	-2,147,483,648...2,147,483,647 (32 bits)
Software limits range	-2,147,483,648...2,147,483,647 (32 bits)

Pulse Output Modes

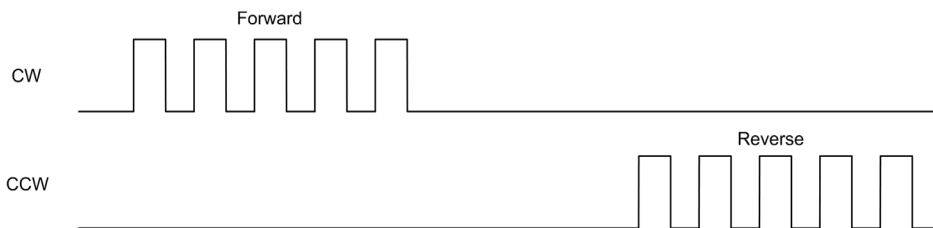
Overview

There are two possible output modes:

- ClockWise / CounterClockwise
- Pulse / Direction

ClockWise (CW) / CounterClockwise (CCW) Mode

This mode generates a signal that defines the motor operating speed and direction. This signal is implemented on the first PTO channel (PTO0 only).



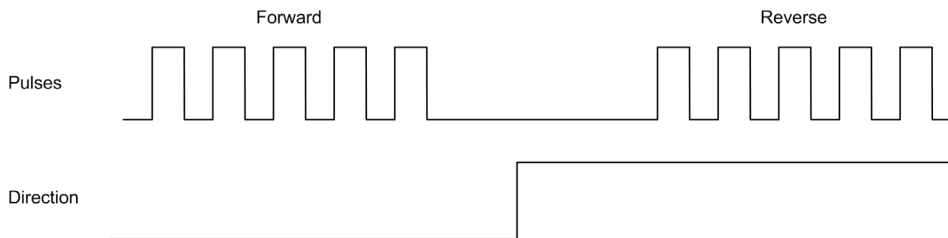
NOTE: PTO1 is not available when choosing this mode.

Pulse / Direction Mode

This mode generates two signals on the PTO channels:

- The pulse output provides the motor operating speed (Pulses).
- The direction output provides the motor rotation direction (Direction).

NOTE: The direction output can be disabled if not needed for the application.



Special Cases

Special Case	Description
Effect of a cold restart (%S0=TRUE)	<ul style="list-style-type: none">● The axis is set to <i>Disabled</i> state.● The PTO function blocks are initialized.
Effect of a warm restart (%S1=TRUE)	<ul style="list-style-type: none">● The axis is set to <i>Disabled</i> state.● The PTO function blocks are initialized.
Effect at controller stop	<ul style="list-style-type: none">● The axis is set to <i>ErrorStop</i> state.● The outputs are reset to 0.
Effect of online modification	None

Acceleration / Deceleration Ramp

Start Velocity

The **Start Velocity** is the minimum frequency at which a stepper motor can produce movement, with a load applied, without the loss of steps.

Start Velocity parameter is used when starting a motion from velocity 0.

Start Velocity must be in the range 0...MaxVelocityAppl.

Value 0 means that the **Start Velocity** parameter is not used. In this case, the motion starts at a velocity = acceleration rate x 1 ms.

Stop Velocity

The **Stop Velocity** is the maximum frequency at which a stepper motor stops producing movement, with a load applied, without loss of steps.

Stop Velocity is only used when moving from a higher velocity than **Stop Velocity**, down to velocity 0.

Stop Velocity must be in the range 0...MaxVelocityAppl.

Value 0 means that the **Stop Velocity** parameter is not used. In this case, the motion stops at a velocity = deceleration rate x 1 ms.

Acceleration / Deceleration

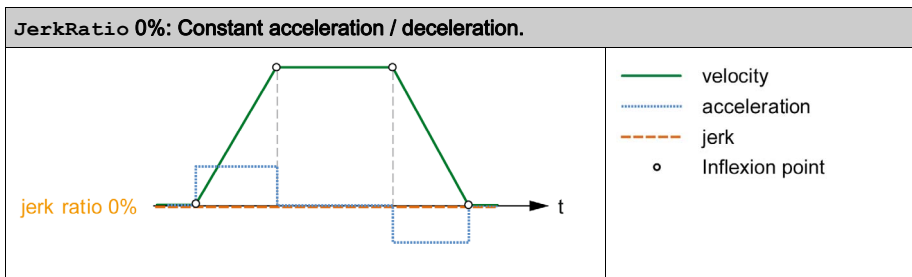
Acceleration is the rate of velocity change, starting from **Start Velocity** to target velocity.

Deceleration is the rate of velocity change, starting from target velocity to **Stop Velocity**. These velocity changes are implicitly managed by the PTO function in accordance with `Acceleration`, `Deceleration` and `JerkRatio` parameters following a **trapezoidal** or an **S-curve** profile.

Acceleration / Deceleration Ramp with a Trapezoidal Profile

When the jerk ratio parameter is set to 0, the acceleration / deceleration ramp has a trapezoidal profile.

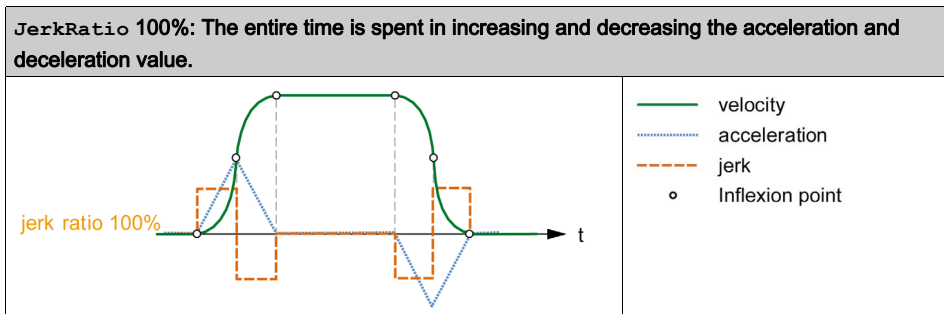
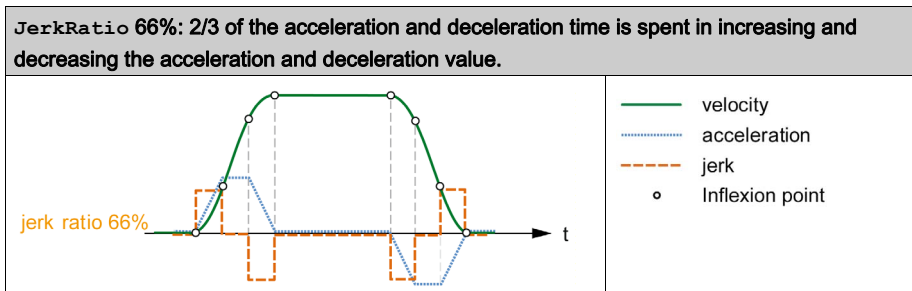
Expressed in Hz/ms, the `acceleration` and `deceleration` parameters represent the rate of velocity change.



Acceleration / Deceleration Ramp with an S-curve Profile

When the JerkRatio parameter is greater than 0, the acceleration / deceleration ramp has an S-curve profile.

The S-curve ramp is used in applications controlling high inertia, or in those that manipulate fragile objects or liquids. The S-curve ramp enables a smoother and progressive acceleration / deceleration, as demonstrated in the following graphics:



NOTE: The JerkRatio parameter value is common for acceleration and deceleration so that concave time and convex time are equal.

Affect of the S-Curve Ramp on Acceleration / Deceleration

The duration for the acceleration / deceleration is maintained, whatever the `JerkRatio` parameter may be. To maintain this duration, the acceleration or deceleration is other than that configured in the function block (`Acceleration` or `Deceleration` parameters).

When the `JerkRatio` is applied, the acceleration / deceleration is affected.

When the `JerkRatio` is applied at 100%, the acceleration / deceleration is two times that of the configured `Acceleration/Deceleration` parameters.

NOTE: If the `JerkRatio` parameter value is invalid, the value is re-calculated to respect the `MaxAccelerationAppl` and `MaxDecelerationAppl` parameters. `JerkRatio` is invalid when its value is greater than 100. In this case, a `JerkRatio` of 100 is applied.

Probe Event

Description

The Probe input is enabled by configuration, and activated using the `MC_TouchProbe_PTO` function block.

The Probe input is used as an event to:

- capture the position,
- start a move independently of the task.

Both functions can be active at the same time, that is, the same event captures the position and start a motion function block.

NOTE: Only the first event after the rising edge at the `MC_TouchProbe_PTO` function block `Busy` output is valid. Once the `Done` output is set to `TRUE`, subsequent events are ignored. The function block needs to be reactivated to respond to other events.

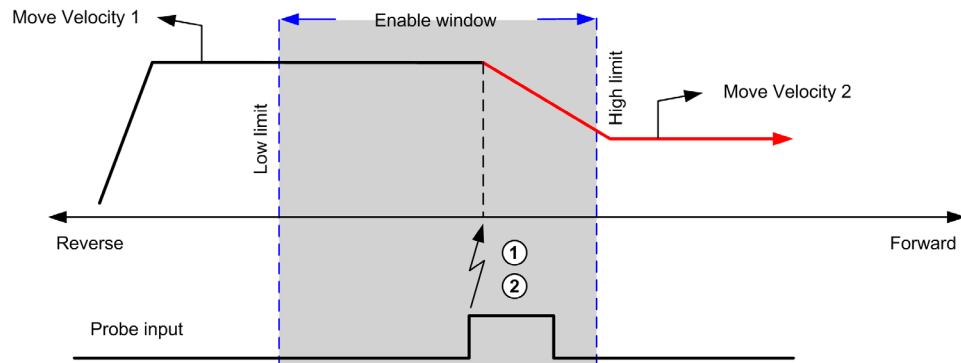
Position Capture

The position captured is available in `%MC_TouchProbe_PTO.RecordedPos`.

Motion Trigger

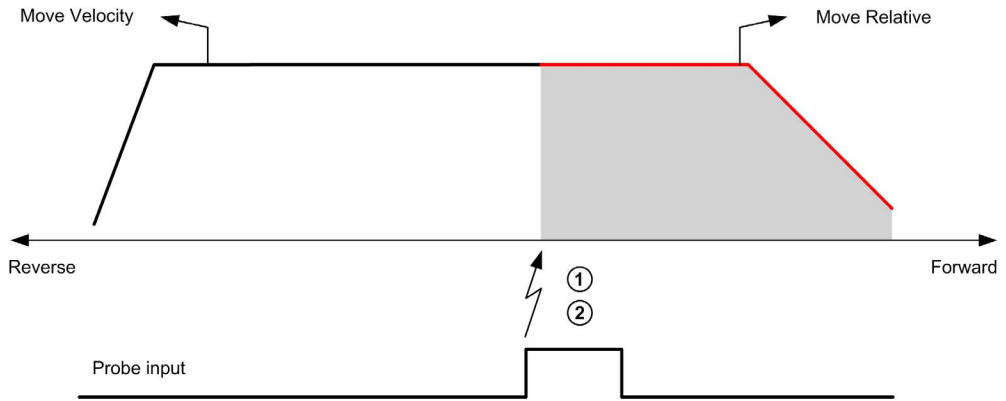
The `BufferMode` input of a motion function block must be set to `seTrigger`.

This example illustrates a change target velocity with enable window:



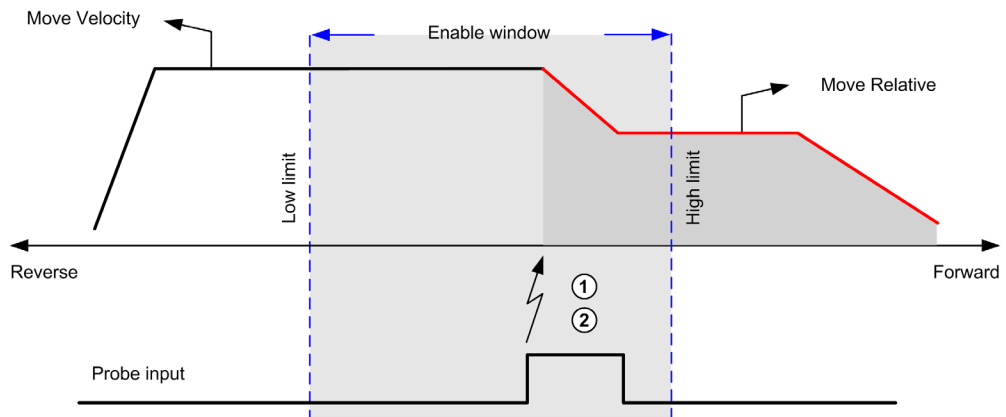
- 1 Capture the position counter value
- 2 Trigger `Move Velocity` function block

This example illustrates a move of pre-programmed distance, with simple profile and no enable window:



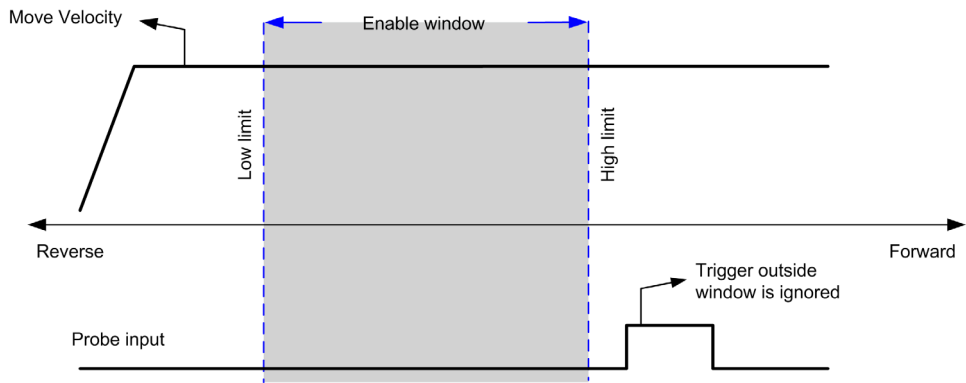
- 1 Capture the position counter value
- 2 Trigger `Move Relative` function block

This example illustrates a move of pre-programmed distance, with complex profile and enable window:



- 1 Capture the position counter value
- 2 Trigger `Move Relative` function block

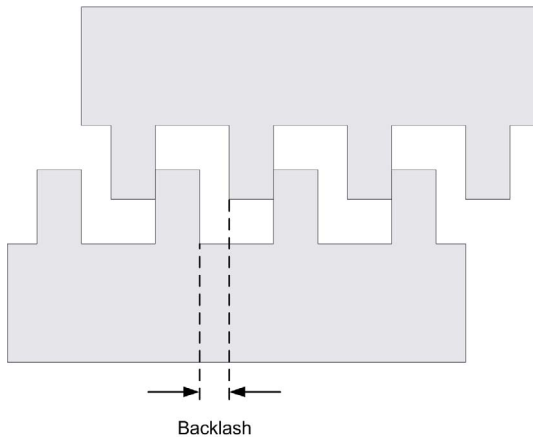
This example illustrates a trigger event out of enable window:



Backlash Compensation

Description

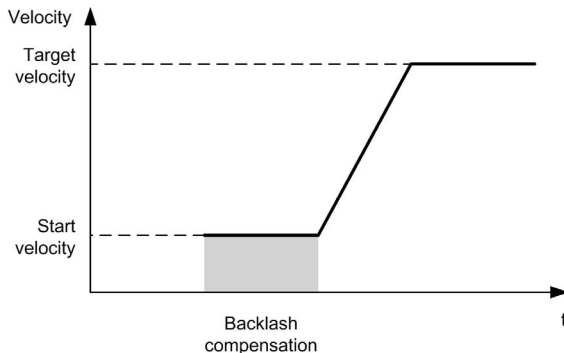
The **Backlash Compensation** parameter is defined as the amount of motion needed to compensate for the mechanical clearance in gears (backlash) when a movement is reversed:



NOTE: The function does not take into account external sources of movement, such as inertia movement or other forms of induced movement.

Backlash compensation is set in number of pulses (0...65535, default value is 0). When set, at each direction reversal, the specified number of pulses is first emitted at start velocity, and then the programmed movement is executed. The backlash compensation pulses are not added to the position counter.

This figure illustrates the backlash compensation:



NOTE:

- Before the initial movement is started, the function cannot determine the amount of backlash to compensate for. Therefore, the backlash compensation is only active after a first move is performed and the compensation is applied at the first direction reversal.
- If an aborting command is received or an error detected before the backlash completion, the absolute position remains unchanged.
- After an abort command, the backlash resumes from current backlash position when a new move is started.

For more details, refer to the Configuring Pulse Train Output (*see Modicon M221, Logic Controller, Programming Guide*).

Positioning Limits

Introduction

Positive and negative limits can be set to control the movement boundaries in both directions. Both hardware and software limits are managed by the controller.

Hardware and software limit switches are used to manage boundaries in the controller application only. They are not intended to replace any functional safety limit switches wired to the drive. The controller application limit switches must necessarily be activated before the functional safety limit switches wired to the drive. In any case, the type of functional safety architecture, which is beyond the scope of the present document, that you deploy depends on your safety analysis, including, but not limited to:

- risk assessment according to EN/ISO 12100
- FMEA according to EN 60812

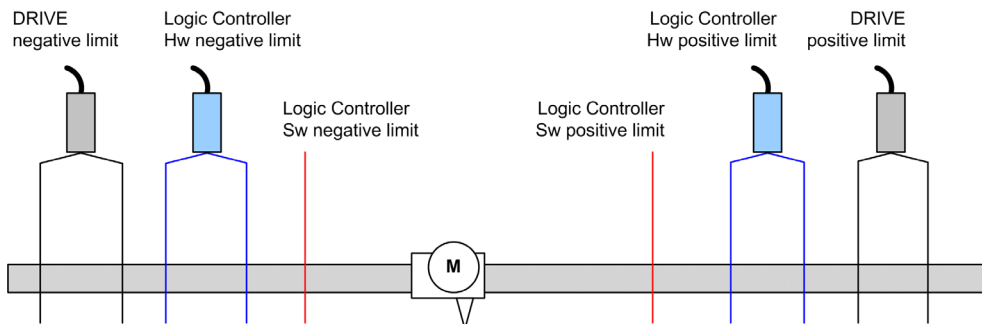
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that a risk assessment is conducted and respected according to EN/ISO 12100 during the design of your machine.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The figure illustrates hardware and software limit switches:

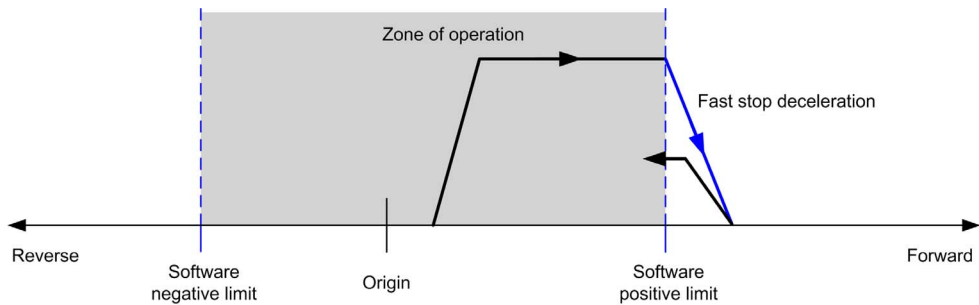


Once either the controller hardware or software limits are crossed, an error is detected and a Fast stop deceleration is performed:

- the axis switches to **ErrorStop** state, with `AxisErrorId` 1002 to 1005. Refer to `MC_ReadAxisError_PTO` (see page 198) and `Axis Control Advisory Alerts` (see page 148).
- the current direction becomes invalid and the associated PTO parameter `EnableDirPos` (1004) or `EnableDirNeg` (1005) is reset to 0 by the system.
- the function block under execution detects the error state,
- on other applicable function blocks, the `CmdAborted` outputs are set to TRUE.

To clear the axis error state, and return to a **Standstill** state, execution of `MC_Reset_PTO` is required as any motion command will be rejected (refer to PTO parameters (*see page 147*) `EnableDirPos` or `EnableDirNeg`) while the axis remains outside the limits (function block terminates with `ErrorId=InvalidDirectionValue`). It is only possible to execute a motion command in the opposite direction under these circumstances.

Once the axis is inside the limits, the `EnableDirPos` or `EnableDirNeg` parameter is restored to 1 (valid) by the system.



NOTE: In previous diagram, the axis move back in the limits is the result of the execution of `MC_Reset_PTO` (it is not performed automatically).

Software Limits

Software limits can be set to control the movement boundaries in both directions.

Limit values are enabled and set in the configuration screen, such that:

- Positive limit > Negative limit
- Values in the range -2,147,483,648 to 2,147,483,647

They can also be enabled, disabled, or modified in the application program (`MC_WritePar_PTO` and PTO Parameter (*see page 147*)).

NOTE: When enabled, the software limits are valid after an initial homing is successfully performed (that is, the axis is homed, `MC_Home_PTO`).

Hardware Limits

Hardware limits are required for the homing procedure, and for helping to prevent damage to the machine. The appropriate inputs must be used on the `%MC_Power_PTO.LimP` and `%MC_Power_PTO.LimN` inputs. The hardware limit devices must be of a normally closed type such that the input to the function block is FALSE when the respective limit is reached.

NOTE: The restrictions over movement are valid while the limit inputs are FALSE and regardless of the sense of direction. When they return to TRUE, movement restrictions are removed and the hardware limits are functionnally rearmed. Therefore, use falling edge contacts leading to RESET output instructions prior to the function block. Then use those bits to control these function block inputs. When operations are complete, SET the bits to restore normal operation.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Ensure that controller hardware limit switches are integrated in the design and logic of your application.
- Mount the controller hardware limit switches in a position that allows for an adequate braking distance.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTE: Adequate braking distance is dependent on the maximum velocity, maximum load (mass) of the equipment being moved, and the value of the Fast stop deceleration parameter.

Section 7.2 Configuration

Overview

This section describes how to configure a PTO channel and the associated parameters.

What Is in This Section?

This section contains the following topics:

Topic	Page
PTO Configuration	120
Motion Task Table	121

PTO Configuration

Overview

To configure the `Pulse Generator` resource, refer to the Modicon M221 Logic Controller Programming Guide, Configuring Pulse Generators (*see Modicon M221, Logic Controller, Programming Guide*).

To configure the `Pulse Generator` resource as a PTO, refer to the Modicon M221 Logic Controller Programming Guide, PTO Configuration (*see Modicon M221, Logic Controller, Programming Guide*).

Motion Task Table

Overview

The Motion Task Table is a programming possibility for motion function blocks, dedicated to repetitive motion sequences. A sequence of movements is defined for an axis at configuration time (a sequence can be compared as a recipe that mixes various movements).

The Motion Task Table can be dedicated to several axes and offers a graphical overview of the configured motion sequence.

Use the `MC_MotionTask_PTO` function block to execute a Motion Task Table. When the table is called by the `MC_MotionTask_PTO` function block, it needs to be associated to a specific axis. The Motion Task Table is applied to the axis used by the `MC_MotionTask_PTO` function block. Several `MC_MotionTask_PTO` function blocks can execute the same %MT Motion Task Table instances simultaneously.

Features

The maximum number of Motion Task Table (%MT) instances is 4.

A Motion Task Table contains a sequence of single-axis movements:

- A sequence is a succession of steps.
- Each step defines the parameters of a movement.
- Each step uses a dedicated motion function block instance.

Movements that can be used in the Motion Task Table:

- Move absolute
- Move relative
- Halt
- Set position
- Move velocity

Configuring a Motion Task Table

The **Motion Task Table Assistant** allows you to configure each movement in an ordered sequence and visualize an estimated global movement profile.

To display the **Motion Task Table Assistant**, proceed as follows:

Step	Action																														
1	<p>Select the Programming → Tools module tab and click PTO objects → Motion Task Tables in the hardware tree to display the Motion Task Table properties.</p> <p>tooltipMotionTaskTables properties</p> <table border="1"> <thead> <tr> <th></th> <th>Configured</th> <th>Address</th> <th>Symbol</th> <th>Configuration</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td></td> <td><input type="checkbox"/></td> <td>%MT0</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td><input type="checkbox"/></td> <td>%MT1</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td><input type="checkbox"/></td> <td>%MT2</td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td><input type="checkbox"/></td> <td>%MT3</td> <td></td> <td></td> <td></td> </tr> </tbody> </table>		Configured	Address	Symbol	Configuration	Comment		<input type="checkbox"/>	%MT0					<input type="checkbox"/>	%MT1					<input type="checkbox"/>	%MT2					<input type="checkbox"/>	%MT3			
	Configured	Address	Symbol	Configuration	Comment																										
	<input type="checkbox"/>	%MT0																													
	<input type="checkbox"/>	%MT1																													
	<input type="checkbox"/>	%MT2																													
	<input type="checkbox"/>	%MT3																													
2	Click [...] to configure the Motion Task Table.																														

Motion Task Table properties window description:

Parameter	Editable	Value	Default Value	Description
Configured	No	True/False	False	Indicates whether the Motion Task Table contains configured steps.
Address	No	%MTx	%MTx	Displays the address of the Motion Task Table where x is the table number.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the Motion Task Table. Double-click the cell to edit the field.
Configuration	Yes	[...] (Button)	Enabled	Allows you to configure the sequence of movements using the Motion Task Table Assistant .
Comment	Yes	–	–	Allows you to specify a comment to associate with the Motion Task Table. Double-click the cell to edit the field.

Motion Task Table Assistant:

Motion Task Table Assistant
✕

Steps

Step	Type	Pos	Distance	Vel	Acc	Dec	Jerk ratio	Next step	Event	Delay	Software Object
1	MC_MoveAbs_P	2000		5000	20	50	0	Done		10	%MC_MOVEA
2	MC_MoveRel_P*		5000	7500	20	100	0	Done		0	%MC_MOVER
3	MC_MoveRel_P*		5000	4000	20	200	0	SW event %M1	%M1	1000	%MC_MOVER
4	MC_Halt_PTO					1	0	Done		0	%MC_HAL_PT
5	None										
6	None										

Use probe event range

First position
 Last position

Motion overview

? *The graph presented below may not represent real-world events. Consult the product documentation for more information.*

Motion Task Table Assistant main areas:

- **Steps** : lists the sequence of single axis movements and input parameters for each movement.
- **Motion overview** : click the refresh button, or **F5**, to generate a graphical view of the movement implemented by the steps sequence.

The curve provides a general overview of the movement. The curve is based on the following assumptions:

- Initial position is 0.
- Position limits are not enabled.
- Axis default motion configuration parameters are used.
- An event (probe input, POU) occurs after the step completion and a 100 ms delay.
- A %MWx delay is graphically represented by a 100 ms delay.

Steps window description:

Parameter	Value	Default Value	Description
Step	1...16	–	Single axis movement number in the sequence.
Type	None Move absolute Move relative Halt Set position Move velocity	None	Motion command. The motion command uses one motion function block instance indicated in the Software Objects parameter.
Pos	See each software object function block parameter value.	<i>empty</i>	The move parameters are the parameters of the software object assigned to the step. Parameter description: <ul style="list-style-type: none"> ● Pos: Position ● Distance: Distance ● Vel: Velocity ● Acc: Acceleration ● Dec: Deceleration ● Jerk ratio: Jerk ratio NOTE: Vel parameter for the move velocity motion command is a combination of velocity and direction. In the table, the velocity range for the MC_MoveVel_PTO motion command is: - Max Velocity...+ Max Velocity. A negative velocity indicates a negative direction, a positive velocity indicates a positive direction.
Distance			
Vel			
Acc			
Dec			
Jerk ratio			

Parameter	Value	Default Value	Description
Next step	Done / In velocity, Blending previous, Probe input event, SW event, Delay	<i>empty</i>	<p>The condition that needs to be fulfilled to proceed to the next step in the table sequence.</p> <p>Condition description:</p> <ul style="list-style-type: none"> ● Done / In velocity: <ul style="list-style-type: none"> ○ Done: Proceed to the next step when the present step is completed. This parameter is available for the different motion commands except move velocity. ○ In velocity: Proceed to the next step when the requested velocity is reached. This parameter is only available for the move velocity motion command. ● Blending previous: The velocity of next step is blended with the velocity at the end-position of this step. ● Probe input event: Proceed to the next step when a defined event is detected on the Probe input. The edge is defined in the Event parameter. An input field opens in the bottom of the Steps window, Use probe event range, described in next table. NOTE: One occurrence of Probe input event can be used per Motion Task Table. ● SW event: Proceed to the next step when the memory bit address (%Mx) set in the Event parameter is set to 1. ● Delay: Proceed to the next step when the delay (starting at the beginning of the step) elapses. The delay is defined in the Delay parameter. <p>NOTE: When the Probe input event, or SW event, or Delay event occurs, the next step is started even if the present step is not completed.</p>
Event	– 0/1 %Mx	<i>empty</i>	<p>Event value complements the conditions described in Next step parameter.</p> <p>Next step choice and corresponding Event choice:</p> <ul style="list-style-type: none"> ● Probe input event: <ul style="list-style-type: none"> ○ 0: Falling edge ○ 1: Rising edge <p>NOTE: The probe input event is independent of the application task cycle and the motion task cycle.</p> <ul style="list-style-type: none"> ● SW event: Memory bit %Mx. <p>NOTE: %Mx is evaluated every 4 ms.</p>

Parameter	Value	Default Value	Description
Delay	0...65535 %MWx	<i>empty</i>	<p>Delay value represents the amount of time before proceeding to the next step. Depending on the . Next step parameter value, the Delay is evaluated from the beginning or the end of the step:</p> <ul style="list-style-type: none"> ● Done / In velocity: The delay starts when the present step is Done or In Velocity. ● Blending previous: Not available. ● Probe input event and SW event: The delay starts at the beginning of the step. <ul style="list-style-type: none"> ○ An elapsed delay generates a timeout if the event did not occur, and next step is proceeded. ○ If the event occurs before the end of the delay, the next step is proceeded and the delay timeout is aborted. <p>NOTE: If Delay remains to its default value (0), the motion command waits for the probe input or software event to occur, without timeout.</p> <ul style="list-style-type: none"> ● Delay: The delay starts at the beginning of the step. The next step is proceeded when the delay is elapsed. <p>NOTE: An immediate value cannot be modified in an application POU, whereas, a %MWx value must be set by an application POU. The Motion Task Table Delay parameter is not modified if MC_ReadPar_PTO or MC_WritePar_PTO are set using ParNumber = 1000 (delay).</p>
Software Objects	%MC_MOVEABS_PTOx %MC_MOVEREL_PTOx %MC_HALT_PTOx %MC_SETPOS_PTOx %MC_MOVEVEL_PTOx	<i>empty</i>	Shows the software object allocated to the step. It is allocated by the system and is a read-only parameter. Those software objects are function block instances.
Symbol	–	<i>empty</i>	Allows to specify a symbol to associate with the step software object. Double-click the cell to edit the field.

Use probe event range parameter in the **Steps** window:

Parameter	Value	Default Value	Description
Use probe event range	True/False	False	When TRUE, a trigger event is only recognized within the position range defined between First position and Last position . The parameter can be modified if Next step is set to <code>Probe input event</code> in the Motion Task Table.
First position	- 2147483648... 2147483647 %MDx	- 2147483648	NOTE: First position must be less than Last position .
Last position	- 2147483648... 2147483647 %MDx	2147483647	
Illustration of the position range influence on triggering is provided in the section on Probe Event (<i>see page 111</i>).			
NOTE: The position where the trigger event was detected is not recorded.			

Managing Step Parameters and Event

The parameters and event defined in a step are only valid at the start of the step execution, therefore:

- A step parameter value modified by the application is only valid if it is modified before the step is active. The parameter can be modified using system allocated software object parameter in a POU.
- A memory object value (%MW or %MWx) is only valid if updated before the step is active.
- An event is only evaluated once the step is active. In the case of a `Probe input event`, an event occurring before the step is active cannot be detected.

Managing Function Block Instances Used in a Motion Task Table

System allocated software object instances:

- cannot be used in an application POU to control an axis motion.
- Output parameters are not updated by the system during the execution of the Motion Task Table. In other words, the output bits and output parameters are not valid.
- Input parameters:
 - cannot be modified in the software object instance editor, or in the **Programming** tab.
 - can be used to dynamically modify the Motion Task Table in an application POU. To dynamically modify a system allocated software object instance input parameter, use the parameter address or its associated symbol.

NOTE: The executing step can be modified but the modifications will not be taken into account until the next execution of the step.

Example of movement described in a Motion Task Table:

- Step: 2
- Movement type: Move relative
- Software object: %MC_MOVEREL_PTO1
- Symbol: Move_Relative_Label2

In previous example, the velocity input parameter can be modified by program using one of the following syntaxes:

- %MC_MOVEREL_PTO1.Vel
- Move_Relative_Label2.Vel

Management of the function block instances used in a Motion Task Table:

- When a Motion Task Table is configured, the reserved function block instances are set as **Used**.
- If all the instances of a specific function block are reserved, the associated move type cannot be used anymore.

Section 7.3

Programming

Overview

This section lists the function blocks used to program the PTO function and describes how to add or remove those function blocks.

What Is in This Section?

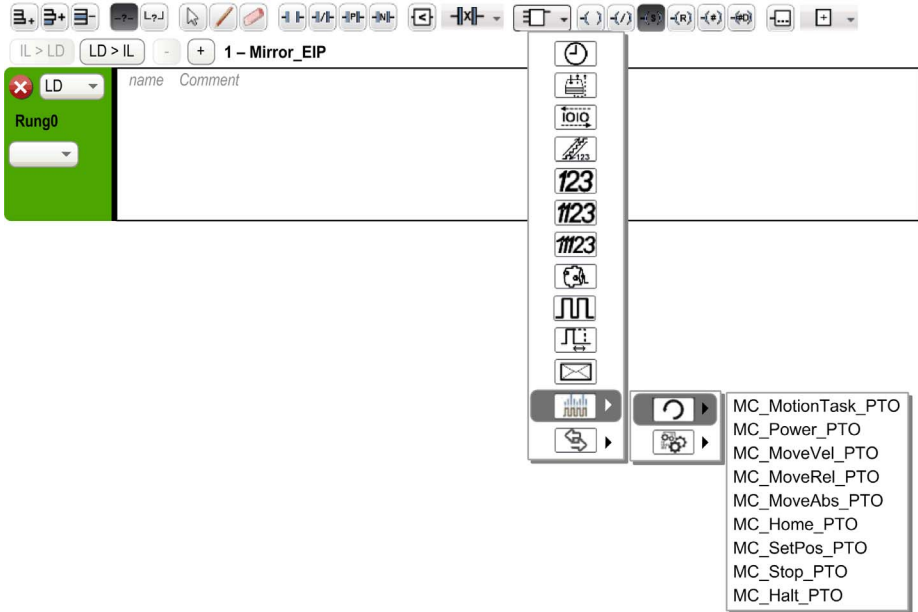
This section contains the following topics:

Topic	Page
Adding / Removing a Function Block	130
PTO Function Blocks	132

Adding / Removing a Function Block

Adding a Function Block

Follow these steps to add an instance of a PTO function block:

Step	Action
1	Select the Programming tab.
2	<p>Select Function Blocks → PTO → Administrative or Function Blocks → PTO → Motion as shown in the following graphic:</p> 
3	Click into the rung to place the selected function block.
4	Associate the input/output variables of the function block.

NOTE: Set the parameters in the **Configuration** tab.

For more details, refer to the Modicon M221 Logic Controller Programming Guide, PTO Configuration.

Removing a Function Block

Follow these steps to remove an instance of a PTO function block:

Step	Action
1	In the Programming tab, click the instance of the function block.
2	Press Delete to remove the selected function block.

PTO Function Blocks

Function Blocks

The PTO function is programmed in SoMachine Basic using the following function blocks:

Category	Function Block	Description
Motion (single axis) (see page 156)	MC_MotionTask_PTO (see page 157)	Calls a Motion Task Table.
	MC_Power_PTO (see page 161)	Enables power to the axis, switching the axis state from Disabled to Standstill. While the %MC_Power_PTO.Status bit is FALSE, no motion function block can be executed for that axis.
	MC_MoveVel_PTO (see page 164)	Causes the specified axis to move at the specified speed, and transfer the axis to the state Continuous. This continuous movement is maintained until a software limit is reached, an aborting move is triggered, or a transition to ErrorStop state is detected.
	MC_MoveRel_PTO (see page 169)	Moves the specified axis an incremental distance at the specified speed, and transfer the axis to the state Discrete. The target position is referenced from the current position at execution time, incremented by a distance.
	MC_MoveAbs_PTO (see page 174)	Causes the specified axis to move towards a given position at the specified speed, and transfer the axis to the state Discrete. The function block terminates with Error set to TRUE if the axis is not Homed (no absolute reference position is defined). In this case, ErrorId is set to InvalidAbsolute.
	MC_Home_PTO (see page 178)	Commands the axis to perform the sequence defining the absolute reference position, and transfers the axis to the state Homing (see page 134). The details of this sequence depend on Homing configuration parameters setting.
	MC_SetPos_PTO (see page 181)	Modifies the coordinates of the axis without any physical movement.
	MC_Stop_PTO (see page 183)	Commands a controlled motion stop and transfers the axis to the state Stopping. It aborts any ongoing move execution.
	MC_Halt_PTO (see page 186)	Commands a controlled motion stop until the velocity is zero, and transfers the axis to the state Discrete. With the Done output set to TRUE, the state is transferred to Standstill.

Category	Function Block	Description
Administrative (see page 189)	MC_ReadActVel_PTO (see page 190)	Returns the value of the velocity of the axis.
	MC_ReadActPos_PTO (see page 192)	Returns the value of the position of the axis.
	MC_ReadSts_PTO (see page 194)	Returns the state diagram (see page 152) status of the axis.
	MC_ReadMotionState_PTO (see page 196)	Returns the motion status of the axis.
	MC_ReadAxisError_PTO (see page 198)	Returns an axis control error, if any.
	MC_Reset_PTO (see page 200)	Resets all axis-related errors, conditions permitting to allow a transition from the states <code>ErrorStop</code> to <code>Standstill</code> . It does not affect the output of the function blocks instances.
	MC_TouchProbe_PTO (see page 202)	Activates a trigger event on the probe input. This trigger event allows to record the axis position, and/or to start a buffered move.
	MC_AbortTrigger_PTO (see page 204)	Aborts function blocks which are connected to trigger events (for example, <code>MC_TouchProbe_PTO</code>).
	MC_ReadPar_PTO (see page 206)	Gets parameters from the PTO.
	MC_WritePar_PTO (see page 208)	Writes parameters to the PTO.

NOTE: The motion function blocks act on the position of the axis according to the motion state diagram. The administrative function blocks do not influence the motion state.

NOTE: The `MC_Power_PTO` (see page 161) function block is mandatory before a move command can be issued.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use the same function block instance in different program tasks.
- Do not change the function block reference (AXIS) while the function block is executing.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Section 7.4

Home Modes

Overview

This section describes the PTO home modes.

What Is in This Section?

This section contains the following topics:

Topic	Page
Homing Modes	135
Position Setting	138
Long Reference	139
Short Reference No Reversal	141
Short Reference Reversal	143
Home Offset	145

Homing Modes

Description

Homing is the method used to establish the reference point or origin for absolute movement.

A homing movement can be made using different methods. The M221 PTO channels provide several standard homing movement types:

- position setting (*see page 138*),
- long reference (*see page 139*),
- short reference reversal (*see page 143*),
- short reference no reversal (*see page 141*),

A homing movement must be terminated without interruption for the new reference point to be valid.

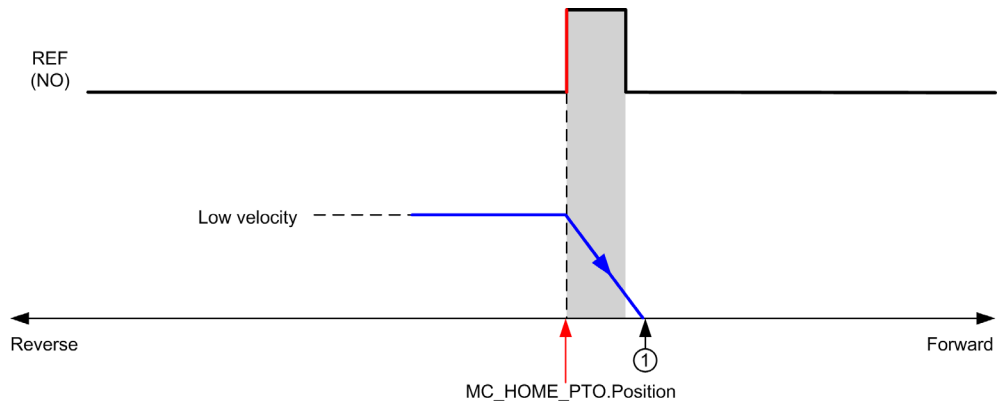
- `%MC_ReadSts_PTO.IsHomed` is set to TRUE when a homing movement is finished successfully. If the homing movement is interrupted, it needs to be started again.
- `%MC_ReadSts_PTO.IsHomed` is set to FALSE when the axis state is DISABLED, or when no homing movement was finished successfully.

Refer to `MC_Home_PTO` (*see page 178*) and home modes function block object codes (*see page 147*).

Home Position

Homing is done with an external switch and the homing position is defined on the switch edge. Then the motion is decelerated until stop.

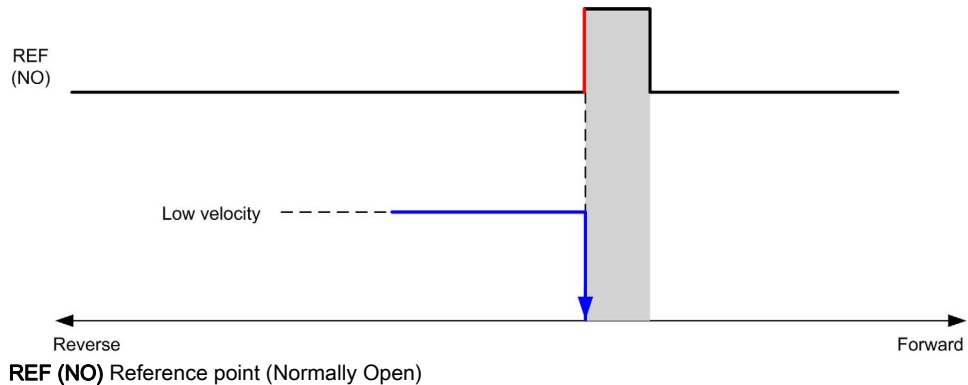
The actual position of the axis at the end of the motion sequence may therefore differ from the position parameter set on the function block:



REF (NO) Reference point (Normally Open)

1 Position at the end of motion = `%MC_HOME_PTO.Position` + “deceleration to stop” distance.

To simplify the representation of a stop in the homing mode diagrams, the following presentation is made to represent the actual position of the axis:



Limits

Hardware limits are necessary for the correct functioning of the MC_Home_PTO function block (Positioning Limits [\(see page 116\)](#) and MC_Power_PTO). Depending on the movement type you request with the homing mode, the hardware limits help assure that the end of travel is respected by the function block.

When a homing action is initiated in a direction away from the reference switch, the hardware limits serve to either:

- indicate a reversal of direction is required to move the axis toward the reference switch or,
- indicate that an error has been detected as the reference switch was not found before reaching the end of travel.

For homing movement types that allow for reversal of direction, when the movement reaches the hardware limit the axis stops using the configured deceleration, and resumes motion in a reversed direction.

In homing movement types that do not allow for the reversal of direction, when the movement reaches the hardware limit, the homing procedure is aborted and the axis stops with the Fast stop deceleration.

 **WARNING**

UNINTENDED EQUIPMENT OPERATION

- Ensure that controller hardware limit switches are integrated in the design and logic of your application.
- Mount the controller hardware limit switches in a position that allows for an adequate braking distance.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTE: Adequate braking distance is dependent on the maximum velocity, maximum load (mass) of the equipment being moved, and the value of the Fast stop deceleration parameter.

Position Setting

Description

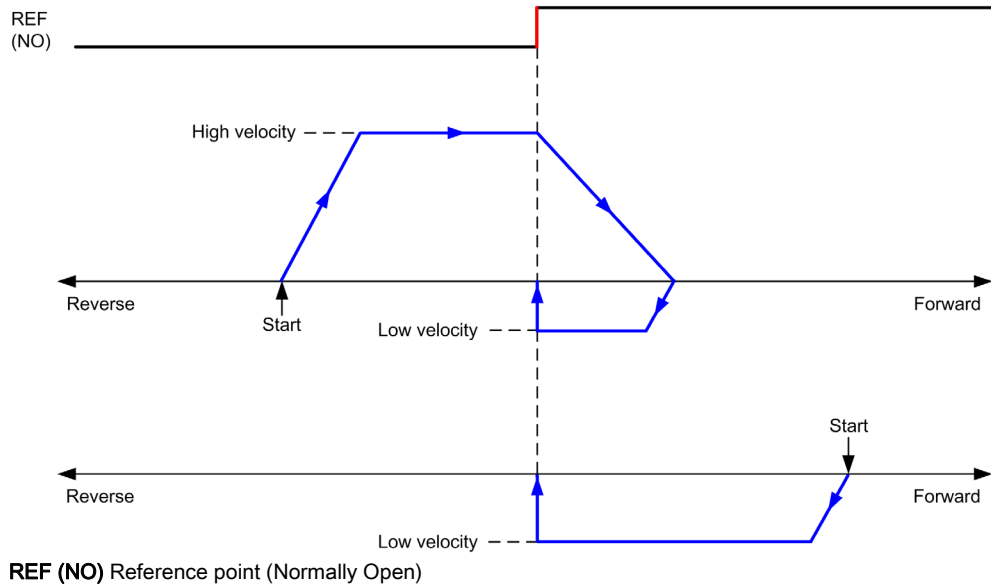
In the case of position setting, the current position is set to the specified position value. No move is performed.

Long Reference

Long Reference: Positive Direction

Homes to the reference switch falling edge in reverse direction.

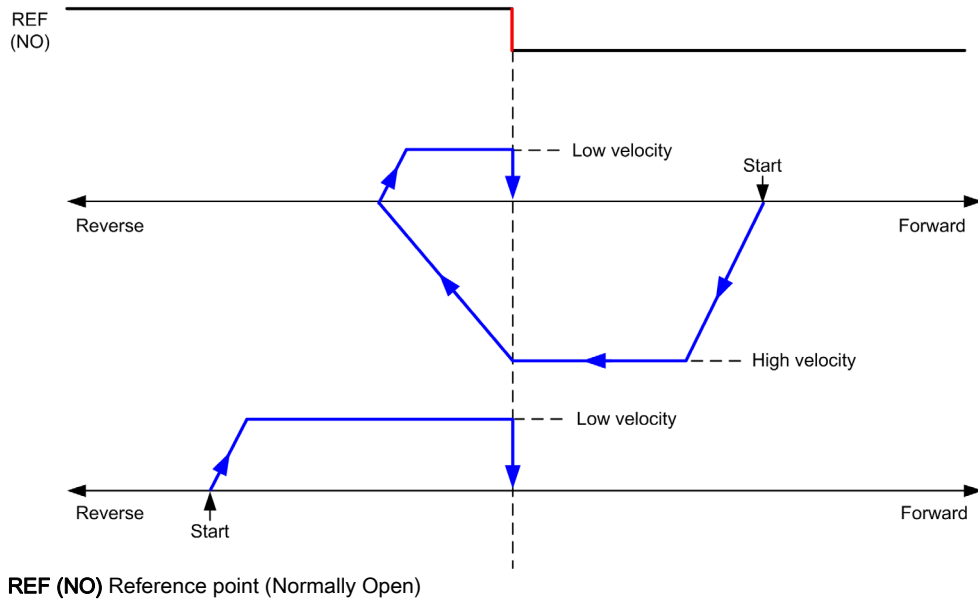
The initial direction of motion is dependent on the state of the reference switch:



Long Reference: Negative Direction

Homes to the reference switch falling edge in forward direction.

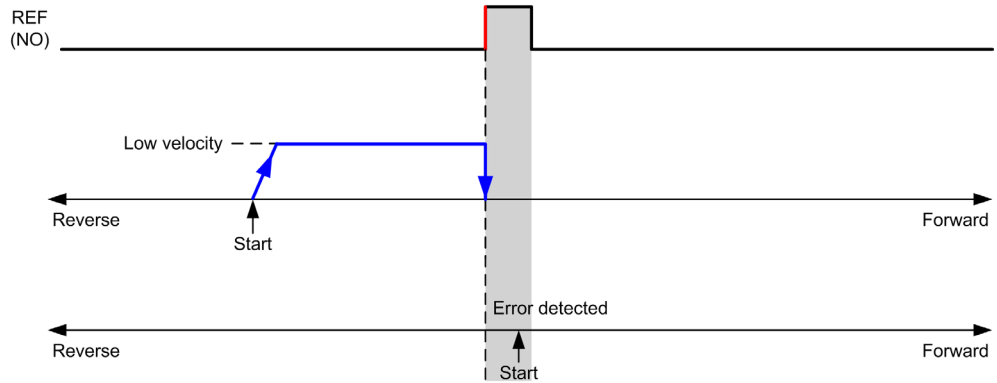
The initial direction of motion is dependent on the state of the reference switch:



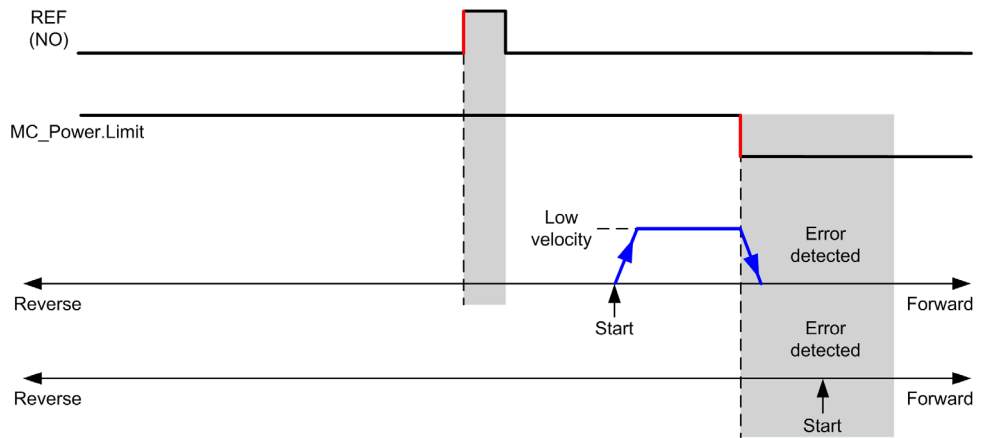
Short Reference No Reversal

Short Reference No Reversal: Positive Direction

Homes at low speed to the reference switch rising edge in forward direction, with no reversal:



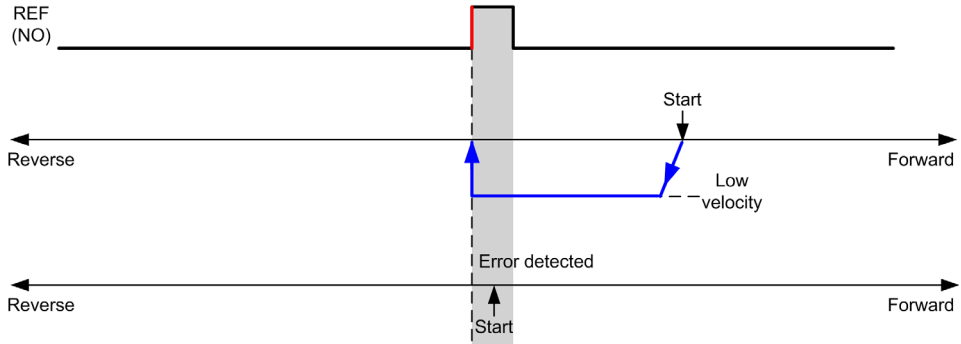
REF (NO) Reference point (Normally Open)



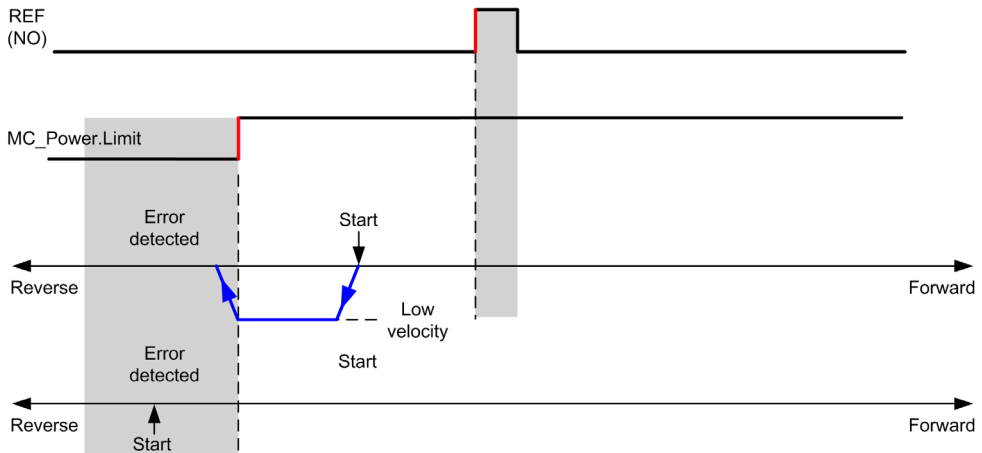
REF (NO) Reference point (Normally Open)

Short Reference No Reversal: Negative Direction

Homes at low speed to the reference switch falling edge in reverse direction, with no reversal:



REF (NO) Reference point (Normally Open)



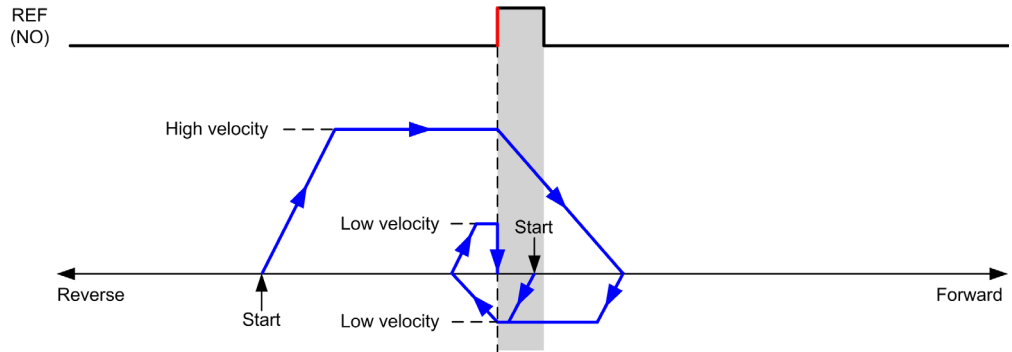
REF (NO) Reference point (Normally Open)

Short Reference Reversal

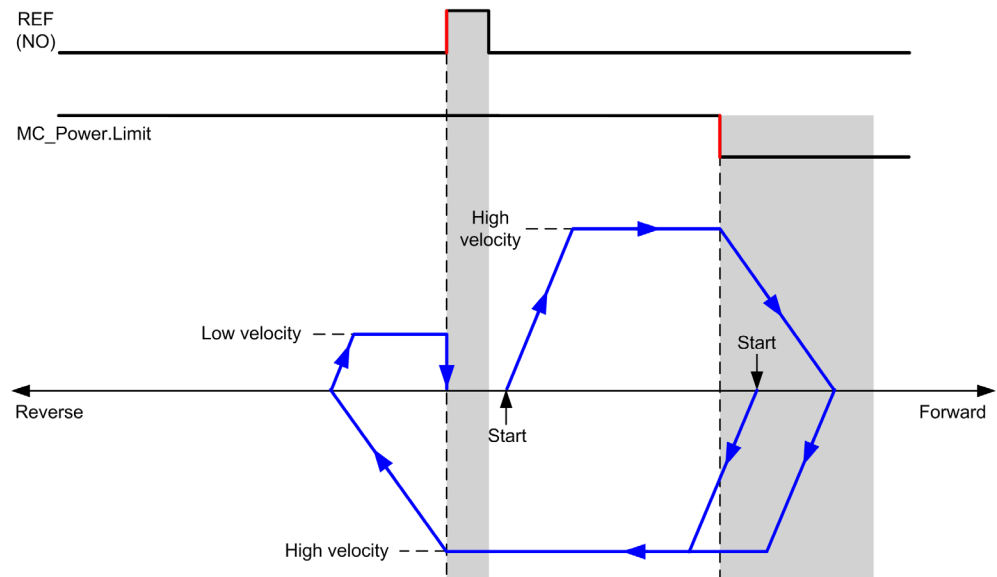
Short Reference Reversal: Positive Direction

Homes to the reference switch rising edge in forward direction.

The initial direction of motion is dependent on the state of the reference switch:



REF (NO) Reference point (Normally Open)

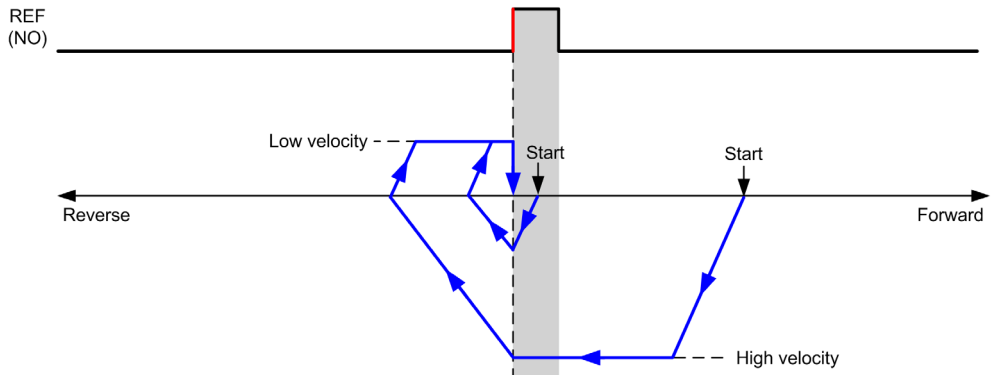


REF (NO) Reference point (Normally Open)

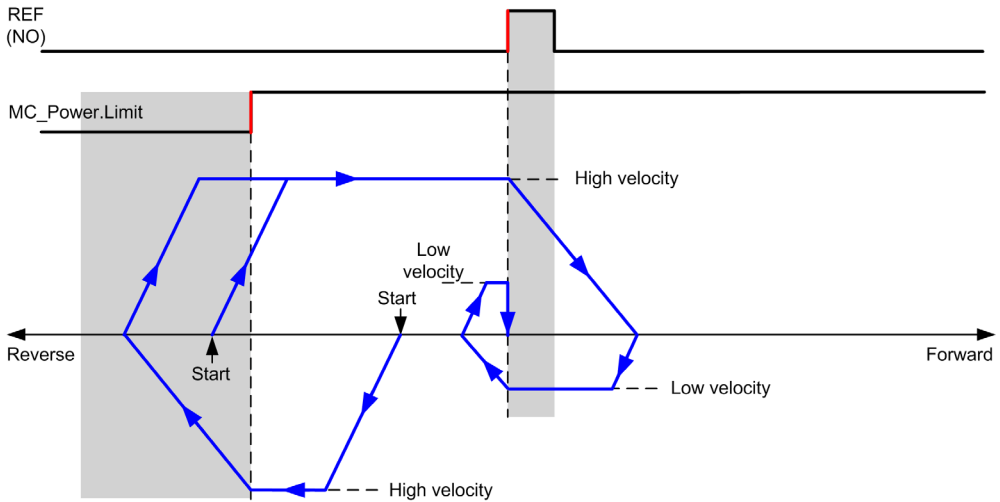
Short Reference Reversal: Negative Direction

Homes to the reference switch rising edge in forward direction.

The initial direction of motion is dependent on the state of the reference switch:



REF (NO) Reference point (Normally Open)



REF (NO) Reference point (Normally Open)

Home Offset

Description

If the origin cannot be defined by switches with enough accuracy, it is possible to make the axis move to a specific position away from the origin switch. Home offset allows making a difference between mechanical origin and electrical origin.

Home offset is set in number of pulses (-2,147,483,648...2,147,483,647, default value 0). When set by configuration, the `MC_Home_PTO` command is executed first, and then the specified number of pulses is output at the home low velocity in the specified direction.

NOTE: The wait time between `MC_Home_PTO` command stop on origin switch and start of offset movement is fixed, set to 500 ms. The `MC_Home_PTO` command busy flag is only released after origin offset has been completed.

Section 7.5

Data Parameters

Function Block Object Codes

Direction

This table lists the values for the direction function block object codes:

Name	Value	Description
mcPositiveDirection	1	CW, forward, positive (according to Output Mode configuration setting).
mcNegativeDirection	-1	CCW, backward, reverse, negative (according to Output Mode configuration setting).

Buffer Modes

This table lists the values for the buffer modes function block object codes:

Name	Value	Description
mcAborting	0	Start FB immediately (default mode). Any ongoing motion is aborted. The move queue is cleared.
mcBuffered	1	Start FB after current motion has finished (<i>Done</i> or <i>InVel</i> bit is set to TRUE). There is no blending.
mcBlendingPrevious	3	The velocity is blended with the velocity of the first FB (blending with the velocity of <i>FB1</i> at end-position of <i>FB1</i>).
seTrigger	10	Start FB immediately when an event on the Probe input is detected. Any ongoing motion is aborted. The move queue is cleared.
seBufferedDelay	11	Start FB after current motion has finished (<i>Done</i> or <i>InVel</i> output is set to TRUE) and the time delay has elapsed. There is no blending. The <i>Delay</i> parameter is set using <i>MC_WritePar_PTO</i> , with <i>ParameterNumber</i> 1000.

Homing Modes

This table lists the values for the homing modes function block object codes:

Name	Value	Description
PositionSetting	0	Position.
LongReference	1	Long reference.
ShortReference_Reversal	20	Short reference.
ShortReference_NoReversal	21	Short reference no reversal.

PTO Parameter

This table lists the values for the PTO parameters function block object codes:

Name	Parameter Number	R/W	Description
CommandedPosition	1	R	Commanded position.
SWLimitPos (High Limit)	2	R/W	Positive software position limit.
SWLimitNeg (Low Limit)	3	R/W	Negative software position limit.
EnableLimitPos (Enable the Software Position Limits)	4	R/W	Enable positive software limit switch (0...1).
EnableLimitNeg (Enable the Software Position Limits)	5	R/W	Enable negative software limit switch (0...1).
MaxVelocityAppl (Max. Velocity)	9	R/W	Maximal allowed velocity of the axis in the application (0...100,000).
ActualVelocity	10	R	Velocity of the axis.
CommandedVelocity	11	R	Commanded velocity.
MaxAccelerationAppl (Max. acc.)	13	R/W	Maximal allowed acceleration of the axis in the application (0...100,000).
MaxDecelerationAppl (Max. dec.)	15	R/W	Maximal allowed deceleration of the axis in the application (0...100,000).
Reserved	16 to 999	-	Reserved for the PLCopen standard.
Delay	1000	R/W	Time in ms (0...65,535) Default value: 0

Name	Parameter Number	R/W	Description
EnableDirPos	1004	R/W	<p>Enable positive direction. When value = 0, the positive direction is not allowed on the axis. A move function block that would generate a move in a positive direction ends with <code>InvalidDirectionValue</code> error detected (3006). If there is an ongoing movement in the negative direction, and it is interrupted by a new move command in the positive direction, the error will be detected only at the end of the deceleration of the ongoing negative movement. Default value: 1</p> <p>NOTE: A value change is only taken into account at the next move command or the next occurrence of velocity = 0.</p>
EnableDirNeg	1005	R/W	<p>Enable negative direction. When value = 0, the negative direction is not allowed on the axis. A move function block that would generate a move in a negative direction ends with <code>InvalidDirectionValue</code> error detected (3006). If there is an ongoing movement in the positive direction, and it is interrupted by a new move command in the negative direction, the error will be detected only at the end of the deceleration of the ongoing positive movement. Default value: 1</p> <p>NOTE: A value change is only taken into account at the next move command or the next occurrence of velocity = 0.</p>

PTO Axis Error Codes

This table lists the values for the PTO axis error codes:

Name	Value	Description
NoError	0	No error detected.
Axis Control Alerts		
InternalError	1000	Motion controller internal error detected.
DisabledAxis	1001	The move could not be started or has been aborted because the axis is not ready.
HwPositionLimitP	1002	Hardware positive position limit <code>limP</code> exceeded.
HwPositionLimitN	1003	Hardware negative position limit <code>limN</code> exceeded.
SwPositionLimitP	1004	Software positive position limit exceeded.

Name	Value	Description
SwPositionLimitN	1005	Software negative position limit exceeded.
ApplicationStopped	1006	Application execution has been stopped (controller in STOPPED or HALT state).
OutputProtection	1007	Short-circuit output protection is active on the PTO channels. Refer to the description of %S10 and %SW139 in the Modicon M221 Logic Controller - Programming Guide, system bits and system words (see <i>Modicon M221, Logic Controller, Programming Guide</i>).
Axis Control Advisories		
WarningVelocityValue	1100	Commanded Velocity parameter is out of range, therefore velocity is limited to the configured maximum velocity.
WarningAccelerationValue	1101	Commanded Acceleration parameter is out of range, therefore acceleration is limited to the configured maximum acceleration.
WarningDecelerationValue	1102	Commanded Deceleration parameter is out of range, therefore deceleration is limited to the configured maximum deceleration.
WarningJerkRatioValue	1103	Commanded jerk ratio parameter is limited by the configured maximum acceleration or deceleration. In this case, the jerk ratio is recalculated to respect these maximums.

An **Axis Control Alert** switches the axis in **ErrorStop** state (MC_Reset_PTO is mandatory to get out of **ErrorStop** state). The resulting axis status is reflected by MC_ReadSts_PTO and MC_ReadAxisError_PTO.

PTO Motion Command Error Codes

This table lists the values for the PTO motion command error codes:

Name	Value	Description
NoError	0	No error detected.
Motion State Advisory Alerts		
ErrorStopActive	2000	The move could not be started or has been aborted because motion is prohibited by an ErrorStop condition.
StoppingActive	2001	The move could not be started because motion is prohibited by MC_Stop_PTO having control of the axis (either the axis is stopping, or MC_Stop_PTO.Execute input is held TRUE).
InvalidTransition	2002	Transition not allowed, refer to the Motion State Diagram.

Name	Value	Description
InvalidSetPosition	2003	MC_SetPos_PTO cannot be executed while the axis is moving.
HomingError	2004	Homing sequence cannot start on reference cam in this mode.
InvalidProbeConf	2005	The Probe input must be configured.
InvalidHomingConf	2006	The Ref input must be configured for this homing mode.
InvalidAbsolute	2007	An absolute move cannot be executed while the axis is not successfully homed to an origin position. A homing sequence must be executed first (MC_Home_PTO).
MotionQueueFull	2008	The move could not be buffered because the motion queue is full.
InvalidTransitionMotionTask	2009	The motion task and the other motion function blocks linked to the same axis cannot be executed concurrently.
Range Advisory Alerts		
InvalidAxis	3000	The function block is not applicable for the specified axis.
InvalidPositionValue	3001	Position parameter is out of limits, or distance parameter gives an out of limits position.
InvalidVelocityValue	3002	Velocity parameter is out of range.
InvalidAccelerationValue	3003	Acceleration parameter is out of range.
InvalidDecelerationValue	3004	Deceleration parameter is out of range.
InvalidBufferModeValue	3005	Buffer mode does not correspond to a valid value.
InvalidDirectionValue	3006	Direction does not correspond to a valid value, or direction is invalid due to software position limit exceeded.
InvalidHomeMode	3007	Homing mode is not applicable.
InvalidParameter	3008	The parameter number does not exist for the specified axis.
InvalidParameterValue	3009	Parameter value is out of range.
ReadOnlyParameter	3010	Parameter is read-only.
InvalidStepMotionTask	3011	Motion task step type is not defined.

A **Motion State Alert** or a **Range Alert** does not affect the axis state, nor any move currently executing, nor the move queue. In this case, the error is only local to the applicable function block: the `Error` output is set to TRUE, and the `ErrorId` object output is set to the appropriate PTO motion command error code.

Section 7.6

Operation Modes

Overview

This section describes the operation modes.

What Is in This Section?

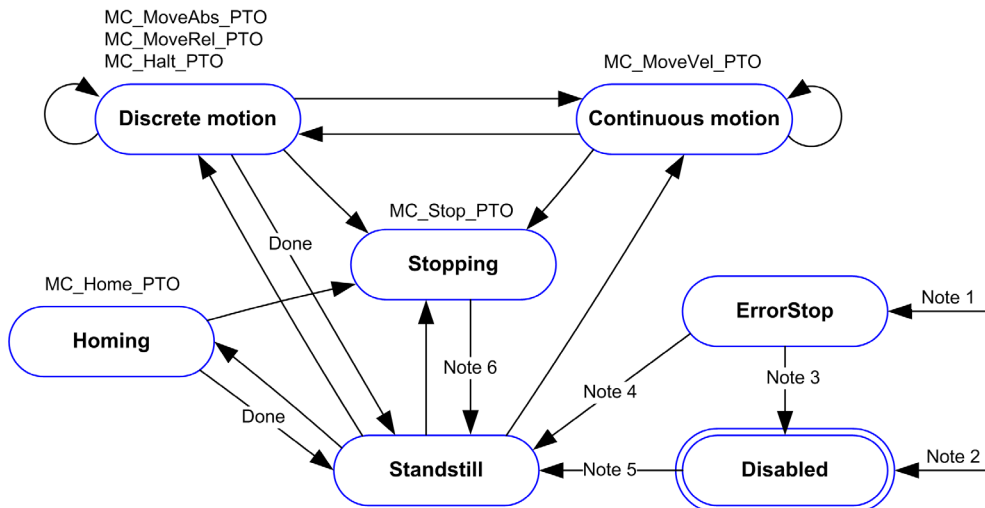
This section contains the following topics:

Topic	Page
Motion State Diagram	152
Buffer Mode	154

Motion State Diagram

State Diagram

The axis is always in one of the defined states in this diagram:



Note 1 From any state, when an error is detected.

Note 2 From any state except `ErrorStop`, when `%MC_Power_PTO.Status = FALSE`.

Note 3 `%MC_Reset_PTO.Done = TRUE` and `%MC_Power_PTO.Status = FALSE`.

Note 4 `%MC_Reset_PTO.Done = TRUE` and `%MC_Power_PTO.Status = TRUE`.

Note 5 `%MC_Power_PTO.Status = TRUE`.

Note 6 `%MC_Stop_PTO.Done = TRUE` and `%MC_Stop_PTO.Execute = FALSE`.

The table describes the axis states:

State	Description
Disabled	Initial state of the axis, no motion command is allowed. The axis is not homed.
Standstill	Power is on, no error is detected, and no motion commands are active on the axis. Motion command is allowed.
ErrorStop	Highest priority, applicable when an error is detected on the axis or in the controller. Any ongoing move is aborted by a Fast Stop Deceleration . <code>Error</code> output is set to <code>TRUE</code> on applicable function blocks, and an <code>ErrorId</code> sets the error code. As long as an error is pending, the state remains <code>ErrorStop</code> . No further motion command is accepted until a reset has been done using <code>MC_Reset_PTO</code> .
Homing	Applicable when <code>MC_Home_PTO</code> controls the axis.
Discrete	Applicable when <code>MC_MoveRel_PTO</code> , <code>MC_MoveAbs_PTO</code> , or <code>MC_Halt_PTO</code> controls the axis.

State	Description
Continuous	Applicable when MC_MoveVel_PTO controls the axis.
Stopping	Applicable when MC_Stop_PTO controls the axis.

NOTE: Function blocks which are not listed in the state diagram do not affect a change of state of the axis.

The entire motion command including acceleration and deceleration ramps cannot exceed 4,294,967,295 pulses. At the maximum frequency of 100 kHz, the acceleration and deceleration ramps are limited to 80 seconds.

Motion Transition Table

The PTO channel can respond to a new command while executing (and before completing) the current command according to the following table:

Command		Next					
		Home	MoveVel	MoveRel	MoveAbs	Halt	Stop
Current	Standstill	Allowed	Allowed ⁽¹⁾	Allowed ⁽¹⁾	Allowed ⁽¹⁾	Allowed	Allowed
	Home	Rejected	Rejected	Rejected	Rejected	Rejected	Allowed
	MoveVel	Rejected	Allowed	Allowed	Allowed	Allowed	Allowed
	MoveRel	Rejected	Allowed	Allowed	Allowed	Allowed	Allowed
	MoveAbs	Rejected	Allowed	Allowed	Allowed	Allowed	Allowed
	Halt	Rejected	Allowed	Allowed	Allowed	Allowed	Allowed
	Stop	Rejected	Rejected	Rejected	Rejected	Rejected	Rejected
<p>⁽¹⁾ When the axis is at standstill, for the buffer modes mcAborting/mcBuffered/mcBlendingPrevious, the move starts immediately. Allowed the new command begins execution even if the previous command has not completed execution. Rejected the new command is ignored and results in the declaration of an error.</p>							

NOTE: When an error is detected in the motion transition, the axis goes into **ErrorStop** state. The **ErrorId** is set to **InvalidTransition**.

Buffer Mode

Description

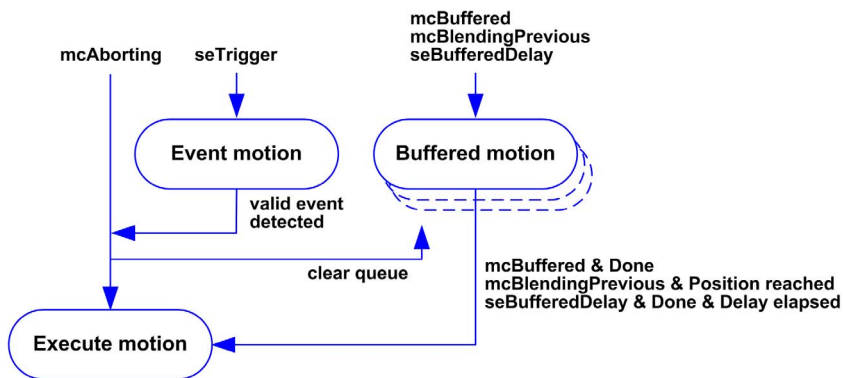
Some of the motion function blocks have an input object called `BufferMode`. With this input object, the function block can either start immediately, start on probe event, or be buffered.

The available options are defined in the buffer modes function block object codes (*see page 146*):

- An aborting motion (`mcAborting`) starts immediately, aborting any ongoing move, and clearing the motion queue.
- An event motion (`seTrigger`) is an aborting move, starting on probe event (*see page 111*).
- A buffered motion (`mcBuffered`, `mcBlendingPrevious`, `seBufferedDelay`) is queued, that is, appended to any moves currently executing or waiting to execute, and starts when the previous motion is done.

Motion Queue Diagram

The figure illustrates the motion queue diagram:



The buffer can contain only one motion function block.

The execution condition of the motion function block present in the buffer is:

- `mcBuffered`: when the current continuous motion is `InVel`, or when the current discrete motion stops.
- `seBufferedDelay`: when the specified delay has elapsed, from the current continuous motion is `InVel`, or from the current discrete motion stops.
- `mcBlendingPrevious`: when the position and velocity targets of current function block are reached.

The motion queue is cleared (all buffered motions are deleted):

- When an aborting move is triggered (`mcAborting` or `seTrigger`): `CmdAborted` output is set to TRUE on buffered function blocks.
- When a `MC_Stop_PTO` function is executed: `Error` output is set to TRUE on cleared buffered function blocks, with `ErrorId=StoppingActive`.
- When a transition to **ErrorStop** state is detected: `Error` output is set to TRUE on buffered function blocks, with `ErrorId=ErrorStopActive`.

NOTE:

- Only a valid motion can be queued. If the function block execution terminates with the `Error` output set to TRUE, the move is not queued, any move currently executing is not affected, and the queue is not cleared.
- When the queue is already full, the `Error` output is set to TRUE on the applicable function block, and `ErrorId` output returns the error `MotionQueueFull`.

Section 7.7

Motion Function Blocks

Overview

This section describes the **Motion** function blocks.

What Is in This Section?

This section contains the following topics:

Topic	Page
MC_MotionTask_PTO Function Block	157
MC_Power_PTO Function Block	161
MC_MoveVel_PTO Function Block	164
MC_MoveRel_PTO Function Block	169
MC_MoveAbs_PTO Function Block	174
MC_Home_PTO Function Block	178
MC_SetPos_PTO Function Block	181
MC_Stop_PTO Function Block	183
MC_Halt_PTO Function Block	186

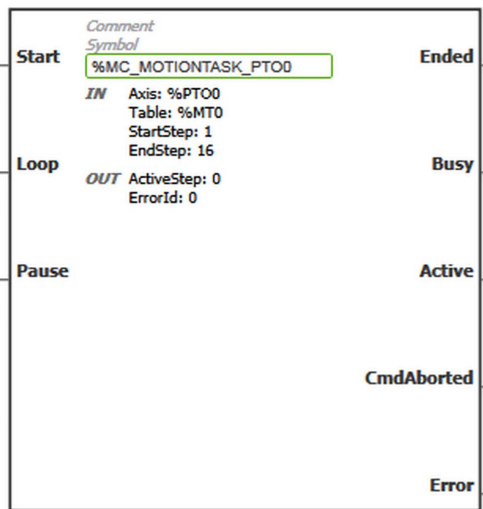
MC_MotionTask_PTO Function Block

Function Description

Both single movement motion function blocks and the Motion Task Table function block (MC_MotionTask_PTO) can be used for an axis.

However, the MC_MotionTask_PTO function block cannot be executed concurrently with another motion function block. If so, an error is detected and the `ErrorId` is set to `InvalidTransition-MotionTask` (2009) (*see page 149*).

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis and motion task table. Double-click the function block to display the function block properties, choose the axis and table, then click `Apply`.

Inputs

This table describes the inputs of the function block:

Input	Initial Value	Description
Start	FALSE	<p>On rising edge, starts the function block execution. The <code>Loop</code> and <code>Pause</code> inputs can be changed during the function block execution and they affect the ongoing execution. The <code>Axis</code>, <code>Table</code>, <code>StartStep</code>, and <code>EndStep</code> input objects values define the motion sequence when the rising edge occurs. A subsequent change in these input objects does not affect the ongoing execution. The outputs are set when the function block execution terminates. When FALSE:</p> <ul style="list-style-type: none"> • When the execution is ongoing (move is <code>Busy</code> and <code>Active</code>), outputs are refreshed. • When the execution is terminated, the outputs are reset one cycle later.
Loop	FALSE	<p>When TRUE, once the function block execution terminates with no detected error, the motion task sequence starts again on <code>StartStep</code>. The <code>Ended</code> output is set for one cycle. The input is tested when the function block execution terminates with no detected error (<code>Ended</code> output is true).</p>
Pause	FALSE	<p>When TRUE:</p> <ul style="list-style-type: none"> • <code>Active</code> = 1 and <code>Busy</code> = 1 • Forces the axis to the Halt state. To reach the Halt state, the axis is decelerating in Discrete motion state, then the axis goes to the Standstill state when velocity = 0. • The Halt state is kept as long as the <code>Pause</code> input is TRUE. • Keeps the <code>Active</code> output set even if velocity is equal to 0. <p>When reset to FALSE after being set to TRUE, the motion task execution resumes in the following conditions:</p> <ul style="list-style-type: none"> • The motion task resumes with the value of the ongoing velocity. • The active step parameters are used. • The absolute target position is not changed. If the motion task is a move relative type, there is no distance added. • In the step, the Next step condition is reset (for example: the delay is restarted from 0, <code>Probe</code> input event is enabled and waiting for the configured edge).

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	%PTOx	–	PTO axis instance for which the function block is to be executed. The parameter is set in the function block instance reached in the Programming → Tools module tab. Select the Axis parameter in PTO objects → Motion → MC_MotionTask_PTO → MC_MotionTask_PTO_properties dialog box.
Table	%MT	–	Table instance for which the function block is to be executed. The parameter is set in the function block instance reached in the Programming → Tools module tab. Select the Table parameter in PTO objects → Motion → MC_MotionTask_PTO → MC_MotionTask_PTO_properties dialog box.
StartStep	Byte	1	Step number that defines the first step executed in the Motion Task Table. The sequence is executed from StartStep to EndStep. Restriction: StartStep ≤ EndStep.
EndStep	Byte	16	Step number that defines the last step executed in the Motion Task Table. The sequence is executed from StartStep to EndStep. Restriction: StartStep ≤ EndStep. NOTE: If EndStep is greater than the maximum number of steps defined in the Motion Task Table, the last step of the table is used.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Ended	0	<p>When TRUE, function block execution is finished with no error detected.</p> <p>Ended output behavior:</p> <ul style="list-style-type: none"> ● If the last step of the motion sequence is a discrete movement, the output behaves like a Done output, the other outputs (Busy, Active, CmdAborted, Error) are reset to 0. ● If the last step of the motion sequence is a continuous movement (move velocity), the output behaves like an InVel output. <p>Other outputs behavior:</p> <ul style="list-style-type: none"> ○ Busy and Active are TRUE (1). ○ CmdAborted and Error are FALSE (0). <p>If a loop is requested (Loop input), the Ended output is TRUE for one task cycle.</p>

Output	Initial Value	Description
Busy	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated.
Active	-	When TRUE, the function block instance has control of the axis. Only one function block at a time can set <code>Active</code> TRUE for the same axis.
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command (<code>MC_Stop_PTO</code>) or an axis error detected.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output objects of the function block:

Output Object	Type	Initial Value	Description
ActiveStep	Byte	0	Number of the step that is being executed in the Motion Task Table.
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

Operating Modes

The execution of a Motion Task Table called by `MC_MotionTask_PTO` function block complies with the motion state diagram ([see page 152](#)).

`MC_MotionTask_PTO` start: The function block can only be started from **Standstill** state.

`MC_MotionTask_PTO` stop: The function block can be stopped by one of the following actions:

- Setting `Pause` input to TRUE.
- Executing a `MC_Stop_PTO`

Function block behavior on detected errors:

- If a motion state or range error is detected during the function block execution:
 - A motion stop command is applied to the motion task using the present step deceleration parameter value. If the step deceleration parameter is not valid, a fast stop deceleration is applied.
 - During the controlled motion stop, the function block outputs `Active` and `Busy` remain TRUE, with the output object `ActiveStep` = 0.
 - Once the motion is stopped, the function block execution is finished with `Error` = 1, and the `ErrorId` output object set to the value corresponding to the detected error type.
- If an axis control error is detected, the axis switches to the **ErrorStop** state. The function block execution is finished with `Error` = 1, and `ErrorId` = 2000.

MC_Power_PTO Function Block

Behavior

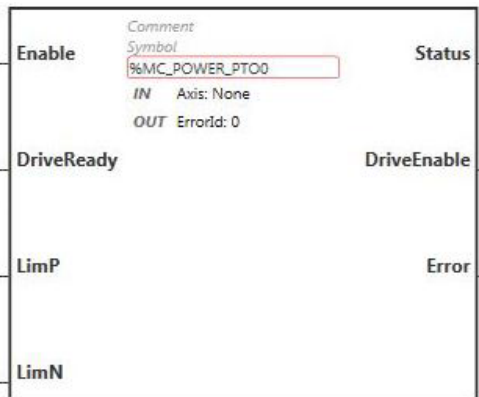
The axis is disabled, when:

- %MC_Power_PTO.Enable = FALSE, or
- %MC_Power_PTO.DriveReady = FALSE, or
- an Hardware limit error is detected (HwPositionLimitP / HwPositionLimitN)

When the axis is disabled, then:

- the Axis switches from Standstill to Disabled state, or from any ongoing move, to ErrorStop, and then Disabled state (when the error is reset).
- %MC_ReadSts_PTO.IsHomed is reset to 0 (a new homing procedure is required).

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the inputs of the function block:

Input	Initial Value	Description
Enable	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.
DriveReady	FALSE	Signal from the drive indicating its readiness. Is set to TRUE when the drive is ready to start executing motion. If the drive signal is connected to the controller, use the appropriate controller input. If the drive does not provide this signal, you can force the value TRUE for this input with any TRUE boolean value.
LimP	TRUE	Hardware limit switch information, in positive direction. Is set to FALSE when the hardware limit switch is reached. If the hardware limit switch signal is connected to the controller, use the appropriate controller input. If this signal is not available, you can force the value TRUE for this input with any TRUE boolean value.
LimN	TRUE	Hardware limit switch information, in negative direction. Is set to FALSE when the hardware limit switch is reached. If the hardware limit switch signal is connected to the controller, use the appropriate controller input. If this signal is not available, you can force the value TRUE for this input with any TRUE boolean value.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

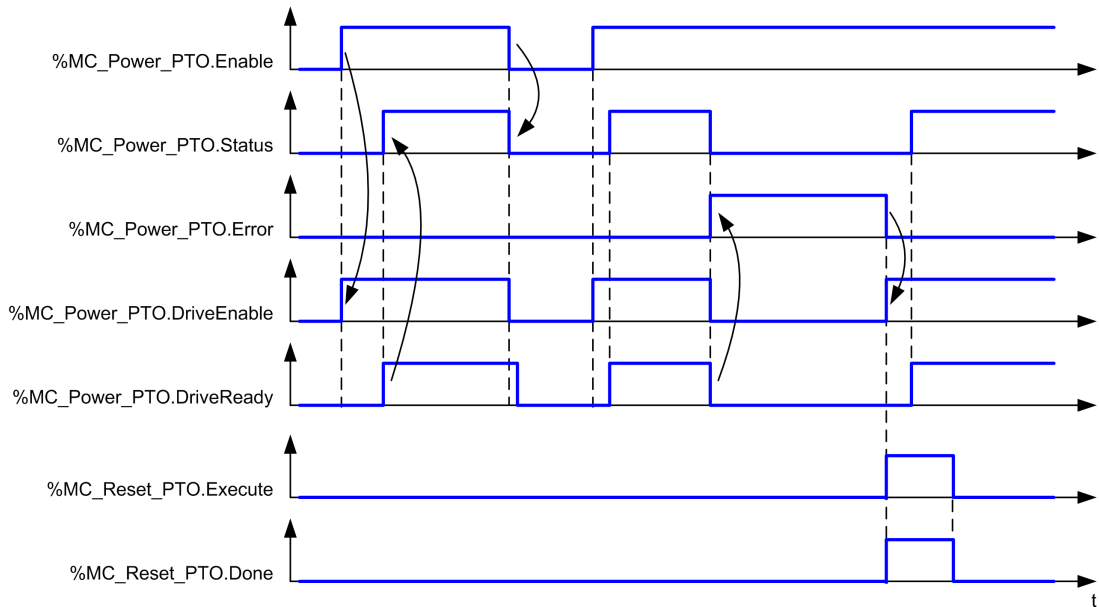
Output	Initial Value	Description
Status	FALSE	When TRUE, the drive is reported as ready to accept motion commands.
DriveEnable	FALSE	When TRUE, indicates to the drive that it can accept motion commands and that it should, therefore, enable power. If the drive input is connected to the controller, use the appropriate controller output. If the drive does not have an input for this signal, you can leave this function block output unused.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table (see page 149).

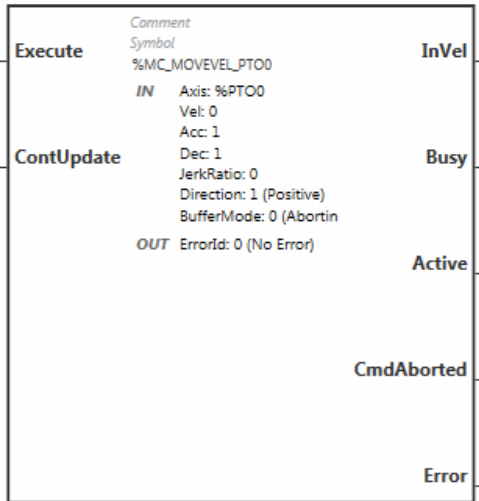
Timing Diagram Example

The diagram illustrates the operation of the MC_Power_PTO function block:



MC_MoveVel_PTO Function Block

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the inputs of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution unless the <code>ContUpdate</code> input is TRUE. The outputs are set when the function block terminates. If a second rising edge is detected during the execution of the function block, the current execution is aborted and the function block is executed again.

Input	Initial Value	Description
ContUpdate	FALSE	<p>When TRUE, makes the function block use any modified values of the input objects (<i>Vel</i>, <i>Acc</i>, <i>Dec</i>, and <i>Direction</i>), and apply it to the ongoing command.</p> <p>This input must be TRUE prior to the rising edge on the <i>Execute</i> input to be taken into account.</p> <p>NOTE: A modification to the value of the <i>Axis</i> parameter is not taken into account. You must set <i>Execute</i> to 0 and then to 1 to change the <i>Axis</i>.</p>

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
<i>Axis</i>	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
<i>Vel</i>	DINT	0	Target velocity. Range Hz: 0...MaxVelocityAppl (<i>see page 147</i>)
<i>Acc</i>	DINT	0	Acceleration in Hz/ms Range (Hz/ms): 1...MaxAccelerationAppl (<i>see page 147</i>)
<i>Dec</i>	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl (<i>see page 147</i>)
<i>JerkRatio</i>	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile (<i>see page 109</i>). Range: 0...100
<i>Direction</i>	INT	mcPositiveDirection	Direction of the movement for PTO type CW/CCW forward (CW) = 1 (mcPositiveDirection) reverse (CCW) = -1 (mcNegativeDirection)
<i>BufferMode</i>	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table (<i>see page 146</i>).

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
<i>InVel</i>	FALSE	When TRUE, the target velocity has been reached.
<i>Busy</i>	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as <i>Busy</i> is TRUE.
<i>Active</i>	-	When TRUE, the function block instance has control of the axis. Only one function block at a time can set <i>Active</i> TRUE for the same axis.

Output	Initial Value	Description
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

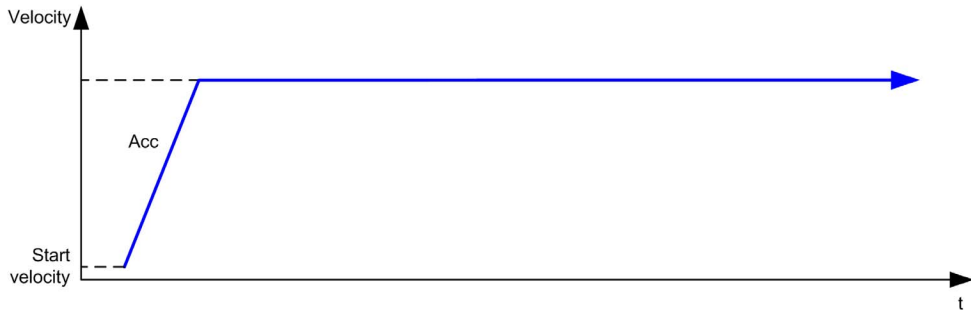
Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table (see page 149).

NOTE:

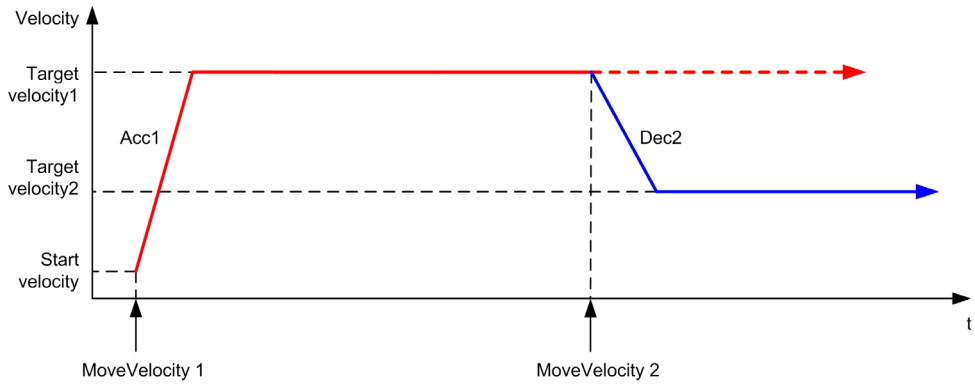
- To stop the motion, the function block has to be interrupted by another function block issuing a new command.
- If a motion is ongoing, and the direction is reversed, first the motion is halted with the deceleration of the MC_MoveVel_PTO function block, and then the motion resumes backward.
- The acceleration/deceleration duration of the segment block must not exceed 80 seconds.

Timing Diagram Example

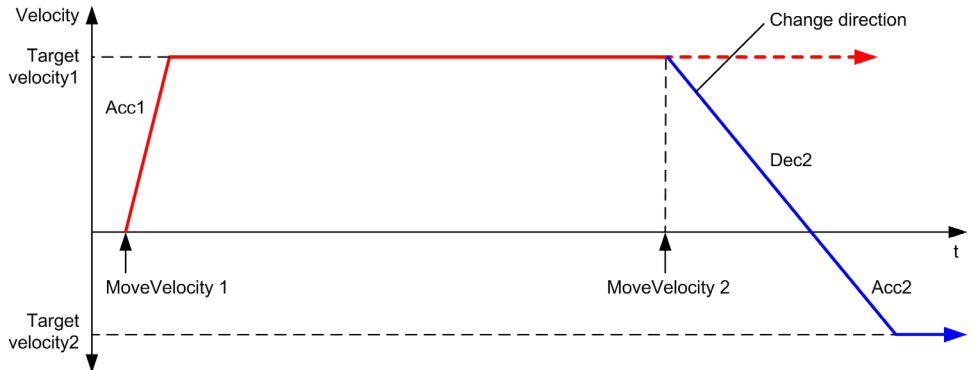
The diagram illustrates a simple profile from **Standstill** state:



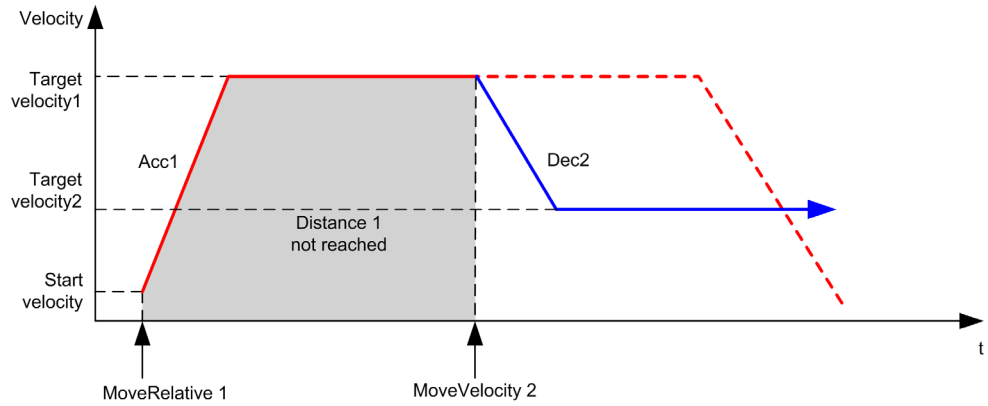
The diagram illustrates a complex profile from **Continuous** state:



The diagram illustrates a complex profile from **Continuous** state with change of direction:

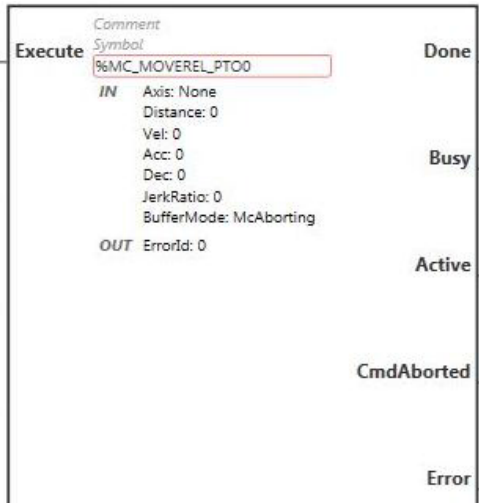


The diagram illustrates a complex profile from **Discrete** state:



MC_MoveRel_PTO Function Block

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click *Apply*.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <i>Execute</i> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
Distance	DINT	0	Relative distance for the motion, in pulses. The sign specifies the direction.

Input Object	Type	Initial Value	Description
Vel	DINT	0	Target velocity. Range Hz: 0...MaxVelocityAppl (<i>see page 147</i>)
Acc	DINT	0	Acceleration in Hz/ms Range (Hz/ms): 1...MaxAccelerationAppl (<i>see page 147</i>)
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl (<i>see page 147</i>)
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile (<i>see page 109</i>). Range: 0...100
BufferMode	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table (<i>see page 146</i>).

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected. When one movement on an axis is interrupted with another movement on the same axis before the commanded action has been completed, <code>CmdAborted</code> is set to TRUE and <code>Done</code> is set to FALSE.
Busy	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as <code>Busy</code> is TRUE.
Active	-	When TRUE, the function block instance has control of the axis. Only one function block at a time can set <code>Active</code> TRUE for the same axis.
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

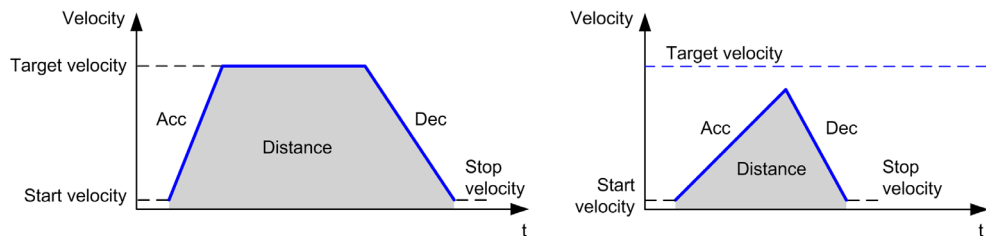
Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (<i>see page 149</i>).

NOTE:

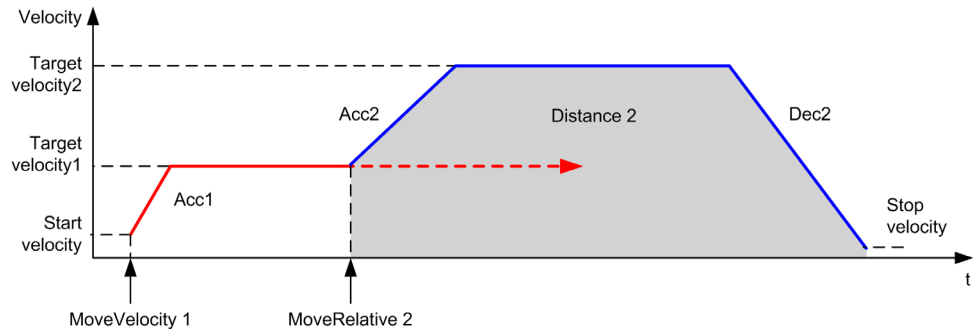
- The function block completes with velocity zero if no further blocks are pending.
- If the distance is too short for the target velocity to be reached, the movement profile is triangular, rather than trapezoidal.
- If a motion is ongoing, and the commanded distance is exceeded due to the current motion parameters, the direction reversal is automatically managed: the motion is first halted with the deceleration of the MC_MoveRel_PTO function block, and then the motion resumes backward.
- The acceleration/deceleration duration of the segment block must not exceed 80 seconds.

Timing Diagram Example

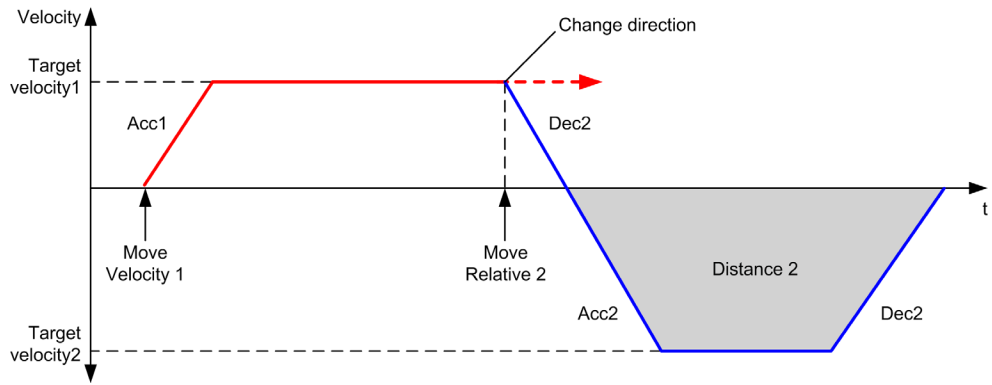
The diagram illustrates a simple profile from **Standstill** state:



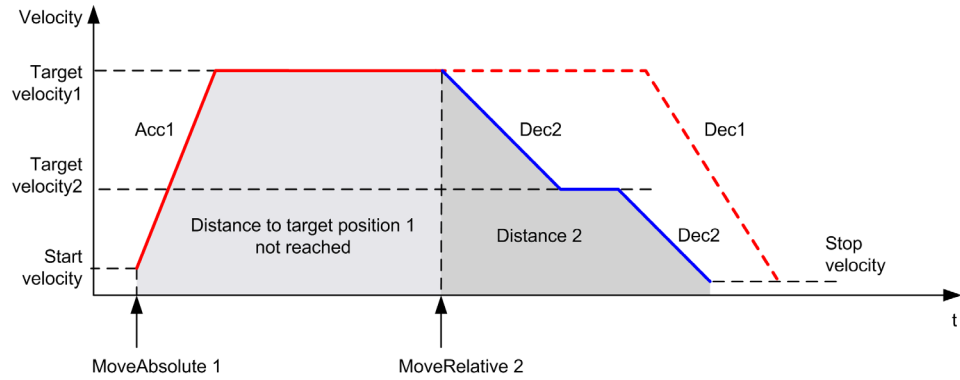
The diagram illustrates a complex profile from **Continuous** state:



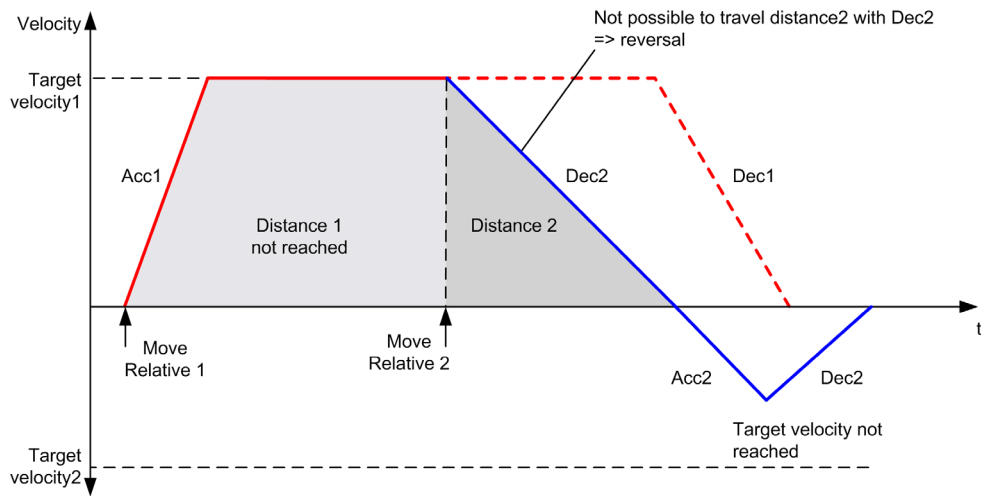
The diagram illustrates a complex profile from **Continuous** state with change of direction:



The diagram illustrates a complex profile from **Discrete** state:

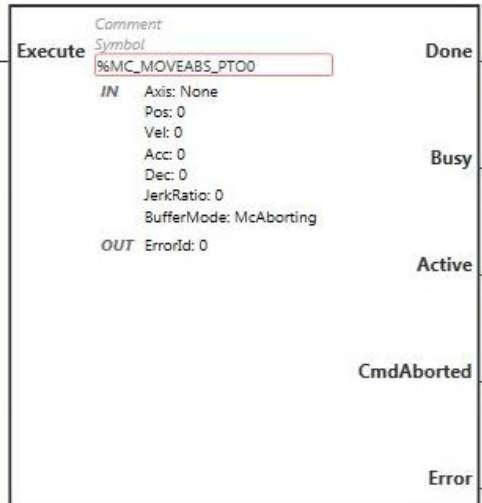


The diagram illustrates a complex profile from **Discrete** state with change of direction:



MC_MoveAbs_PTO Function Block

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
Pos	DINT	0	Position of the axis.

Input Object	Type	Initial Value	Description
Vel	DINT	0	Target velocity. Range Hz: 0...MaxVelocityAppl (<i>see page 147</i>)
Acc	DINT	0	Acceleration in Hz/ms Range (Hz/ms): 1...MaxAccelerationAppl (<i>see page 147</i>)
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl (<i>see page 147</i>)
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile (<i>see page 109</i>). Range: 0...100
BufferMode	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table (<i>see page 146</i>).

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected. When one movement on an axis is interrupted with another movement on the same axis before the commanded action has been completed, <code>CmdAborted</code> is set to TRUE and <code>Done</code> is set to FALSE.
Busy	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as <code>Busy</code> is TRUE.
Active	-	When TRUE, the function block instance has control of the axis. Only one function block at a time can set <code>Active</code> TRUE for the same axis.
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

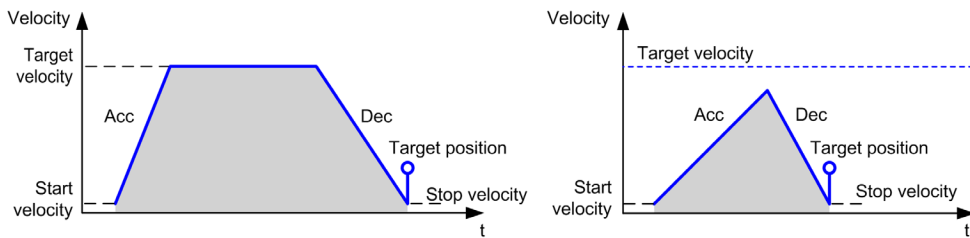
Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (<i>see page 149</i>).

NOTE:

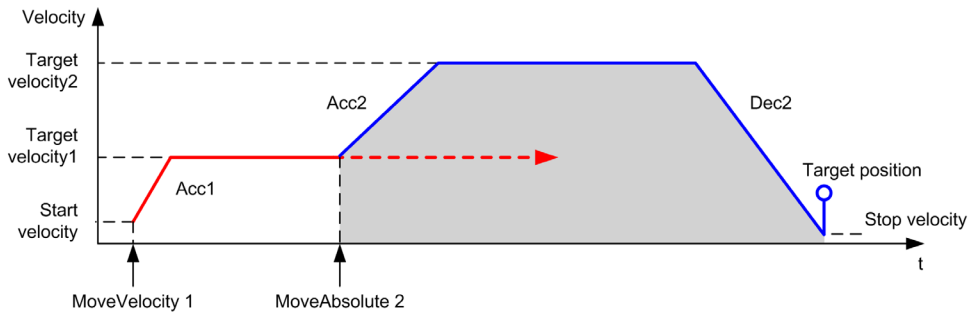
- The function block completes with velocity zero if no further blocks are pending.
- The motion direction is automatically set, according to the current and target positions.
- If the distance is too short for the target velocity to be reached, the movement profile is triangular, rather than trapezoidal.
- If the position cannot be reached with the current direction, the direction reversal is automatically managed. If a motion is ongoing, it is first halted with the deceleration of the MC_MoveAbsolute_PTO function block, and then the motion resumes backward.
- The acceleration/deceleration duration of the segment block must not exceed 80 seconds.

Timing Diagram Example

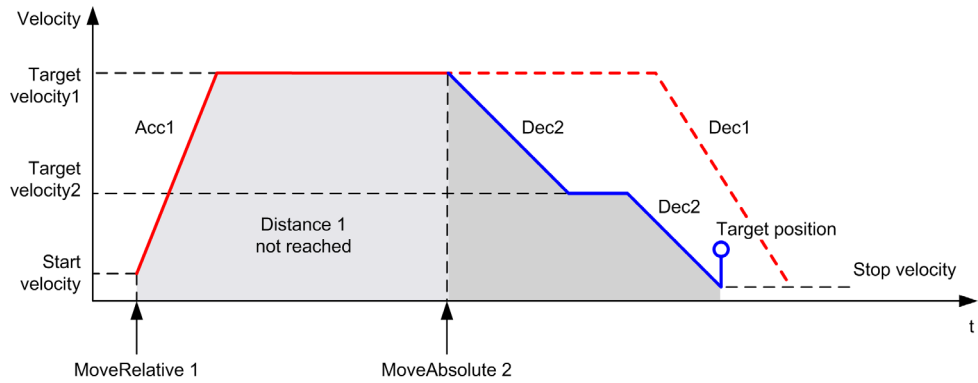
The diagram illustrates a simple profile from **Standstill** state:



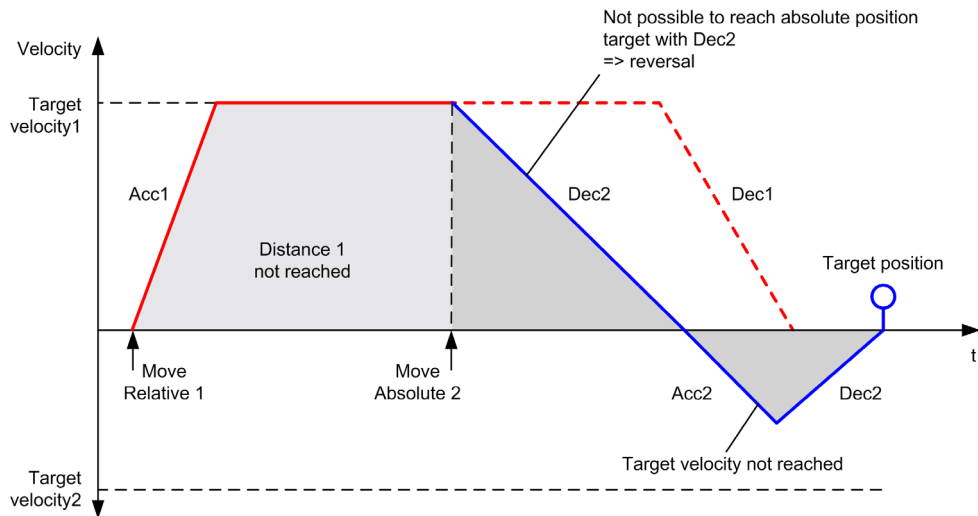
The diagram illustrates a complex profile from **Continuous** state:



The diagram illustrates a complex profile from **Discrete** state:

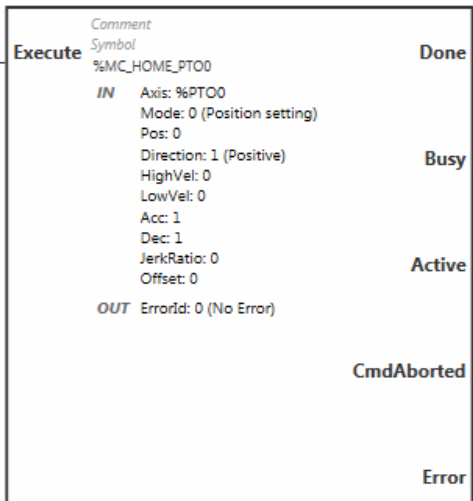


The diagram illustrates a complex profile from **Discrete** state with change of direction:



MC_Home_PTO Function Block

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
Mode	BYTE	0	Predefined homing sequence type (<i>see page 147</i>).
Pos	DINT	0	Position of the axis.

Input Object	Type	Initial Value	Description
HighVel	DINT	0	Target homing velocity for searching the limit or reference switch. Range Hz: 1...MaxVelocityAppl (<i>see page 147</i>)
LowVel	DINT	0	Target homing velocity for searching the reference switch signal. The movement stops when the limit or reference switch is detected. Range Hz: 1...HighVelocity
Acc	DINT	0	Acceleration in Hz/ms Range (Hz/ms): 1...MaxAccelerationAppl (<i>see page 147</i>)
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl (<i>see page 147</i>)
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile (<i>see page 109</i>). Range: 0...100
Direction	INT	mcPositive-Direction	Direction of the movement for PTO type CW/CCW forward (CW) = 1 (mcPositiveDirection) reverse (CCW) = -1 (mcNegativeDirection)
Offset	DINT	0	Distance from origin point. When the origin point is reached, the motion resumes until the distance is covered. Direction depends on the sign (Home offset (<i>see page 145</i>)). Range: -2,147,483,648...2,147,483,647

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected. When one movement on an axis is interrupted with another movement on the same axis before the commanded action has been completed, CmdAborted is set to TRUE and Done is set to FALSE.
Busy	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as Busy is TRUE.
Active	-	When TRUE, the function block instance has control of the axis. Only one function block at a time can set Active TRUE for the same axis.
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table (see page 149).

NOTE: The acceleration/deceleration duration of the segment block must not exceed 80 seconds.

Timing Diagram Example

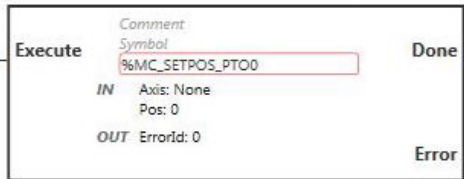
Home modes ([see page 135](#))

MC_SetPos_PTO Function Block

Behavior

This function block modifies the coordinates of the actual position of the axis without any physical movement. It can only be used when the axis is in a `Standstill` state.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
Pos	DINT	0	Position of the axis.

Outputs

This table describes the outputs of the function block:

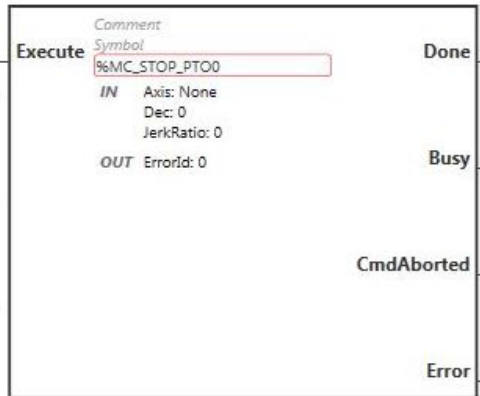
Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table (see page 149).

MC_Stop_PTO Function Block

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl (see page 147)
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile (see page 109). Range: 0...100

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected. When one movement on an axis is interrupted with another movement on the same axis before the commanded action has been completed, <code>CmdAborted</code> is set to TRUE and <code>Done</code> is set to FALSE.
Busy	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as <code>Busy</code> is TRUE.
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

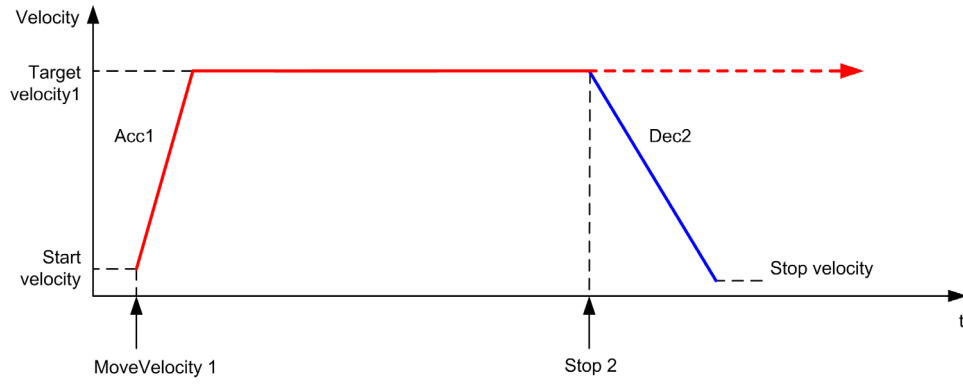
Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

NOTE:

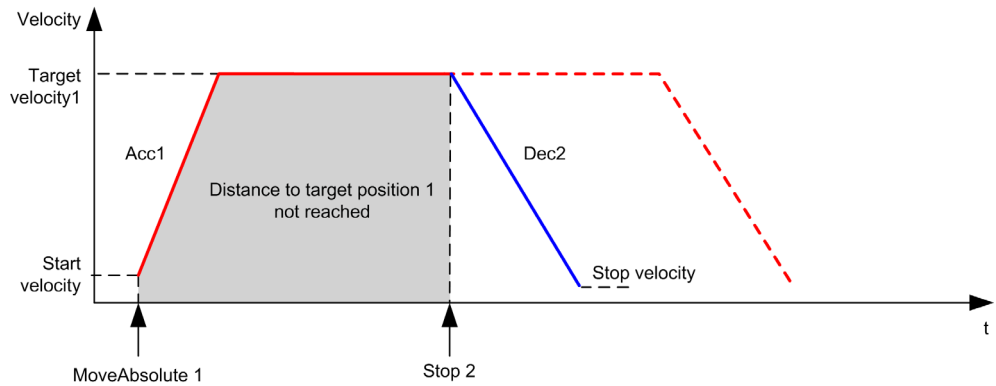
- Calling this function block in state **Standstill** changes the state to **Stopping**, and back to **Standstill** when `Execute` is FALSE.
- The state **Stopping** is kept as long as the input `Execute` is TRUE.
- The `Done` output is set when the stop ramp is finished.
- If `Deceleration` = 0, the fast stop deceleration is used.
- The function block completes with velocity zero.
- The deceleration duration of the segment block must not exceed 80 seconds.

Timing Diagram Example

The diagram illustrates a simple profile from **Continuous** state:



The diagram illustrates a simple profile from **Discrete** state:



MC_HaIt_PTO Function Block

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl (<i>see page 147</i>)
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile (<i>see page 109</i>). Range: 0...100
BufferMode	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table (<i>see page 146</i>).

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected. When one movement on an axis is interrupted with another movement on the same axis before the commanded action has been completed, CmdAborted is set to TRUE and Done is set to FALSE.
Busy	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as Busy is TRUE.
Active	-	When TRUE, the function block instance has control of the axis. Only one function block at a time can set Active TRUE for the same axis.
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

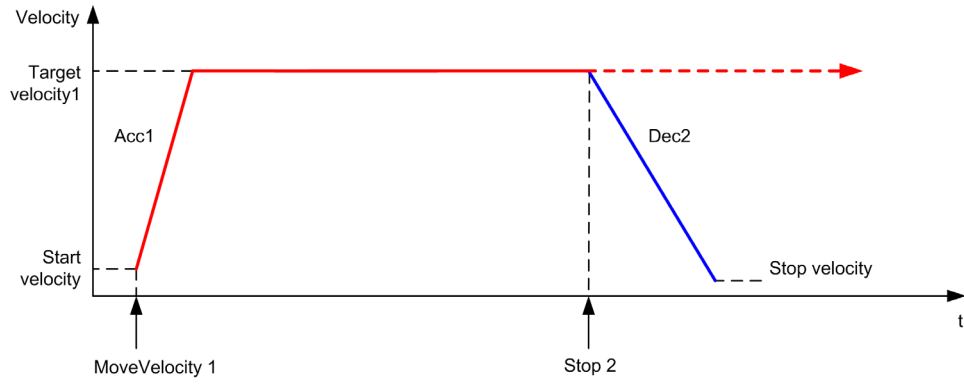
This table describes the output object of the function block:

Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table (<i>see page 149</i>).

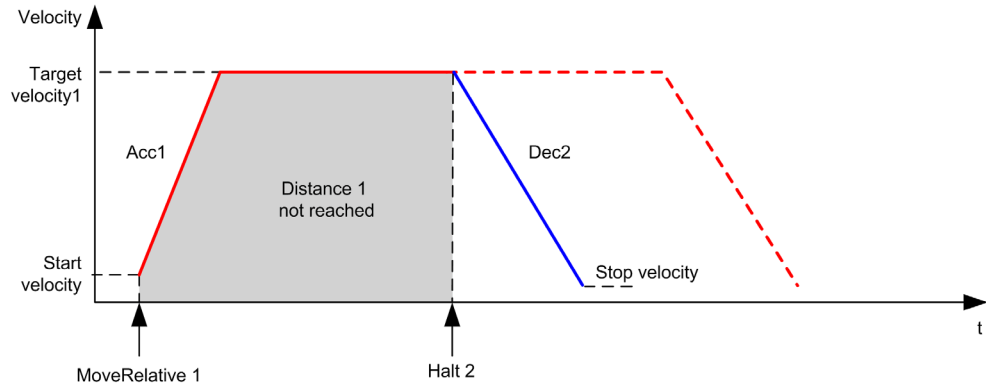
NOTE: The function block completes with velocity zero.

Timing Diagram Example

The diagram illustrates a simple profile from **Continuous** state:



The diagram illustrates a simple profile from **Discrete** state:



Section 7.8

Administrative Function Blocks

Overview

This section describes the **Administrative** function blocks.

What Is in This Section?

This section contains the following topics:

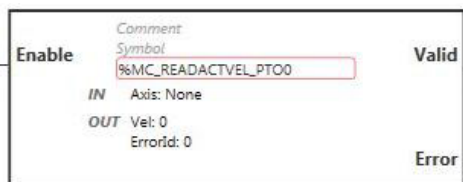
Topic	Page
MC_ReadActVel_PTO Function Block	190
MC_ReadActPos_PTO Function Block	192
MC_ReadSts_PTO Function Block	194
MC_ReadMotionState_PTO Function Block	196
MC_ReadAxisError_PTO Function Block	198
MC_Reset_PTO Function Block	200
MC_TouchProbe_PTO Function Block	202
MC_AbortTrigger_PTO Function Block	204
MC_ReadPar_PTO Function Block	206
MC_WritePar_PTO Function Block	208

MC_ReadActVel_PTO Function Block

Function Description

This function block returns the value of the actual velocity of the axis.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Enable	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Valid	-	If TRUE, the function block object data is valid.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output objects of the function block:

Output Object	Type	Initial Value	Description
Vel	DINT	-	Velocity of the axis.
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

MC_ReadActPos_PTO Function Block

Function Description

This function block returns the value of the actual position of the axis.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Enable	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Valid	-	If TRUE, the function block object data is valid.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output objects of the function block:

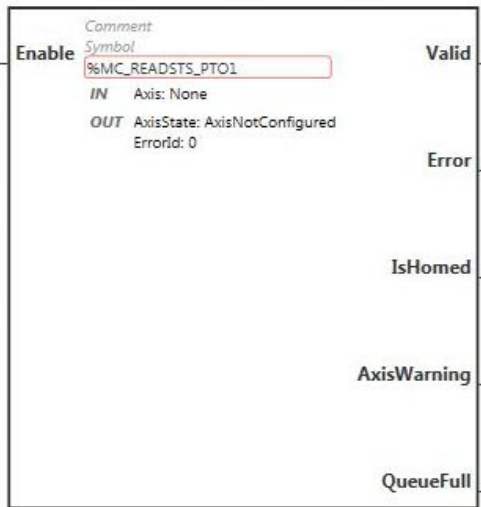
Output Object	Type	Initial Value	Description
Pos	DINT	-	Position of the axis.
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

MC_ReadSts_PTO Function Block

Function Description

This function block returns the state diagram status of the axis.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Enable	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Valid	-	If TRUE, the function block object data is valid.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
IsHomed	FALSE	When TRUE, it indicates that the axis has been homed such that the absolute reference point is valid, and absolute motion commands are allowed.
AxisWarning	FALSE	When TRUE, an alert or an advisory has been provoked by a motion command. Use <code>MC_ReadAxisError_PTO</code> function block to obtain detailed information. (<i>see page 198</i>)
QueueFull	FALSE	When TRUE, the motion queue is full and no additional buffered motion commands are allowed.

This table describes the output objects of the function block:

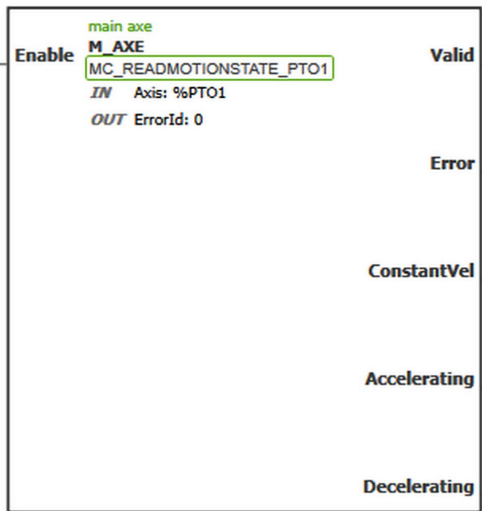
Output Object	Type	Initial Value	Description
AxisState	-	-	Code for the state of the axis: 0 = axis not configured 1 = ErrorStop 2 = Disabled 4 = Stopping 8 = Homing 16 = Standstill 32 = Discrete motion 64 = Continuous motion For more information, refer to the States description table (<i>see page 152</i>).
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (<i>see page 149</i>).

MC_ReadMotionState_PTO Function Block

Function Description

This function block returns the actual motion status of the axis.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Enable	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Valid	-	If TRUE, the function block object data is valid.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
ConstantVel	-	When TRUE, the velocity of the axis is constant.
Accelerating	-	When TRUE, the velocity of the axis is increasing.
Decelerating	-	When TRUE, the velocity of the axis is decreasing.

This table describes the output object of the function block:

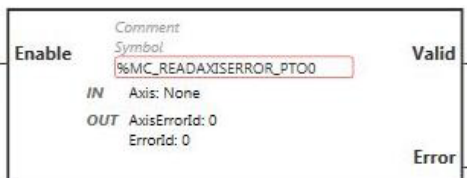
Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table (see page 149).

MC_ReadAxisError_PTO Function Block

Function Description

This function block retrieves the axis control error. If no axis control error is pending, the function block returns `AxisErrorId = 0`.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Enable	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Valid	-	If TRUE, the function block object data is valid.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output objects of the function block:

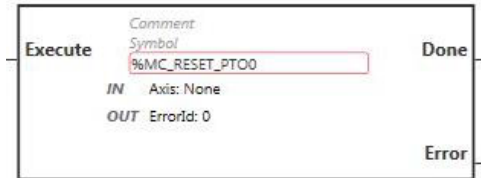
Output Object	Type	Initial Value	Description
AxisErrorId	-	-	Axis error codes, valid when <code>AxisWarning</code> output is TRUE. Refer to PTO axis error code table (<i>see page 148</i>).
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (<i>see page 149</i>).

MC_Reset_PTO Function Block

Behavior

This function block resets all axis-related errors, conditions permitting, to allow a transition from the states **ErrorStop** to **Standstill**. It does not affect the output of the function blocks instances.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

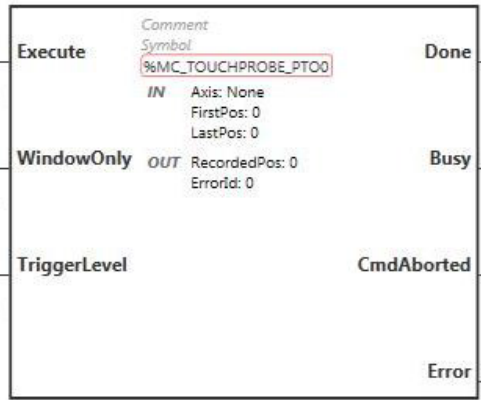
Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table (see page 149).

MC_TouchProbe_PTO Function Block

Function Description

This function block is used to activate a trigger event on the probe input. This trigger event allows to record the axis position, and/or to start a buffered move.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click *Apply*.

Inputs

This table describes the inputs of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <i>Execute</i> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates. If a second rising edge is detected during the execution of the function block, the current execution is aborted and the function block is executed again.
WindowOnly	FALSE	When TRUE, a trigger event is only recognized within the position range (window) defined by <i>FirstPosition</i> and <i>LastPosition</i> .
TriggerLevel	FALSE	When TRUE, position captured or event triggered at rising edge. When FALSE, position captured or event triggered at falling edge.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
FirstPos	DINT	0	Start of the absolute position from where trigger events are accepted (value included in enable window).
LastPos	DINT	0	End of the absolute position from which trigger events are accepted (value included in enable window).

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected. When one movement on an axis is interrupted with another movement on the same axis before the commanded action has been completed, <code>CmdAborted</code> is set to TRUE and <code>Done</code> is set to FALSE.
Busy	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as <code>Busy</code> is TRUE.
CmdAborted	-	When TRUE, function block execution is terminated due to another motion command.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output objects of the function block:

Output Object	Type	Initial Value	Description
RecordedPos	-	-	Position where trigger event was detected.
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

NOTE:

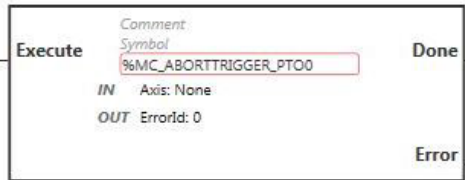
- Only one instance of this function block is allowed on the same axis.
- Only the first event after the rising edge at the `MC_TouchProbe_PTO` function block `Busy` output is valid. Once the `Done` output is set to TRUE, subsequent events are ignored. The function block needs to be reactivated to respond to other events.

MC_AbortTrigger_PTO Function Block

Function Description

This function block is used to abort function blocks which are connected to trigger events (for example, MC_TouchProbe_PTO).

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

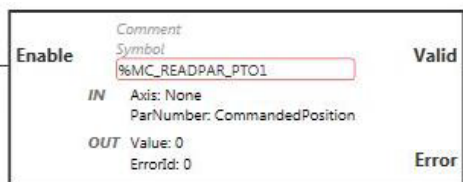
Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

MC_ReadPar_PTO Function Block

Function Description

This function block is used to get parameters from the PTO.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Enable	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
ParNumber	DINT	0	Code for the parameter you wish to read or write. For more information, refer to PTO Parameter table (see page 147).

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Valid	-	If TRUE, the function block object data is valid.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output objects of the function block:

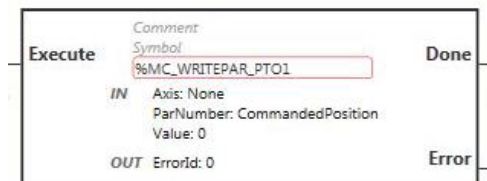
Output Object	Type	Initial Value	Description
Value	DINT	0	Value of the requested parameter.
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

MC_WritePar_PTO Function Block

Function Description

This function block is used to write parameters to the PTO.

Graphical Representation



NOTE: When you first enter the function block, you must configure it to use the intended axis. Double-click on the function block to display the function block properties, choose the axis and click `Apply`.

Inputs

This table describes the input of the function block:

Input	Initial Value	Description
Execute	FALSE	On rising edge, starts the function block execution. The values of the other function block inputs control the execution of the function block on the rising edge of <code>Execute</code> . A subsequent change in these input parameters does not affect the ongoing execution. The outputs are set when the function block terminates.

This table describes the input objects of the function block:

Input Object	Type	Initial Value	Description
Axis	PTOx	-	Instance for which the function block is to be executed. The name is declared in the controller configuration.
ParNumber	DINT	0	Code for the parameter you wish to read or write. For more information, refer to PTO Parameter table (see page 147).
Value	DINT	0	Value to be written to the parameter chosen with the <code>ParNumber</code> input object.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	FALSE	When TRUE, function block execution is finished with no error detected.
Error	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Motion command error codes, valid when <code>Error</code> output is TRUE. Refer to PTO motion command error code table (see page 149).

Chapter 8

Frequency Generator (%FREQGEN)


What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	212
Configuration	214

Description

Introduction

The frequency generator `FREQGEN` function block  commands a square wave signal output at a specified frequency.

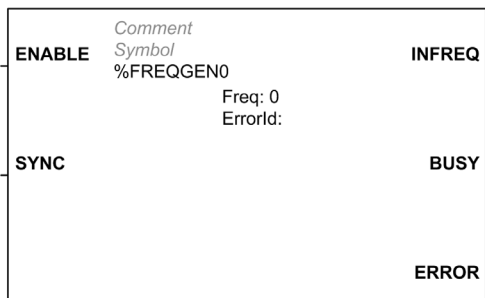
The frequency is configurable from 0 Hz to 100 kHz with a 1 Hz step.

The `FREQGEN` function has the following characteristics:

Characteristic	Value
Number of channels	2 or 4, depending on the reference
Minimum frequency	1 Hz
Maximum frequency	10000 Hz
Accuracy on frequency	1 %

Illustration

This illustration is a `FREQGEN` function block:



Inputs

This table describes the inputs of the function block:

Input	Initial Value	Description
ENABLE	FALSE	When TRUE, the function block is executed. The values of the other function block inputs can be modified continuously, and the function block outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.
SYNC	FALSE	When a rising edge is detected, the target frequency is emitted without waiting for the end of the ongoing period output.

This table describes the input object of the function block:

Input Object	Type	Initial Value	Description
Freq	DWORD	-	Frequency of the <code>Frequency Generator</code> output signal in Hz. Specify the frequency in the Pulse Generators properties (<i>see page 214</i>) table (Range: minimum 0 (0 Hz)...maximum 100000 (100 kHz))

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
INFREQ	-	If TRUE, the frequency generator signal is being output at the frequency specified in the <code>Freq</code> input object.
BUSY	-	When TRUE, function block execution is in progress. When FALSE, execution of the function block has been terminated. The function block must be kept in an active task of the application program for at least as long as <code>BUSY</code> is TRUE.
ERROR	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.

This table describes the output object of the function block:

Output Object	Type	Initial Value	Description
ErrorId	Word	NoError	Error codes, valid when <code>ERROR</code> output is TRUE. Refer to the ErrorId Error Codes table (<i>see page 213</i>).

ErrorId Error Codes

This table lists the values for the function block error codes

Name	Value	Description
NoError	0	No errors detected.
OutputProtection	1007	One or more PTO objects have digital output protection active. Refer to system objects %S10 and %SW139 (<i>see Modicon M221, Logic Controller, Programming Guide</i>) for more details.
InvalidFrequencyValue	3002	The frequency <code>Freq</code> input object is outside the allowed range.

Configuration

Overview

To configure the `Pulse Generator` resource, refer to *Configuring Pulse Generators (see Modicon M221, Logic Controller, Programming Guide)*.

To configure the `Pulse Generator` resource as a `FREQGEN`, refer to *Configuring Frequency Generator (see Modicon M221, Logic Controller, Programming Guide)*

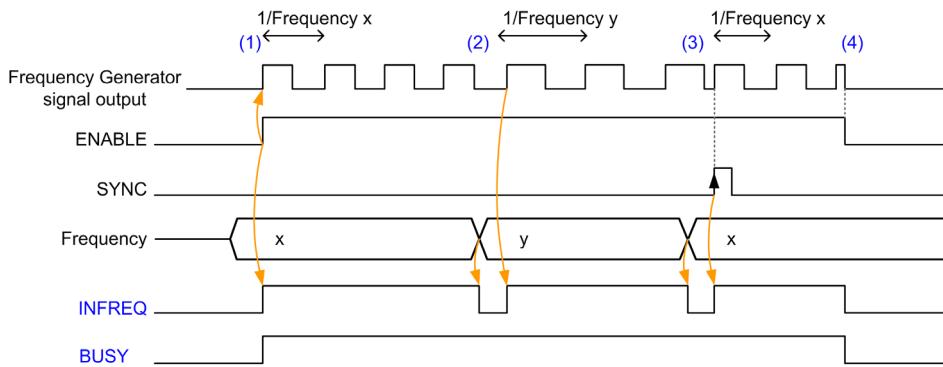
Properties

The `FREQGEN` function block has the following properties:

Property	Description	Value
Used	Address used	If selected, this address is in use in a program.
Address	<code>%FREQGENi</code> Frequency generator address	The instance identifier, where <i>i</i> is from 0 to the number of objects available on this logic controller. For the maximum number of <code>FREQGEN</code> objects, refer to the table <i>Maximum Number of Objects (see Modicon M221, Logic Controller, Programming Guide)</i> .
Symbol	Symbol	The symbol associated with this object. For details, refer to <i>Defining and Using Symbols (see SoMachine Basic, Operating Guide)</i> .
Freq	Frequency	The frequency of the frequency generator output signal in Hz. Minimum value: 0 (0 Hz). Maximum value: 100000 (100 kHz) The default value is 0.
Comment	Comment	An optional comment can be associated with this object. Double-click in the Comment column and type a comment.

Timing Diagram

This diagram displays the timing for the `FREQGEN` function block:



- (1) The `ENABLE` input is set to 1. The frequency generator signal is generated at the dedicated output. The `INFREQ` output is set to 1. The `BUSY` output is set to 1.

- (2) The frequency value is changed. The `INFREQ` output is set to 0 until the new frequency is being generated at the dedicated output. The `BUSY` output remains set to 1.
- (3) The `SYNC` input is set to 1. The current frequency generator cycle stops and a new cycle starts. The `INFREQ` output is set to 1. The `BUSY` output remains set to 1.
- (4) The `ENABLE` input is set to 0. Frequency generation stops. The `INFREQ` output is set to 0. The `BUSY` output is set to 0.

When the application is stopped, frequency generation stops without waiting for the end of the pulse generation cycle. The `Error` output remains at `FALSE`.

If an error is detected, it is automatically acknowledged when leaving the error condition.

Part IV

Advanced Software Functions

Chapter 9

PID Function

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	PID Operating Modes	220
9.2	PID Auto-Tuning Configuration	222
9.3	PID Standard Configuration	226
9.4	PID Assistant	238
9.5	PID Programming	251

Section 9.1

PID Operating Modes

PID Operating Modes

Introduction

The SoMachine Basic PID controller offers 4 distinct operating modes, configurable in the **General** tab (*see page 241*) of the **PID Assistant** in SoMachine Basic.

The PID operating modes are:

- PID mode
- AT + PID mode
- AT mode
- Word address

PID Mode

The simple PID controller mode is active by default when the PID controller starts up. The gain values Kp, Ti, and Td to be specified in the **PID** tab (*see page 245*) must be known in advance to successfully control the process. You can choose the corrector type of the controller (PID or PI) in the **PID** tab of the **PID Assistant** screen (*see page 238*). If the PI corrector type is selected, the derivative time Td field is disabled.

Using PID mode, the Auto-Tuning function is disabled and the **AT** tab (*see page 247*) of the **Assistant Configuration** screen is therefore unavailable.

AT + PID Mode

In this mode, the Auto-Tuning function is active when the PID controller starts up. The Auto-Tuning function then calculates the gain values Kp, Ti, and Td (*see page 245*) and the type of PID action (*see page 249*). At the end of the Auto-Tuning sequence, the controller switches to PID mode for the adjusted setpoint, using the parameters calculated by Auto-Tuning.

If the Auto-Tuning algorithm detects an error (*see page 255*):

- No PID parameter is calculated.
- The Auto-Tuning output is set to the output that was applied to the process before starting Auto-Tuning.
- An error message appears in the **List of PID States** drop-down list.
- The PID control is cancelled.

While in AT + PID mode, the transition from Auto-Tuning to PID mode is automatic and seamless.

AT Mode

In this mode, the Auto-Tuning function is active when the PID controller starts up and automatically calculates both the gain values Kp, Ti, and Td (*see page 245*) and the type of PID action (*see page 249*). After convergence of the Auto-Tuning process and successful completion with the determination of the Kp, Ti, and Td parameters and the type of PID action (*see page 249*) (or after detection of an error in the Auto-Tuning algorithm), the Auto-Tuning numerical output is set to 0 and the **Auto-Tuning Complete** message appears in the List of PID States (*see page 255*) dropdown. The PID controller then stops and waits. The calculated Kp, Ti, and Td PID coefficients are available in their respective memory words (%MWx).

Word Address

This PID mode is selected by assigning the desired value to the word address associated with this selection:

- %MWxx = 0: The controller is disabled.
- %MWxx = 1: The controller operates in simple PID mode.
- %MWxx = 2: The controller operates in AT+ PID mode.
- %MWxx = 3: The controller operates in AT mode only.
- %MWxx = 4: The controller operates in simple PID mode, with PI corrector type.

This mode word address enables you to manage the PID controller operating mode with the application, thus making it possible to adapt to your requirements.

Section 9.2

PID Auto-Tuning Configuration

PID Auto-Tuning Configuration

Introduction

This section guides you through all the steps necessary to configure the SoMachine Basic PID controller using Auto-tuning (AT).

This section contains the following steps:

Step	Topic
1	Configuring analog channel (<i>see page 222</i>)
2	Pre-requisites for PID configuration (<i>see page 222</i>)
3	Configuring the PID (<i>see page 222</i>)
4	Control set-up (<i>see page 224</i>)

Step 1: Configuring the Analog Channel

A PID controller uses an analog feedback signal (known as the process value) to calculate the algorithm used to control the process. The logic controller has an embedded analog input that can be used to acquire this process value. Refer to the M221 Logic Controller - Programming Guide for more details about analog input configuration.

If an analog output is being used to drive the system to be controlled, make sure that this analog output is correctly configured. Refer to the analog output expansion module of your logic controller.

Step 2: Pre-requisites for PID Configuration

Before configuring the PID controller, ensure that the following phases have been performed:

Phase	Description
1	PID is enabled in the program (<i>see page 252</i>).
2	Scan Mode is set to periodic (<i>see page 254</i>).

Step 3: Configuring the PID

Use a solid state output in conjunction with the PID function. Using a relay output may result in quickly exceeding its life cycle limits resulting in an inoperative relay with contacts either frozen open or soldered closed.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- Do not use relay outputs in conjunction with the PID function.
- Only use solid state outputs if a digital output is required to drive the system to be controlled.


Failure to follow these instructions can result in death, serious injury, or equipment damage.

To implement a PID controller with Auto-Tuning, perform the following steps:

Step	Action
1	In the General tab (<i>see page 241</i>) of the PID Assistant screen (in offline mode), select AT+PID (or AT) or select Word Address setting the associated word to 2 or 3, from the Operating Modes (<i>see page 220</i>).
2	Activate the PID States checkbox and enter the address of the memory word in the field.
3	In the Input tab (<i>see page 244</i>), enter the address of the analog input used as a measurement.
4	If Conversion or Alarms are required, refer to Input tab (<i>see page 244</i>) of PID Assistant screen.
5	In the PID tab (<i>see page 245</i>), enter the value of the setpoint. In general, this value is a memory address or an analog input.
6	Corrector type in the PID tab must be set to PID or PI .
7	Set the Parameters in the PID tab: Kp (x0,01) , Ti (x0,1s) , and Td (x0,1s) . When AT+PID or AT are the Operating modes (<i>see page 220</i>), the parameters should be memory words addresses (%MWxx) so the Auto-Tuning algorithm fills in the computed value of the parameters.
8	Enter the PID Sampling period (Ts (<i>see page 230</i>)) in the PID tab. The Sampling period is a key parameter and must be carefully determined.
9	In the AT tab, the AT Mode must be set to Authorize by default. Enter the Min. and Max. values if the Measurement Range is activated (Authorize checkbox). Select the Dynamic AT corrector from the list that contains Fast , Medium , Slow , or Word address corrector type. For further details, refer to the AT tab in PID Assistant (<i>see page 238</i>).
10	In the AT tab, enter the AT Trigger memory bit to store the value of the step change during Auto-Tuning. For further details, refer to the AT tab in PID Assistant (<i>see page 238</i>).
11	In the Output tab (<i>see page 249</i>), set the Action to Bit Address from the list. Enter the memory bit address in the Bit field. Limits can be configured if necessary from Output tab (<i>see page 249</i>). In Analog output field, set the address of the word: an analog output or a memory word. Set the Output PWM (<i>see page 249</i>) to Authorize . In the manual mode, enter the value in the Period (0.1 s) field or the memory word address of the output in the Output field. For more details about manual mode operation, refer to Output tab (<i>see page 249</i>).
12	Click OK to confirm the PID controller configuration.

Step 4: Control Setup

Use a solid state output in conjunction with the PID function. Using a relay output may result in quickly exceeding its life cycle limits resulting in an inoperative relay with contacts either frozen open or soldered closed.

 WARNING
UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT
<ul style="list-style-type: none"> ● Do not use relay outputs in conjunction with the PID function. ● Only use solid state outputs if a digital output is required to drive the system to be controlled.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

To start operation in AT+PID operating mode (*see page 220*) perform the following steps:

Step	Action
1	Connect the PC to the controller and transfer the application.
2	Switch the controller to RUNNING state.

NOTE: Before switching the controller to RUNNING state, verify that the operating conditions of the machine allow the RUNNING state for the rest of the application.

Step	Action
1	Create an animation table containing the objects defined during configuration. Refer to the <i>SoMachine Basic Operating Guide</i> for further details about animation table creation.
2	<p>Verify the consistency of the process value and application's values. This test is important as successful operation of the PID controller depends on the accuracy of the measurement. If you have any doubt about the accuracy of the measurement, set the logic controller to the STOP state and verify the wiring of analog channels.</p> <p>If the actuator is not controlled:</p> <ul style="list-style-type: none"> ● For analog output verify the output voltage or current from analog channel. ● For PWM output, verify that the: <ul style="list-style-type: none"> ○ LED of the dedicated output is lit ○ wiring of the supplies and 0V circuit ○ actuator power supply is being applied
3	<p>In the animation table, verify that:</p> <ul style="list-style-type: none"> ● Output mode is set to automatic. ● All parameters your application requires are set to the appropriate values.

Step	Action
4	Set the logic controller scan period so that the Sampling period (Ts) value of the PID controller is an exact multiple of the scan period. For further details on how to determine the Sampling period, refer to Tuning PID (<i>see page 230</i>).
5	When the Auto-Tuning sequence is complete, the parameters Kp , Ti , and Td are written in to the RAM memory of the logic controller. The values are saved for as long as the application is valid (power-down less than 30 days) and no cold-start is performed.

The Auto-Tuning process is repeated each time a rising edge is detected on the **AT trigger** memory bit.

NOTE: When the PID autotuning is in the process of calibration to find the new parameters for **Kp**, **Ti**, **Td** and the manual output control is activated, launch PID autotuning again after finishing manual output control so that the parameters are updated.

Section 9.3

PID Standard Configuration

What Is in This Section?

This section contains the following topics:

Topic	Page
PID Word Address Configuration	227
PID Tuning with Auto-Tuning (AT)	230
Manual Mode	233
Determining the Sampling Period (Ts)	235

PID Word Address Configuration

Introduction

This section guides you through all the steps required to configure the SoMachine Basic PID controller using word address operating mode (*see page 220*). This mode provides greater flexibility of use than the other PID modes.

This section contains the following steps:

Step	Topic
1	Prerequisites for PID configuration (<i>see page 227</i>)
2	Configuring the PID (<i>see page 227</i>)
3	Control set-up (<i>see page 228</i>)

Step 1: Prerequisites for PID Configuration

Before configuring the PID, ensure that the following phases have been performed:

Phase	Description
1	An analog input is configured as well as an analog output if required. Refer to the M221 Logic Controller - Programming Guide.
2	PID is enabled in the program (<i>see page 252</i>).
3	Scan mode is set to periodic (<i>see page 254</i>).

Step 2: Configuring the PID

Use a solid state output in conjunction with the PID function. Using a relay output may result in quickly exceeding its life cycle limits resulting in an inoperative relay with contacts either frozen open or soldered closed.

WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- Do not use relay outputs in conjunction with the PID function.
- Only use solid state outputs if a digital output is required to drive the system to be controlled.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The following steps explain how to implement a PID controller in **word address** mode. For more details on how to configure the PID, refer to the PID Assistant section ([see page 238](#)).

For the dynamic modification of the PID parameters (in offline and in online mode), enter the memory addresses in the associated fields, thus avoiding the need to switch to offline mode to make on-the-fly changes to values.

Step	Action
1	In the General tab of the PID Assistant screen (in offline mode), in the Operating Modes ; drop-down list select Word address . Check the box associated to PID States and enter the address of the memory word in the field.
2	In the Input tab (see page 244), enter the address of the analog input used as a measurement. If Conversion or Alarms are required, refer to Input tab (see page 244) of PID Assistant (see page 238).
3	In the PID tab, enter the value of the Setpoint . In general, this value is a memory address or an analog input. The Parameters (Kp, Ti, and Td) should be memory words addresses (%MWxx) . Enter the PID Sampling period (Ts (see page 245)) in the PID tab (see page 245). This parameter can also be a memory word (the value can then be set using the animation table). In Word Address operating mode, the Corrector type is set to Auto and greyed out (it cannot be modified manually).
4	In the AT tab, the AT mode should be checked to Authorize . Enter the Dynamic corrector and the AT Trigger . For further details, refer to AT tab (see page 247) in PID Assistant screen.
5	In the Output tab, Action should be set to Bit Address . Enter a memory bit address . Limits can be configured if necessary from the Output tab (see page 249). In Analog output field set the address of the word: an analog output or a memory word. If required, set the Output PWM , refer to Output tab (see page 249) in PID Assistant (see page 238).
6	Click OK to confirm the PID controller configuration.

Step 3: Verifying the Setup

Step	Action
1	Connect the PC to the logic controller and transfer the application.
2	Switch the logic controller to RUNNING state.

NOTE: Before switching the logic controller to RUNNING state, verify that the operating conditions of the machine allow RUNNING state for the rest of the application. The procedure remains the same as the one used in **AT** and **AT+PID** operating modes. The word address configuration allows you to modify the PID operating modes by software. In the case of the PID mode, the procedure is significantly simplified, assuming the parameters (Kp, Ti, Td, and Ts) are known and there is no need to perform Auto-Tuning.

This table gives the generic procedure to set up the PID controller

Step	Action
1	Create an animation table containing the objects defined during configuration. Refer to the <i>SoMachine Basic Operating Guide</i> for details.
2	<p>Verify the consistency of the process value and other values defined in the animation table. If you have any doubt about the accuracy of the measurement, set the logic controller to STOP and verify the wiring of analog channels.</p> <p>If you see that the actuator is not being controlled:</p> <ul style="list-style-type: none"> ● For analog output, verify the output voltage or current from analog channel. ● For PWM output, verify that the: <ul style="list-style-type: none"> ○ LED of dedicated output is lit ○ wiring of the supplies and 0 V circuit is correct ○ actuator power supply is being applied
3	Set the logic controller scan period so that the Sampling period (Ts) of the PID controller is an exact multiple of the scan period. For further details on Sampling period, please refer to Determining Sampling Period (<i>see page 235</i>)
4	If you plan to use the Auto-Tuning (<i>see page 230</i>) function, you may need to run Manual Mode (<i>see page 233</i>) to know the Dynamic corrector and the AT Trigger defined in the AT tab (<i>see page 247</i>) of the PID Assistant .
5	<p>Power up the loop controller using the animation table:</p> <ul style="list-style-type: none"> ● Set the operating mode (<i>see page 220</i>). ● Enable the PID controller (<i>see page 254</i>). ● Set the values defined during configuration (<i>see page 227</i>) to appropriate values depending on the selected operating mode.

PID Tuning with Auto-Tuning (AT)

Introduction

The Auto-Tuning mode allows automatic tuning of the Kp, Ti, Td, and action parameters to achieve refined convergence of the PID function. The Auto-Tuning function provided by SoMachine Basic is particularly suited for automatic tuning of thermal processes.

This section contains the following topics:

- Auto-Tuning requirements
- Description of Auto-Tuning process
- Storage of Calculated Coefficients
- Adjusting PID parameters
- Repetition of Auto-Tuning
- Limitations on using the Auto-Tuning and the PID control

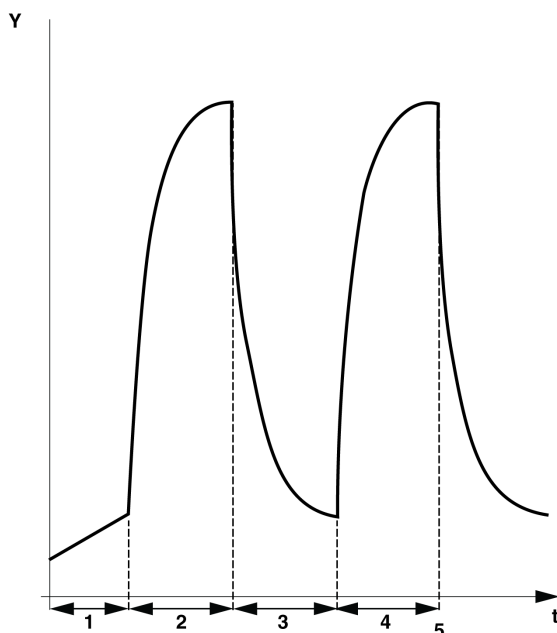
Auto-Tuning Requirements

When using the Auto-Tuning function, make sure that the control process and the logic controller meet the following requirements:

- Process requirements:
 - The process must be a stable open-loop system.
 - The process must be mostly linear over the entire operating range.
 - The process response to a change in level of the analog output follows a transient asymptotic pattern.
 - The process is in a steady state with a null input at the start of the Auto-Tuning sequence.
 - The process must be free of disturbances throughout the entire process. Otherwise, either calculated parameters will be incorrect or the Auto-Tuning process will not operate correctly.
- Configuration requirements:
 - Configure the logic controller to periodic scan mode to ensure a correct run of the Auto-Tuning function.
 - Only use the Auto-Tuning function when no other PID controllers are running.
 - Configure the Kp, Ti, and Td coefficients as memory word addresses (%MWxx).
 - Set the Action type in the **Output** tab to a memory bit address (%Mxx).

Description of Auto-Tuning Process

The Auto-Tuning process is divided into 4 consecutive phases. All phases of the process must be fulfilled in order to bring the Auto-Tuning to a successful conclusion. The following process response curve and table describe the 4 phases of the SoMachine Basic PID Auto-Tuning function:



The Auto-Tuning phases are described in the following table:

Auto-Tuning Phase	Description
1	The stabilization phase ⁽¹⁾ starts when you launch the Auto-Tuning process. During this phase, the Auto-Tuning function performs checks to ensure that the process value is in steady state.
2	The first step change is applied to the process. It produces a process step-response similar to the one shown in the above figure.
3	The relaxation phase ⁽¹⁾ starts when the first step-response has stabilized.
4	The second step change is applied to the process in the same amount and manner as in Phase 2 described above. The Auto-Tuning process ends. The process value is restored ⁽¹⁾ and the Auto-Tuning parameters are calculated and stored in their respective memory words. Refer to Storage of Calculated Coefficients description (see page 232).
(1) The output last applied to the process before start of the Auto-Tuning is used as both the starting point and the relaxation point for the Auto-Tuning process.	

NOTE: The Kp, Ti and Td parameters cannot be found if the manual output control is activated while the Auto-Tuning is in calibration process. The Auto-Tuning calibration process needs to be launched again once the output manual control is finished.

Storage of Calculated Coefficients

After the Auto-Tuning sequence is complete, the memory words assigned to the Kp, Ti, and Td coefficients and the action type are set using the calculated values. These values are written in to the RAM memory and saved in the logic controller as long as the application is valid and no cold start is performed (%S0).

If the system is not influenced by outside disturbances, the calculated values may be written in to the settings of the PID controller (refer to the **PID** tab of the PID Assistant (*see page 249*)). In this way, the PID controller operating mode can be set to PID mode.

Adjusting PID Parameters

The Auto-Tuning method may provide a very dynamic command, leading to unwanted overshoots during step change of setpoints. To refine the process regulation provided by the PID parameters (Kp, Ti, Td) obtained from Auto-Tuning, you also have the ability to adjust these parameter values manually, directly from the **PID** tab of the **PID Assistant** screen or through the corresponding memory words (%MW). For more details on manual parameters adjustments, refer to the appendices (*see page 261*).

Repetition of Auto-Tuning

In the **AT** tab, the **AT Trigger** enables the repetition of Auto-Tuning sequence. The auto-tuning process is launched at each rising edge of the signal linked to **AT Trigger**.

Limitations on Using Auto-Tuning

Thermal processes can often be assimilated to the first order with pure delay model. There are two key parameters that describe this type of model:

- the time constant, τ
- the delay time, θ

Auto-Tuning is best suited for processes in which the time constant (τ) and delay time (θ) meet the following criteria:

- $10 \text{ s} < (\tau + \theta) < 2700 \text{ s}$ (i.e.: 45 min)
- $2 < \tau / \theta < 20$

Manual Mode

Introduction

The manual mode is accessible through the **PID Assistant** screen (**Output** tab (*see page 249*)). This mode allows you to bypass orders from the PID. There are 2 main objectives using Manual mode:

- Initialize the set-up
- Determine the sampling period.

Description

The manual mode lets you specify the **Output** value (*see page 249*). This operation can be particularly well suited for testing the system response.

Setting the **bit address** from the **Output** tab (*see page 249*) to 1 activates the manual mode. If Allow is set, then the manual mode is the only accessible mode.

Application

When the manual mode is active the output is assigned a fixed value that you set. This output value is from 0 to 10,000 (0 to 100% for PWM output).

You can also use manual mode to make trials to determine the minimum/maximum output limitation.

Manual mode is also required to use the process curve response method (*see page 235*) that helps to find the correct sampling time (Ts).

Start the Manual Mode

Before starting manual mode, you should make sure that the logic controller RUN/STOP switch is in the RUN position.

To start manual mode using an animation table:

Step	Description
1	Enable manual mode by setting the dedicated memory bit to 1. For more details refer to the Output tab (<i>see page 249</i>).
2	If using PWM, set the PWM period to the desired value.
3	Set the memory word associated with the Operating mode in the General tab (<i>see page 241</i>) of the PID Assistant to 1 (PID mode). For more details on operating modes using word address refer to the operating mode description (<i>see page 220</i>).
4	Set the memory word associated with the manual output in the Output tab (<i>see page 249</i>) to the desired value. This manual setpoint value can be selected several times on condition that the system is left in its initial state.
5	Enable the loop controller (<i>see page 227</i>).

Stop the Manual Mode

To stop manual mode using an animation table:

Step	Description
1	Disable the loop controller (<i>see page 227</i>).
2	Inhibit the manual mode by setting the dedicated memory bit to 0. For more details refer to the Output tab (<i>see page 249</i>).
3	Set the memory word associated with the Operating mode in the General tab (<i>see page 241</i>) for the PID controller to 0. For more details on operating modes using word address, refer to the operating mode description (<i>see page 220</i>).
4	Set the memory word associated to the manual output in the Output tab (<i>see page 249</i>) to 0.

Determining the Sampling Period (Ts)

Introduction

The Sampling Period (Ts) is the key parameter for PID regulation. The Sampling Period (Ts) should be carefully set in the **PID** tab (*see page 245*) of the **PID Assistant** screen. This parameter is highly correlated with the time constant (τ) of the process to control.

This section describes the use of online mode and two methods to determine the sampling period (Ts) are described in this section:

- Process response curve method,
- Trial-and-error method.

Process Response Curve Method

This method is an open loop process that aims to determine the time constant of the process to be controlled. First, it is necessary to ensure that the process can be described by a first order with time delay model. The principle is quite simple: apply a step change at the input of the process while recording the process output curve. Then use a graphical method to determine the time delay of the process.

To determine the sampling period (Ts) using the process response curve method:

Step	Action
1	It is assumed that you have already configured the various settings in the General , Input , PID , AT and Output tabs of the PID.
2	Select the Output tab (<i>see page 249</i>) from the PID Assistant screen.
3	Select Allow or Address bit from the Manual Mode drop-down list to authorize manual output.
4	Set the Output field to a high level (in the [5,000...10,000] range).
5	Download your application to the logic controller. For further details on how to download an application refer to the <i>SoMachine Basic Operating Guide</i> .
6	Run the PID and check the response curve rise.
7	When the response curve has reached a steady state, stop the PID measurement.
8	Use the following graphical method to determine the time constant (τ) of the control process: <ol style="list-style-type: none"> 1. Calculate the process value output at 63% rise ($S_{[63\%]}$) by using the following formula: $S_{[63\%]} = S_{[initial]} + (S_{[final]} - S_{[initial]}) \times 63\%$ 2. Calculate graphically the time abscissa ($t_{[63\%]}$) that corresponds to $S(63\%)$. 3. Calculate graphically the initial time ($t_{[initial]}$) that corresponds the start of the process response rise. 4. Compute the time constant (τ) of the control process by using the following relationship: $\tau = t_{[63\%]} - t_{[initial]}$
(1) The base unit for the sampling period is 10ms. Therefore, you should round up/down the value of Ts to the nearest 10ms.	
(2) You must choose "n" so that the resulting Scan Period is a positive integer in the range [2...150] ms.	

Step	Action
9	Calculate the sampling period (Ts) ⁽¹⁾ based on the value of (τ) that you determined in the previous step, using the following rule: $T_s = \tau/75$
10	Set the Scan period of the Periodic scan mode so that the Sampling Period (Ts) is an exact multiple of the scan period: $\text{Scan Period} = T_s / n$, where n is a positive integer ⁽²⁾
<p>(1) The base unit for the sampling period is 10ms. Therefore, you should round up/down the value of Ts to the nearest 10ms.</p> <p>(2) You must choose "n" so that the resulting Scan Period is a positive integer in the range [2...150] ms.</p>	

Trial-and-Error Method

The trial-and-error method involves providing successive guesses of the sampling period to the Auto-Tuning function until the algorithm converges successfully towards satisfactory values of Kp, Ti, and Td.

NOTE: Unlike the process response curve method, the trial-and-error method is not based on any approximation law of the process response. However, it has the advantage of converging towards a value of the sampling period that is in the same order of magnitude as the actual value.

To perform a trial-and-error estimation of the Auto-Tuning:

Step	Action
1	Select the AT tab from the PID configuration window.
2	Set the Output limitation of Auto-Tuning to 10,000 .
3	Download your application to the logic controller. For further details on how to download an application, refer to SoMachine Basic Operating Guide (<i>see SoMachine Basic, Operating Guide</i>).
4	Select the PID tab from the PID Assistant screen.
5	Provide the first or n th guess in the Sampling Period ⁽¹⁾ field.
6	Launch Auto-Tuning (<i>see page 222</i>).
7	Wait until the Auto-Tuning process ends.
8	<p>Two cases can occur:</p> <ul style="list-style-type: none"> ● Auto-Tuning completes successfully: Continue to Step 10. ● Auto-Tuning unsuccessful: Refer to Auto-Tuning detected error codes (<i>see page 256</i>). This means that the current guess for the sampling period (Ts) is not correct. Try a new Ts guess and repeat steps 3 through 8, as many times as required until the Auto-Tuning process eventually converges.
<p>(1) If you do not have any first indication of the possible range for the sampling period, set this value to the minimum possible: 1 (1 unit of 10 ms).</p> <p>(2) If the PID regulation provided by this set of control parameters does not provide results that are totally satisfactory, you may still refine the trial-and-error evaluation of the sampling period until you obtain the correct set of Kp, Ti, and Td control parameters.</p>	

Step	Action
9	Follow these guidelines to provide a new Ts guess: <ul style="list-style-type: none"> ● Auto-Tuning ends with the detected error code 800C hex. This means the sampling period Ts is too large. Decrease the value of Ts to provide a new guess. ● Auto-Tuning ends with the detected error code 800A hex. This means the sampling period Ts is too small. Increase the value of Ts to provide a new guess.
10	Adjust the PID control parameters ⁽²⁾ (Kp, Ti, and Td) in the PID tab (<i>see page 245</i>) of the PID Assistant screen, as needed.
<p>(1) If you do not have any first indication of the possible range for the sampling period, set this value to the minimum possible: 1 (1 unit of 10 ms).</p> <p>(2) If the PID regulation provided by this set of control parameters does not provide results that are totally satisfactory, you may still refine the trial-and-error evaluation of the sampling period until you obtain the correct set of Kp, Ti, and Td control parameters.</p>	

Online Mode

In online mode, when the logic controller is in the periodic task, the value displayed in the Ts field (in the **PID Assistant** screen (*see page 238*)) can be different from the parameter entered (%MW). The Ts value is a multiple of the periodic task, whereas the %MW value is the value read by the logic controller.

Section 9.4

PID Assistant

What Is in This Section?

This section contains the following topics:

Topic	Page
Access the PID Assistant	239
General Tab	241
Input Tab	244
PID Tab	245
AT Tab	247
Output Tab	249

Access the PID Assistant

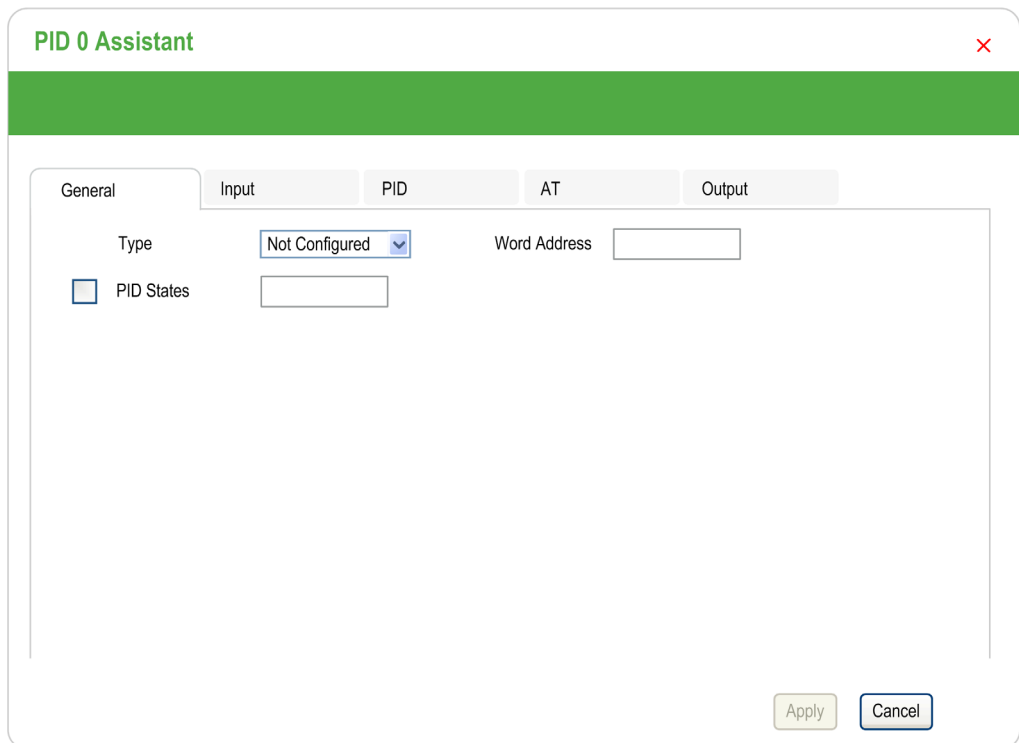
Introduction

Use the **PID Assistant** window of SoMachine Basic to enable you to configure the `PID` controller.

Configuration Assistant

In the `PID` properties table, click the **Configuration [...]** button. The **PID Assistant** screen will appear.


This graphic displays the **PID Assistant** screen:



The screenshot shows the **PID 0 Assistant** window. The title bar reads "PID 0 Assistant" with a red close button (X) on the right. Below the title bar is a green header bar. The main content area has a tabbed interface with five tabs: "General", "Input", "PID", "AT", and "Output". The "General" tab is currently selected. In the "General" tab, there are two main sections. The first section has a "Type" label followed by a dropdown menu showing "Not Configured" and a "Word Address" label followed by an empty text input field. The second section has a checkbox labeled "PID States" followed by an empty text input field. At the bottom right of the window, there are two buttons: "Apply" and "Cancel".

The **PID Assistant** screen displays several tabs, depending whether, you are in offline or online mode:

Tab	Access mode	Link
General	Offline	General tab (see page 241)
Input	Offline	Input tab (see page 244)
PID	Offline	PID tab (see page 245)
AT	Offline	AT tab (see page 247)
Output	Offline	Output tab (see page 249)

Once an operating mode is selected, tabs containing empty fields that require values are shown as display  and the border of the field is filled in red.

General Tab

Introduction

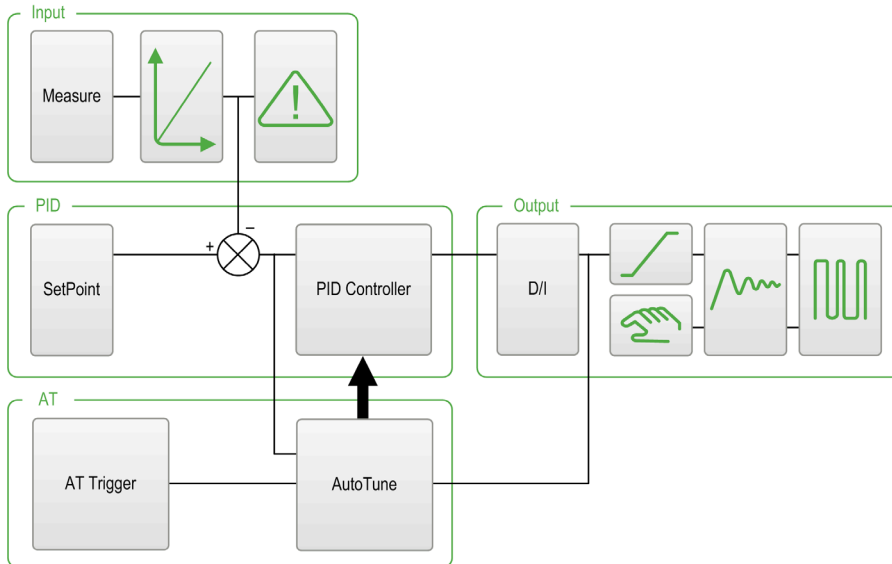
This section describes the **General** tab of the **PID**. **General** tab is displayed by default when you access the **PID** Assistant in offline mode.

Description

The table below describes the settings on the **General** tab.





Parameter	Description
Operating Mode	<p>Represents the PID mode to use:</p> <ul style="list-style-type: none"> ● Not configured ● PID ● AT + PID ● AT ● Word address <p>For further details about operating modes, refer to PID Operating Mode (<i>see page 220</i>).</p>
Word address	<p>You can provide a memory word in this text box (%MWxx) that is used to programmatically set the operating mode. The memory word can take 4 possible values depending on the operating mode you want to set:</p> <ul style="list-style-type: none"> ● %MWx = 0 (PID disabled) ● %MWx = 1 (to set PID only) ● %MWx = 2 (to set AT + PID) ● %MWx = 3 (to set AT only) ● %MWx = 4 (to set PI only)
PID States	<p>If you check the box to enable this option, you can provide a memory word in the associated field (%MWxx) that is used by the PID controller to store the current PID state while running the PID controller and/or the Auto-Tuning function. For more details, refer to PID States and Detected Error Codes (<i>see page 255</i>).</p>

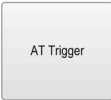







Graphical Assistant



The graphical assistant helps you to visualize how the `PID` function is built. This is a dynamic graphic that is updated according to the configuration.

The icons shown below describe when it is accessible or what happens if you click on it:

Display	Description
	Click this button to display the SetPoint field of the PID tab (<i>see page 245</i>).
	Click this button to display the PID tab (<i>see page 245</i>).
	Click this button to display the Output tab (<i>see page 249</i>).
	Click this button to display the Input tab (<i>see page 244</i>).

Display	Description
	Click this button to display the AT tab (<i>see page 247</i>).
	Click this button to display the AT tab (<i>see page 247</i>).
	This button appears when the Authorize option is checked in the Conversion zone of the Input tab (<i>see page 244</i>).
	This button appears when the Authorize option is checked in the Alarms zone of the Input tab (<i>see page 244</i>).
	This button appears if Limits is not equal to inhibit in the limits zone of the Output tab (<i>see page 249</i>).
	This button appears if manual mode is not equal to Inhibit in the manual mode zone of the Output tab (<i>see page 249</i>).
	Click this button to display the Output tab (<i>see page 249</i>).
	This button appears when the Authorize option is checked in the Output PWM zone of the Output tab (<i>see page 249</i>).

Input Tab

Introduction

This section describes the **Input** tab of PID. The **input** tab is used to enter the PID input parameters.

This tab is only accessible in offline mode and when an operating mode is selected from the **General** tab.

Description

The table below describes the settings that you may define.

Parameter	Description	
Measure	Specify the variable that contains the process value to be controlled. The default scale is from 0 to 10,000. You can enter either a memory word (%MWxx) or an analog input.	
Conversion	Authorize	Activate this box to convert the process value [0...10,000] into a linear range [Min...Max].
	Min value Max value	Specify the minimum and maximum values of the conversion scale. The process value is then automatically rescaled within the [Min value...Max value] interval. Min value or Max value can be memory words (%MWxx), constant words (%KWxx), or a value from -32768 to +32767. Note: The Min value must be less than the Max value .
Filter	Authorize	Activate this box to apply a filter to the measured input.
	(100 ms)	Specify the filter value from 0 to 10000 or a memory word address (%MWxx). The filter time base unit is 100 ms.
Alarms	Authorize	Activate this box to activate alarms in input variables. The alarm values should be determined relative to the process value obtained after the conversion phase. The alarm values must be from Min value to Max value when conversion is active. Otherwise, the alarm values will be from 0 to 10,000.
	Low Output	Specify the low alarm value in the Low field. This value can be a memory word (%MWxx), a constant (%KWxx), or a direct value. Output must contain the address of the bit, which will be set to 1 when the lower limit is reached. Output can be either a memory bit (%Mxx) or an output.
	High Output	Specify the high alarm value in the High field. This value can be a memory word (%MWxx), a constant (%KWxx), or a direct value. Output must contain the address of the bit, which will be set to 1 when the upper limit is reached. Output can be either a memory bit (%Mxx), or an output.

PID Tab

Introduction

Use **PID** tab to enter the internal **PID** parameters.

This tab is only accessible in offline mode and if an operating mode has been selected from the **General** tab.

Description

This table describes the settings that you may define:

Parameter	Description
Setpoint	Specify the PID setpoint value. This value can be a memory word (%MWxx), a constant word (%KWxx), or a direct value. This value must therefore be between 0 and 10,000 when conversion is inhibited. Otherwise it must be between the Min value and the Max value for the conversion.
Corrector type	If the PID or AT + PID operating mode has been previously chosen in the PID properties table, you can select the desired corrector type (PID or PI) from the drop-down list. If other operating modes (AT or Word Address) have been chosen, the Corrector type is set to Auto and greyed out (it cannot be modified manually). If PI is selected from the drop-down list, the Td parameter is forced to 0 and this field is disabled.
Parameters ⁽¹⁾	Kp (x0,01s) Specify the PID proportional gain, multiplied by 100. This value can be a memory word (%MWxx), a constant word (%KWxx), or a direct value. The valid range for the Kp parameter is: $0 < Kp < 10,000$. Note: If Kp is mistakenly set to 0 ($Kp \leq 0$ is invalid), the default value $Kp=100$ is automatically assigned by the PID function.
	Ti (x0,1s) Specify the integral time for a timebase of 0.1 seconds. This value can be a memory word (%MWxx), a constant word (%KWxx), or a direct value. It must be from 0 to 36,000. Note: To disable the integral action of the PID , set this coefficient to 0.
	Td (x0,1s) Specify the derivative time for a timebase of 0.1 seconds. This value can be a memory word (%MWxx), a constant word (%KWxx), or a direct value. It must be from 0 to 10,000. Note: To disable the derivative action of the PID , set this coefficient to 0.
(1) When Auto-Tuning is enabled, you no longer need to set the Kp, Ti, and Td parameters as they are automatically and programmatically set by the Auto-Tuning algorithm. In this case, you must enter in these fields an internal word address only (%MWxx). Do not enter a constant or a direct value when Auto-Tuning is enabled.	

Parameter	Description
Sampling period	<p>Specify the PID sampling period here for a timebase of 10^{-2} seconds (10 ms). This value can be a memory word (%MWxx), a constant word (%KWxx), or a direct value. It must be from 1 (0.01 s) to 10,000 (100 s).</p>
<p>(1) When Auto-Tuning is enabled, you no longer need to set the Kp, Ti, and Td parameters as they are automatically and programmatically set by the Auto-Tuning algorithm. In this case, you must enter in these fields an internal word address only (%MWxx). Do not enter a constant or a direct value when Auto-Tuning is enabled.</p>	

AT Tab

Introduction

The **AT** tab is related to the Auto-Tuning function. For more details, refer to PID tuning with Auto-Tuning (*see page 230*).

This tab is only accessible in offline mode and if an operating mode has been selected from the **General** tab.

Description

PID Auto-Tuning is an open-loop process that acts directly on the control process without regulation or any limitation other than provided by the Process Value (PV) limit and the output setpoint. Therefore, both values must be carefully selected within the allowable range as specified by the process to prevent potential process overload.

WARNING

UNSTABLE PID OPERATION

- The Process Value (PV) limit and the output setpoint values must be set with complete understanding of their effect on the machine or process.
- Do not exceed the allowable range for Process Value and Output Setpoint values.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use a relay output with the PID function.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This table describes the settings that you may define:

Field	Description
AT Mode	<p>Authorize</p> <p>Activate this box to enable Auto-Tuning operation. There are 2 ways to use this checkbox, depending on whether you set the operating mode manually or via a word address in the General tab of the PID function:</p> <ul style="list-style-type: none"> • If you set the Operating mode to PID + AT or AT from the General tab (<i>see page 241</i>), then the Authorize option is activated and not editable. • If you set the operating mode via a word address $\%MWx$ ($\%MWx = 2$: PID + AT; $\%MWx = 3$: AT), then you have to activate the Authorize option manually to allow configuring of the Auto-Tuning parameters.

Field	Description	
Measurement Range	Authorize	Activate this box to enable the range measurement. NOTE: If the range measurement is deactivated the Min. value is set to 0 and the Max. value is set to 10000.
	Min. Max.	Specify the minimum and maximum values of the range measurement. The values can be immediate value from 1 to 10000 or a memory word %MWx. NOTE: The Min. value must be less than the Max. value.
Dynamic AT corrector	Dynamic AT corrector	This parameter allows you to choose the dynamic of the AT process. This parameter affects the proportional gain (Kp) value computed by the AT process.
AT Trigger	AT Trigger	This parameter allows you to launch the AT process each time a rising edge is detected on the dedicated bit (memory bit or digital input bit).

Calculated Kp, Ti, Td Coefficients

Once the Auto-Tuning process is complete, the calculated Kp, Ti, and Td _{PID} coefficients are stored in their respective memory words (%MWx).

Output Tab

Introduction

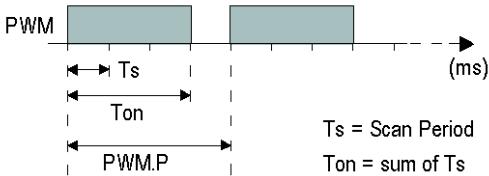
This tab is used to enter the PID output parameters.

This tab is only accessible in offline mode and if an operating mode has been selected from the **General** tab.

Description

This table describes the settings that you may define:

Field	Description
Action	Specify the type of PID action on the process here. Three options are available: Reverse , Direct , and Bit Address . If an increase in the output causes an increase in the process value measurement, define inverted action (Reverse); on the other hand, if this causes a process value reduction, make the PID direct (Direct). If you select Bit Address ⁽¹⁾ , you can modify the action type by modifying the associated bit, which is either a memory bit (%Mxx) or an input address (%Ix.y). The memory bit is set to 1 if the action is Direct and the memory bit is set to 0 if the action is Reverse .
Limits	Specify whether to place limits on the PID output. 3 options are available: Enable , Disable , and Bit Address . Select Enable to set the Bit to 1 or select Disable to set the Bit to 0. Select Bit Address for limit management of the bit by modifying the associated bit, which is either a memory bit (%Mxx) or an input address (%Ix.y). Set the high and low limits for the PID output. Min. or Max can be memory word (%MWxx), constant word (%KWxx), or a value from 1 to 10,000 (0.01% to 100% of the PWM period). Note: The Min. must always be less than the Max .
Manual mode	Specify whether to change the PID to manual mode. 3 options are available: Enable , Disable , and Bit Address . If you select Bit Address , you can switch to manual mode (bit to 1) or automatic mode (bit to 0) using the program, by modifying the associated bit which is either a memory bit (%Mxx) or an input. The Output of manual mode must contain the value that you wish to assign to the analog output when the PID is in manual mode (<i>see page 233</i>). This Output can be either a word (%MWxx) or a direct value in the [0...10,000] format.
Analog output	Specify the PID output to use when in auto-tuning mode. This Analog output ⁽²⁾ can be a memory word address or an analog output address. When using the PWM function of PID, only memory word addresses are allowed.
<p>(1) When Auto-Tuning is enabled, the Auto-Tuning algorithm automatically determines the correct type of action direct or reverse for the control process. You must then enter in the associated Bit Address textbox a memory bit (%Mxx) only.</p> <p>(2) Enter a memory address (%MWxx) or an analog output address (%QWx.y).</p>	

Field	Description
<p>Output PWM</p>	<p>Check this box to use the PWM function of <code>PID</code>.</p> <p>Specify the modulation period in the Period (0.1 s) text box. This period must be from 1 to 500 and can be a memory word (<code>%MWxx</code>) or a constant word (<code>%KWxx</code>). PWM precision depends on both the PWM period and the scan period. The precision is improved when the PWM ratio (<code>%PWM.R</code>) has the greatest number of values. For instance, with scan period = 20 ms and PWM period = 200 ms, <code>PWM.R</code> can take values 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%. With scan period = 50 ms and PWM period = 200 ms, <code>PWM.R</code> can take values 0%, 25%, 50%, 75%, and 100% of the period <code>PWM.P</code>.</p> <p><u>Example</u> : case of <code>PWM.R = 75%</code></p>  <p>Specify the PWM output bit as the value in Output. This can be either a memory bit (<code>%Mxx</code>) or an output address. For further details about PWM function, refer to the chapter Pulse Width Modulation (<code>%PWM</code>) (see page 63).</p>
<p>(1) When Auto-Tuning is enabled, the Auto-Tuning algorithm automatically determines the correct type of action direct or reverse for the control process. You must then enter in the associated Bit Address textbox a memory bit (<code>%Mxx</code>) only.</p> <p>(2) Enter a memory address (<code>%MWxx</code>) or an analog output address (<code>%QWx.y</code>).</p>	

Section 9.5

PID Programming

Using PID Function

This section provides descriptions and programming guidelines for using **PID** function.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	252
Programming and Configuring	254
PID States and Detected Error Codes	255

Description

Introduction

A proportional–integral–derivative (PID) is a generic control loop feedback mechanism (controller) widely used in industrial control systems. The PID controller uses an algorithm that involves 3 separate constant parameters: the proportional, the integral, and derivative values, denoted by P, I, and D respectively.

Key Features

The key features of the SoMachine Basic PID function are as follows:

- Analog input
- Linear conversion of the configurable measurement
- High or low configurable input alarm
- Analog or PWM output
- Cutoff for the configurable output
- Configurable direct or inverse action
- Auto-tuning function

Illustration

This is the PID function in the Ladder editor of SoMachine Basic:



NOTE: There must be a space between PID and the PID number (for example, PID<space>0).

Parameters

Unlike the `Timer` or the `Counter` function blocks, there is no PID function block in SoMachine Basic. The instruction `[PID x]` only enables the PID control loop function, where x is the PID number.

To configure the PID function, goto the **Programming** window, click **Tools** → **PID**, and then edit the PID properties (refer to the table below for the configuration parameters).

The PID function has the following parameters:

Parameter	Description	Value
Used	Checked if the I/O is used somewhere in the project	True/False False (Default)
PID	Name of the current PID object	A program can contain only a limited number of PID functions. For the maximum number of PID objects, refer to the table Maximum Number of Objects (see <i>Modicon M221, Logic Controller, Programming Guide</i>).
Symbol	Symbol of the current PID object	The symbol associated with this PID object. For details, refer to Defining and Using Symbols (see <i>SoMachine Basic, Operating Guide</i>).
[...]	A button to launch the assistant	Click to display the PID Assistant screen. For further details, refer to PID Assistant (see page 238).
Comment	Comment	A comment can be associated with this object.

Programming and Configuring

Introduction

This section describes how to program and configure the SoMachine Basic PID controller.

Enabling the PID Controller

The following example enables the PID 0 controller loop if the bit %M0 is set to 1:

Rung	Instruction
0	LD %M0 [PID 0]

NOTE: Refer to the reversibility procedure (see *SoMachine Basic, Generic Functions Library Guide*) to obtain the equivalent Ladder Diagram.

PID Analog Measurement

The PID function completes a PID correction using an analog measurement and setpoint and produces either an analog command in the same format or a PWM on a digital output.

To use PID at full scale (the highest resolution), configure the analog input dedicated to the PID controller measurement in [0...10,000] format. However, if you use the default configuration [0...4095], the PID controller will still function correctly.

Configuring the Scan Period

When using SoMachine Basic PID controllers, you must configure the scan mode of the logic controller to **Periodic** scan mode (**Program** tab, **Tasks** → **Master Task**). In periodic scan mode, each scan of the logic controller starts at a regular time interval so the sampling rate is constant throughout the measurement period. For further details on configuring the scan mode, refer to the *SoMachine Basic Operating Guide*.

In periodic scan mode, the system bit %S19 is set to 1 by the system if the logic controller scan time is greater than the period defined by the user program.

PID States and Detected Error Codes

Introduction

The SoMachine Basic PID controller has the ability to write the present state of both the PID controller and the Auto-Tuning process to a user-defined memory word. For further information on how to enable and configure the PID States memory word, refer to the **General** tab (*see page 241*) of the PID Assistant (*see page 238*).

The PID States memory word can record the following types of PID information:

- Present state of the PID controller
- Present state of the Auto-Tuning process
- PID detected error codes
- Auto-Tuning detected error codes

NOTE: The PID States memory word is read-only.

PID States Memory Word

PID States	Description
0000 hex	PID control is not active
2000 hex	PID control is in progress
4000 hex	PID setpoint has been reached

Auto-Tuning State Memory Word

Auto-Tuning State	Description
0100 hex	Auto-Tuning phase 1 (<i>see page 231</i>) in progress
0200 hex	Auto-Tuning phase 2 (<i>see page 231</i>) in progress
0400 hex	Auto-Tuning phase 3 (<i>see page 231</i>) in progress
0800 hex	Auto-Tuning phase 4 (<i>see page 231</i>) in progress
1000 hex	Auto-Tuning phase complete

PID Detected Error Codes

This table describes the potential detected errors that may be encountered during PID control:

Detected Error Code	Description
8001 hex	Operating mode value out of range
8002 hex	Linear conversion min and max equal
8003 hex	Upper limit for discrete output lower than lower limit

Detected Error Code	Description
8004 hex	Process value limit out of linear conversion range
8005 hex	Process value limit less than 0 or greater than 10000
8006 hex	Setpoint out of linear conversion range
8007 hex	Setpoint less than 0 or greater than 10000
8008 hex	Control action different from action determined at Auto-Tuning start

Auto-Tuning Detected Error Codes

This table records the Auto-Tuning detected error messages and describes possible causes as well as troubleshooting actions:

Detected Error Code	Description
8009 hex	The Process Value (PV) limit has been reached. As Auto-Tuning is an open-loop process, the Process Value (PV) limit works as maximum allowed value.
800A hex	Either the sampling period is too small or the output setpoint is too low. Increase either the sampling period or the Auto-Tuning output setpoint value.
800B hex	Kp is zero.
800C hex	The time constant is negative so the sampling period may be too large. For more details, refer to Limitations on Using the Auto-Tuning (see page 263).
800D hex	Delay is negative.
800E hex	<p>Detected error when calculating Kp. The Auto-Tuning algorithm is unstable (no convergence). This may be due to:</p> <ul style="list-style-type: none"> ● Disturbances on the process during Auto-Tuning has caused a distortion of the process static gain evaluation. ● The process value transient response is not large enough for Auto-Tuning to determine the static gain. ● A combination of the above. <p>Check the PID and Auto-Tuning parameters and make adjustments to improve convergence. Check also if there is no disturbance that could affect the process value. Try modifying:</p> <ul style="list-style-type: none"> ● the output setpoint ● the sampling period <p>Make sure that there is no process disturbance while Auto-Tuning is in progress.</p>
800F hex	Time constant exceeds delay ratio, $\tau/\theta > 20$. PID regulation may no longer be stable. For more details, refer to Limitations on Using the Auto-Tuning (see page 263).

Detected Error Code	Description
8010 hex	Time constant exceeds delay ratio, $\tau/\theta < 2$. PID regulation may no longer be stable. For more details, refer to Limitations on Using the Auto-Tuning (see page 263).
8011 hex	The limit for static gain K_p has been exceeded, $K_p > 10,000$. Measurement sensitivity of some application variables may be too low. The range must be rescaled within the [0...10,000] interval.
8012 hex	The computed value of integral time constant T_i has been exceeded, $T_i > 20,000$.
8013 hex	The computed value of derivative time constant T_d has been exceeded, $T_d > 10,000$.
8014 hex	Invalid input variables value (out of the range defined by low output and high output alarms (see page 244)).
8015 hex	Filter processing error: <ul style="list-style-type: none">● Cycle time out of range.● Filter time $< 10 \times$ cycle time

Appendices



Appendix A

PID Parameters

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Role and Influence of PID Parameters	262
PID Parameter Adjustment Method	264

Role and Influence of PID Parameters

Introduction

This section describes the role and influence of PID parameters.

PID Controller Model

The SoMachine Basic PID Controller implements a mixed (serial-parallel) PID correction. The integral and derivative actions act both independently and in parallel. The proportional action acts on the combined output of the integral and derivative actions.

Computational Algorithms

Two different computational algorithms are used depending on the value of the integral time constant (T_i):

- If $T_i \neq 0$, an incremental algorithm is used,
- If $T_i = 0$, a positional algorithm is used, along with a +5000 offset that is applied to the PID output.

Influence of Actions

Proportional action is used to influence the process response speed. An increase of the proportional action implies:

- a faster response
- a lower static error
- decrease in stability

Integral action is used to cancel out the static error. An increase of integration action (that is, a decrease of the integral time T_i) induces:

- A faster response
- A decrease in stability

Derivative action is anticipatory. In practice, it adds a term which takes account of the speed of variation in the deviation (which makes it possible to anticipate changes by accelerating process response times when the deviation increases and by slowing them down when the deviation decreases). An increase of derivative action (that is, an increase of the derivative time) implies:

- A slower response
- A reduced overshoot

NOTE: Given the derivative time, T_d is the time used to anticipate the variation of the deviation. Values of T_d that are too low or too high can lead to unwanted oscillations.

For each action, a suitable compromise must be found between speed and stability.

Limits of the PID Control Loop

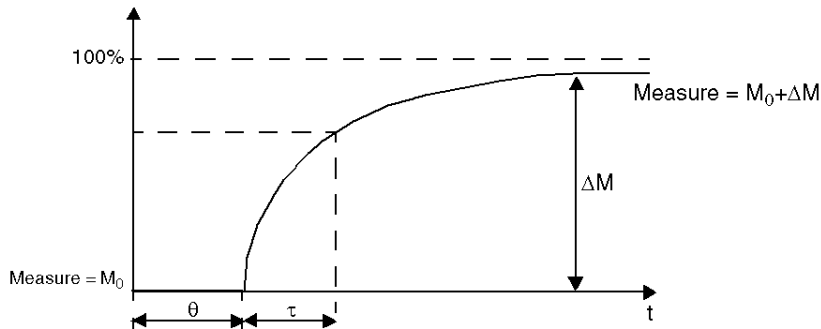
The process is assimilated to a pure delay first order with a transfer function:

$$H(p) = K \times \frac{e^{-\theta p}}{1 + \tau p}$$

where:

τ : model time constant

θ : model delay



The process control performance depends on the ratio $\frac{\tau}{\theta}$

The suitable PID process control is attained in the following domain: $2 < \frac{\tau}{\theta} < 20$

PID process control is best suited for the regulation of processes that satisfy the following condition:

- For $\frac{\tau}{\theta} < 2$, in other words for fast control loops (low θ) or for processes with a large delay (high t) the PID process control is no longer suitable. In such cases more complex algorithms should be used.
- For $\frac{\tau}{\theta} > 20$, a process control using a threshold plus hysteresis is sufficient.

PID Parameter Adjustment Method

Introduction

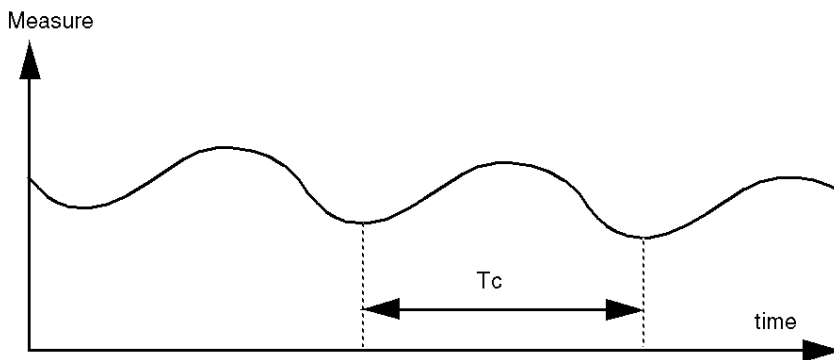
Numerous methods to adjust the PID parameters exist. The preferred method is the Ziegler and Nichols, which has 2 variants:

- closed loop adjustment
- open loop adjustment

Before implementing one of these methods, you must set the PID action (*see page 249*).

Closed Loop Adjustment

This principle uses a proportional command ($T_i = 0, T_d = 0$) to start the process by increasing a proportional coefficient until it starts to oscillate again after having applied a level to the PID corrector setpoint. All that is required is to raise the critical proportional gain (K_{pc}) which has caused the non-damped oscillation and the oscillation period (T_c) to reduce the values giving an optimal regulation.



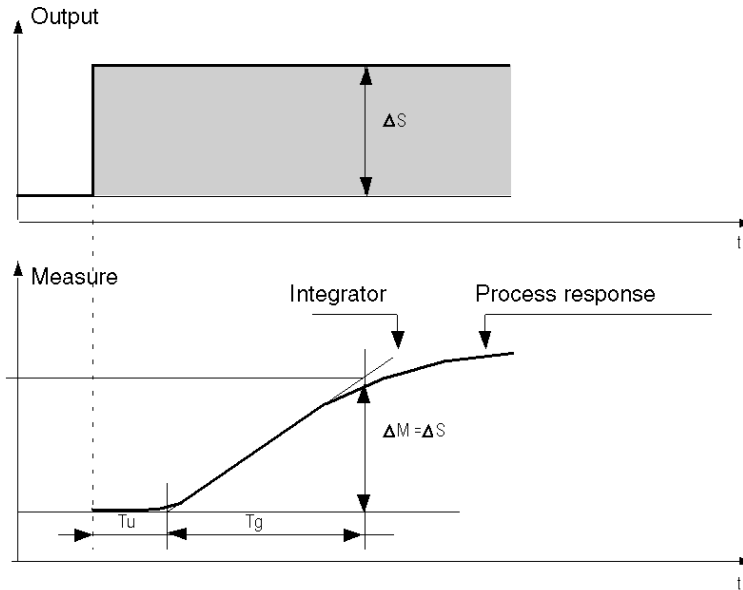
Depending on the correct type used (PID or PI), the adjustment of the coefficients is executed with the following values:

Corrector	Kp: Proportional Gain	Ti: Integration Time	Td: Derivative
PID	$K_{pc}/1.7$	$T_c/2$	$T_c/8$
PI	$K_{pc}/2.22$	$0.83 \times T_c$	–

When the PID is implemented with Auto-Tuning, the **Dynamic AT Corrector** parameter affects the proportional gain (K_p) value. The computation of the proportional gain in AT process depends on the selected speed of the dynamic corrector. Selecting **Fast** provides fast response with more overshoot, whereas selecting **Slow** provides slower response time with less overshoot.

Open Loop Adjustment

As the regulator is in manual mode (*see page 233*), you apply a level to the output and make the procedure response start the same as an integrator with pure delay time.



The intersection point on the right hand side, which is representative of the integrator with the time axes, determines the time T_u . Next, the T_g time is defined as the time necessary for the controlled variable (measurement) to have the same variation size (% of the scale) as the regulator output.

Depending on the corrector type used (PID or PI), the adjustment of the coefficients is executed with the following values:

Corrector	Kp: Proportional Gain	Ti: Integration Time	Td: Derivative
PID	$-1.2 T_g/T_u$	$2 \times T_u$	$0.5 \times T_u$
PI	$-0.9 T_g/T_u$	$3.3 \times T_u$	-

NOTE: For further details about parameter units, refer to **PID** tab (*see page 245*).

This adjustment method also provides a very dynamic command, which can express itself through unwanted overshoots during the change of pulses of the setpoints. In this case, lower the proportional gain until you get the required behavior. The method does not require any assumptions about the nature and the order of the procedure. You can apply it just as well to the stable procedures as to real integrating procedures. In the case of slow procedures (for example, the glass industry), the user only requires the beginning of the response to regulate the coefficients K_p , T_i , and T_d .

Glossary



A

absolute movement

A movement to a position defined from a reference point.

acceleration / deceleration

Acceleration is the rate of velocity change, starting from **Start Velocity** to target velocity.

Deceleration is the rate of velocity change, starting from target velocity to **Stop Velocity**. These velocity changes are implicitly managed by the PTO function in accordance with acceleration, deceleration, and jerk ratio parameters following a trapezoidal or an S-curve profile.

application

A program including configuration data, symbols, and documentation.

C

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

CW/CCW

ClockWise / Counter ClockWise

D

DWORD

(double word) Encoded in 32-bit format.

E

expansion bus

An electronic communication bus between expansion I/O modules and a controller.

F

function

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

Examples: boolean (AND) operators, calculations, conversions (BYTE_TO_INT)

H

homing

The method used to establish the reference point for absolute movement.

I

I/O

(input/output)

J

jerk ratio

The proportion of change of the acceleration and deceleration as a function of time.

P

POU

(program organization unit) A variable declaration in source code and a corresponding instruction set. POU's facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POU's are available to one another.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

S

S-curve ramp

An acceleration / deceleration ramp with a `JerkRatio` parameter greater than 0%.

start velocity

The minimum frequency at which a stepper motor can produce movement, with a load applied, without the loss of steps.

stop velocity

The maximum frequency at which a stepper motor stops producing movement, with a load applied, without the loss of steps.

T**trapezoidal ramp**

An acceleration / deceleration ramp with a `JerkRatio` parameter set to 0%.



Symbols

%DRV, *71*

%FC, *30*

%HSC

high speed counter, *38*

%PLS, *56*

%PWM, *64*

A

acceleration ramp, *108*

acknowledging and resetting error
MC_Reset_ATV, *95*

B

backlash, *114*

BUFFER_MODE, *146*

C

configuring

Drive function blocks, *78*

D

deceleration ramp, *108*

DIRECTION, *146*

Drive function blocks

configuring, *78*

Drive function blocks: error codes, *97*

E

enabling/disabling power stage

MC_Power_ATV, *79*

error codes

Drive function blocks, *97*

error handling

ErrID, *26*

Error, *26*

F

fast counter

configuration, *32*

description, *30*

programming example, *35*

FREQGEN

function block, *212*

function block configuration, *214*

frequency generator

function block, *212*

Function Block Object Codes

BUFFER_MODE, *146*

DIRECTION, *146*

HOMING_MODE, *147*

PTO_PARAMETER, *147*

function blocks

- FC (Fast Counter, *30*)
- frequency generator (%FREQGEN), *212*
- HSC (high speed counter), *38*
- MC_Halt_PTO, *186*
- MC_Home_PTO, *178*
- MC_Jog_ATV, *81*
- MC_Motion_PTO, *157*
- MC_MoveAbs_PTO, *174*
- MC_MoveRel_PTO, *169*
- MC_MoveVel_ATV, *84*
- MC_MoveVel_PTO, *164*
- MC_Power_ATV, *79*
- MC_Power_PTO, *161*
- MC_ReadMotionState_ATV, *92*
- MC_ReadStatus_ATV, *89*
- MC_Reset_ATV, *95*
- MC_SetPost_PTO, *181*
- MC_Stop_ATV, *87*
- MC_Stop_PTO, *183*
- MV_AbortTrigger_PTO, *204*
- MV_ReadActPos_PTO, *192*
- MV_ReadActVel_PTO, *190*
- MV_ReadAxis_PTO, *198*
- MV_ReadMotionState_PTO, *196*
- MV_ReadPar_PTO, *206*
- MV_ReadSts_PTO, *194*
- MV_Reset_PTO, *200*
- MV_TouchProbe_PTO, *202*
- MV_WritePar_PTO, *208*
- pulse, *56*
- pulse width modulation, *64*

functionalities

- PTO, *103*

H

high speed counter

- counting mode, *43*
- description, *38*
- frequency meter mode, *50*

HOMING_MODE, *147*

J

JerkRatio, *108*

M

management of Function Block Inputs and Input Objects

- Execute, *26*

management of Function Block Outputs and Output Objects

- Busy, *26*
- CmdAborted, *26*
- Done, *26*
- ErrID, *26*
- Error, *26*

MC_Halt_PTO

- controlled motion stop until velocity is 0, *186*

MC_Home_PTO

- command axis to perform homing sequence, *178*

MC_Jog_ATV

- starting jog mode, *81*

MC_Motion_PTO

- calling a Motion Task Table, *157*

MC_MoveAbs_PTO

- moving axis to a given position at specified speed, *174*

MC_MoveRel_PTO

- moving axis an incremental distance at specified speed, *169*

MC_MoveVel_ATV

- moving at specified velocity, *84*

MC_MoveVel_PTO

- moving an axis at specified speed, *164*

MC_Power_ATV

- enabling/disabling power stage, *79*

MC_Power_PTO

- enabling power to an axis, *161*

MC_ReadMotionState_ATV

- reading motion state, *92*

MC_ReadStatus_ATV

- reading device status, *89*

MC_Reset_ATV

- acknowledging and resetting error, *95*

MC_SetPost_PTO
 moving axis to specified position, *181*
 MC_Stop_ATV
 stopping movement, *87*
 MC_Stop_PTO
 commanding a controlled motion stop,
 183
 motion task table
 PTO, *121*
 moving at specified velocity
 MC_MoveVel_ATV, *84*
 MV_AbortTrigger_PTO
 aborting function blocks connected to trig-
 ger events, *204*
 MV_ReadActPos_PTO
 getting the position of the axis, *192*
 MV_ReadActVel_PTO
 getting the velocity of the axis, *190*
 MV_ReadAxisError_PTO
 getting an axis control error, *198*
 MV_ReadMotionState_PTO
 getting the motion status of the axis, *196*
 MV_ReadPar_PTO
 getting parameters from the PTO, *206*
 MV_ReadSts_PTO
 getting the status of the axis, *194*
 MV_Reset_PTO
 resetting axis-related errors, *200*
 MV_TouchProbe_PTO
 activating a trigger event on probe input,
 202
 MV_WritePar_PTO
 writing parameters to the PTO, *208*

P

PID
 AT tab, *247*
 auto-tuning, *222*
 closed loop adjustment, *264*
 configuration assistant, *239*
 description, *252*
 general tab, *241*
 Input tab, *244*
 open loop adjustment, *265*
 operating modes, *220*
 output tab, *249*
 parameter, *262*
 PID tab, *245*
 programming and configuring, *254*
 standard configuration, *226*
 states and detected error codes, *255*
 PTO
 configuration, *120*
 functionalities, *103*
 motion task table, *121*
 PTO_ERROR, *148, 149*
 PTO_PARAMETER, *147*
 pulse
 description, *56*
 function block configuration, *58*
 programming example, *62*
 pulse width modulation
 description, *64*
 function block configuration, *65*
 programming example, *69*

R

reading device status
 %MC_ReadStatus_ATV, *89*
 reading motion state
 MC_ReadMotionState_ATV, *92*

S

starting jog mode, MC_Jog_ATV, *81*
 stopping movement
 MC_Stop_ATV, *87*

