Modicon M100/M200 Logic Controller

Programming Guide

09/2015

SoMachine Basic

EIO0000002019.02

www.schneider-electric.com

Schneider Electric

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.

© 2015 Schneider Electric. All rights reserved.

Table of Contents

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, 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.

About the Book

At a Glance

Document Scope

This document describes the configuration and programming of the Modicon M100/M200 Logic Controller for SoMachine Basic. For further information, refer to the separate documents provided in the SoMachine Basic online help.

Validity Note

This document has been updated with the release of SoMachine Basic V1.3 SP3 EL.

Go to the Schneider Electric home page *www.schneider-electric.com/cn/zh*.

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	EIO0000001354 (ENG) EIO0000001359 (CHS)
SoMachine Basic Generic Functions - Library Guide	EIO0000001474 (ENG) EIO0000001479 (CHS)
Modicon M100/M200 Logic Controller - Hardware Guide	EIO0000002023 (ENG) EIO0000002024 (CHS)
Modicon TM3 Expansion Modules Configuration - Programming Guide	EIO0000001396 (ENG) EIO0000001401 (CHS)
Modicon TM2 Expansion Modules Configuration - Programming Guide	EIO0000001390 (ENG) EIO0000001395 (CHS)

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

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.

Part I
Introduction

Overview

This part provides general information about the Modicon M100/M200 Logic Controller and its configuration and programming features.

What Is in This Part?

This part contains the following chapters:

Chapter 1
About the Modicon M100/M200 Logic Controller

M100/M200 Logic Controller Description

Overview

The M100/M200 Logic Controller has various powerful features and can service a wide range of applications.

Software configuration, programming, and commissioning are accomplished with the SoMachine Basic software described in the SoMachine Basic Operating Guide and the SoMachine Basic Generic Functions Library Guide *(see SoMachine Basic, Generic Functions Library Guide)*.

Programming Languages

The M100/M200 Logic Controller is configured and programmed with the SoMachine Basic software, which supports the following IEC 61131-3 programming languages:

- IL: Instruction List
- LD: Ladder Diagram
- Grafcet (List)

Power Supply

The power supply of the M100 Logic Controller is 100...240 Vac.

The power supply of the M200 Logic Controller is 24 Vdc or 100...240 Vac.

Real Time Clock

The M200 Logic Controller includes a Real Time Clock (RTC) system *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

Run/Stop

The M100/M200 Logic Controller can be operated externally by the following:

- A hardware Run/Stop switch *(see Modicon M100/M200 Logic Controller, Hardware Guide)*
- A Run/Stop *(see Modicon M100/M200 Logic Controller, Hardware Guide)* operation by a dedicated digital input, defined in the software configuration. For more information, refer to Configuring Digital Inputs *(see page 62)*.
- SoMachine Basic software. For more information, refer to the SoMachine Basic Operating Guide.

Memory

This table describes the different types of memory:

Memory Type	Size	Used to...
RAM	512 Kbytes RAM memory: 256 Kbytes for internal variables and 256 Kbytes for application and data.	execute the application and contain data.
Flash	1 Mbyte, of which 256 Kbytes is used to back up the application and data in case of power outage.	save the application.

Embedded Inputs/Outputs

The following embedded I/O types are available, depending on the controller reference:

- Regular inputs
- Fast inputs associated with counters
- Regular sink transistor outputs
- Fast sink transistor outputs associated with pulse generators
- Relay outputs

Removable Storage

The M100/M200 Logic Controller includes an embedded micro SD card slot.

The M100/M200 Logic Controller allows the following types of file management with an micro SD card:

- Clone management *(see page 150)*: back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management *(see page 152)*: download firmware updates directly to the logic controller
- Application management *(see page 153)*: back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management *(see page 155)*: add, change, or delete the post configuration file of the logic controller
- Error log management *(see page 157)*: back up or delete the error log file of the logic controller

Embedded Communication Features

The following types of communication ports are available depending on the controller reference:

- Ethernet
- USB Mini-B
- Serial Line

Chapter 2
Configuration Features

Introduction

This chapter provides information related to M100/M200 Logic Controller memory mapping, task, states, behaviors, objects, and functions. The topics explained in this chapter allow the operator to understand the featured specifications of M100/M200 Logic Controller that are primarily needed to configure and program the controller in SoMachine Basic.

What Is in This Chapter?

This chapter contains the following sections:

Section 2.1
Objects

What Is in This Section?

This section contains the following topics:

Overview of Objects

Definition

In SoMachine Basic, the term *object* is used to represent an area of logic controller memory reserved for use by an application. Objects can be:

- Simple software variables, such as memory bits and words
- Addresses of digital and analog inputs and outputs
- Controller-internal variables, such as system words and system bits
- Predefined system functions or function blocks, such as timers and counters.

Controller memory is either pre-allocated for certain object types, or automatically allocated when an application is downloaded to the logic controller.

Objects can only be addressed by a program once memory has been allocated. Objects are addressed using the prefix `%`. For example, `%MW12` is the address of a memory word, `%Q0.3` is the address of an embedded digital output, and `%TM0` is the address of a `Timer` function block.

Object Types

Introduction

The language object types for the M100/M200 Logic Controller are described in the following table:

Object Type	Object	Object Function	Description
Memory objects	%M	Memory bits	Stores memory bit.
	%MW	Memory words	Stores 16-bit memory word.
	%MD	Memory double words	Stores 32-bit memory word.
	%MF	Memory floating point	Stores memory floating point in a mathematical argument which has a decimal in its expression.
	%KW	Constant words	Stores 16-bit constant word.
	%KD	Constant double words	Stores 32-bit constant word.
	%KF	Constant floating points	Stores constant floating point in a mathematical argument which has a decimal in its expression.
System objects	%S	System bits *(see page 358)*	Stores system bit.
	%SW	System words *(see page 366)*	Stores system word.

Object Type	Object	Object Function	Description
I/O objects	%I	Input bits *(see page 166)*	Stores value of the digital input.
	%Q	Output bits *(see page 167)*	Stores value of the digital output.
	%IW	Analog input words *(see page 168)*	Stores value of the analog input.
	%QW	Analog output words *(see page 170)*	Stores value of the analog output.
	%FC	Fast counters *(see page 172)*	Serves as either up-counter or down-counter and counts the rising edge of discrete inputs in single word or double word computational mode.
	%HSC	High speed counters *(see page 178)*	Counts of discrete input in single word or double word computational mode.
	%PLS	Pulse *(see page 197)*	Generates a square wave pulse signal on dedicated output channels.
	%PWM	Pulse width modulation *(see page 205)*	Generates a modulated wave signal on dedicated output channels with a variable duty cycle.
	%PTO	Pulse train output *(see page 211)*	Generates a pulse train output to control a linear single-axis stepper or servo drive in open loop mode.

Object Type	Object	Object Function	Description
Software objects	%TM	Timers	Specifies a time before triggering an action.
	%C	Counters	Provides up and down counting of actions.
	%MSG	Messages	Stores the status message at the communication port.
	%R	LIFO/FIFO registers	Stores memory up to 16 words of16 bits each in 2 different ways, queue, and stacks.
	%DR	Drum registers	Operates on a principle similar to an electromechanical drum controller which changes step according to external events.
	%SBR	Shift bit registers	Provides a left or right shift of binary data bits (0 or 1).
	%SC	Step counters	Provides a series of steps to which actions can be assigned.
	SCH	Schedule blocks	Controls actions at a predefined month, day, and time.
	PID	PID *(see page 311)*	Provides a generic control loop feedback in which output is proportional, integral, and derivative of the input.

Object Type	Object	Object Function	Description
PTO objects			
Refer to Pulse Train Output *(see page 211)*.			
Communication objects	%READ_ VAR	Read data from a remote device	The %READ_VAR function block is used to read data from a remote device on Modbus SL or Modbus TCP
	%WRITE_ VAR	Write data to a Modbus device	The %WRITE_VAR function block is used to write data to an external device using the Modbus SL or Modbus TCP protocol
	%WRITE_ READ_VAR	Read data from and write data to a Modbus device	The %WRITE_READ_VAR function block is used to read and write data stored in internal memory words to an external device using the Modbus SL or Modbus TCP protocol.
	%SEND_ RECV_ MSG	Communication on an ASCII link	The %SEND_RECV_MSG function block is used to send or receive data on a serial line configured for the ASCII protocol.

Memory objects, software objects, and communication objects are generic objects used in SoMachine Basic, whereas system objects and I/O objects are controller-specific. All controller-specific objects are discussed in the section *Programming (see page 159)*.

For programming details of memory objects, software objects, and communication objects, refer to the SoMachine Basic Generic Functions Library Guide.

Addressing Objects

Addressing Examples

This table presents addressing examples for various object types:

Object Type	Syntax	Example	Description
Memory objects			
Memory bits	%M*i*	%M25	Internal memory bit 25.
Memory words	%MW*i*	%MW15	Internal memory word 15.
Memory double words	%MD*i*	%MD16	Internal memory double word 16.
Memory floating points	%MF*i*	%MF17	Internal memory floating point 17.
Constant words	%KW*i*	%KW26	Constant word 26.
Constant double words	%KD*i*	%KD27	Internal constant double word 27.
Constant floating points	%KF*i*	%KF28	Internal constant floating point 28.
System objects			
System bits	%S*i*	%S8	System bit 8.
System words	%SW*i*	%SW30	System word 30.
I/O objects			
Digital inputs	%I*y.z*	%I0.5	Digital input 5 on the controller (embedded I/O).
Digital outputs	%Q*y.z*	%Q3.4	Digital output 4 on the expansion module at address 3 (expansion module I/O).
Analog inputs	%IW0.*y0z*	%IW0.101	Analog input 1 on the cartridge 1.
Analog outputs	%QW0.*m0n*	%QW0.202	Analog output 2 on the cartridge 2.
Fast counters	%FC*i*	%FC2	Fast counter 2 on the controller.
High speed counters	%HSC*i*	%HSC1	High speed counter 1 on the controller.
Pulse	%PLS*i*	%PLS0	Pulse output 0 on the controller.
Pulse width modulation	%PWM*i*	%PWM1	Pulse width modulation output 1 on the controller.
Pulse train output	%PTO*i*	%PTO1	Pulse train output 1 on the controller.

i Object instance identifier that indicates the instance of the object on the controller. For the maximum number of instances of each object, refer to *Maximum Number of Objects* *(see page 26)*. If n is the maximum number of an object, the instance range is 0...n-1.
m Cartridge number on the controller.
n Channel number on the cartridge.
y Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.
z Channel number on the controller or expansion module.

Object Type	Syntax	Example	Description
Software objects			
Timers	%TM*i*	%TM5	Timer instance 5.
Counters	%C*i*	%C2	Counter instance 2.
Message	%MSG*i*	%MSG1	Program compilation status message 1.
LIFO/FIFO registers	%R*i*	%R3	FIFO/LIFO registers instance 3.
Drums	%DR*i*	%DR6	Drum register 6 on the controller.
Shift bit registers	%SBR*i*	%SBR5	Shift bit register 5 on the controller.
Step counters	%SC*i*	%SC5	Step counter 5 on the controller.
Schedule blocks	SCH *i*	SCH 3	Schedule block 3 on the controller.
PID	PID *i*	PID 7	PID feedback object 7 on the controller.
PTO objects			
MC_Power_PTO (motion function block)	%MC_POWER_PTO*i*	%MC_POWER_PTO1	MC_POWER_PTO function block instance 1. For more information on PTO function blocks, refer to *Pulse Train Output (%PTO)* *(see page 211)*.
MC_Reset_PTO (administrative function block)	%MC_RESET_PTO*i*	%MC_RESET_PTO0	MC_RESET_PTO function block instance 0. For more information on PTO function blocks, refer to *Pulse Train Output (%PTO)* *(see page 211)*.
Communication objects			
Read Var	%READ_VAR*i*	%READ_VAR2	READ_VAR function block instance 2.
Write Var	%WRITE_VAR*i*	%WRITE_VAR4	WRITE_VAR function block instance 4.
WriteRead Var	%WRITE_READ_VAR*i*	%WRITE_READ_VAR0	WRITE_READ_VAR function block instance 0.
Send Receive Message	%SEND_RECV_MSG*i*	%SEND_RECV_MSG6	SEND_RECV_MSG function block instance 6.

i Object instance identifier that indicates the instance of the object on the controller. For the maximum number of instances of each object, refer to *Maximum Number of Objects (see page 26)*. If n is the maximum number of an object, the instance range is 0...n-1.
m Cartridge number on the controller.
n Channel number on the cartridge.
y Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.
z Channel number on the controller or expansion module.

Maximum Number of Objects

Description

This table provides information about the maximum number of objects supported by the M100/M200 Logic Controller:

Object Type	Object	Maximum Number Allowed
Memory objects	%M	1024
	%MW	• 4000 for TM100••• • 8000 for TM200•••
	%MD / %MF	• 3999 for TM100••• • 7999 for TM200•••
	%KW	512
	%KD / %KF	511
System objects	%S	160
	%SW	234
I/O objects	%I	• 9 for TM100C16R / TM200C•16• • 14 for TM100C24R / TM200C•24• • 24 for TM100C40R / TM200C•40• • 24 for TM100C40R / TM200C•40• • 36 for TM200C•60•
	%Q	• 7 for TM100C16R / TM200C•16• • 10 for TM100C24R / TM200C•24• • 16 for TM100C40R / TM200C•40• • 24 for TM200C•60•
	%IW	**NOTE:** The M100/M200 Logic Controller has no embedded analog I/Os. Use cartridges or expansion modules to add analog I/Os to your configuration.
	%QW	
	%FC	4
	%HSC	4
	%PLS / %PWM / %PTO	• 0 for TM100C•••R / TM200C•••R • 2 for TM200C•••U / TM200C•••T

Object Type	Object	Maximum Number Allowed
Software objects	%TM	255
	%C	255
	%MSG	1
	%R	4
	%DR	8
	%SBR	8
	%SC	8
	%SCH	16
	PID	14
PTO objects	%MC_MOTIONTASK_PTO	2
	%MC_POWER_PTO	86
	%MC_MOVEVEL_PTO	86
	%MC_MOVEREL_PTO	86
	%MC_MOVEABS_PTO	86
	%MC_HOME_PTO	86
	%MC_READACTVEL_PTO	40
	%MC_READACTPOS_PTO	40
	%MC_READSTS_PTO	40
	%MC_READMOTIONSTATE_PTO	40
	%MC_READAXISERROR_PTO	40
	%MC_RESET_PTO	40
	%MC_TOUCHPROBE_PTO	40
	%MC_ABORTTRIGGER_PTO	40
	%MC_READPAR_PTO	40
	%MC_WRITEPAR_PTO	40
Communication objects	%READ_VAR	16
	%WRITE_VAR	16
	%WRITE_READ_VAR	16
	%SEND_RCV_MSG	16

Section 2.2
Task Structure

What Is in This Section?

This section contains the following topics:

Tasks and Scan Modes

Overview

SoMachine Basic has the following scan modes:

- **Normal mode**
  Continuous cyclic scanning mode (Freewheeling mode); a new scan starts immediately after the previous scan has completed.
- **Periodic mode**
  Periodic cyclic scanning mode; a new scan starts only after the configured scan time of the previous scan has elapsed. Every scan is therefore the same duration.

SoMachine Basic offers the following task types:

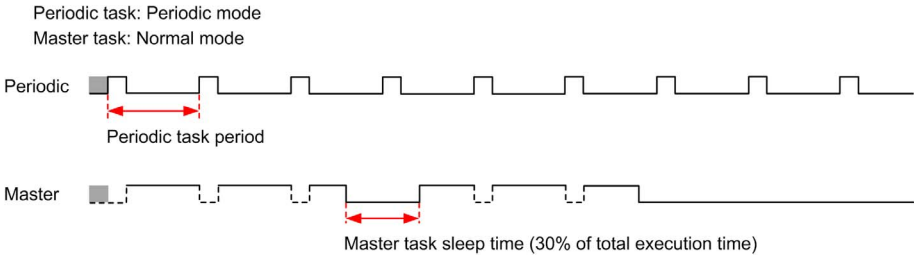
- **Master task**: Main task of the application.
  Master task is triggered by continuous cyclic scanning (in normal scan mode) or by the software timers (in periodic scan mode) by specifying the scan period of 2...150 ms (default 10 ms).
- **Periodic task**: A short duration subroutine processed periodically.
  Periodic tasks are triggered by software timers, so are configured by specifying the scan period of 2...255 ms (default 255 ms) in the periodic scan mode.
- **Event task**: A very short duration subroutine to reduce the response time of the application.
  Event tasks are triggered by the physical inputs or the `HSC` function blocks. These events are associated with embedded digital inputs (%I0.2...%I0.5) (rising, falling or both edges) or with the high speed counters (`%HSC0` and `%HSC1`) (when the count reaches the high speed counter threshold). You can configure 2 events for each HSC function block.

Periodic tasks and events are configured in periodic scan mode. Master task can be configured in either normal scan mode or periodic scan mode.

For more information, refer to the Configuring Program Behavior and Tasks *(see SoMachine Basic, Operating Guide)*.

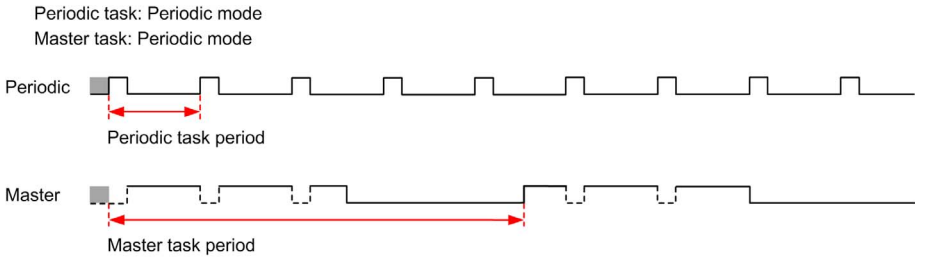
Master Task in Normal Scan Mode

This graphic presents the relationship between master tasks and periodic task execution when the master task is configured in normal scan mode:



Master Task in Periodic Scan Mode

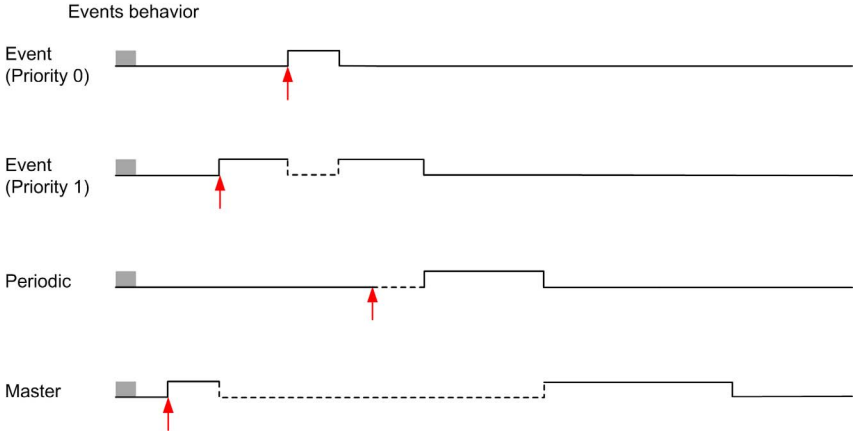
This graphic presents the relationship between master tasks and periodic tasks when the master task is configured in periodic scan mode:



Event Priority Over Master and Periodic Tasks

Event priorities control the relationship between the event tasks, master tasks, and periodic tasks. The event task interrupts the master task and periodic task execution.

This figure presents the relationship between event tasks, master tasks, and periodic tasks in the periodic mode:



The event tasks are triggered by a hardware interruption that sends a task event to the event task.

Watchdog Timer

You can configure a specific watchdog timer for the master task and periodic tasks. If the task execution time exceeds the configured watchdog timer period, the logic controller goes to the `HALTED` state. This watchdog timer is managed by the software timers.

A system watchdog timer verifies whether the program is using more than 80% of the processing capacity.

Maximum Number of Tasks and Priorities

Tasks Priorities

This table summarizes the task types and their priorities:

Task Type	Scan Mode	Triggering Condition	Configurable Range	Maximum Number of Tasks	Priority
Master	Normal	Normal	Not applicable	1	Lowest
	Periodic	Software timer	2...150 ms		
Periodic	Periodic	Software timer	2...255 ms	1	Higher than master task and lower than event tasks
Event	Periodic	Physical inputs	%I0.2...%I0.5	4	Highest
		`%HSC` function blocks	Up to two events per %HSC object	4	

Events Priorities

Refer to Event Priorities and Queues *(see SoMachine Basic, Operating Guide)*.

Section 2.3
Controller States and Behaviors

Introduction

This section provides you with information on controller states, state transitions, and behaviors in response to system events. It begins with a detailed controller state diagram and a description of each state. It then defines the relationship of output states to controller states before explaining the commands and events that result in state transitions. It concludes with information about persistent variables and the effect of SoMachine Basic task programming options on the behavior of your system.

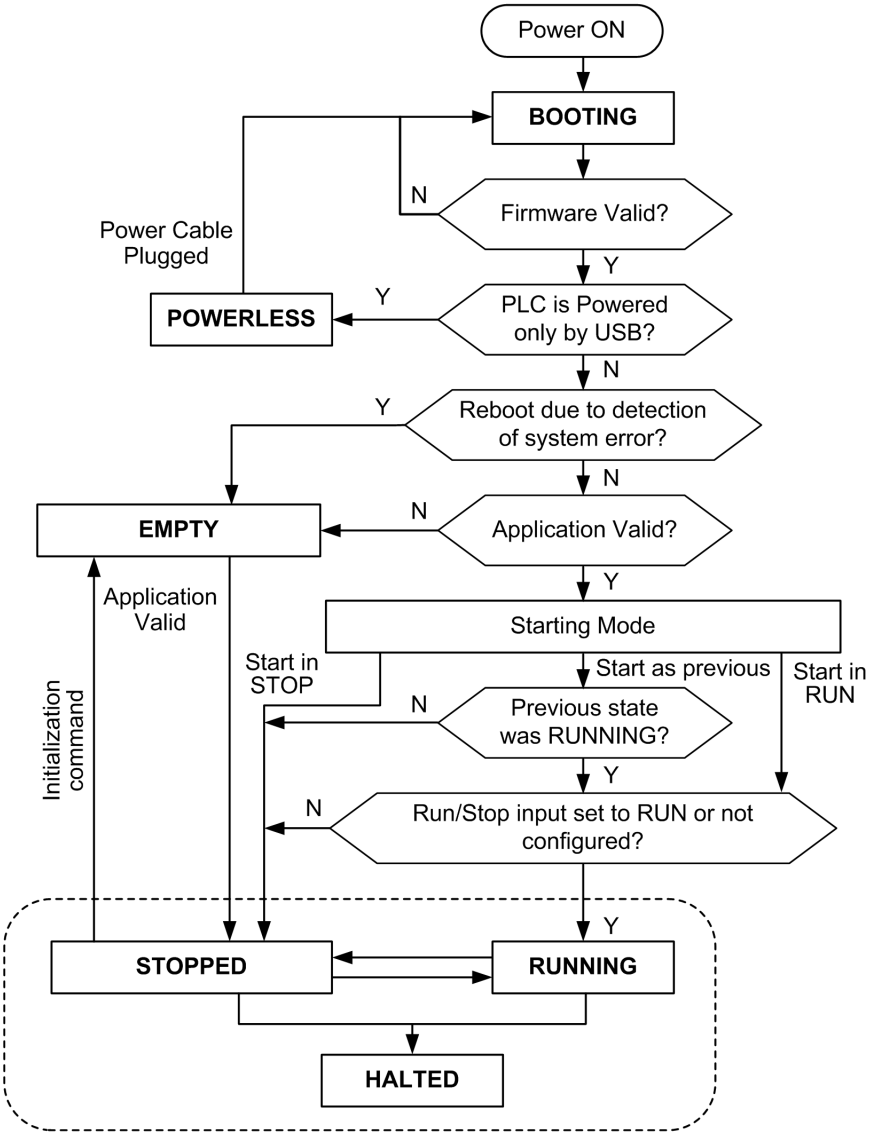
What Is in This Section?

This section contains the following topics:

Controller States Diagram

Controller States Diagram

This figure describes the controller operating states:



Controller States Description

Introduction

This section provides a detailed description of the controller states.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Never assume that your controller is in a certain controller state before commanding a change of state, configuring your controller options, uploading a program, or modifying the physical configuration of the controller and its connected equipment.
- Before performing any of these operations, consider the effect on all connected equipment.
- Before acting on a controller, always positively confirm the controller state by viewing its LEDs, confirming the condition of the Run/Stop input, checking for the presence of output forcing, and reviewing the controller status information via SoMachine Basic.

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

When using the Start In Run feature, the controller will start executing program logic when power is applied to the equipment. It is essential to know in advance how automatic reactivation of the outputs will affect the process or machine being controlled. Configure the Run/Stop input to help control the Start In Run feature. In addition, the Run/Stop input is designed to give local control over remote RUN commands. If the possibility of a remote RUN command after the controller had been stopped locally by SoMachine would have unintended consequences, you must configure and wire the Run/Stop input to help control this situation.

⚠ WARNING

UNINTENDED MACHINE START-UP

- Confirm that the automatic reactivation of the outputs does not produce unintended consequences before using the Start In Run feature.
- Use the Run/Stop input to help control the Start In Run feature and to help prevent the unintentional start-up from a remote location.
- Verify the state of security of your machine or process environment before applying power to the Run/Stop input or before issuing a Run command from a remote location.

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

Controller States Table

This table provides detailed description of the controller operating states:

Controller state	Description	Communication	Application execution	LED		
				PWR	RUN	ERR
BOOTING	The logic controller does not have valid firmware. The communication channels are enabled to allow updating of the runtime firmware. It is not possible to log in with SoMachine Basic. Outputs are set to initialization values *(see page 41)*.	Restricted	No	On	Off	On
EMPTY	This state indicates that there is not a valid application. It is possible to log in with SoMachine Basic (*download/watchlist*). Inputs are forced to 0. Outputs are set to initialization values *(see page 41)*.	Yes	No	On	Off	1 flash
STOPPED	This state indicates that the logic controller has a valid application which is stopped. Inputs are read. Outputs are set to fallback values *(see page 42)*, or forced values *(see page 42)* from SoMachine Basic. Status alarm output is set to 0.	Yes	No	On	Flashing	Off
RUNNING	This state indicates that the logic controller is executing the application. Inputs are read by the application tasks. Outputs are written by the application tasks, or from SoMachine Basic in online mode (watchlist, output forcing *(see page 42)*). Status alarm output is set to 1.	Yes	Yes	On	On	Off

Controller state	Description	Communication	Application execution	LED		
				PWR	RUN	ERR
HALTED	This state indicates that the application is stopped because of an application software error. *(see page 42)* Objects retain their current values, allowing analysis of the cause of the detected error. All tasks are stopped on the current instruction. The communication capabilities are the same as in STOPPED state. Inputs are not read, and keep their last values. Outputs are set to fallback values *(see page 42)*. Status alarm output is set to 0.	Yes	No	On	Flashing	On
POWERLESS	This state indicates that the logic controller is powered only by the USB cable. This mode can only be used to update the firmware (by USB) or to download/upload the user application (by USB). In this state, all state transitions are not allowed. So the only way to change the state of the logic controller is to connect the main power. In this case, the logic controller boots and reloads all installed components. It is possible to log in with SoMachine Basic (*download/upload/watchlist*). Inputs are forced to 0. Outputs are set to initialization values *(see page 41)*.	Yes (only USB)	No	Off	Flashing	Off

NOTE: The system word %SW6 indicates the logic controller state (EMPTY, STOPPED, RUNNING, HALTED, and POWERLESS).

Controller State Transitions

Boot Controller

Effect: Command a reboot of the logic controller. For details about power-on sequence, refer to the controller state diagram *(see page 33)*.

Methods:

- Power cycle
- Reboot by script
  - The script on a micro SD card can issue a REBOOT as its last command.

Application Download

Effect: Download the application into the logic controller memory.

Methods:

- SoMachine Basic online button:
  - Select the **PC to controller (download)** command.
    Effect: Erase the application in the logic controller and set the logic controller in `EMPTY` state. Download the application in the logic controller memory. If download is successful, a Cold Start is done and the logic controller is set in `STOPPED` state.
- Application file transfer by SD card:
  - Effect: At the next reboot, erase the application in the logic controller and download the application files from the micro SD card to the logic controller memory. If download is successful, a Cold Start is done and the logic controller is set in `STOPPED` state.

Initialize Controller

Effect: Set the controller in `EMPTY` state, and then, after a Cold Start, in `STOPPED` state.

Methods:

- SoMachine Basic online button:
  - Select the **Initialize controller** command.

RUN Controller

Effect: Command a transition to the RUNNING state.

Methods:

- Run/Stop *(see Modicon M100/M200 Logic Controller, Hardware Guide)* switch on front face:
  - It commands a transition to RUNNING state on rising edge.
- Run/Stop *(see Modicon M100/M200 Logic Controller, Hardware Guide)* input:
  - The input must be configured in the application (Configuring Digital Inputs *(see page 62)*).
  - It commands a transition to RUNNING state on rising edge.
- SoMachine Basic online button:
  - Select the **Run Controller** command.
- Application starting mode *(see SoMachine Basic, Operating Guide)* setting:
  - **Start in Run** or **Start in Previous State**.

STOP Controller

Effect: Command a transition to the STOPPED state.

Methods:

- Run/Stop *(see Modicon M100/M200 Logic Controller, Hardware Guide)* switch on front face:
  - It forces a transition to STOPPED state on low level.
- Run/Stop *(see Modicon M100/M200 Logic Controller, Hardware Guide)* input:
  - The input must be configured in the application (Configuring Digital Inputs *(see page 62)*).
  - It forces a transition to STOPPED state on low level.
- SoMachine Basic online button:
  - Select the **Stop Controller** command.
- Application starting mode *(see SoMachine Basic, Operating Guide)* setting:
  - **Start in Stop** or **Start in Previous State**.
- **Download** command:
  - It needs the controller to be set in STOPPED state (after the download the controller is in STOPPED state).

Error Detected (Transition to HALTED State)

Effect: Command a transition to the HALTED state.

Reasons for switching to HALTED state:

- Application Watchdog timeout (configured by the user)
- System Watchdog timeout (system overrun, over 80% of the CPU processing capacity is used)

Cold Start

Cold Start is defined to be a power-up with all data initialized to its default values, and program started from the beginning with program variables cleared. Software and hardware settings are initialized.

Cold Start occurs for the following reasons:

- Boot controller without validated application online modification.
- Download application
- Initialize logic controller

Effects of the Cold Start:

- Initialize the function blocks.
- Clear the user memory.
- Put the system bits `%S` and system words `%SW` to their initial values.
- Reload parameters from post configuration (changes in the post configuration are applied).
- Restore application from flash memory (unsaved online changes are lost).
- Restart the internal components of the controller.

Persistent Variables

Automatic Save on Power Outage

The controller automatically saves the first 3000 memory words (%MW0 to %MW2999) in the internal flash memory following any removal of power. The data is restored to the memory word region during the initialization, even if the controller performs a cold start.

These *automatically saved* persistent variables are reinitialized in case of a new download, INIT command, or %S0 activation. Refer to System Bits *(see page 358)*.

Save by User Request

You can save up to 1000 memory words (%MW3000 up to %MW3999) in the flash memory. The number of memory words saved is specified in the system word %SW148. Refer to System Words *(see page 366)*.

The save operation can only be performed when the controller is in the STOP mode. To perform the save operation, set system bit %S93 to 1. Refer to System Bits *(see page 359)*. The save operation occupies the controller for approximately 40 ms. The system bit %S92 is set to 1 to signal the end of the save operation. Refer to System Bits *(see page 359)*.

The writing of memory words to flash memory is performed in stages between MAST cycles. The flash memory region is erased at the end of the MAST cycle.

Output Behavior

Introduction

The controller defines output behavior in response to commands and system events in a way that allows for greater flexibility. An understanding of this behavior is necessary before discussing the commands and events that affect controller states.

The possible output behaviors and the controller states to which they apply are:

- Managed by application program
- Initialization values
- Fallback Mode *(see SoMachine Basic, Operating Guide)*
  - Maintain values
  - Fallback values
- Output forcing

Managed by Application

Your application manages outputs normally. This applies in the `RUNNING` state.

Hardware Initialization Values

This output state applies in the `BOOTING`, `EMPTY` and `POWERLESS` states.

In the initialization state, the outputs assume the following values:

- For embedded outputs:
  - Fast transistor output: 0 Vdc
  - Regular transistor output: 0 Vdc
  - Relay output: Open
  - Expert I/O functions (HSC, PLS, PWM, and PTO): 0 Vdc
- For expansion module outputs:
  - Regular transistor output: 0 Vdc
  - Relay output: Open

Software Initialization Values

This output state applies when downloading or when resetting the application. It applies at the end of the download or at the end of a cold start.

Input objects %I and `%IW` are set to 0. Output objects %Q and %QW are set according to the fallback strategy (refer to Fallback Behavior).

Fallback Values

This output state applies in the STOPPED and HALTED states.

In the fallback mode, the outputs assume the following values:

- For embedded outputs:
  - Fast transistor output: according to fallback setting (Fallback Behavior *(see SoMachine Basic, Operating Guide)*)
  - Regular transistor output: according to fallback setting (Fallback Behavior)
  - Relay output: according to fallback setting (Fallback Behavior)
  - Expert I/O functions (HSC, PLS, PWM, and PTO):
    - Fallback value: according to fallback setting (Fallback Behavior)
    - Maintain values *(see SoMachine Basic, Operating Guide)*: 0 Vdc
- For expansion module outputs:
  - Regular transistor output: according to fallback setting (Fallback Behavior)
  - Relay output: according to fallback setting (Fallback Behavior)

Output Forcing

The controller allows you to force the state of selected outputs to a defined value for the purposes of system testing, commissioning, and maintenance.

You are only able to force the value of an output while your controller is connected to SoMachine Basic.

To do so, use the **Force** command in an animation table.

Output forcing overrides all other commands to an output irrespective of the task programming that is being executed.

The forcing is not released by online change or logout of SoMachine Basic.

The forcing is automatically released by Cold Start *(see page 39)* and Application Download *(see page 37)* command.

The forcing does not apply for expert I/O functions (HSC, PLS, PWM, and PTO).

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- You must have a thorough understanding of how forcing will affect the outputs relative to the tasks being executed.
- Do not attempt to force I/O that is contained in tasks that you are not certain will be executed in a timely manner, unless your intent is for the forcing to take affect at the next execution of the task whenever that may be.
- If you force an output and there is no apparent affect on the physical output, do not exit SoMachine Basic without removing the forcing.

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

Section 2.4
Post Configuration

Introduction

This section describes how to manage and configure the post configuration file of the Modicon M100/M200 Logic Controller.

What Is in This Section?

This section contains the following topics:

Post Configuration

Introduction

Post configuration is an option that allows you to modify some parameters of the application without changing the application. Post configuration parameters are defined in a file called **Machine.cfg**, which is stored in the controller.

By default, all communication parameters are set in the configuration of the application. However, under certain conditions, some or all of these parameters can be modified automatically using the post configuration mechanism. One or more communication parameters can be specified in the post configuration file, and those parameters can override the parameters specified by the configuration. For example, one parameter may be stored in the post configuration file to change the EtherNet/IP address of the controller while leaving the other Ethernet parameters, such as the gateway address, unchanged.

Parameters

The post configuration file allows you to modify network parameters.

Ethernet parameters:
- Address configuration mode
- IP address
- Subnet mask
- Gateway address
- Device name

Serial line parameters, for each serial line in the application (embedded port or TMCR2SL1/TMCR2SL1A cartridge):
- Physical medium
- Baud rate
- Parity
- Data bits
- Stop bit
- Modbus address
- Polarization (for RS-485)

Operating Mode

The post configuration file is read and applied:
- after a Cold Start *(see page 39)*
- after a reboot *(see page 37)*
- after an application download *(see page 37)*

For further details on controller states and transitions, refer to Controller States and Behaviors *(see page 32)*.

Post Configuration File Management

Introduction

The file **Machine.cfg** must be stored in the directory `/usr/cfg` of the controller.

The post configuration file can be transferred, modified, or deleted with a micro SD card.

The Ethernet parameters of the post configuration file can also be modified with SoMachine Basic during the connection with a logic controller.

NOTE: A post configuration file example is available in the directory `Firmwares & PostConfiguration\PostConfiguration\add_change\usr\cfg` of the SoMachine Basic installation directory.

Post Configuration File Format

A valid configuration must use the following format:

- The character '#' means beginning of comment, everything after this sign until the end of the line is ignored. Comments are not saved in the post configuration area of the M100/M200 Logic Controller.
- Rule is `channel.parameter=value` (no space around the '=' sign).
- `Channel` and `parameter` are case-sensitive.
- Allowed channel, parameter, and values are in the following table.

Channel	Parameter	Description	Value
ETH	IPMODE	Address configuration mode	0 = Fixed 1 = BOOTP 2 = DHCP
	IP	IP address	Dotted decimal string
	MASK	Subnet mask	Dotted decimal string
	GATEWAY	Gateway address	Dotted decimal string
	NETWORKNAME	Device name on the network	ASCII string (maximum 16 characters)
SL1 SL2	HW	Physical medium	0 = RS-232 1 = RS-485
	BAUDS	Data transmission rate	1200, 2400, 4800, 9600, 19200, 38400, 57600 or 115200
	PARITY	Parity for error detection	0 = None 1 = Odd 2 = Even
	DATAFORMAT	Data format	7 or 8
	STOPBIT	Stop bit	1 or 2
	MODBUSADDR	Modbus address	1...247
	POLARIZATION	Polarization	0 = No 1 = Yes

Post Configuration File Transfer

After creating and modifying your post configuration file, it must be transferred to the logic controller. The transfer is performed by using a script to copy the post configuration file to a micro SD card.

Refer to Adding or Changing a Post Configuration *(see page 155)*.

Modifying a Post Configuration File

Use a text editor to modify the post configuration file on the PC.

NOTE: Do not change the text file encoding. The default encoding is ANSI.

NOTE: The Ethernet parameters of the post configuration file can be modified with SoMachine Basic. For more information, refer to Connecting to a Logic Controller *(see SoMachine Basic, Operating Guide)*.

Deleting the Post Configuration File

Refer to Removing a Post Configuration File *(see page 156)*.

NOTE: The parameters defined in the application will be used instead of the corresponding parameters defined in the post configuration file.

Part II
Configuring the M100/M200 Logic Controller

Overview

This part provides information about how to configure the M100/M200 Logic Controller references.

What Is in This Part?

This part contains the following chapters:

Chapter 3
How to Configure a Controller

Overview

This chapter describes how to build a configuration in SoMachine Basic and configure the M100/M200 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

Building a Configuration

Introduction

Configure a controller by building a configuration in SoMachine Basic. To build a configuration, first create a new project or open an existing project.

Refer to *SoMachine Basic Operating Guide* for information on how to:

- Create or open an existing project
- Replace the default logic controller
- Add an expansion module to the logic controller
- Add a cartridge to the logic controller
- Save the project.

Some general information about the SoMachine Basic user interface is provided below.

Start Page

The start page window is always displayed when you launch SoMachine Basic. Use this window to register the SoMachine Basic software, manage the connection to the logic controller, and create or select a project to work with.

SoMachine Basic Window

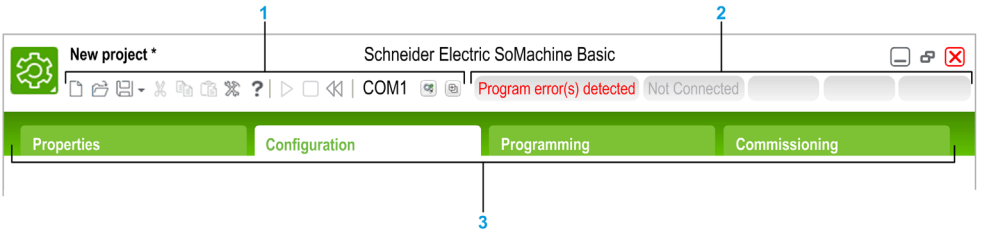
Once you have selected a project to work with, SoMachine Basic displays the main window.

At the top of the main window, a toolbar *(see SoMachine Basic, Operating Guide)* contains icons that allow you to perform common tasks, including returning to the start page window.

Next to the toolbar, the status bar *(see SoMachine Basic, Operating Guide)* displays informational messages about the state of the connection to the logic controller.

Below the toolbar and the status bar, the main window is divided into a number of *modules*. Each module controls a different stage of the development cycle, and is accessible by clicking the module tab.

This figure presents the toolbar, status bar, and the module tabs in the main window:



1 Toolbar
2 Status bar
3 Module tabs

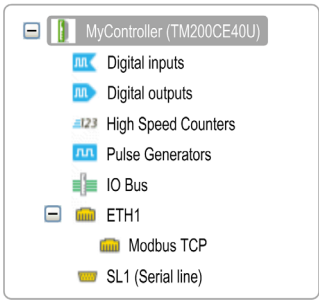
Item	Description
Toolbar	Provides easy access to commonly used functions. For more information, refer to the Toolbar *(see SoMachine Basic, Operating Guide)*.
Status bar	Displays status and information messages on the system status. For more information, refer to the Status bar *(see SoMachine Basic, Operating Guide)*.
Module tabs	To develop an application, work your way through the module tabs from left to right: ● **Properties** Set up the project properties. ● **Configuration** Replicate and configure the hardware configuration of the logic controller and associated expansion modules. ● **Programming** Develop the program in one of the supported programming languages. ● **Commissioning** Manage the connection between SoMachine Basic and the logic controller, upload/download applications, test, and commission the application.

Hardware Tree

The hardware tree is displayed on left-hand side in the **Configuration** window. It presents a structured view of the hardware configuration. When you add a controller, an expansion module, or a cartridge to the project, several nodes are automatically added to the hardware tree.

NOTE: The nodes in the hardware tree are specific to the controller and the hardware configuration. These nodes depend on the I/O functions that the controller, expansion modules, and cartridges provide.

This figure presents the hardware tree of the controller configuration:



Item	Description
Digital inputs	Use to configure the embedded digital inputs of the logic controller.
Digital outputs	Use to configure the embedded digital outputs of the logic controller.
High Speed Counters	Use to configure the embedded high speed counting functions (HSC).
Pulse Generators	Use to configure the embedded pulse generator functions (PLS/PWM/PTO).
IO Bus	Use to configure the expansion modules and cartridges connected to the logic controller.
ETH1	Use to configure the embedded Ethernet communications.
Modbus TCP	Use to configure the Modbus TCP protocol for Ethernet communications.
SLn **(Serial line)**	Use to configure the embedded serial line or the serial line added using a cartridge. **NOTE:** All M100/M200 references can support only one serial line cartridge.
n Serial line number (1 or 2, controller-specific).	

Editor

The editor area is displayed in center of the **Configuration** window. It displays the graphical representation of hardware configuration of the devices. The hardware configuration in a project can be:

- Only a controller
- A controller with cartridge(s)
- A controller with expansion modules
- A controller with cartridge(s) and expansion modules.

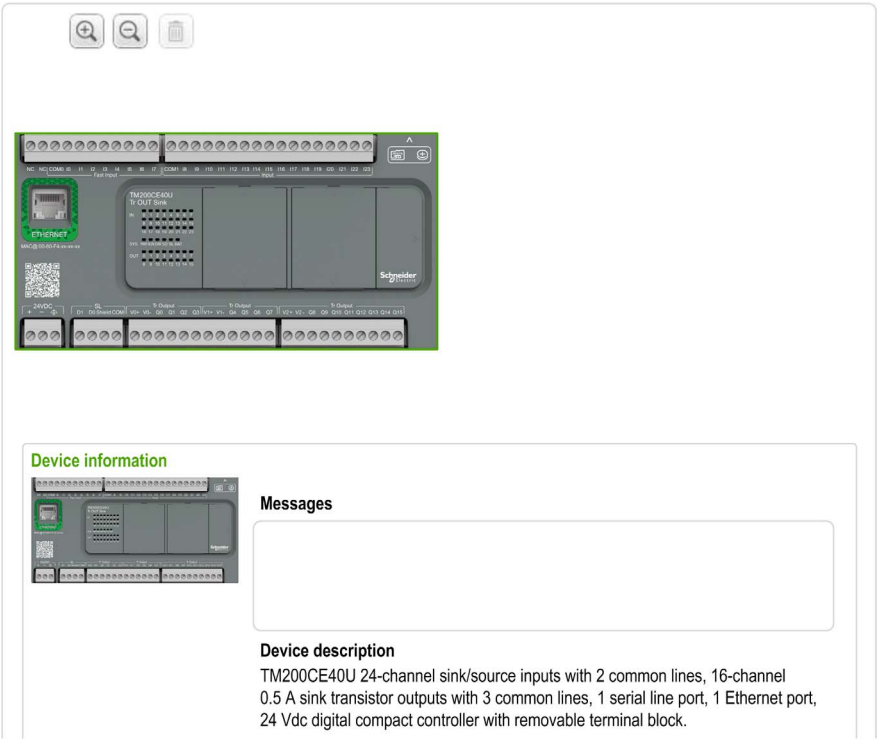
The editor area displays:

- A short description about the device when you click the device image or when you click the device node in the hardware tree.
- Configuration properties of the item selected in the hardware tree.

If you add an expansion module to the configuration, the expansion module appears at the right-hand side of the controller or the previously added expansion module. Cartridges are added on the controller in the cartridge slot.

When configuring a controller, a cartridge, or an expansion module, the configuration properties of the node selected in the hardware tree are displayed below the graphical configuration. These properties allow you to configure the device.

This figure presents the configuration of a controller with an expansion module (the controller is selected):

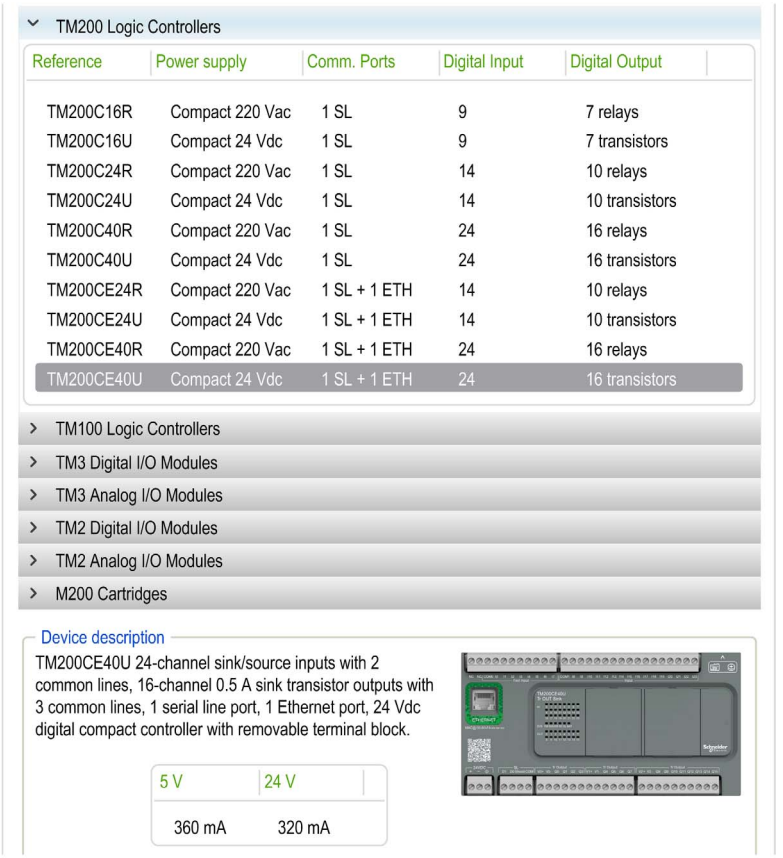


Catalog

The catalog area is displayed on right-hand side in the **Configuration** window. It displays the complete range of the logic controllers, expansion modules, and cartridges that can be configured using SoMachine Basic. It also provides a short description of the selected device.

You can drag-and-drop the objects from the catalog area to the editor area. You can also replace the existing controller by a different controller with simple drag-and-drop from the catalog.

This figure presents the catalog of the logic controllers and the expansion modules:



Optional I/O Expansion Modules

Presentation

I/O expansion modules can be marked as optional in the configuration. The **Optional module** feature provides a more flexible configuration by the acceptance of the definition of modules that are not physically attached to the logic controller. Therefore, a single application can support multiple physical configurations of I/O expansion modules, allowing a greater degree of scalability without the necessity of maintaining multiple application files for the same application.

Without the **Optional module** feature, when the logic controller starts up the I/O expansion bus (following a power cycle, application download or initialization command), it compares the configuration defined in the application with the physical I/O modules attached to the I/O bus. Among other diagnostics made, if the logic ontroller determines that there are I/O modules defined in the configuration that are not physically present on the I/O bus, an error is detected and the I/O bus does not start.

With the **Optional module** feature, the logic controller ignores the absent I/O expansion modules that you have marked as optional, which then allows the logic controller to start the I/O expansion bus.

The logic controller starts the I/O expansion bus at configuration time (following a power cycle, application download, or initialization command) even if optional expansion modules are not physically connected to the logic controller.

The following module types can be marked as optional:
- TM3R I/O expansion modules
- TM3 I/O expansion modules
- TM2 I/O expansion modules

The application must be configured with a functional level of at least **Level 3.2** for modules marked as optional to be recognized as such by the logic controller.

You must be fully aware of the implications and impacts of marking I/O modules as optional in your application, both when those modules are physically absent and present when running your machine or process. Be sure to include this feature in your risk analysis.

⚠ WARNING

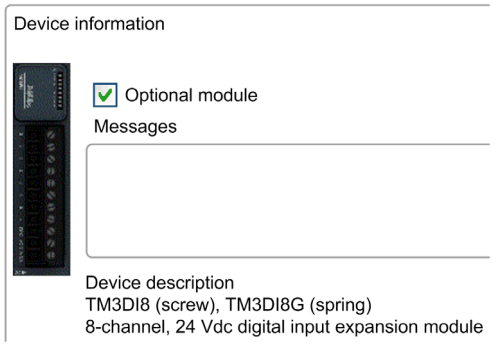
UNINTENDED EQUIPMENT OPERATION

Include in your risk analysis each of the variations of I/O configurations that can be realized marking I/O expansion modules as optional, and in particular the establishment of TM3 Safety modules (TM3S…) as optional I/O modules, and make a determination whether it is acceptable as it relates to your application.

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

Marking an I/O Expansion Module as Optional in Offline Mode

To add a module and mark it as optional in the configuration:

Step	Action
1	Drag-and-drop the I/O expansion module from the catalog to the editor.
2	In the **Device information** area, select the **Optional module** check box: 

To mark an existing I/O expansion module as optional in the configuration:

Step	Action
1	Select the I/O expansion module in the editor.
2	In the **Device information** area, select the **Optional module** check box.

Optional I/O Expansion Modules in Online Mode

SoMachine Basic operates in online mode when a physical connection to a logic controller has been established.

When in SoMachine Basic online mode, the modification of the **Optional module** feature is disabled. You can visualize the downloaded configuration in the application:

- An I/O expansion module represented in yellow is marked as optional and not physically connected to the logic controller at start-up. An information message to that effect is displayed in the **Device information** area.
- An I/O expansion module represented in red is not marked as optional and not detected at start-up. An information message to that effect is displayed in the **Device information** area.

The selection of the **Optional module** feature is used by the logic controller to start the I/O bus. The following system words are updated to indicate the status of the physical I/O bus configuration:

System Word	Comment
%SW118 Logic controller status word	Bits 13 and 14 are pertinent to the I/O module status relative to the I/O bus. Bit 13, if FALSE, indicates that there are mandatory modules as defined by the I/O expansion bus configuration that are absent or otherwise inoperative when the logic controller attempts to start the I/O expansion bus. In this case, the I/O bus does not start. Bit 14, if FALSE, indicates that one or more modules have ceased communication with the logic controller after the I/O expansion bus is started. This is the case whether an I/O expansion module is defined as mandatory or as an optional module but present at start-up.
%SW119 I/O expansion module configuration	Each bit, starting with bit 1 (bit 0 is reserved), is dedicated to a configured I/O expansion module and indicates whether the module is optional (TRUE) or mandatory (FALSE) when the controller attempts to start the I/O bus.
%SW120 I/O expansion module status	Each bit, starting with bit 1 (bit 0 is reserved), is dedicated to a configured I/O expansion module and indicates the status of the module. When the logic controller attempts to start the I/O bus, if the value of %SW120 is non-zero (indicating that an error is detected for at least one of the modules), the I/O expansion bus does not start unless the corresponding bit in %SW119 is set to TRUE (indicating the module is marked as an optional module). When the I/O bus is started, if the value of %SW120 is modified by the system, it indicates that an error is detected on one or more I/O expansion modules (regardless of the **Optional module** feature).

Refer to System Words *(see page 366)* for more information.

Configuring the M100/M200 Logic Controller

Controller Configuration

Controller configuration depends on the number and type of embedded input/outputs, I/O objects, and communication ports.

Use the **Configuration** tab to configure the properties of your controller and the expansion modules. Select a node in the hardware tree to configure the properties of the controller.

This table shows the available configurations of the M100/M200 Logic Controller:

Reference	Digital inputs	Digital Outputs	High Speed Counters	Pulse Generators	Ethernet	Serial Line
TM100C••R / TM200C••R	X	X	X	–	–	X
TM200C••U	X	X	X	X	–	X
TM200CE••R	X	X	X	–	X	X
TM200CE••U	X	X	X	X	X	X
TM200C•••T	X	X	X	X	–	X
TM200CE••T	X	X	X	X	X	X

– Not available for configuration in SoMachine Basic.

X Available for configuration in SoMachine Basic. For information on how to configure:
- Digital inputs, refer to Configuring Digital Inputs *(see page 62)*
- Digital outputs, refer to Configuring Digital Outputs *(see page 66)*
- High speed counters, refer to Configuring High Speed Counters *(see page 78)*
- Pulse generators, refer to Configuring Pulse Generators *(see page 68)*
- Ethernet, refer to Configuring Ethernet *(see page 132)*
- Serial line, refer to Configuring Serial Line *(see page 138)*.

Updating Firmware Using Executive Loader Wizard

Overview

You can update the executives of the controller using the executive loader wizard (OS loader).

Refer to *Controller States and Behavior* *(see page 32)* for information on the logic controller operating states and status of the LEDs.

Updating the Firmware of the Controller

To launch the Exec Loader Wizard, follow these steps:

Step	Action
1	Close all Windows applications, including virtual machines.
2	Click **Start** → **Programs** → **Schneider Electric** → **SoMachine Basic** → **SoMachine Basic Firmware Update** or run the *ExecLoaderWizard.exe* from the *SoMachine Basic installation folder\Execloader* folder.
3	Follow the steps on the wizard to complete the firmware update.

Chapter 4
Embedded Input/Output Configuration

Overview

This chapter describes how to configure the embedded I/O objects of the M100/M200 Logic Controller.

The number of embedded inputs and outputs depends on the controller reference. For more information, refer to the table of M100/M200 Logic Controller references *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

What Is in This Chapter?

This chapter contains the following sections:

Section 4.1
Configuring Digital Inputs

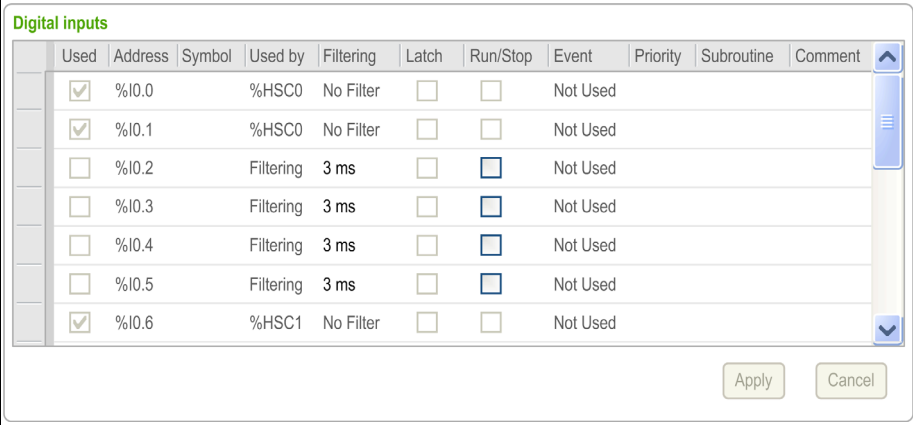
Configuring Digital Inputs

Introduction

By default, all digital inputs are used as regular inputs. Some of the digital inputs are fast and can be used by configuring the high speed counters *(see page 78)* while other inputs can be configured as event sources.

Digital Inputs Configuration

This table describes how to configure the digital inputs:

Step	Action
1	Click the **Digital inputs** node in the hardware tree to display the digital input properties. This figure shows the properties of the digital inputs in the editor area: 
2	Edit the properties to configure the digital inputs. For detailed information on the digital input configuration parameters, refer to the table below.

This table describes each parameter of the digital input configuration:

Parameter	Editable	Value	Default value	Description
Used	No	True/False	False	Indicates whether the input channel is being used in a program or not.
Address	No	%I0.x	–	Displays the address of the digital input on the controller, where x represents the channel number. If the controller has 8 digital input channels, x varies from 0...7. If the controller has 16 digital input channels, x varies from 0...15. For example, `%I0.2` is the third digital input channel of the logic controller.
Symbol	Yes	–	–	Allows you to associate a symbol with the digital input object. Double-click in the **Symbol** column, type the name of the symbol, and press Enter.
Used by	No	*Any*	**Filtering**	Displays the name of the component that uses the input channel. For example, if the input channel is used by a subroutine, this field displays **User logic**. The possible values in this field are: ● **User logic** ● **Filtering** ● **Latch** ● **Run/Stop** ● **Event** ● **%HSCx** where x is the high speed counter instance on the controller ● **%FC**y where y is the fast counter instance on the controller If an input is being used by more than one operation, all values, separated by commas, are displayed in this field.
Filtering	Yes	**No Filter** **3 ms** **12 ms**	**3 ms**	Allows you to select the noise filter duration for the input channel. Using a filter for the digital inputs reduces the noise on the controller input. If you select filter for an input, you cannot configure that input for: ● **Latch** ● **Event**

Parameter	Editable	Value	Default value	Description
Latch	Yes	True/False	False	Allows you to enable or disable latching for the inputs configured as events (`%I0.2...%I0.5`). By default, this option is disabled due to default value of **Filtering**. Set the **Filtering** to **No Filter** to enable the **Latch** option. Latching enables pulses with a duration shorter than the controller scan time to be memorized. When a pulse duration is shorter than a scan time and has a value greater than or equal to 1 ms, the controller latches the pulse, which is then updated in the next scan. If you enable **Latch** for an input, you cannot configure that input for: ● **Filtering** ● **Run/Stop** ● **Event**
Run/Stop	Yes	True/False	False	Allows you to configure 1 digital input as an additional Run/Stop switch. If you configure a digital input as Run/Stop switch, you cannot use the input in any other function block (for example, high speed counter function block, fast counter function block, and so on). If you enable **Run/Stop** for an input, you cannot configure that input for: ● **Latch** ● **Event**
Event	Yes	**Not Used** **Falling Edge** **Rising Edge** **Both edges**	**Not Used**	Allows you to select an event that triggers the inputs `%I0.2...%I0.5`. By default, this option is disabled due to default value of **Filtering**. Set the **Filtering** to **No Filter** to enable the **Event** option. If you select an event from the drop-down list (other than **Not Used**), the **Priority** parameter enables for editing to set the priority of the event.
Priority	Yes	0...7	7	Allows you to set the priority of the triggering event for the inputs `%I0.2...%I0.5`. You can set the priority of each event using the **Priority** parameter that is editable only for the inputs configured as event. Assign each configured event a different priority: if 2 events have same priority, a detected error message appears in the window.

Parameter	Editable	Value	Default value	Description
Subroutine	No	–	–	Displays the number of the subroutine associated with an input configured as an event.
Comment	Yes	–	–	Allows you to associate a comment with the digital input object. Double-click in the **Comment** column, type an optional comment, and press Enter.

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Digital Inputs (%I) *(see page 166)*.

Section 4.2
Configuring Digital Outputs

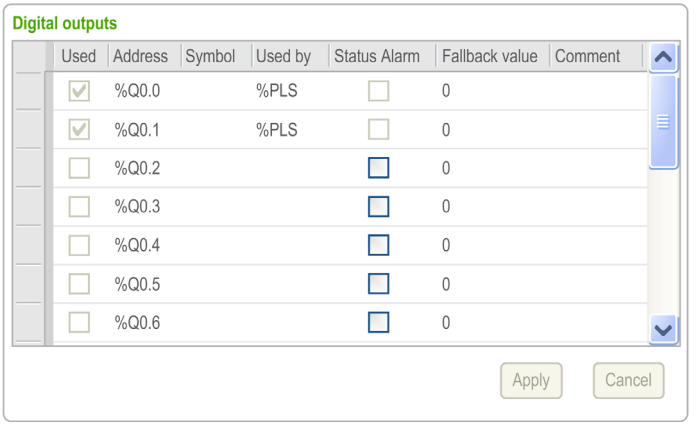
Configuring Digital Outputs

Introduction

By default, all digital outputs are used as regular outputs. For controllers equipped with transistor outputs, two outputs are fast transistor outputs and can be used by configuring the pulse generators *(see page 68)*.

Digital Outputs Configuration

This table describes how to configure the digital outputs:

Step	Action
1	Click the **Digital outputs** node in the hardware tree to display the digital output properties. This figure shows the properties of the digital outputs in the editor area: 
2	Edit the properties to configure the digital outputs. For detailed information on the digital output configuration parameters, refer to the table below.

This table describes each parameter of the digital output configuration:

Parameter	Editable	Value	Default value	Description
Used	No	True/False	False	Indicates whether the output channel is being used in a program or not.
Address	No	%Q0.x	–	Displays the address of the digital output on the controller, where x represents the channel number. If the controller has 8 digital output channels, x varies from 0...7. If the controller has 16 digital output channels, x varies from 0...15. If the controller has 24 digital output channels, x varies from 0...23. For example, `%Q0.2` is the third digital output channel on the controller.
Symbol	Yes	–	–	Allows you to associate a symbol with the digital output object. Double-click in the **Symbol** column, type the name of the symbol, and press Enter.
Used by	No	*Any*	*Empty*	Displays the name of the component that uses the output channel. For example, if the output channel is used as status alarm, it displays **Alarm**.
Status Alarm	Yes	True/False	False	Allows you to enable or disable the status alarm for the output (`%Q0.0`...`%Q0.7`). You can configure only one output channel for the status alarm. You cannot configure an output as status alarm if the output is used in a program. The value of the status alarm is 1 when the controller is in the `RUNNING` state, and 0 in all other states.
Fallback value	Yes	1 or 0	0	Specifies the value to apply to this output (fallback to 0 or fallback to 1) when the logic controller enters the `STOPPED` or an exception state. The default value is 0. If **Maintain values** fallback mode is configured, the output retains its current value when the logic controller enters the `STOPPED` or an exception state. This field is disabled for the output configured as **Status Alarm**.
Comment	Yes	–	–	Allows you to associate a comment with the digital output object. Double-click in the **Comment** column, type an optional comment, and press Enter.

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Digital Outputs (%Q) *(see page 167)*.

Section 4.3
Configuring Pulse Generators

Configuring Pulse Generators

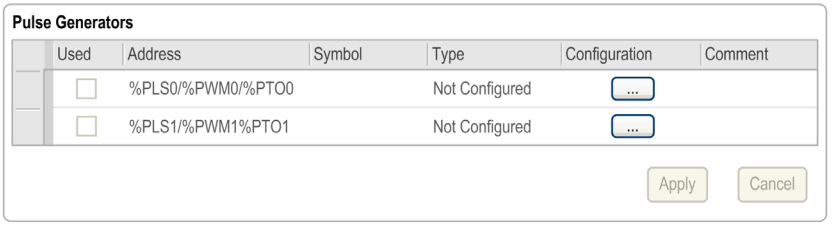
Introduction

The pulse generator function blocks, `Pulse (PLS)`, `Pulse Width Modulation (PWM)`, and `Pulse Train Output (PTO)` are used to generate square or modulated wave signals on dedicated output channels `%Q0.0` or `%Q0.1`.

The PWM outputs feature a modulated wave signal with a variable width and duty cycle while the PTO outputs generate a square wave to control a linear single-axis stepper or servo drive in open loop mode. The PLS also creates a square wave for a programmed number of pulses.

Pulse Generators Configuration

This table describes how to configure the pulse generators:

Step	Action
1	Click the **Pulse Generators** node in the hardware tree to display the pulse generator properties. This figure presents the properties of the pulse generators in the editor area: 
2	Edit the properties to configure the pulse generator output. For detailed information on the pulse generator configuration parameters, refer to the table below.

This table describes each parameter of the pulse generator configuration:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the pulse generated output is being used in a program or not.
Address	No	%PLSx %PWMx %PTOx	%PLSx/%PWMx/ %PTOx	Displays the address of the `Pulse` output, `Pulse Width Modulation` output, or `Pulse Train Output`, where x is the output number. If the controller has 2 pulse generators, then the value for x is 0 and 1. For example, `%PLS0` is the address of the first pulse output on the controller. If you select the output type **PLS**, the **Address** field displays only %PLSx and if you select **PWM**, it displays only %PWMx.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the pulse generator object. Double-click in the **Symbol** column, type the name of the symbol and press **Enter**. **NOTE:** This field is enabled only if the pulse generator is configured.
Type	No	**Not Configured** **PLS** **PWM** **PTO**	**Not Configured**	Displays the type of the pulse generator used for the output channel.
Configuration	Yes	**[...]** (Button)	Enabled	Allows you to choose the type of pulse generator. Click this button to open the **Pulse Generator Assistant** window *(see page 70)* for configuration, where x is the pulse generator number on the controller.
Comment	Yes	–	–	Allows you to specify a comment to associate with the pulse generator object. Double-click in the **Comment** column, type the comment and press **Enter**.

Pulse Generator Assistant Window

This graphic presents the **Pulse Generator Assistant** window:

This table describes the parameter of the **Pulse Generator Assistant** window:

Parameter	Editable	Value	Default Value	Description
General				
Type of pulse generator	Yes	**Not Configured** **PLS** **PWM** **PTO**	**Not Configured**	Allows you to choose the type of pulse generator and configure the output properties. Select: ● **PLS** to configure the output channels in `PLS` mode. Refer to PLS Configuration *(see page 71)*. ● **PWM** to configure the output channels in `PWM` mode. Refer to PWM Configuration *(see page 73)*. ● **PTO** to configure the output channels in `PTO` mode. Refer to PTO Configuration *(see page 74)*.

PLS Configuration

This graphic presents the **Pulse Generator Assistant** window when the **Type of pulse generator** is set to **PLS**:

The table describes each parameter available when the channel is configured in **PLS** mode:

Parameter	Editable	Value	Default Value	Description
Behavior				
Double Word	Yes	False	True/False	Allows you to toggle between the data size of Word (16 bits) and Double Word (32 bits). By default, this parameter is disabled, which indicates that the current data size is Word (16 bits). Enabling this field changes the data size to Double Word (32 bits).
Period				
Time Base	Yes	**0.1 ms** **1 ms** **10 ms** **1 s**	**1 s**	Allows you to select the time base for the frequency measurement.
Preset	Yes	Refer to the table below for complete range of preset values for `PLS` type pulse generator.	0	Allows you to specify the preset value for the pulse output.

This table presents the range of values of the **Preset** parameter:

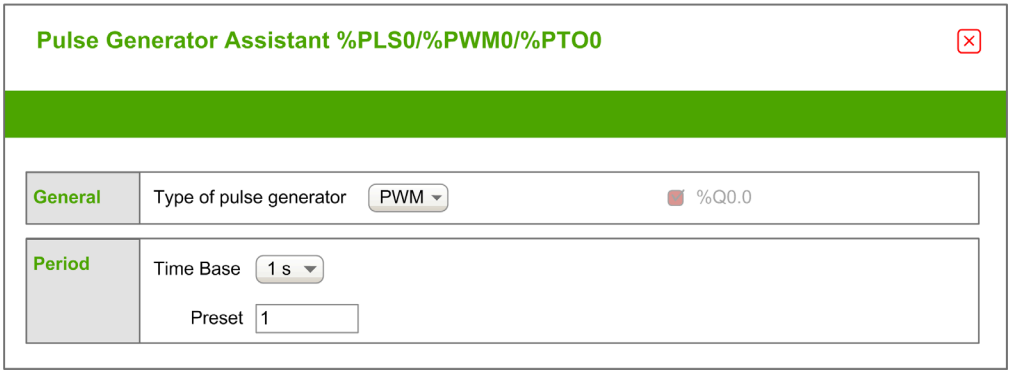
Type	Time Base	Preset Value Range
`PLS`	0.1 ms	1...20000
	1 ms	1...2000
	10 ms	1...200
	1 s	1 or 2

Additional configuration details are displayed in the **Programming** tab.

For more details on the `Pulse` function block, refer to Pulse (%PLS) *(see page 197)*.

PWM Configuration

This graphic presents the **Pulse Generator Assistant** window when the **Type of pulse generator** is set to **PWM**:



The table describes each parameter available when the channel is configured in **PWM** mode:

Parameter	Editable	Value	Default Value	Description
Period				
Time Base	Yes	**0.1 ms** **1 ms** **10 ms** **1 s**	**1 s**	Allows you to select the time base for the frequency measurement.
Preset	Yes	Refer to the table below for complete range of preset values for `PWM` type pulse generator.	0	Allows you to specify the preset value for the `PWM` output.

This table presents the range of values of the **Preset** parameter:

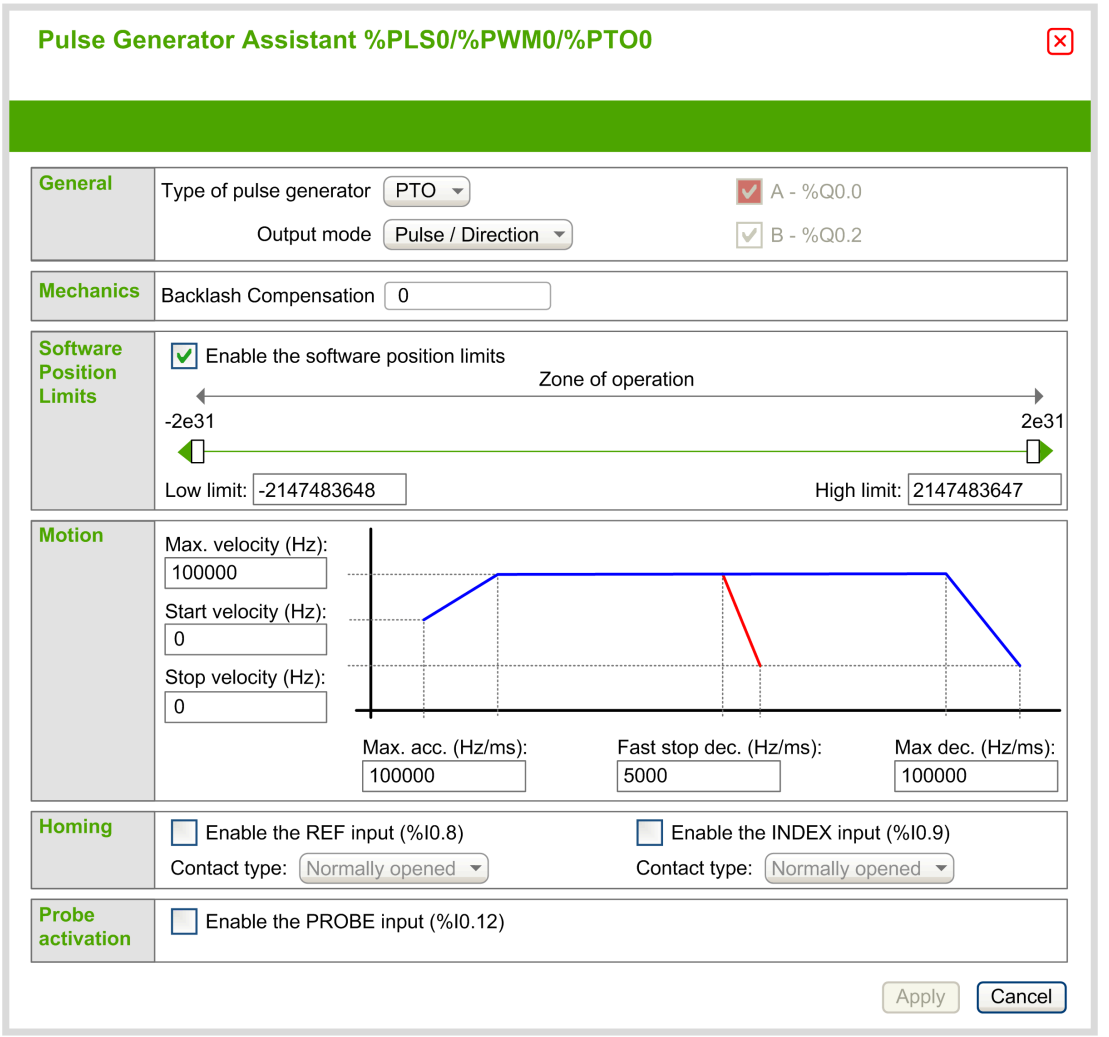
Type	Time Base	Preset Value Range
`PWM`	0.1 ms	1...10000
	1 ms	1...1000
	10 ms	1...100
	1 s	1

Additional configuration details are displayed in the **Programming** tab.

For more details on the `Pulse Width Modulation` function block, refer to Pulse Width Modulation (%PWM) *(see page 205)*.

PTO Configuration

This graphic presents the **Pulse Generator Assistant** window when the **Type of pulse generator** is set to **PTO**:



The table describes each parameter available when the channel is configured in **PTO** mode:

Parameter	Value	Default	Description
General			
Output Mode *(see page 219)*	Clock Wise / Counter Clock Wise Pulse / Direction	Pulse / Direction	Select the pulse output mode. CW = ClockWise / CCW = CounterClockWise **NOTE:** The CW / CCW output mode is only valid for PTO0. This mode disables PTO1.
Software Position Limits			
Enable the software position limits *(see page 227)*	Enabled Disabled	Enabled	Select whether to use the software limits.
Low Limit	-2,147,483,648... 2,147,483,647	-2,147,483,648	Set the software limit position to be detected in the negative direction.
High Limit	-2,147,483,648... 2,147,483,647	2,147,483,647	Set the software limit position to be detected in the positive direction.
Motion			
Maximum Velocity	0...100,000	100,000	Set the pulse output maximum velocity (in Hz).
Start Velocity *(see page 221)*	0...100,000	0	Set the pulse output start velocity (in Hz). 0 if not used.
Stop Velocity *(see page 221)*	0...100,000	0	Set the pulse output stop velocity (in Hz). 0 if not used.
Maximum Acceleration	1...100,000	100,000	Set the acceleration maximum value (in Hz).
Fast Stop Deceleration	1...100,000	5,000	Set the deceleration value in case an error is detected (in Hz)
Maximum Deceleration	1...100,000	100,000	Set the deceleration maximum value (in Hz).
Homing			
Enable the REF input	Enabled Disabled	Disabled	Select whether to use the REF input to set the homing position.
Contact type	Normally opened Normally closed	Normally opened	Select whether the switch contact default state is open or closed. **NOTE:** The input type is only available when the “Enable the REF input” is selected.
Enable the INDEX input (%I0.x)	Enabled Disabled	Disabled	Select whether to use the INDEX input as a positive limit signal.

Parameter	Value	Default	Description
Contact type	Normally opened Normally closed	Normally opened	Select whether the switch contact default state is open or closed. **NOTE:** The input type is only available when the “Enable the INDEX input” is selected.
Probe activation			
Enable the PROBE input	Enabled Disabled	Disabled	Select whether to use the PROBE input.

Additional configuration details are displayed in the **Programming** tab.

For more details on the `Pulse Train Output` function block, refer to Pulse Train Output (%PTO) *(see page 211)*.

Section 4.4
Configuring High Speed Counters

What Is in This Section?

This section contains the following topics:

Configuring High Speed Counters

Introduction

You can configure high speed counters to perform any one of the following functions:

- Single phase
- Dual phase [Pulse / Direction]
- Dual phase [Clock Wise / Counter Clock Wise]
- Dual phase [Quadrature X1]
- Dual phase [Quadrature X2]
- Dual phase [Quadrature X4]
- Frequency meter

Several modes are available with **Single Phase** and **Dual Phase** HSC type counter:

- Single Phase
  - One Shot
  - Modulo Loop
  - Free Large
- Dual Phase
  - Modulo Loop
  - Free Large

In One Shot mode, each pulse applied to the input increments the current value. The counter stops when its current value reaches the maximum value. The overflow flag is then set. For more details, refer to One-shot Counting Mode *(see page 191)*.

In Modulo Loop mode, the counter repeatedly counts from 0 to a user-defined modulo value. It then returns to 0 and restarts counting. In reverse, the counter counts down from the modulo value to 0 then presets to the modulo value and restarts counting. For more details, refer to Modulo-Loop Counting Mode *(see page 192)*.

In Free Large mode, the counter behaves like a standard up and down counter. This mode can be used with one encoder.

The high speed counter supports counting of digital inputs up to frequencies of 100 kHz in single word or double word computational mode.

Dedicated I/O Assignments

The `High Speed Counter` function blocks use dedicated inputs and auxiliary inputs and outputs. These inputs and outputs are not reserved for the exclusive use of `High Speed Counter` function blocks.

If `%I0.0` is in use by the program as a regular digital input, `%HSC0` is not available.

If `%I0.1` is in use by the program as a regular digital input, `%HSC0`and `%HSC2` are not available.

If `%I0.6` is in use by the program as a regular digital input, `%HSC1` is not available.

If `%I0.7` is in use by the program as a regular digital input, `%HSC1` and `%HSC3` are not available.

This table summarizes the assignments for %HSC0 and %HSC1:

Counter type	Main inputs		Auxiliary inputs		Reflex outputs			
					TM200•••U/TM200•••T		TM100•••R / TM200•••R	
%HSC0	**%I0.0**	**%I0.1**	**%I0.2**	**%I0.3**	**%Q0.4**	**%Q0.5**	**%Q0.0**	**%Q0.1**
%HSC1	**%I0.6**	**%I0.7**	**%I0.5**	**%I0.4**	**%Q0.6**	**%Q0.7**	**%Q0.2**	**%Q0.3**
Single phase	Pulse input	Not used	Preset input*	Catch input*	Reflex output R*	Reflex output S*	Reflex output R*	Reflex output S*
Dual phase [Pulse / Direction]	Pulse input	Direction input	Preset input*	Catch input*	Reflex output R*	Reflex output S*	Reflex output R*	Reflex output S*
Dual phase [Clock Wise / Counter Clock Wise]	CW input Phase A	CCW input Phase B	Preset input*	Catch input*	Reflex output R*	Reflex output S*	Reflex output R*	Reflex output S*
Dual phase [Quadrature X1]	Pulse input Phase A	Pulse input Phase B	Preset input*	Catch input*	Reflex output R*	Reflex output S*	Reflex output R*	Reflex output S*
Dual phase [Quadrature X2]	Pulse input Phase A	Pulse input Phase B	Preset input*	Catch input*	Reflex output R*	Reflex output S*	Reflex output R*	Reflex output S*
Dual phase [Quadrature X4]	Pulse input Phase A	Pulse input Phase B	Preset input*	Catch input*	Reflex output R*	Reflex output S*	Reflex output R*	Reflex output S*
Frequency meter	Pulse input	Not used	Not used	Not used	Not used	Not used	Not used	Not used

* When not used, the input or output functions as a normal digital I/O available to be managed by the application in the main task cycle.

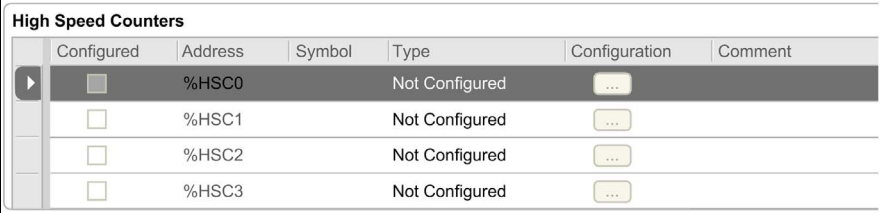
This table summarizes the assignments for %HSC2 and %HSC3:

Counter type	Main inputs		Auxiliary inputs		Reflex outputs			
					TM200•••U/TM200•••T		TM100•••R / TM200•••R	
%HSC2	**%I0.0**	**%I0.1**	**%I0.2**	**%I0.3**	**%Q0.4**	**%Q0.5**	**%Q0.0**	**%Q0.1**
%HSC3	**%I0.6**	**%I0.7**	**%I0.5**	**%I0.4**	**%Q0.6**	**%Q0.7**	**%Q0.2**	**%Q0.3**
Single phase	Not used	Pulse input	Not used	Not used	Not used	Not used	Not used	Not used

* When not used, the input or output functions as a normal digital I/O available to be managed by the application in the main task cycle.

High Speed Counters Configuration

This table describes how to configure the high speed counters:

Step	Action
1	Click the **High Speed Counters** node in the hardware tree to display the high speed counter properties. **Result:** The properties of the high speed counters are displayed in the editor area:  For detailed information on the high speed counter configuration, refer to the following table.
2	Click the **[...]** button to configure a high speed counter. **Result:** The high speed counter configuration parameters are displayed in the assistant window. For more details on configuration of single phase and dual phase modes with the assistant, refer to Configuring Single Phase and Dual Phase *(see page 82)*. For more details on configuration of frequency meter mode with the assistant, refer to Configuring Frequency Meter *(see page 88)*.

This table describes each parameter of the high speed counters configuration:

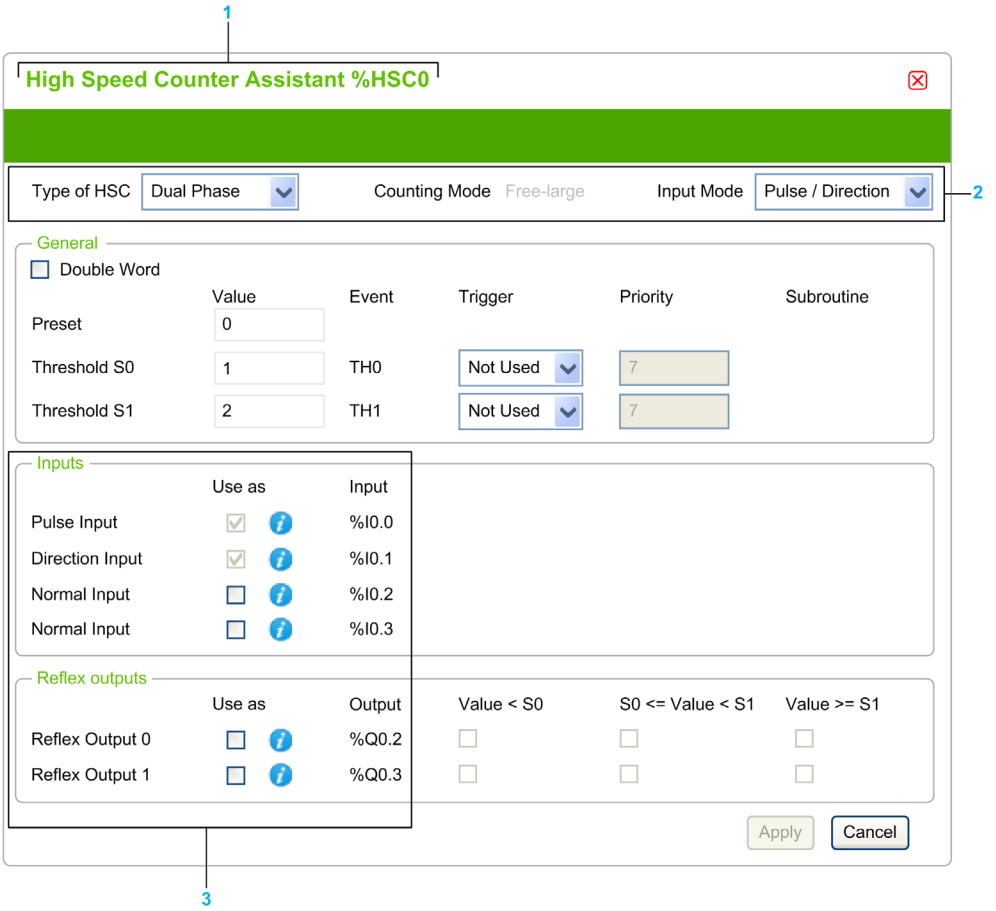
Parameter	Editable	Value	Default value	Description
Used	No	True/False	False	Indicates whether the high speed counter is being used in a program or not.
Address	No	%HSCx		Displays the address of the high speed counter, where x is the object number. For example, `%HSC1` is the address of the second high speed counter on the controller.
Symbol	Yes	–	–	Allows you to associate a symbol with the high speed counter object. Double-click in the **Symbol** column, type the name of the symbol, and press Enter.
Type	Yes	**Not Configured** **Single Phase** **Dual Phase** **Frequency Meter**	**Not Configured**	Allows you to select the counter operational mode from the drop-down list.

Parameter	Editable	Value	Default value	Description
Configuration	Yes	**[...]** (Button)	Disabled	Allows you to configure the high speed counter parameters using an assistant window. This button is enabled only if the high speed counter function is selected from the list. When the counter type is **Not Configured**, the assistant configuration button is disabled. The **High Speed Counter Assistant (%HSCx)** window appears when you click the configuration button, where x is the counter number on the controller.
Comment	Yes	–	–	Allows you to associate a comment with the high speed counter object. Double-click in the **Comment** column, type an optional comment, and press Enter.

Configuring Single Phase and Dual Phase

Single Phase and Dual Phase with Assistant Window

This figure presents an instance of the assistant window for %HSC0 configured as **Dual Phase**:



Item	Description
1	Displays the title of the assistant dialog window. If you are configuring the counter %HSC0, the window title appears as **High Speed Counter Assistant (%HSC0)** and for the counter %HSC1, the window title appears as **High Speed Counter Assistant (%HSC1)**.

Item	Description
2	Displays the input mode of particular type of high speed counter.
3	Displays the dedicated inputs, auxiliary inputs, and reflex outputs. Properties in this area of the assistant window are different for each counter type and %HSC instance. For more details, refer to the Dedicated I/O Assignments *(see page 78)* section.

This table describes each parameter of the assistant screen for high speed counters for both %HSC0 and %HSC1:

Parameter	Editable	Value	Default value	Description
Pre-Configuration				
Type of HSC	Yes	**Not Configured** **Single Phase** **Dual Phase** **Frequency Meter**	**Not Configured**	Indicates the selected counter operational mode and allows you to change it.
Counting Mode when configuring single phase	Yes	**Free Large** **Modulo Loop** **One Shot**	**Free Large**	Indicates the selected counter operational mode and allows you to change it. The options depend on the instance and on the type of HSC in the other instances. Refer to Dedicated I/O Assignments *(see page 78)*.
Counting Mode when configuring dual phase	Yes	**Free Large** **Modulo Loop**	**Free Large**	Indicates the selected counter operational mode and allows you to change it. The options depend on the instance and on the type of HSC in the other instances. Refer to Dedicated I/O Assignments *(see page 78)*.
Input Mode when configuring dual phase	Yes	**Pulse / Direction** **Clock Wise / Counter Clock Wise** **Quadrature X1** **Quadrature X2** **Quadrature X4**	**Pulse / Direction**	Indicates the selected counter operational mode and allows you to change it. The options depend on the instance and on the type of HSC in the other instances. Refer to Dedicated I/O Assignments.
General				
Double Word	Yes	True/False	False	Allows you to toggle between input data size of Word (16 bits) and Double Word (32 bits). By default, this parameter is disabled, which indicates that the current data size is Word (16 bits). Enabling this field changes the data size to Double Word (32 bits).
* for **X1** the value is limited to 2147483647				

Parameter	Editable	Value	Default value	Description
Preset	Yes	0...65535 (Word) 0...4294967295* (Double Word) when configuring **Dual Phase** **[Quadrature X1]**	0 (Word) 0 (Double Word)	Allows you to specify the preset value for the counting functions.
Modulo Value (%HSC.P/.PD)	Yes	0...65535 (Word) 0...4294967295* (Double Word) 0...2147483647 (Double Word) when configuring Dual Phase [Quadrature X1]	0 (Word) 0 (Double Word)	Allows you to specify the modulo value for the counting functions.
Threshold S0	Yes	0...65535 (Word) 0...4294967295* (Double Word) 0...2147483647 (Double Word) when configuring Dual Phase [Quadrature X1]	65535 (Word) 4294967295* (Double Word)	Allows you to specify the value of the HSC flag S0 that contains the value of the threshold TH0.
Threshold S1	Yes	0...65535 (Word) 0...4294967295* (Double Word) 0...2147483647 (Double Word) when configuring Dual Phase [Quadrature X1]	65535 (Word) 4294967295* (Double Word)	Allows you to specify the value for the HSC flag S1 that contains the value of the threshold TH1.
Trigger	Yes	**Not Used** **Falling Edge** **Rising Edge** **Both edges**	**Not Used**	Allows you to select a triggering function for an event (for both threshold TH0 and TH1) from the drop-down list. If you select a triggering function from the drop-down list (other than **Not Used**), the **Priority** parameter enables for editing to set the priority of the event.
* for **X1** the value is limited to 2147483647				

Parameter	Editable	Value	Default value	Description
Priority	Yes	0...7	7	Allows you to set the priority of the triggering function of an event (for both threshold TH0 and TH1). This field enables only when you select a **Trigger** function for the event.
Subroutine	No	*Any*	*Empty*	Displays the subroutine associated with an input configured as an event (for both threshold TH0 and TH1).
Inputs Configuration				
When configuring **Single Phase**:				
Pulse Input	No	True/False	True	● For `%HSC0`:`%I0.0` is used as pulse input. ● For `%HSC1`:`%I0.6` is used as pulse input.
When configuring **Dual Phase [Pulse / Direction]**:				
Pulse Input	No	True/False	True	● For `%HSC0`:`%I0.0` is used as pulse input. ● For `%HSC1`:`%I0.6` is used as pulse input.
Direction Input	No	True/False	True	● For `%HSC0`:`%I0.1` is used as directional input. ● For `%HSC1`:`%I0.7` is used as directional input.
When configuring **Dual Phase [Clock Wise / Counter Clock Wise]**:				
Clock Wise Phase A	No	True/False	True	● For `%HSC0`:`%I0.0` is used as pulse input for phase A. ● For `%HSC1`:`%I0.6` is used as pulse input for phase A.
Counter Clock Wise Phase B	No	True/False	True	● For `%HSC0`:`%I0.1` is used as pulse input for phase B. ● For `%HSC1`:`%I0.7` is used as pulse input for phase B.
When configuring **Dual Phase [Quadrature X1]**, **Dual Phase [Quadrature X2]**, and **Dual Phase [Quadrature X4]**:				
Pulse Input Phase A	No	True/False	True	● For `%HSC0`:`%I0.0` is used as pulse input for phase A. ● For `%HSC1`:`%I0.6` is used as pulse input for phase A.
Pulse Input Phase B	No	True/False	True	● For `%HSC0`:`%I0.1` is used as pulse input for phase B. ● For `%HSC1`:`%I0.7` is used as pulse input for phase B.
* for **X1** the value is limited to 2147483647				

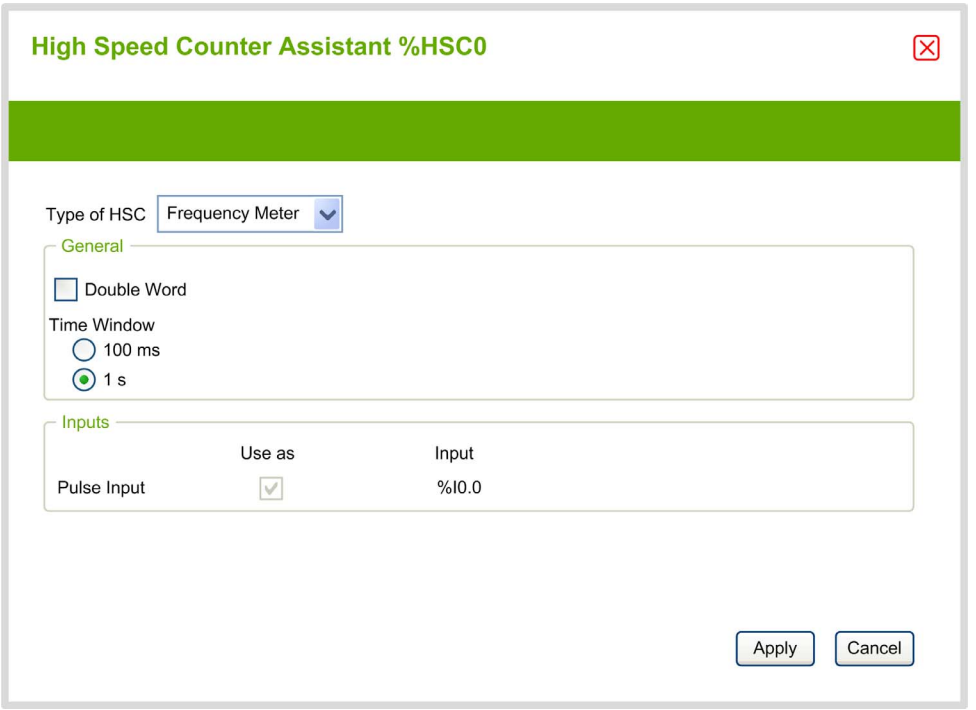
Parameter	Editable	Value	Default value	Description
When configuring **Single Phase** and **Dual Phase**:				
Normal Input	Yes	True/False	False	• For %HSC0: %I0.2 is used as normal input. • For %HSC1: %I0.5 is used as normal input. Click the **Use as** check box to use this input as **Preset Input**.
Normal Input	Yes	True/False	False	• For %HSC0: %I0.3 is used as normal input. • For %HSC1: %I0.4 is used as normal input. Click the **Use as** check box to use this input as **Catch Input**.
Reflex Outputs Configuration				
Reflex Output 0	No	True/False	False	Allows you to enable or disable the reflex output at the address: For TM200•••U/TM200•••T: • %Q0.4 for %HSC0 • %Q0.6 for %HSC1 For TM100•••R/TM200•••R: • %Q0.0 for %HSC0 • %Q0.2 for %HSC1
Reflex Output 1	No	True/False	False	Allows you to enable or disable the reflex output at the address: For TM200•••U/TM200•••T: • %Q0.5 for %HSC0 • %Q0.7 for %HSC1 For TM100•••R/TM200•••R: • %Q0.1 for %HSC0 • %Q0.3 for %HSC1
Value < S0	Yes	True/False	False	Allows you to enable or disable the condition in which the counter is constantly compared to the output value to set the reflex output when the output value is less than the value of HSC flag S0.
* for **X1** the value is limited to 2147483647				

Parameter	Editable	Value	Default value	Description
S0 <= Value < S1	Yes	True/False	False	Allows you to enable or disable the condition in which the counter is constantly compared to the output value to set the reflex output when the output value is: ● Greater than or equal to the value of the HSC flag S0 and ● Less than the value of the HSC flag S1. **NOTE:** S1 is not a variable when configuring Modulo-loop mode.
Value >= S1	Yes	True/False	False	Allows you to enable or disable the condition in which the counter is constantly compared to the output value to set the reflex output when the output value is greater than or equal to the value of HSC flag S1. **NOTE:** S1 is not a variable when configuring Modulo-loop mode.
* for **X1** the value is limited to 2147483647				

Configuring Frequency Meter

Frequency Meter Assistant Window

This figure presents the **High Speed Counter Assistant (%HSC0)** window for the counter type **Frequency Meter**:



This table describes each parameter of the **High Speed Counter Assistant (%HSCx)** window for the counter type **Frequency Meter**:

Parameter	Editable	Value	Default value	Description
Type of HSC	Yes	**Not Configured** **Single Phase** **Dual Phase** **Frequency Meter**	**Not Configured**	Indicates the selected counter operational mode and allows you to change it. The Frequency Meter is configurable on `%HSC0` and/or `%HSC1`. Refer to the Frequency Meter I/O Assignment.
(1) By default, the time base value is set to 1 s. You can select only 1 time base value for the counter.				

Parameter	Editable	Value	Default value	Description
General				
Double Word	Yes	TRUE/FALSE	FALSE	Allows you to toggle between the input data size from Word (16 bits) to Double Word (32 bits). Enabling this field changes the data size from Word (16 bits) to Double Word (32 bits).
Time Window 100 ms (1)	Yes	TRUE/FALSE	FALSE	Allows you to select the time base of 100 ms to measure the frequency between 100 Hz and 100 kHz.
Time Window 1 s (1)	Yes	TRUE/FALSE	TRUE	Allows you to select the time base of 1 s to measure the frequency between 100 Hz and 100 kHz.
Inputs				
Pulse Input	No	TRUE/FALSE	TRUE	Indicates the input used as pulse input, `%I0.0` for `%HSC0` or `%I0.6` for `%HSC1`.
(1) By default, the time base value is set to 1 s. You can select only 1 time base value for the counter.				

Additional configuration details are displayed in the **Programming** tab.

For more details on the `High Speed Counter` function block, refer to High Speed Counter Function Block (%HSC) *(see page 178)*.

Chapter 5
I/O Bus Configuration

Overview

This chapter describes how to configure the I/O bus (expansion modules) of the M100/M200 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

I/O Configuration General Practices

Match Software and Hardware Configuration

The I/O that may be embedded in your controller is independent of the I/O that you may have added in the form of I/O expansion. It is crucial that the logical I/O configuration within your program matches the physical I/O configuration of your installation. If you add or remove any physical I/O to or from the I/O expansion bus, or, depending on the controller reference, to or from the controller (in the form of cartridges), it is imperative that you update your application configuration. This is also true for any field bus devices you may have in your installation. Otherwise, there is the possibility that the I/O expansions will no longer function while the embedded I/O that may be present in your controller will continue to operate.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Update the configuration of your program each time you add or delete any type of I/O expansions on your I/O bus, or you add or delete any devices on your field bus.

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

Presentation of the Optional Feature for I/O Expansion Modules

I/O expansion modules can be marked as optional in the configuration. The **Optional module** feature provides a more flexible configurtion by the acceptance of the definition of modules that are not physically attached to the logic controller. Therefore, a single application can support multiple physical configurations of I/O expansion modules, allowing a greater degree of scalability without the necessity of maintaining multiple application files for the same application.

You must be fully aware of the implications and impacts of marking I/O modules as optional in your application, both when those modules are physically absent and present when running your machine or process. Be sure to include this feature in your risk analysis.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Include in your risk analysis each of the variations of I/O configurations that can be realized marking I/O expansion modules as optional, and in particular the establishment of TM3 Safety modules (TM3S…) as optional I/O modules, and make a determination whether it is acceptable as it relates to your application.

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

NOTE: For more details about this feature, refer to Optional I/O Expansion Modules *(see page 55)*.

Configuring Expansion Modules

Introduction

In your project, you can add the following devices to the controller:

- TM3 Digital I/O Modules
- TM3R Digital Mixed I/O Modules
- TM3 Analog I/O Modules
- TM2 Digital I/O Modules
- TM2 Analog I/O Modules

TM3/TM3R Expansion Modules

For more information about module configuration, refer to the following programming and hardware guides of each expansion module type:

Expansion module type	Hardware Guide	Programming Guide
TM3 Digital I/O Expansion Modules	TM3 Digital I/O Expansion Modules Hardware Guide	TM3 Expansion Modules Programming Guide
TM3R Digital Mixed I/O Expansion Modules	Modicon M100/M200 Logic Controller Hardware Guide	TM3R Expansion Module Configuration *(see page 125)*
TM3 Analog I/O Expansion Modules	TM3 Analog Modules Hardware Guide	TM3 Expansion Modules Programming Guide

TM2 Expansion Modules

For more information about module configuration, refer to the programming and hardware guides of each expansion module type:

Expansion module type	Hardware Guide	Programming Guide
TM2 Digital I/O Modules	TM2 Digital I/O Modules Hardware Guide	TM2 Expansion Modules Programming Guide
TM2 Analog I/O Modules	TM2 Analog I/O Modules Hardware Guide	

Chapter 6
Cartridge Configuration

Overview

This chapter describes how to configure the cartridges of the M100/M200 Logic Controller.

What Is in This Chapter?

This chapter contains the following sections:

Section 6.1
Cartridge Configuration General Information

What Is in This Section?

This section contains the following topics:

General Description

Introduction

The TMCR2••• cartridges connect to Modicon M100/M200 Logic Controllers to increase the number of I/Os or serial lines available on the controller.

Cartridges can be:
- Digital cartridges
- Analog cartridges
- Serial line cartridges

Cartridges Features

The following table describes the TMCR2••• cartridge features:

Reference	Description
TMCR2DM4U	TMCR2 cartridge with 2 digital inputs and 2 transistor sink outputs
TMCR2AI2	TMCR2 cartridge with 2 analog voltage or current inputs (0...10 V, 0...20 mA, 4...20 mA), 12 bits
TMCR2AQ2C	TMCR2 cartridge with 2 analog current outputs (4...20 mA), 12 bits
TMCR2AQ2V	TMCR2 cartridge with 2 analog voltage outputs (0...10 V), 12 bits
TMCR2AM3	TMCR2 cartridge with 2 analog voltage inputs and 1 analog voltage output (0...10 V), 16 bits
TMCR2TI2	TMCR2 cartridge with 2 analog temperature inputs (thermocouple, RTD), 14 bits
TMCR2SL1	TMCR2 cartridge with 1 serial line (RS-232 or RS-485)
TMCR2SL1A	TMCR2 cartridge with 1 isolated serial line (RS-485)

Logic Controller Compatibility

NOTE: For more information on cartridge compatibility with specific controllers, refer to the Modicon M100/M200 Logic Controller hardware guide *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

The following table describes the number of TMCR2••• cartridges that can be installed in a Modicon M100/M200 Logic Controller:

Reference	Cartridge Slots	Compatible Cartridges Combination	
		TMCR2DM4U TMCR2AI2 TMCR2AQ2V TMCR2AQ2C TMCR2AM3 TMCR2TI2	**TMCR2SL1 TMCR2SL1A**
TM200C•16• TM200C•24•	1	1	0
		0	1
TM200C•40• TM200C•60•	2 (1)	1	0
		0	1
		1	1
		2	0
(1) Only one serial line cartridge (TMCR2SL1 or TMCR2SL1A) may be added to a logic controller.			

NOTICE

ELECTROSTATIC DISCHARGE

- Verify that empty cartridge slots have their covers in place before applying power to the controller.
- Do not touch the contacts of the cartridge.
- Only handle the cartridge on the housing.
- Take the necessary protective measures against electrostatic discharges.

Failure to follow these instructions can result in equipment damage.

Using Cartridges in a Configuration

Adding a Cartridge

TMCR2••• cartridges can be connected to the M100/M200 Logic Controller with 1 or 2 cartridge slots.

NOTE: It is not possible to add 2 serial line cartridges to the same logic controller. For more information on cartridge compatibility with the M100/M200 Logic Controller, refer to Logic Controller Compatibility *(see page 98)*.

NOTE: The controller must have at least one free cartridge slot.

The following steps explain how to add a cartridge to a logic controller in a SoMachine Basic configuration:

Step	Description
1	Click the **Configuration** tab in the SoMachine Basic window.
2	In the hardware catalog area of the window, select **M200 Cartridges**.
3	Select a cartridge reference. **Result**: A description of the physical characteristics of the selected cartridge appears in the bottom right-hand corner of the SoMachine Basic window.
4	Drag and drop the cartridge onto an empty cartridge slot of the M100/M200 Logic Controller. **Result**: The cartridge is added to the **MyController** → **IO Bus** area of the device tree. For serial line cartridges, the **SL2 (Serial line)** node appears. For analog cartridges, the **Analog inputs** or **Analog outputs** subnode appears immediately below the cartridge reference. The following information about the selected cartridge is displayed in the lower central area of the SoMachine Basic window: ● Information about the current status of the cartridge. ● For application cartridges, a list of project templates available for the cartridge.

Replacing an Existing Cartridge

To replace an existing cartridge with a difference reference, drag and drop the new cartridge onto the cartridge to be replaced.

A message appears asking you to confirm the operation. Click **Yes** to continue.

Removing a Cartridge

To remove a cartridge from a controller, either click the cartridge and press the **Delete** key, or right-click on the cartridge and click **Remove** on the contextual menu that appears.

If the cartridge contains at least one address being used in the user logic of the program, a message appears asking you to confirm the operation. Click **Yes** to continue.

Configuring Cartridges

Overview

You can configure cartridges on:

- The **Configuration** tab
- The **Programming** tab

Displaying Configuration Details

The steps below describe how to view the configuration of digital inputs on the **Configuration** tab:

Step	Description
1	Select the **Configuration** tab.
2	For digital cartridges, select **Cartridge x** → **Digital inputs** or **Cartridge x** → **Digital outputs** in the device tree on the left of the SoMachine Basic window. For analog cartridges, select **Cartridge x** → **Analog inputs** or **Cartridge x** → **Analog outputs** in the device tree on the left of the SoMachine Basic window. For serial line cartridges, select **Cartridge x** → **SL2 (Serial line)** in the device tree on the left of the SoMachine Basic window. The properties of the selected cartridge are displayed.
3	Refer to TMCR2 Standard Cartridges Configuration *(see page 102)* for configuration details.

Displaying Programming Properties

The **Programming** tab allows you to configure programming-related properties of digital or analog cartridges, such as symbols and comments.

To display cartridge properties in the **Programming** tab:

Step	Description
1	Select the **Programming** tab.
2	For digital cartridges, click **Tools** → **I/O objects** → **Digital inputs** or **Tools** → **I/O objects** → **Digital outputs** For analog cartridges, click **Tools** → **I/O objects** → **Analog inputs** or **Tools** → **I/O objects** → **Analog outputs** A list of I/O addresses appears in the lower central area of the SoMachine Basic window (%I for digital inputs, %Q for digital outputs, %IW for analog inputs, %QW for analog outputs).

Step	Description
3	Scroll down to the range of addresses corresponding to the cartridge you are configuring. The following properties are displayed: ● **Used**. Whether the address is being used in your program ● **Address**. The analog input or analog output address. ● **Symbol**. An optional symbol associated with the address. Double-click in the **Symbol** column and type the name of a symbol to associate with this input. If a symbol already exists, right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol in the application. ● **Comment**. An optional comment associated with the address. Double-click in the **Comment** column and type a comment to associate with this address.

Section 6.2
TMCR2••• Cartridges Configuration

What Is in This Section?

This section contains the following topics:

TMCR2DM4U

Introduction

The TMCR2DM4U is a standard M200 cartridge featuring 2 digital sink/source inputs and 2 digital transistor sink outputs.

For further hardware information, refer to TMCR2DM4U *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

Configuring the Cartridge Module

For each digital input, you can define:

Parameter	Value	Default value	Description
Used	True/False	False	Indicates whether the address is being used in a program.
Address	`%I0.x0y`	-	Shows the address of the input channel, where *x* is the cartridge number and *y* is the channel number
Symbol		-	Specify a optional symbol to associate with the corresponding digital input object to be used in the program.
Comment		-	Type an optional comment to associate with the corresponding digital input object.

For each digital output, you can define:

Parameter	Value	Default value	Description
Used	True/False	False	Indicates whether the address is being used in a program.
Address	`%Q0.x0y`	-	Shows the address of the output channel, where *x* is the cartridge number and *y* is the channel number
Symbol		-	Specify a optional symbol to associate with the corresponding digital output object to be used in the program.
Fallback value	**0**...**1**	0	Specifies the fallback value of the output channel.
Comment		-	Type an optional comment to associate with the corresponding digital output object.

TMCR2AI2

Introduction

The TMCR2AI2 is a standard cartridge featuring 2 analog voltage or current input channels with 12-bit resolution.

The channel input types are:

- 0...10 V
- 0...20 mA
- 4...20 mA

For further hardware information, refer to TMCR2AI2 *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

If you have physically wired, for example, your analog channel for a voltage signal and you configure the channel for a current signal in SoMachine Basic, you may damage the analog circuit.

NOTICE
INOPERABLE EQUIPMENT Verify that the physical wiring of the analog circuit is compatible with the software configuration for the analog channel. **Failure to follow these instructions can result in equipment damage.**

Configuring the Module

For each input, you can define:

Parameter		Value	Default value	Description
Used		True/False	False	Indicates whether the address is being used in a program.
Address		`%IW0.x0y`	-	The address of the input channel, where *x* is the cartridge number and *y* is the channel number
Symbol		-	-	Double-click and type an optional symbol to associate with the corresponding analog input object to be used in the program.
Type		**0 - 10 V** **0 - 20 mA** **4 - 20 mA**	**0 - 10 V**	Select the mode of the channel.
Scope		**Normal**	**Normal**	The range of values for a channel.
Min.	**0 - 10 V**	-32768...32767	0	Specifies the lower measurement limit.
	0 - 20 mA		0	
	4 - 20 mA		4000	

Parameter		Value	Default value	Description
Max.	**0 - 10 V**	-32768...32767	10000	Specifies the upper measurement limit.
	0 - 20 mA		20000	
	4 - 20 mA		20000	
Filter		0...100	0	Specifies the filtering value. Multiply by the **Filter Unit** value to obtain the filtering time.
Filter Unit		100 ms	100 ms	Specifies the unit of time for the filtering value.
Units		-	-	-
Comment		-	-	Double-click and type an optional comment to associate with the channel.

TMCR2AQ2C

Introduction

The TMCR2AQ2C is a standard cartridge featuring 2 analog current output channels with 12-bit resolution.

The channel output types are:

- 4...20 mA

For further hardware information, refer to TMCR2AQ2C *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

If you have physically wired, for example, your analog channel for a voltage signal and you configure the channel for a current signal in SoMachine Basic, you may damage the analog circuit.

NOTICE
INOPERABLE EQUIPMENT Verify that the physical wiring of the analog circuit is compatible with the software configuration for the analog channel. **Failure to follow these instructions can result in equipment damage.**

Configuring the Cartridge Module

For each output, you can define:

Parameter	Value	Default value	Description
Used	True/False	False	Indicates whether the address is being used in a program.
Address	`%QW0.x0y`	-	Shows the address of the output channel, where *x* is the cartridge number and *y* is the channel number
Symbol	-	-	Double-click and type an optional symbol to associate with the corresponding analog output object to be used in the program.
Type	**4 - 20 mA**	**4 - 20 mA**	The mode of the channel.
Scope	**Normal**	**Normal**	The range of values for a channel.
Minimum	-32768...32767	4000	Specifies the lower measurement limit.
Maximum	-32768...32767	20000	Specifies the upper measurement limit.
Fallback value	**Min.**...**Max.**	0 (**Min.** if 0 is not in the range)	Specifies the fallback value of the output channel.
Units	-	-	-
Comment	-	-	Double-click and type an optional comment to associate with the channel.

TMCR2AQ2V

Introduction

The TMCR2AQ2V is a standard cartridge featuring 2 analog voltage output channels with 12-bit resolution.

The channel output types are:

- 0...10 V

For further hardware information, refer to TMCR2AQ2V *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

If you have physically wired, for example, your analog channel for a voltage signal and you configure the channel for a current signal in SoMachine Basic, you may damage the analog circuit.

NOTICE
INOPERABLE EQUIPMENT Verify that the physical wiring of the analog circuit is compatible with the software configuration for the analog channel. **Failure to follow these instructions can result in equipment damage.**

Configuring the Cartridge Module

For each output, you can define:

Parameter	Value	Default value	Description
Used	True/False	False	Indicates whether the address is being used in a program.
Address	`%QW0.x0y`	-	Shows the address of the output channel, where *x* is the cartridge number and *y* is the channel number
Symbol	-	-	Double-click and type an optional symbol to associate with the corresponding analog output object to be used in the program.
Type	**0 - 10 V**	**0 - 10 V**	The mode of the channel.
Scope	**Normal**	**Normal**	The range of values for a channel.
Minimum	-32768...32767	0	Specifies the lower measurement limit.
Maximum	-32768...32767	10000	Specifies the upper measurement limit.
Fallback value	**Min.**...**Max.**	0 (**Min.** if 0 is not in the range)	Specifies the fallback value of the output channel.
Units	-	-	-
Comment	-	-	Double-click and type an optional comment to associate with the channel.

TMCR2AM3

Introduction

The TMCR2AM3 is a standard M200 cartridge featuring 2 analog current or voltage input channels and 1 analog current or voltage output channels with 16-bit resolution.

The channel input types are:
- 0...5V
- 0...10 V
- 0...20 mA
- 4...20 mA

The channel output types are:
- 0...5V
- 0...10 V
- 0...20 mA
- 4...20 mA

For further hardware information, refer to TMCR2AM3 *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

If you have physically wired, for example, your analog channel for a voltage signal and you configure the channel for a current signal in SoMachine Basic, you may damage the analog circuit.

NOTICE
INOPERABLE EQUIPMENT Verify that the physical wiring of the analog circuit is compatible with the software configuration for the analog channel. **Failure to follow these instructions can result in equipment damage.**

Configuring the Cartridge Module

For each input, you can define:

Parameter	Value	Default value	Description
Used	True/False	False	Indicates whether the address is being used in a program.
Address	`%IW0.x0y`	-	Shows the address of the input channel, where *x* is the cartridge number and *y* is the channel number
Symbol	-	-	Double-click and type an optional symbol to associate with the corresponding analog input object to be used in the program.

Parameter	Value	Default value	Description
Type	**0 - 5 V** **0 - 10 V** **0 - 20 mA** **4 - 20 mA**	**0 - 5 V**	The mode of the channel.
Scope	**Customized** **Normal**	**Customized**	The range of values for the channel.
Minimum	-32768...32767	4000	Specifies the lower measurement limit.
Maximum	-32768...32767	20000	Specifies the upper measurement limit.
Filter	0...6	0	Specifies the filtering level to apply on this channel: ● 0: No filtering ● 1, 2: Smooth filtering ● 3, 4: Average filtering ● 5, 6: High filtering
Filter Unit		-	-
Units		-	-
Comment			Double-click and type an optional comment to associate with the channel.

For the output, you can define:

Parameter	Value	Default value	Description
Used	True/False	False	Indicates whether the address is being used in a program.
Address	`%QW0.x0y`	-	Shows the address of the output channel, where *x* is the cartridge number and *y* is the channel number
Symbol	-	-	Double-click and type an optional symbol to associate with the corresponding analog output object to be used in the program.
Type	**0 - 5 V** **0 - 10 V** **0 - 20 mA** **4 - 20 mA**	**0 - 5 V**	The mode of the channel.
Scope	**Customized** **Normal**	**Customized**	The range of values for a channel.
Minimum	-32768...32767	4000	Specifies the lower measurement limit.
Maximum	-32768...32767	20000	Specifies the upper measurement limit.
Fallback value	**Min.**...**Max.**	0 (**Min.** if 0 is not in the range)	Specifies the fallback value of the output channel.
Units		-	-
Comment			Double-click and type an optional comment to associate with the channel.

TMCR2TI2

Introduction

The TMCR2TI2 is a standard cartridge featuring 2 analog input channels with 14-bit resolution.

The channel input types are:

- K Thermocouple
- J Thermocouple
- R Thermocouple
- S Thermocouple
- B Thermocouple
- E Thermocouple
- T Thermocouple
- N Thermocouple
- C Thermocouple
- PT100
- PT1000
- NI100
- NI1000

For further hardware information, refer to TMCR2TI2 *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

If you have physically wired, for example, your analog channel for a voltage signal and you configure the channel for a current signal in SoMachine Basic, you may damage the analog circuit.

NOTICE
INOPERABLE EQUIPMENT Verify that the physical wiring of the analog circuit is compatible with the software configuration for the analog channel. **Failure to follow these instructions can result in equipment damage.**

Configuring the Module

For each input, you can define:

Parameter	Value	Default value	Description
Used	True/False	False	Indicates whether the address is being used in a program.
Address	`%IW0.x0y`	-	The address of the input channel, where *x* is the module number and *y* is the channel number

Parameter	Value	Default value	Description
Symbol	-	-	Double-click and type an optional symbol to associate with the corresponding analog input object to be used in the program./P
Type	**K Thermocouple** **J Thermocouple** **R Thermocouple** **S Thermocouple** **B Thermocouple** **E Thermocouple** **T Thermocouple** **N Thermocouple** **C Thermocouple** **PT100** **PT1000** **NI100** **NI1000**	**K Thermocouple**	Choose the mode of the channel.
Scope	**Normal** **Celsius (0.1°C)** **Fahrenheit (0.1°F)** (except Thermocouple B and C) **Fahrenheit (0.2°F)** (for Thermocouple B and C only)	**Normal**	Choose the temperature units for a channel.
Minimum	See the table below		Specifies the lower measurement limit.
Maximum	See the table below		Specifies the upper measurement limit.
Filter	0...100	0	Specifies the filtering value. Multiply by the **Filter Unit** value to obtain the filtering time.
Filter Unit	100 ms	100 ms	Specifies the unit of time for the filtering value.
Units	See the table below		Displays the temperature unit configured.
Comment	-	-	Double-click and type an optional symbol to associate with the corresponding analog input object to be used in the program.

Type	Customized		Celsius			Fahrenheit		
	Min.	Max.	Min.	Max.	Units	Min.	Max.	Units
K Thermocouple	-32768	32767	-2000	13000	0.1 °C	-3280	23720	0.1 °F
J Thermocouple	-32768	32767	-2000	10000	0.1 °C	-3280	18320	0.1 °F
R Thermocouple	-32768	32767	0	17600	0.1 °C	320	32000	0.1 °F
S Thermocouple	-32768	32767	0	17600	0.1 °C	320	32000	0.1 °F
B Thermocouple	-32768	32767	0	18200	0.1 °C	160	16540	0.2 °F
E Thermocouple	-32768	32767	-2000	8000	0.1 °C	-3280	14720	0.1 °F
T Thermocouple	-32768	32767	-2000	4000	0.1 °C	-3280	7520	0.1 °F
N Thermocouple	-32768	32767	-2000	13000	0.1 °C	-3280	23720	0.1 °F
C Thermocouple	-32768	32767	0	23150	0.1 °C	160	20995	0.2 °F
PT100	-32768	32767	-2000	8500	0.1 °C	-3280	15620	0.1 °F
PT1000	-32768	32767	-2000	6000	0.1 °C	-3280	11120	0.1 °F
NI100	-32768	32767	-600	1800	0.1 °C	-760	3560	0.1 °F
NI1000	-32768	32767	-600	1800	0.1 °C	-760	3560	0.1 °F

TMCR2SL1

Introduction

The TMCR2SL1 is a standard cartridge module featuring 1 serial line.

For further hardware information, refer to TMCR2SL1 *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

The serial line can be configured for any one of the following protocols:

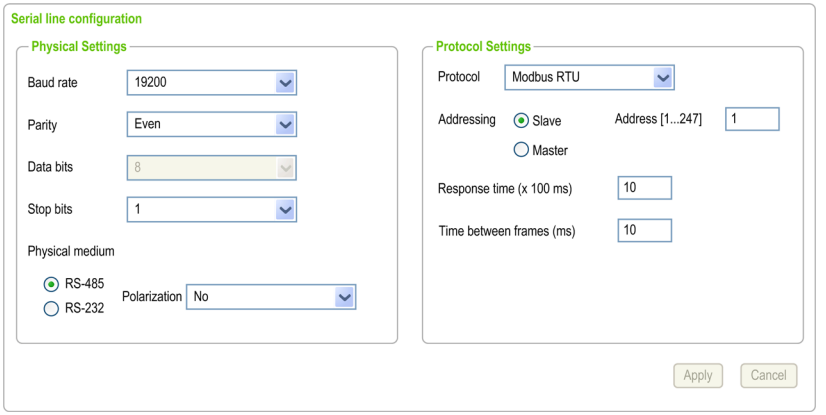
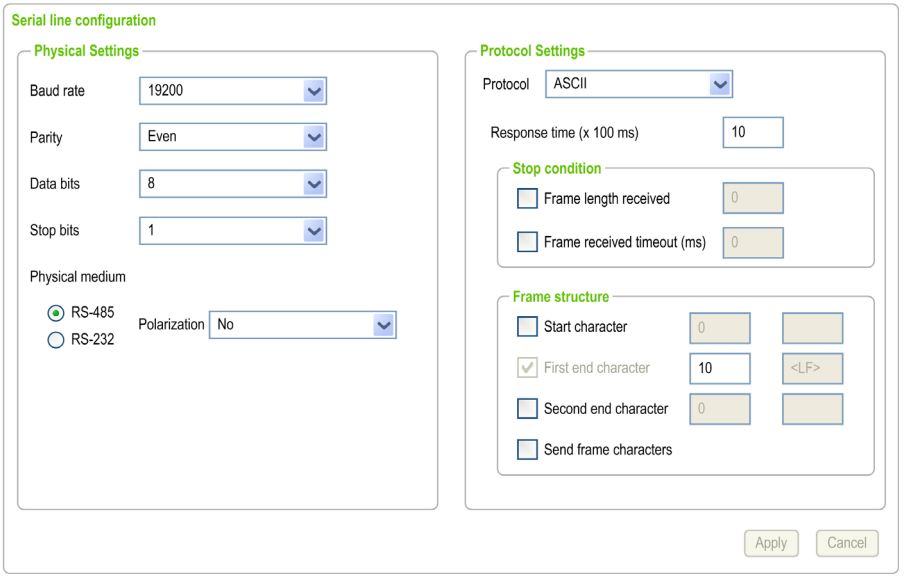
- Modbus RTU
- Modbus ASCII
- ASCII

You can configure both physical and protocol settings for the serial line. Serial lines are configured for the Modbus RTU protocol by default.

NOTE: You can only add one serial line cartridge to the controller.

Serial Line Configuration

This table describes how to configure the serial line:

Step	Action
1	Click the **SL2 (Serial line)** node in the **Hardware Tree** to display the serial line properties. This figure shows the properties of the serial line for **Modbus RTU** and **Modbus ASCII** protocols:  This figure shows the properties of the serial line for **ASCII** protocol: 
2	Edit the properties to configure the serial line. For detailed information on the serial line configuration parameters, refer to the table below.

This table describes each parameter of the serial line:

Parameter	Editable	Value	Default value	Description
Physical settings				
Baud rate	Yes	**1200** **2400** **4800** **9600** **19200** **38400** **57600** **115200**	**19200**	Allows you to select the data transmission rate (bits per second) for the modem from the drop-down list.
Parity	Yes	**None** **Even** **Odd**	**Even**	Allows you to select the parity of the transmitted data for error detection. Parity is a method of error detection in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit, is always odd or always even. If a byte is received with the wrong number of 1 bits, the byte is corrupt. However, an even number of detected errors can pass the parity check.
Data bits	Yes (only for the **ASCII** protocol	**7** **8**	**7** for Modbus ASCII, **8** for Modbus RTU	Allows you to select the number of data bits from the drop-down list. The number of data bits in each character can be 7 (for true ASCII) or 8 (for any kind of data, as this matches the size of a byte). 8 data bits are almost universally used in all applications.
Stop bits	Yes	**1** **2**	**1**	Allows you to select the number of stop bits from the drop-down list. A stop bit is a bit indicating the end of a byte of data. For electronic devices, 1 stop bit is usually used. For slow devices like electromechanical teleprinters, 2 stop bits are used.

Parameter	Editable	Value	Default value	Description
Physical medium	Yes	**RS-485** True/False **RS-232** True/False	**RS-485** True	Allows you to select the physical medium for communication. You can only select either the **RS-485** or **RS-232** medium. Enabling one medium disables the other one. A physical medium in data communications is the transmission path over which a signal propagates. It is an interface for interconnection of devices with the logic controller.
Polarization	Yes	**Yes** **No**	**No**	Polarization resistors are integrated in the cartridge module. Specify whether to switch on or off polarization.
Protocol settings				
Protocol	Yes	**Modbus RTU** **Modbus ASCII** **ASCII**	**Modbus RTU**	Allows you to select the protocol transmission mode for communication from the drop-down list. Protocol advanced parameters are displayed based on the selected protocol. Refer to the following figures and tables.
Protocol settings for the **Modbus RTU** and **Modbus ASCII** protocols:				
Addressing	Yes	**Slave** **Master**	**Slave**	Allows you to select the addressing mode. You can only select either of the **Slave** or **Master** addressing. Enabling one addressing mode disables the other one.
Address [1...247]	Yes	1...247	1	Allows you to specify the address ID of the slave. **NOTE:** This field is displayed only for the addressing of the slave. For master, this field does not appear on the screen.
Response time (x 100 ms)	Yes	10...255 ms	10	Allows you to specify the response time of the protocol to the queries.
Time between frames (ms)	Yes	3...255 ms	10	Allows you to specify the time between frames of the protocol.

Parameter	Editable	Value	Default value	Description
Protocol settings for the **ASCII** protocol:				
Stop condition				
Response time (x 100 ms)	Yes	1...255	10	Allows you to specify the response time of the protocol to the queries.
Frame length received	Yes	0...255	0	Allows you to specify the frame length received.
Frame received timeout (ms)	Yes	0...255	10	Allows you to specify the frame received timeout.
Frame structure				
Start character	Yes	0...255	58 (if check box is selected)	Allows you to specify the start character of the frame.
First end character	Yes	0...255	10 (if check box is selected)	Allows you to specify the first end character of the frame.
Second end character	Yes	0...255	10 (if check box is selected)	Allows you to specify the second end character of the frame.
Send frame characters	Yes	True/False	False	Allows you to enable or disable sending first end character of the frame to the ASCII protocol.

TMCR2SL1A

Introduction

The TMCR2SL1A is a standard cartridge module featuring 1 isolated serial line.

For further hardware information, refer to TMCR2SL1A *(see Modicon M100/M200 Logic Controller, Hardware Guide)*.

The serial line can be configured for any one of the following protocols:

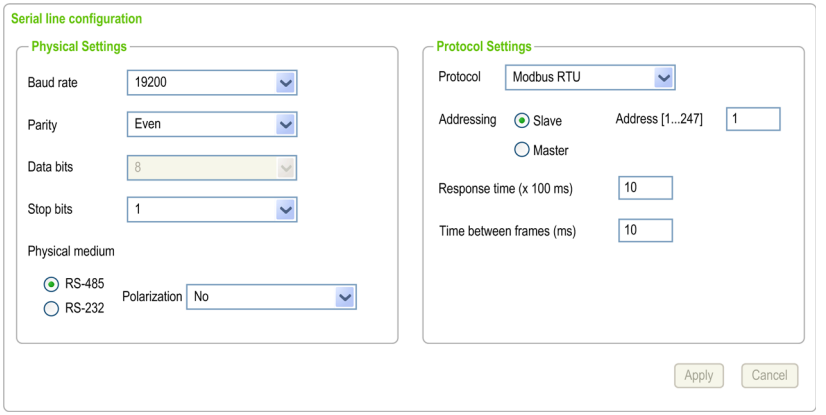
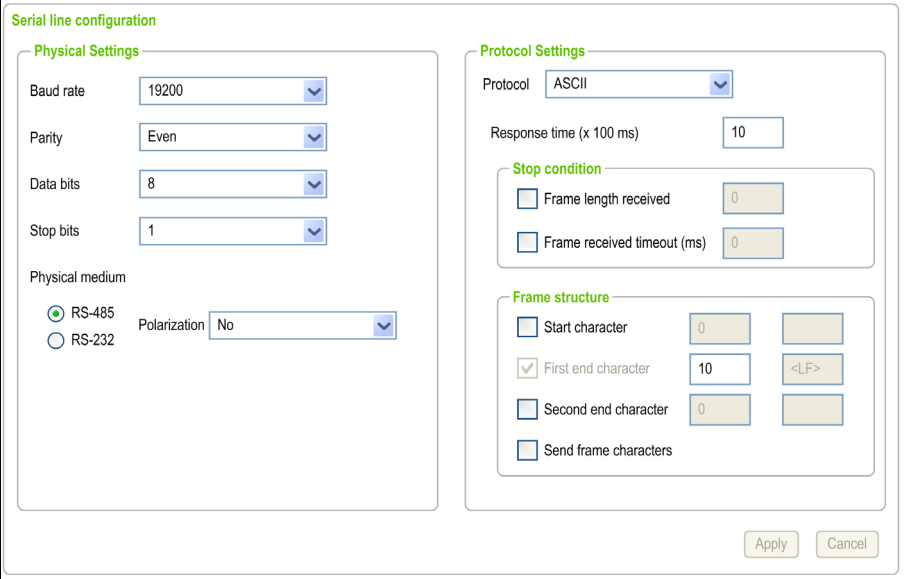
- Modbus RTU
- Modbus ASCII
- ASCII

You can configure both physical and protocol settings for the serial line. Serial lines are configured for the Modbus RTU protocol by default.

NOTE: You can only add one serial line cartridge to the controller.

Serial Line Configuration

This table describes how to configure the serial line:

Step	Action
1	Click the **SL2 (Serial line)** node in the **Hardware Tree** to display the serial line properties. This figure shows the properties of the serial line for **Modbus RTU** and **Modbus ASCII** protocols:  This figure shows the properties of the serial line for **ASCII** protocol: 
2	Edit the properties to configure the serial line. For detailed information on the serial line configuration parameters, refer to the table below.

This table describes each parameter of the serial line:

Parameter	Editable	Value	Default value	Description
Physical settings				
Baud rate	Yes	**1200** **2400** **4800** **9600** **19200** **38400** **57600** **115200**	**19200**	Allows you to select the data transmission rate (bits per second) for the modem from the drop-down list.
Parity	Yes	**None** **Even** **Odd**	**Even**	Allows you to select the parity of the transmitted data for error detection. Parity is a method of error detection in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit, is always odd or always even. If a byte is received with the wrong number of 1 bits, the byte is corrupt. However, an even number of detected errors can pass the parity check.
Data bits	Yes (only for the **ASCII** protocol	**7** **8**	**7** for Modbus ASCII, **8** for Modbus RTU	Allows you to select the number of data bits from the drop-down list. The number of data bits in each character can be 7 (for true ASCII) or 8 (for any kind of data, as this matches the size of a byte). 8 data bits are almost universally used in all applications.
Stop bits	Yes	**1** **2**	**1**	Allows you to select the number of stop bits from the drop-down list. A stop bit is a bit indicating the end of a byte of data. For electronic devices, 1 stop bit is usually used. For slow devices like electromechanical teleprinters, 2 stop bits are used.

Parameter	Editable	Value	Default value	Description
Physical medium	Yes	**RS-485** True/False **RS-232** True/False	**RS-485** True	Allows you to select the physical medium for communication. You can only select either the **RS-485** or **RS-232** medium. Enabling one medium disables the other one. A physical medium in data communications is the transmission path over which a signal propagates. It is an interface for interconnection of devices with the logic controller.
Polarization	Yes	**Yes** **No**	**No**	Polarization resistors are integrated in the cartridge module. Specify whether to switch on or off polarization.
Protocol settings				
Protocol	Yes	**Modbus RTU** **Modbus ASCII** **ASCII**	**Modbus RTU**	Allows you to select the protocol transmission mode for communication from the drop-down list. Protocol advanced parameters are displayed based on the selected protocol. Refer to the following figures and tables.
Protocol settings for the **Modbus RTU** and **Modbus ASCII** protocols:				
Addressing	Yes	**Slave** **Master**	**Slave**	Allows you to select the addressing mode. You can only select either of the **Slave** or **Master** addressing. Enabling one addressing mode disables the other one.
Address [1...247]	Yes	1...247	1	Allows you to specify the address ID of the slave. **NOTE:** This field is displayed only for the addressing of the slave. For master, this field does not appear on the screen.
Response time (x 100 ms)	Yes	10...255 ms	10	Allows you to specify the response time of the protocol to the queries.
Time between frames (ms)	Yes	3...255 ms	10	Allows you to specify the time between frames of the protocol.
Protocol settings for the **ASCII** protocol:				
Stop condition				
Response time (x 100 ms)	Yes	1...255	10	Allows you to specify the response time of the protocol to the queries.
Frame length received	Yes	0...255	0	Allows you to specify the frame length received.
Frame received timeout (ms)	Yes	0...255	10	Allows you to specify the frame received timeout.

Parameter	Editable	Value	Default value	Description
Frame structure				
Start character	Yes	0...255	58 (if check box is selected)	Allows you to specify the start character of the frame.
First end character	Yes	0...255	10 (if check box is selected)	Allows you to specify the first end character of the frame.
Second end character	Yes	0...255	10 (if check box is selected)	Allows you to specify the second end character of the frame.
Send frame characters	Yes	True/False	False	Allows you to enable or disable sending first end character of the frame to the ASCII protocol.

Chapter 7
TM3R Expansion Module Configuration

Overview

This chapter describes how to configure the TM3R expansion modules of the M100/M200 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

I/O Configuration General Practices

Match Software and Hardware Configuration

The I/O that may be embedded in your controller is independent of the I/O that you may have added in the form of I/O expansion. It is crucial that the logical I/O configuration within your program matches the physical I/O configuration of your installation. If you add or remove any physical I/O to or from the I/O expansion bus, or, depending on the controller reference, to or from the controller (in the form of cartridges), it is imperative that you update your application configuration. This is also true for any field bus devices you may have in your installation. Otherwise, there is the possibility that the I/O expansions will no longer function while the embedded I/O that may be present in your controller will continue to operate.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Update the configuration of your program each time you add or delete any type of I/O expansions on your I/O bus, or you add or delete any devices on your field bus.

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

Configuring the TM3R Digital I/O Modules

Introduction

The range of TM3R digital I/O expansion modules includes:

- TM3R Digital Mixed Input/Output Modules

Configuring the Modules

Configuration tab: Displaying Configuration Details in the Configuration Tab *(see page 129)* describes how to view the configuration of these modules.

Programming tab: Displaying Configuration Details in the Programming Tab *(see page 130)* describes how to view and update programming-related properties of these modules.

Using I/O Modules in a Configuration

Adding a Module

The following steps explain how to add a TM3R expansion module to the logic controller in a SoMachine Basic project:

Step	Action
1	Click the **Configuration** tab in the SoMachine Basic window.
2	In the catalog area, click one of the following module types to expand the list of expansion modules: • **TM3 Digital I/O Modules**
3	Select the TM3R expansion module to add from the list. **Result**: The description of the physical characteristics of the selected expansion module appears in the bottom of the catalog area.
4	Drag the selected expansion module to the editor area and drop the module on the right-hand side of the controller or the last expansion module in the configuration. **Result**: The module is added under the **My Controller** → **I/O Bus** branch of the hardware tree and the description of the physical characteristics of the selected module appears in the bottom of the editor area.

Inserting a Module Between two Existing Modules

Drag the module between two modules, or between the controller and the first module until a vertical green bar appears and then drop the module.

NOTE: The addresses change when you change the position of modules by inserting a new module. For example, if you move an input module from position 4 to position 2, the addresses change from `I4.x` to `I2.x`, and all corresponding addresses in the program are automatically renamed.

The I/O that may be embedded in your controller is independent of the I/O that you may have added in the form of I/O expansion. It is important that the logical I/O configuration within your program matches the physical I/O configuration of your installation. If you add or remove any physical I/O to or from the I/O expansion bus, update your application configuration (this is also true for any field bus devices you may have in your installation). Otherwise, there is the potential that the expansion bus or field bus will no longer function while the embedded I/O that may be present in your controller will continue to operate.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Update the configuration of your program each time you add or delete any type of I/O expansions on your I/O bus, or you add or delete any devices on your field bus.

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

Replacing an Existing Expansion Module

You can replace an existing module with a new module by dragging the new module and dropping it onto the module to be replaced.

A message appears asking you to confirm the operation. Click **Yes** to continue.

Removing a Module

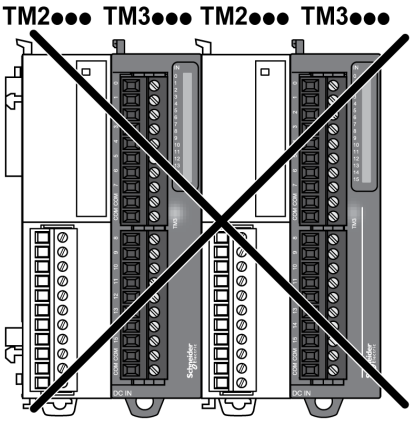
You can remove an expansion module by pressing the **Delete** key or by right-clicking the module and clicking **Remove** on the contextual menu that appears.

If the expansion module contains at least one address being used in a program, a message appears asking you to confirm the operation. Click **Yes** to continue.

Mixing Expansion Module Types

You can mix different I/O module types on the same logic controller (for example, TM3R, TM3, and TM2 modules).

Place any TM2 module(s) at the end of your configuration after any TM3 module(s):



In this case, however, the I/O bus of the logic controller operates at the speed of the slower module type. For example, when both TM2 and TM3 modules are used, the I/O bus of the logic controller operates at the speed of the TM2 modules.

Maximum Hardware Configuration

SoMachine Basic displays a message when:

- The maximum number of modules supported by the logic controller is exceeded.
- The total power consumption of all expansion modules directly connected to the logic controller exceeds the maximum current delivered by the logic controller.

Refer to the hardware guide of your controller for more information on the maximum supported configuration.

Configuring Digital I/Os

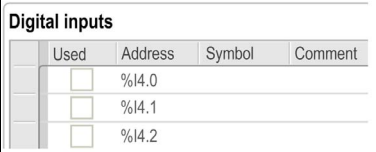
Overview

You can configure digital I/Os of your expansion module using:

- **Configuration** tab:
  - Digital inputs
  - Digital outputs
- **Programming** tab.

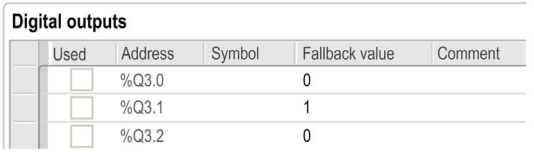
Configuring Digital Inputs in the Configuration Tab

Follow these steps to display and configure the digital input properties in the **Configuration** tab:

Step	Description
1	Click the **Configuration** tab in the SoMachine Basic window.
2	In the hardware tree, click **MyController** → **IO Bus** → **Module x** → **Digital inputs**, where x is the expansion module number on the controller. **Result**: The digital input properties of the selected module are displayed in the editor area, for example: 
3	Edit the properties to configure the digital inputs: ● **Used**: Indicates whether the corresponding address is being used in the program or not. ● **Address**: Displays the address of the digital input on the expansion module. For details on addressing I/O objects, refer to I/O Addressing. ● **Symbol**: Allows you to specify a symbol to associate with the corresponding digital input object to be used in the program. Double-click in the **Symbol** column, type the symbol name of the corresponding object, and press **Enter**. ● **Comment**: Allows you to specify a comment to associate with the corresponding digital input object. Double-click in the **Comment** column, type a comment for the corresponding object, and press **Enter**.
4	Click **Apply** to save the changes.

Configuring Digital Outputs in the Configuration Tab

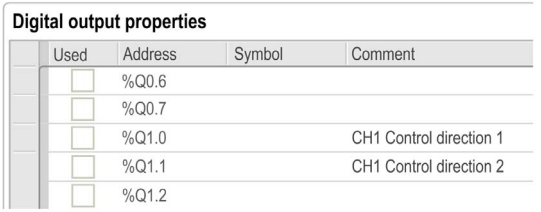
Follow these steps to display and configure the digital output properties in the **Configuration** tab:

Step	Description
1	Click the **Configuration** tab in the SoMachine Basic window.
2	In the hardware tree, click **MyController** → **IO Bus** → **Module x** → **Digital outputs**, where x is the expansion module number on the controller. **Result**: The digital output properties of the selected module are displayed in the editor area, for example:  Digital outputs Used | Address | Symbol | Fallback value | Comment %Q3.0 0 %Q3.1 1 %Q3.2 0
3	Edit the properties to configure the digital outputs: ● **Used**: Indicates whether the corresponding address is being used in the program or not. ● **Address**: Displays the address of the digital output on the expansion module. For details on addressing I/O objects, refer to I/O Addressing. ● **Symbol**: Allows you to specify a symbol to associate with the corresponding digital output object to be used in the program. Double-click in the **Symbol** column, type the symbol name of the corresponding object, and press **Enter**. ● **Fallback value**. Allows you to specify the value to apply to the corresponding output (fallback to 0 or fallback to 1) when the logic controller enters the STOPPED or an exception state. The default value is 0. If **Maintain values** fallback mode is configured, the output retains its current value when the logic controller enters the STOPPED or an exception state. For more details on maintaining output values, refer to Fallback Behavior. ● **Comment**: Allows you to specify a comment to associate with the corresponding digital output object. Double-click in the **Comment** column, type a comment for the corresponding object, and press **Enter**.
4	Click **Apply** to save the changes.

Displaying Configuration Details in the Programming Tab

The **Programming** tab displays configuration details of all inputs/outputs and allows you to update programming-related properties such as symbols and comments.

Follow these steps to view and update details of I/O modules in the **Programming** tab:

Step	Description
1	Click the **Programming** tab in the SoMachine Basic window.
2	In the left-hand area of the **Programming** tab, click on the **Tools** tab and from the **I/O objects** branch, select one of the following I/O types to display the properties: ● **Digital inputs** ● **Digital outputs** ● **Analog inputs** ● **Analog outputs** **Result**: A list of all embedded and expansion module I/O addresses appears in the lower central area of the SoMachine Basic window, for example: 
3	Scroll down to the range of addresses corresponding to the expansion module you are configuring. The following properties are displayed: ● **Used**: Indicates whether the corresponding address is being used in the program or not. ● **Address**: Displays the address of the digital output on the expansion module. For details on addressing I/O objects, refer to I/O Addressing. ● **Symbol**: Allows you to specify a symbol to associate with the corresponding I/O object to be used in the program. Double-click in the **Symbol** column, type the symbol name of the corresponding object, and press **Enter**. If a symbol already exists, right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments. ● **Comment**: Allows you to specify a comment to associate with the corresponding I/O object. Double-click in the **Comment** column, type a comment for the corresponding object, and press **Enter**.
4	Click **Apply** to save the changes.

Chapter 8
Embedded Communication Configuration

Overview

This chapter describes how to configure the communication features of the M100/M200 Logic Controller.

What Is in This Chapter?

This chapter contains the following sections:

Section 8.1
Ethernet Configuration

What Is in This Section?

This section contains the following topics:

Configuring Ethernet Network

Introduction

You can configure the TCP/IP connection to the logic controller by configuring the Ethernet network. The Ethernet establishes a local area network (LAN) between the logic controller and other devices. The Ethernet configuration provides you the ability to configure the IP address of the network device.

NOTE: The controller-PC link uses the TCP/IP protocol. It is required for this protocol to be installed on the PC.

You can obtain the IP address by the following protocols:
- Dynamic Host Configuration Protocol (DHCP)
- Bootstrap Protocol (BOOTP)

You can also specify the IP address by specifying the following addresses:
- IP address
- Subnet mask
- Gateway address

Ethernet Services

The logic controller supports the following services:
- Modbus TCP Server
- Modbus TCP Client
- Modbus TCP Slave Device

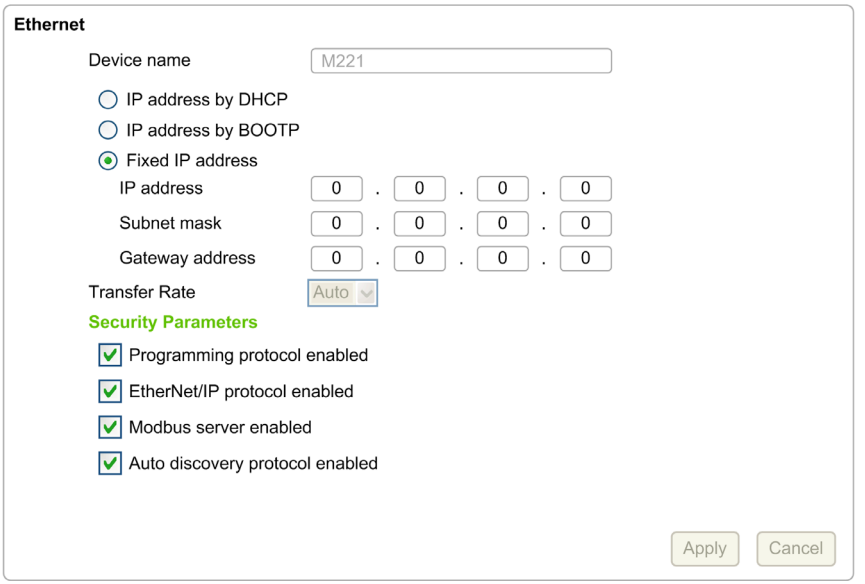
This table presents the maximum number of TCP server connections:

Connection Type	Maximum Number of Connections
Modbus Server	8
Modbus Client	1

Each server based on TCP manages its own set of connections. When a client tries to open a connection that exceeds the poll size, the logic controller closes the oldest connection. If the connections are busy (exchange in progress), when a client tries to open a new one, the new connection is denied. The server connections stay open as long as the logic controller stays in its present operational state (RUNNING, STOPPED, or HALTED). The server connections is closed when a transition is made from its present operational state (RUNNING, STOPPED, or HALTED), except in case of power outage (because the controller does not have time to close the connections).

Ethernet Configuration

This table describes how to configure the Ethernet:

Step	Action
1	Click the **ETH1** node in the hardware tree to display the Ethernet properties in the editor area. 
2	Edit the properties to configure the Ethernet. For detailed information on the Ethernet configuration parameters, refer to the table below.

NOTE: The Security Parameters displayed depend on the functional level selected for the application.

This table describes each parameter of the Ethernet configuration:

Parameter	Editable	Value	Default Value	Description
Ethernet				
Device name	No	*Any*	**M200** (if the controller used in the configuration is M100/M200 Logic Controller)	Displays the name of the device that is connected with the Ethernet network.
(1) You can select any 1 option for IP addressing. Selecting any 1 option, disables the other options. **(2)** These options are enabled only if you select the option **Fixed IP Address** for IP addressing. **(3)** *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range 0...255.				

Parameter	Editable	Value	Default Value	Description
IP address by DHCP	Yes(1)	True/False	False	Allows you to obtain the IP address from the DHCP server on the network.
IP address by BOOTP	Yes(1)	True/False	False	Allows you to obtain the IP address from the Boot PROM configuration server on the network.
Fixed IP address	Yes(1)	True/False	True	Allows you to specify the IP address manually for host or network interface identification.
IP address	Yes(2)	*w.x.y.z*(3)	0.0.0.0	Allows you to specify the IP address of the device in the Ethernet network. Assigning 0.0.0.0 as IP address for the M100/M200 Logic Controller forces the firmware to generate an IP address from the MAC address.
Subnet mask	Yes(2)	*w.x.y.z*(3)	0.0.0.0	Allows you to specify the address of the subnetwork to authorize a group of devices for data exchange. It determines which bits in an IP address correspond to the network address and which bits correspond to the subnet portions of the address.
Gateway address	Yes(2)	*w.x.y.z*(3)	0.0.0.0	Allows you to specify the IP address of the node (a router) on a TCP/IP network that serves as an access point to another network.
Transfer Rate	No	–	**Auto**	Displays the transfer rate for obtaining the IP address.
Security Parameters				
Programming protocol enabled	Yes	True/False	True	Allows you to enable or disable programming protocol for communication with the other devices in the network.
Auto discovery protocol enabled	Yes	True/False	True	Allows you to enable or disable auto discovery protocol to automatically detect the devices in a network.
Modbus server enabled	Yes	True/False	True	Allows you to enable or disable Modbus server for serial device connectivity.

(1) You can select any 1 option for IP addressing. Selecting any 1 option, disables the other options.
(2) These options are enabled only if you select the option **Fixed IP Address** for IP addressing.
(3) *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range 0...255.

NOTE: When a protocol listed in **Security Parameters** is disabled, requests from the corresponding server type are ignored. The corresponding configuration screen remains accessible; however, program execution is not affected.

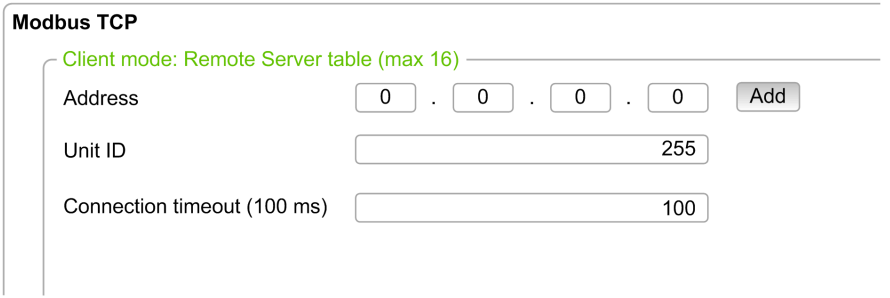
Configuring Modbus TCP

Introduction

You can configure the Ethernet port to enable the embedded Modbus TCP server giving the logic controller Modbus TCP abilities.

Modbus TCP Configuration

This table describes how to configure the Modbus TCP:

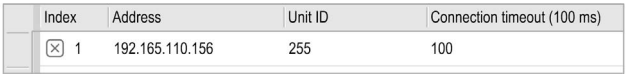
Step	Action
1	Click the **Modbus TCP** node that appears below the **ETH1** node in the hardware tree to display the Ethernet/IP adapter properties in the editor area. 
2	Select **Enabled** to edit the properties to configure the Modbus TCP protocol. For detailed information on the Modbus TCP configuration parameters, refer to the table below.

This table describes each parameter of the Modbus TCP configuration:

Parameter	Editable	Value	Default Value	Description
Client mode: Remote Server table (max 16)				
Address	Yes	*w.x.y.z*(1)	0.0.0.0	Allows you to specify the IP address of the remote server. Also, refer to Adding Remote Servers *(see page 137)*.
Unit ID	Yes	0...255	255	Allows you to specify the unit ID of the remote server.
Connection timeout (100 ms)	Yes	0...65535	100	Allows you to specify the connection timeout duration.
(1) *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range 0...255.				

Adding Remote Servers

This table describes how to add a remote server for Modbus TCP:

Step	Action
1	Enter the IP address in the **Address** field.
2	Enter the value for **Unit ID** and **Connection timeout (100 ms)**.
3	Click the **Add** button. **Result**: A list of remote servers that you have added appears on the screen. This figure presents the table that lists the remote servers: 

This table describes each column of the table listing the remote servers:

Parameter	Editable	Value	Default value	Description
Index	No	1...16	–	Displays the index number of the servers which are remotely connected.
Address	Yes	*w.x.y.z*(1)	0.0.0.0	Displays the IP address of the remote server.
Unit ID	Yes	0...255	255	Displays the unit ID of the remote server.
Connection timeout (100 ms)	Yes	0...65535	100	The connection timeout duration. The period of time (in units of 100 ms) during which the controller attempts to establish a TCP connection to the remote device. If at the end of this period a TCP connection is still not established, the controller stops connection attempts until the next connection request with an `EXCH` instruction.
(1) *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range 0...255.				

Click the close button in the row to remove a remote server.

Double-click the remote server entry in a row to edit the values.

Section 8.2
Serial Line Configuration

Configuring Serial Line

Introduction

The M100/M200 Logic Controller references are equipped with one serial line (SL1).

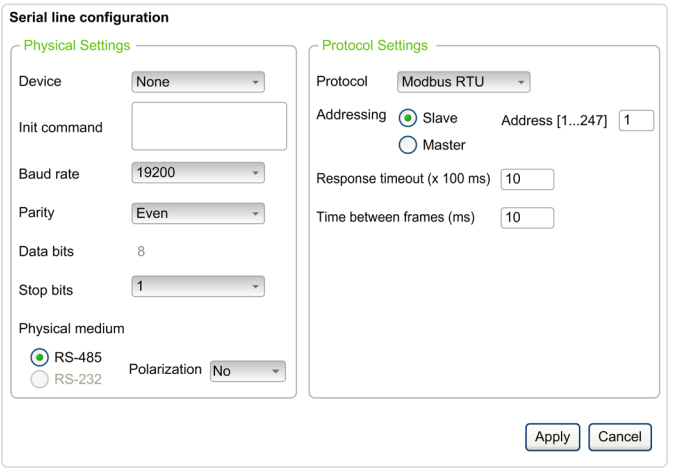
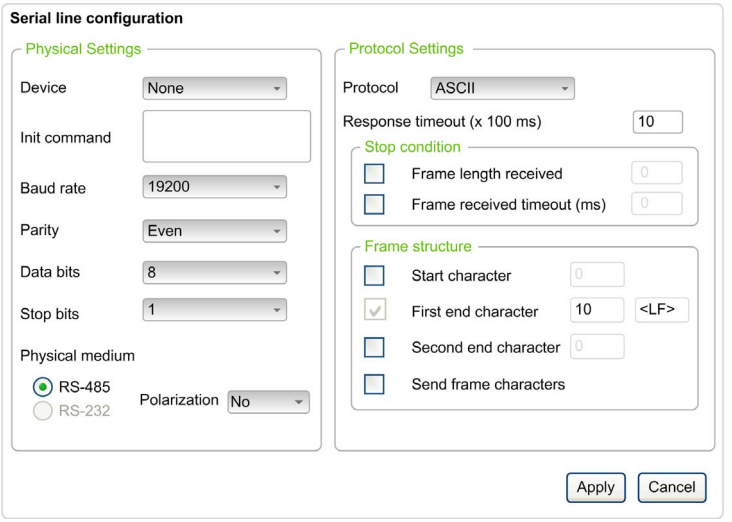
The serial line can be configured for any one of these protocols:

- Modbus RTU
- Modbus ASCII
- ASCII

You can configure both physical and protocol settings for the serial line. The serial line is configured for the Modbus RTU protocol by default.

Serial Line Configuration

This table describes how to configure the serial line:

Step	Action
1	Click the **SL1 (Serial line)** node in the hardware tree to display the serial line properties. This figure presents the properties of the serial line for **Modbus RTU** and **Modbus ASCII** protocols:  This figure presents the properties of the serial line for **ASCII** protocol: 
2	Edit the properties to configure the serial line. For detailed information on the serial line configuration parameters, refer to the table below.

This table describes each parameter of the serial line:

Parameter	Editable	Value	Default value	Description
Physical settings				
Baud rate	Yes	**1200** **2400** **4800** **9600** **19200** **38400** **57600** **115200**	**19200**	Allows you to select the data transmission rate (bits per second) for the modem from the drop-down list.
Parity	Yes	**None** **Even** **Odd**	**Even**	Allows you to select the parity of the transmitted data for error detection. Parity is a method of error detection in transmission. When parity is used with a serial port, an extra data bit is sent with each data character. It is arranged so that the number of bits set to 1 in each character, including the parity bit, is always odd or always even. If a byte is received with the wrong number of bits set to 1, the byte is corrupted.
Data bits	Yes (only for the **ASCII** protocol)	**7** **8**	**8**	Allows you to select the data bit from the drop-down list. The number of data bits in each character can be 7 (for true ASCII) or 8.
Stop bits	Yes	**1** **2**	**1**	Allows you to select the stop bit from the drop-down list. Stop bit is a bit indicating the end of a byte of data. For electronic devices usually 1 stop bit is used. For slow devices like electromechanical teleprinters, 2 stop bits are used.
Physical medium	Yes (for the controller)	**RS-485** True/False	**RS-485** True	Allows you to select the physical medium for communication. A physical medium in data communications is the transmission path over which a signal propagates. It is an interface for interconnection of devices with the logic controller.

Parameter	Editable	Value	Default value	Description
Polarization	No (for the controller)	**Yes** **No**	**No**	Polarization resistors are integrated in the cartridge module. For the controller, this parameter is disabled and for the cartridges, this parameter allows you to switch on or off polarization.
Protocol settings				
Protocol	Yes	**Modbus RTU** **Modbus ASCII** **ASCII**	**Modbus RTU**	Allows you to select the protocol transmission mode for communication from the drop-down list. Protocol advanced parameters are displayed based on the selected protocol. Refer to the following figures and tables.
Protocol settings for the **Modbus RTU** and **Modbus ASCII** protocols:				
Addressing	Yes	**Slave** True/False **Master** True/False	**Slave** True	Allows you to select the addressing mode. You can only select either of the **Slave** or **Master** addressing. Enabling any of the addressing mode, disables the other one.
Address [1...247]	Yes	1...247	1	Allows you to specify the address ID of the slave. **NOTE:** This field is displayed only for the addressing of the slave. For master, this field does not appear on the screen.
Response time (x 100 ms)	Yes	0...255 ms	10	Allows you to specify the response time of the protocol to the queries.
Time between frames (ms)	Yes	2...255 ms	10	Allows you to specify the period of time between frames of the protocol. (corresponds to inter-frame delay used in other products). The value specified is automatically adjusted to conform to a Modbus standard 3.5 character time delay.

Parameter	Editable	Value	Default value	Description
Protocol settings for the **ASCII** protocol:				
Response time (x 100 ms)	Yes	0...255 ms	10	Allows you to specify the response time of the protocol to the queries.
Stop condition				
Frame length received	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 1 (if check box is selected)	Allows you to specify the length of the received frame. **NOTE:** You can configure only one parameter for stop condition that is either **Frame length received** or **Frame received timeout (ms)**.
Frame received timeout (ms)	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 10 (if check box is selected)	Allows you to specify the timeout duration for the received frame.
Frame structure				
Start character	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 58 (if check box is selected)	Allows you to specify the start character of the frame. The ASCII character corresponding to the start character value is displayed on right-hand side of the value field.
First end character	Yes	1...255	0 (if check box is not selected) 10 (if check box is selected)	Allows you to specify the first end character of the frame. **NOTE:** To be able to enable or disable the **First end character**, configure at least one stop condition parameter. The ASCII character corresponding to the first end character value is displayed on right-hand side of the value field.
Second end character	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 10 (if check box is selected)	Allows you to specify the second end character of the frame. **NOTE:** This field is disabled with the disabled **First end character** parameter. The ASCII character corresponding to the second end character value is displayed on right-hand side of the value field.
Send frame characters	Yes	True/False	False	Allows you to enable or disable sending first end character of the frame to the ASCII protocol.

Section 8.3
Supported Modbus Function Codes

Supported Modbus Function Codes

Presentation

This table lists the function codes supported by both serial Modbus and Modbus TCP and their effect on controller memory variables.:

Supported Modbus Function Code	Supported Sub-Function Code	Description
1 (0x01) or 2 (0x02)	–	Read multiple internal bits %M
3 (0x03) or 4 (0x04)	–	Read multiple internal registers %MW
5 (0x05)	–	Write single internal bit %M
6 (0x06)	–	Write single internal register %MW
8 (0x08)	0 (0x00), 10 (0x0A)...18 (0x12)	Diagnostics
15 (0x0F)	–	Write multiple internal bits %M
16 (0x10)	–	Write multiple internal registers %MW
23 (0x17)	–	Read/write multiple internal registers %MW
43 (0x2B)	14 (0x0E)	Read device identification (regular service)

NOTE:
The impact of Modbus function codes used by a master M221 Logic Controller depends on the slave device type. In the major types of slave device:

- Internal bit means %M
- Input bit means %I
- Internal register means %MW
- Input register means %IW

Depending on the type of slave and the slave address, an internal bit should be a %M or %Q; an input bit should be a %I or %S, an input register should be a %IW or a %SW and an internal register should be a %MW or a %QW. For more details, refer to the slave device.

Chapter 9
Micro SD Card

Introduction

The Modicon M100/M200 Logic Controller allows file transfers with a Micro SD card.

This chapter describes how to manage Modicon M100/M200 Logic Controller files with a Micro SD card.

What Is in This Chapter?

This chapter contains the following topics:

File Management Operations

Introduction

The Modicon M100/M200 Logic Controller allows the following types of file management with an micro SD card:

- Clone management *(see page 150)*: back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management *(see page 152)*: download firmware updates directly to the logic controller
- Application management *(see page 153)*: back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management *(see page 155)*: add, change, or delete the post configuration file of the logic controller
- Error log management *(see page 157)*: back up or delete the error log file of the logic controller

NOTE:

- Logic controller operation is not affected during file transfers.
- The Modicon M100/M200 Logic Controller accepts only micro SD cards formatted in FAT or FAT32.
- Micro SD card operations are performed regardless of any user access-rights that may be enabled in the target logic controller.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- You must have operational knowledge of your machine or process before connecting this device to your controller.
- Ensure that guards are in place so that any potential unintended equipment operation will not cause injury to personnel or damage to equipment.

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

If there is a power outage or communication interruption during the transfer of the application program or a firmware change, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer.

NOTICE

INOPERABLE EQUIPMENT

- Do not interrupt the transfer of the application program or a firmware change once the transfer has begun.
- Do not place the device into service until the transfer has completed successfully.

Failure to follow these instructions can result in equipment damage.

NOTE: Before inserting a micro SD card in the controller, verify that the micro SD card contains a valid script file if the micro SD card is not empty. Otherwise, the controller remains in BOOTING mode as it does not detect any valid script during booting. In this case, remove the micro SD card to make the controller start normally.

SD Card Supported File Types

Introduction

This table lists the file locations and types of file that can be managed by cloning or script commands:

Folder	Description	Default file name
/	Script file	Script.cmd
/	Script log	Script.log
/sys/os	Firmware file	M200M100.mfw
/usr/app	Application file	*.smbk
/usr/cfg	Post configuration file	Machine.cfg
/sys/log	Detected error log file	PlcLog.csv

Script File Commands

A script file is a text file stored in the root directory of the micro SD card containing commands to manage exchanges with the controller.

This table describes the supported script commands:

Command	Description	Source	Destination
Download	Download a file from the micro SD card to the controller.	Select the file to download.	Select the controller destination folder.
Upload	Upload files contained in folder of controller memory to the micro SD card.	Select the folder.	Select the micro SD card folder.
Delete	Delete files contained in a controller folder.	Select the folder and enter a specific file name. **Important:** by default, all folder files are selected.	-
Reboot	Restart the controller (this command must be the last command in the script).	-	-

NOTE:

- All commands can only be executed when the controller in the STOP or BOOTING state. No command is executed if a micro SD card is inserted when the controller is in RUN mode.
- When a micro SD card command is in progress, start controller commands are ignored. When the micro SD card command completes, you must start the controller manually.

Script File Examples

Download commands:

```
Download "/usr/Cfg"
Download "/sys/os/M200M100.mfw"
```

Upload commands:

```
Upload "/usr/app/*"
Upload "/usr/cfg/Machine.cfg"
```

Delete commands:

```
Delete "/usr/app/*"
```

Reboot commands:

```
Reboot
```

Script Log

A `script.log` file is automatically created in the SD card root directory after script operations. The status of the script operations can be verified reading this file.

Clone Management

Cloning

Cloning allows you to automatically back up the application, firmware, and post configuration (if it exists) of the Modicon M100/M200 Logic Controller to the micro SD card.

The micro SD card can then be used to later restore the firmware, application, and post configuration (if it exists) to the logic controller, or copy them to another logic controller with the same reference.

Before cloning a controller, the M100/M200 Logic Controller verifies whether the application is copy-protected or not . For details, refer to Password Protecting an Application *(see SoMachine Basic, Operating Guide)*.

NOTE:

- The micro SD card must be empty and correctly formatted to perform this procedure.
- The detected error log and data memory are not cloned.
- If the application is password-protected, the clone operation is completed but the user application cannot be restored and the **ERR** LED is permanently on.

Creating a Clone Micro SD Card

This procedure describes how to copy the application, firmware, and post configuration (if it exists) from the controller to a micro SD card:

Step	Action
1	Format a micro SD card on the PC.
2	Insert the micro SD card in the controller. **Result:** The clone operation starts automatically. During the clone operation, the following LEDs are ON: **PWR**, and **SD**.
3	Wait until the clone operation is completed (the **SD** LED turns OFF). **NOTE:** The clone operation lasts 2 to 3 minutes. The clone operation has a low priority in order to minimize impact on the user logic and communication performance of the logic controller. Depending on the amount of free time in your program, the operation may take longer to complete if the logic controller is in STOP mode.
4	Remove the micro SD card from the controller.

Restoring or Copying from a Clone Micro SD Card

This procedure describes how to download the application, firmware, and post configuration (if it exists) stored in the micro SD card to your controller:

Step	Action
1	Remove power from the controller.
2	Insert the micro SD card into the controller.
3	Restore power to the controller. **Result:** The clone operation is in progress. **NOTE:** The **SD** LED is turned on during the operation.
4	Wait until the end of the download (the **SD** LED is turned off). If an error is detected, the **SD** LED is flashes, and the **ERR** LED starts flashing.
5	Remove the micro SD card to restart the controller.

NOTE: Downloading a cloned application to the controller first removes the existing application from controller memory, regardless of any user access-rights that may be enabled in the target controller.

Firmware Management

Overview

You can use a micro SD card to download firmware updates directly to the logic controller.

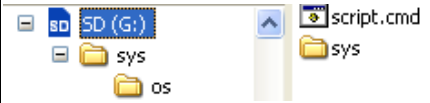
Refer to Controller States and Behavior *(see page 32)* for information on the logic controller operating states and status of the LEDs.

Downloading Firmware to the Controller

This table describes how to download a firmware to the logic controller using an SD card:

Step	Action
1	Stop the logic controller and unplug the USB programming cable if connected.
2	Insert an empty micro SD card into the PC that is running SoMachine Basic.
3	Create a file called `script.cmd` in the SD card root directory.

4	Edit the file and insert the following commands:
	`Download "/sys/os"`
	`Reboot`

5	Create the folder path `\sys\os` in the SD card root directory and copy the firmware file in the `os` folder:  **NOTE:** A firmware file example and the script are available in the directory `Firmwares & PostConfiguration\M200M100\` of the SoMachine Basic installation directory. The default firmware file name for the M100/M200 Logic Controller is `M200M100.mfw`.
6	Remove the micro SD card from the PC and insert it into the micro SD card slot of the logic controller.
7	Start the logic controller. **Result:** Copying of the firmware file begins. During the operation, the **SD** system LED on the logic controller is on. **NOTE:** Do not stop the logic controller while the operation is in progress.
8	When the **SD** system LED is turned off, remove the SD card.
9	Reconnect the USB programming cable to the logic controller and login to the logic controller with the SoMachine Basic software.
10	The status of the controller firmware update can be verified reading the `script.log` file created automatically in the SD card root directory.

Application Management

Overview

You can use a micro SD card to back up and restore your controller application, or copy it to another controller with the same reference.

Backing Up an Application

This table describes how to back up the controller application on the micro SD card:

Step	Action
1	Create a `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Upload "/usr/app"`
3	Copy the script file to the root folder of the micro SD card.
4	Insert the prepared micro SD card in the controller. **Result:** Copying of the application file begins. During the operation, the **SD** system LED on the logic controller is on. Do not stop the logic controller while the operation is in progress. **NOTE:** The application backup process has a low priority in order to minimize impact on the program and communication performance of the logic controller. Depending on the amount of free time in your program, the operation may take considerably longer to complete if the logic controller is in STOPPED state than if it is in BOOTING state.
5	When the **SD** system LED is off, remove the micro SD card. **Result:** The application file (`*.smbk`) is saved on the micro SD card.
6	The status of the application backup can be verified reading the `script.log` file created on the micro SD card root directory.

Restoring an Application or Copying an Application to Another Controller

This table describes how to transfer the controller application from the micro SD card to the controller:

Step	Action
1	Edit the `script.cmd` file in the root folder of the micro SD card with a text editor.

2	Replace the content of the script by the following lines:
	`Delete "/usr/app"`
	`Download "/usr/app"`
	`Reboot`

3	Remove power from the controller.
4	Insert the prepared micro SD card in the controller.

Step	Action
5	Restore power to the controller. **Result:** Copying of the application file begins. During the operation, the **SD** system LED on the logic controller is on. Do not stop the logic controller while the operation is in progress. **NOTE:** Before the executing a `Download` command, the controller verifies the format of the `Machine.cfg` file and that the communication channels, parameters and configured values are valid. If errors are detected, the download is cancelled: the **SD** LED is on and a message written to the log file. If this occurs during system power-up, remove the **SD** to initialize the controller.
6	When the **SD** system LED is off, remove the micro SD card to restart the controller.
7	The status of the application transfer can be verified reading the `script.log` file created on the micro SD card root directory.

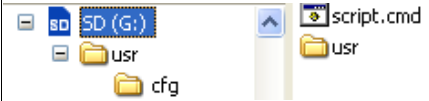
Post Configuration Management

Overview

You can use an SD card to add, change, or delete the post configuration file of your controller.

Adding or Changing a Post Configuration

This table describes how to add or change the controller post configuration:

Step	Action
1	Insert an empty SD card into the PC that is running SoMachine Basic.
2	Copy the post configuration file (`Machine.cfg`) to the folder `\usr\cfg` and the script file to the root folder of the SD card:  **NOTE:** A post configuration file example and the associated script are available in the directory `Firmwares & PostConfiguration\PostConfiguration\add_change\` of the SoMachine Basic installation directory.
3	If necessary, edit the `Machine.cfg` file to configure your post configuration parameters.
4	Insert the prepared SD card in the controller. **Result:** Downloading of the post configuration file begins. During the operation, the **SD** system LED on the logic controller is on. Do not stop the logic controller while the operation is in progress. **NOTE:** Before the download the file format is checked, as well as if all of the channels, parameters, and values configured are valid; in case of detected error the download is aborted.
5	When the **SD** system LED is turned off, remove the SD card.
6	Restart the controller to apply the new post configuration file.
7	The status of the operation can be verified reading the `script.log` file created on the SD card root directory.

Removing a Post Configuration File

This table describes how to remove the post configuration file of the controller:

Step	Action
1	Insert an empty SD card into the PC that is running SoMachine Basic.
2	Copy the script file available in the directory `Firmwares & PostConfiguration\PostConfiguration\remove\` of the SoMachine Basic installation directory to the root directory of the SD card.
3	Insert the prepared SD card in the controller. **Result:** The post configuration file is removed. During the operation, the **SD** system LED on the logic controller is on. Do not stop the logic controller while the operation is in progress.
4	When the **SD** system LED is turned off, remove the SD card.
5	Restart the controller to apply the application parameters.
6	The status of the operation can be verified reading the `script.log` file created automatically in the SD card root directory.

Error Log Management

Overview

You can use the micro SD card to back up or delete the error log file of the logic controller.

Backing Up the Error Log

This table describes how to back up the logic controller error log file on the micro SD card:

Step	Action
1	Create a `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Upload "/sys/log"`
3	Copy the script file to the root folder of the micro SD card.
4	Insert the prepared micro SD card in the logic controller. **Result:** Transfer of the error log file begins. During the operation, the **SD** system LED on the logic controller is on. Do not stop the logic controller while the operation is in progress. **NOTE:** The backup process has a low priority in order to minimize impact on the program and communication performance of the logic controller. Depending on the amount of free time in your program, the operation may require more time to complete if the logic controller is in STOPPED state than if it is in BOOTING state.
5	When the **SD** system LED is turned off, remove the micro SD card. **Result:** The error log file (`PlcLog.csv`) is saved on the micro SD card.
6	The status of the operation can be verified reading the `script.log` file created automatically in the micro SD card root directory.

Deleting the Error Log

This table describes how to delete the error log file in the logic controller:

Step	Action
1	Create a `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Delete "/sys/log"`
3	Copy the script file to the root folder of the micro SD card.
4	Insert the prepared micro SD card in the logic controller. **Result:** Deleting of the error log file begins. During the operation, the **SD** system LED on the logic controller is on. Do not stop the logic controller while the operation is in progress.
5	When the **SD** system LED is turned off, remove the micro SD card. **Result:** The error log file (`PlcLog.csv`) is deleted from the logic controller.
6	The status of the operation can be verified reading the `script.log` file created automatically in the micro SD card root directory.

Error Log Format

The logic controller provides an error list containing the last 10 detected errors in the log memory. Each error entry into the error log file is composed of four parts:

- Date and time
- Level
- Context
- Error code

After an upload through the micro SD card, the code is represented as in the example below:

```
02/06/14, 12:04:01, 0x01110001
```

This table describes the meaning of the hexadecimal error representation:

Group	Hexadecimal error code	Error description	Result
Operating system	0F01xxxx	Operating system error detected	Transistion to HALTED state
Memory management	0F030009	Internal memory allocation error detected	Transistion to HALTED state
Watchdog timer	0104000A	Logic controller resource utilixation greater than 80% - first detection	Watchdog timeout signaled: ● %s11 set to 1 ERR LED flashes
	0804000B	Logic controller resource utilixation greater than 80% - second consecutive detection	Transistion to HALTED state
	0804000C	User watchdog timer in master task	Transistion to HALTED state
	0804000D	User watchdog timer in periodic task	Transistion to HALTED state
User application	0804000F	Application is not compatible with firmware	Transistion to EMPTY state
	08070010	Checksum error detected	Transistion to EMPTY state
Ethernet	010B0014	Duplicate IP address detected	Duplicate IP signaled ● %SW62 set to 1 ● %SW118.bit9 set to 0. ERR LED flashes

Part III
Programming the M100/M200 Logic Controller

Overview

This part provides information about the system and I/O objects specific to the M100/M200 Logic Controller. These objects are displayed in the **Programming** tab.

For descriptions of all other objects, refer to SoMachine Basic Generic Functions Library Guide.

What Is in This Part?

This part contains the following chapters:

Chapter 10
How to Use the Source Code Examples

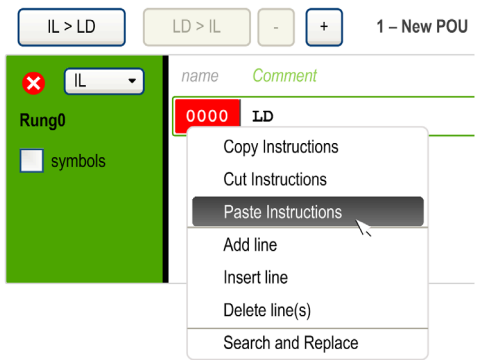
How to Use the Source Code Examples

Overview

Except where explicitly mentioned, the source code examples contained in this book are valid for both the Ladder Diagram and Instruction List programming languages. A complete example may require more than one rung.

Reversibility Procedure

To obtain the equivalent Ladder Diagram source code:

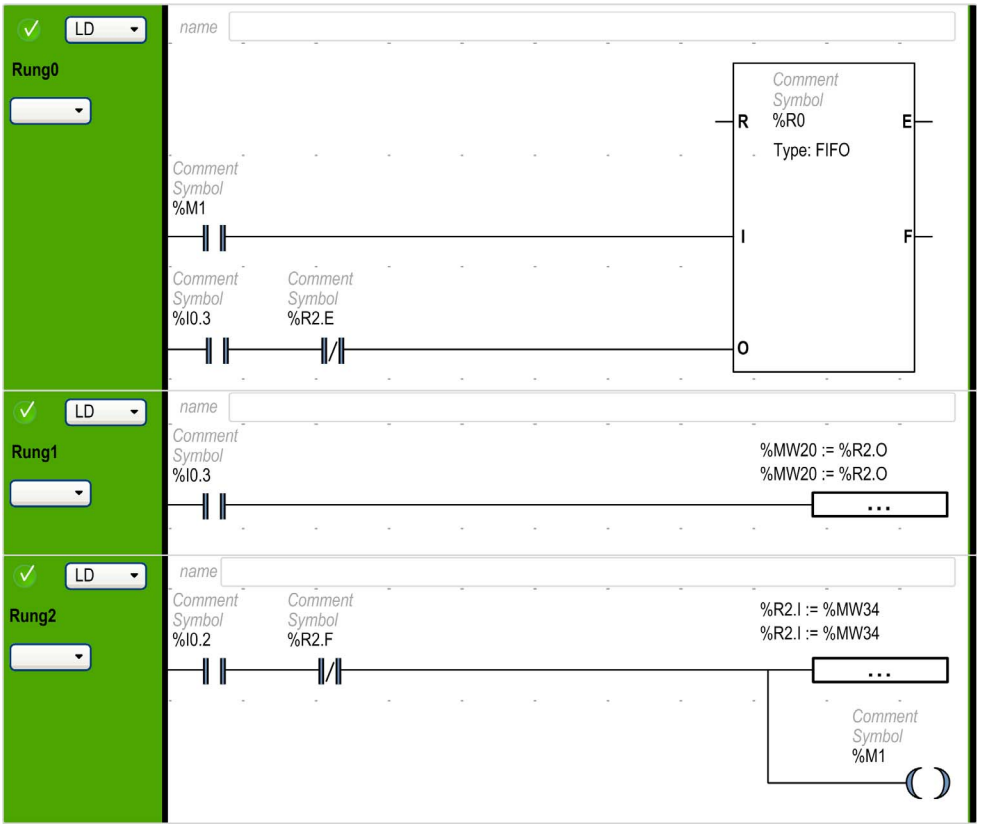
Step	Action
1	Select and copy (**Ctrl+C**) the source code for the first rung of the sample program shown in this manual.
2	In SoMachine Basic, create a new rung by clicking  on the toolbar.
3	In this rung, click the **LD > IL** button to display Instruction List source code.
4	Select the line number **0000**, then right-click and choose **Paste Instructions** to paste the source code into the rung:  **NOTE:** Remember to delete the **LD** instruction from the last line of the rung if you have pasted the instructions by inserting the line(s) before the default LD operator.
5	Click the **IL > LD** button to display the Ladder Diagram source code.
6	Repeat the previous steps for any additional rungs in the sample program.

Example

Instruction List program:

Rung	Source Code
0	BLK %R0 LD %M1 I LD %I0.3 ANDN %R2.E O END_BLK
1	LD %I0.3 [%MW20:=%R2.O]
2	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

Corresponding Ladder Diagram:



Chapter 11
I/O Objects

What Is in This Chapter?

This chapter contains the following topics:

Digital Inputs (%I)

Introduction

Digital input bit objects are the image of digital inputs on the logic controller.

Displaying Digital Input Properties

Follow these steps to display properties of the digital inputs:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects** → **Digital inputs**. **Result**: Digital input properties appear on the screen.

Digital Inputs Properties

This table describes each property of the digital input:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the input channel is being referenced in a program.
Address	No	%I0.i	–	Displays the address of the digital input on the controller, where i represents the channel number. If the controller has n digital input channels, the value of i is given as 0...n-1. For example, `%I0.2` is the digital input at the digital input channel number 2 of the logic controller.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this input. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this address. Double-click in the **Comment** column and type an optional comment to associate with this channel.

Digital Outputs (%Q)

Introduction

Digital output bit objects are the image of digital outputs on the logic controller.

Displaying Digital Output Properties

Follow these steps to display properties of the digital outputs:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects** → **Digital outputs**. **Result**: Digital output properties appear on the screen.

Digital Outputs Properties

This table describes each property of the digital output:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the output channel is being referenced in a program.
Address	No	%Q0.i	–	Displays the address of the digital output on the controller, where i represents the channel number. If the controller has n digital output channels, the value of i is given as 0...n-1. For example, `%Q0.3` is the digital output at the digital output channel number 3 of the logic controller.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this output. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	The comment associated with this address. Double-click in the **Comment** column and type an optional comment to associate with this channel.

Analog Inputs (%IW)

Introduction

Analog input word objects are the digital values of an analog signal connected to the logic controller.

Two 0-10V analog inputs are embedded in the cartridges TMCR2AI2, TMCR2TI2, and TMCR2AM3. The embedded analog inputs use a 10 bits resolution converter so that each increment is approximately 10 mV ($10V/2^{10}$-1). Once the system detects the value 1023, the channel is considered to be saturated.

Refer to the M100/M200 Hardware Guide *(see Modicon M100/M200 Logic Controller, Hardware Guide)* for more details.

Displaying Analog Input Properties

Follow these steps to display properties of the analog inputs:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects** → **Analog inputs**. **Result**: Analog input properties appear on the screen.

Analog Inputs Properties

This table describes each property of the analog input:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the input channel is being referenced in a program.
Address	No	%IW0.i	–	Displays the address of the embedded analog input on the controller, where i represents the channel number. If the controller has n analog input channels, the value of i is given as 0...n-1. For example, `%IW0.1` is the analog input at the analog input channel number 1 of the logic controller.
		%IW0.x0y	–	Displays the address of the analog output channel on the cartridge, where x is the cartridge number and and y is the channel number.

Parameter	Editable	Value	Default Value	Description
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this input. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	The comment associated with this address. Double-click in the **Comment** column and type a comment to associate with this address.

Analog Outputs (%QW)

Introduction

Analog output word objects are the digital values of the analog signals received from the logic controller using cartridges.

Two 0-10 V analog outputs and two 4-20 mA analog outputs are embedded in the cartridges TMCR2AQ2C and TMCR2AQ2V respectively.

One 0-5 V/0-10 V analog voltage output or 4-20 mA analog current output is embedded in the cartridge TMCR2AM3.

Refer to Modicon M100/M200 Logic Controller Hardware Guide *(see Modicon M100/M200 Logic Controller, Hardware Guide)* for more details.

Displaying Analog Output Properties

Follow these steps to display properties of the analog outputs:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects** → **Analog outputs**. **Result**: Analog output properties appear on the screen.

Analog Outputs Properties

This table describes each property of the analog output:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the output channel is being referenced in a program.
Address	No	%QW0.x0y	–	Displays the address of the analog output channel on the cartridge, where x is the cartridge number and y is the channel number.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this output. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	The comment associated with this address. Double-click in the **Comment** column and type a comment to associate with this address.

Chapter 12
Function Blocks

What Is in This Chapter?

This chapter contains the following sections:

Section 12.1
Fast Counter (%FC)

Using Fast Counter Function Blocks

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

What Is in This Section?

This section contains the following topics:

Description

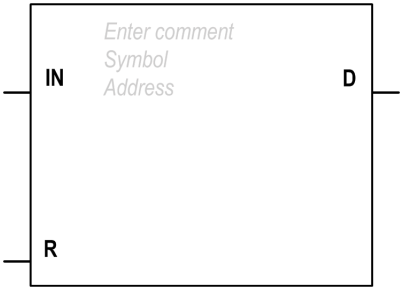
Introduction

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 `%FC.P` or `%FC.PD` if configured as a down-counter. The Done bit `%FC.D` is set back to its default value.

Outputs

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

Label	Description	Value
D	Done (`%FCi.D`)	This bit is set to 1 when: ● `%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. This read-only bit is reset only by setting `%FCi.R` to 1.

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	`%FCi` `Fast Counter` address	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 *(see page 26)* for the maximum number of `Fast Counters`.
Input	`%IO.i`	The dedicated input associated with this function block instance. `%IO.2...%IO.5`
Symbol	Symbol	The symbol associated with this object. Refer to the SoMachine Basic Operating Guide (Defining and Using Symbols) *(see SoMachine Basic, Operating Guide)* for details.
Configured	Whether to count up or down	Set to one of: ● **Not used** ● **Up Counter** ● **Down Counter**
Preset	Preset value (`%FCi.P` or `%FCi.PD`)	Initial value may be set: ● Using associated object `%FCi.P` from 0 to 65535 in single word mode, ● Using associated object `%FCi.PD` from 0 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 all the `Fast Counter` attributes with the values configured or user application. Refer to System Bits (%S) *(see page 358)*.
Effect of controller stops	The `Fast Counter` continues to count with the parameter settings enabled at the time the controller was stopped.

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 %FC0 is the dedicated input %I0.2. When the preset value is reached, %FC0.D is set to 1 and retains the same value until %FC0.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 page 159)* to obtain the equivalent Ladder Diagram.

Section 12.2
High Speed Counter (%HSC)

Using High Speed Counter Function Blocks

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

What Is in This Section?

This section contains the following topics:

Description

Introduction

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

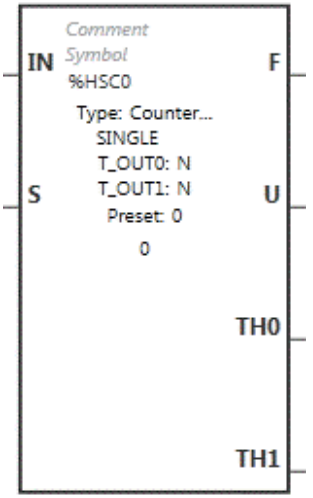
- Single Phase
- Dual Phase [Pulse / Direction]
- Dual Phase [Clock Wise / Counter Clock Wise]
- Dual Phase [Quadrature X1]
- Dual Phase [Quadrature X2]
- Dual Phase [Quadrature X4]
- Frequency Meter

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

The `High Speed Counter` function block uses dedicated inputs and auxiliary inputs and outputs. Refer to the M100/M200 Logic Controller - Hardware Guide *(see Modicon M100/M200 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 page 78)*.

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 rising edge, initializes the present value with the preset value.: 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.	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 **NOTE:** It is modulo value when configured to Modulo- Loop mode.	See Auxiliary Inputs *(see page 184)*.
`%HSCi.S0` `%HSCi.S0D`	WORD DOUBLE WORD	Threshold 0	See Output Threshold in Counting Modes *(see page 184)*.
`%HSCi.S1` `%HSCi.S1D`	WORD DOUBLE WORD	Threshold 1	See Output Threshold in Counting Modes *(see page 184)*.
`%HSCi.T`	WORD	Time base	See High Speed Counter in Frequency Meter Mode *(see page 88)*.
`%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.

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

Label	Description	Value
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	See High Speed Counter in Counting Modes *(see page 183)*.
`%HSCi.C` `%HSCi.CD`	WORD DOUBLE WORD	Capture value	See Auxiliary Inputs *(see page 184)*.
`%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. Refer to the maximum number of objects *(see page 26)* for the number of available `High Speed Counter` objects.
Symbol	User-defined text	The symbol that uniquely identifies this object. For details, refer to the SoMachine Basic Operating Guide (Defining and Using Symbols) *(see SoMachine Basic, Operating Guide)*.
Preset	• from 0 to 65535 for `%HSCi.P` • from 0 to 4294967295 for `%HSCi.PD`	Preset value to initialize the `HSC` current value (`%HSCi.P`, `%HSCi.PD`). Not valid for the Frequency Meter.

Property	Value	Description
S0	• from 0 to 65535 for %HSCi.S0 • from 0 to 4294967295 for %HSCi.S0D	Threshold value 0 is used as a comparator with the current value. The value of S0 must be less than S1 (%HSCi.S1).
S1	• from 0 to 65535 for %HSCi.S1 • from 0 to 4294967295 for %HSCi.S1D	Threshold value 1 is used as a comparator with the current 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 presents a list of special cases when programming the High Speed Counter function block:

Special Case	Description
Effect of cold restart (%S0=1)	Resets all the High Speed Counter attributes with the values configured by the program.
Effect of controller stop	The High Speed Counter stops its function and the outputs stay in their current state. **NOTE:** When the controller stops, the reflex outputs stay in their current state only if the fallback behavior of the tasks is configured to maintain values of the outputs. For more information on configuring fallback behavior, refer to Fallback Behavior *(see SoMachine Basic, Operating Guide)*.

High Speed Counter in Counting Modes

Introduction

The `High Speed Counter` function block works at a maximum frequency of 100 kHz, with a range of 0 to 65535 in single word mode and 0 to 4294967295 in double word mode.

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

Function	Description	Input type	%HSC0	%HSC1	%HSC2	%HSC3
Single Phase	The pulses are applied to the physical input associated to **Pulse Input**.	**Pulse Input**	%I0.0	%I0.6	%I0.1	%I0.7
Dual Phase [Pulse / Direction]	The pulses are applied to the physical input associated to **Pulse Input**, the current operation (upcount/downcount) is given by the state of the **Direction Input**.	**Pulse Input**	%I0.0	%I0.6	–	–
		Direction Input	%I0.1	%I0.7	–	–
Dual Phase [Clock Wise / Counter Clock Wise]	The pulses are applied to the physical inputs associated to **Clock Wise Input** and **Counter Clock Wise Input**.	**ClockWise Input**	%I0.0	%I0.6	–	–
		CounterClockWise Input	%I0.1	%I0.7	–	–
Dual Phase [Quadrature X1]	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	–	–
Dual Phase [Quadrature X2]	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	–	–
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	–	–

Output Thresholds

During counting, the current value is compared to two thresholds: %HSCi.S0 or %HSCi.S0D and %HSCi.S1 or %HSCi.S1D.

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 Single Phase and Dual Phase *(see page 82)*.

%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

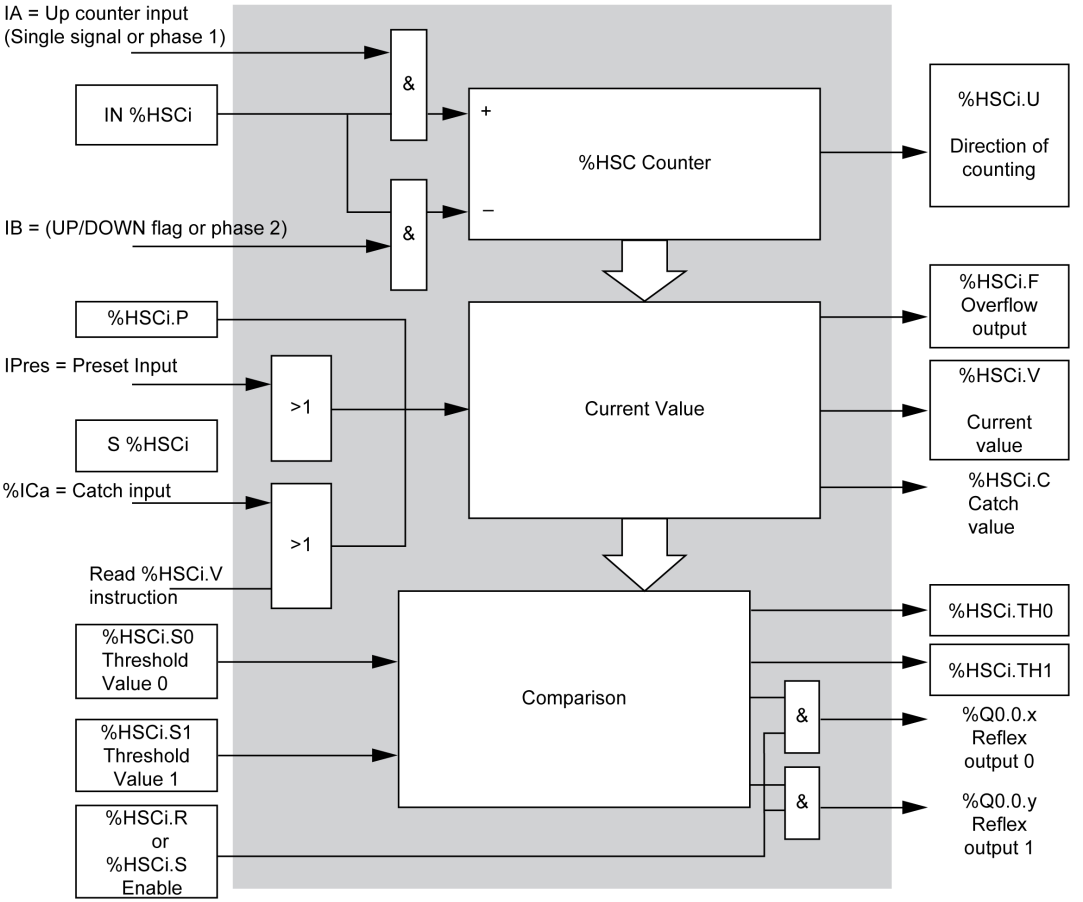
Counting operations are made on the rising edge of pulses, and only if the counting function block is enabled.

There are two optional inputs used in counting mode: **Catch Input** and **Preset Input**:
- 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.
- When the **Preset Input** is active, the current value is affected in the following ways:
  - %HSCi.V and %HSCi.VD are respectively written with the content of %HSCi.P or %HSCi.PD.
  - For frequency counting, %HSCi.V or %HSCi.VD is set to 0.

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):



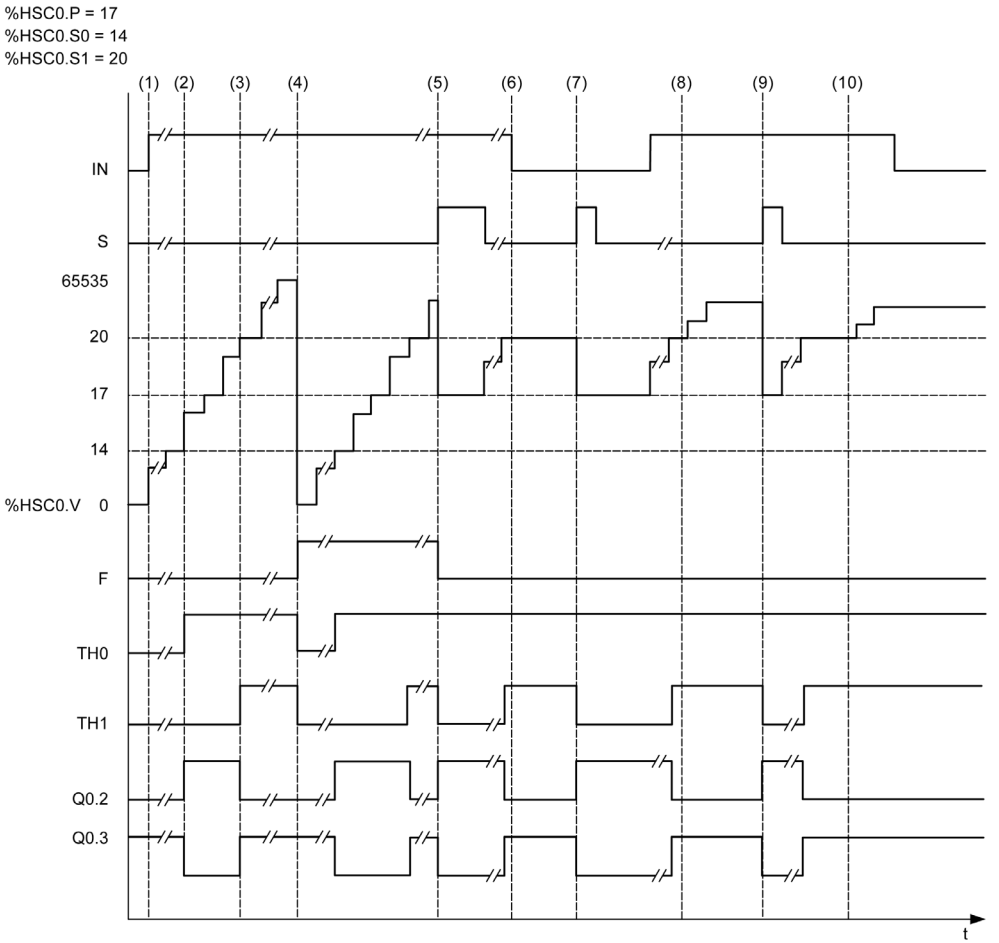
NOTE: Reflex outputs are managed independently from the controller cycle time.

Single Phase Timing Diagram

Reflex output configuration example:

Reflex Output	Value < %HSC0.S0	%HSC0.S0 <= Value < %HSC0.S1	Value >= %HSC0.S1
%Q0.4	–	X	–
%Q0.5	X	–	X

Timing diagram:



(1) IN is set to 1: the counting function is activated (%HSC0.U = 1 because %HSC0 is an up-counter)
(2) %Q0.4 (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

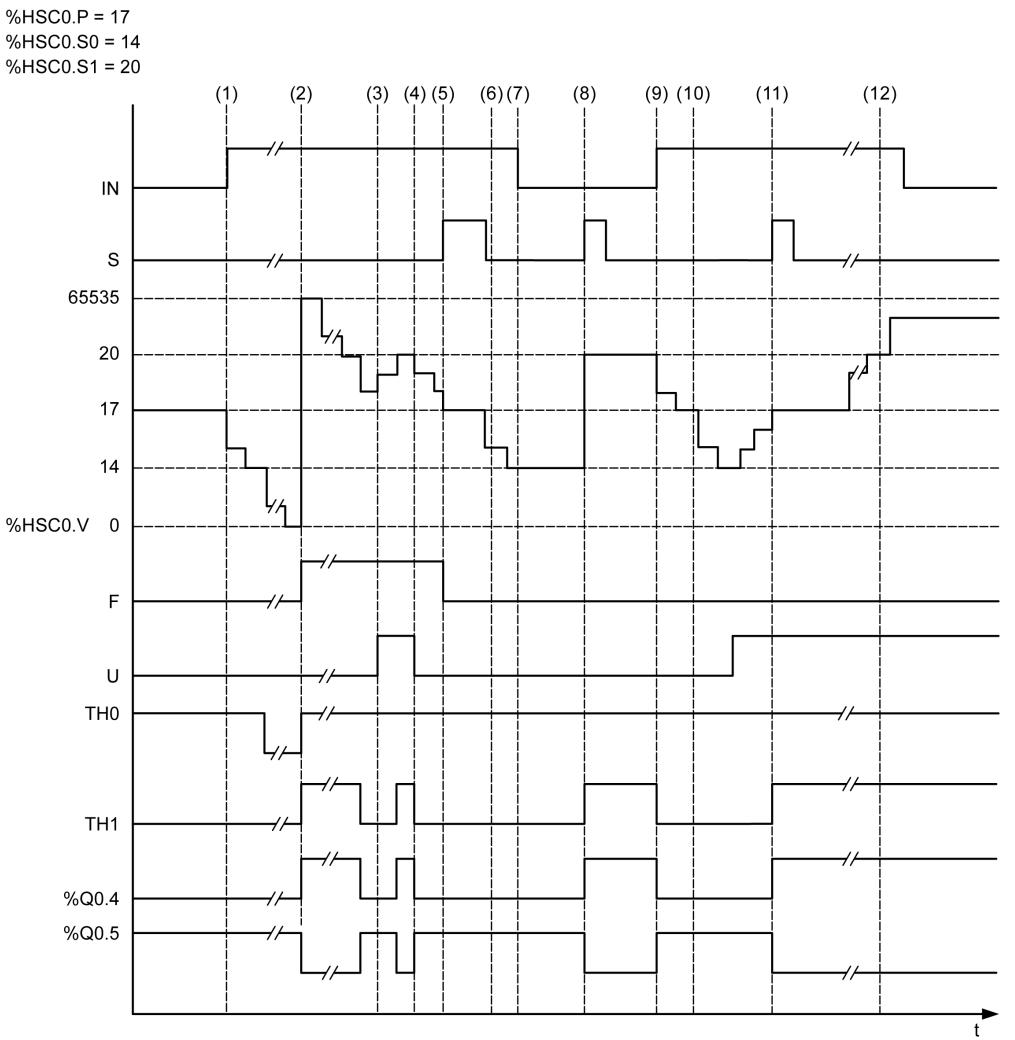
(5) `S` is set to 1, the current value, `%HSC0.V`, is set to preset value.
(6) The current function is inhibited while `IN` is set to 0
(7) While the function is inhibited, `S` is set to 1 so the current value is set to preset value 17
(8) Change of threshold value `S1` to 17
(9) `S` is set to 1 so the new value of `S1` will be granted at the next count
(10) Catch input is set to 1 so `%HSC0.C = 20`

Dual Phase [Pulse / Direction] Timing Diagram

Reflex output configuration example:

Reflex Output	Value < %HSC0.S0	%HSC0.S0 <= Value < %HSC0.S1	Value >= %HSC0.S1
`%Q0.4`	–	–	X
`%Q0.5`	X	X	–

Timing diagram:

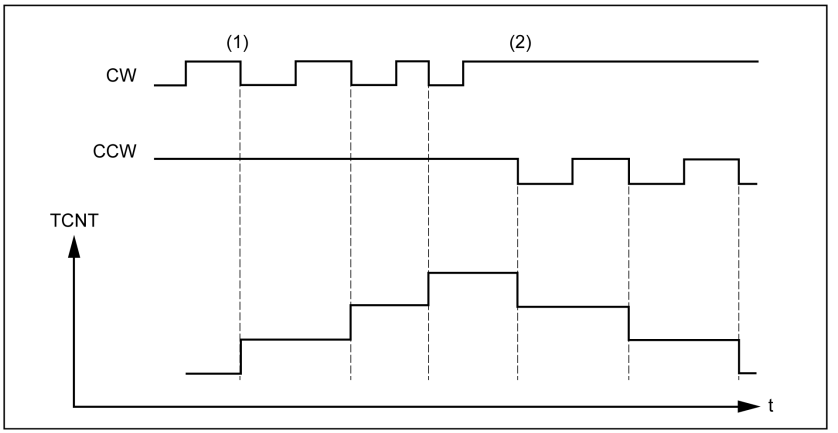


(1) Input IN is set to 1 so down-counting mode starts (`%HSC0.U = 0` that is, `IB = 0`)
(2) The current value reaches 0 so `F` output flag is set to 1 and `%HSC0.V` is set to 65535 at the next count
(3) Change at the `IB` input, the counter is now in up counting mode and `%HSC0.U = 1`
(4) `IB` input is set to 0 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
(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 preset value 17
(12) A catch of the current value `%HSC0.V` is made so `%HSC0.C = 20`

Dual Phase [Clock Wise / Counter Clock Wise] Timing Diagram

Timing diagram:

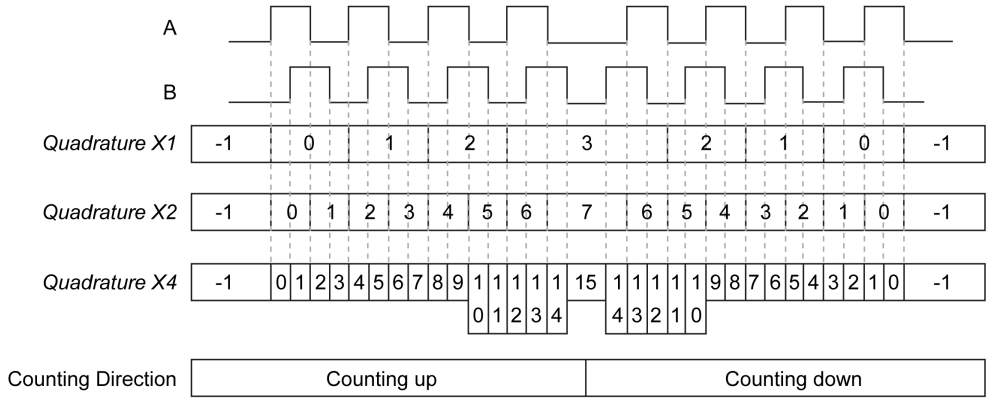


(1) Up-counting starts
(2) Down-counting starts

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

The encoder signal is counted according to the input mode selected, as shown below:
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.

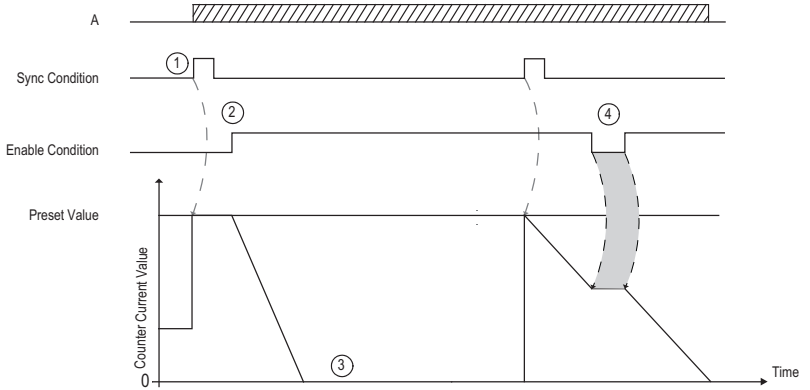
Quadrature X4 The counter increments or decrements on each edge of channels A and B. Whether the counter increments or decrements depends on which channel leads the other. Each cycle results in 4 increments or decrements.

One-shot Counting Mode

Overview

- Only one input is required for pulse.
- On the rising edge of the preset condition, the counter is enabled and the current value is set to the preset value.
- When counter is enabled, each pulse applied to the input increments the current value. The counter stops when its current value reaches the maximum value. The overflow flag is then set.
- The counter value remains at maximum value even if new pulses are applied to the input.
- A new preset is needed to activate the counter again.

One-shot Mode Timing Diagram



This table explains the stages from the preceding graphic:

Stage	Action
1	On the rising edge of the Preset condition, the preset value is loaded in the counter (regardless of the current value) and the counter is enabled.
2	When Enable condition = 1, the current counter value increments on each pulse on input A until it reaches 0.
3	The counter waits until the next rising edge of the Preset condition. **Note:** At this point, pulses on input A have no effect on the counter.
4	When Enable condition = 0, the counter ignores the pulses from input A and retains its current value until the Enable condition = 1. The counter resumes counting pulses from input A on the rising edge of the Enable input from the held value.

Modulo-loop Counting Mode

Overview

The **Modulo-loop** mode can be used for repeated actions on a series of moving objects, such as packaging and labeling applications.

Principle

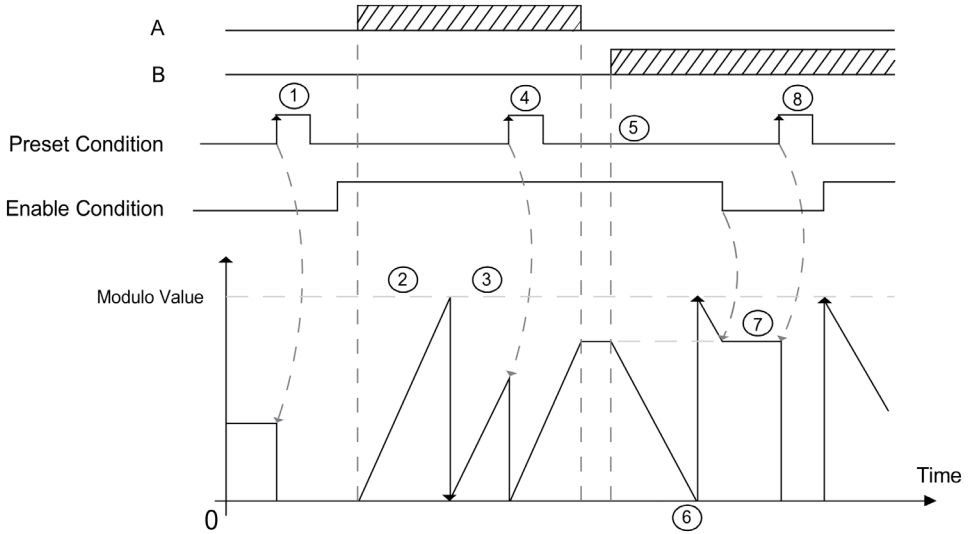
On a rising edge of the preset condition, the counter is enabled and the current value is reset to 0.

When counter is enabled:

Incrementing direction: the counter increments until it reaches the modulo value. At the next pulse, the counter is reset to 0, a modulo flag is set to 1, and the counting continues.

Decrementing direction: the counter decrements until it reaches 0. At the next pulse, the counter is set to the modulo value, a modulo flag is set to 1, and the counting continues.

Modulo-loop Mode Timing Diagram



Stage	Action
1	On the rising edge of preset condition, the current value is reset to 0 and the counter is enabled.
2	When Enable condition = 1, each pulse on A increments the counter value.
3	When the counter reaches the (modulo-1) value, the counter loops to 0 at the next pulse and the counting continues. `Modulo_Flag` is set to 1.
4	On the rising edge of preset condition, the current counter value is reset to 0.

Stage	Action
5	When Enable condition = 1, each pulse on B decrements the counter.
6	When the counter reaches 0, the counter loops to (modulo-1) at the next pulse and the counting continues.
7	When Enable condition = 0, the pulses on the inputs are ignored.
8	On the rising edge of preset condition, the current counter value is reset to 0.

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.

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

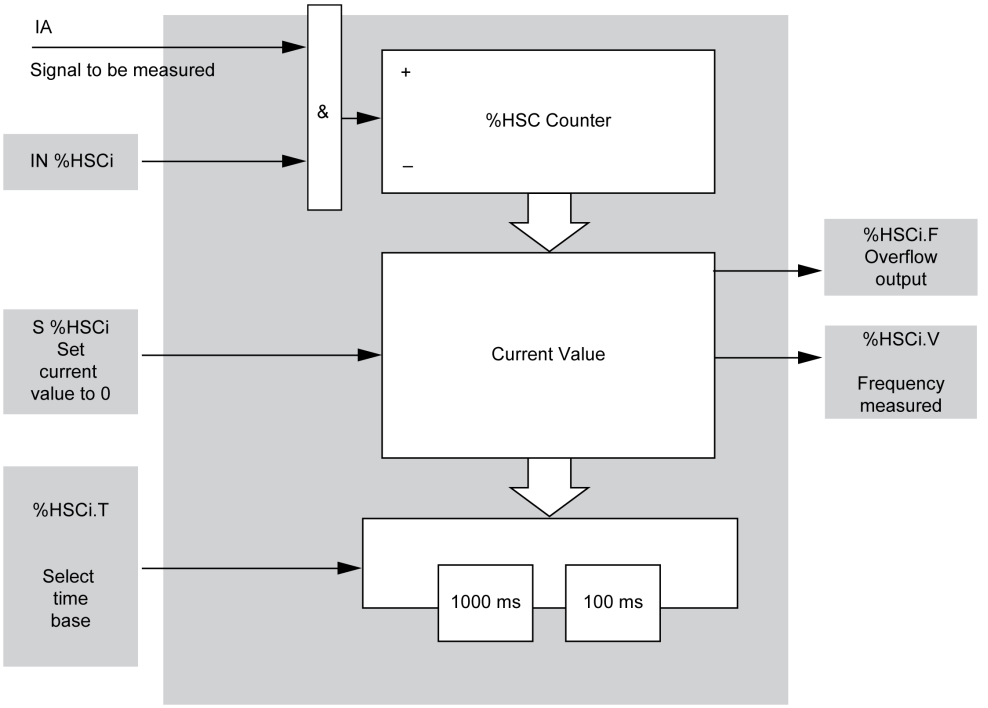
Time base	Accuracy	Update
100 ms	0.05% for 100 kHz 10% for 100 Hz	10 times per second
1 s	0.005% for 100 kHz 10% for 10 Hz	Once per second

Accuracy Measurement

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

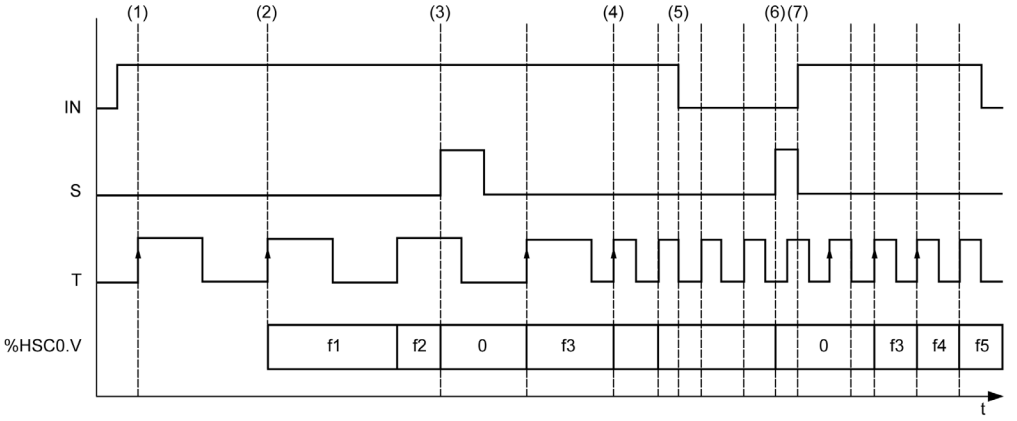
Operation

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` is updated after one period of the `TB`
(3) On input `S` rising edge, the current value `%HSC0.V` is set to 0
(4) `%HSC0.T` is set to 100 ms, so the current 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` is held
(6) On input `S` rising edge, the current value `%HSC0.V` is set to 0
(7) `S` is set to 0 and `IN` is set to 1, so the measurement starts at the next rising edge of the `TB` signal

f*x* corresponds to the current frequency value.

Section 12.3
Pulse (%PLS)

Using Pulse Function Blocks

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

What Is in This Section?

This section contains the following topics:

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 M100/M200 Logic Controller - Hardware Guide for more information on inputs and outputs.

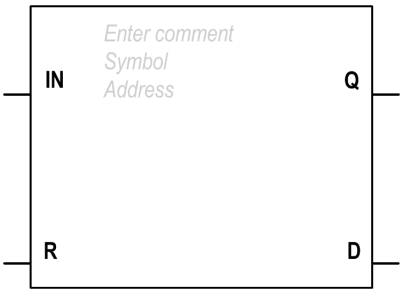
The `Pulse` function block allows only a single signal width, or duty cycle, of 50%.

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 page 68)*.

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 `Pulse` 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.

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 `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 is from 0 to the number of objects available on this logic controller. Refer to Maximum Number of Objects table *(see page 26)* for the maximum number of `Pulse` objects.
Symbol	Symbol	The symbol associated with this object. Refer to the SoMachine Basic Operating Guide (Defining and Using Symbols) *(see SoMachine Basic, Operating Guide)* for details.
Preset	Preselection of the period (`%PLSi.P`)	• 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`	This value gives the percentage of the signal in state 1 in a period. The width Tp is thus equal to: `TP = T x (%PLSi.R:100)`. The user application writes the value for `%PLSi.R`. It is this word which controls the duty cycle of the period. The default value is 0 and values greater than 100 are considered to be equal to 100.
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` x.**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

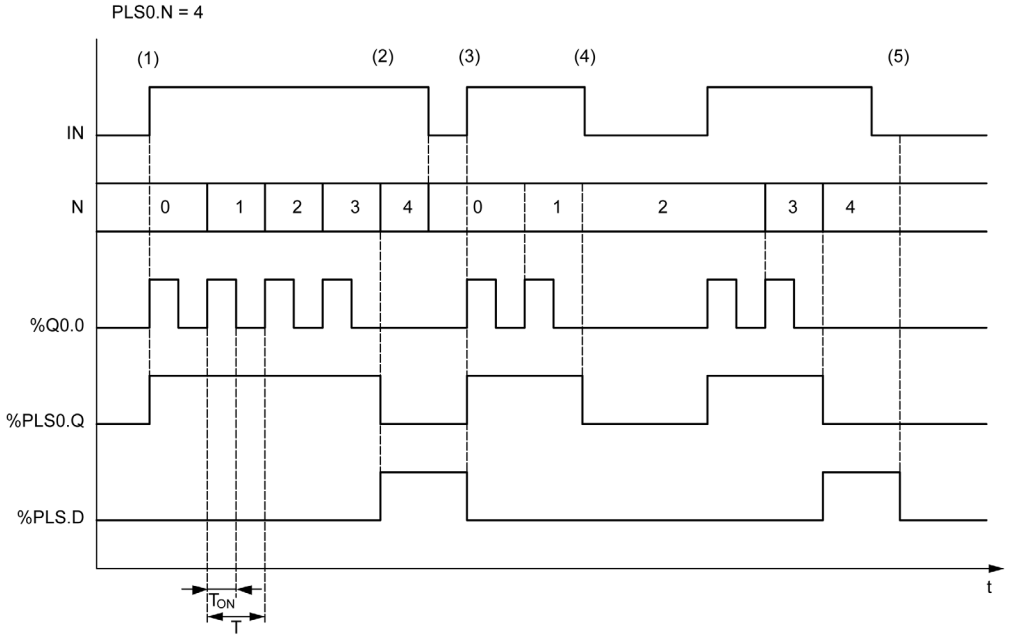
The **Time Base** is set on the **Configuration** → **Pulse Generators** and cannot be modified. Refer to Configuring Pulse Generators *(see page 68)*.

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 pulses 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 pulses 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=1`)	● Pulse generation is stopped.
	● During the controller initialization, output is reset to 0.
	● If after the controller initialization:
	  ● the controller enters the STOPPED state, the fallback behavior is applied to the output.
	  ● the controller enters the RUN state, the configuration parameters are restored.

Effect at controller stop	● Pulse generation is stopped ● The **Fallback** behavior is applied to the output.
Effect of online modification	None

Programming Example

Introduction

The `Pulse` function block can be configurated 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 page 159)* to obtain the equivalent Ladder Diagram.

Section 12.4
Pulse Width Modulation (%PWM)

Using Pulse Width Modulation Function Blocks

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

What Is in This Section?

This section contains the following topics:

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.

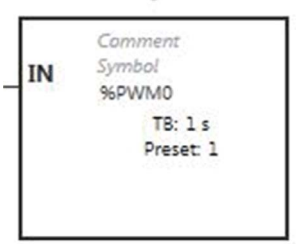
Controllers with relay outputs for these two channels do not support this function.

`%PWM0` uses dedicated output `%Q0.0` and `%PMW1` 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 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 page 68)*.

Illustration

This illustration is 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 page 68)*.

To configure the `Pulse Generator` resource as a PWM, refer to PWM Configuration *(see page 73)*.

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

Property	Value	Description
Used	Activated / deactivated checkbox	Indicates whether the address is in use.
Address	`%PWMi`,where i is 0 or 1	i is the instance identifier.
Symbol	User-defined text	The symbol that uniquely identifies this object. For details, refer to the SoMachine Basic Operating Guide (Defining and Using Symbols) *(see SoMachine Basic, Operating Guide)* .
Preset	• `%PWMi.P`=1 if **Time Base** = 1 s • 1<=`%PWMi.P`<=100 if **Time Base**= 10 ms • 1<=`%PWMi.P`<=1000 if **Time Base** = 1 ms • 1<=`%PWMi.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 `%PWMi.R` and is the percentage of the signal in state 1 in the period. The width of state 1 (Tp) is thus equal to: `TP = T x (%PWMi.R/100)`. The user application writes the value for `%PWMi.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:
- changed, 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 on the **Configuration** → **Pulse Generators** and can only be modified under the **Configuration** tab. Refer to Configuring Pulse Generators *(see page 68)*.

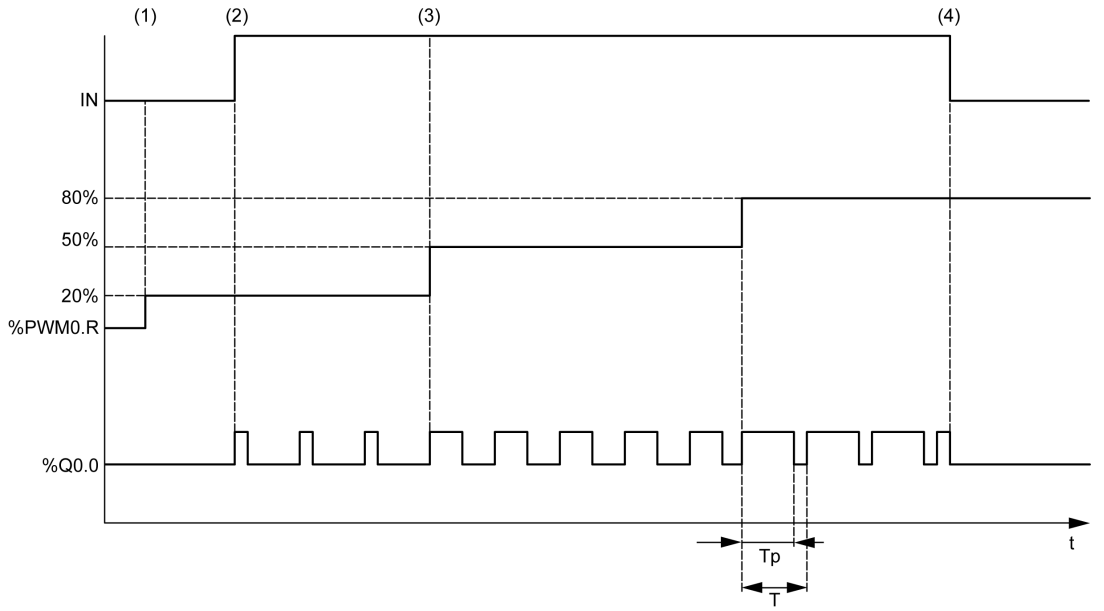
The output signal period `T` is set with the **Preset** and **Time Base** parameters such that `T = %PWMi.P` x **Time Base**.

This table shows the range of available periods:

Time Base	Frequency
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 displays 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 (`Tp`) 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`)	● Pulse generation is stopped.
	● During the controller initialization, output is reset to 0.
	● If after the controller initialization:
	  ● the controller enters the STOPPED state, the fallback behavior is applied to the output.
	  ● the controller enters the RUN state, the configuration parameters are restored.

Effect at controller stop	● Pulse generation is stopped. ● The fallback behavior is applied to the output.
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% x 500 ms = 100 ms.
- If `%I0.0` is set to 0 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 page 159)* to obtain the equivalent Ladder Diagram.

Chapter 13
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 13.1
Description

Pulse Train Output (PTO)

Introduction

The M200 PTO function provides two 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 one or two 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 PTO, PWM (pulse width modulation), and PLS (pulse) functions use the same dedicated outputs. Only one of these three functions can be used on the same channel. Using different functions on channel 0 and channel 1 is allowed.

A PTO channel can use up to:
- five physical inputs, if optional interface signals for homing (Ref), event (Probe), limits (LimP, LimN), or drive interface (DriveReady) are used,
- three physical outputs, if optional drive interface signal (DriveEnable) is used, along with the two PTO fast outputs.

Automatic origin offset is also managed to improve positioning accuracy. Diagnostics are available for status monitoring, allowing a comprehensive and quick troubleshooting.

Supported Functions

The two 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)
- limits (hardware and software)
- diagnostics

PTO Function Blocks

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

Category	Subcategory	Function Block	Description
Motion (single axis)	Power	MC_Power_PTO *(see page 258)*	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.
	Movement	MC_MoveVel_PTO *(see page 262)*	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 266)*	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 271)*	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.
	Stopping and position	MC_Home_PTO *(see page 282)*	Commands the axis to perform the sequence defining the absolute reference position, and transfers the axis to the state Homing *(see page 229)*. The details of this sequence depend on Homing configuration parameters setting.
		MC_SetPos_PTO *(see page 285)*	Modifies the coordinates of the axis without any physical movement.
		MC_Stop_PTO *(see page 275)*	Commands a controlled motion stop and transfers the axis to the state Stopping. It aborts any ongoing move execution.
		MC_Halt_PTO *(see page 278)*	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	Subcategory	Function Block	Description
Administrative	Status	MC_ReadActVel_PTO *(see page 288)*	Returns the value of the velocity of the axis.
		MC_ReadActPos_PTO *(see page 290)*	Returns the value of the position of the axis.
		MC_ReadSts_PTO *(see page 292)*	Returns the state diagram *(see page 248)* status of the axis.
		MC_ReadMotionState_PTO *(see page 294)*	Returns the motion status of the axis.
	Error handling	MC_ReadAxisError_PTO *(see page 302)*	Returns an axis control error, if any.
		MC_Reset_PTO *(see page 304)*	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.
	Probe	MC_TouchProbe_PTO *(see page 297)*	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 299)*	Aborts function blocks which are connected to trigger events (for example, MC_TouchProbe_PTO).
	Parameters	MC_ReadPar_PTO *(see page 307)*	Gets parameters from the PTO.
		MC_WritePar_PTO *(see page 309)*	Writes parameters to the PTO.

NOTE: The motion function blocks act on the position of the axis according to the motion state diagram *(see page 248)*. The administrative function blocks do not influence the motion state.

NOTE: The MC_Power_PTO 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.

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 (`%I0.2` for `%PTO0` and `%I0.5` for `%PTO1`)
  - Probe input (`%I0.3` for `%PTO0` and `%I0.4` for `%PTO1`)
- Three are assigned to the `MC_Power_PTO` 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` 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 (respectively `%Q0.0` and `%Q0.1` for `%PTO0` only)
  - Pulse / Direction (respectively `%Q0.0` and `%Q0.2` for `%PTO0` and/or `%Q0.1` and `%Q0.3` for `%PTO1`)
- The other output, `DriveEnable`, is associated with the `MC_Power_PTO` function block. It has no fixed assignment and is written with all other outputs.

The PTO function has the following characteristics:

Characteristic	Value
Number of channels	2
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)

Section 13.2
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:

PTO Configuration

Overview

To configure the `Pulse Generator` resource, refer to Configuring Pulse Generators *(see page 68)*.

To configure the `Pulse Generator` resource as a PTO, refer to PTO Configuration *(see page 74)*.

Pulse Output Modes

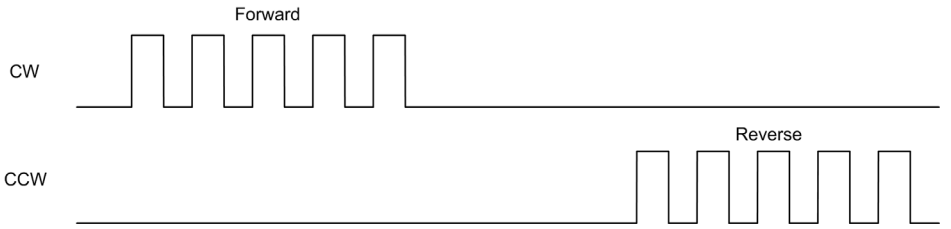
Overview

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).

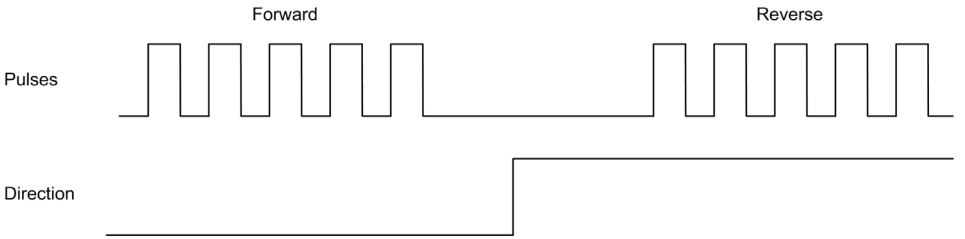


NOTE: `PTO1` is not available when choosing this mode.

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`).



Special Cases

Special Case	Description
Effect of a cold restart (`%S0=TRUE`)	• The axis is set to `Disabled` state. • The PTO function blocks are initialized.
Effect at controller stop	The axis is set to `ErrorStop` state.
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

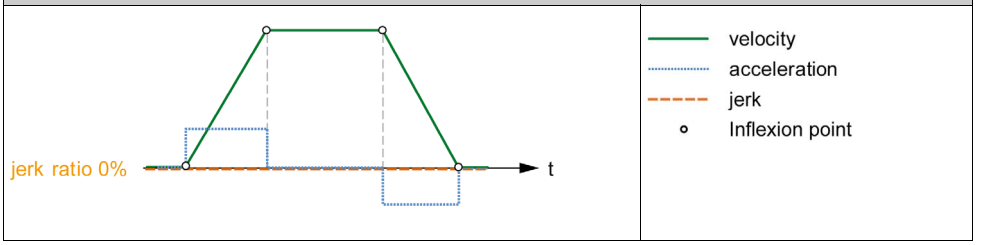
Acceleration is the rate of velocity change, starting from **Start Velocity** to target velocity. Deceleration is the rate of velocity change, starting from target velocity to **Stop Velocity**. These velocity changes are implicitly managed by the PTO function in accordance with `Acceleration`, `Deceleration` 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.

`JerkRatio` 0%: Constant acceleration / deceleration.

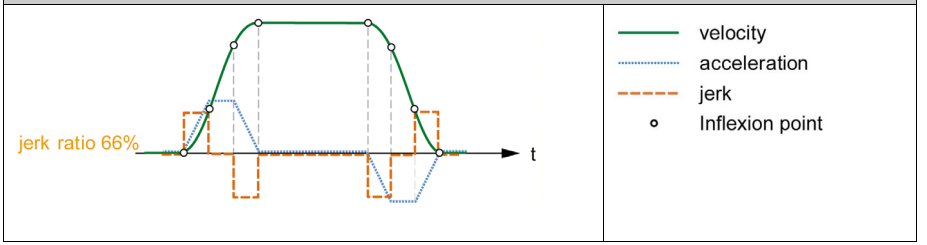


Acceleration / Deceleration Ramp with an S-curve Profile

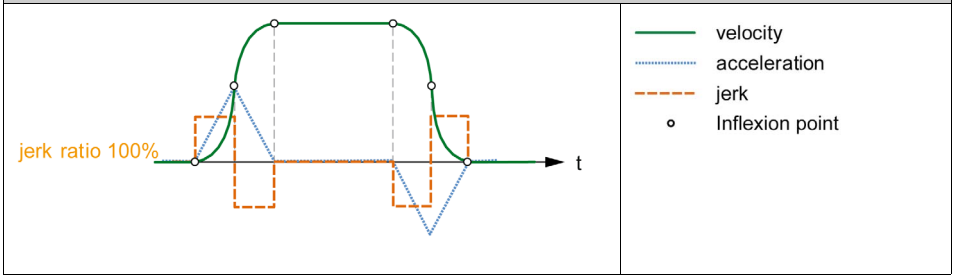
When the jerk ratio parameter is greater than 0, the acceleration / deceleration ramp has an S-curve profile.

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:

`JerkRatio` 66%: 2/3 of the acceleration and deceleration time is spent in increasing and decreasing the acceleration and deceleration value.



`JerkRatio` 100%: The entire time is spent in increasing and decreasing the acceleration and deceleration value.



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: The `JerkRatio` is re-calculated to respect the `MaxAccelerationAppl` and `MaxDecelerationAppl` parameters.

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 *(see page 213)*.

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.

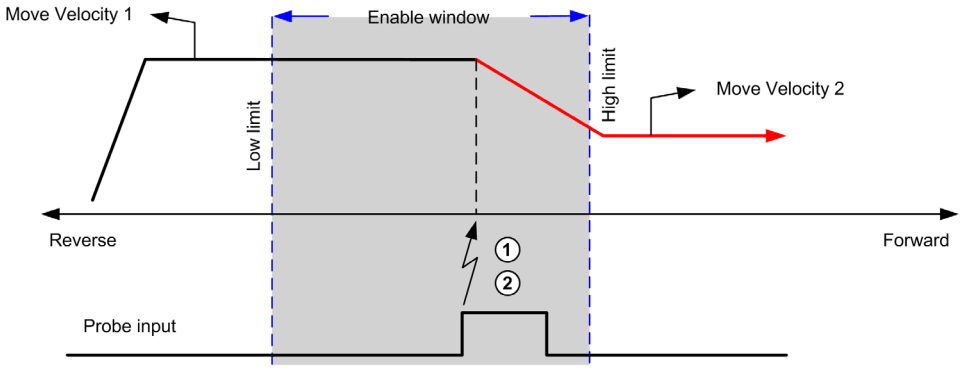
Position Capture

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

Motion Trigger

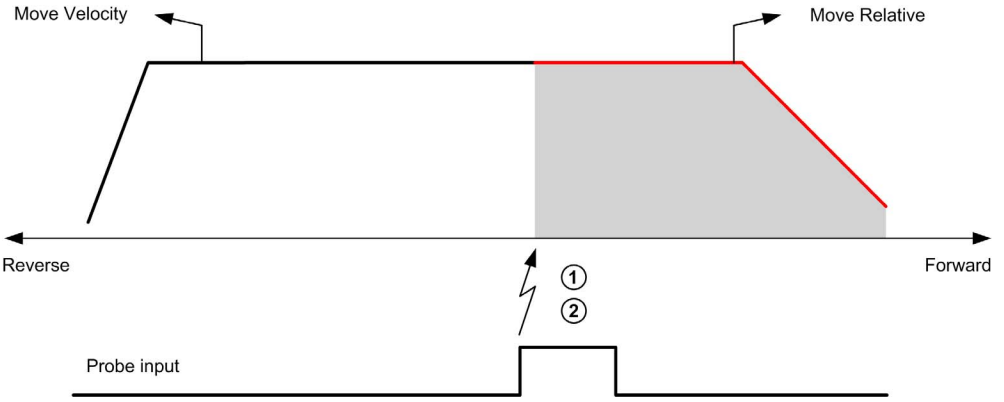
The `BufferMode` input of a motion function block must be set to `seTrigger`.

This example illustrates a change target velocity with enable window:



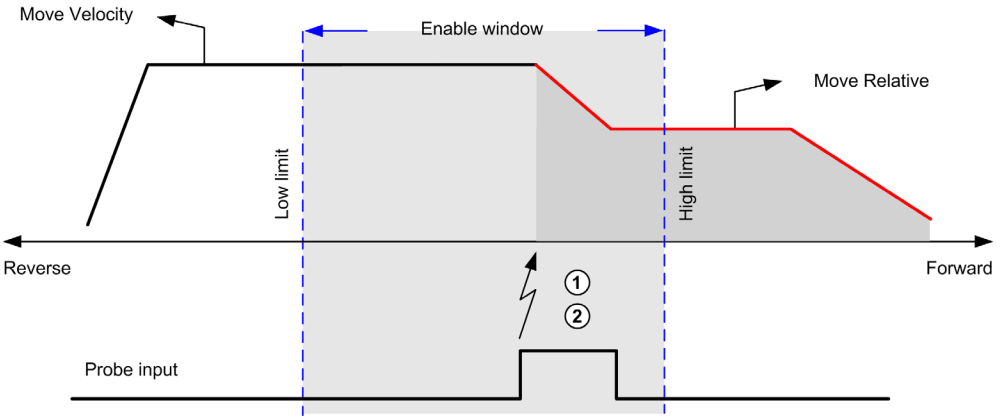
1 Capture the position counter value
2 Trigger `Move Velocity` function block

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



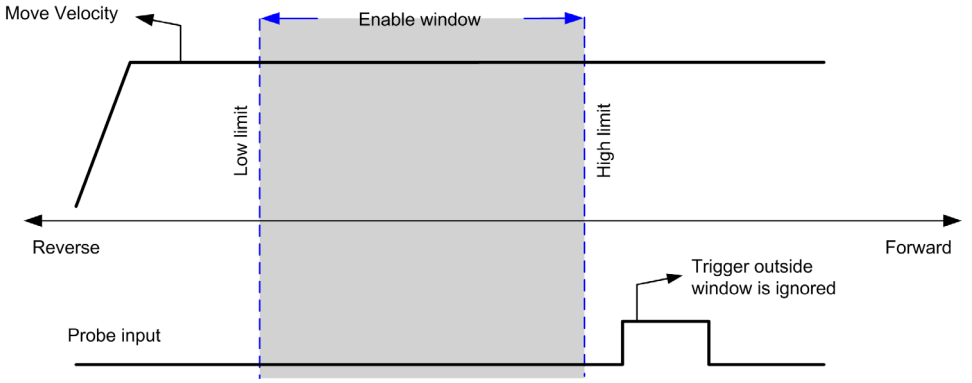
1 Capture the position counter value
2 Trigger `Move Relative` function block

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:



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

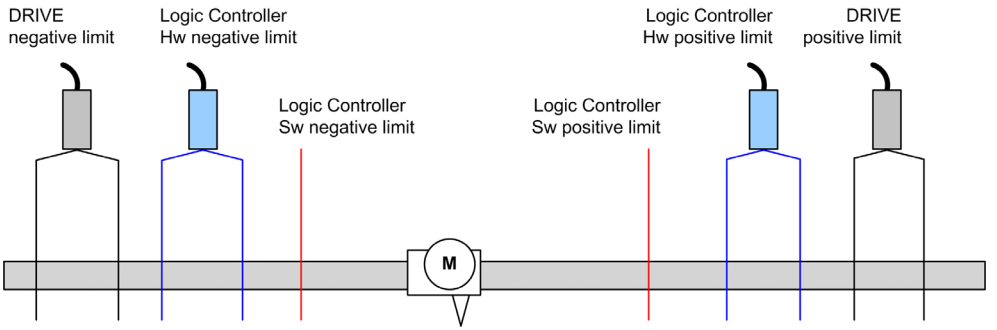
⚠ WARNING

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.

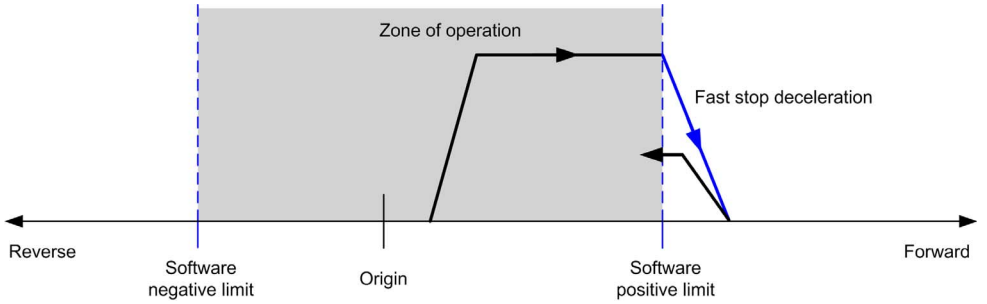
The figure illustrates hardware and software limit switches:



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 302)* and Axis Control Advisory Alerts *(see page 244)*.
- 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 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.



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 243)*).

NOTE: When enabled, the software limits are valid after an initial homing is successfully performed (that is, the axis is homed, `MC_Home_PTO` *(see page 282)*).

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 13.3
Home Modes

Overview

This section describes the PTO home modes.

What Is in This Section?

This section contains the following topics:

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 M200 PTO channels provide several standard homing movement types:

- position setting *(see page 233)*,
- long reference *(see page 234)*,
- short reference no reversal *(see page 236)*,
- short reference reversal *(see page 238)*,
- short reference with INDEX *(see page 238)*.

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

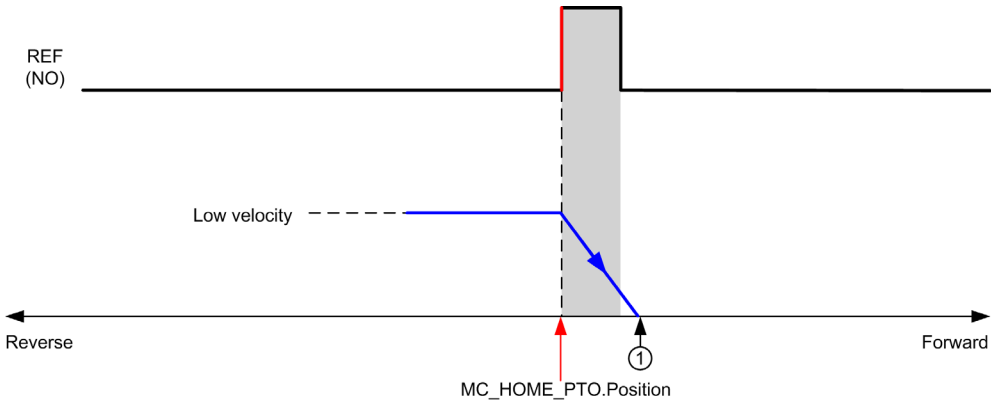
- `%MC_ReadSts_PTO.IsHomed` is set to TRUE when a homing movement is finished successfully. If the homing movement is interrupted, it needs to be started again.
- `%MC_ReadSts_PTO.IsHomed` is set to FALSE when the axis state is DISABLED, or when no homing movement was finished successfully.

Refer to `MC_Home_PTO` *(see page 282)* and home modes function block object codes *(see page 242)*.

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.

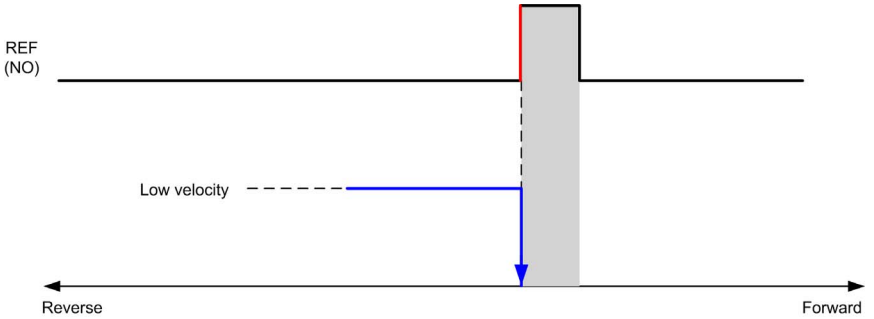
The actual position of the axis at the end of the motion sequence may therefore differ from the position parameter set on the function block:



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:



REF (NO) Reference point (Normally Open)

Limits

Hardware limits are necessary for the correct functioning of the `MC_Home_PTO` function block (Positioning Limits *(see page 226)* and `MC_Power_PTO` *(see page 258)*). 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

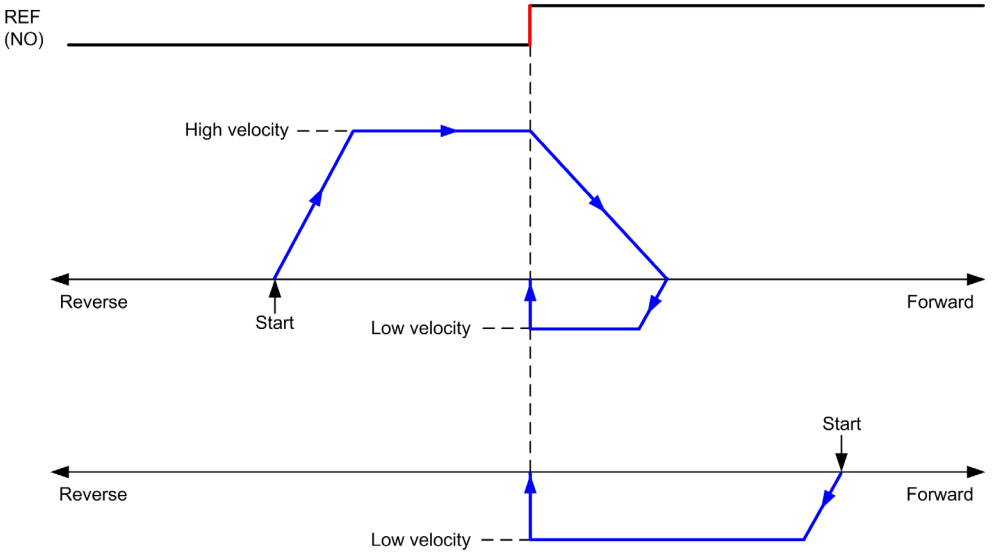
In the case of position setting, the current position is set to the specified position value. No move is performed.

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:

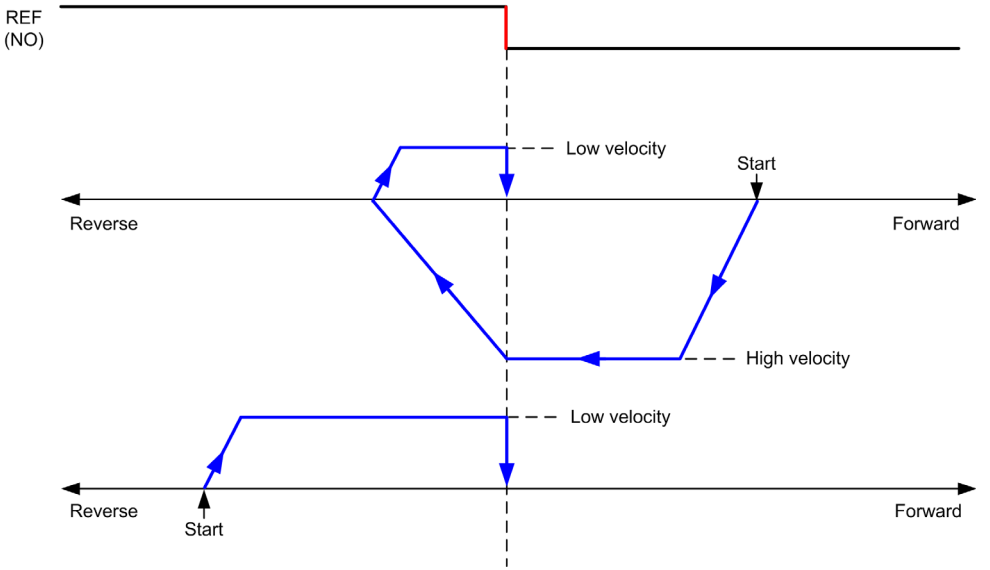


REF (NO) Reference point (Normally Open)

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:

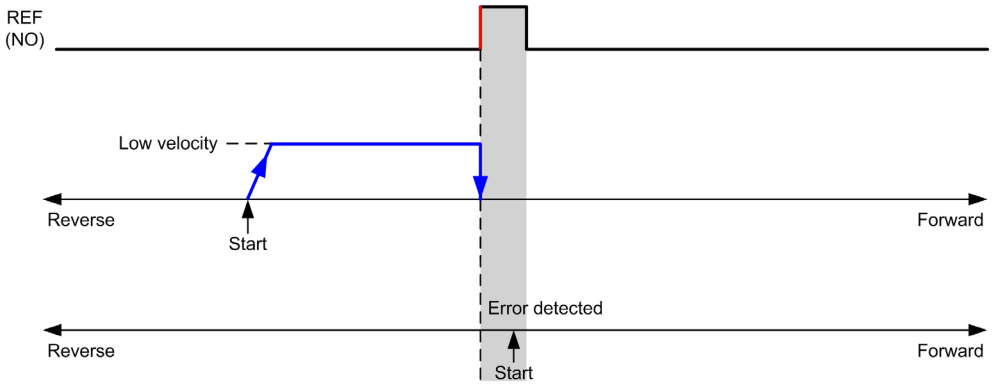


REF (NO) Reference point (Normally Open)

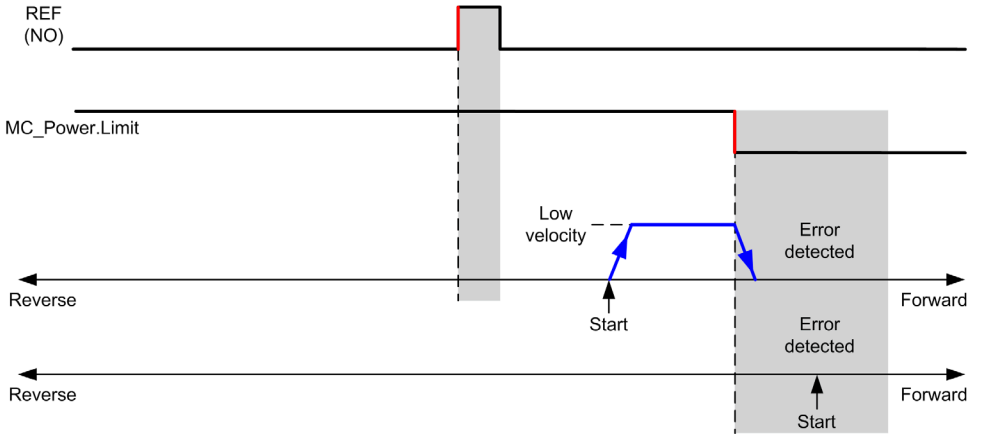
Short Reference No Reversal

Short Reference No Reversal: Positive Direction

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



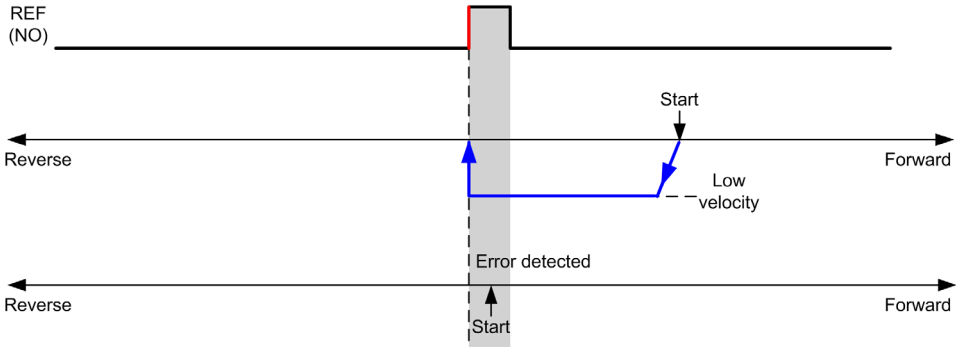
REF (NO) Reference point (Normally Open)



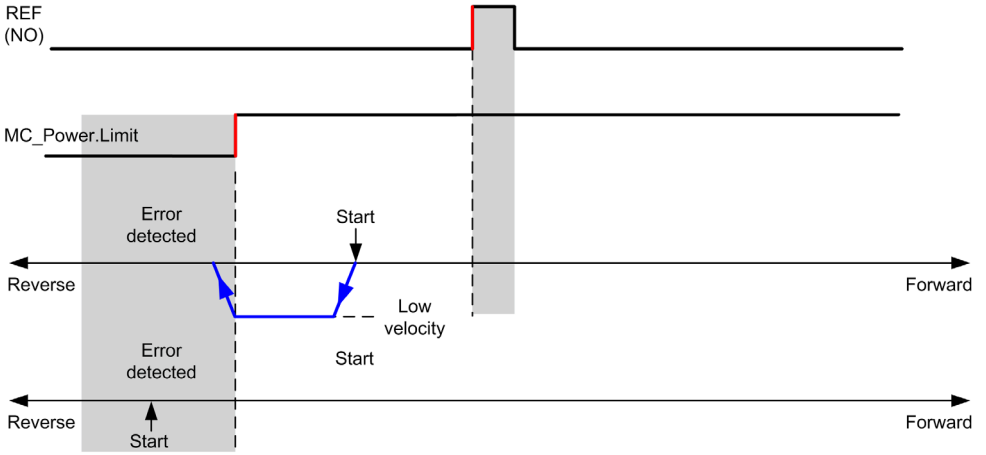
REF (NO) Reference point (Normally Open)

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)



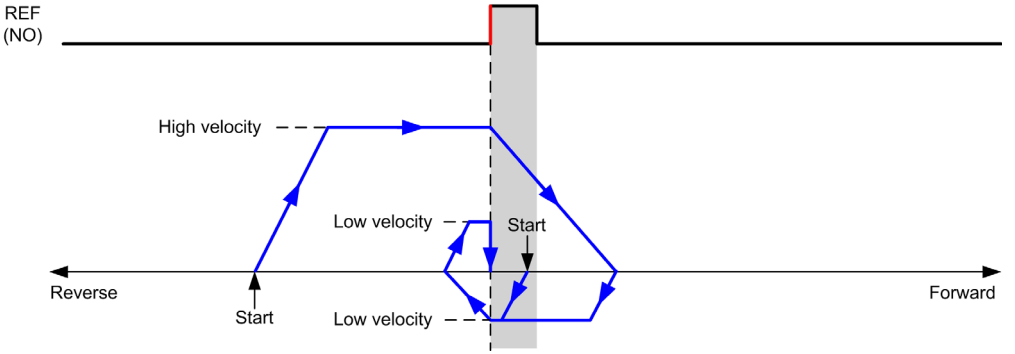
REF (NO) Reference point (Normally Open)

Short Reference Reversal

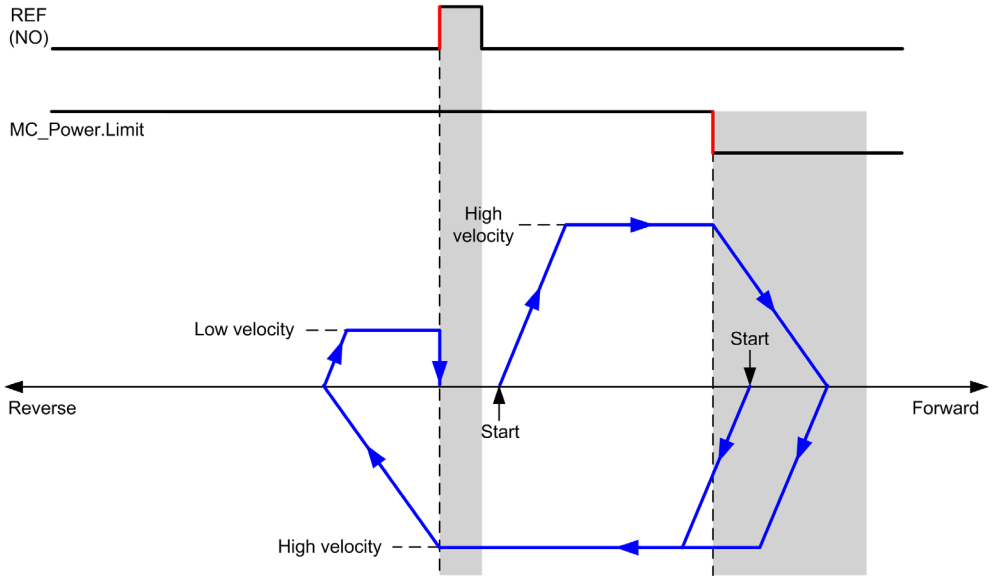
Short Reference Reversal: Positive Direction

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)

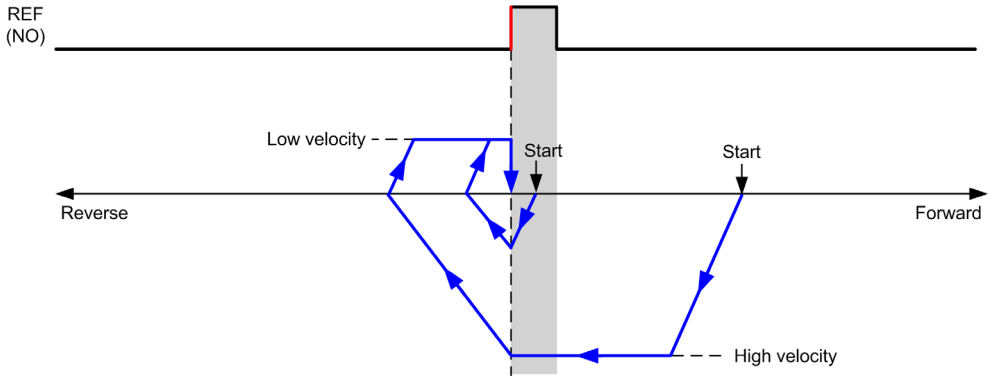


REF (NO) Reference point (Normally Open)

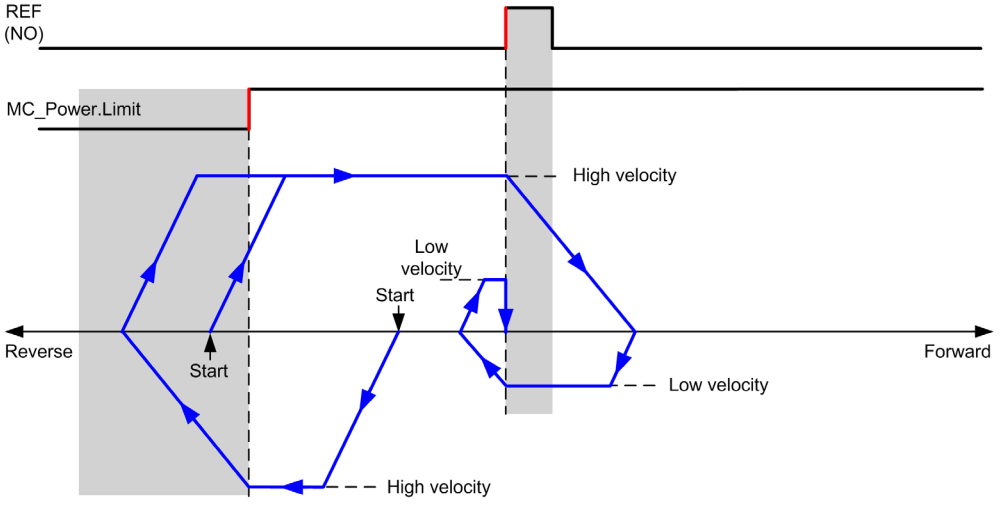
Short Reference Reversal: Negative Direction

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)

Short Reference with INDEX

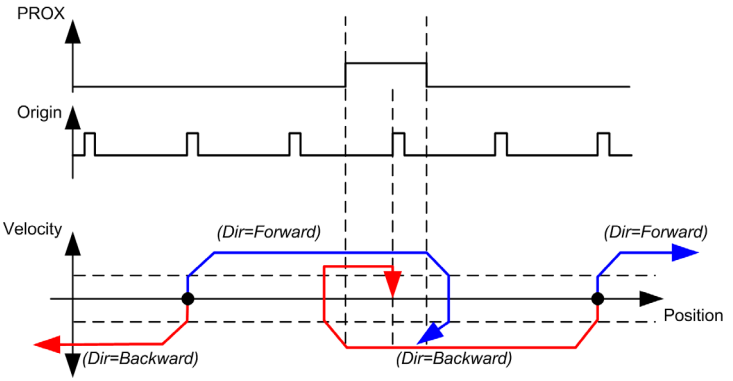
Description

The `PTOHome` and `PTOEnhancedHome` function blocks are used to set the axis to a reference position.

The **Short Reference with INDEX** homing method uses the two homing-specific inputs:

- The **PROX** input, used as the positive limit signal: On the rising edge of the signal (negative side), the axis must change direction.
- The **Origin** input, used as the zero marker signal (Z)

The following diagram illustrates the **Short Reference with INDEX** homing method:



Error Detection

When the **PROX** input is enabled, an error is reported in the `LIMIT_FLT` status object when the limit is crossed.

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` *(see page 282)* command is executed first, and then the specified number of pulses is output at the home low velocity in the specified direction.

NOTE: The `MC_Home_PTO` command busy flag is only released after origin offset has been completed.

Section 13.4
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 (`Done` or `InVel` 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 `FB1` at end-position of `FB1`).
`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 (`Done` or `InVel` output is set to TRUE) and the time delay has elapsed. There is no blending. The `Delay` parameter is set using `MC_WritePar_PTO` *(see page 309)*, with `ParameterNumber` 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.
ShortReference_with_INDEX	22	Short reference with INDEX.

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.
EnableLimitNeg (**Enable the Software Position Limits**)	5	R/W	Enable negative software limit switch.
MaxVelocityAppl (**Max. Velocity**)	9	R/W	Maximal allowed velocity of the axis in the application.
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.
MaxDecelerationAppl (**Max. dec.**)	15	R/W	Maximal allowed deceleration of the axis in the application.
Reserved	to 999	-	Reserved for the PLCopen standard.
Delay	1000	R/W	Time in ms (0...65,535) Default value: 0

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 limP exceeded.
HwPositionLimitN	1003	Hardware negative position limit limN exceeded.
SwPositionLimitP	1004	Software positive position limit exceeded.
SwPositionLimitN	1005	Software negative position limit exceeded.
ApplicationStopped	1006	Application execution has been stoppped (power cycle, 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 system bits *(see page 358)* and system words *(see page 366)*.
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 *(see page 248)*.
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 *(see page 282)*).
MotionQueueFull	2008	The move could not be buffered because the motion queue is full.
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.

Name	Value	Description
`InvalidParameterValue`	3009	Parameter value is out of range.
`ReadOnlyParameter`	3010	Parameter is read-only.

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 13.5
Operation Modes

Overview

This section describes the operation modes.

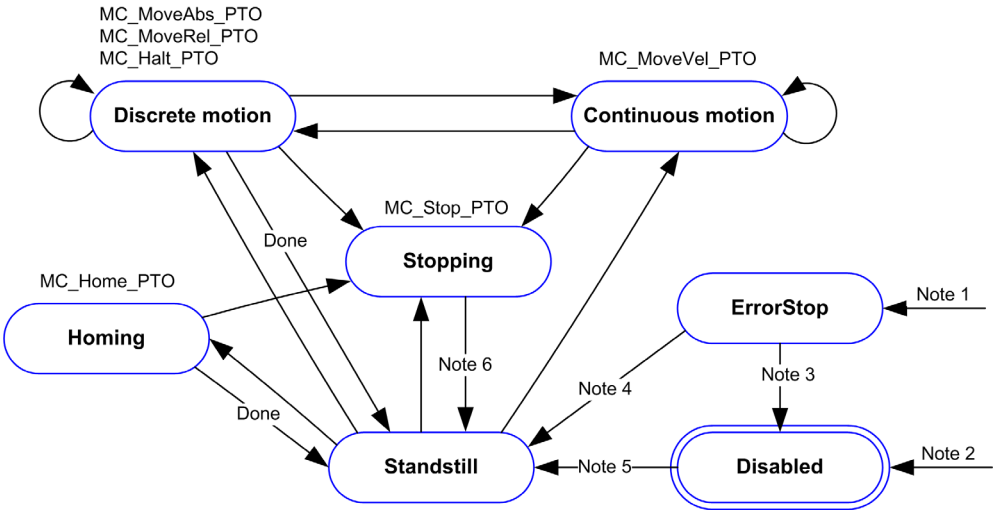
What Is in This Section?

This section contains the following topics:

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**. `Error` output is set to TRUE on applicable function blocks, and an `ErrorId` sets the error code. As long as an error is pending, the state remains `ErrorStop`. No further motion command is accepted until a reset has been done using `MC_Reset_PTO`.
Homing	Applicable when `MC_Home_PTO` controls the axis.
Discrete	Applicable when `MC_MoveRel_PTO`, `MC_MoveAbs_PTO`, or `MC_Halt_PTO` 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

(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.

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

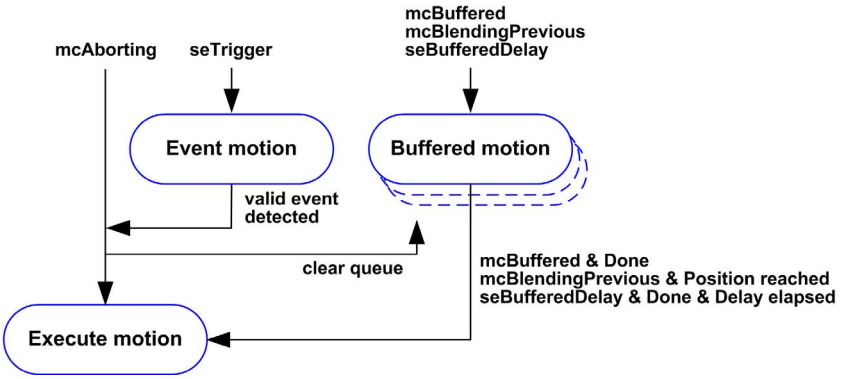
Some of the motion function blocks have an input object called `BufferMode`. With this input object, the function block can either start immediately, start on probe event, or be buffered.

The available options are defined in the buffer modes function block object codes *(see page 242)*:

- 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 223)*.
- 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` *(see page 245)*.
- When a transition to **ErrorStop** state is detected: `Error` output is set to TRUE on buffered function blocks, with `ErrorId=ErrorStopActive` *(see page 245)*.

NOTE:

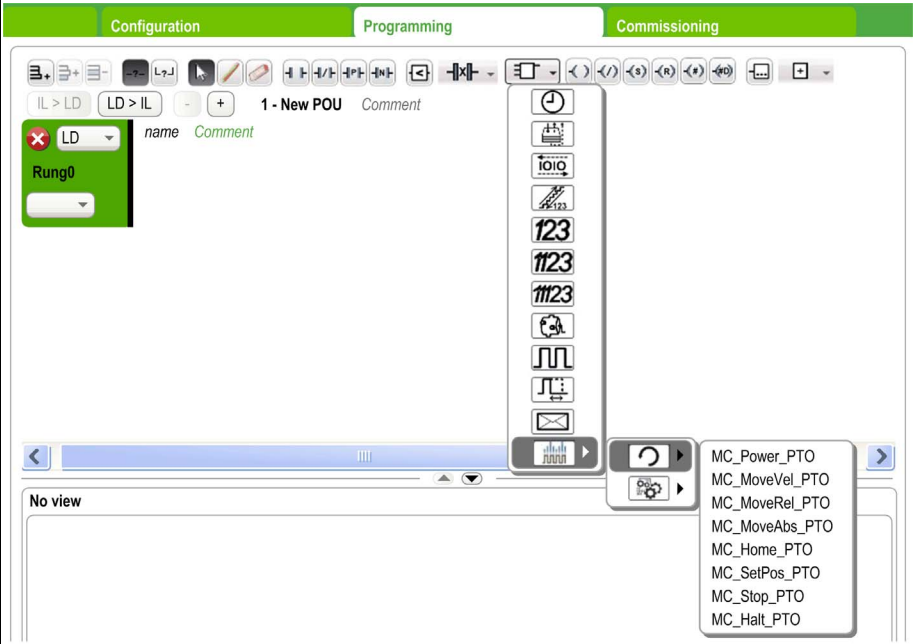
- Only a valid motion can be queued. If the function block execution terminates with the `Error` output set to TRUE, the move is not queued, any move currently executing is not affected, and the queue is not cleared.
- When the queue is already full, the `Error` output is set to TRUE on the applicable function block, and `ErrorId` output returns the error `MotionQueueFull` *(see page 245)*.

Section 13.6
Adding / Removing a Function Block

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	Select **Function Blocks** → **PTO** → **Motion** or **Function Blocks** → **PTO** → **Administrative** as shown in the following graphic: 
3	Click into the rung to place the selected function block.
4	Associate the input/output variables *(see page 213)* of the function block.

NOTE: Set the parameters in the **Configuration** tab.
For more details, refer to PTO Configuration *(see page 74)*.

Removing a Function Block

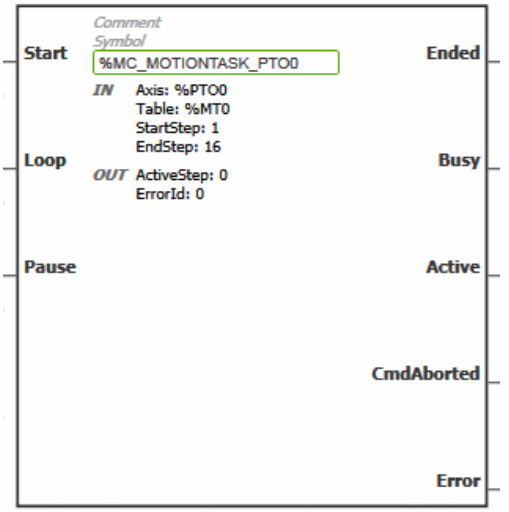
Follow these steps to remove an instance of a PTO function block:

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

Section 13.7
Motion Task Function Block

MC_MotionTask_PTO Function Block

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	On rising edge, starts the function block execution. The `Loop` and `Pause` inputs can be changed during the function block execution and they affect the ongoing execution. The `Axis`, `Table`, `StartStep`, and `EndStep` input objects value 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, the outputs are reset one cycle after the function block execution is terminated.
`Loop`	FALSE	When TRUE, once the function block execution terminates with no detected error, the motion task sequence starts again on first step. The `Ended` output is set for one cycle. The input is tested when the function block execution terminates with no detected error.
`Pause`	FALSE	When TRUE: ● Forces the axis to the **Stopping** state. To reach the **Stopping** state, the current step deceleration value is used, if this value is invalid, the fast stop deceleration is used. ● Does not allow a new execution of the function block. ● Keeps the `Active` output set even if velocity is equal to 0. When reset to FALSE after being set to TRUE, the current motion task execution resumes in the following conditions: ● The motion task resumes whatever the value of the current velocity. ● The current active step parameters are used. ● The absolute target position is not changed. If the current motion task is `MC_MoveRel_PTO`, there is no distance added. ● In the current step, the **Next step** condition is reset (for example: the delay is restarted from 0, `Probe input event` is enabled and waiting for an 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 funtion block is to be executed. The parameter is set in the function block instance reached in the **Programming** → **Tools** module tab (**PTO objects** → **Motion**, **Axis** parameter).

Input Object	Type	Initial Value	Description
`Table`	%MT	–	Table instance for which the funtion block is to be executed. The parameter is set in the function block instance reached in the **Programming** → **Tools** module tab (**PTO objects** → **Motion**, **Table** parameter).
`StartStep`	Byte	1	Step number that defines the first step executed in the motion sequence. 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 sequence. 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 current last step of the table is used.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
`Ended`	0	When TRUE, function block execution is finished with no error detected. `Ended` output behavior: ● If the last step of the motion sequence is a **discrete** movement, the output behaves like a `Done` output. Outputs behavior: (`Busy`, `Active`, `CmdAborted`, `Error`) are FALSE (0). ● If the last step of the motion sequence is a **continuous** movement (move velocity), the output behaves like an `InVel` output. Outputs behavior: ● `Busy` and `Active` are TRUE (1). ● `CmdAborted` and `Error` are FALSE (0). If a loop is requested (`Loop` input), the `Ended` output is TRUE for one cycle when the sequence starts again.
`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.

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 objects of the function block:

Output Objects	Type	Initial Value	Description
ActiveStep	Byte	0	Number of the step currently executed in the Motion Task Table.
ErrorId	Word	-	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

Operating Modes

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

The execution of the steps in the motion task follows the same rules and restrictions as each single-axis function block. Generally, in case of detected errors the function block behaves as follows:
- 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 current 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 **Stopping** state. The function block execution is finished with Error = 1, and the ErrorId output object set to the value corresponding to the detected error type.

Section 13.8
Power Function Block

MC_Power_PTO Function Block

Behavior

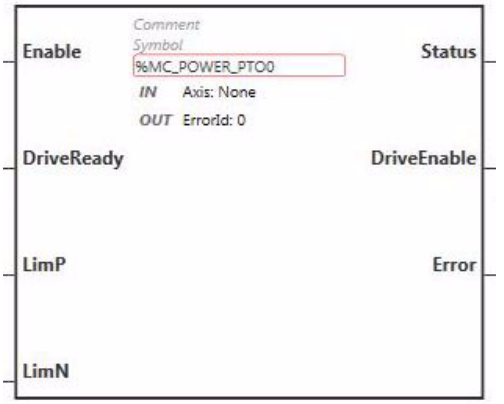
The axis is disabled, when:

- %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 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.

Outputs

This table describes the outputs of the function block:

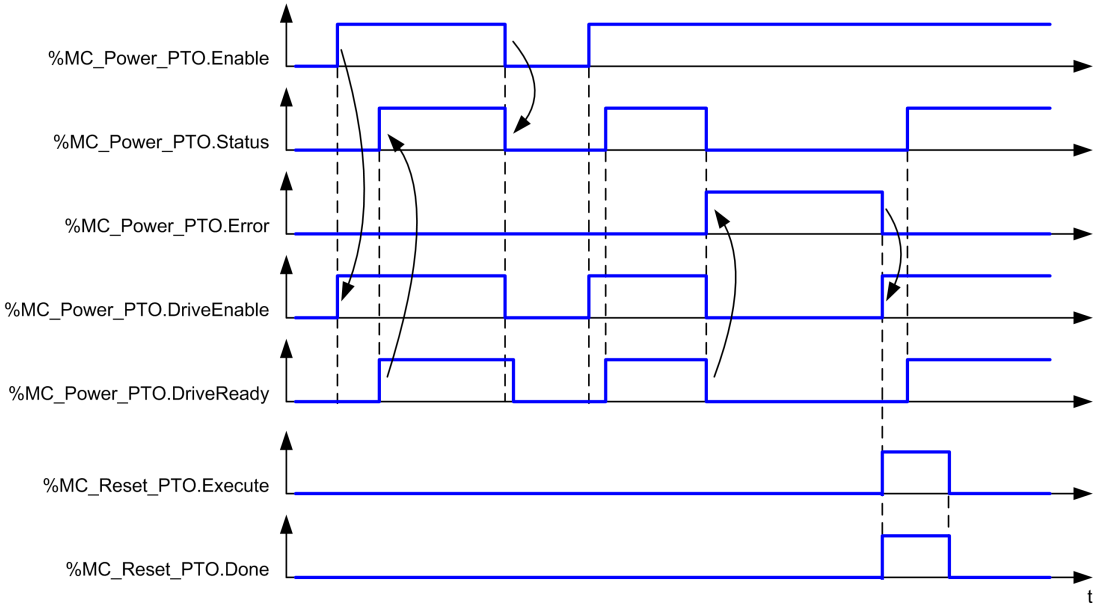
Output	Initial Value	Description
`Status`	FALSE	When TRUE, the drive is reported as ready to accept motion commands.
`DriveEnable`	FALSE	When TRUE, indicates to the drive that it can accept motion commands and that it should, therefore, enable power. If the drive input is connected to the controller, 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 objects of the function block:

Output Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

Timing Diagram Example

The diagram illustrates the operation of the `MC_Power_PTO` function block:



Section 13.9
Movement Function Blocks

Overview

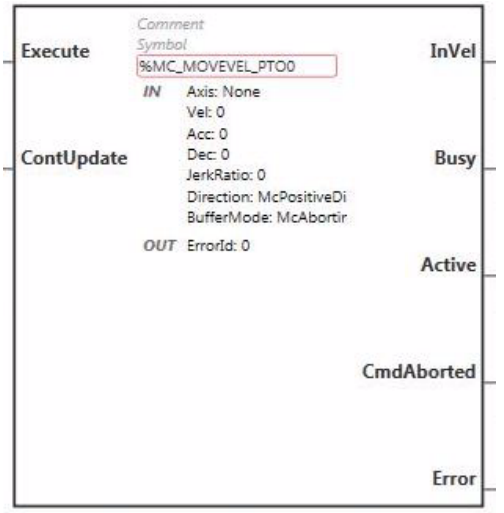
This section describes the movement function blocks.

What Is in This Section?

This section contains the following topics:

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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.
`ContUpdate`	FALSE	When TRUE, makes the function block use any modified values of the input objects (`Vel`, `Acc`, `Dec`, and `Direction`), and apply it to the ongoing command. This input must be TRUE prior to the rising edge on the `Execute` input to be taken into account.

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.
Vel	DINT	0	Target velocity. Range Hz: 0....MaxVelocityAppl *(see page 243)*
Acc	DINT	0	Acceleration in Hz/ms Range (Hz/ms): 1...MaxAccelerationAppl *(see page 243)*
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl *(see page 243)*
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile *(see page 222)*. Range : 0…100
Direction	INT	mcPositiveDirection	Direction of the movement for PTO type CW/CCW forward (CW) = 1 (mcPositiveDirection) reverse (CCW) = -1 (mcNegativeDirection)
BufferMode	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table *(see page 241)*.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
InVel	FALSE	When TRUE, the target velocity has been reached.
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 objects of the function block:

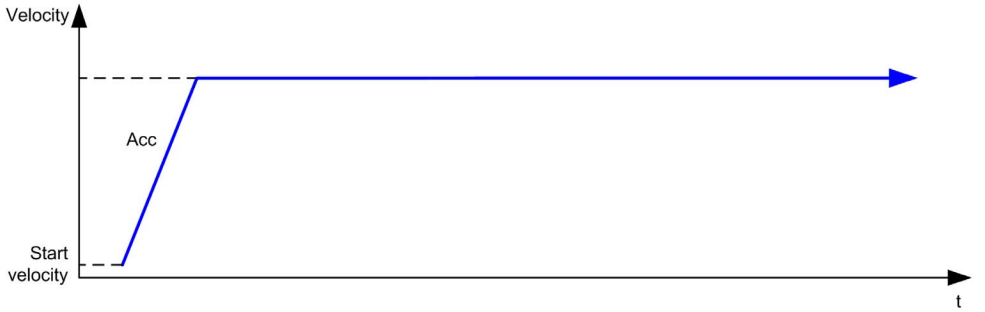
Output Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

NOTE:

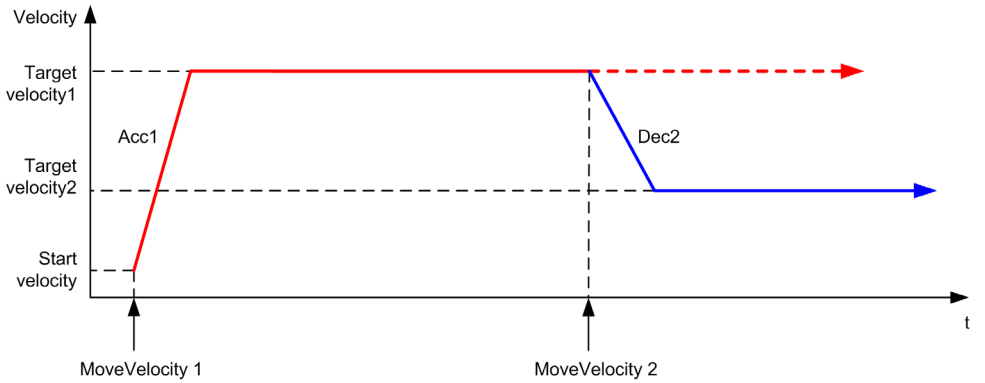
- To stop the motion, the function block has to be interrupted by another function block issuing a new command.
- If a motion is ongoing, and the direction is reversed, first the motion is halted with the deceleration of the `MC_MoveVel_PTO` function block, and then the motion resumes backward.
- The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

Timing Diagram Example

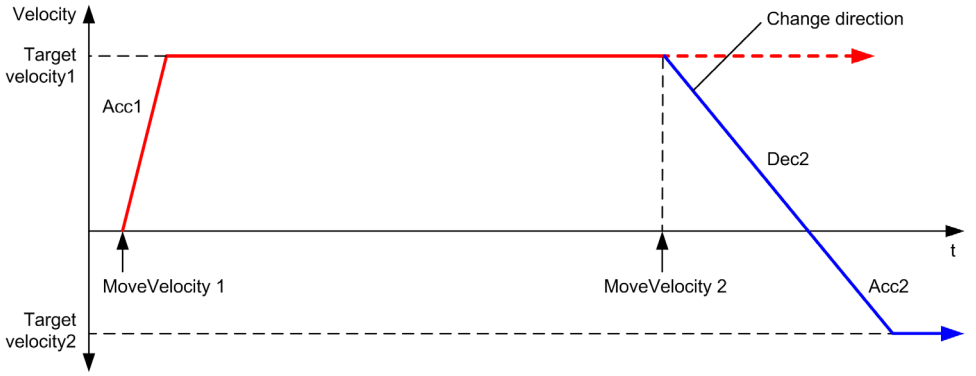
The diagram illustrates a simple profile from **Standstill** state:



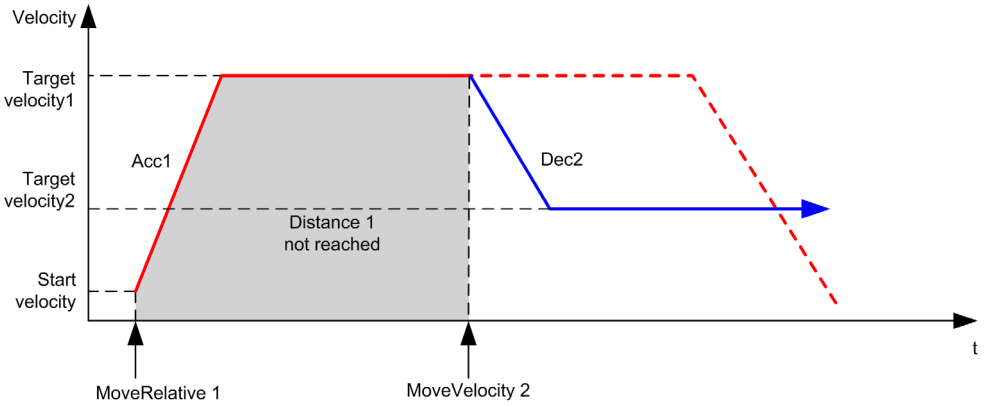
The diagram illustrates a complex profile from **Continuous** state:



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

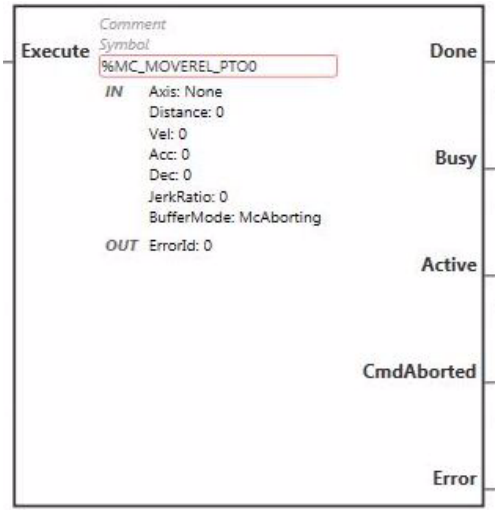


The diagram illustrates a complex profile from **Discrete** state:



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 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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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.
Vel	DINT	0	Target velocity. Range Hz: 0....MaxVelocityAppl *(see page 243)*
Acc	DINT	0	Acceleration in Hz/ms Range (Hz/ms): 1...MaxAccelerationAppl *(see page 243)*
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl *(see page 243)*
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile *(see page 222)*. Range : 0…100
BufferMode	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table *(see page 241)*.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	-	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 objects of the function block:

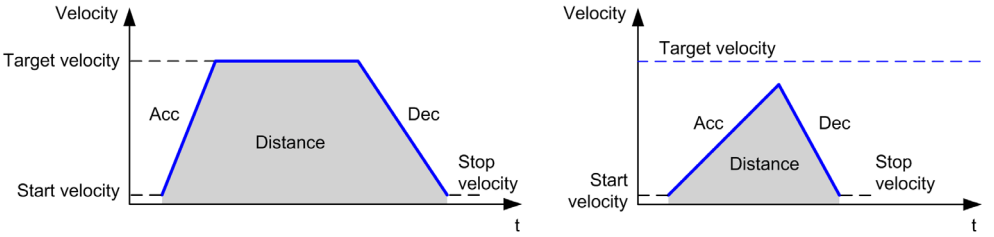
Output Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

NOTE:

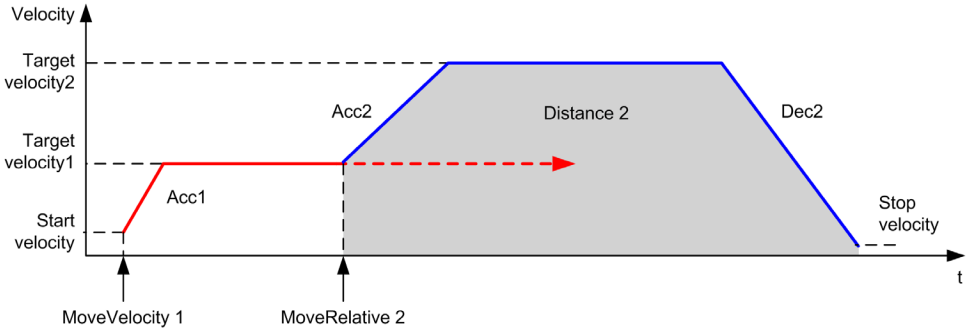
- The function block completes with velocity zero if no further blocks are pending.
- If the distance is too short for the target velocity to be reached, the movement profile is triangular, rather than trapezoidal.
- If a motion is ongoing, and the commanded distance is exceeded due to the current motion 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

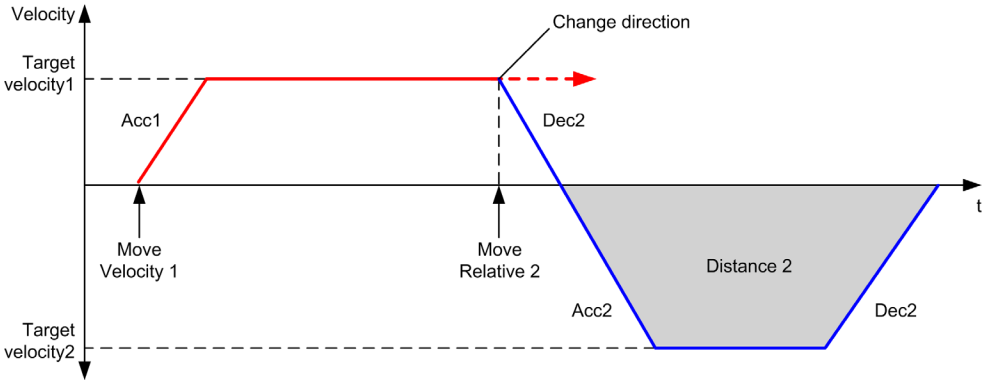
The diagram illustrates a simple profile from **Standstill** state:



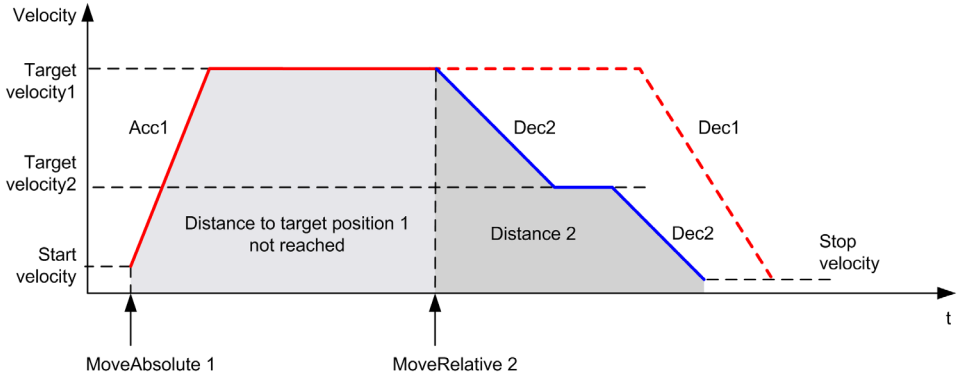
The diagram illustrates a complex profile from **Continuous** state:



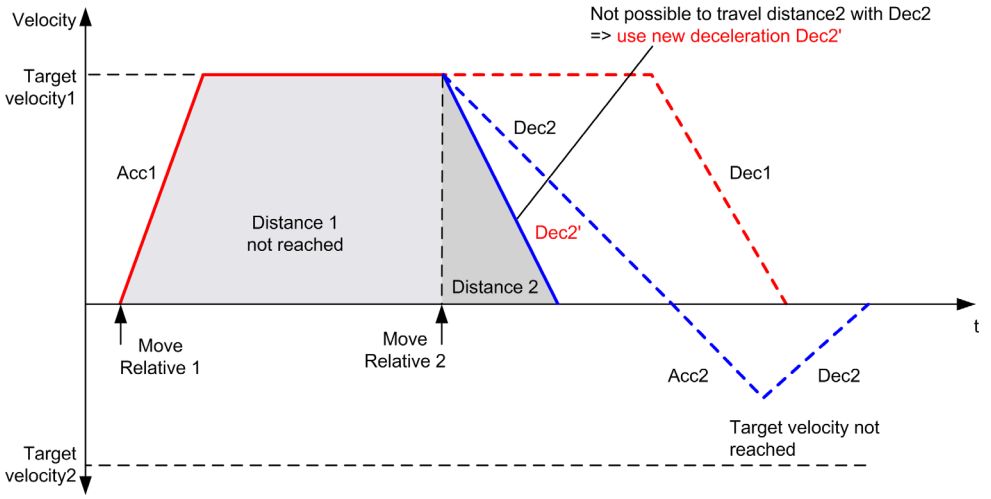
The diagram illustrates a complex profile from **Continuous** state with change of direction:



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

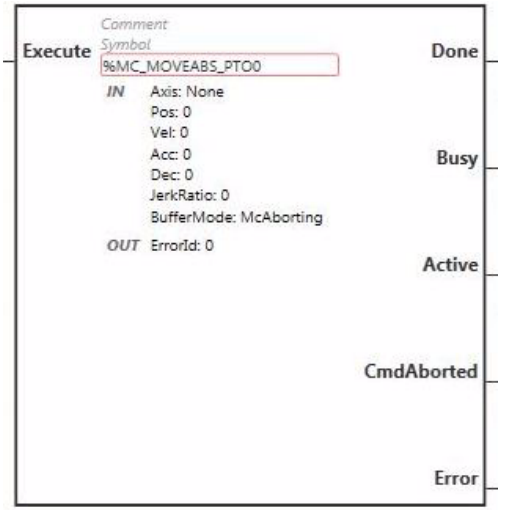


The diagram illustrates a complex profile from **Discrete** state with change of direction:



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 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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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 *(see page 243)*
Acc	DINT	0	Acceleration in Hz/ms Range (Hz/ms): 1...MaxAccelerationAppl *(see page 243)*
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl *(see page 243)*
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile *(see page 222)*. Range : 0…100
BufferMode	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table *(see page 241)*.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	-	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 objects of the function block:

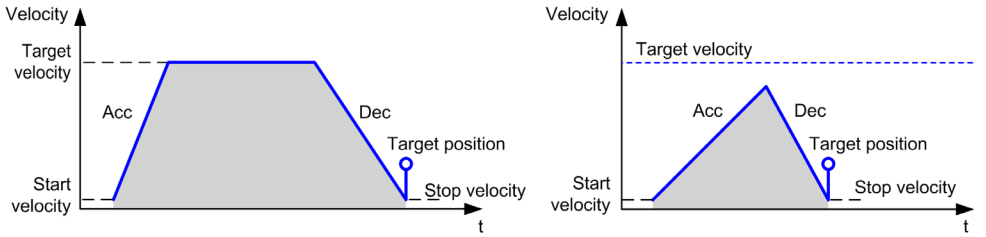
Output Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

NOTE:

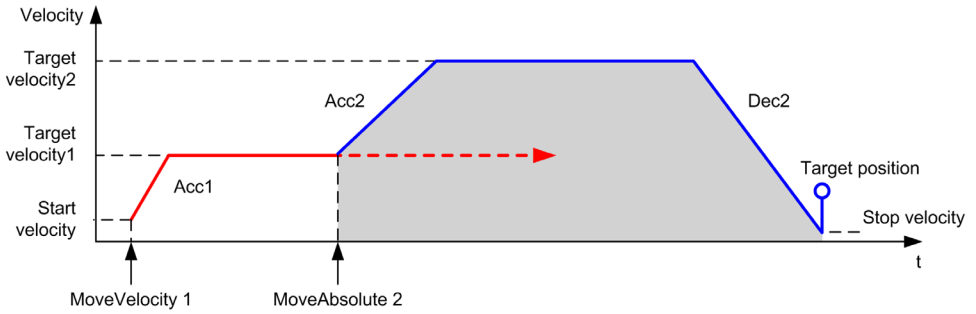
- The function block completes with velocity zero if no further blocks are pending.
- The motion direction is automatically set, according to the current and target positions.
- If the distance is too short for the target velocity to be reached, the movement profile is triangular, rather than trapezoidal.
- 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

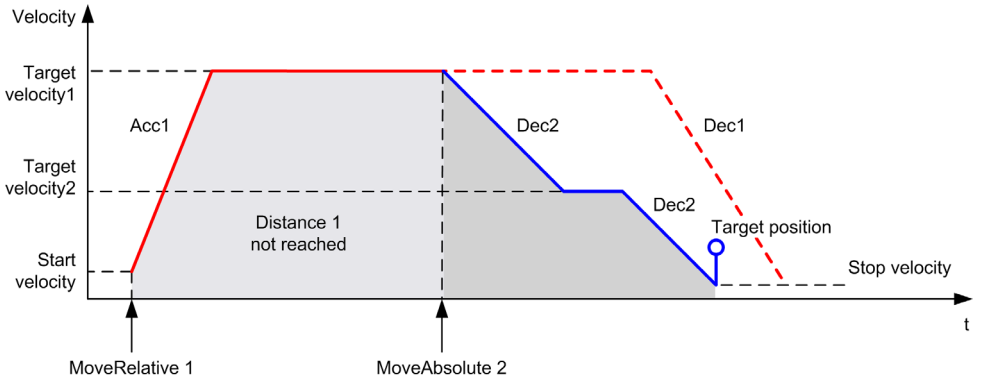
The diagram illustrates a simple profile from **Standstill** state:



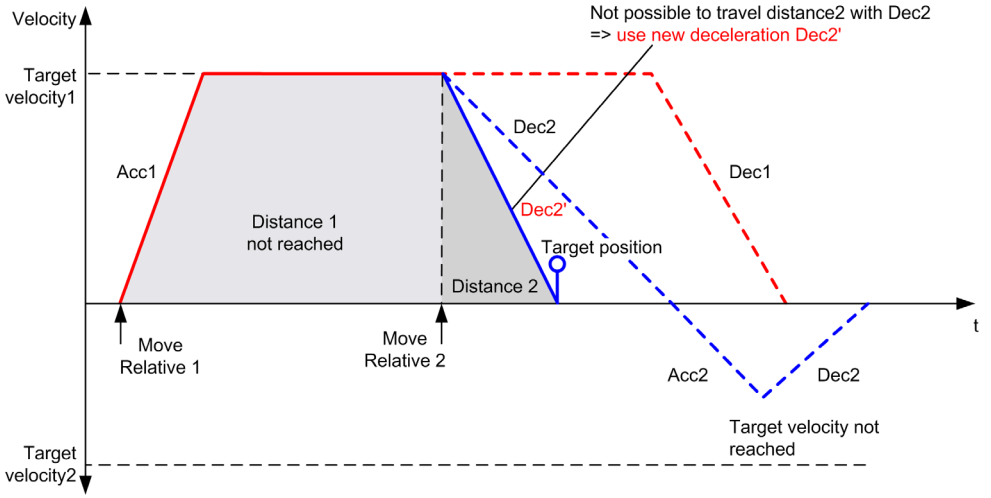
The diagram illustrates a complex profile from **Continuous** state:



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

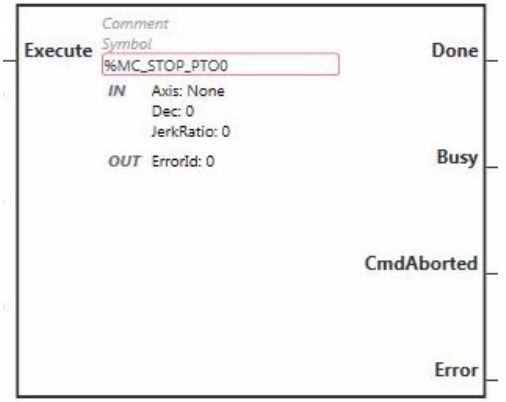


The diagram illustrates a complex profile from **Discrete** state with change of direction:



MC_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 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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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 243)*

Input Object	Type	Initial Value	Description
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile *(see page 222)*. Range : 0…100

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	-	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.
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:

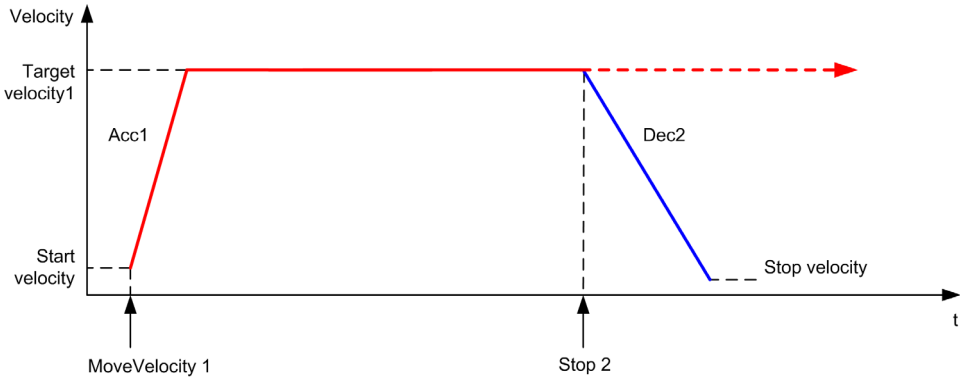
Output Objects	Type	Initial Value	Description
ErrorId	Word	-	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

NOTE:

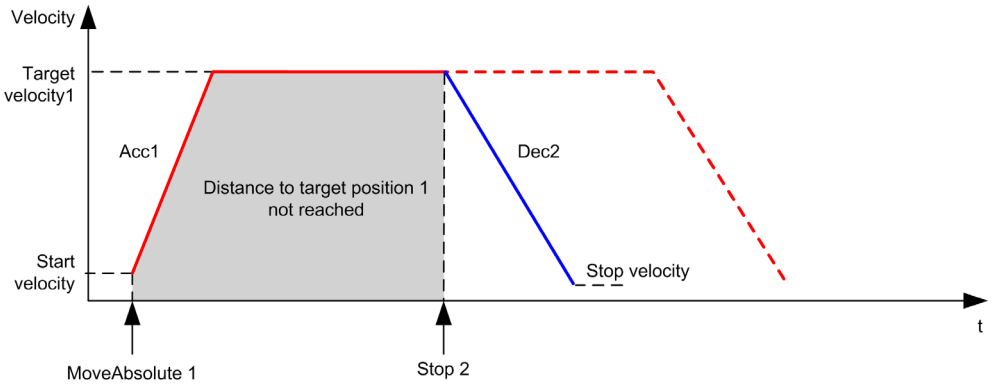
- Calling this function block in state **Standstill** changes the state to **Stopping**, and back to **Standstill** when Execute is FALSE.
- The state **Stopping** is kept as long as the input Execute is TRUE.
- The Done output is set when the stop ramp is finished.
- If Deceleration = 0, the fast stop 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:

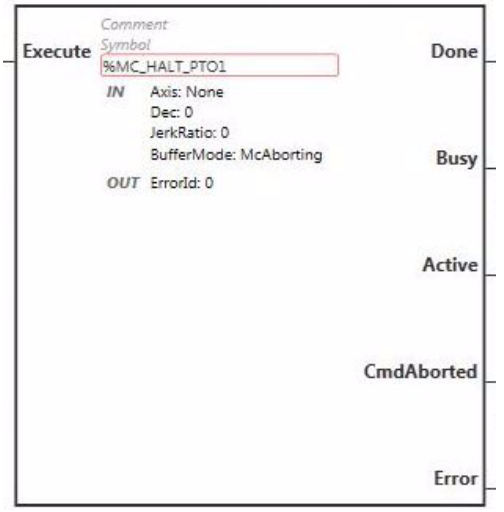


The diagram illustrates a simple profile from **Discrete** state:



MC_Halt_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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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 243)*
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile *(see page 222)*. Range : 0…100
BufferMode	INT	mcAborting	Transition mode from ongoing move. Refer to Buffer Modes table *(see page 241)*.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	-	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.

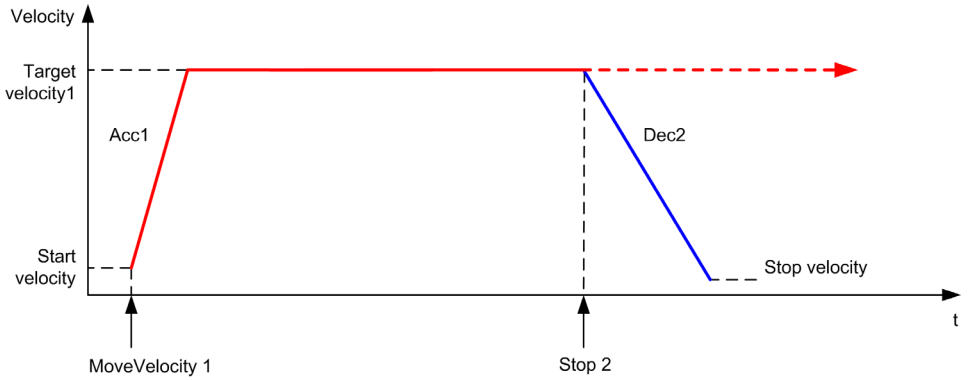
This table describes the output objects of the function block:

Output Objects	Type	Initial Value	Description
ErrorId	Word	-	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

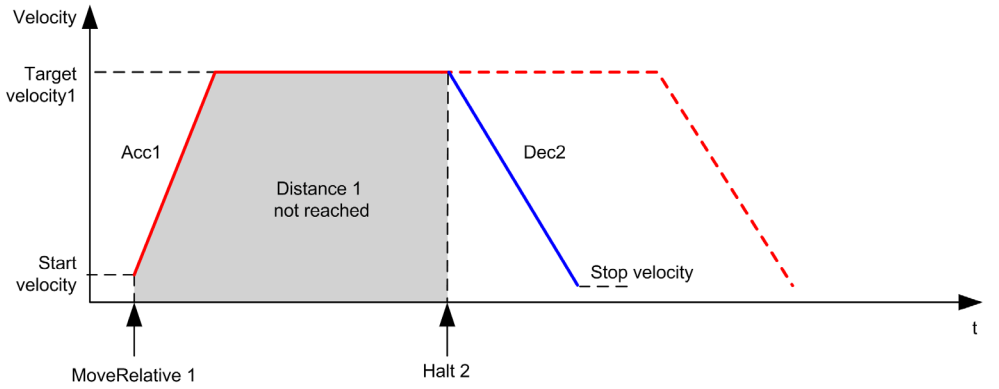
NOTE: The function block completes with velocity zero.

Timing Diagram Example

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



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



Section 13.10
Stopping / Position Function Blocks

Overview

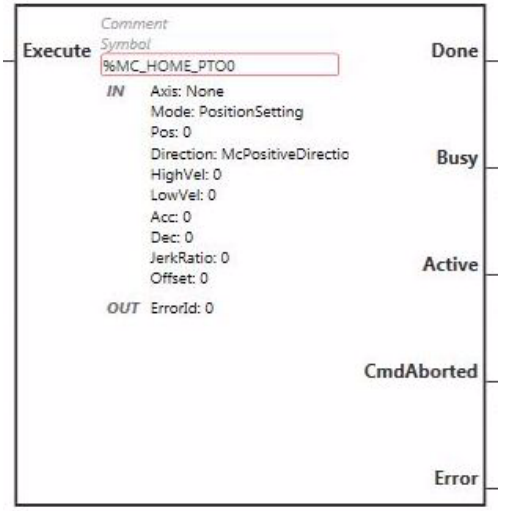
This section describes the stopping and the position function blocks.

What Is in This Section?

This section contains the following topics:

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 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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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 *(see page 243)*.
Pos	DINT	0	Position of the axis.
HighVel	DINT	0	Target homing velocity for searching the limit or reference switch. Range Hz: 1...MaxVelocityAppl *(see page 243)*
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 *(see page 243)*
Dec	DINT	0	Deceleration in Hz/ms Range (Hz/ms): 1...MaxDecelerationAppl *(see page 243)*
JerkRatio	INT	0	Percentage of acceleration / deceleration adjustment used to create the S-curve profile *(see page 222)*. Range : 0…100
Direction	INT	mcPositiveDirection	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 *(see page 241)*). Range: -2,147,483,648...2,147,483,647

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
Done	-	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.

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 objects of the function block:

Output Objects	Type	Initial Value	Description
ErrorId	Word	-	Motion command error codes, valid when Error output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

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

Timing Diagram Example

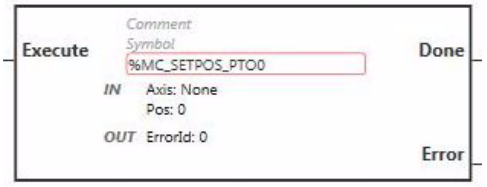
Home modes *(see page 230)*

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 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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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`	-	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.
`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 Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

Section 13.11
Status Function Blocks

Overview

This section describes the status function blocks.

What Is in This Section?

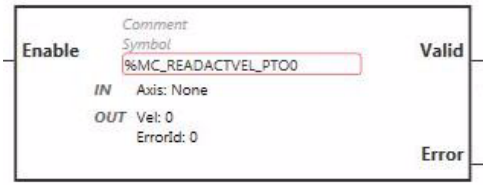
This section contains the following topics:

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 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.

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.

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 Objects	Type	Initial Value	Description
`Vel`	DINT	-	Velocity of the axis.
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

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 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.

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.

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:

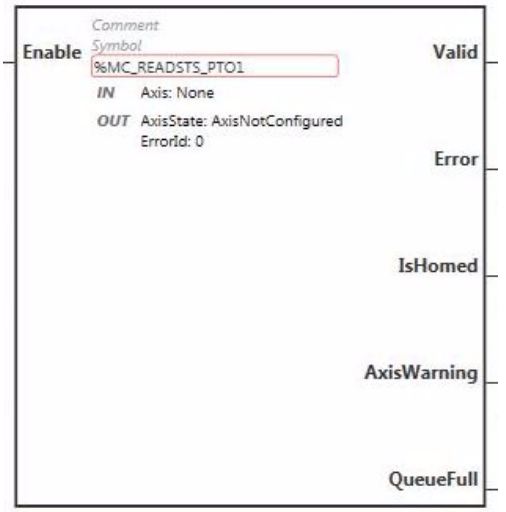
Output Objects	Type	Initial Value	Description
`Pos`	DINT	-	Position of the axis.
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

MC_ReadSts_PTO Function Block

Function Description

This function block returns the state diagram *(see page 248)* 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 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.

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.

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 `MC_ReadAxisError_PTO` function block to obtain detailed information. *(see page 302)*
`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:

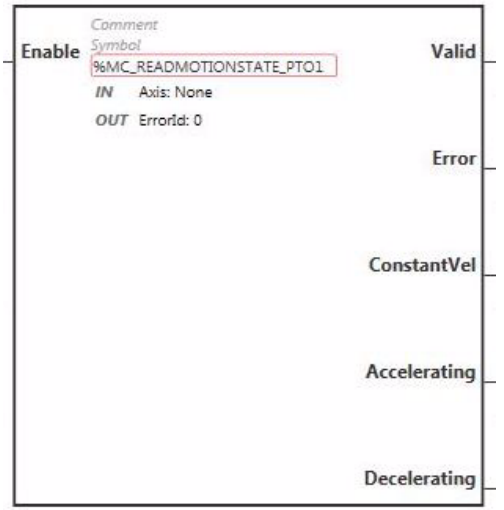
Output Objects	Type	Initial Value	Description
`AxisState`	-	-	Code for the state of the axis: 0 = axis not configured 1 = ErrorStop 2 = Disabled 4 = Stopping 8 = Homing 16 = Standstill 32 = Discrete motion 64 = Continuous motion For more information, refer to the States description table *(see page 248)*.
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

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 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.

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.

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 objects of the function block:

Output Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

Section 13.12
Probe Function Blocks

Overview

This section describes the probe function blocks.

What Is in This Section?

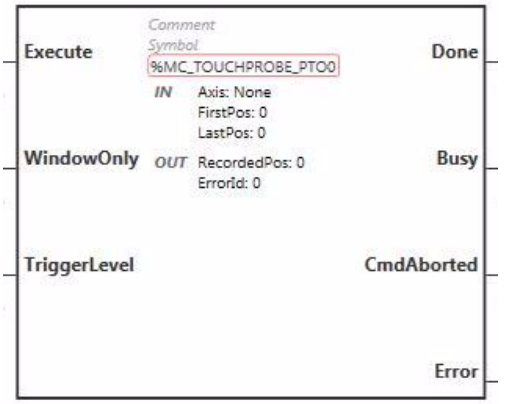
This section contains the following topics:

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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.
`WindowOnly`	FALSE	When TRUE, a trigger event is only recognized within the position range (window) defined by `FirstPosition` and `LastPosition`.
`TriggerLevel`	FALSE	When TRUE, position captured at rising edge. When FALSE, position captured 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`	-	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.
`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 Objects	Type	Initial Value	Description
`RecordedPos`	-	-	Position where trigger event was detected.
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

NOTE:

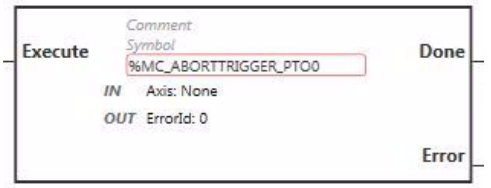
- Only one instance of this function block is allowed on the same axis.
- Only the first event after the rising edge at the `MC_TouchProbe_PTO` function block `Busy` output is valid. Once the `Done` output is set to TRUE, subsequent events are ignored. The function block needs to be reactivated to respond 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 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 Execute. A subsequent change in these input parameters does not affect the ongoing execution unless the ContUpdate 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.

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.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
`Done`	-	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.
`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 Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

Section 13.13
Error Handling Function Blocks

Overview

This section describes the error handling function blocks.

What Is in This Section?

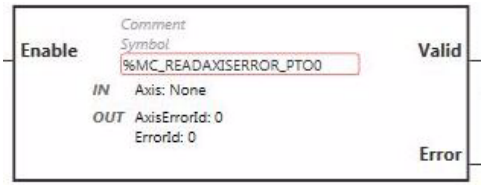
This section contains the following topics:

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 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.

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.

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:

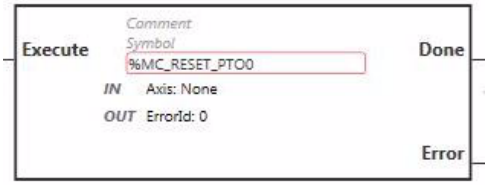
Output Objects	Type	Initial Value	Description
`AxisErrorId`	-	-	Axis error codes, valid when `AxisWarning` output is TRUE. Refer to PTO axis error code table *(see page 244)*.
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

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 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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
`Done`	-	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.
`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 Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

Section 13.14
Parameters Function Blocks

Overview

This section describes the parameters function blocks.

What Is in This Section?

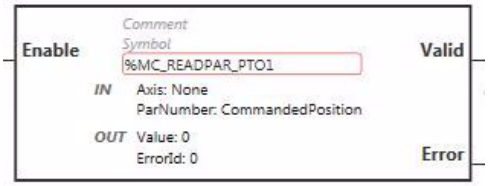
This section contains the following topics:

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 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.

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 243)*.

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:

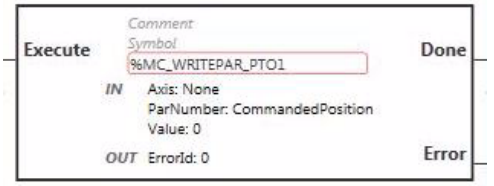
Output Objects	Type	Initial Value	Description
`Value`	-	-	Value read from the parameter chosen with the `ParNumber` input object.
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

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 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 `Execute`. A subsequent change in these input parameters does not affect the ongoing execution unless the `ContUpdate` 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.

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 243)*.
`Value`	DINT	0	Value to be written to the parameter chosen with the `ParNumber` input object.

Outputs

This table describes the outputs of the function block:

Output	Initial Value	Description
`Done`	-	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.
`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 Objects	Type	Initial Value	Description
`ErrorId`	Word	-	Motion command error codes, valid when `Error` output is TRUE. Refer to PTO motion command error code table *(see page 245)*.

Chapter 14
PID Function

What Is in This Chapter?

This chapter contains the following sections:

Section 14.1
PID Operating Modes

PID Operating Modes

Introduction

The SoMachine Basic PID controller offers 4 distinct operating modes, configurable in the **General** tab *(see page 333)* 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 338)* 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 330)*. 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 340)* 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 338)* and the type of PID action *(see page 342)*. 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 349)*:

- 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 338)* and the type of PID action *(see page 342)*. 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 342)* (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 348)* drop-down. 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 14.2
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 *(see page 314)*
2	Pre-requisites for PID configuration *(see page 314)*
3	Configuring the PID *(see page 315)*
4	Control set-up *(see page 316)*

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.

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 *(see page 345)*.
2	**Scan Mode** is set to periodic *(see page 347)*.

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

INOPERABLE EQUIPMENT OR UNINTENDED EQUIPMENT OPERATION

- 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 *(see page 333)* 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 *(see page 312)*.
2	Activate the **PID States** checkbox and enter the address of the memory word in the field.
3	In the **Input** tab *(see page 336)*, enter the address of the analog input used as a measurement.
4	If **Conversion** or **Alarms** are required, refer to **Input** tab *(see page 336)* of **PID Assistant** screen.
5	In the **PID** tab *(see page 338)*, 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 *(see page 312)*, 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 *(see page 327)*) 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 *(see page 340)*.
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 *(see page 340)*.

Step	Action
11	In the **Output** tab *(see page 342)*, 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 *(see page 342)*. In **Analog output** field, set the address of the word: an analog output or a memory word. Set the **Output PWM** *(see page 342)* 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 *(see page 342)*.
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

INOPERABLE EQUIPMENT OR UNINTENDED EQUIPMENT OPERATION

- 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 312)* perform the following steps:

Step	Action
1	Connect the PC to the controller and transfer the application.
2	Switch the controller to RUN mode.

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

Step	Action
1	Create an animation table containing the objects defined during configuration. Refer to the *SoMachine Basic Operating Guide* for further details about animation table creation.

Step	Action

2	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.
	If the actuator is not controlled:
	● For analog output verify the output voltage or current from analog channel.
	● For PWM output, verify that the:
	  ● LED of the dedicated output is lit
	  ● wiring of the supplies and 0V circuit
	  ● actuator power supply is being applied

3	In the animation table, verify that: ● Output mode is set to automatic. ● All parameters your application requires are set to the appropriate values.
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 *(see page 322)*.
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.

Section 14.3
PID Standard Configuration

What Is in This Section?

This section contains the following topics:

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 312)*. 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 *(see page 319)*
2	Configuring the PID *(see page 319)*
3	Control set-up *(see page 320)*

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.
2	PID is enabled in the program *(see page 345)*.
3	Scan mode is set to periodic *(see page 347)*.

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

INOPERABLE EQUIPMENT OR UNINTENDED EQUIPMENT OPERATION

- 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 *(see page 330)*.

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 336)*, enter the address of the analog input used as a measurement. If **Conversion** or **Alarms** are required, refer to **Input** tab *(see page 336)* of PID Assistant *(see page 330)*.
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 327)*) in the PID tab *(see page 338)*. 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 340)* 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 342)*. 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 342)* in PID Assistant *(see page 330)*.
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 RUN mode.

NOTE: Before switching the logic controller to RUN mode, verify that the operating conditions of the machine allow RUN mode 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 *SoMachine Basic Operating Guide* for details.

2	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.
	If you see that the actuator is not being controlled:
	● For analog output, verify the output voltage or current from analog channel.
	● For PWM output, verify that the:
	  ● 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 *(see page 327)*.
4	If you plan to use the Auto-Tuning *(see page 322)* function, you may need to run Manual Mode *(see page 325)* to know the **Dynamic corrector** and the **AT Trigger** defined in the **AT** tab *(see page 340)* of the **PID Assistant**.
5	Power up the loop controller using the animation table: ● Set the operating mode *(see page 312)*. ● Enable the PID controller *(see page 345)*. ● Set the values defined during configuration *(see page 319)* 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

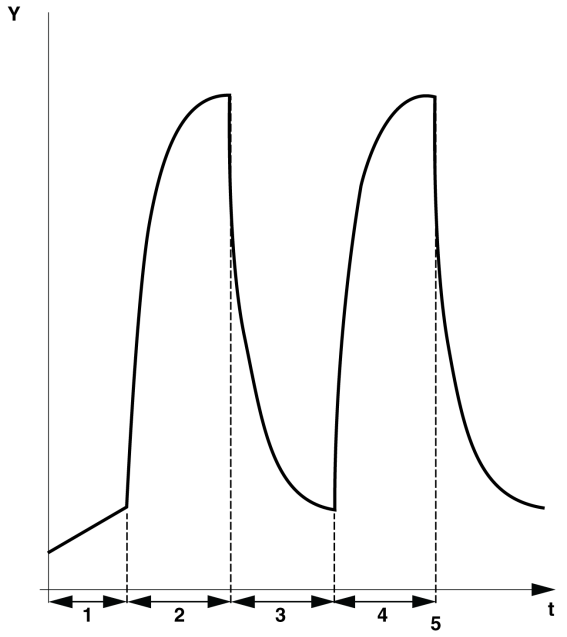
Auto-Tuning Requirements

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.
(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.	

Auto-Tuning Phase	Description
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 324)*.
(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.	

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 338)*). 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 351)*.

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:

- $10\ s < (\tau + \theta) < 2700\ s$ (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 342)*). 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 342)*. This operation can be particularly well suited for testing the system response.

Setting the **bit address** from the **Output** tab *(see page 342)* 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 response curve method *(see page 327)* 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 *(see page 342)*.
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 *(see page 333)* of the **PID Assistant** to 1 (PID mode). For more details on operating modes using word address refer to the operating mode description *(see page 312)*.
4	Set the memory word associated with the manual output in the **Output** tab *(see page 342)* 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 *(see page 319)*.

Stop the Manual Mode

To stop manual mode using an animation table:

Step	Description
1	Disable the loop controller *(see page 319)*.
2	Inhibit the manual mode by setting the dedicated memory bit to 0. For more details refer to the **Output** tab *(see page 342)*.
3	Set the memory word associated with the Operating mode in the **General** tab *(see page 333)* for the `PID` controller to 0. For more details on operating modes using word address, refer to the operating mode description *(see page 312)*.
4	Set the memory word associated to the manual output in the **Output** tab *(see page 342)* 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 338)* 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 *(see page 342)* 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 *SoMachine Basic Operating Guide*.
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: **1.** Calculate the process value output at 63% rise ($S_{[63\%]}$) by using the following formula: $S_{[63\%]} = S_{[initial]} + (S_{[final]}-S_{[initial]})x63\%$ **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: Ts = τ/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:Scan Period = Ts / n , where n is a positive integer(2)

(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.

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.
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 *(see page 314)*.
7	Wait until the Auto-Tuning process ends.
8	Two cases can occur: ● **Auto-Tuning completes successfully:** Continue to Step 10. ● **Auto-Tuning unsuccessful:** Refer to Auto-Tuning detected error codes *(see page 349)*. 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.

(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).
(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.

Step	Action
9	Follow these guidelines to provide a new Ts guess: ● 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 *(see page 338)* of the **PID Assistant** screen, as needed.

(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).
(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.

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 330)*) 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 14.4
PID Assistant

What Is in This Section?

This section contains the following topics:

Access the PID Assistant

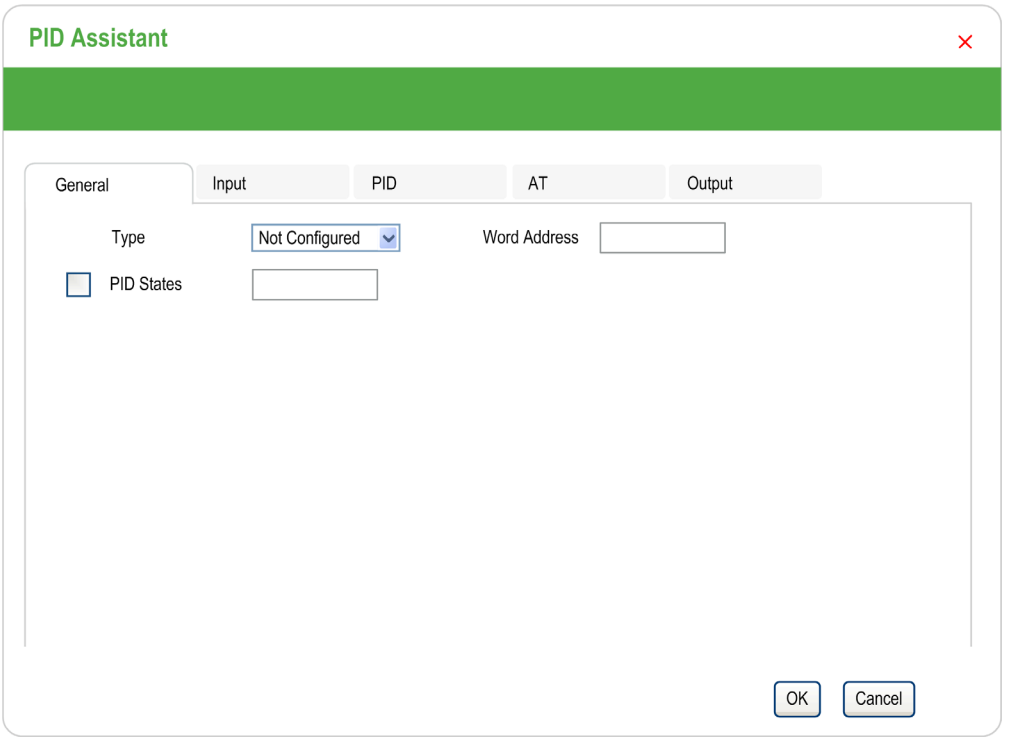
Introduction

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 **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 333)*
Input	Offline	Input tab *(see page 336)*
PID	Offline	PID tab *(see page 338)*

Tab	Access mode	Link
AT	Offline	AT tab *(see page 340)*
Output	Offline	Output tab *(see page 342)*

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

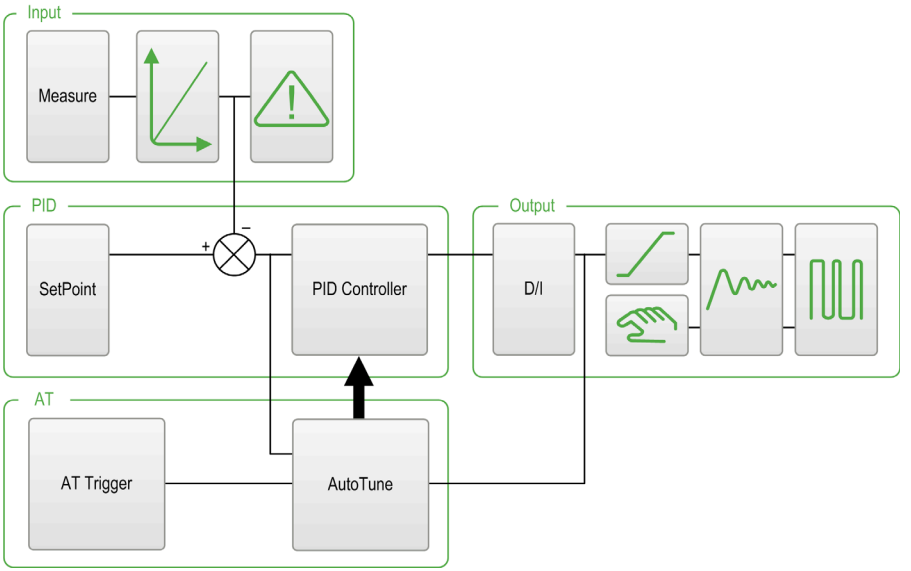
This section describes the **General** tab of the PID. **General** tab is displayed by default when you access the PID Assistant in offline mode.

Description

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

Parameter	Description
Operating Mode	Represents the PID mode to use: ● Not configured ● PID ● AT + PID ● AT ● Word address For further details about operating modes, refer to PID Operating Mode *(see page 312)*.
Word address	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: ● %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	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 *(see page 348)*.

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 *(see page 338)*.
	Click this button to display the **PID** tab *(see page 338)*.
	Click this button to display the **Output** tab *(see page 342)*.
	Click this button to display the **Input** tab *(see page 336)*.

Display	Description
	Click this button to display the **AT** tab *(see page 340)*.
	Click this button to display the **AT** tab *(see page 340)*.
	This button appears when the **Authorize** option is checked in the **Conversion** zone of the **Input** tab *(see page 336)*.
	This button appears when the **Authorize** option is checked in the **Alarms** zone of the **Input** tab *(see page 336)*.
	This button appears if **Limits** is not equal to inhibit in the limits zone of the **Output** tab *(see page 342)*.
	This button appears if manual mode is not equal to Inhibit in the manual mode zone of the **Output** tab *(see page 342)*.
	Click this button to display the **Output** tab *(see page 342)*.
	This button appears when the **Authorize** option is checked in the Output PWM zone of the **Output** tab *(see page 342)*.

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.

NOTE: To retain input values following a cold restart, use memory words (%MW) and not analog inputs (%IW).

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.

Parameter	Description	
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 ≤ 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	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).
(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.	

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 322)*.

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	**Authorize**	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: ● If you set the **Operating mode** to **PID + AT** or **AT** from the **General** tab *(see page 333)*, 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.
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.

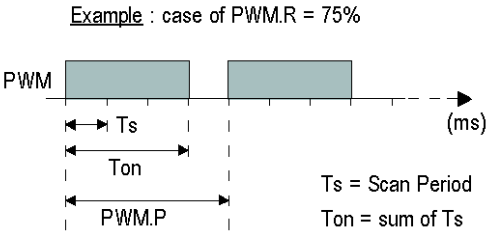
NOTE: To retain output values following a cold restart, use memory words (%MW) and not analog outputs (%QW).

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. **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 *(see page 325)*. 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.

(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.
(2) Enter a memory address (%MWxx) or an analog output address (%QWx.y).

Field	Description
Output PWM	Check this box to use the PWM function of `PID`. 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 (%MWxx) or a constant word (%KWxx). PWM precision depends on both the PWM period and the scan period. The precision is improved when the PWM ratio (%PWM.R) has the greatest number of values. For instance, with scan period = 20ms and PWM period = 200ms, PWM.R can take values 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%. With scan period = 50 ms and PWM period = 200 ms, PWM.R can take values 0%, 25%, 50%, 75% and 100% of the period PWM.P.  Specify the PWM output bit as the value in **Output**. This can be either a memory bit (%Mxx) or an output address. For further details about PWM function, refer to the *Functions Library Guide* of your logic controller.
(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. **(2)** Enter a memory address (%MWxx) or an analog output address (%QWx.y).	

Section 14.5
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:

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.

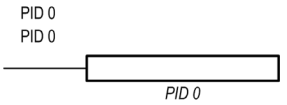
Key Features

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. Refer to the maximum number of objects table for the maximum number of PID objects available with your logic controller.
Symbol	Symbol of the current PID object	The symbol associated with this PID object. For more information, refer to Defining Symbols *(see SoMachine Basic, Operating Guide)*.
[...]	A button to launch the assistant	Click to display the **PID Assistant** screen. For further details, refer to PID Assistant *(see page 330)*.
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 page 159)* 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 current 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 333)* of the PID Assistant *(see page 330)*.

The PID state memory word can record the following types of PID information:

- Current state of the PID controller
- Current 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 State Memory Word

PID State	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 *(see page 323)* in progress
0200 hex	Auto-Tuning phase 2 *(see page 323)* in progress
0400 hex	Auto-Tuning phase 3 *(see page 323)* in progress
0800 hex	Auto-Tuning phase 4 *(see page 323)* 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
8004 hex	Process value limit out of linear conversion range

Detected Error Code	Description
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 324)*.
800D hex	Delay is negative.
800E hex	Detected error when calculating Kp. The Auto-Tuning algorithm is unstable (no convergence). This may be due to: ● 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. 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: ● the output setpoint ● the sampling period Make sure that there is no process disturbance while Auto-Tuning is in progress.
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 324)*.
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 324)*.

Detected Error Code	Description
8011 hex	The limit for static gain Kp has been exceeded, Kp>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 Ti has been exceeded, Ti > 20,000.
8013 hex	The computed value of derivative time constant Td has been exceeded, Td > 10,000.

Section 14.6
PID Parameters

What Is in This Section?

This section contains the following topics:

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 (Ti):

- If Ti $\neq$ 0, an incremental algorithm is used,
- If Ti = 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 Ti) 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, Td is the time used to anticipate the variation of the deviation. Values of Td that are too low or too high can lead to unwanted oscillations.

For each action, a suitable compromize must be found between speed and stability.

Limits of the PID Control Loop

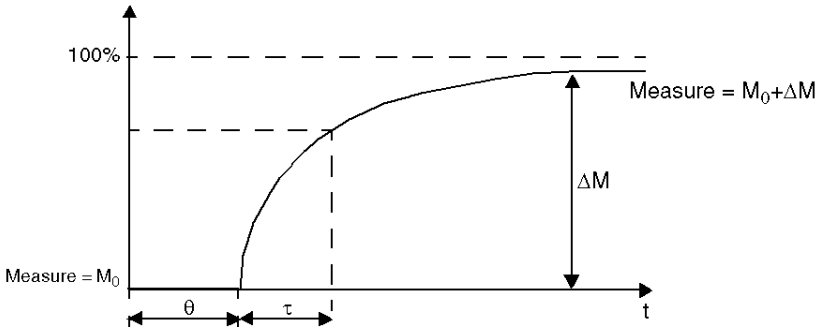
The process is assimilated to a pure delay first order with a transfer function:

$$H(p) = K \times \frac{e^{-\theta p}}{1+\tau p}$$

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 hysterisis is sufficient.

PID Parameter Adjustment Method

Introduction

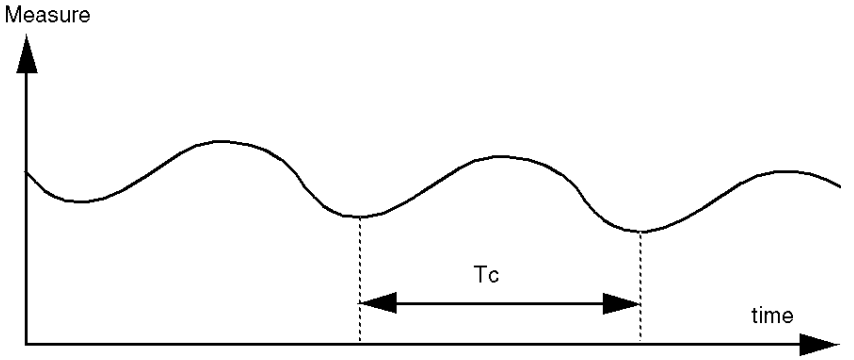
Numerous methods to adjust the PID parameters exist. The preferred method is the Ziegler and Nichols, which has 2 variants:

- closed loop adjustment
- open loop adjustment

Before implementing one of these methods, you must set the PID action *(see page 342)*.

Closed Loop Adjustment

This principle uses a proportional command (Ti = 0, Td = 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 (Kpc) which has caused the non-damped oscillation and the oscillation period (Tc) to reduce the values giving an optimal regulation.



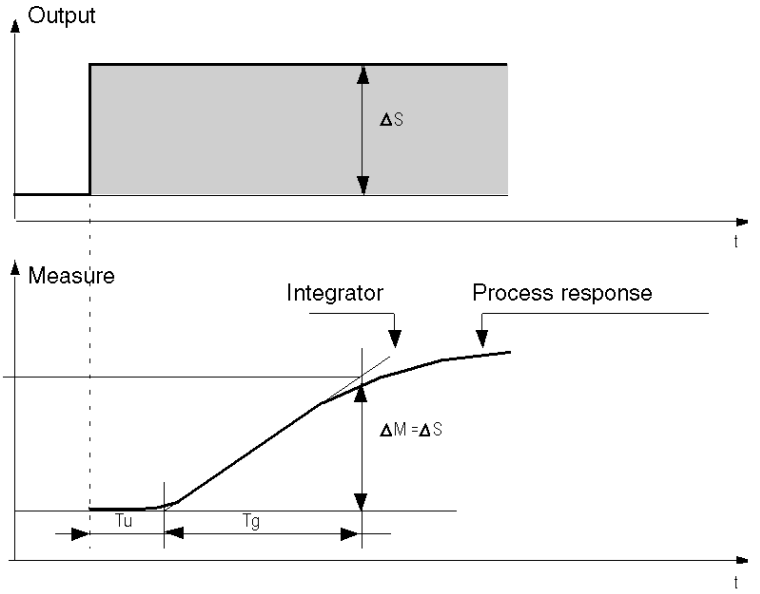
Depending on the correct type used (PID or PI), the adjustment of the coefficients is executed with the following values:

Corrector	Kp: Proportional Gain	Ti: Integration Time	Td: Derivative
PID	Kpc/1.7	Tc/2	Tc/8
PI	Kpc/2.22	0.83 x Tc	–

When the PID is implemented with Auto-Tuning, the **Dynamic AT Corrector** parameter affects the proportional gain (Kp) 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 325)*, 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 Tu. Next, the Tg 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 Tg/Tu	2 x Tu	0.5 x Tu
PI	-0.9 Tg/Tu	3.3 x Tu	–

NOTE: For further details about parameter units, refer to **PID** tab *(see page 338)*.

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 Kp, Ti, and Td.

Chapter 15
System Objects

What Is in This Chapter?

This chapter contains the following topics:

System Bits (%S)

Introduction

This section provides information about the function of system bits.

Displaying System Bits Properties

Follow these steps to display properties of the system bits:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **System objects** → **System Bits**. **Result**: System bit properties appear on the screen.

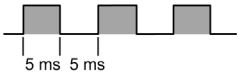
System Bits Properties

This table describes each property of the system bit:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the system bit is being referenced in a program.
Address	No	%Si	–	Displays the system bit address, where i is the bit number that represents the sequential position of the system bit in the memory. If the controller has maximum n system bits, the value of i is given as 0...n-1. For example, `%S4` is system bit 4.
Symbol	Yes	–	–	The symbol associated with the system bit. Double-click in the **Symbol** column and type the name of the symbol to associate with the system bit. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of the symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with the system bit. Double-click in the **Comment** column and type an optional comment to associate with the system bit.

System Bits Description

This table presents the description of the system bits and how they are controlled:

System Bit	Function	Description	Init State	Control
%S0	Cold start	Normally set to 0, it is set to 1 by: ● A power return with loss of data (battery malfunction), ● The program or an animation table. This bit is set to 1 during the first complete scan. It is reset to 0 by the system before the next scan.	0	S or U→S, SIM
%S4 %S5 %S6 %S7	Time base: 10 ms Time base: 100 ms Time base: 1 s Time base: 1 min	The rate of status changes is measured by an internal clock. They are not synchronized with the controller scan. Example: %S4  	–	S, SIM (except %S4)
%S10	I/O communication status	Normally set to 1 (TRUE on control panel). This bit can be set to 0 (FALSE on control panel) by the system when an I/O communication interruption is detected.	1	S
%S11	Watchdog overflow	Normally set to 0. This bit can be set to 1 by the system when the program execution time (scan time) exceeds the maximum scan time (software watchdog) or overload of processor (%SW75 exceeds 80%). Watchdog overflow causes the controller state to change to HALT.	0	S
%S12	Logic Controller in RUNNING state	This bit indicates that the controller in RUNNING state. The system sets the bit to: ● 1 when the controller state is RUNNING, ● 0 for STOPPED, BOOTING or any other state.	0	S, SIM
%S13	First cycle in RUNNING state	Normally set to 0, this bit is set to 1 by the system during the first scan after the controller state has been changed to RUNNING.	1	S, SIM
%S15	Input forced	Normally set to 0. Set to 1 by the system if at least one input is being forced.	0	S, SIM
%S16	Output forced	Normally set to 0. Set to 1 by the system if at least one output is being forced.	0	S, SIM

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S17	Last ejected bit	Normally set to 0. It is set by the system according to the value of the last ejected bit. It indicates the value of the last ejected bit.	0	S→U, SIM
%S18	Arithmetic overflow or error	Normally set to 0. It is set to 1 in the case of an overflow when a 16-bit operation is performed, that is: ● A result greater than + 32767 or less than - 32768, in single length, ● A result greater than + 2147483647 or less than - 2147483648, in double length, ● A result greater than + 3.402824E+38 or less than - 3.402824E+38, in floating point, ● Division by 0, ● The square root of a negative number, ● BTI or ITB conversion not significant: BCD value out of limits. It must be tested by the program after each operation where there is a risk of an overflow; then reset to 0 by the program if an overflow occurs.	0	S→U, SIM
%S19	Scan period overrun (periodic scan)	Normally at 0, this bit is set to 1 by the system in the event of a scan period overrun (scan time greater than the period defined by the program at configuration or programmed in %SW0). This bit is reset to 0 by the program.	0	S→U
%S20	Index overflow	Normally at 0, it is set to 1 when the address of the indexed object becomes less than 0 or more than the maximum size of an object. It must be tested by the program, after each operation where there is a risk of overflow; then reset to 0 if an overflow occurs.	0	S→U, SIM
%S21	GRAFCET initialization	Normally set to 0, it is set to 1 by: ● A cold restart, %S0 = 1, ● The program, in the preprocessing program part only, using a Set Instruction (S %S21) or a set coil –(S)– %S21, ● The terminal. At state 1, it causes GRAFCET initialization. Active steps are deactivated and initial steps are activated. It is reset to 0 by the system after GRAFCET initialization.	0	U→S, SIM

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S22	GRAFCET reset	Normally set to 0, it can only be set to 1 by the program in pre-processing. At state 1, it causes the active steps of the entire GRAFCET to be deactivated. It is reset to 0 by the system at the start of the execution of the sequential processing.	0	U→S, SIM
%S23	Preset and freeze GRAFCET	Normally set to 0, it can only be set to 1 by the program in the pre-processing program module. Set to 1, it validates the pre-positioning of GRAFCET. Maintaining this bit at 1 freezes the GRAFCET (freezes the chart). It is reset to 0 by the system at the start of the execution of the sequential processing to ensure that the GRAFCET chart moves on from the frozen situation.	0	U→S, SIM
%S33	Read or Write selection for Ethernet server configuration read/change	Normally set to 0. ● Set to 0, the %SW33 to %SW38 contains the Ethernet parameters in use (IP declared or IP assigned by BOOTP or automatic IP self assigned). These parameters are those configured in the application or those of the post configuration in SD card (in this case, %SW98 or %SW99 or %SW100 is different from 0). ● Set to 1 (if there is no post configuration in use), then the new configuration is given by %SW33 to %SW38. This bit can be set to its initial state 0 by the program and the system (on cold restart). Then, the Ethernet is reset to apply the application configuration whatever the current configuration is. This bit cannot be set to 1 if a post configuration is in use.	0	U→S
%S34	Ethernet autonegotiation	Set to 0 to allow the autonegotiation of the speed and half or full duplex mode. Set to 1 to force some specific configuration set in %S35 and %S36. **NOTE:** A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel, so after the modification, the Ethernet channel is not available for few minutes.	0	U

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S35	Ethernet half/full duplex mode	In case of the %S34 = 0 (autonegotiation) this bit is set by the system, and is read-only for the user. But if %S34 = 1, the mode is forced based on the value of this bit set by the user: ● Set to 0 if Half Duplex, ● Set to 1 if Full Duplex. **NOTE:** A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel, so after the modification, the Ethernet channel is not available for few minutes.	–	U or S
%S36	Ethernet speed	In case of the %S34 = 0 (autonegotiation) this bit is set by the system, and is read-only for the user. But if %S34 = 1, the mode is forced based on the value of this bit set by the user: ● Set to 0 if 10 Mbps, ● Set to 1 if 100 Mbps. **NOTE:** A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel, so after the modification, the Ethernet channel is not available for few minutes.	–	U or S
%S38	Permission for events to be placed in the events queue	Normally at 1. ● Set to 0, events cannot be placed in the events queue. ● Set to 1, events are placed in the events queue as soon as they are detected. This bit can be set to its initial state 1 by the program and the system (on cold restart).	1	U→S
%S39	Saturation of the events queue	Normally at 0. ● Set to 0, all events are reported. ● Set to 1, at least one event is lost. This bit can be set to 0 by the program and the system (on cold restart).	0	U→S

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S49	Output rearming	Normally set to 0, this bit can be set to 1 or 0 by the program. ● Set to 0, the automatic re-arming of outputs following a short circuit is disabled. ● Set to 1, the automatic re-arming of outputs following a short circuit is enabled. **NOTE:** The bit is reset to 0 after a cold start, otherwise the bit value is retained The system bit %S10 can be used to detect within your program that an output error has occurred. You can then use the system word %SW139 to determine programmatically in which cluster outputs a short circuit or overload has occurred. **NOTE:** %S10 and %S1W139 are reset to their initial value when %S49 is set to 1.	0	U→S
%S50	Updating the date and time using words %SW49 to %SW53	Normally set to 0, this bit can be set to 1 or 0 by the program. ● Set to 0, the date and time can be read. ● Set to 1, the date and time can be updated. The internal RTC controller is updated on a falling edge of %S50.	0	U→S
%S51	Time-of-day clock status	Normally set to 0, this bit can be set to 1 or 0 by the program. ● Set to 0, the date and time are consistent. ● Set to 1, the date and time must be initialized by the program. When this bit is set to 1, the time of day clock data is not valid. The date and time may never have been configured, the battery may be low, or the controller correction constant may be invalid (never configured, difference between the corrected clock value and the saved value, or value out of range). State 1 transitioning to state 0 forces a Write of the correction constant to the RTC.	0	U→S, SIM

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S52	RTC = error detected	This bit managed by the system indicates that the RTC correction has not been entered. Therefore, the date and time are assumed to be incorrect. ● At state 0, the RTC correction has been entered and the date and time are assumed to be correct, ● At state 1, the RTC correction has not been entered and the date and time are assumed to be incorrect, and must be initialized.	0	S, SIM
%S59	Update the date and time set in word %SW59	Normally set to 0, this bit can be set to 1 or 0 by the program: ● Set to 0, the system word %SW59 is not managed. ● On a rising edge of this bit, the date and time are incremented or decremented according to the control bits set in %SW59.	0	U
%S75	Battery status	This system bit is set by the system and can be read by the user. It indicates the battery status: ● Set to 0, the external battery is operating normally. ● Set to 1, external battery power is low, or no external battery is detected.	0	S
%S91	Erase persistant variables in flash memory	Set to 1 to erase the persistant variables stored in the flash memory.	–	U→S
%S92	%MW variables saved on flash memory	Set to 1 if there are memory word (%MW) variables stored in flash memory. Set to 0 if the data block is invalid or the write operation is in progress.	–	S
%S93	Backup %MW in flash memory	Set to 1 to store the %MW variables in flash memory (up to 1000). The logic controller must be in the STOPPED state to perform this operation.	–	U→S
%S94	Restore %MW	Set 1 to restore the data saved in the flash memory.	–	U→S
%S96	Backup program OK	This bit can be read at any time (either by the program or while adjusting), in particular after a cold start. ● Set to 0, the backup program is invalid. ● Set to 1, the backup program is valid.	0	S, SIM

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S101	Changing a port address (Modbus protocol)	Used to change a port address using system words %SW101 (port 1) and %SW102 (port 2). To do this, %S101 must be set to 1. ● Set to 0, the address cannot be changed. The value of %SW101 and %SW102 matches the current port address, ● Set to 1, the address can be changed by changing the values of %SW101 (port 1) and %SW102 (port 2). **NOTE:** After a cold start (%S0=1) all dynamic values are lost and the initial port address values are restored. **NOTE:** %S101 cannot be set to 1 if a post configuration file is defined on port 1 or port 2.	0	U
%S103 %S104	Using the ASCII protocol	Enables the use of the ASCII protocol on Comm 1 (%S103) or Comm 2 (%S104). The ASCII protocol is configured using system words %SW103 and %SW105 for Comm 1, and %SW104 and %SW106 for Comm 2. ● Set to 0, the protocol used is the one configured in SoMachine Basic, ● Set to 1, the ASCII protocol is used on Comm 1 (%S103) or Comm 2 (%S104). In this case, the system words %SW103, %SW105, and %sw121 must be previously configured for COM 1, and %SW104, %SW106, and %SW122 for COM 2. Each change of those %SW will be taken into account after a rising edge to %S103 or %S104. **NOTE:** A rising or falling edge of %S103 or %S104 cancels an exchange in progress (EXCH instruction) **NOTE:** Setting %S103 or %S104 to 0 reconfigures the serial line with the SoMachine Basic parameters.	0	U
%S119	Local I/O error detected	Normally set to 1. This bit can be set to 0 when an I/O communication interruption is detected on the logic controller. %SW118 determines the nature of the communication interruption. Resets to 1 when the communication interruption disappears.	1	S

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Words (%SW)

Introduction

This section provides information about the function of system words.

Displaying System Word Properties

Follow these steps to display properties of the system words:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **System objects** → **System Words**. **Result**: System word properties appear on the screen.

System Word Properties

This table describes each property of the system word:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the system word is being referenced in a program.
Address	No	%Si	–	Displays the system word address, where i is the word number that represents the sequential position of the system word in the memory. If the controller has maximum n system words, the value of i is given as 0...n-1. For example, `%SW50` is system bit 50.
Symbol	Yes	–	–	The symbol associated with the system word. Double-click in the **Symbol** column and type the name of the symbol to associate with the system word. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of the symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with the system word. Double-click in the **Comment** column and type an optional comment to associate with the system word.

System Words Description

This table presents the description of the system words and how they are controlled:

System Words	Function	Description	Control
%SW0	Controller scan period (master task set to periodic scan mode)	Modifies controller scan period defined at configuration through the program in an animation table.	U, SIM
%SW1	Periodic task period	Modifies the cycle time [2...255 ms] of the periodic task, without losing the **Period** value specified in the periodic task properties window. Allows you to recover the **Period** value saved in the periodic task properties window: ● In case of a cold start, or ● If the value you write in %SW1 is outside [2...255] range. The %SW1 value can be modified in the program at each end of a cycle, in the program or in an animation table without having to stop the program. Cycle times can be correctly observed while the program is running.	U, SIM
%SW6	Controller state %MW60012	Controller state: 0 = EMPTY 2 = STOPPED 3 = RUNNING 4 = HALTED 5 = POWERLESS	S, SIM
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control

%SW7	Controller state	• Bit [0]: Backup/restore in progress:
		  • Set to 1 if backup/restore of the program is in progress,
		  • Set to 0 if backup/restore of the program is complete or disabled.
		• Bit [1]: Configuration of the controller is OK:
		  • Set to 1 if configuration ok.
		• Bit [2]: SD card status bits:
		  • Set to 1 if SD card is present.
		• Bit [3]: SD card status bits:
		  • Set to 1 if SD card is being accessed.
		• Bit [4]: Application memory status:
		  • Set to 1 if application in RAM memory is different from that in flash memory.
		• Bit [6]: Not used (status 0)
		• Bit [7]: Controller reserved:
		  • Set to 1 when the controller is in connected mode with SoMachine Basic.
		• Bit [8]: Application in Write mode:
		  • Set to 1 if application is protected. In this case, the clone operation does not replicate the application (see Clone Management *(see page 150)*)
		• Bit [9]: Not used (status 0)
		• Bit [10]: Second serial port installed as cartridge (compact only):
		  • 0 = No serial cartridge
		  • 1 = Serial cartridge installed
		• Bit [11]: Second serial port type:
		  • Set to 1 = EIA RS-485
		• Bit [12]: Validity of the application in internal memory:
		  • Set to 1 if the application is valid.
		• Bit [14]: Validity of the application in RAM memory:
		  • Set to 1 if the application is valid.
		• Bit [15]: Ready for execution:
		  • Set to 1 if ready for execution.
			S, SIM

%SW11	Software watchdog value	Contains the maximum value of the watchdog. The value (10...500 ms) is defined by the configuration.	U, SIM
%SW13	BOOT version Vxx.yy	For example, if %SW13 = 000E hex: • 8 MSB = 00 in hexadecimal, then xx=0 in decimal • 8 LSB = 0E in hexadecimal, then yy=14 in decimal As a result, boot loader version is 0.14, displayed as 14 decimal.	U, SIM

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW14	Commercial version, Vxx.yy	For example, if %SW14 = 0232: ● 8 MSB = 02 in hexadecimal, then xx=2 in decimal ● 8 LSB = 32 in hexadecimal, then yy=50 in decimal As a result, commercial version is 2.50, displayed as 250 decimal.	S, SIM
%SW15 - %SW16	Firmware version, aa.bb.cc.dd	For example, if %SW15 = 0003 hex: ● 8 MSB = 00 in hexadecimal, then aa = 00 in decimal ● 8 LSB = 03 in hexadecimal, then bb = 03 in decimal For example, if %SW16 = 0B16 hex: ● 8 MSB = 0B in hexadecimal, then cc = 11 in decimal ● 8 LSB = 16 in hexadecimal, then dd = 22 in decimal As a result, firmware version is 0.3.11.22 displayed as 00031122 decimal.	S, SIM
%SW17	Default status for floating operation	When an error is detected in a floating arithmetic operation, bit %S18 is set to 1 and the default status of %SW17 is updated according to the following coding: ● Bit[0]: Invalid operation, result is not a number (NaN) ● Bit[1]: Reserved ● Bit[2]: Division by 0, result is invalid (-Infinity or +Infinity) ● Bit[3]: Result greater in absolute value than +3.402824e+38, result is invalid (-Infinity or +Infinity)	S and U, SIM
%SW18–%SW19	100 ms absolute timer counter	This counter works using 2 words: ● %SW18 represents the least significant word, ● %SW19 represents the most significant word. The double word (%SW18 - %SW19) increases from 0 to 2^31 each 100 ms as a counter modulo 2^31. This double word is also reset during the INIT phase and on a reset of %S0.	S and U, SIM
%SW30	Last scan time (master task)	Indicates the execution time of the last controller scan cycle (in ms). **NOTE:** This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a master task scan cycle. If the scan time is 2.250 ms, the %SW30 is 2 and the %SW70 is 250.	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW31	Max. scan time (master task)	Indicates the execution time of the longest controller scan cycle since the last cold start (in ms). **NOTE:** ● This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the maximum scan time is 2.25 ms, the %SW31 is 2 and the %SW71 is 250. ● To ensure proper detection of a pulse signal when the latching input option is selected, the pulse width (T_{ON}) and the period (P) must meet the following 2 requirements: ● $T_{ON} \geq 1$ ms ● The input signal period (P) must follow the Nyquist-Shannon sampling rule stating that the input signal period (P) must be at least twice the maximum program scan time (%SW31): P ≥ 2 x %SW31. **Note:** If this condition is not fulfilled, some pulses may be missed.	S
%SW32	Min. scan time (master task)	Indicates the execution time of shortest controller scan cycle since the last cold start (in ms). **NOTE:** This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the minimum scan time is 2.250 ms, the %SW32 will be 2 and the %SW72 will be 250.	S
%SW33 %SW34 %SW35 %SW36 %SW37 %SW38	IP address for Ethernet server configuration read/write	The IP settings can be modified. The read or write selection is done using the system bit %S33. The system words %SW34...%SW38 contain the Ethernet parameters: ● IP address: %SW33 and %SW34 For IP address AA.BB.CC.DD: %SW33 = CC.DD and %SW34 = AA.BB ● Subnetwork mask: %SW35 and %SW36 For subnetwork mask AA.BB.CC.DD: %SW35 = CC.DD and %SW36 = AA.BB ● Gateway address: %SW37 and %SW38 For gateway address AA.BB.CC.DD: %SW37 = CC.DD and %SW38 = AA.BB	U
%SW39	Periodic average time	Indicates the average execution time in µs of the periodic task (last 5 times)	–
%SW40	Event 0 average time	Indicates the average execution time in µs of the event task associated with the input %I0.2 (last 5 times)	–
%SW41	Event 1 average time	Indicates the average execution time in µs of the event task associated with the input %I0.3 (last 5 times)	–
%SW42	Event 2 average time	Indicates the average execution time in µs of the event task associated with the input %I0.4 (last 5 times)	–
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description		Control
%SW43	Event 3 average time	Indicates the average execution time in µs of the event task associated with the input %I0.5 (last 5 times)		–
%SW44	Event 4 average time	Indicates the average execution time in µs of the event task associated with the Threshold 0 of HSC0 or HSC2 (last 5 times)		–
%SW45	Event 5 average time	Indicates the average execution time in µs of the event task associated with the Threshold 1 of HSC0 or HSC2 (last 5 times)		–
%SW46	Event 6 average time	Indicates the average execution time in µs of the event task associated with the Threshold 0 of HSC1 or HSC3 (last 5 times)		–
%SW47	Event 7 average time	Indicates the average execution time in µs of the event task associated with the Threshold 1 of HSC1 or HSC3 (last 5 times)		–
%SW48	Number of events	Indicates how many events have been executed since the last cold start. (Counts all events except cyclic events.) **NOTE:** Set to 0 (after application loading and cold start), increments on each event execution.		S, SIM
%SW49 %SW50 %SW51 %SW52 %SW53	Real-Time Clock (RTC)	RTC functions: words containing current date and time values (in BCD):		S and U, SIM
		%SW49	xN Day of the week (N=1 for Monday)	
		%SW50	00SS Seconds	
		%SW51	HHMM: hour and minute	
		%SW52	MMDD: month and day	
		%SW53	CCYY: century and year	
		These words are controlled by the system when bit %S50 is at 0. These words can be written by the program or by the terminal when bit %S50 is set to 1. On a falling edge of %S50 the internal RTC controller is updated from the values written in these words.		
%SW54 %SW55 %SW56 %SW57	Date and time of the last stop	System words containing the date and time of the last power outage or controller stop (in BCD):		S, SIM
		%SW54	SS Seconds	
		%SW55	HHMM: hour and minute	
		%SW56	MMDD: month and day	
		%SW57	CCYY: century and year	

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description			Control
%SW58	Code of last stop	Displays code giving cause of last stop:			S, SIM
		1	Run/Stop input edge		
		2	Stop at detected software error (controller scan overshoot)		
		3	Stop command (SoMachine Basic online button)		
		4	Power outage		
		5	Stop at detected hardware error		
		6	Init in cold start		
		7	Start in stop		
		8	Low battery		
		9	Controller is not OK to run		
%SW59	Adjust current date and time	Contains 2 sets of 8 bits used to increment or decrement the current date and time. Set the appropriate bit of %SW59 first, then set %S59 from 0 to 1. The operation is performed when the bit is set on the rising edge of %S59.			U
		Increment	**Decrement**	**Parameter**	
		Bit 0	Bit 8	Day of week	Not used
		Bit 1	Bit 9	Seconds	
		Bit 2	Bit 10	Minutes	
		Bit 3	Bit 11	Hours	
		Bit 4	Bit 12	Days	
		Bit 5	Bit 13	Month	
		Bit 6	Bit 14	Years	
		Bit 7	Bit 15	Centuries	Not used
%SW62	Ethernet error detection	Indicates the error code: **0** No error detected **1** Duplicate IP: the M200 Logic Controller is configured with its default IP address (generated from the MAC address) **2** DHCP in progress **3** BOOTPin progressl **4** Invalid parameters : port is disabled **5** Fixed IP adress: initialization in progress **6** Ethernet link is down			S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW63	EXCH1 block error code	EXCH1 error code: 0 - operation was successful 1 - number of bytes to be transmitted exceeds the limit (> 255) 2 - insufficient transmission table 3 - insufficient word table 4 - receive table overflowed 5 - time-out elapsed 6 - transmission 7 - incorrect command within table 8 - selected port not configured/available 9 - reception error: This error code reflects a an incorrect or corrupted reception frame. It can be caused due to an incorrect configuration in the physical parameters (for example, parity, data bits, baudrate, and so on) or an unreliable physical connection causing signal degradation. 10 - cannot use %KW if receiving 11 - transmission offset larger than transmission table 12 - reception offset larger than reception table 13 - controller stopped EXCH processing	S
%SW64	EXCH2 block error code	EXCH2 error code: See %SW63.	S
%SW65	EXCH3 block error code	1-4, 6-13: See %SW63. (Note that error code 5 is invalid and replaced by the Ethernet-specific error codes 109 and 122 described below.) The following are Ethernet-specific error codes: 101 - incorrect IP address 102 - no TCP connection 103 - no socket available (all connection channels are busy) 104 - network is down 105 - network cannot be reached 106 - network dropped connection on reset 107 - connection aborted by peer device 108 - connection reset by peer device 109 - connection time-out elapsed 110 - rejection on connection attempt 111 - host is down 120 - incorrect index (remote device is not indexed in configuration table) 121 - system error (MAC, chip, duplicate IP) 122 - receiving process timed-out after data was sent 123 - Ethernet initialization in progress	S
%SW67	Function and type of controller	Contains the logic controller code ID. For more information, refer to the M100/M200 Logic Controller Code ID table *(see page 383)*.	S, SIM

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW70	Scan time microseconds resolution	Indicates the execution time of the last controller scan cycle (in µs). **NOTE:** This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a master task scan cycle. If the scan time is 2.250 ms, the %SW30 will be 2 and the %SW70 will be 250.	–
%SW71	Max. scan time microseconds resolution	Indicates the execution time of the longest controller scan cycle since the last cold start (in ms). **NOTE:** This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the scan time is 2.250 ms, the %SW31 will be 2 and the %SW71 will be 250.	–
%SW72	Min. scan time microseconds resolution	Indicates execution time of the shortest controller scan cycle since the last cold start (in ms). **NOTE:** This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the scan time is 2.250 ms, the %SW32 will be 2 and the %SW72 will be 250.	–
%SW75	Load of processor	Indicates the percentage of processing load. Processing load is defined as the percentage of the total available processing time that is used to process your program tasks (this value is an average and is calculated every second). In case of processing load higher than 80% for two consecutive periods of time, the controller goes to HALTED state.	S
%SW76 to %SW79	Down counters 1-4	These 4 words serve as 1 ms timers. They are decremented individually by the system every ms if they have a positive value. This gives 4 down counters down counting in ms which is equal to an operating range of 1 ms to 32767 ms. Setting bit 15 to 1 can stop decrementation.	S and U, SIM
%SW89	Cartridge slot 1 version	Indicates the firmware version of the cartridge in slot 1.	S
%SW90	Cartridge slot 2 version	Indicates the firmware version of the cartridge in slot 2.	S
%SW91	TM3 slot 1 version	Indicates the firmware version of the TM3 expansion module in slot 1.	S
%SW92	TM3 slot 2 version	Indicates the firmware version of the TM3 expansion module in slot 2.	S
%SW93	TM3 slot 3 version	Indicates the firmware version of the TM3 expansion module in slot 3.	S
%SW94 %SW95	Application's signature %MW60028- %MW60034	In case of an application change, in terms of configuration or programming data, the signature (sum of all checksums) changes so. If %SW94 = 91F3 in hexadecimal, the application's signature is 91F3 in hexadecimal.	S, SIM

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control

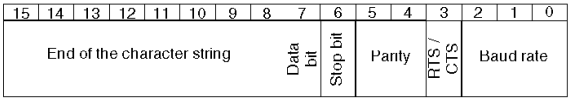
%SW96	Diagnostics for save/restore function of program and %MW	• Bit [1]: This bit is set by the firmware to indicate when the save is complete:
		  • Set to 1 if the backup is complete.
		  • Set to 0 if a new backup request is requested.
		• Bit [2]: Back up error detected, refer to bits 8, 9, 10, 12 and 14 for further information:
		  • Set to 1 if an error is detected.
		  • Set to 0 if a new backup request is requested.
		• Bit [6]: Set to 1 if the controller contains a valid application in RAM memory.
		• Bit [10]: Difference between internal RAM and Flash memory (1 = yes).
		  • Set to 1 if there is a difference.
		• Bit [12]: Indicates if a restore error has occurred:
		  • Set to 1 if an error is detected.
		• Bit [14]: Indicates if a Flash memory write error has occurred:
		  • Set to 1 if an error is detected.
			S, SIM

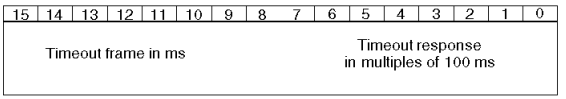
%SW97	TM3 slot 4 version	Indicates the firmware version of the TM3 expansion module in slot 4.	S
%SW98	Post configuration status (Serial Line 1)	The bits are set to 1 when the post configuration was applied for the parameter: • Bit[0]: Hardware option (RS-485 or RS-232) • Bit[1]: Baudrate • Bit[2]: Parity • Bit[3]: Data size • Bit[4]: Number of stop bits • Bit[5]: Modbus address • Bit[6]: Polarization (if available in the port)	S
%SW99	Post configuration status (Serial Line 2)	The bits are set to 1 when the post configuration was applied for the parameter: • Bit[0]: Hardware option (RS-485) • Bit[1]: Baudrate • Bit[2]: Parity • Bit[3]: Data size • Bit[4]: Number of stop bits • Bit[5]: Modbus address • Bit[6]: Polarization (if available in the port)	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW100	Post configuration status (Ethernet)	The bits are set to 1 when the post configuration was applied for the parameter: ● Bit[0]: IP mode (fixed, DHCP, or BOOTP) ● Bit[1]: IP address ● Bit[2]: Network submask ● Bit[3]: Default gateway ● Bit[4]: Device name The post configuration has priority over the configuration provided by your application. The configuration of your application is not taken into account if the M100/M200 logic controller has a post configuration.	S
%SW101 %SW102	Value of the Modbus address port	When bit %S101 is set to 1, you can change the Modbus address of port 1 or port 2. The address of port 1 is %SW101. The address of port 2 is %SW102. **NOTE:** The update is applied immediately after writing a new address to %SW101 or %SW102. After a cold start (%S0 = 1), all dynamic values are lost and the initial port address values are restored. %S101 is also updated.	S
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control

%SW103
%SW104
	Configuration for use of the ASCII protocol	When bit %S103 (Comm 1) or %S104 (Comm 2) is set to 1, the ASCII protocol is used. System word %SW103 (Comm 1) or %SW104 (Comm 2) must be set according to the elements below:
		 
		
		• Baud rate:
		  • 000: 1200 baud,
		  • 001: 2400 baud,
		  • 010: 4800 baud,
		  • 011: 9600 baud,
		  • 100: 19200 baud,
		  • 101: 38400 baud.
		  • 110: 57600 baud.
		  • 111: 115200 baud.
		• RTS/CTS:
		  • 0: disabled,
		  • 1: enabled.
		• Parity:
		  • 00: none,
		  • 10: odd,
		  • 11: even.
		• Stop bit:
		  • 0: 1 stop bit,
		  • 1: 2 stop bits.
		• Data bits:
		  • 0: 7 data bits,
		  • 1: 8 data bits.
			S, U

%SW105 %SW106	Configuration for use of the ASCII protocol	When bit %S103 (Comm 1) or %S104 (Comm 2) is set to 1, the ASCII protocol is used. System word %SW105 (Comm 1) or %SW106 (Comm 2) must be set according to the elements below:  	S, U

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW107 %SW108 %SW109	MAC address	Indicates the controller MAC address (only references with Ethernet channel). For MAC address AA:BB:CC:DD:EE:FF: ● %SW107 = AA:BB ● %SW108 = CC:DD ● %SW107 = EE:FF	S
%SW114	Enable schedule blocks	Enables or disables operation of schedule blocks by the program. Bit 0: 1 = enables schedule block number 0 ... Bit 15: 1 = enables schedule block number 15 Initially all schedule blocks are enabled. If schedule blocks are configured the default value is FFFF hex If no schedule blocks are configured, the default value is 0.	S and U, SIM
%SW118	Logic controller status word	Indicates conditions on logic controller. All the other bits of this word are set to 1 and are reserved. For a controller operating normally, the value of this word is FFFF hex. Bit 9: 0 = External error detected or communication interruption, for example duplicate IP address. Bit 10: 0 = Invalid internal configuration; contact Schneider Electric customer care. Bit 13: 0 = Configuration error detected (mandatory modules, as defined by the I/O expansion bus configuration, are absent or otherwise inoperative when the logic controller attempts to start the I/O expansion bus). In this case, the I/O bus does not start. Bit 14: 0 = One or more modules have stopped communication with the logic controller after the I/O expansion bus is started. This is the case when an I/O expansion module is defined as mandatory or optional but present at start-up. Bit 15: 0 = Cartridge error detected (configuration or runtime operation). **NOTE:** The other bits of this word are set to 1 and are reserved.	S, SIM
%SW119	Optional module feature configuration	1 bit for each expansion module in the configuration. Bit 0 = Reerved for the logic controller Bit n = Module n. 0 = Module is marked as optional in the configuration. 1 = Module is not marked as optional in the configuration.	S, SIM

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW120	Expansion I/O module status	1 bit for each expansion module in the configuration. Bit 0 = Reerved for the logic controller When the logic controller attempts to start the I/O bus, bit n: ● 0 = no error detected ● 1 = error detected or module not present. The I/O expansion bus does not start unless the corresponding bit in %SW119 is set to TRUE (indicating the module is marked as optional). When the I/O bus is started, bit n: ● 0 = no error detected ● 1 = error detected on the I/O expansion module (regardless, if it is a module marked as optional).	S, SIM
%SW121 %SW122	Configuration for use of ASCII protocol	When bit %S103 (Comm 1) or %S104 (Comm 2) is set to 1, the ASCII protocol is used. You can change the ASCII frame size of port 1 or port 2. The ASCII frame size of port 1 is %SW121, and that of port 2 is %SW122. The value is used only on EXCH instruction start. Then, if some bytes are already received, you cannot stop the reception until the last byte.	U

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW128	Cartridge 1 status	Indicates the status code for the cartridge: ● LSB: presents the status of the I/O channel 1 ● MSB: presents the status of the I/O channel 2 General status: ● 0x80: Cartridge is not present and it is not configured in SoMachine Basic. ● 0x81: Module is present but not configured. ● 0x82: Internal communication error with the cartridge. ● 0x83: Internal communication error with the cartridge. ● 0x84: Detected cartridge different from the configuration. ● 0x85: Configured cartridge is not detected. Input channel operation status: ● 0x00: Normal. ● 0x01: Conversion in progress. ● 0x02: Initialization. ● 0x03: Input operation setting error detected or module without input. ● 0x04: Reserved. ● 0x05: Wiring error detected (High limit range out). ● 0x06: Wiring error detected (Low limit range out). ● 0x07: Flash memory error detected. ● Others: Reserved. Output channel operation status: ● 0x00: Normal. ● 0x01: Reserved. ● 0x02: Initialization. ● 0x03: Output operation setting error detected or module without output. ● 0x04: Reserved. ● 0x05: Reserved. ● 0x06: Reserved. ● 0x07: Flash memory error detected. ● Others: Reserved.	S, SIM
%SW129	Cartridge 2 status		
%SW130	Event execution time	Indicates the last execution time in µs of the event input %I0.2.	S
%SW131	Event execution time	Indicates the last execution time in µsof the event input %I0.3.	S
%SW132	Event execution time	Indicates the last execution time in µs of the event input %I0.4.	S
%SW133	Event execution time	Indicates the last execution time in µs of the event input %I0.5.	S
%SW134	Event execution time	Indicates the last execution time in µs of the event task associated with the Threshold 0 of HSC0 or HSC2	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW135	Event execution time	Indicates the last execution time in µs of the event task associated with the Threshold 1 of HSC0 or HSC2	S
%SW136	Event execution time	Indicates the last execution time in µs of the event task associated with the Threshold 0 of HSC1 or HSC3	S
%SW137	Event execution time	Indicates the last execution time in µs of the event task associated with the Threshold 1 of HSC1 or HSC3	S
%SW138	Periodic task execution time	Indicates the last execution time in µs of the periodic task.	S
%SW139	Embedded digital output protection	Indicates the protection error status of output blocks: Bit0 = 1 - Q0 - Q3 protect error - Block0 Bit1 = 1 - Q4 - Q7 protect error - Block1 Bit2 = 1 - Q8 - Q11 protect error - Block2 Bit3 = 1 - Q12 - Q15 protect error - Block3	S
%SW148	Number of persistent variables	Maximum 1,000 variables. For more information, refer to Persistent Variables Saved bw User Request *(see page 40)*.	U
%SW149	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.2 .	S
%SW150	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.3 .	S
%SW151	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.4.	S
%SW152	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.5.	S
%SW153	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 0 of HSC0 or HSC2.	S
%SW154	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 1 of HSC0 or HSC2.	S
%SW155	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 0 of HSC1 or HSC3.	S
%SW156	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 1 of HSC1 or HSC3.	S
%SW157	Periodic execution time	Indicates the last execution time in ms of the periodic task.	S
%SW158	Periodic average time	Indicates the average execution time (last 5 times) of the periodic task in ms.	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW159	Event 0 average time	Indicates the average execution time (last 5 times) of event 0 in ms.	S
%SW160	Event 1 average time	Indicates the average execution time (last 5 times) of event 1 in ms.	S
%SW161	Event 2 average time	Indicates the average execution time (last 5 times) of event 2 in ms.	S
%SW162	Event 3 average time	Indicates the average execution time (last 5 times) of event 3 in ms.	S
%SW163	Event 4 average time	Indicates the average execution time (last 5 times) of event 4 in ms.	S
%SW164	Event 5 average time	Indicates the average execution time (last 5 times) of event 5 in ms.	S
%SW165	Event 6 average time	Indicates the average execution time (last 5 times) of event 6 in ms.	S
%SW166	Event 7 average time	Indicates the average execution time (last 5 times) of event 7 in ms.	S
%SW168	Modbus TCP - Connections in use	Indicates the number of Ethernet Modbus TCP server connections in use **NOTE:** If you disconnect the cable, the connection is not closed immediately. Each time the cable is re-connected to the network, it requests a new connection and the number of connections in use indicated by %SW168 increases.	S
%SW170	Frames transmitted - Serial line 1	Indicates the number of frames transmitted by the serial line 1	S
%SW171	Frames transmitted - Serial line 2	Indicates the number of frames transmitted by the serial line 2	S
%SW172	Frames transmitted - USB	Indicates the number of frames transmitted by the USB channel	S
%SW173	Frames transmitted - Modbus TCP	Indicates the number of frames transmitted by Modbus TCP on Ethernet	S
%SW174	Frames received successfully - Serial line 1	Indicates the number of frames correctly received by the serial line 1	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW175	Frames received successfully - Serial line 2	Indicates the number of frames correctly received by the serial line 2	S
%SW176	Frames received successfully - USB	Indicates the number of frames correctly received by the USB channel	S
%SW177	Frames received successfully - Modbus TCP	Indicates the number of frames correctly received by Modbus TCP on Ethernet	S
%SW178	Frames received with an error - Serial line 1	Indicates the number of frames received with an error detected by the serial line 1	S
%SW179	Frames received with an error - Serial line 2	Indicates the number of frames received with an error detected by the serial line 2	S
%SW180	Frames received with an error - USB	Indicates the number of frames received with an error by the USB channel	S
%SW181	Frames received with an error - Modbus TCP	Indicates the number of frames received with an error by Modbus TCP on Ethernet	S
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

M100/M200 Logic Controller Code ID

This table presents the code IDs of the M100/M200 Logic Controller references:

Reference	Code ID
TM100C16R	0x07C1
TM100C24R	0x07C2
TM100C40R	0x07C4
TM200C16R	0x07A1
TM200C16T	0x07A3
TM200C16U	0x07A2

Reference	Code ID
TM200C24R	0x07A7
TM200CE24R	0x07A8
TM200C24T	0x07AB
TM200CE24T	0x07AC
TM200C24U	0x07A9
TM200CE24U	0x07AA
TM200C40R	0x07AD
TM200CE40R	0x07AE
TM200C40T	0x07B1
TM200CE40T	0x07B2
TM200C40U	0x07AF
TM200CE40U	0x07B0
TM200C60R	0x07B3

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 `JerkRatio` parameters following a trapezoidal or an S-curve profile.

analog input

Converts received voltage or current levels into numerical values. You can store and process these values within the logic controller.

analog output

Converts numerical values within the logic controller and sends out proportional voltage or current levels.

application

A program including configuration data, symbols, and documentation.

ASCII

(*American standard code for Information Interchange*) A protocol for representing alphanumeric characters (letters, numbers, certain graphics, and control characters).

B

BOOTP

(*bootstrap protocol*) A UDP network protocol that can be used by a network client to automatically obtain an IP address (and possibly other data) from a server. The client identifies itself to the server using the client MAC address. The server, which maintains a pre-configured table of client device MAC addresses and associated IP addresses, sends the client its pre-configured IP address. BOOTP was originally used as a method that enabled diskless hosts to be remotely booted over a network. The BOOTP process assigns an infinite lease of an IP address. The BOOTP service utilizes UDP ports 67 and 68.

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).

D

DHCP
(*dynamic host configuration protocol*) An advanced extension of BOOTP. DHCP is more advanced, but both DHCP and BOOTP are common. (DHCP can handle BOOTP client requests.)

digital I/O
(*digital input/output*) An individual circuit connection at the electronic module that corresponds directly to a data table bit. The data table bit holds the value of the signal at the I/O circuit. It gives the control logic digital access to I/O values.

DWORD
(*double word*) Encoded in 32-bit format.

E

EtherNet/IP
(*Ethernet industrial protocol*) An open communications protocol for manufacturing automation solutions in industrial systems. EtherNet/IP is in a family of networks that implement the common industrial protocol at its upper layers. The supporting organization (ODVA) specifies EtherNet/IP to accomplish global adaptability and media independence.

expansion bus
An electronic communication bus between expansion I/O modules and a controller.

G

GRAFCET
The functioning of a sequential operation in a structured and graphic form.

This is an analytical method that divides any sequential control system into a series of steps, with which actions, transitions, and conditions are associated.

H

homing
The method used to establish the reference point for absolute movement.

I

I/O
(*input/output*)

IEC 61131-3
Part 3 of a 3-part IEC standard for industrial automation equipment. IEC 61131-3 is concerned with controller programming languages and defines 2 graphical and 2 textual programming language standards. The graphical programming languages are ladder diagram and function block diagram. The textual programming languages include structured text and instruction list.

IL
(*instruction list*) A program written in the language that is composed of a series of text-based instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand (refer to IEC 61131-3).

instruction list language
A program written in the instruction list language that is composed of a series of text-based instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand (see IEC 61131-3).

J

jerk ratio
The proportion of change of the acceleration and deceleration as a function of time.

L

ladder diagram language
A graphical representation of the instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller (see IEC 61131-3).

LAN
(*local area network*) A short-distance communications network that is implemented in a home, office, or institutional environment.

LD
(*ladder diagram*) A graphical representation of the instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller (refer to IEC 61131-3).

LSB

(*least significant bit/byte*) The part of a number, address, or field that is written as the right-most single value in conventional hexadecimal or binary notation.

M

MAST

A processor task that is run through its programming software. The MAST task has 2 sections:

- **IN:** Inputs are copied to the IN section before execution of the MAST task.
- **OUT:** Outputs are copied to the OUT section after execution of the MAST task.

Modbus

The protocol that allows communications between many devices connected to the same network.

MSB

(*most significant bit/byte* The part of a number, address, or field that is written as the left-most single value in conventional hexadecimal or binary notation.

P

periodic execution

The task is executed either cyclically or periodically. In periodic mode, you determine a specific time (period) in which the task is executed. If it is executed under this time, a waiting time is generated before the next cycle. If it is executed over this time, a control system indicates the overrun. If the overrun is too high, the controller is stopped.

PID

(*proportional, integral, derivative*) A generic control loop feedback mechanism (controller) widely used in industrial control systems.

post configuration

(*post configuration*) An option that allows to modify some parameters of the application without changing the application. Post configuration parameters are defined in a file that is stored in the controller. They are overloading the configuration parameters of the application.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

protocol

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

R

RS-485

A standard type of serial communication bus, based on 2 wires (also known as EIA RS-485).

RTC

(*real-time clock*) A battery-backed time-of-day and calender clock that operates continuously, even when the controller is not powered for the life of the battery.

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

TCP

(*transmission control protocol*) A connection-based transport layer protocol that provides a simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

trapezoidal ramp

An acceleration / deceleration ramp with a `JerkRatio` parameter set to 0%.

Index

Symbols

A

B

C

D

E

F

H

I

J

L

M

O

P

R

S

T

U

V

W