Modicon M241

Logic Controller
User Guide

06/2020



EIO0000004267.00

www.schneider-electric.com

Schneider Electric

Table of Contents

1 Modicon M241 Logic Controller - Programming Guide . . . . . Part I

2 Modicon M241 Logic Controller - System Functions and Variables PLC System Library Guide. . . . . Part II

3 Modicon M241 Logic Controller - High Speed Counting HSC Library Guide . . . . . Part III

4 Modicon M241 Logic Controller - PTOPWM Library Guide. . . . . Part IV

5 Modicon M241 Logic Controller - Hardware Guide . . . . . Part V

6 Modicon TMC4 Cartridges - Programming Guide . . . . . Part VI

7 Modicon TMC4 Cartridges - Hardware Guide . . . . . Part VII

Modicon M241 Logic Controller

Programming Guide

12/2019



EIO0000003059.02

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.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

© 2019 Schneider Electric. All rights reserved.

Table of Contents

	Safety Information	7
	About the Book	9
Chapter 1	**About the Modicon M241 Logic Controller**	15
	M241 Logic Controller Description	15
Chapter 2	**How to Configure the Controller**	21
	How to Configure the Controller	21
Chapter 3	**Libraries**	25
	Libraries	25
Chapter 4	**Supported Standard Data Types**	27
	Supported Standard Data Types	27
Chapter 5	**Memory Mapping**	29
	Controller Memory Organization	30
	RAM Memory Organization	31
	Flash Memory Organization	33
	Relocation Table	37
Chapter 6	**Tasks**	41
	Maximum Number of Tasks	42
	Task Configuration Screen	43
	Task Types	45
	System and Task Watchdogs	48
	Task Priorities	49
	Default Task Configuration	52
Chapter 7	**Controller States and Behaviors**	53
7.1	Controller State Diagram	54
	Controller State Diagram	54
7.2	Controller States Description	59
	Controller States Description	59
7.3	State Transitions and System Events	63
	Controller States and Output Behavior	64
	Commanding State Transitions	67
	Error Detection, Types, and Management	74
	Remanent Variables	75

Chapter 8 Controller Device Editor 77
Controller Parameters 78
Communication Settings 80
PLC Settings 81
Services 83
Users Rights 85
Chapter 9 Embedded Inputs and Outputs Configuration 87
Embedded I/Os Configuration 87
Chapter 10 Expert Functions Configuration 93
Expert Functions Overview 94
Counting Function 97
Pulse Generators Embedded Function 99
Chapter 11 Cartridge Configuration 101
TMC4 Cartridge Configuration 101
Chapter 12 Expansion Modules Configuration 103
TM3 I/O Configuration General Description 104
TM3 I/O Bus Configuration 110
TM4 Expansion Module Configuration 111
TM3/TM2 Expansion Module Configuration 112
Optional I/O Expansion Modules 113
Chapter 13 Ethernet Configuration 117
13.1 Ethernet Services 118
Presentation 119
IP Address Configuration 121
Modbus TCP Client/Server 127
Web Server 129
FTP Server 145
FTP Client 147
SNMP 148
Controller as a Target Device on EtherNet/IP 149
Controller as a Slave Device on Modbus TCP 175
Changing the Modbus TCP Port 180
13.2 Firewall Configuration 182
Introduction 183
Dynamic Changes Procedure 185
Firewall Behavior 186
Firewall Script Commands 188

Chapter 14 Industrial Ethernet Manager 193
Industrial Ethernet 194
DHCP Server 198
Fast Device Replacement 199
Chapter 15 Serial Line Configuration 201
Serial Line Configuration 202
Machine Expert Network Manager 204
Modbus Manager 205
ASCII Manager 209
Modbus Serial IOScanner 211
Adding a Device on the Modbus Serial IOScanner 213
Adding a Modem to a Manager 220
Chapter 16 CANopen Configuration 221
CANopen Interface Configuration 221
Chapter 17 J1939 Configuration 225
J1939 Interface Configuration 225
Chapter 18 OPC UA Server Configuration 229
OPC UA Server Overview 230
OPC UA Server Configuration 231
OPC UA Server Symbols Configuration 234
OPC UA Server Performance 236
Chapter 19 Post Configuration 239
Post Configuration Presentation 240
Post Configuration File Management 242
Post Configuration Example 244
Chapter 20 Connecting a Modicon M241 Logic Controller to a PC . . 247
Connecting the Controller to a PC 247
Chapter 21 SD Card 251
Script Files 252
SD Card Commands 253
Updating Modicon M241 Logic Controller Firmware 260
Chapter 22 Firmware Management 263
Updating TM3 Expansion Modules Firmware 263
Appendices 267
Appendix A How to Change the IP Address of the Controller 269
changeIPAddress: Change the IP address of the controller 269

Appendix B Functions to Get/Set Serial Line Configuration in User Program 273
GetSerialConf: Get the Serial Line Configuration 274
SetSerialConf: Change the Serial Line Configuration 275
SERIAL_CONF: Structure of the Serial Line Configuration Data Type 277
Appendix C Controller Performance 279
Processing Performance 279
Glossary 281
Index 293

Safety Information

Important Information

NOTICE

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



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



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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

The purpose of this document is to help you to program and operate your Modicon M241 Logic Controller with the EcoStruxure Machine Expert software.

NOTE: Read and understand this document and all related documents *(see page 9)* before installing, operating, or maintaining your Modicon M241 Logic Controller.

The Modicon M241 Logic Controller users should read through the entire document to understand all features.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.2.

The technical characteristics of the devices described in the present document also appear online. To access the information online, go to the Schneider Electric home page *www.schneider-electric.com*.

The characteristics that are described in the present document 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 document and online information, use the online information as your reference.

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert Programming Guide	*EIO0000002854 (ENG)* *EIO0000002855 (FRE)* *EIO0000002856 (GER)* *EIO0000002858 (SPA)* *EIO0000002857 (ITA)* *EIO0000002859 (CHS)*
Modicon M241 Logic Controller Hardware Guide	*EIO0000003083 (ENG)* *EIO0000003084 (FRE)* *EIO0000003085 (GER)* *EIO0000003086 (SPA)* *EIO0000003087 (ITA)* *EIO0000003088 (CHS)*

Title of Documentation	Reference Number
Modicon TM2 Expansion Modules Configuration Programming Guide	*EIO0000003432 (ENG)* *EIO0000003433 (FRE)* *EIO0000003434 (GER)* *EIO0000003435 (SPA)* *EIO0000003436 (ITA)* *EIO0000003437 (CHS)*
Modicon TM3 Expansion Modules Configuration Programming Guide	*EIO0000003119 (ENG)* *EIO0000003120 (FRE)* *EIO0000003121 (GER)* *EIO0000003122 (SPA)* *EIO0000003123 (ITA)* *EIO0000003124 (CHS)*
Modicon TM3 Bus Coupler - Programming Guide (EcoStruxure Machine Expert)	*EIO0000003635 (ENG)* *EIO0000003636 (FRA)* *EIO0000003637 (GER)* *EIO0000003638 (SPA)* *EIO0000003639 (ITA)* *EIO0000003640 (CHS)*
Modicon TM4 Expansion Modules Programming Guide	*EIO0000003149 (ENG)* *EIO0000003150 (FRE)* *EIO0000003151 (GER)* *EIO0000003152 (SPA)* *EIO0000003153 (ITA)* *EIO0000003154 (CHS)*
Modicon TMC4 Cartridges Programming Guide	*EIO0000003107 (ENG)* *EIO0000003108 (FRE)* *EIO0000003109 (GER)* *EIO0000003110 (SPA)* *EIO0000003111 (ITA)* *EIO0000003112 (CHS)*
Modicon M241 Logic Controller PLCSystem Library Guide	*EIO0000003065 (ENG)* *EIO0000003066 (FRE)* *EIO0000003067 (GER)* *EIO0000003068 (SPA)* *EIO0000003069 (ITA)* *EIO0000003070 (CHS)*
Modicon M241 Logic Controller HSC Library Guide	*EIO0000003071 (ENG)* *EIO0000003072 (FRE)* *EIO0000003073 (GER)* *EIO0000003074 (SPA)* *EIO0000003075 (ITA)* *EIO0000003076 (CHS)*

Title of Documentation	Reference Number
Modicon TM3 Expert I/O Modules - HSC Library Guide	*EIO0000003683 (ENG)* *EIO0000003684 (FRE)* *EIO0000003685 (GER)* *EIO0000003686 (SPA)* *EIO0000003687 (ITA)* *EIO0000003688 (CHS)* *EIO0000003689 (POR)* *EIO0000003690 (TUR)*
Modicon M241 Logic Controller PTO/PWM Library Guide	*EIO0000003077 (ENG)* *EIO0000003078 (FRE)* *EIO0000003079 (GER)* *EIO0000003080 (SPA)* *EIO0000003081 (ITA)* *EIO0000003082 (CHS)*
EcoStruxure Machine Expert Controller Assistant User Guide	*EIO0000001671 (ENG)* *EIO0000001672 (FRE)* *EIO0000001673 (GER)* *EIO0000001675 (SPA)* *EIO0000001674 (ITA)* *EIO0000001676 (CHS)*
FTPRemoteFileHandling Library Guide	*EIO0000002779 (ENG)* *EIO0000002780 (FRE)* *EIO0000002781 (GER)* *EIO0000002783 (SPA)* *EIO0000002782 (ITA)* *EIO0000002784 (CHS)*
SNMP Library Guide	*EIO0000002797 (ENG)* *EIO0000002798 (FRE)* *EIO0000002799 (GER)* *EIO0000002801 (SPA)* *EIO0000002800 (ITA)* *EIO0000002802 (CHS)*

You can download these technical publications and other technical information from our website at https://www.se.com/ww/en/download/ .

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Chapter 1
About the Modicon M241 Logic Controller

M241 Logic Controller Description

Overview

The M241 Logic Controller has various powerful features and can service a wide range of applications.

Software configuration, programming, and commissioning is accomplished with the EcoStruxure Machine Expert software described in detail in the EcoStruxure Machine Expert Programming Guide and the M241 Logic Controller Programming Guide.

Programming Languages

The M241 Logic Controller is configured and programmed with the EcoStruxure Machine Expert software, which supports the following IEC 61131-3 programming languages:

- IL: Instruction List
- ST: Structured Text
- FBD: Function Block Diagram
- SFC: Sequential Function Chart
- LD: Ladder Diagram

EcoStruxure Machine Expert software can also be used to program these controllers using CFC (Continuous Function Chart) language.

Power Supply

The power supply of the M241 Logic Controller is 24 Vdc or 100...240 Vac.

Real Time Clock

The M241 Logic Controller includes a Real Time Clock (RTC) system *(see Modicon M241 Logic Controller, Hardware Guide)*.

Run/Stop

The M241 Logic Controller can be operated externally by the following:

- A hardware Run/Stop switch.
- A Run/Stop operation by a dedicated digital input, defined in the software configuration. For more information, refer to Configuration of Digital Inputs *(see page 88)*.
- An EcoStruxure Machine Expert software command.

Memory

This table describes the different types of memory:

Memory Type	Size	Used to
RAM	64 Mbytes, of which 8 Mbytes available for the application	execute the application.
Non-volatile	128 Mbytes	save the program and data in case of a power interruption.

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/source transistor outputs
- Fast sink/source transistor outputs associated with pulse generators
- Relay outputs

Removable Storage

The M241 Logic Controllers include an embedded SD card slot.

The main uses of the SD card are:

- Initializing the controller with a new application
- Updating the controller firmware
-
- Storing recipes files
- Receiving data logging files

Embedded Communication Features

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

- CANopen Master
- Ethernet
- USB Mini-B
- Serial Line 1
- Serial Line 2

Expansion Module and Bus Coupler Compatibility

Refer to the compatibility tables in the EcoStruxure Machine Expert - Compatibility and Migration User Guide.

M241 Logic Controller

Reference	Digital Inputs	Digital Outputs	Communication Ports	Terminal Type	Power supply
TM241C24R	6 regular inputs(1) 8 fast inputs (counters)(2)	6 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	100...240 Vac
TM241CE24R	6 regular inputs(1) 8 fast inputs (counters)(2)	6 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	100...240 Vac
TM241CEC24R	6 regular inputs(1) 8 fast inputs (counters)(2)	6 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 Ethernet port 1 CANopen master port 1 USB programming port	Removable screw terminal blocks	100...240 Vac
TM241C24T	6 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc
TM241CE24T	6 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc
TM241CEC24T	6 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port 1 CANopen master port	Removable screw terminal blocks	24 Vdc
TM241C24U	6 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc

(1) The regular inputs have a maximum frequency of 1 kHz.
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.
(3) The fast transistor outputs can be used either as regular transistor outputs, as reflex outputs for counting function (HSC), or as fast transistor outputs for pulse generator functions (FreqGen / PTO / PWM).

Reference	Digital Inputs	Digital Outputs	Communication Ports	Terminal Type	Power supply
TM241CE24U	6 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc
TM241CEC24U	6 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port 1 CANopen master port	Removable screw terminal blocks	24 Vdc
TM241C40R	16 regular inputs(1) 8 fast inputs (counters)(2)	12 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	100...240 Vac
TM241CE40R	16 regular inputs(1) 8 fast inputs (counters)(2)	12 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	100...240 Vac
TM241C40T	16 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc
TM241CE40T	16 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc
TM241C40U	16 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc

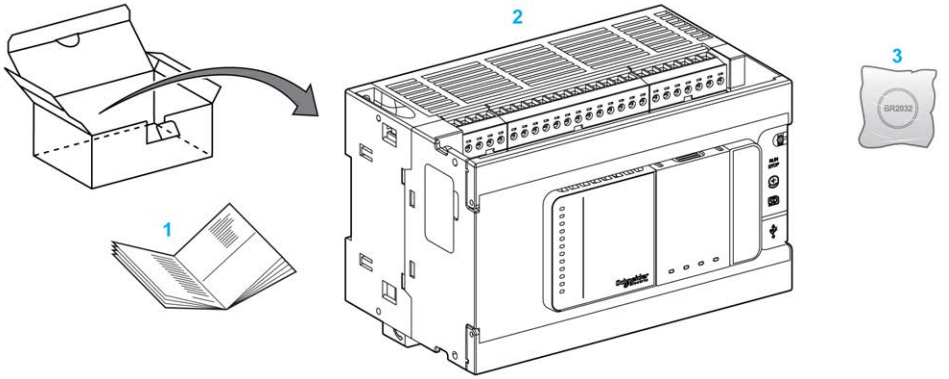
(1) The regular inputs have a maximum frequency of 1 kHz.
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.
(3) The fast transistor outputs can be used either as regular transistor outputs, as reflex outputs for counting function (HSC), or as fast transistor outputs for pulse generator functions (FreqGen / PTO / PWM).

Reference	Digital Inputs	Digital Outputs	Communication Ports	Terminal Type	Power supply
TM241CE40U	16 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc

(1) The regular inputs have a maximum frequency of 1 kHz.
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.
(3) The fast transistor outputs can be used either as regular transistor outputs, as reflex outputs for counting function (HSC), or as fast transistor outputs for pulse generator functions (FreqGen / PTO / PWM).

Delivery Content

The following figure presents the content of the delivery for a M241 Logic Controller:



1 M241 Logic Controller Instruction Sheet
2 M241 Logic Controller
3 Lithium carbon monofluoride battery, type Panasonic BR2032.

Chapter 2
How to Configure the Controller

How to Configure the Controller

Introduction

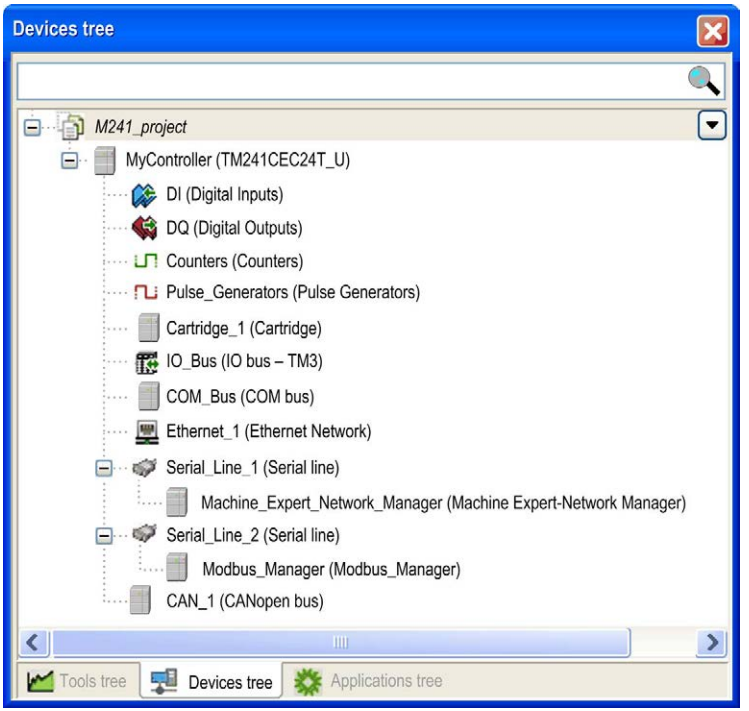
First, create a new project or open an existing project in the EcoStruxure Machine Expert software.

Refer to the EcoStruxure Machine Expert Programming Guide for information on how to:

- add a controller to your project
- add expansion modules to your controller
- replace an existing controller
- convert a controller to a different but compatible device

Devices Tree

The **Devices tree** presents a structured view of the current hardware configuration. When you add a controller to your project, a number of nodes are added to the **Devices tree**, depending on the functions the controller provides.



Item	Use to Configure...
DI	Embedded digital inputs of the logic controller
DQ	Embedded digital outputs of the logic controller
Counters	Embedded counting functions (HSC)
Pulse_Generators	Embedded pulse generator functions (PTO/PWM/FreqGen)
Cartridge_x	Cartridges plugged into the logic controller
IO_Bus	Expansion modules connected to the logic controller
COM_Bus	Communications bus of the logic controller
Ethernet_x	Embedded Ethernet, serial line, or CANopen communications interfaces **NOTE:** Ethernet and CANopen are only available on some references.
Serial_Line_x	
CAN_x	

Applications Tree

The **Applications tree** allows you to manage project-specific applications as well as global applications, POUs, and tasks.

Tools Tree

The **Tools tree** allows you to configure the HMI part of your project and to manage libraries.

Chapter 3
Libraries

Libraries

Introduction

Libraries provide functions, function blocks, data types and global variables that can be used to develop your project.

The **Library Manager** of EcoStruxure Machine Expert provides information about the libraries included in your project and allows you to install new ones. For more information on the **Library Manager**, refer to the Functions and Libraries User Guide.

Modicon M241 Logic Controller

When you select a Modicon M241 Logic Controller for your application, EcoStruxure Machine Expert automatically loads the following libraries:

Library name	Description
IoStandard	**CmpIoMgr** configuration types, **ConfigAccess**, Parameters and help functions: manages the I/Os in the application.
Standard	Contains functions and function blocks which are required matching IEC61131-3 as standard POUs for an IEC programming system. Link the standard POUs to the project (standard.library).
Util	Analog Monitors, BCD Conversions, Bit/Byte Functions, Controller Datatypes, Function Manipulators, Mathematical Functions, Signals.
PLCCommunication *(see EcoStruxure Machine Expert, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	**SysMem, Standard**. These functions facilitate communications between specific devices. Most of them are dedicated to Modbus exchange. Communication functions are processed asynchronously with regard to the application task that called the function.
M241 PLCSystem *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*	Contains functions and variables to get information and send commands to the controller system.
M241 HSC *(see Modicon M241 Logic Controller, High Speed Counting, HSC Library Guide)*	Contains function blocks and variables to get information and send commands to the Fast Inputs/Outputs of the Modicon M241 Logic Controller. These function blocks permit you to implement HSC (High Speed Counting) functions on the Fast Inputs/Outputs of the Modicon M241 Logic Controller.

Library name	Description
M241 PTOPWM *(see Modicon M241 Logic Controller, PTOPWM, Library Guide)*	Contains function blocks and variables to get information and send commands to the Fast Inputs/Outputs of the Modicon M241 Logic Controller. These function blocks permit you to implement PTO (Pulse Train Output) and PWM (Pulse Width Modulation) functions on the Fast Outputs of the Modicon M241 Logic Controller.
Relocation Table *(see page 37)*	Allows you organization of data to optimize exchanges between the Modbus client and the controller, by regrouping non-contiguous data into a contiguous table of registers.

Chapter 4
Supported Standard Data Types

Supported Standard Data Types

Supported Standard Data Types

The controller supports the following IEC data types:

Data Type	Lower Limit	Upper Limit	Information Content
BOOL	FALSE	TRUE	1 Bit
BYTE	0	255	8 Bit
WORD	0	65,535	16 Bit
DWORD	0	4,294,967,295	32 Bit
LWORD	0	$2^{64}-1$	64 Bit
SINT	-128	127	8 Bit
USINT	0	255	8 Bit
INT	-32,768	32,767	16 Bit
UINT	0	65,535	16 Bit
DINT	-2,147,483,648	2,147,483,647	32 Bit
UDINT	0	4,294,967,295	32 Bit
LINT	-2^{63}	$2^{63}-1$	64 Bit
ULINT	0	$2^{64}-1$	64 Bit
REAL	1.175494351e-38	3.402823466e+38	32 Bit
STRING	1 character	255 characters	1 character = 1 byte
WSTRING	1 character	255 characters	1 character = 1 word
TIME	-	-	32 Bit

For more information on `ARRAY`, `LTIME`, `DATE`, `TIME`, `DATE_AND_TIME`, and `TIME_OF_DAY`, refer to the EcoStruxure Machine Expert Programming Guide.

Chapter 5
Memory Mapping

Introduction

This chapter describes the memory maps and sizes of the different memory areas in the Modicon M241 Logic Controller. These memory areas are used to store user program logic, data and the programming libraries.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Controller Memory Organization	30
RAM Memory Organization	31
Flash Memory Organization	33
Relocation Table	37

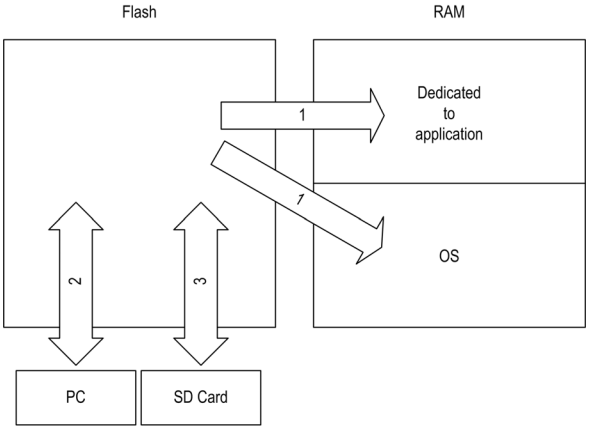
Controller Memory Organization

Introduction

The controller memory is composed of two types of physical memory:

- The Flash memory *(see page 33)* contains files (application, configuration files, and so on).
- The Random Access Memory (RAM) *(see page 31)* is used for application execution.

Files Transfers in Memory



Item	Controller State	File Transfer Events	Connection	Description
1	–	Initiated automatically at Power ON and Reboot	Internal	Files transfer from Flash memory to RAM. The content of the RAM is overwritten.
2	All states except INVALID_OS (1)	Initiated by user	Ethernet or USB programming port	Files can be transferred via: ● Web server *(see page 129)* ● FTP server *(see page 145)* ● **Controller Assistant** ● EcoStruxure Machine Expert
3	All states	Initiated automatically by script (data transfer) or by power cycle (cloning) when an SD card is inserted	SD card	Up/download with SD card(1).
(1) If the controller is in the INVALID_OS state, the only accessible memory is the SD card and only for firmware upgrades.				

NOTE: The modification of files in Flash memory does not affect a running application. Any changes to files in Flash memory are taken into account at the next reboot.

RAM Memory Organization

Introduction

This section describes the RAM (Random Access Memory) size for different areas of the Modicon M241 Logic Controller.

Memory Mapping

The RAM size is 64 Mbytes.

The RAM is composed of 2 areas:

- dedicated application memory
- OS memory

This table describes the dedicated application memory:

Area	Element	Size
System area 192 Kbytes	System Area Mappable Addresses %MW0...%MW59999	128 Kbytes
	System and diagnostic variables (%MW60000...%MW60199) This memory is accessible through Modbus requests only. These must be read-only requests.	
	Dynamic Memory Area: Read Relocation Table *(see page 37)* (%MW60200...%MW61999) This memory is accessible through Modbus requests only. These must be read-only requests.	
	System and diagnostic variables (%MW62000...%MW62199) This memory is accessible through Modbus requests only. These can be read or write requests.	
	Dynamic Memory Area: Write Relocation Table *(see page 37)* (%MW62200...%MW63999) This memory is accessible through Modbus requests only. These can be read or write requests.	
	%MW64000…%MW65535 Reserved	
	Retain and Persistent data *(see page 33)*	64 Kbytes
User area 8 Mbytes	Symbols	Dynamic allocation
	Variables	
	Application	
	Libraries	

System and Diagnostic Variables

Variables	Description
PLC_R	Structure of controller read-only system variables.
PLC_W	Structure of controller read/write system variables.
ETH_R	Structure of Ethernet read-only system variables.
ETH_W	Structure of Ethernet read/write system variables.
PROFIBUS_R	Structure of PROFIBUS DP read-only system variables.
SERIAL_R	Structure of Serial Lines read-only system variables.
SERIAL_W	Structure of Serial Lines read/write system variables.
TM3_MODULE_R	Structure of TM3 modules read-only system variables.

For more information on system and diagnostic variables, refer to *M241 PLCSystem Library Guide*.

Memory Addressing

This table describes the memory addressing for the address sizes Double Word (%MD), Word (%MW), Byte (%MB), and Bit (%MX):

Double Words	Words	Bytes	Bits		
%MD0	%MW0	%MB0	%MX0.7	...	%MX0.0
		%MB1	%MX1.7	...	%MX1.0
	%MW1	%MB2	%MX2.7	...	%MX2.0
		%MB3	%MX3.7	...	%MX3.0
%MD1	%MW2	%MB4	%MX4.7	...	%MX4.0
		%MB5	%MX5.7	...	%MX5.0
	%MW3	%MB6	%MX6.7	...	%MX6.0
		%MB7	%MX7.7	...	%MX7.0
%MD2	%MW4	%MB8	%MX8.7	...	%MX8.0
		...	...	...	...
	...	...	...	...	...
		...	...	...	...

Example of overlap of memory ranges:

%MD0 contains %MB0 (...) %MB3, %MW0 contains %MB0 and %MB, %MW1 contains %MB2 and %MB3.

NOTE: The Modbus communication is asynchronous with the application.

Flash Memory Organization

Introduction

The Flash memory contains the file system used by the controller.

File Type

The Modicon M241 Logic Controller manages the following file types:

Type	Description
Boot application	This file resides in Flash memory and contains the compiled binary code of the executable application. Each time the controller is rebooted, the executable application is extracted from the boot application and copied into the controller RAM (1).
Application source	Source file that can be uploaded from Flash memory to the PC if the source file is not available on the PC (2).
Post configuration	File that contains Ethernet, serial line, and firewall parameters. The parameters specified in the file override the parameters in the executable application at each reboot.
Data logging	Files in which the controller logs events as specified by the application.
HTML page	HTML pages displayed by the web server for the website embedded in the controller.
Operating System (OS)	Controller firmware that can be written to Flash memory. The firmware file is applied at next reboot of the controller.
Retain variable	Remanent variables
Retain-persistent variable	

(1) The creation of a boot application is optional in EcoStruxure Machine Expert, according to application properties. Default option is to create the boot application on download. When you download an application from EcoStruxure Machine Expert to the controller, you are transferring only the binary executable application directly to RAM.

(2) EcoStruxure Machine Expert does not support uploading of either the executable application or the boot application to a PC for modification. Program modifications must be made to the application source. When you download your application, you have the option to store the source file to Flash memory.

File Organization

This table shows the file organization of the flash memory:

Disk	Directory	File	Content	Up/Downloaded Data Type
/sys	OS	M241M251FW1v_XX.YY (1)	Firmware of core 1	Firmware
		M241M251FW2v_XX.YY (1)	Firmware of core 2	
		Version.ini	Control file for firmware version	
	Web	Index.htm	HTML pages served by the web server for the website embedded in the controller.	Website
		Conf.htm		–
		...		–
/usr	App	Application.app	Boot application	Application
		Application.crc		–
		Application.map		–
		Archive.prj (2)	Application source	–
		settings.conf (3)	OPC UA configuration	Configuration
		OpcUASymbolConf.map (3)	OPC UA symbols configuration	Configuration
	Cfg	Machine.cfg (2)	Post configuration file *(see page 239)*	Configuration
		CodesysLateConf.cfg (2)	• Name of application to launch • Routing table (main/sub net)	Configuration

(1): v_XX.YY represents the version
(2): if any
(3): if OPC UA *(see page 231)* is configured
(4): the Fdr/FDRS directory is hidden

Disk	Directory	File	Content	Up/Downloaded Data Type
/usr	Log	*UserDefinedLogName_1*.log	All *.log files created using the data logging functions *(see SoMachine, Data Logging Functions, DataLogging Library Guide)*. You must specify the total number of files created and the names and contents of each log file.	log file
		...	–	–
		UserDefinedLogName_n.log	–	–
	Rcp		Main directory for Recipe	–
	Syslog	crashC1.txt(2) crashC2.txt(2) crashBoot.txt(2)	This file contains a record of detected system errors. For use by Schneider Electric Technical Support.	Log file
		PlcLog.txt (2)	This file contains system event data that is also visible online in EcoStruxure Machine Expert by viewing the **Log** tab of the **Controller Device Editor** *(see page 78)*.	–
		FwLog.txt	This file contains a record of firmware system events. For use by Schneider Electric Technical Support.	–
/usr	Fdr/FDRS (4) only for TM241CE•	Device1.prm	Parameter files stored by the FDR client device1	FDR *(see page 199)*
		/data	–	
		/sd0	–	
–	User files		–	–

(1): v_XX.YY represents the version
(2): if any
(3): if OPC UA *(see page 231)* is configured
(4): the Fdr/FDRS directory is hidden

NOTE: For more information on libraries and available function blocks, refer to Libraries *(see page 25)*.

Files Redirection

When system, program or certain user activity creates specific file types, the M241 Logic Controller examines the file extension and automatically moves the file to a corresponding folder in flash memory.

The following table lists the file types that are moved in this way and the destination folder in flash memory:

File extensions	Flash memory folder
*.app, *.ap_, *.err, *.crc, *.frc, *.prj	/usr/App
*.cfg, *.cf_	/usr/Cfg
*.log	/usr/Log
*.rcp, *.rsi	/usr/Rcp

Backup Data Logging File

Data logging files can become large to the point of exceeding the space available in the file system. Therefore, you should develop a method to archive the log data periodically on an SD card. You could split the log data into several files, for example `LogMonth1, LogMonth2`, and use the **ExecuteScript** *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* command to copy the first file to an SD card. Afterwards, you may remove it from the internal file system while the second file is accumulating data. If you allow the data logging file to grow and exceed the limits of the file size, you could lose data.

NOTICE
LOSS OF APPLICATION DATA ● Backup SD card data regularly. ● Do not remove power or reset the controller, and do not insert or remove the SD card while it is being accessed. **Failure to follow these instructions can result in equipment damage.**

Relocation Table

Introduction

The **Relocation Table** allows you to organize data to optimize communication between the controller and other equipment by regrouping non-contiguous data into a contiguous table of located registers, accessible through Modbus.

NOTE: A relocation table is considered an object. Only one relocation table object can be added to a controller.

Relocation Table Description

This table describes the **Relocation Table** organization:

Register	Description
60200...61999	Dynamic Memory Area: Read Relocation Table
62200...63999	Dynamic Memory Area: Write Relocation Table

For further information, refer to *M241 PLCSystem Library Guide*.

Adding a Relocation Table

This table describes how to add a **Relocation Table** to your project:

Step	Action
1	In the **Applications tree** tab, select the **Application** node.
2	Click the right mouse button.
3	Click **Objects → Relocation Table...**. **Result:** The **Add Relocation Table** window is displayed.
4	Click **Add**. **Result:** The new relocation table is created and initialized. **NOTE:** As a Relocation Table is unique for a controller, its name is **Relocation Table** and cannot be changed.

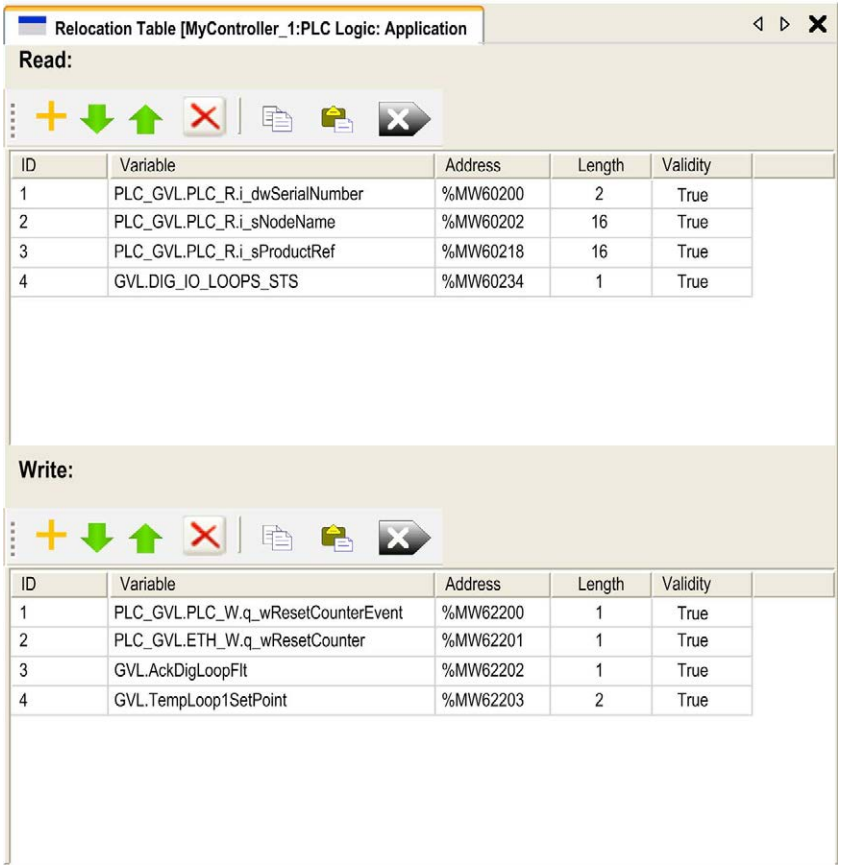
Relocation Table Editor

The relocation table editor allows you to organize your variables in the relocation table.

To access the relocation table editor, double-click the **Relocation Table** node in the **Tools tree** tab:



This picture describes the relocation table editor:



ID	Variable	Address	Length	Validity
1	PLC_GVL.PLC_R.i_dwSerialNumber	%MW60200	2	True
2	PLC_GVL.PLC_R.i_sNodeName	%MW60202	16	True
3	PLC_GVL.PLC_R.i_sProductRef	%MW60218	16	True
4	GVL.DIG_IO_LOOPS_STS	%MW60234	1	True

ID	Variable	Address	Length	Validity
1	PLC_GVL.PLC_W.q_wResetCounterEvent	%MW62200	1	True
2	PLC_GVL.ETH_W.q_wResetCounter	%MW62201	1	True
3	GVL.AckDigLoopFlt	%MW62202	1	True
4	GVL.TempLoop1SetPoint	%MW62203	2	True

Icon	Element	Description
	New Item	Adds an element to the list of system variables.
	Move Down	Moves down the selected element of the list.
	Move Up	Moves up the selected element of the list.
	Delete Item	Removes the selected elements of the list.
	Copy	Copies the selected elements of the list.
	Paste	Pastes the elements copied.
	Erase Empty Item	Removes all the elements of the list for which the "Variable" column is empty.
-	ID	Automatic incremental integer (not editable).
-	Variable	The name or the full path of a variable (editable).
-	Address	The address of the system area where the variable is stored (not editable).
-	Length	Variable length in word.
-	Validity	Indicates if the entered variable is valid (not editable).

NOTE: If a variable is undefined after program modifications, the content of the cell is displayed in red, the related **Validity** cell is False, and **Address** is set to -1.

Chapter 6
Tasks

Introduction

The **Task Configuration** node in the **Applications tree** allows you to define one or more tasks to control the execution of your application program.

The task types available are:

- Cyclic
- Freewheeling
- Event
- External event

This chapter begins with an explanation of these task types and provides information regarding the maximum number of tasks, the default task configuration, and task prioritization. In addition, this chapter introduces the system and task watchdog functions and explains its relationship to task execution.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Maximum Number of Tasks	42
Task Configuration Screen	43
Task Types	45
System and Task Watchdogs	48
Task Priorities	49
Default Task Configuration	52

Maximum Number of Tasks

Maximum Number of Tasks

The maximum number of tasks you can define for the Modicon M241 Logic Controller are:

- Total number of tasks = 19
- Cyclic tasks = 5
- Freewheeling tasks = 1
- Event tasks = 8
- External Event tasks = 8

Special Considerations for Freewheeling

A Freewheeling task *(see page 46)* does not have a fixed duration. In Freewheeling mode, each task scan starts when the previous scan has been completed and after a period of system processing (30% of the total duration of the Freewheeling task). If the system processing period is reduced to less than 15% for more than 3 seconds due to interruptions by other tasks, a system error is detected. For more information, refer to the System Watchdog *(see page 48)*.

NOTE: You may wish to avoid using a Freewheeling task in a multi-task application when some high priority and time-consuming tasks are running. Doing so may provoke a task Watchdog Timeout. You should not assign CANopen to a freewheeling task. CANopen should be assigned to a cyclic task.

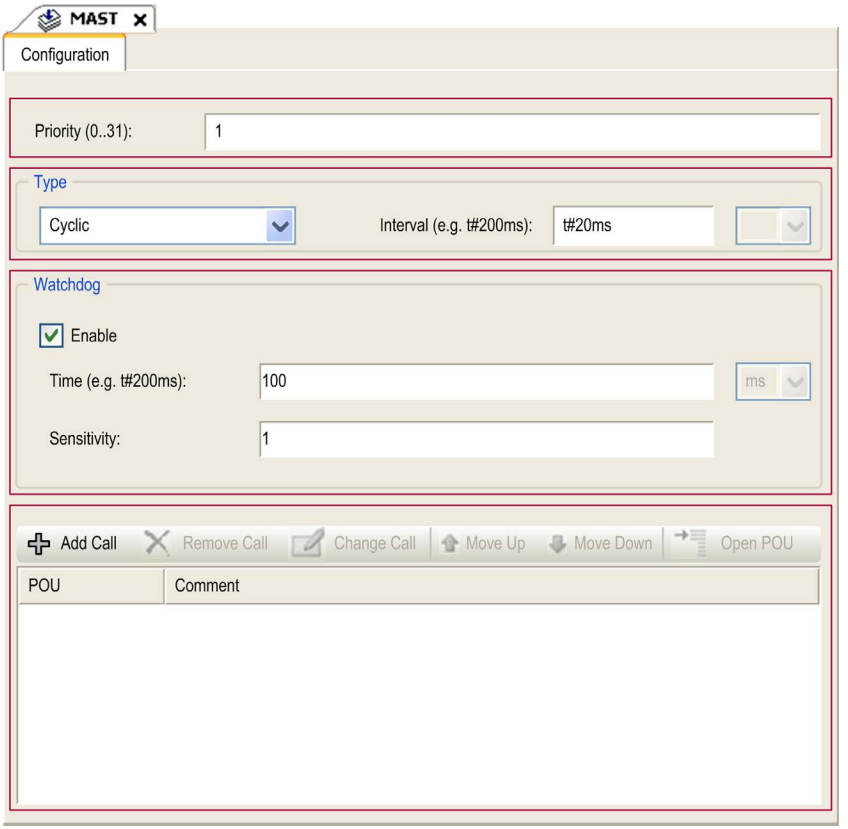
Task Configuration Screen

Screen Description

This screen allows you to configure the tasks. Double-click the task that you want to configure in the **Applications tree** to access this screen.

Each configuration task has its own parameters that are independent of the other tasks.

The **Configuration** window is composed of 4 parts:



The table describes the fields of the **Configuration** screen:

Field Name	Definition
Priority	Configure the priority of each task with a number from 0 to 31 (0 is the highest priority, 31 is the lowest). Only one task at a time can be running. The priority determines when the task will run: ● a higher priority task will pre-empt a lower priority task ● tasks with same priority will run in turn (2 ms time-slice) **NOTE:** Do not assign tasks with the same priority. If there are yet other tasks that attempt to pre-empt tasks with the same priority, the result could be indeterminate and unpredicable. For important safety information, refer to Task Priorities *(see page 49)*.
Type	These task types are available: ● Cyclic *(see page 45)* ● Event *(see page 47)* ● External *(see page 47)* ● Freewheeling *(see page 46)*
Watchdog	To configure the watchdog *(see page 48)*, define these 2 parameters: ● **Time**: enter the timeout before watchdog execution. ● **Sensitivity**: defines the number of expirations of the watchdog timer before the controller stops program execution and enters a HALT state.
POUs	The list of POUs *(see EcoStruxure Machine Expert, Programming Guide)* (Programming Organization Units) controlled by the task is defined in the task configuration window: ● To add a POU linked to the task, use the command **Add Call** and select the POU in the **Input Assistant** editor. ● To remove a POU from the list, use the command **Remove Call**. ● To replace the currently selected POU of the list by another one, use the command **Change Call**. ● **POU**s are executed in the order shown in the list. To move the **POU**s in the list, select a **POU** and use the command **Move Up** or **Move Down**. **NOTE:** You can create as many POUs as you want. An application with several small POUs, as opposed to one large POU, can improve the refresh time of the variables in online mode.

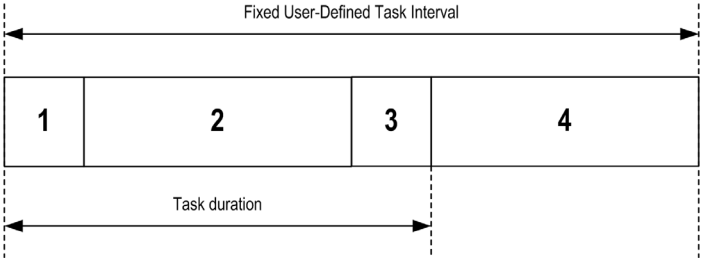
Task Types

Introduction

The following section describes the various task types available for your program, along with a description of the task type characteristics.

Cyclic Task

A Cyclic task is assigned a fixed cycle time using the Interval setting in the Type section of Configuration subtab for that task. Each Cyclic task type executes as follows:



1. **Read Inputs:** The physical input states are written to the %I input memory variables and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variables are updated according to your application program instructions but not yet written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variables are modified with any output forcing that has been defined; however, the writing of the physical outputs depends upon the type of output and instructions used.
   For more information on defining the bus cycle task, refer to the EcoStruxure Machine Expert Programming Guide and Modicon M241 Logic Controller Settings *(see page 81)*.
   For more information on I/O behavior, refer to Controller States Detailed Description *(see page 59)*.
4. **Remaining Interval time:** The controller firmware carries out system processing and any other lower priority tasks.

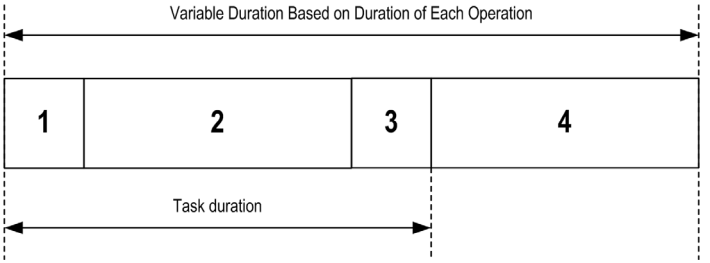
NOTE: If you define too short a period for a cyclic task, it will repeat immediately after the write of the outputs and without executing other lower priority tasks or any system processing. This will affect the execution of all tasks and cause the controller to exceed the system watchdog limits, generating a system watchdog exception.

NOTE: When the task cycle time is set to a value less than 3 ms, the actual task duration should first be monitored through the Task Monitoring screen during commissioning to ensure that it is consistently lower than the configured task cycle time. If greater, the task cycle may not be respected without causing a task cycle watchdog time-out and the controller transitioning to a HALT state. To avoid this condition to a certain degree, when the task cycle time is set to a value of less than 3 ms, real limits of +1 ms are imposed if, on any given cycle, the calculated cycle time slightly exceeds the configured cycle time.

NOTE: Get and set the interval of a Cyclic Task by application using the **GetCurrentTaskCycle** and **SetCurrentTaskCycle** function. (Refer to Toolbox Advance Library Guide for further details.)

Freewheeling Task

A Freewheeling task does not have a fixed duration. In Freewheeling mode, each task scan begins when the previous scan has been completed and after a short period of system processing. Each Freewheeling task type executes as follows:



1. **Read Inputs:** The physical input states are written to the %I input memory variables and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variables are updated according to your application program instructions but not yet written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variables are modified with any output forcing that has been defined; however, the writing of the physical outputs depends upon the type of output and instructions used.
   For more information on defining the bus cycle task, refer to the EcoStruxure Machine Expert Programming Guide and Modicon M241 Logic Controller Settings *(see page 81)*.
   For more information on I/O behavior, refer to Controller States Detailed Description *(see page 59)*.
4. **System Processing:** The controller firmware carries out system processing and any other lower priority tasks (for example: HTTP management, Ethernet management, parameters management).

NOTE: If you want to define the task interval, refer to Cyclic Task *(see page 45)*.

Event Task

This type of task is event-driven and is initiated by a program variable. It starts at the rising edge of the boolean variable associated to the trigger event unless pre-empted by a higher priority task. In that case, the Event task will start as dictated by the task priority assignments.

For example, if you have defined a variable called `my_Var` and would like to assign it to an Event, proceed as follows:

Step	Action
1	Double-click the **TASK** in the **Applications tree**.
2	Select **Event** from the **Type** list in the **Configuration** tab.
3	Click the **Input Assistant** button  to the right of the **Event** field. **Result**: The **Input Assistant** window appears.
4	Navigate in the tree of the **Input Assistant** dialog box to find and assign the `my_Var` variable.

NOTE: When the event task is triggered at a too high frequency, the controller will go to the HALT state (Exception).The maximum rate of events is 6 events per millisecond. If the event task is triggered at a higher frequency than this, the message 'ISR Count Exceeded' is logged in the application log page.

External Event Task

This type of task is event-driven and is initiated by the detection of a hardware or hardware-related function event. It starts when the event occurs unless pre-empted by a higher priority task. In that case, the External Event task will start as dictated by the task priority assignments.

For example, an External event task could be associated with an HSC Stop event. To associate the **HSC0_STOP** event to an External event task, select it from the **External event** drop-down list on the **Configuration** tab.

Depending on the controller, there are up to 4 types of events that can be associated with an External event task:

- Rising edge on an advanced input (`DI0...DI15`)
- HSC thresholds
- HSC Stop
- CAN Sync

NOTE: CAN Sync is a specific event object, depending on the **CANopen manager** configuration.

NOTE: The maximum frequency of events is 6 per millisecond. If the external event task is triggered at a higher frequency than this, the controller goes to the HALT state (Exception) and an “ISR Count Exceeded” message is logged on the application log page.

System and Task Watchdogs

Introduction

Two types of watchdog functionality are implemented for the Modicon M241 Logic Controller:

- **System Watchdogs**: These watchdogs are defined in and managed by the controller firmware. These are not configurable by the user.
- **Task Watchdogs**: These watchdogs are optional watchdogs that you can define for each task. These are managed by your application program and are configurable in EcoStruxure Machine Expert.

System Watchdogs

Three system watchdogs are defined for the Modicon M241 Logic Controller. They are managed by the controller firmware and are therefore sometimes referred to as hardware watchdogs in the EcoStruxure Machine Expert online help. When one of the system watchdogs exceeds its threshold conditions, an error is detected.

The threshold conditions for the 3 system watchdogs are defined as follows:

- If all of the tasks require more than 85% of the processor resources for more than 3 seconds, a system error is detected. The controller enters the HALT state.
- If the total execution time of the tasks with priorities between 0 and 24 reaches 100% of processor resources for more than 1 second, an application error is detected. The controller responds with an automatic reboot into the EMPTY state.
- If the lowest priority task of the system is not executed during an interval of 10 seconds, a system error is detected. The controller responds with an automatic reboot into the EMPTY state.

NOTE: System watchdogs are not configurable by the user.

Task Watchdogs

EcoStruxure Machine Expert allows you to configure an optional task watchdog for every task defined in your application program. (Task watchdogs are sometimes also referred to as software watchdogs or control timers in the EcoStruxure Machine Expert online help). When one of your defined task watchdogs reaches its threshold condition, an application error is detected and the controller enters the HALT state.

When defining a task watchdog, the following options are available:

- **Time**: This defines the allowable maximum execution time for a task. When a task takes longer than this, the controller will report a task watchdog exception.
- **Sensitivity**: The sensitivity field defines the number of task watchdog exceptions that must occur before the controller detects an application error.

To access the configuration of a task watchdog, double-click the **Task** in the **Applications tree**.

NOTE: For more information on watchdogs, refer to EcoStruxure Machine Expert Programming Guide.

Task Priorities

Task Priority Configuration

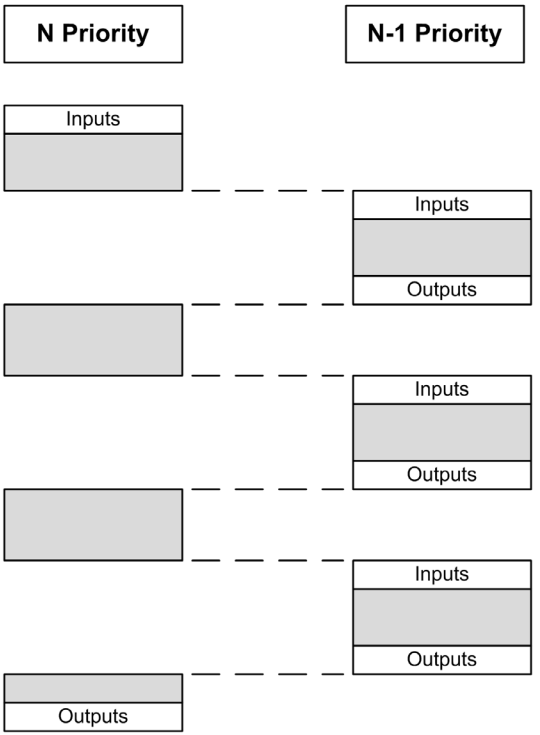
You can configure the priority of each task between 0 and 31 (0 is the highest priority, 31 is the lowest). Each task must have a unique priority. Assigning the same priority to more than one task leads to a build error.

Task Priority Suggestions

- Priority 0 to 24: Controller tasks. Assign these priorities to tasks with a high availability requirement.
- Priority 25 to 31: Background tasks. Assign these priorities to tasks with a low availability requirement.

Task Priorities of Embedded I/Os

When a task cycle starts, it can interrupt any task with lower priority (task preemption). The interrupted task resumes when the higher priority task cycle is finished.



NOTE: If the same input is used in different tasks the input image may change during the task cycle of the lower priority task.
To improve the likelihood of proper output behavior during multitasking, a build error message is displayed if outputs in the same byte are used in different tasks.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Map your inputs so that tasks do not alter the input images in an unexpected manner.

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

Task Priorities of TM2/TM3 Modules and CANopen I/Os

You can select the task that drives TM3 and CANopen physical exchanges. In the **PLC settings**, select **Bus cycle task** to define the task for the exchange. By default, the task is set to **MAST**. This definition at the controller level can be overridden by the I/O bus configuration *(see page 110)*. During the read and write phases, all physical I/Os are refreshed at the same time. TM3/TM2 and CANopen data is copied into a virtual I/O image during a physical exchanges phase, as shown in this figure:



Inputs are read from the I/O image table at the beginning of the task cycle. Outputs are written to the I/O image table at the end of the task.

NOTE: Event tasks cannot drive the TM3/TM2 bus cycle.

Default Task Configuration

Default Task Configuration

Default Task Configuration

The MAST task can be configured in Freewheeling or Cyclic mode. The MAST task is automatically created by default in Cyclic mode. Its preset priority is medium (15), its preset interval is 20 ms, and its task watchdog service is activated with a time of 100 ms and a sensitivity of 1. Refer to Task Priorities *(see page 49)* for more information on priority settings. Refer to Task Watchdogs *(see page 48)* for more information on watchdogs.

Designing an efficient application program is important in systems approaching the maximum number of tasks. In such an application, it can be difficult to keep the resource utilization below the system watchdog threshold. If priority reassignments alone are not sufficient to remain below the threshold, some lower priority tasks can be made to use fewer system resources if the SysTaskWaitSleep function, contained in the SysTask library, is added to those tasks.

NOTE: Do not delete or change the name of the MAST task. Otherwise, EcoStruxure Machine Expert detects an error when you attempt to build the application, and you are not able to download it to the controller.

Chapter 7
Controller States and Behaviors

Introduction

This chapter 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 Remanent variables and the effect of EcoStruxure Machine Expert task programming options on the behavior of your system.

What Is in This Chapter?

This chapter contains the following sections:

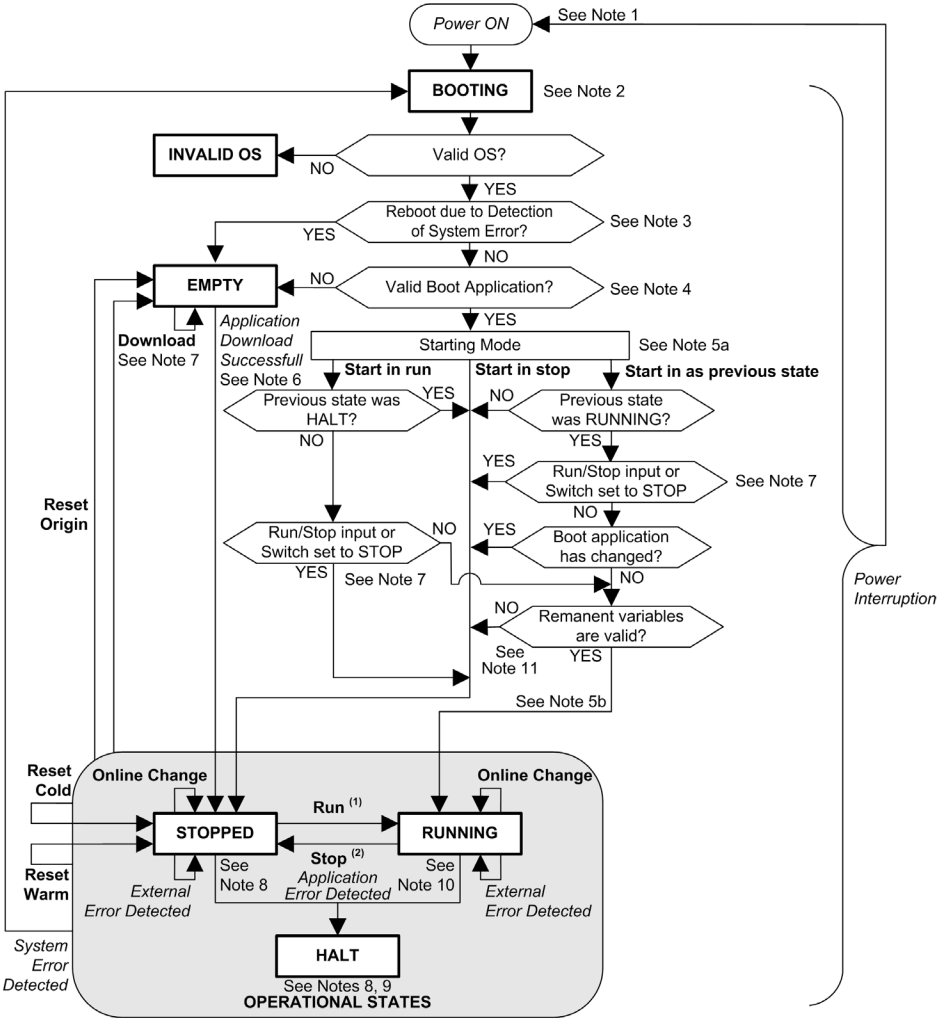
Section	Topic	Page
7.1	Controller State Diagram	54
7.2	Controller States Description	59
7.3	State Transitions and System Events	63

Section 7.1
Controller State Diagram

Controller State Diagram

Controller State Diagram

This diagram describes the controller operating mode:



Legend:
- Controller states are indicated in **ALL-CAPS BOLD**
- User and application commands are indicated in **Bold**
- System events are indicated in *Italics*
- Decisions, decision results, and general information are indicated in normal text

(1) For details on STOPPED to RUNNING state transition, refer to Run Command *(see page 67)*.

(2) For details on RUNNING to STOPPED state transition, refer to Stop Command *(see page 67)*.

Note 1

The Power Cycle (Power Interruption followed by a Power ON) deletes all output forcing settings. Refer to Controller State and Output Behavior *(see page 64)* for further details.

Note 2

The outputs will assume their hardware initialization values.

Note 3

In some cases, when a system error is detected, it will cause the controller to reboot automatically into the EMPTY state as if no Boot application were present in the Flash memory. However, the Boot application is not deleted from the Flash memory. In this case, the ERR LED (Red) flashes regularly.

Note 4

After verification of a valid Boot application the following events occur:
- The application is loaded into RAM.
- The Post Configuration *(see page 239)* file settings (if any) are applied.

During the load of the boot application, a Check context test occurs to assure that the Remanent variables are valid. If the Check context test is invalid, the boot application will load but the controller will assume STOPPED state *(see page 70)*.

Note 5a

The **Starting Mode** is set in the **PLC settings** tab of the **Controller Device Editor** *(see page 81)*.

Note 5b

When a power interruption occurs, the controller continues in the RUNNING state for at least 4 ms before shutting down. If you have configured and provide power to the Run/Stop input from the same source as the controller, the loss of power to this input will be detected immediately, and the controller will behave as if a STOP command was received. Therefore, if you provide power to the controller and the Run/Stop input from the same source, your controller will normally reboot into the STOPPED state after a power interruption when **Starting Mode** is set to **Start as previous state**.

Note 6

During a successful application download the following events occur:

- The application is loaded directly into RAM.
- By default, the Boot application is created and saved into the Flash memory.

Note 7

The default behavior after downloading an application program is for the controller to enter the STOPPED state irrespective of the Run/Stop input setting, the Run/Stop switch position or the last controller state before the download.

However, there are 2 considerations in this regard:

Online Change: An online change (partial download) initiated while the controller is in the RUNNING state returns the controller to the RUNNING state if successful and provided the Run/Stop input is configured and set to Run or Run/Stop switch is set to Run. Before using the **Login with online change** option, test the changes to your application program in a virtual or non-production environment and confirm that the controller and attached equipment assume their expected conditions in the RUNNING state.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Always verify that online changes to a RUNNING application program operate as expected before downloading them to controllers.

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

NOTE: Online changes to your program are not automatically written to the Boot application, and will be overwritten by the existing Boot application at the next reboot. If you wish your changes to persist through a reboot, manually update the Boot application by selecting **Create boot application** in the online menu (the controller must be in the STOPPED state to achieve this operation).

Multiple Download: EcoStruxure Machine Expert has a feature that allows you to perform a full application download to multiple targets on your network or fieldbus. One of the default options when you select the **Multiple Download...** command is the **Start all applications after download or online change** option, which restarts all download targets in the RUNNING state, provided their respective Run/Stop inputs are commanding the RUNNING state, but irrespective of their last controller state before the multiple download was initiated. Deselect this option if you do not want all targeted controllers to restart in the RUNNING state. In addition, before using the **Multiple Download** option, test the changes to your application program in a virtual or non-production environment and confirm that the targeted controllers and attached equipment assume their expected conditions in the RUNNING state.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION
Always verify that your application program will operate as expected for all targeted controllers and equipment before issuing the "Multiple Download..." command with the "Start all applications after download or online change" option selected.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

NOTE: During a multiple download, unlike a normal download, EcoStruxure Machine Expert does not offer the option to create a Boot application. You can manually create a Boot application at any time by selecting **Create boot application** in the **Online menu** on all targeted controllers.

Note 8

The EcoStruxure Machine Expert software platform allows many powerful options for managing task execution and output conditions while the controller is in the STOPPED or HALT states. Refer to Controller States Description *(see page 59)* for further details.

Note 9

To exit the HALT state it is necessary to issue one of the Reset commands (Reset Warm, Reset Cold, Reset Origin), download an application or cycle power.

In case of non-recoverable event (hardware watchdog or internal error), a cycle power is mandatory.

Note 10

The RUNNING state has 2 exception conditions:

- RUNNING with External Error: this exception condition is indicated by the I/O LED, which displays solid Red. You may exit this state by clearing the external error (probably changing the application configuration). No controller commands are required, but may however include the need of a power cycle of the controller. For more information, refer to I/O Configuration General Description *(see page 104)*.
- RUNNING with Breakpoint: this exception condition is indicated by the RUN LED, which displays a single flash. Refer to Controller States Description *(see page 59)* for further details.

Note 11

The boot application can be different from the application loaded. It can happen when the boot application was downloaded through SD card, FTP, or file transfer or when an online change was performed without creating the boot application.

Section 7.2
Controller States Description

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, verifying the presence of output forcing, and reviewing the controller status information via EcoStruxure Machine Expert.(1)

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

(1) The controller states can be read in the PLC_R.i_wStatus system variable of the M241 PLCSystem library *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*

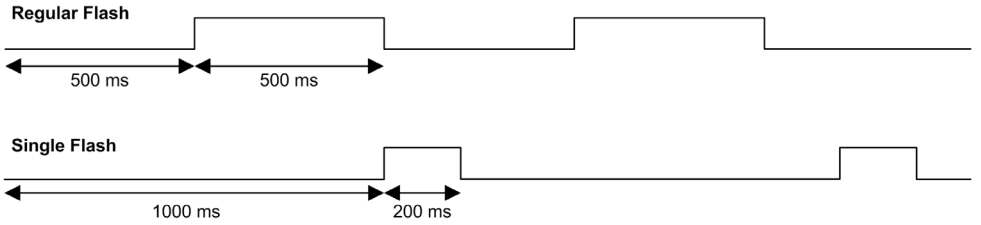
Controller States Table

The following table describes the controller states:

Controller State	Description	LED		
		RUN (Green)	ERR (Red)	I/O (Red)
BOOTING	The controller executes the boot firmware and its own internal self-tests. It then checks the checksum of the firmware and user applications.	OFF	OFF	ON
		OFF	ON	ON
		OFF	ON	OFF
INVALID_OS	There is not a valid firmware file present in the flash memory. The controller does not execute the application. Refer to the Firmware Upgrade section to restore a correct state.	OFF	Regular flash	OFF

Controller State	Description	LED		
		RUN (Green)	ERR (Red)	I/O (Red)
EMPTY	The controller has no application.	OFF	Single flash	OFF
EMPTY after a system error detected	This state is the same as the normal EMPTY state. But the application is present, and is intentionally not loaded. A next reboot (power cycle), or a new application download, will restore correct state.	OFF	Fast flash	OFF
RUNNING	The controller is executing a valid application.	ON	OFF	OFF
RUNNING with breakpoint	This state is same as the RUNNING state with the following exceptions: ● The task-processing portion of the program does not resume until the breakpoint is cleared. ● The LED indications are different. ● For more information on breakpoint management, refer to EcoStruxure Machine Expert Programming Guide.	Single flash	OFF	OFF
RUNNING with external error detected	Configuration, TM3, SD card, or other I/O error detected. When I/O LED is ON, the details about the detected error can be found in `PLC_R.i_lwSystemFault_1` and `PLC_R.i_lwSystemFault_2`. Any of the detected error conditions reported by these variables cause the I/O LED to be ON.	ON	OFF	ON
STOPPED	The controller has a valid application that is stopped. See details of the STOPPED state *(see page 61)* for an explanation of the behavior of outputs and field buses in this state.	Regular flash	OFF	OFF
STOPPED with external error detected	Configuration, TM3, SD card, or other I/O error detected.	Regular flash	OFF	ON
HALT	The controller stops executing the application because it has detected an application error	Regular flash	ON	–
Boot Application not saved	The controller has an application in memory that differs from the application in Flash memory. At next power cycle, the application will be changed by the one from Flash memory.	ON or regular flash	Single flash	OFF

This figure shows the difference between the regular flash and single flash:



Details of the STOPPED State

The following statements are true for the STOPPED state:

- The input configured as the Run/Stop input remains operational.
- The output configured as the Alarm output remains operational and goes to a value of 0.
- Ethernet, Serial (Modbus, ASCII, and so on), and USB communication services remain operational and commands written by these services can continue to affect the application, the controller state, and the memory variables.
- All outputs initially assume their configured default state (**Keep current values** or **Set all outputs to default**) or the state dictated by output forcing if used. For output used by a PTO function, the default value is ignored, in order not to generate an extra pulse. The subsequent state of the outputs depends on the value of the **Update IO while in stop** setting and on commands received from remote devices.

Task and I/O Behavior When Update IO While In Stop Is Selected

When the **Update IO while in stop** setting is selected:

- The Read Inputs operation continues normally. The physical inputs are read and then written to the %I input memory variables.
- The Task Processing operation is not executed.
- The Write Outputs operation continues. The %Q output memory variables are updated to reflect either the **Keep current values** configuration or the **Set all outputs to default** configuration, adjusted for any output forcing, and then written to the physical outputs.
  NOTE: Expert functions cease operating. For example, a counter will be stopped.

  - If **Keep current values** configuration is selected:

  PTO, PWM, FreqGen (frequency generator), and HSC reflex outputs are set to 0.

  - If **Set all outputs to default** configuration is selected:

  PTO outputs are set to 0.

  PWM, FreqGen (frequency generator) and HSC reflex outputs are set to the configured default values.

CAN Behavior When Update IO While In Stop Is Selected

The following is true for the CAN buses when the **Update IO while in stop** setting is selected:

- The CAN bus remains fully operational. Devices on the CAN bus continue to perceive the presence of a functional CAN Master.
- TPDO and RPDO continue to be exchanged.
- The optional SDO, if configured, continue to be exchanged.
- The Heartbeat and Node Guarding functions, if configured, continue to operate.
- If the **Behaviour for outputs in Stop** field is set to **Keep current values**, the TPDOs continue to be issued with the last actual values.
- If the **Behaviour for outputs in Stop** field is **Set all outputs to default** the last actual values are updated to the default values and subsequent TPDOs are issued with these default values.

Task and I/O Behavior When Update IO While In Stop Is Not Selected

When the **Update IO while in stop** setting is not selected, the controller sets the I/O to either the **Keep current values** or **Set all outputs to default** condition (as adjusted for output forcing if used). After this, the following becomes true:

- The Read Inputs operation ceases. The `%I` input memory variables are frozen at their last values.
- The Task Processing operation is not executed.
- The Write Outputs operation ceases. The `%Q` output memory variables can be updated via the Ethernet, Serial, and USB connections. However, the physical outputs are unaffected and retain the state specified by the configuration options.
  NOTE: Expert functions cease operating. For example, a counter will be stopped.

  - If **Keep current values** configuration is selected:

  PTO, PWM, FreqGen (frequency generator), and HSC reflex outputs are set to 0.

  - If **Set all outputs to default** configuration is selected:

  PTO outputs are set to 0.

  PWM, FreqGen (frequency generator), and HSC reflex outputs are set to the configured default values.

CAN Behavior When Update IO While In Stop Is Not Selected

The following is true for the CAN buses when the **Update IO while in stop** setting is not selected:

- The CAN Master ceases communications. Devices on the CAN bus assume their configured fallback states.
- TPDO and RPDO exchanges cease.
- Optional SDO, if configured, exchanges cease.
- The Heartbeat and Node Guarding functions, if configured, stop.
- The current or default values, as appropriate, are written to the TPDOs and sent once before stopping the CAN Master.

Section 7.3
State Transitions and System Events

Overview

This section begins with an explanation of the output states possible for the controller. It then presents the system commands used to transition between controller states and the system events that can also affect these states. It concludes with an explanation of the Remanent variables, and the circumstances under which different variables and data types are retained through state transitions.

What Is in This Section?

This section contains the following topics:

Topic	Page
Controller States and Output Behavior	64
Commanding State Transitions	67
Error Detection, Types, and Management	74
Remanent Variables	75

Controller States and Output Behavior

Introduction

The Modicon M241 Logic 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. For example, typical controllers define only two options for output behavior in stop: fallback to default value or keep current value.

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

- Managed by **Application Program**
- **Keep current values**
- **Set all outputs to default**
- Hardware **Initialization Values**
- Software **Initialization Values**
- **Output Forcing**

Managed by Application Program

Your application program manages outputs normally. This applies in the RUNNING and RUNNING with External Error Detected states.

NOTE: An exception to this is if the RUNNING with External Error Detected state has been provoked by a I/O expansion bus error. For more information, refer to I/O Configuration General Description *(see page 104)*.

Keep Current Values

Select this option by choosing **Controller Editor → PLC settings → Behavior for outputs in Stop → Keep current values**. To access the Controller Editor, right-click on the controller in the device tree and select **Edit Object**.

This output behavior applies in the STOPPED controller state. It also applies to CAN bus in the HALT controller state. Outputs are set to and maintained in their current state, although the details of the output behavior vary greatly depending on the setting of the **Update I/O while in stop** option and the actions commanded via configured fieldbusses. Refer to Controller States Description *(see page 59)* for more details on these variations.

NOTE: The **Keep current values** setting does not apply to PTO, PWM, FreqGen (frequency generator), and HSC reflex outputs. These outputs are always set to 0 when the controller passes to the STOPPED state, irrespective of the **Keep current values** setting.

Set All Outputs to Default

Select this option by choosing **Controller Editor → PLC settings → Behavior for outputs in Stop → Set all outputs to default**. To access the **Controller Editor**, right-click on the controller in the device tree and select **Edit Object**.

This output behavior applies:

- when the controller is going from RUN state to STOPPED state.
- if the controller is going from RUN state to HALT state.
- after application download.
- after reset warm/cold command.
- after a reboot.

It also applies to CAN bus in the HALT controller state. Outputs are set to and maintained in their current state, although the details of the output behavior vary greatly depending on the setting of the **Update I/O while in stop** option and the actions commanded via configured fieldbusses.Refer to Controller States Description *(see page 59)* for more details on these variations.

The outputs driven by a PTO, PWM, FreqGen, and HSC expert functions will not apply the default value.

Hardware Initialization Values

This output state applies in the BOOTING, EMPTY (following power cycle with no boot application or after the detection of a system error), and INVALID_OS states.

In the initialization state, analog, transistor, and relay outputs assume the following values:

- For an analog output: Z (high impedance)
- For a fast transistor output: Z (high impedance)
- For a regular transistor output: 0 Vdc
- For a 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 reset warm or cold.

The software **Initialization Values** are the initialization values of outputs images (%I, %Q, or variables mapped on %I or %Q).

By default, they are set to 0 but it is possible to map the I/O in a GVL and assign to the outputs a value different than 0.

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 EcoStruxure Machine Expert.

To do so, use the **Force values** command in the **Debug** menu.

Output forcing overrides all other commands (except write immediate) to an output irrespective of the task programming that is being executed.

When you logout of EcoStruxure Machine Expert when output forcing has been defined, you are presented with the option to retain output forcing settings. If you select this option, the output forcing continues to control the state of the selected outputs until you download an application or use one of the Reset commands.

When the option **Update I/O while in stop**, if supported by your controller, is checked (default state), the forced outputs keep the forcing value even when the logic controller is in STOP.

Output Forcing Considerations

The output you wish to force must be contained in a task that is currently being executed by the controller. Forcing outputs in unexecuted tasks, or in tasks whose execution is delayed either by priorities or events has no effect on the output. However, once the task that had been delayed is executed, the forcing takes effect at that time.

Depending on task execution, the forcing could impact your application in ways that may not be obvious to you. For example, an event task could turn on an output. Later, you may attempt to turn off that output but the event is not being triggered at the time. This would have the effect of the forcing being apparently ignored. Further, at a later time, the event could trigger the task at which point the forcing would take effect.

The outputs driven by a PTO, PWM, FreqGen, and HSC expert functions cannot be forced.

⚠ 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 EcoStruxure Machine Expert without removing the forcing.

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

Commanding State Transitions

Run Command

Effect: Commands a transition to the RUNNING controller state.

Starting Conditions: BOOTING or STOPPED state.

Methods for Issuing a Run Command:

- Run/Stop Input: If configured, command a rising edge to the Run/Stop input (assuming the Run/Stop switch is in the RUN position). Set the Run/Stop to 1 for all of the subsequent options to be effective.
  Refer to Run/Stop Input *(see page 89)* for more information.
- EcoStruxure Machine Expert Online Menu: Select the **Start** command.
- RUN command from Web Server
- By an external call via Modbus request using the PLC_W.q_wPLCControl and PLC_W.q_uiOpenPLCControl system variables of the M241 PLCSystem library.
- **Login with online change** option: An online change (partial download) initiated while the controller is in the RUNNING state returns the controller to the RUNNING state if successful.
- **Multiple Download** Command: sets the controllers into the RUNNING state if the **Start all applications after download or online change** option is selected, irrespective of whether the targeted controllers were initially in the RUNNING, STOPPED, HALT, or EMPTY state.
- The controller is restarted into the RUNNING state automatically under certain conditions.

Refer to Controller State Diagram *(see page 54)* for further details.

Stop Command

Effect: Commands a transition to the STOPPED controller state.

Starting Conditions: BOOTING, EMPTY, or RUNNING state.

Methods for Issuing a Stop Command:

- Run/Stop Input: If configured, command a value of 0 to the Run/Stop input. Refer to Run/Stop Input *(see page 89)* for more information.
- EcoStruxure Machine Expert Online Menu: Select the **Stop** command.
- STOP command from WebServer
- By an internal call by the application or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M241 PLCSystem library.
- **Login with online change** option: An online change (partial download) initiated while the controller is in the STOPPED state returns the controller to the STOPPED state if successful.
- **Download** Command: implicitly sets the controller into the STOPPED state.
- **Multiple Download** Command: sets the controllers into the STOPPED state if the **Start all applications after download or online change** option is not selected, irrespective of whether the targeted controllers were initially in the RUNNING, STOPPED, HALT, or EMPTY state.

- REBOOT by Script: The file transfer script on an SD card can issue a REBOOT as its final command. The controller is rebooted into the STOPPED state provided the other conditions of the boot sequence allow this to occur. Refer to Reboot *(see page 70)* for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram *(see page 54)* for further details.

Reset Warm

Effect: Resets all variables, except for the remanent variables, to their default values. Places the controller into the STOPPED state.

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Warm Command:

- EcoStruxure Machine Expert Online Menu: Select the **Reset warm** command.
- By an internal call by the application or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M241 PLCSystem library.

Effects of the Reset Warm Command:

1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for errors are reset.
4. The values of the retain variables are maintained.
5. The values of the retain-persistent variables are maintained.
6. All non-located and non-remanent variables are reset to their initialization values.
7. The values of the first 1000 `%MW` registers are maintained.
8. The values of `%MW1000` to `%MW59999` registers are reset to 0.
9. All fieldbus communications are stopped and then restarted after the reset is complete.
10. All inputs are reset to their initialization values. All outputs are reset to their software initialization values or their default values if no software initialization values are defined.
11. The Post Configuration file is read *(see page 239)*.

For details on variables, refer to Remanent Variables *(see page 75)*.

Reset Cold

Effect: Resets all variables, except for the retain-persistent type of remanent variables, to their initialization values. Places the controller into the STOPPED state.

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Cold Command:

- EcoStruxure Machine Expert Online Menu: Select the **Reset cold** command.
- By an internal call by the application or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M241 PLCSystem library.

Effects of the Reset Cold Command:

1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for errors are reset.
4. The values of the retain variables are reset to their initialization value.
5. The values of the retain-persistent variables are maintained.
6. All non-located and non-remanent variables are reset to their initialization values.
7. The values of the first 1000 %MW registers are maintained.
8. The values of %MW1000 to %MW59999 registers are reset to 0.
9. All fieldbus communications are stopped and then restarted after the reset is complete.
10. All inputs are reset to their initialization values. All outputs are reset to their software initialization values or their default values if no software initialization values are defined.
11. The Post Configuration file is read *(see page 239)*.

For details on variables, refer to Remanent Variables *(see page 75)*.

Reset Origin

Effect: Resets all variables, including the remanent variables, to their initialization values. Erases all user files on the controller. Places the controller into the EMPTY state.

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Origin Command:

- EcoStruxure Machine Expert Online Menu: Select the **Reset origin** command.

Effects of the Reset Origin Command:

1. The application stops.
2. Forcing is erased.
3. The web visu files are erased.
4. All user files (Boot application, data logging, Post Configuration) are erased.
5. Diagnostic indications for errors are reset.
6. The values of the retain variables are reset.
7. The values of the retain-persistent variables are reset.
8. All non-located and non-remanent variables are reset.
9. The values of the first 1000 %MW registers are reset to 0.
10. The values of %MW1000 to %MW59999 registers are reset to 0.
11. All fieldbus communications are stopped.
12. Embedded Expert I/O are reset to their previous user-configured default values.
13. All other inputs are reset to their initialization values.
    All other outputs are reset to their hardware initialization values.

For details on variables, refer to Remanent Variables *(see page 75)*.

Reboot

Effect: Commands a reboot of the controller.

Starting Conditions: Any state.

Methods for Issuing the Reboot Command:

- Power cycle
- REBOOT by Script *(see page 253)*

Effects of the Reboot:

1. The state of the controller depends on a number of conditions:
   a. The controller state is RUNNING if:
   The Reboot was provoked by a power cycle and:
   - the **Starting Mode** is set to **Start in run**, and if the Run/Stop input is not configured, and if the controller was not in HALT state before the power cycle, and if the remanent variables are valid.
   - the **Starting Mode** is set to **Start in run**, and if the Run/Stop input is configured and set to RUN, and if the controller was not in HALT state before the power cycle, and if the remanent variables are valid.
   - the **Starting Mode** is set to **Start as previous state**, and Controller state was RUNNING before the power cycle, and if the Run/Stop input is set to not configured and the boot application has not changed and the remanent variables are valid.
   - the **Starting Mode** is set to **Start as previous state**, and Controller state was RUNNING before the power cycle, and if the Run/Stop input is configured and is set to RUN.
   The Reboot was provoked by a script and:
   - the **Starting Mode** is set to **Start in run**, and if the Run/Stop input or switch is configured and set to RUN, and if the controller was not in HALT state before the power cycle, and if the remanent variables are valid.
   b. The controller state is STOPPED if:
   The Reboot was provoked by a power cycle and:
   - the **Starting Mode** is set to **Start in stop**.
   - the **Starting Mode** is set to **Start as previous state** and the controller state was not RUNNING before the power cycle.
   - the **Starting Mode** is set to **Start as previous state** and the controller state was RUNNING before the power cycle, and if the Run/Stop input is set to not configured, and if the boot application has changed.
   - the **Starting Mode** is set to **Start as previous state** and the controller state was RUNNING before the power cycle, and if the Run/Stop input is set to not configured, and if the boot application has not changed, and if the remanent variables are not valid.
   - the **Starting Mode** is set to **Start as previous state** and the controller state was RUNNING before the power cycle, and if the Run/Stop input is configured and is set to STOP.
   - the **Starting Mode** is set to **Start in run** and if the controller state was HALT before the power cycle.
   - the **Starting Mode** is set to **Start in run**, and if the controller state was not HALT before the power cycle, and if the Run/Stop input is configured and is set to STOP.

- the **Starting Mode** is set to **Start as previous state** and if the Run/Stop input or switch is configured and set to RUN, and if the controller was not in HALT state before the power cycle.
- the **Starting Mode** is set to **Start as previous state** and if the Run/Stop input or switch is not configured, and if the controller was not in HALT state before the power cycle.

c. The controller state is EMPTY if:
- There is no boot application or the boot application is invalid, or
- The reboot was provoked by specific System Errors.

d. The controller state is INVALID_OS if there is no valid firmware.

2. Forcing is maintained if the boot application is loaded successfully. If not, forcing is erased.
3. Diagnostic indications for errors are reset.
4. The values of the retain variables are restored if saved context is valid.
5. The values of the retain-persistent variables are restored if saved context is valid.
6. All non-located and non-remanent variables are reset to their initialization values.
7. The values of the first 1000 `%MW` registers are restored if saved context is valid.
8. The values of `%MW1000` to `%MW59999` registers are reset to 0.
9. All fieldbus communications are stopped and restarted after the boot application is loaded successfully.
10. All inputs are reset to their initialization values. All outputs are reset to their hardware initialization values and then to their software initialization values or their default values if no software initialization values are defined.
11. The Post Configuration file is read *(see page 239)*.
12. The controller file system is initialized and its resources (sockets, file handles, and so on) are deallocated.
    The file system employed by the controller needs to be periodically re-established by a power cycle of the controller. If you do not perform regular maintenance of your machine, or if you are using an Uninterruptible Power Supply (UPS), you must force a power cycle (removal and reapplication of power) to the controller at least once a year.

NOTICE
DEGRADATION OF PERFORMANCE
Reboot your controller at least once a year by removing and then reapplying power.
Failure to follow these instructions can result in equipment damage.

For details on variables, refer to Remanent Variables *(see page 75)*.

NOTE: The Check context test concludes that the context is valid when the application and the remanent variables are the same as defined in the Boot application.

NOTE: If you provide power to the Run/Stop input from the same source as the controller, the loss of power to this input is detected immediately, and the controller behaves as if a STOP command was received. Therefore, if you provide power to the controller and the Run/Stop input from the same source, your controller reboots normally into the STOPPED state after a power interruption when **Starting Mode** is set to **Start as previous state**.

NOTE: If you make an online change to your application program while your controller is in the RUNNING or STOPPED state but do not manually update your Boot application, the controller detects a difference in context at the next reboot, the remanent variables are reset as per a Reset cold command, and the controller enters the STOPPED state.

Download Application

Effect: Loads your application executable into the RAM memory. Optionally, creates a Boot application in the Flash memory.

Starting Conditions: RUNNING, STOPPED, HALT, and EMPTY states.

Methods for Issuing the Download Application Command:

- EcoStruxure Machine Expert:
  2 options exist for downloading a full application:
  - Download command.
  - Multiple Download command.

  For important information on the application download commands, refer to Controller State Diagram.
- FTP: Load Boot application file to the Flash memory using FTP. The updated file is applied at the next reboot.
- SD card: Load Boot application file using an SD card in the controller SD card slot. The updated file is applied at the next reboot. Refer to File Transfer with SD Card *(see page 259)* for further details.

Effects of the EcoStruxure Machine Expert Download Command:

1. The existing application stops and then is erased.
2. If valid, the new application is loaded and the controller assumes a STOPPED state.
3. Forcing is erased.
4. Diagnostic indications for errors are reset.
5. The values of the retain variables are reset to their initialization values.
6. The values of any existing retain-persistent variables are maintained.
7. All non-located and non-remanent variables are reset to their initialization values.
8. The values of the first 1000 `%MW` registers are maintained.
9. The values of `%MW1000` to `%MW59999` registers are reset to 0.
10. All fieldbus communications are stopped and then any configured fieldbus of the new application is started after the download is complete.
11. Embedded Expert I/O are reset to their previous user-configured default values and then set to the new user-configured default values after the download is complete.
12. All inputs are reset to their initialization values. All outputs are reset to their hardware initialization values and then to their software initialization values or their default values if no software initialization values are defined, after the download is complete.
13. The Post Configuration file is read *(see page 239)*.

For details on variables, refer to Remanent Variables *(see page 75)*.

Effects of the FTP or SD Card Download Command:

There are no effects until the next reboot. At the next reboot, the effects are the same as a reboot with an invalid context. Refer to Reboot *(see page 70)*.

Error Detection, Types, and Management

Error Management

The controller detects and manages three types of errors:

- External errors
- Application errors
- System errors

This table describes the types of errors that may be detected:

Type of Error Detected	Description	Resulting Controller State
External Error	External errors are detected by the system while RUNNING or STOPPED but do not affect the ongoing controller state. An external error is detected in the following cases: ● A connected device reports an error to the controller. ● The controller detects an error with an external device, for example, when the external device is communicating but not properly configured for use with the controller. ● The controller detects an error with an output. ● The controller detects a communication interruption with a device. ● The controller is configured for an expansion module that is not present or not detected, and has not otherwise been declared as an optional module(1). ● The boot application in Flash memory is not the same as the one in RAM.	RUNNING with External Error Detected Or STOPPED with External Error Detected
Application Error	An application error is detected when improper programming is encountered or when a task watchdog threshold is exceeded.	HALT
System Error	A system error is detected when the controller enters a condition that cannot be managed during runtime. Most such conditions result from firmware or hardware exceptions, but there are some cases when incorrect programming can result in the detection of a system error, for example, when attempting to write to memory that was reserved during runtime, or when a system watchdog occurs. **NOTE:** There are some system errors that can be managed by runtime and are therefore treated like application errors.	BOOTING → EMPTY

(1) Expansion modules may appear to be absent for any number of reasons, even if the absent I/O module is physically present on the bus. For more information, refer to I/O Configuration General Description *(see page 104)*.

NOTE: Refer to the M241 PLCSystem library Guide for more detailed information on diagnostics.

Remanent Variables

Overview

Remanent variables can either be reinitialized or retain their values in the event of power outages, reboots, resets, and application program downloads. There are multiple types of remanent variables, declared individually as retain or persistent, or in combination as retain-persistent.

NOTE: For this controller, variables declared as persistent behave in the same way as variables declared as retain-persistent.

This table describes the behavior of remanent variables in each case:

Action	VAR	VAR RETAIN	VAR GLOBAL RETAIN PERSISTENT
Online change to application program	X	X	X
Online change modifying the boot application (1)	–	X	X
Stop	X	X	X
Power cycle	–	X	X
Reset warm	–	X(2)	X
Reset cold	–	–	X
Reset origin	–	–	–
Download of application program(3)	–	–	X

X The value is maintained.
– The value is reinitialized.
(1) Retain variable values are maintained if an online change modifies only the code part of the boot application (for example, `a:=a+1; => a:=a+2;`). In all other cases, retain variables are reinitialized.
(2) For more details on VAR RETAIN, refer to Effects of the Reset warm Command *(see page 68)*.
(3) If the application is downloaded using an SD card, any existing retain-persistent variables used by the application are reinitialized. If the application is downloaded using EcoStruxure Machine Expert, however, existing retain-persistent variables maintain their values. In both cases, if the downloaded application contains the same retain-persistent variables as the existing application, the existing retain variables maintain their values.

NOTE: The first 1000 `%MW` are automatically retained and persistent if no variable is associated to them. Their values are kept after a reboot / Reset warm / Reset cold. The other `%MW` are managed as VAR.

For example, if you have in your program:

```
VAR myVariable AT %MW0 : WORD; END_VAR
```

`%MW0` behaves like `myVariable` (not retained and not persistent).

Adding Retain-Persistent Variables

Declare retain-persistent (**VAR GLOBAL PERSISTENT RETAIN**) symbols in the **PersistentVars** window:

Step	Action
1	In the **Applications tree**, select the **Application** node.
2	Click the right mouse button.
3	Choose **Add Objects → Persistent variables**
4	Click **Add**. **Result:** The **PersistentVars** window is displayed.

Retain and Persistent Variables: Performance Impact

Retain or retain-persistent variables are located in a dedicated non-volatile memory. Each time these variables are accessed during Programming Organization Unit (POU) execution, the non-volatile memory is accessed. The access time of these variables is slower than the access time of regular variables, which can impact performance. This is an important fact to take into account when writing performance-sensitive POUs.

Chapter 8
Controller Device Editor

Introduction

This chapter describes how to configure the controller.

What Is in This Chapter?

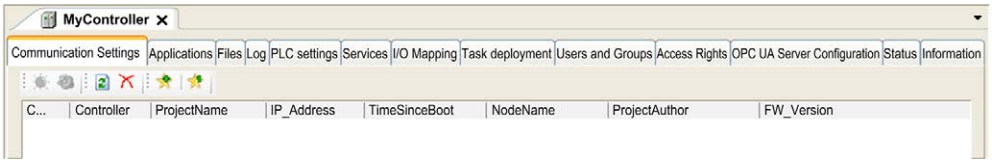
This chapter contains the following topics:

Topic	Page
Controller Parameters	78
Communication Settings	80
PLC Settings	81
Services	83
Users Rights	85

Controller Parameters

Controller Parameters

To open the device editor, double-click **MyController** in the **Devices tree**:



Tabs Description

Tab	Description	Restriction
Communication Settings *(see page 80)*	Manages the connection between the PC and the controller: ● helping you find a controller in a network, ● presenting the list of available controllers, so you can connect to the selected controller and manage the application in the controller, ● helping you physically identify the controller from the device editor, ● helping you change the communication settings of the controller. The controller list is detected through NetManage or through the Active Path based on the communication settings. To access the **Communication settings**, click **Project → Project Settings...** in the menu bar. For more information, refer to the EcoStruxure Machine Expert Programming Guide (*Communication Settings*).	Online mode only
Applications	Presents the application running on the controller and allows removing the application from the controller.	Online mode only
Files *(see page 33)*	File management between the PC and the controller. Only one logic controller disk at a time can be seen through this tab. When an SD card is inserted, this file displays the content of the SD card. Otherwise, this tab displays the content of the */usr* directory of the internal flash memory of the controller.	Online mode only
Log	View the controller log file.	Online mode only
PLC settings *(see page 81)*	Configuration of: ● application name ● I/O behavior in stop ● bus cycle options	–
Services *(see page 83)*	Lets you configure the online services of the controller (RTC, device identification).	Online mode only

Tab	Description	Restriction
I/O Mapping	Mapping of the input and output channels of an I/O device on project (application) variables.	–
Task deployment	Displays a list of I/Os and their assignments to tasks.	After compilation only
Users and Groups	The **Users and Groups** tab is provided for devices supporting online user management. It allows setting up users and access-rights groups and assigning them access rights to control the access on EcoStruxure Machine Expert projects and devices in online mode. For more details, refer to the EcoStruxure Machine Expert Programming Guide.	–
OPC UA Server Configuration	Displays the OPC UA Server Configuration *(see page 231)* window.	–
Access Rights	The **Access Rights** tab is provided for devices supporting online user management. It serves to grant or deny the currently defined user groups certain permissions, thus defining the access rights for users on files or objects (for example, an application) on the controller during runtime. For more details, refer to the EcoStruxure Machine Expert Programming Guide.	–
Status	No information delivered.	–
Information	Displays general information about the device (name, description, provider, version, image).	–

Communication Settings

Introduction

This tab allows you to manage the connection from the PC to the controller:

- Helping you find a controller in a network.
- Presenting the list of controllers, so you can connect to the selected controller and manage the application inside the controller.
- Helping you physically identify the controller from the device editor.
- Helping you change the communication settings of the controller.

You can change the display mode of the **Communication Settings** tab:

- **Simple mode**. Refer to EcoStruxure Machine Expert, Programming Guide.
- **Classic mode**. Refer to EcoStruxure Machine Expert, Programming Guide.
- **Controller selection mode**. Refer to EcoStruxure Machine Expert, Programming Guide.

Edit Communication Settings

In **Controller selection mode**, the **Edit communication settings** window lets you change the Ethernet communication settings. To do so, click **Communication Settings** tab. The list of controllers available in the network appears. Select and right-click the required row and click **Edit communication settings ...** in the context menu.

You can configure the Ethernet settings in the **Edit communication settings** window in 2 ways:

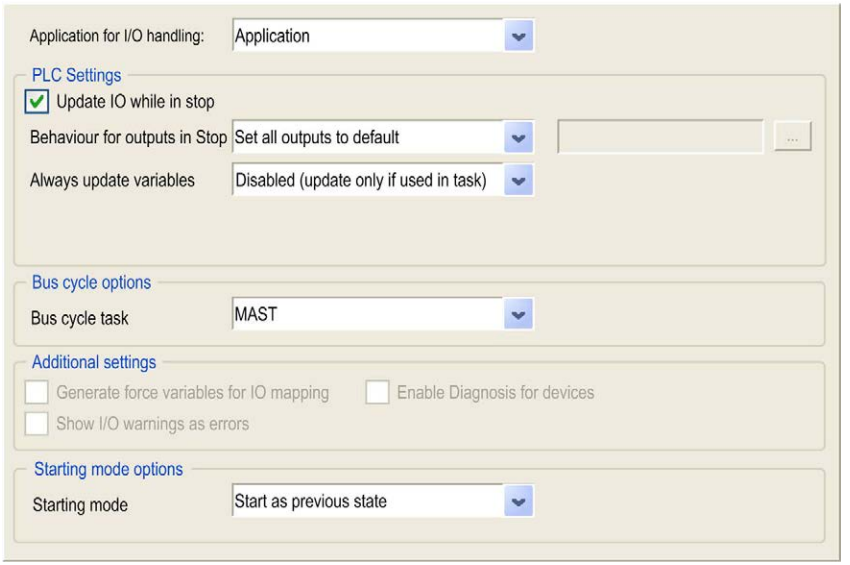
- Without the **Save settings permanently** option:
  Configure the communication parameters and click **OK**. These settings are immediately taken into account and are not kept if the controller is reset. For the next resets, the communication parameters configured into the application are taken into account.
- With the **Save settings permanently** option:
  You can also activate the **Save settings permanently** option before you click **OK**. Once this option is activated, the Ethernet parameters configured here are always taken into account on reset instead of the Ethernet parameters configured into the EcoStruxure Machine Expert application.

For more information on the **Communication Settings** view of the device editor, refer to the EcoStruxure Machine Expert Programming Guide.

PLC Settings

Overview

The figure below presents the **PLC Settings** tab:



Element		Description
Application for I/O handling		By default, set to Application because there is only one application in the controller.
PLC settings	**Update IO while in stop**	If this option is activated (default), the values of the input and output channels get also updated when the controller is stopped.
	Behavior for outputs in Stop	From the selection list, choose one of the following options to configure how the values at the output channels should be handled in case of controller stop: • **Keep current values** • **Set all outputs to default**
	Always update variables	By default, set to **Enabled 1 (use bus cycle task if not used in task)** and not editable.
Bus cycle options	**Bus cycle task**	This configuration setting is the parent for all **Bus cycle task** parameters used in the application device tree. Some devices with cyclic calls, such as a **CANopen manager**, can be attached to a specific task. In the device, when this setting is set to **Use parent bus cycle setting**, the setting set for the controller is used. The selection list offers all tasks currently defined in the active application. The default setting is the MAST task. NOTE: **<unspecified>** means that the task is in "slowest cyclic task" mode.
Additional settings	**Generate force variables for IO mapping**	Not used.
	Enable Diagnosis for devices	Not used.
	Show I/O warnings as errors	Not used.
Starting mode Options	**Starting mode**	This option defines the starting mode on a power-on. For further information, refer to State behavior diagram *(see page 54)*. Select with this option one of these starting modes: • **Start as previous state** • **Start in stop** • **Start in run**

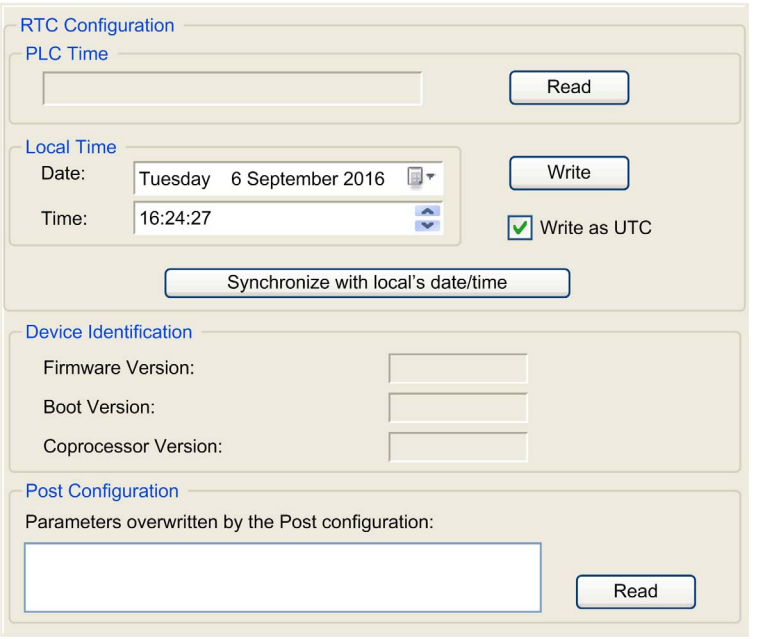
Services

Services Tab

The **Services** tab is divided in three parts:

- RTC Configuration
- Device Identification
- Post Configuration

The figure below shows the **Services** tab:



NOTE: To have controller information, you must be connected to the controller.

Element		Description
RTC Configuration	**PLC Time**	Displays the date and time read from the controller when the **Read** button is clicked, with no conversion applied. This read-only field is initially empty. If **Write as UTC** is selected, **PLC Time** is in UTC.
	Read	Reads the date and time saved on the controller and displays the values in the **PLC Time** field.
	Local Time	Lets you define a date and a time that are sent to the controller when the **Write** button is clicked. If necessary, modify the default values before clicking the **Write** button. A message box informs you about the result of the command. The date and time fields are initially filled with the current PC settings.
	Write	Writes the date and time defined in the **Local time** field to the logic controller. A message box informs you of the result of the command. Select the **Write as UTC** checkbox before running this command to write the values in UTC format.
	Synchronize with local date/time	Lets you directly send the PC settings. A message box informs you of the result of the command. Select **Write as UTC** before running this command to use UTC format. Use UTC time when using secure communication.
Device Identification		Displays the **Firmware Version**, the **Boot Version**, and the **Coprocessor Version** of the selected controller, if connected.
Post Configuration		Displays the application parameters overwritten by the Post configuration *(see page 239)*.

Users Rights

Introduction

Users and Groups and **Access Right** tabs *(see page 78)* allows to manage user accounts, user access rights groups and the associated access rights, to control the access on projects. For more informations, refer to the EcoStruxure Machine Expert Programming Guide.

Login and passwords

Login and password are set by default. They must be activated and can be reset as origin.

This table describes how to log in:

Server/feature	First connection Login / Password	User rights Login / Password	Connection after reset to default Login / Password
EcoStruxure Machine Expert	Administrator / Administrator	Administrator / configured password	Administrator / Administrator
HTTP	No login possible	Administrator / configured password	No login possible
FTP	No login possible	Administrator / configured password	No login possible
OPC-UA	No login possible	Administrator / configured password	No login possible
Change Device Name feature	No login possible	Administrator / configured password	No login possible

⚠ WARNING

UNAUTHORIZED DATA AND/OR APPLICATION ACCESS

- Secure access to the FTP/Web/OPC-UA server(s) using User Rights.
- If you disable User Rights, disable the server(s) to prevent any unwanted or unauthorized access to your application and/or data.

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

NOTE: Anonymous login can be restored by removing the user rights in the **User Management** page *(see page 142)* of the web server.

NOTE: Password should not contain any of the following special characters: “ !”#$%&’()*+,-./:;<=>?@[\]^_‘{|}~”.

Default users and groups

One user and two groups are set by default:

- User: **Administrator**
- Groups: **Administrator** and **Everyone**

Access Rights

You can give **Access Rights** to groups.

You can allow the following operations through the access rights:

- **VIEW**
- **MODIFY**
- **EXECUTE**
- **ADD_REMOVE**

Troubleshooting

The only way to gain access to a controller that has user access-rights enabled and for which you do not have the password(s) is by performing an Update Firmware operation. This clearing of User Rights can only be accomplished by using a SD card or USB key (depending on the support of your particular controller) to update the controller firmware. In addition, you may clear the User Rights in the controller by running a script (for more information, refer to EcoStruxure Machine Expert Programming Guide) . This effectively removes the existing application from the controller memory, but restores the ability to access the Controller.

Chapter 9
Embedded Inputs and Outputs Configuration

Embedded I/Os Configuration

Overview

The embedded I/O function allows configuration of the controller inputs and outputs.

The M241 logic controller provides:

I/O Type	24 I/O References	40 I/O References
	TM241•24•	TM241•40•
Fast inputs	8	8
Regular inputs	6	16
Fast outputs	4	4
Regular outputs	6	12

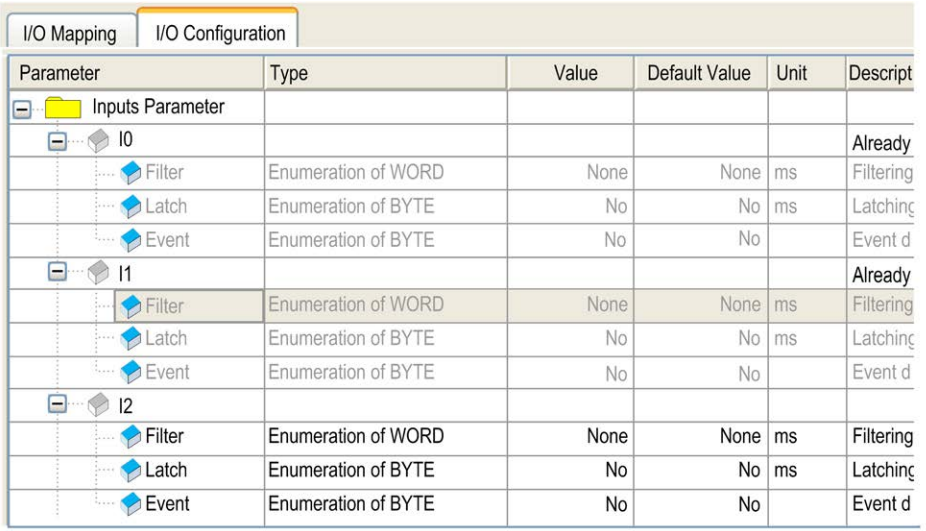
Accessing the I/O Configuration Window

Follow these steps to access the I/O configuration window:

Step	Description
1	Double-click **DI** (digital inputs) or **DQ** (digital outputs) in the **Devices tree**. Refer to Devices tree *(see page 22)*.
2	Select the **I/O Configuration** tab.

Configuration of Digital Inputs

This figure shows the **I/O Configuration** tab for digital inputs:



Parameter	Type	Value	Default Value	Unit	Descript
Inputs Parameter					
I0					Already
Filter	Enumeration of WORD	None	None	ms	Filtering
Latch	Enumeration of BYTE	No	No	ms	Latching
Event	Enumeration of BYTE	No	No		Event d
I1					Already
Filter	Enumeration of WORD	None	None	ms	Filtering
Latch	Enumeration of BYTE	No	No	ms	Latching
Event	Enumeration of BYTE	No	No		Event d
I2					
Filter	Enumeration of WORD	None	None	ms	Filtering
Latch	Enumeration of BYTE	No	No	ms	Latching
Event	Enumeration of BYTE	No	No		Event d

NOTE: For more information on the **I/O Mapping** tab, refer to the EcoStruxure Machine Expert Programming Guide.

Digital Input Configuration Parameters

For each digital input, you can configure the following parameters:

Parameter	Value	Description	Constraint
Filter	None 1 ms 4 ms* 12 ms	Reduces the effect of noise on a controller input.	Available if **Latch** and **Event** are disabled. In the other cases, this parameter is disabled and its value is **None**.
Latch	No* Yes	Allows incoming pulses with amplitude widths shorter than the controller scan time to be captured and recorded.	This parameter is only available for the fast inputs I0 to I7. Available if **Event** disabled AND **Filter** disabled. Use latch inputs in MAST task only.
Event	No* Rising edge Falling edge Both edges	Event detection	This parameter is only available for the fast inputs I0 to I7. Available if **Latch** disabled AND **Filter** disabled. When **Both edges** is selected, and the input state is TRUE before the controller is powered on, the first falling edge is ignored.
* Parameter default value			

Parameter	Value	Description	Constraint
Bounce	0.000 ms 0.001 ms 0.002 ms* 0.005 ms 0.010 ms 0.05 ms 0.1 ms 0.5 ms 1 ms 5 ms	Reduces the effect of bounce on a controller input.	Available if **Latch** is enabled or **Event** is enabled. In the other cases, this parameter is disabled and its value is 0.002.
Run/Stop Input	None I0...I13 (TM241•24• references) I0...I23 (TM241•40• references)	The Run/Stop input can be used to run or stop the controller application.	Select one of the inputs to use as the Run/Stop Input.
* Parameter default value			

NOTE: The selection is grey and inactive if the parameter is unavailable.

Run/Stop Input

This table presents the different states:

Input states	Result
State 0	Stops the controller and ignores external Run commands.
A rising edge	From the STOPPED state, initiate a start-up of an application in RUNNING state, if no conflict with Run/Stop switch position.
State 1	The application can be controlled by: • EcoStruxure Machine Expert (Run/Stop) • A hardware Run/Stop switch • Application (Controller command) • Network command (Run/Stop command) Run/Stop command is available through the Web Server command.

NOTE: Run/Stop input is managed even if the option **Update I/O while in stop** is not selected in Controller Device Editor (**PLC settings** tab) *(see page 81)*.
Inputs assigned to configured expert functions cannot be configured as Run/Stop inputs.

For further details about controller states and states transitions, refer to Controller State Diagram *(see page 54)*.

⚠ WARNING

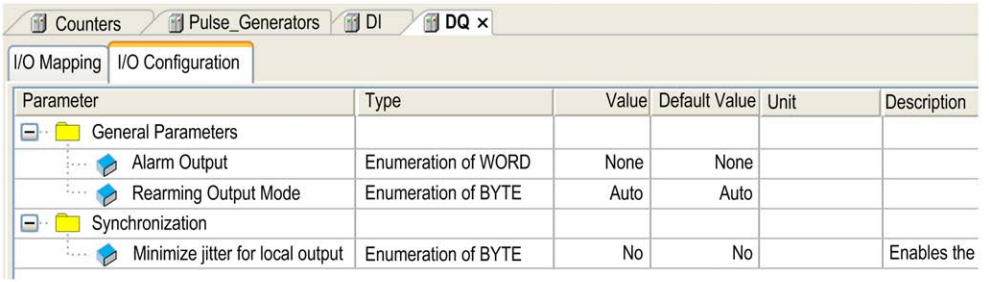
UNINTENDED MACHINE OR PROCESS START-UP

- Verify the state of security of your machine or process environment before applying power to the Run/Stop input.
- Use the Run/Stop input to help prevent the unintentional start-up from a remote location.

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

Configuration of Digital Outputs

This figure shows the **I/O Configuration** tab for digital outputs:



NOTE: For more information on the **I/O Mapping** tab, refer to the EcoStruxure Machine Expert Programming Guide.

Digital Output Configuration Parameters

This table presents the function of the different parameters:

Parameter	Function
General Parameters	
Alarm Output	Select an output to be used as alarm output *(see page 91)*.
Rearming Output Mode	Select the rearming output mode *(see page 92)*.
Synchronization	
Minimize jitter for local Output	Select this option to minimize jitter on local outputs *(see page 92)*.

NOTE: The selection is grey and inactive if the parameter is unavailable.

Alarm Output

This output is set to logical 1 when the controller is in the RUNNING state and the application program is not stopped at a breakpoint.

The alarm output is set to 0 when a task is stopped at a breakpoint to signal that the controller has stopped executing the application.

The alarm output is set to 0 when a shortcut is detected.

NOTE: Outputs assigned to configured expert functions cannot be configured as the alarm output.

Rearming Output Mode

Fast outputs of the Modicon M241 Logic Controller use push/pull technology. In case of detected error (short-circuit or over temperature), the output is put in tri-state and the condition is signaled by status bit and PLC_R.i_wLocalIOStatus.

Two behaviors are possible:

- **Automatic rearming**: as soon as the detected error is corrected, the output is set again according to the current value assigned to it and the diagnostic value is reset.
- **Manual rearming**: when an error is detected, the status is memorized and the output is forced to tri-state until user manually clears the status (see I/O mapping channel).

In the case of a short-circuit or current overload, the common group of outputs automatically enters into thermal protection mode (all outputs in the group are set to 0), and are then periodically rearmed (each second) to test the connection state. However, you must be aware of the effect of this rearming on the machine or process being controlled.

⚠ WARNING

UNINTENDED MACHINE START-UP

Inhibit the automatic rearming of outputs if this feature is an undesirable behavior for your machine or process.

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

Minimize Jitter for Local Output

This option allows the embedded I/Os to be read or set at predictable time intervals, regardless of the task duration. Minimizes jitter on outputs by delaying the write to the physical outputs until the read input operation of the next bus cycle task starts. The end time of a task is often less easy to predict than the start time.

Normal scheduling of input/ouput phases is:



When the **Minimize Jitter for Local Output** option is selected, the scheduling of the IN and OUT phases becomes:



Chapter 10
Expert Functions Configuration

Overview

This chapter describes the expert functions of the M241.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Expert Functions Overview	94
Counting Function	97
Pulse Generators Embedded Function	99

Expert Functions Overview

Introduction

The inputs and outputs available on the M241 logic controller can be connected to expert functions.

The M241 logic controller supports the following expert functions:

Functions		Description
Counters	HSC Simple	The HSC functions can execute fast counts of pulses from sensors, switches, etc. that are connected to the fast or regular inputs. HSC functions connected to regular inputs operate at a maximum frequency of 1 kHz. For more information about the HSC functions, refer to High Speed Counter types *(see Modicon M241 Logic Controller, High Speed Counting, HSC Library Guide)*.
	HSC Main Single Phase	
	HSC Main Dual Phase	
	Frequency Meter	
	Period Meter	
Pulse Generators	PTO *(see Modicon M241 Logic Controller, PTOPWM, Library Guide)*	The PTO function provides 2 pulse train output channels to control 2 independent linear single-axis stepper or servo drives in open loop mode. The PTO function connected to regular transistor outputs operates at a maximum frequency of 1 kHz.
	PWM *(see Modicon M241 Logic Controller, PTOPWM, Library Guide)*	The PWM function generates a square wave signal on dedicated output channels with a variable duty cycle. The PWM function connected to regular transistor outputs operates at a maximum frequency of 1 kHz.
	Frequency Generator *(see Modicon M241 Logic Controller, PTOPWM, Library Guide)*	The frequency generator function generates a square wave signal on dedicated output channels with a fixed duty cycle (50%). The Frequency Generator function connected to regular transistor outputs operates at a maximum frequency of 1 kHz.

As of the release of EcoStruxure Machine Expert, any regular I/O not already in use can be configured for use by any of the expert function types, in the same way as fast I/Os.

NOTE:

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

For more details, refer to Embedded Functions Configuration *(see page 93)*.

Maximum Number of Expert Functions

The maximum number of expert functions that can be configured depends on:

1. The logic controller reference.
2. The expert function types and number of optional functions *(see Modicon M241 Logic Controller, High Speed Counting, HSC Library Guide)* configured. Refer to Embedded Expert I/O Assignment *(see Modicon M241 Logic Controller, High Speed Counting, HSC Library Guide)*.
3. The number of I/Os that are available.

Maximum number of expert functions by logic controller reference:

Expert Function Type		24 I/O References (TM241•24•)	40 I/O References (TM241•40•)
Total number of HSC functions		14	16
HSC	Simple	14	16
	Main Single Phase	4	
	Main Dual Phase		
	Frequency Meter(1)		
	Period Meter		
PTO			
PWM			
FreqGen			
(1) When the maximum number is configured, only 12 additional HSC Simple functions can be added.			

The maximum number of expert functions possible may be further limited by the number of I/Os used by each expert function.

Example configurations:

- 4 PTO(2) + 14 HSC Simple on 24 I/O controller references
- 4 FreqGen(2) + 16 HSC Simple on 40 I/O controller references
- 4 HSC Main Single Phase + 10 HSC Simple on 24 I/O controller references
- 4 HSC Main Dual Phase + 8 HSC Simple on 40 I/O controller references
- 2 PTO(2) + 2 HSC Main Single Phase + 14 HSC Simple on 40 I/O controller references

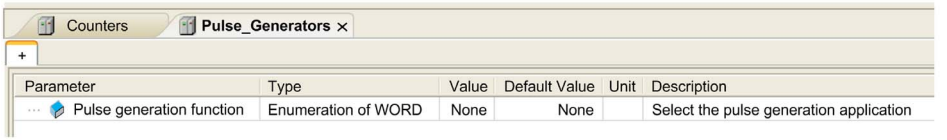
(2) With no optional I/O configured

The performance of the expert function is limited by the I/Os used:

- HSC with fast inputs: 100 kHz/200 kHz
- HSC with regular inputs: 1 kHz

Configuring an Expert Function

To configure an expert function, proceed as follows:

Step	Description
1	Double-click the **Counters** or **Pulse_Generators** node in the **Devices Tree**. **Result:** The **Counters** or **Pulse_Generators** configuration window appears: 
2	Double-click **None** in the **Value** column and choose the expert function type to assign. **Result:** The default configuration of the expert function appears when you click anywhere in the configuration window.
3	Configure the expert function parameters, as described in the following chapters.
4	To configure an additional expert function, click the **+** tab. **NOTE:** If the maximum number of expert functions is already configured, a message appears at the bottom of the configuration window informing you that you can now add only HSC Simple functions.

Regular I/O Configured as Expert Function

When regular I/Os are configured as expert functions, note the following:

- Inputs can be read through memory variables.
- An input cannot be configured as an expert function if it has already been configured as a Run/Stop input.
- An output cannot be configured in an expert function if it has already been configured as an alarm.
- Short-Circuit management applies on the outputs. Status of outputs are available.
- The I/O that are not used by expert functions can be used as any other regular I/O.
- When inputs are used in expert functions (Latch, HSC,…), integrator filter is replaced by anti-bounce filter. Filter value is configured in the configuration screen.

Counting Function

Overview

The Counting function can execute fast counts of pulses from sensors, encoders, switches, and so on, that are connected to fast inputs. The Counting function can also be connected to regular inputs, in which case the function operates at a lower frequency.

There are 2 types of embedded counting functions:

- **Simple** type: a single input counter.
- **Main** type: a counter that uses up to 4 inputs and 2 reflex outputs.

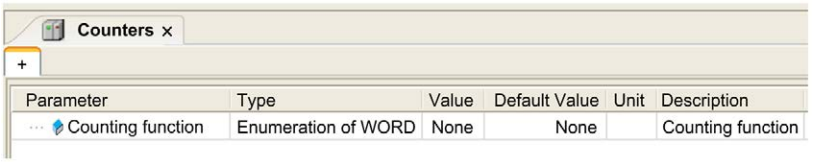
Based on the embedded counting functions, there are 5 types of counters that you can configure in EcoStruxure Machine Expert:

- **HSC Simple**
- **HSC Main Single Phase**
- **HSC Main Dual Phase**
- **Frequency Meter**
- **Period Meter**

The **Frequency Meter** type and the **Period Meter** type are based on an **HSC Main** type.

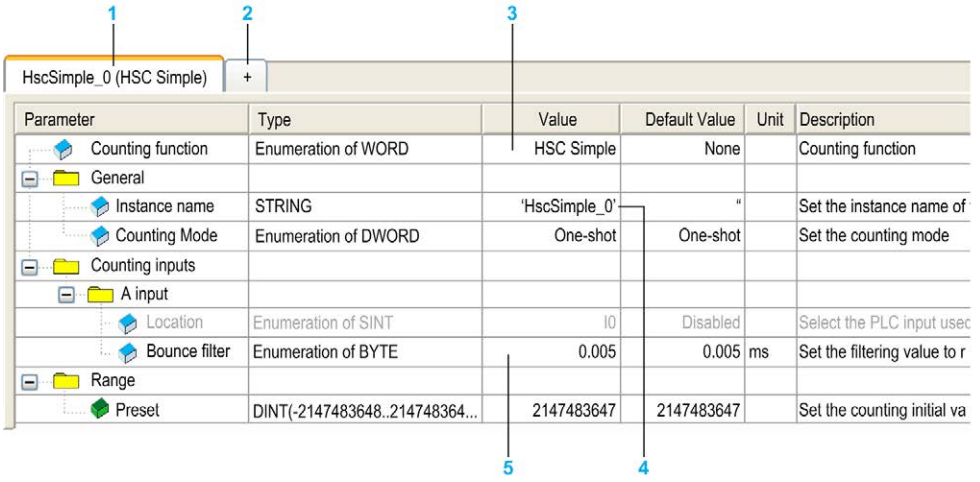
Accessing the Counting Function Configuration Window

Follow these steps to access the embedded counting function configuration window:

Step	Description
1	Double-click **Counters** in the **Devices tree**. The Counting Function window appears: 
2	Double-click **Value** and choose the counting function type to assign.

Counting Function Configuration Window

The following figure shows a sample HSC configuration window:



The following table describes the areas of the **Counters** configuration window:

Number	Action
1	The instance name of the function and the currently configured counting function type .
2	Click **+** to configure a new instance of counting function.
3	Double-click the **Value** column to display a list of the counter function types available.
4	Double-click the **Instance name** value to edit the instance name of the function. The **Instance name** is automatically given by EcoStruxure Machine Expert. The **Instance name** parameter is editable and allows you to define the instance name. However, whether the **Instance name** is software-defined or user-defined, use the same instance name as an input to the function blocks dealing with the counter, as defined in the **Counters** editor.
5	Configure each parameter by clicking the plus sign next to it to access its settings. The parameters available depend on the mode used.

For detail information on configuration parameters, refer to M241 HSC Library Guide.

Pulse Generators Embedded Function

Overview

The pulse generated embedded functions available with the M241 are:

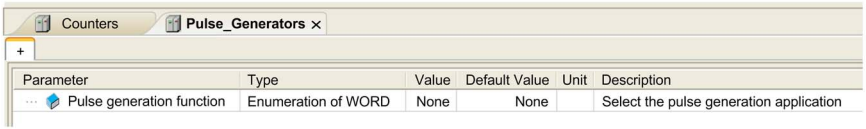
PTO The PTO (Pulse Train Output) implements digital technology that provides precise positioning for open loop control of motor drives.

PWM The PWM (Pulse Width Modulation) function generates a programmable square wave signal on a dedicated output with adjustable duty cycle and frequency.

FreqGen The FreqGen (Frequency Generator) function generates a square wave signal on dedicated output channels with a fixed duty cycle (50%).

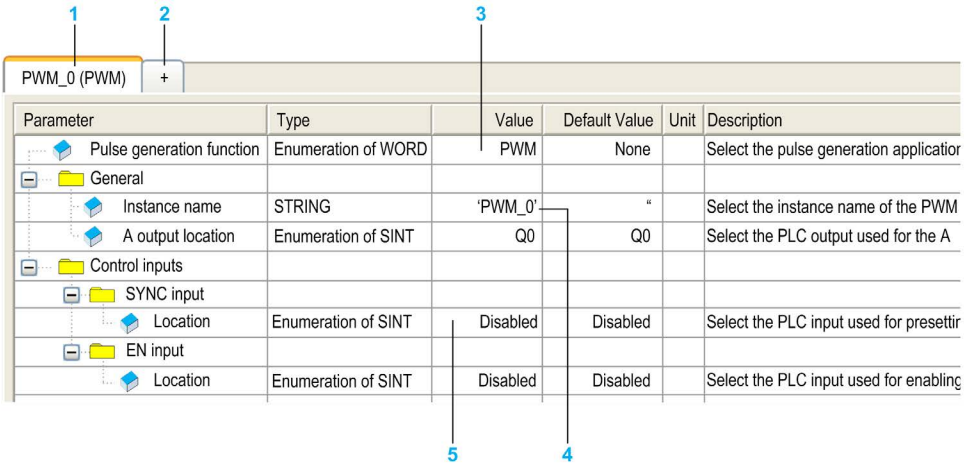
Accessing the Pulse Generators Configuration Window

Follow these steps to access the Pulse Generators configuration window:

Step	Description
1	Double-click **Pulse Generators** on the **Devices tree**. The Pulse Generation Function window appears: 
2	Double-click **Value** and choose the pulse generator function type to assign.

Pulse Generators Configuration Window

The figure shows a sample **Pulse_Generators** configuration window used to configure a PTO, PWM, or FreqGen function:



The following table describes the areas of the **Pulse_Generators** configuration window:

Number	Action
1	The instance name of the function and the currently configured pulse generator function type .
2	Click **+** to configure a new instance of pulse generator function.
3	Double-click the **Value** column to display a list of the pulse generator function types available.
4	Double-click the **Instance name** value to edit the instance name of the function. The **Instance name** is automatically given by EcoStruxure Machine Expert. The **Instance name** parameter is editable and allows you to define the instance name. However, whether the **Instance name** is software-defined or user-defined, use the same instance name as an input to the function blocks dealing with the counter, as defined in the **Counters** editor.
5	Configure each parameter by clicking the plus sign next to it to access its settings. The parameters available depend on the type of pulse generator used.

For detailed information on configuration parameters, refer to the M241 PTOPWM Library Guide.

Chapter 11
Cartridge Configuration

TMC4 Cartridge Configuration

Introduction

The Modicon M241 Logic Controller supports the following cartridges:

- TMC4 standard cartridges
- TMC4 application cartridges

For further information about the TMC4 cartridge configuration, refer to the TMC4 Cartridges Programming Guide *(see Modicon TMC4, Cartridges, Programming Guide)*.

⚠ 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.

Adding a TMC4 Cartridge

To add a cartridge to your controller, select the cartridge in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Chapter 12
Expansion Modules Configuration

Overview

This chapter describes how to configure the TM4, TM3, and TM2 expansion modules for the Modicon M241 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TM3 I/O Configuration General Description	104
TM3 I/O Bus Configuration	110
TM4 Expansion Module Configuration	111
TM3/TM2 Expansion Module Configuration	112
Optional I/O Expansion Modules	113

TM3 I/O Configuration General Description

Introduction

In your project, you can add I/O expansion modules to your M241 Logic Controller to increase the number of digital and analog inputs and outputs over those native to the controller (embedded I/O).

You can add either TM3 or TM2 I/O expansion modules to the logic controller, and further expand the number of I/O via TM3 transmitter and receiver modules to create remote I/O configurations. Special rules apply in all cases when creating local and remote I/O expansions, and when mixing TM2 and TM3 I/O expansion modules (refer to Maximum Hardware Configuration *(see Modicon M241 Logic Controller, Hardware Guide)*).

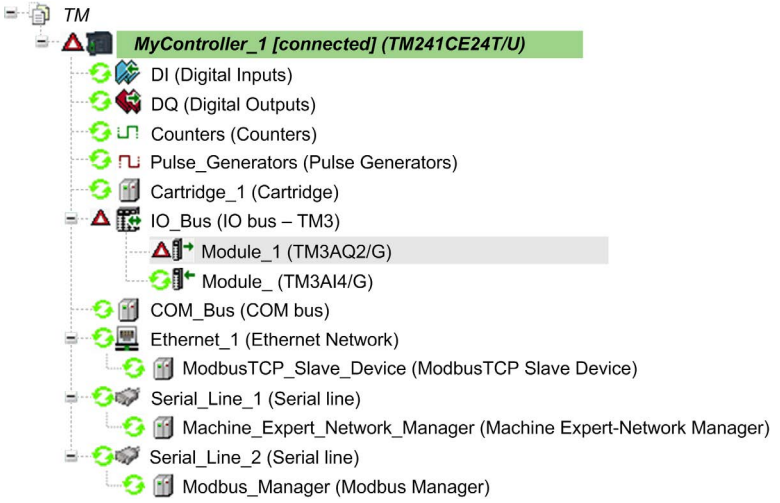
The I/O expansion bus of the M241 Logic Controller is created when you assemble the I/O expansion modules to the logic controller. I/O expansion modules are considered as external devices in the logic controller architecture and, as such, are treated differently than the embedded I/Os of the logic controller.

I/O Expansion Bus Errors

If the logic controller cannot communicate with one or more I/O expansion modules contained in the program configuration, and those modules are not configured as optional modules (refer to Optional I/O Expansion Modules *(see page 113)*), the logic controller considers it as an I/O expansion bus error. The unsuccessful communication may be detected during the startup of the logic controller or during runtime, and there may be any number of causes. Causes of communication exceptions on the I/O expansion bus include, among other things, disconnection of or physically missing I/O modules, electromagnetic radiation beyond published environmental specifications, or otherwise inoperative modules.

If an I/O expansion bus error is detected:

- The system status LED **I/O** of the logic controller is illuminated indicating an I/O error.
- When EcoStruxure Machine Expert is in online mode, a red triangle appears next to the TM3 expansion module or modules in error and next to the **IO_Bus** node on the **Devices tree** window:



The following diagnostic information is also available:

- Bit 0 and bit 1 of the `PLC_R.i_lwSystemFault_1` system variable are set to 0.
- The `PLC_R.i_wIOStatus1` and `PLC_R.i_wIOStatus2` system variables are set to `PLC_R_IO_BUS_ERROR`.
- The `TM3_MODULE_R[i].i_wModuleState` system variable, where `[i]` identifies the TM3 expansion module in error, is set to `TM3_BUS_ERROR`.
- The `TM3_GetModuleBusStatus` function block returns the `TM3_ERR_BUS` error code *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*.

Refer to PLC_R *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* and TM3_MODULE_R *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* structures for details on system variables.

Active I/O Expansion Bus Error Handling

The TM3_BUS_W.q_wIOBusErrPassiv system variable is set to ERR_ACTIVE by default to specify the use of active I/O error handling. The application can set this bit to ERR_PASSIVE to use passive I/O error handling instead.

By default, when the logic controller detects a TM3 module in bus communication error it sets the bus to a "bus off" condition whereby the TM3 expansion module outputs, the input image value and the output image value are set to 0. A TM3 expansion module is considered to be in bus communication error when an I/O exchange with the expansion module has been unsuccessful for at least two consecutive bus task cycles. When a bus communication error occurs, the TM3_MODULE_R[i].i_wModuleState system variable, where [i] is the expansion module number in error, is set to TM3_BUS_ERROR. All other bits are set to TM3_OK.

Normal I/O expansion bus operation can only be restored after eliminating the source of the error and performing one of the following:

- Power cycle
- New application download
- Restarting the I/O Bus by setting the TM3_BUS_W.q_wIOBusRestart system variable to 1. The bus is restarted if at least one expansion module is in error (TM3_MODULE_R[i].i_wModuleState = TM3_BUS_ERROR). Refer to Restarting the I/O Expansion Bus *(see page 108)*.
- Issuing a **Reset Warm** or **Reset Cold** command with EcoStruxure Machine Expert *(see page 67)*.

Passive I/O Expansion Bus Handling

The application can set the system variable TM3_BUS_W.q_wIOBusErrPassiv to ERR_PASSIVE to use passive I/O error handling. This error handling is provided to afford compatibility with previous firmware versions.

When passive I/O error handling is in use, the logic controller attempts to continue data bus exchanges with the modules during bus communication errors. While the expansion bus error persists, the logic controller attempts to re-establish communication on the bus with incommunicative modules, depending on the type of I/O expansion module:

- For TM3 I/O expansion modules, the value of the I/O channels is maintained (**Keep current values**) for approximately 10 seconds while the logic controller attempts to re-establish communication. If the logic controller cannot re-establish communications within that time, all affected TM3 I/O expansion outputs are set to 0.
- For TM2 I/O expansion modules that may be part of the configuration, the value of the I/O channels is maintained indefinitely. That is to say, the outputs of the TM2 I/O expansion modules are set to “Keep current values” until either power is cycled on the logic controller system, or you issue a **Reset Warm** or **Reset Cold** command with EcoStruxure Machine Expert *(see page 67)*.

In either case, the logic controller continues to solve logic and, if your controller is so equipped, the embedded I/O continues to be managed by the application ("managed by application program *(see page 64)*") while it attempts to re-establish communication with the incommunicative I/O expansion modules. If the communication is successful, the I/O expansion modules resume to be managed by the application. If communication with the I/O expansion modules is unsuccessful, you must resolve the reason for the unsuccessful communication, and then cycle power on the logic controller system, or issue a **Reset Warm** or **Reset Cold** command with EcoStruxure Machine Expert *(see page 67)*.

The value of the incommunicative I/O expansion modules input image is maintained and the output image value is set by the application.

Further, if the incommunicative I/O module(s) disturb the communication with unaffected modules, the unaffected modules are also considered to be in error and the `TM3_MODULE_R[i].i_wModuleState` system variable (where `[i]` is the expansion module number) is set to `TM3_BUS_ERROR`. However, with the ongoing data exchanges that characterize the Passive I/O Expansion Bus Error Handling, the unaffected modules apply the data sent, and do not apply the fallback values as for the incommunicative module.

Therefore, you must monitor within your application the state of the bus and the error state of the module(s) on the bus, and take the appropriate action necessary given your particular application.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Include in your risk assessment the possibility of unsuccessful communication between the logic controller and any I/O expansion modules.
- If the "Keep current values" option deployed during an I/O expansion module external error is incompatible with your application, use alternate means to control your application for such an event.
- Monitor the state of the I/O expansion bus using the dedicated system variables and take appropriate actions as determined by your risk assessment.

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

For more information on the actions taken upon startup of the logic controller when an I/O expansion bus error is detected, refer to Controller States Description *(see page 59)*.

Restarting the I/O Expansion Bus

When active I/O error handling is being applied, that is, embedded and TM3 outputs set to 0 when a bus communication error is detected, the application can request a restart of the I/O expansion bus while the logic controller is still running (without the need for a Cold Start, Warm Start, power cycle, or application download).

The `TM3_BUS_W. q_wIoBusRestart` system variable is available to request restarts of the I/O expansion bus. The default value of this bit is 0. Provided at least one TM3 expansion module is in error (`TM3_MODULE_R[i].i_wModuleState` set to `TM3_BUS_ERROR`), the application can set `TM3_BUS_W. q_wIoBusRestart` to 1 to request a restart of the I/O expansion bus. On detection of a rising edge of this bit, the logic controller reconfigures and restarts the I/O expansion bus if all of the following conditions are met:

- The `TM3_BUS_W.q_wIOBusErrPassiv` system variable is set to `ERR_ACTIVE` (that is, I/O expansion bus activity is stopped)
- Bit 0 and bit 1 of the `PLC_R.i_lwSystemFault_1` system variable are set to 0 (I/O expansion bus is in error)
- The `TM3_MODULE_R[i].i_wModuleState` system variable is set to `TM3_BUS_ERROR` (at least one expansion module is in bus communication error)

If the `TM3_BUS_W.q_wIoBusRestart` system variable is set to 1 and any of the above conditions is not met, the logic controller takes no action.

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 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 or, depending on the controller reference, to or from the controller (in the form of cartridges), then you must 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 no longer function while the embedded I/O that may be present in your controller continues 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 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.

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 113)*.

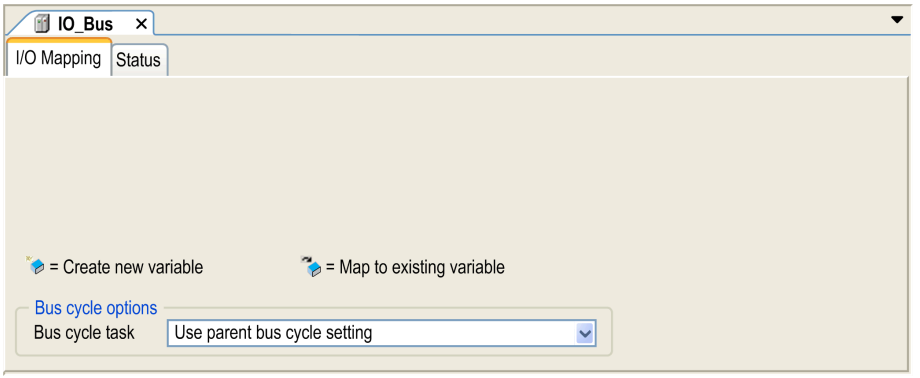
TM3 I/O Bus Configuration

Overview

TM3 I/O bus configuration enables you to select the task that drives TM3 physical exchanges. It can also override the configuration defined in the **PLC settings** *(see page 81)* bus cycle task.

Configuring the I/O Bus

Follow these steps to configure the TM3 I/O bus:

Step	Description
1	In the **Devices tree**, double-click **IO_Bus**. **Result**: The **IO_Bus** editor tab appears: 
2	Set the **Bus cycle task** from the list to either of the following: • **Use parent bus cycle setting** (default) Sets the task for bus exchange as defined in the **PLC settings**. • **MAST** Sets the Master task for bus exchange irrespective of the task defined in the **PLC settings**.

TM4 Expansion Module Configuration

Introduction

The Modicon M241 Logic Controller supports the TM4 communication expansion modules.

For further information about the TM4 expansion modules configuration, refer to the TM4 Expansion Modules Configuration Programming Guide.

⚠ 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.

Adding an Expansion Module

To add an expansion module to your controller, select the expansion module in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

TM3/TM2 Expansion Module Configuration

Introduction

The Modicon M241 Logic Controller supports the following expansion modules:

- TM3 expansion modules
  - Digital I/O modules
  - Analog I/O modules
  - Expert I/O modules
  - Safety modules
  - Transmitter and receiver modules
- TM2 expansion modules
  - Digital I/O modules
  - Analog I/O modules
  - Expert modules
  - Communication modules

For further information about the TM3 and TM2 expansion modules configuration, refer to the TM3 Expansion Modules Configuration Programming Guide and TM2 Expansion Modules Configuration Programming Guide respectively.

⚠ 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.

Adding an Expansion Module

To add an expansion module to your controller, select the expansion module in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

- Using the Drag-and-Drop Method *(see EcoStruxure Machine Expert, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

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 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 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 controller 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 controller ignores the absent I/O expansion modules that you have marked as optional, which then allows the controller to start the I/O expansion bus.

The 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 controller.

The following module types can be marked as optional:

- TM3 I/O expansion modules
- TM2 I/O expansion modules

NOTE: TM3 Transmitter/Receiver modules (the TM3XTRA1 and the TM3XREC1) and TMC4 cartridges cannot be marked as optional.

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

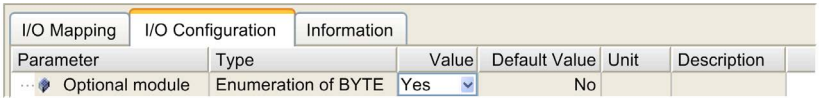
UNINTENDED EQUIPMENT OPERATION

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

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

Step	Action
1	Add the expansion module to your controller .
2	In the **Devices tree**, double-click the expansion module.
3	Select the **I/O Configuration** tab.
4	In the **Optional module** line, select **Yes** in the **Value** column: 

Shared Internal ID Codes

Controllers and bus couplers identify expansion modules by a simple internal ID code. This ID code is not specific to each reference, but identifies the logical structure of the expansion module. Therefore, different references can share the same ID code.

You cannot have two modules with the same internal ID code declared as optional without at least one mandatory module placed between them.

This table groups the module references sharing the same internal ID code:

Modules sharing the same internal ID code
TM2DDI16DT, TM2DDI16DK
TM2DRA16RT, TM2DDO16UK, TM2DDO16TK
TM2DDI8DT, TM2DAI8DT
TM2DRA8RT, TM2DDO8UT, TM2DDO8TT
TM2DDO32TK, TM2DDO32UK
TM3DI16K, TM3DI16, TM3DI16G
TM3DQ16R, TM3DQ16RG, TM3DQ16T, TM3DQ16TG, TM3DQ16TK, TM3DQ16U, TM3DQ16UG, TM3DQ16UK
TM3DQ32TK, TM3DQ32UK
TM3DI8, TM3DI8G, TM3DI8A
TM3DQ8R, TM3DQ8RG, TM3DQ8T, TM3DQ8TG, TM3DQ8U, TM3DQ8UG
TM3DM8R, TM3DM8RG
TM3DM24R, TM3DM24RG
TM3SAK6R, TM3SAK6RG

Modules sharing the same internal ID code
TM3SAF5R, TM3SAF5RG
TM3SAC5R, TM3SAC5RG
TM3SAFL5R, TM3SAFL5RG
TM3AI2H, TM3AI2HG
TM3AI4, TM3AI4G
TM3AI8, TM3AI8G
TM3AQ2, TM3AQ2G
TM3AQ4, TM3AQ4G
TM3AM6, TM3AM6G
TM3TM3, TM3TM3G
TM3TI4, TM3TI4G
TM3TI4D, TM3TI4DG
TM3TI8T, TM3TI8TG
TM3XHSC202, TM3XHSC202G

Chapter 13
Ethernet Configuration

Introduction

This chapter describes how to configure the Ethernet network interface of the Modicon M241 Logic Controller.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
13.1	Ethernet Services	118
13.2	Firewall Configuration	182

Section 13.1
Ethernet Services

What Is in This Section?

This section contains the following topics:

Topic	Page
Presentation	119
IP Address Configuration	121
Modbus TCP Client/Server	127
Web Server	129
FTP Server	145
FTP Client	147
SNMP	148
Controller as a Target Device on EtherNet/IP	149
Controller as a Slave Device on Modbus TCP	175
Changing the Modbus TCP Port	180

Presentation

Ethernet Services

The controller supports the following services:

- Modbus TCP Server *(see page 127)*
- Modbus TCP Client *(see page 127)*
- Web Server *(see page 129)*
- FTP Server *(see page 145)*
- SNMP *(see page 148)*
- Controller as Target Device On EtherNet/IP *(see page 149)*
- Controller as Slave Device On Modbus TCP *(see page 175)*
- IEC VAR ACCESS *(see page 120)*

Ethernet Protocols

The controller supports the following protocols:

- IP (Internet Protocol)
- UDP (User Datagram Protocol)
- TCP (Transmission Control Protocol)
- ARP (Address Resolution Protocol)
- ICMP (Internet Control Messaging Protocol)
- IGMP (Internet Group Management Protocol)

Connections

This table shows the maximum number of connections:

Connection Type	Maximum Number of Connections
Modbus Server	8
Modbus Client	8
EtherNet/IP Target	16
FTP Server	4
Web Server	10
Machine Expert Protocol (EcoStruxure Machine Expert software, trace, Web visualization, HMI devices)	8

NOTE: When at least one EtherNet/IP target is configured, the total number of connections (EtherNet/IP plus Modbus TCP) is limited to 16. Only if the Modbus TCP IOScanner is exclusively used, the total number of slave devices can be up to 64. These maximums are controlled for at build time.

Each connection based on TCP manages its own set of connections as follows:

1. When a client tries to open a connection that exceeds the poll size, the controller closes the oldest connection.
2. If all connections are busy (exchange in progress) when a client tries to open a new one, the new connection is denied.
3. All server connections stay open as long as the controller stays in operational states (`RUNNING`, `STOPPED`, `HALT`).
4. All server connections are closed when leaving or entering operational states (`RUNNING`, `STOPPED`, `HALT`), except in case of power outage (because the controller does not have time to close the connections).

Connections can be closed when the originator of the connection requests to close the connection it had previously opened.

Services Available

With an Ethernet communication, the **IEC VAR ACCESS** service is supported by the controller. With the **IEC VAR ACCESS** service, data can be exchanged between the controller and an HMI.

The **NetWork variables** service is also supported by the controller. With the **NetWork variables** service, data can be exchanged between controllers.

NOTE: For more information, refer to the EcoStruxure Machine Expert Programming Guide.

IP Address Configuration

Introduction

There are different ways to assign the IP address to the added Ethernet interface of the controller:

- Address assignment by DHCP server
- Address assignment by BOOTP server
- Fixed IP address
- Post configuration file *(see page 239)*. If a post configuration file exists, this assignment method has priority over the others.

The IP address can also be changed dynamically through the:

- Communication Settings *(see EcoStruxure Machine Expert, Programming Guide)* tab in EcoStruxure Machine Expert
- **changeIPAddress** function block *(see page 269)*

NOTE: If the attempted addressing method is unsuccessful, the link uses a default IP address *(see page 124)* derived from the MAC address.

Carefully manage the IP addresses because each device on the network requires a unique address. Having multiple devices with the same IP address can cause unintended operation of your network and associated equipment.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

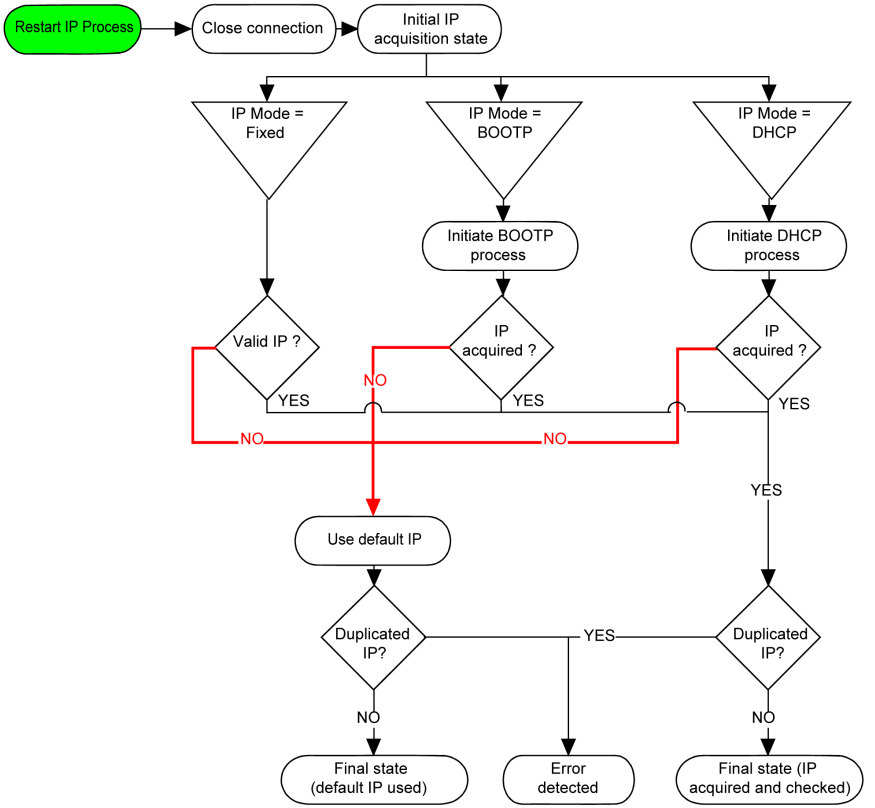
- Verify that there is only one master controller configured on the network or remote link.
- Verify that all devices have unique addresses.
- Obtain your IP address from your system administrator.
- Confirm that the IP address of the device is unique before placing the system into service.
- Do not assign the same IP address to any other equipment on the network.
- Update the IP address after cloning any application that includes Ethernet communications to a unique address.

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

NOTE: Verify that your system administrator maintains a record of assigned IP addresses on the network and subnetwork, and inform the system administrator of any configuration changes performed.

Address Management

This diagram shows the different types of address systems for the controller:



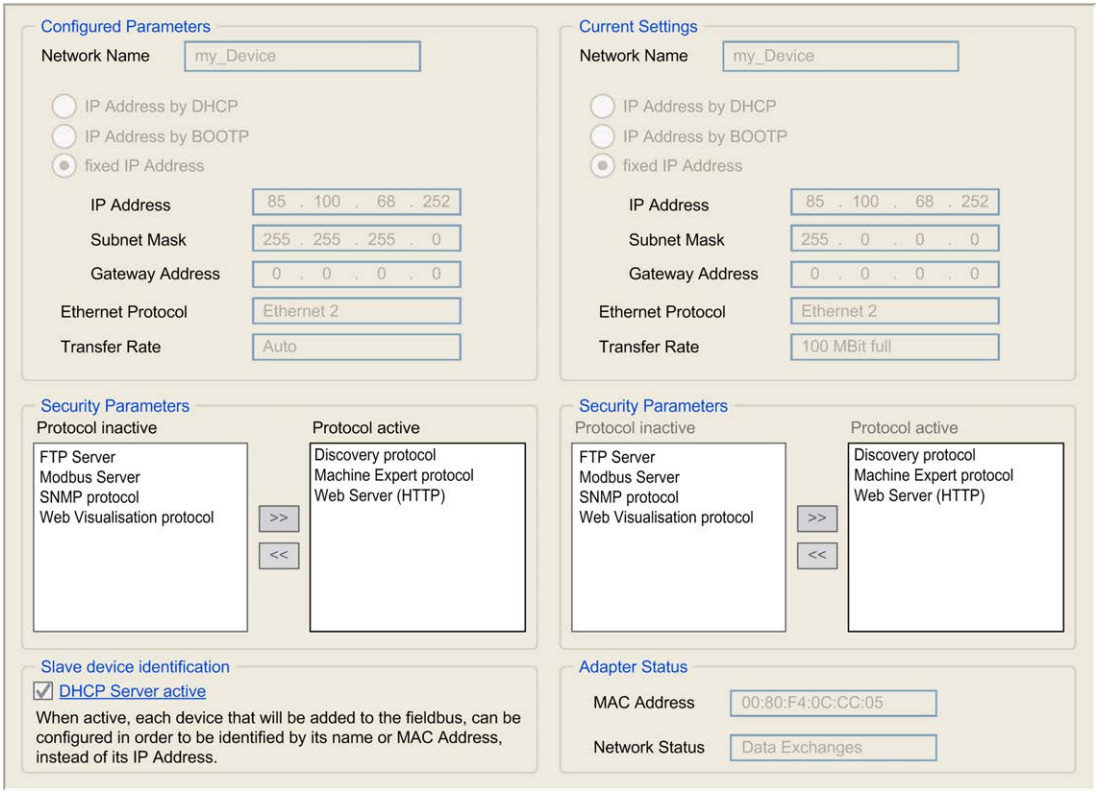
NOTE: If a device programmed to use the DHCP or BOOTP addressing methods is unable to contact its respective server, the controller uses the default IP address. It repeats its request constantly.

The IP process restarts in the following cases:

- Controller reboot
- Ethernet cable reconnection
- Application download (if IP parameters change)
- DHCP or BOOTP server detected after a prior addressing attempt was unsuccessful.

Ethernet Configuration

In the **Devices tree**, double-click **Ethernet_1**:



Note: If you are in online mode, you see the two windows. You cannot edit them. If you are in offline mode, you see the **Configured Parameters** window. You can edit it.

This table describes the configured parameters:

Configured Parameters	Description
Network Name	Used as device name to retrieve IP address through DHCP, maximum 15 characters.
IP Address by DHCP	IP address is obtained by DHCP server.
IP Address by BOOTP	IP address is obtained by BOOTP server.
Fixed IP Address	IP address, Subnet Mask, and Gateway Address are defined by the user.

Configured Parameters	Description
Ethernet Protocol	Protocol type used (Ethernet 2 or IEEE 802.3) **NOTE:** If you change the Ethernet Protocol, a power cycle is required before it will be recognized by the controller.
Transfer Rate	Speed and Duplex are in auto-negotiation mode.

Default IP Address

The IP address by default is 10.10.x.x.

The last two fields in the default IP address are composed of the decimal equivalent of the last two hexadecimal bytes of the MAC address of the port.

The MAC address of the port can be retrieved on the label placed on the front side of the controller.

The default subnet mask is Default Class A Subnet Mask of 255.0.0.0.

NOTE: A MAC address is written in hexadecimal format and an IP address in decimal format. Convert the MAC address to decimal format.

Example: If the MAC address is 00.80.F4.01.80.F2, the default IP address is 10.10.128.242.

Address Classes

The IP address is linked:

- to a device (the host)
- to the network to which the device is connected

An IP address is always coded using 4 bytes.

The distribution of these bytes between the network address and the device address may vary.This distribution is defined by the address classes.

The different IP address classes are defined in this table:

Address Class	Byte1					Byte 2	Byte 3	Byte 4
Class A	0	Network ID				Host ID		
Class B	1	0	Network ID				Host ID	
Class C	1	1	0	Network ID				Host ID
Class D	1	1	1	0	Multicast Address			
Class E	1	1	1	1	0	Address reserved for subsequent use		

Subnet Mask

The subnet mask is used to address several physical networks with a single network address. The mask is used to separate the subnetwork and the device address in the host ID.

The subnet address is obtained by retaining the bits of the IP address that correspond to the positions of the mask containing 1, and replacing the others with 0.

Conversely, the subnet address of the host device is obtained by retaining the bits of the IP address that correspond to the positions of the mask containing 0, and replacing the others with 1.

Example of a subnet address:

IP address	192 (11000000)	1 (00000001)	17 (00010001)	11 (00001011)
Subnet mask	255 (11111111)	255 (11111111)	240 (11110000)	0 (00000000)
Subnet address	192 (11000000)	1 (00000001)	16 (00010000)	0 (00000000)

NOTE: The device does not communicate on its subnetwork when there is no gateway.

Gateway Address

The gateway allows a message to be routed to a device that is not on the current network.

If there is no gateway, the gateway address is 0.0.0.0.

The gateway address must be defined on Ethernet_1 interface. The traffic to unknown networks is sent through this interface.

Security Parameters

This table describes the different security parameters:

Security Parameters	Description	Default settings
Discovery protocol	This parameter deactivates Discovery protocol. When deactivated, Discovery requests are ignored.	Active
FTP Server	This parameter deactivates the FTP Server of the controller. When deactivated, FTP requests are ignored.	Inactive
Machine Expert protocol	This parameter deactivates the Machine Expert protocol on Ethernet interfaces. When deactivated, every Machine Expert request from every device is rejected, , including those from the UDP or TCP connection. Therefore, no connection is possible on Ethernet from a PC with EcoStruxure Machine Expert, from an HMI target that wants to exchange variables with this controller, from an OPC server, or from Controller Assistant.	Active
Modbus Server	This parameter deactivates the Modbus Server of the controller. When deactivated, every Modbus request to the controller is ignored.	Inactive
SNMP protocol	This parameter deactivates the SNMP server of the controller. When deactivated, SNMP requests are ignored.	Inactive
Web Server (HTTP)	This parameter deactivates the Web Server of the controller. When deactivated, HTTP requests to the controller Web Server are ignored.	Active
WebVisualisation protocol	This parameter deactivates the Web visualization pages of the controller. When deactivated, HTTP requests to the logic controller WebVisualisation protocol are ignored.	Inactive

Slave Device Identification

When **DHCP Server active** is selected, devices added to the fieldbus can be configured to be identified by their name or MAC address, instead of their IP address. Refer to DHCP Server *(see page 198)*.

Modbus TCP Client/Server

Introduction

Unlike Modbus serial link, Modbus TCP is not based on a hierarchical structure, but on a client/server model.

The Modicon M241 Logic Controller implements both client and server services so that it can initiate communications to other controllers and I/O devices, and to respond to requests from other controllers, SCADA, HMIs, and other devices. By default, Modbus Server functionality is not active.

Without any configuration, the embedded Ethernet port of the controller supports Modbus server.

The Modbus client/server is included in the firmware and does not require any programming action from the user. Due to this feature, it is accessible in RUNNING, STOPPED and EMPTY states.

Modbus TCP Client

The Modbus TCP client supports the following function blocks from the PLCCommunication library without any configuration:

- ADDM
- READ_VAR
- SEND_RECV_MSG
- SINGLE_WRITE
- WRITE_READ_VAR
- WRITE_VAR

For further information, refer to the Function Block Descriptions *(see EcoStruxure Machine Expert, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*.

Modbus TCP Server

The Modbus server supports the Modbus requests:

Function Code Dec (Hex)	Subfunction Dec (Hex)	Function
1 (1)	–	Read digital outputs (%Q)
2 (2)	–	Read digital inputs (%I)
3 (3)	–	Read holding register (%MW)
6 (6)	–	Write single register (%MW)
8 (8)	–	Diagnostic
15 (F)	–	Write multiple digital outputs (%Q)
16 (10)	–	Write multiple registers (%MW)
23 (17)	–	Read/write multiple registers (%MW)
43 (2B)	14 (E)	Read device identification

NOTE: The embedded Modbus server only ensures time-consistency for a single word (2 bytes). If your application requires time-consistency for more than 1 word, add and configure *(see Modicon TM4, Expansion Modules, Programming Guide)* a **Modbus TCP Slave Device** so that the contents of the `%IW` and `%QW` buffers are time-consistent in the associated IEC task (MAST by default).

Web Server

Introduction

As standard equipment, the controller provides an embedded Web server with a predefined, built-in website. You can use the pages of the website for module setup and control as well as application diagnostics and monitoring. These pages are ready to use with a Web browser. No configuration or programming is required.

The Web server can be accessed by the web browsers listed below:
- Google Chrome (version 30.0 or greater)
- Mozilla Firefox (version 1.5 or greater)

The Web server can maintain 10 simultaneous open sessions *(see page 119)*.

NOTE: The Web server can be disabled by unchecking the **Web Server active** parameter in the Ethernet Configuration tab *(see page 123)*.

The Web server is a tool for reading and writing data, and controlling the state of the controller, with full access to all data in your application. However, if there are security concerns over these functions, you must at a minimum assign a secure password to the Web Server or disable the Web server to prevent unauthorized access to the application. By enabling the Web server, you enable these functions.

The Web server allows you to monitor a controller and its application remotely, to perform various maintenance activities including modifications to data and configuration parameters, and change the state of the controller. Care must be taken to ensure that the immediate physical environment of the machine and process is in a state that will not present safety risks to people or property before exercising control remotely.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Configure and install the RUN/STOP input for the application, if available for your particular controller, so that local control over the starting or stopping of the controller can be maintained regardless of the remote commands sent to the controller.
- Define a secure password for the Web Server and do not allow unauthorized or otherwise unqualified personnel to use this feature.
- Ensure that there is a local, competent, and qualified observer present when operating on the controller from a remote location.
- You must have a complete understanding of the application and the machine/process it is controlling before attempting to adjust data, stopping an application that is operating, or starting the controller remotely.
- Take the precautions necessary to assure that you are operating on the intended controller by having clear, identifying documentation within the controller application and its remote connection.

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

NOTE: The Web server must only be used by authorized and qualified personnel. A qualified person is one who has the skills and knowledge related to the construction and operation of the machine and the process controlled by the application and its installation, and has received safety training to recognize and avoid the hazards involved. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this feature.

Web Server Access

Access to the Web server is controlled by User Rights when they are enabled in the controller. For more information, refer to **Users and Groups** Tab Description *(see page 78)*.

To access the Web server you must first connect to the controller with EcoStruxure Machine Expert or Controller Assistant and modify the default user password.

⚠ WARNING

UNAUTHORIZED DATA ACCESS

- Secure access to the FTP/Web server using User Rights.
- If you disable User Rights, disable the FTP/Web server to prevent any unwanted or unauthorized access to data in your application.

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

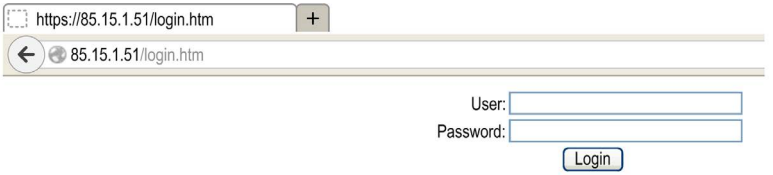
In order to change the password, go to **Users and Groups** tab of the device editor. For more information, refer to the EcoStruxure Machine Expert Programming Guide.

NOTE: The only way to gain access to a controller that has user access-rights enabled and for which you do not have the password(s) is by performing an Update Firmware operation. This clearing of User Rights can only be accomplished by using a SD card or USB key (depending on the support of your particular controller) to update the controller firmware. In addition, you may clear the User Rights in the controller by running a script (for more information, refer to EcoStruxure Machine Expert Programming Guide) . This effectively removes the existing application from the controller memory, but restores the ability to access the Controller.

Home Page Access

To access the website home page, type in your navigator the IP address of the controller.

This figure shows the Web Server site login page:



This figure shows the home page of the Web Server site once you have logged in:



NOTE: Schneider Electric adheres to industry best practices in the development and implementation of control systems. This includes a "Defense-in-Depth" approach to secure an Industrial Control System. This approach places the controllers behind one or more firewalls to restrict access to authorized personnel and protocols only.

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

- Evaluate whether your environment or your machines are connected to your critical infrastructure and, if so, take appropriate steps in terms of prevention, based on Defense-in-Depth, before connecting the automation system to any network.
- Limit the number of devices connected to a network to the minimum necessary.
- Isolate your industrial network from other networks inside your company.
- Protect any network against unintended access by using firewalls, VPN, or other, proven security measures.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Prepare a recovery plan including backup of your system and process information.

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

Monitoring: Data Parameters

Monitoring Web Server Variables

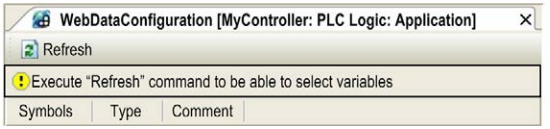
To monitor Web server variables, you must add a **Web Data Configuration** object to your project. Within this object, you can select all variables you want to monitor.

This table describes how to add a **Web Data Configuration** object:

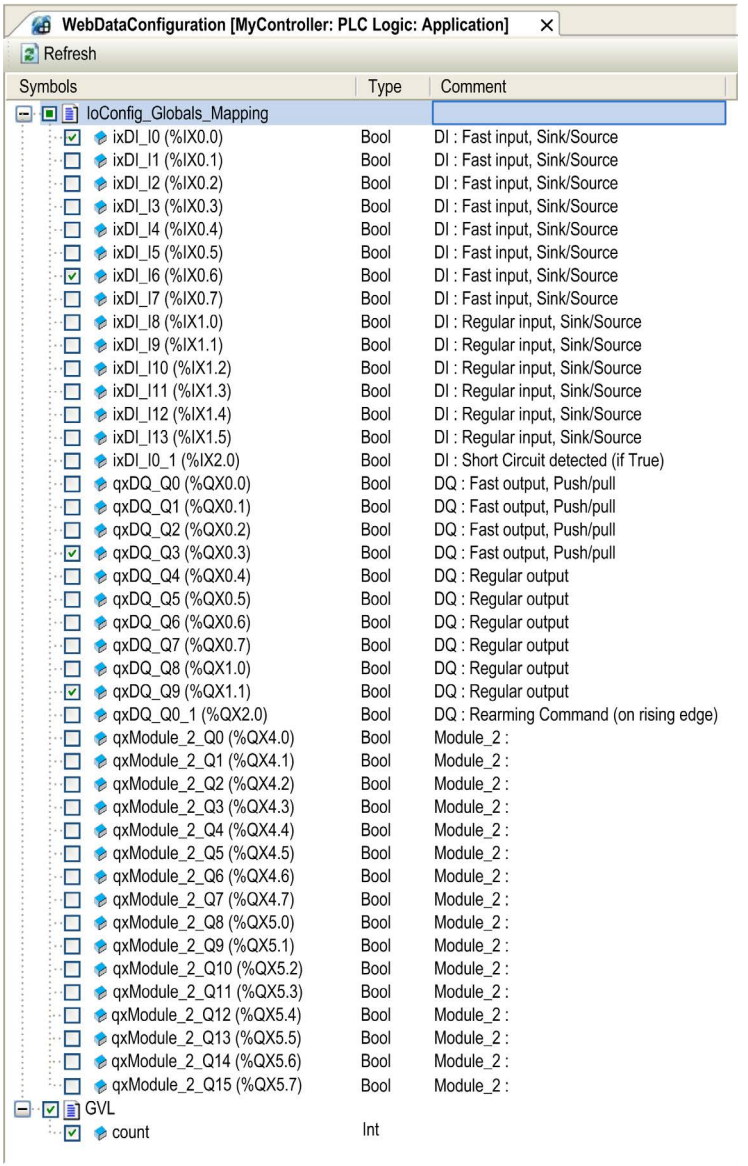
Step	Action
1	Right click the **Application** node in the **Applications tree** tab.
2	Click **Add Object → Web Data Configuration...**. **Result:** The **Add Web Data Configuration** window is displayed.
3	Click **Add**. **Result:** The **Web Data Configuration** object is created and the **Web Data Configuration** editor is open. **NOTE:** As a **Web Data Configuration** object is unique for a controller, its name cannot be changed.

Web Data Configuration Editor

Click the **Refresh** button to be able to select variables, this action will display all the variables defined in the application.



Select the variables you want to monitor in the web server:



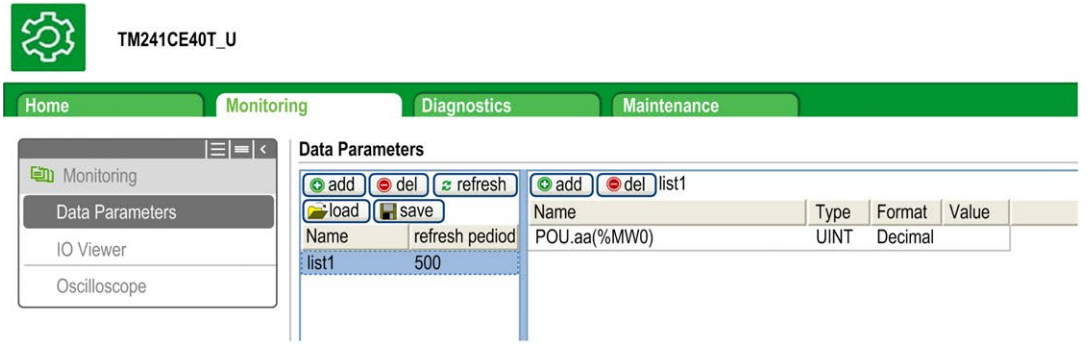
NOTE: The variable selection is possible only in offline mode.

Monitoring: Data Parameters Submenu

The **Data Parameters** submenu allows you to create and monitor some lists of variables. You can create several lists of variables (maximum 10 lists), each one containing several variables of the controller application (maximum 20 variables per list).

Each list has a name, and a refresh period. The lists are saved in the Flash memory of the controller, so that a created list can be accessed (loaded, modified, saved) from any Web client application accessing this controller.

The **Data Parameters** submenu allows you to display and modify variable values:



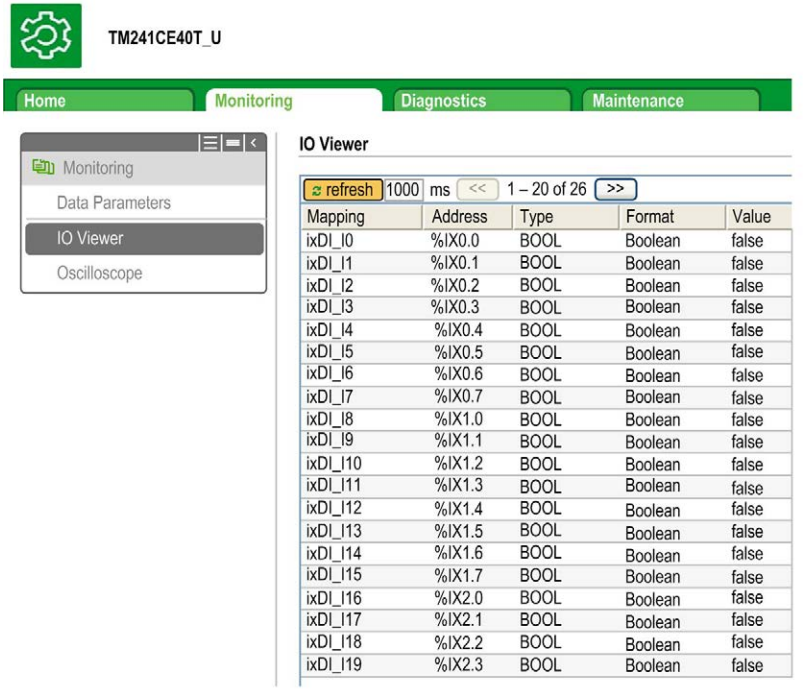
Element	Description
Add	Adds a list description or a variable
Del	Deletes a list description or a variable
Refresh period	Refreshing period of the variables contained in the list description (in ms)
Refresh	Enables I/O refreshing: ● Gray button: refreshing disabled ● Orange button: refreshing enabled
Load	Loads saved lists from the controller internal Flash to the Web server page
Save	Saves the selected list description in the controller (*/usr/web* directory)

NOTE: The IEC objects (`%IX`, `%QX`) are not directly accessible. To access IEC objects you must first group their contents in located registers (refer to Relocation Table *(see page 37)*).

NOTE: Bit memory variables (`%MX`) cannot be selected.

Monitoring: IO Viewer Submenu

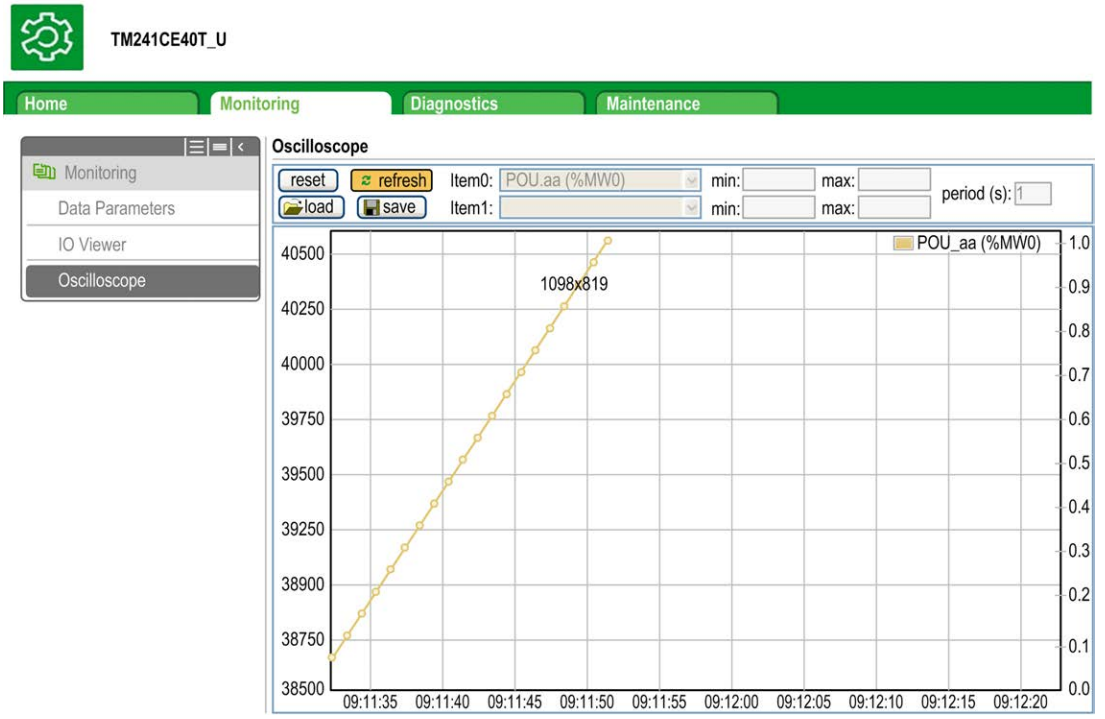
The **IO Viewer** submenu allows you to display and modify the current I/O values:



Element	Description
Refresh	Enables I/O refreshing: ● Gray button: refreshing disabled ● Orange button: refreshing enabled
1000 ms	I/O refreshing period in ms
<<	Goes to previous I/O list page
>>	Goes to next I/O list page

Monitoring: Oscilloscope Submenu

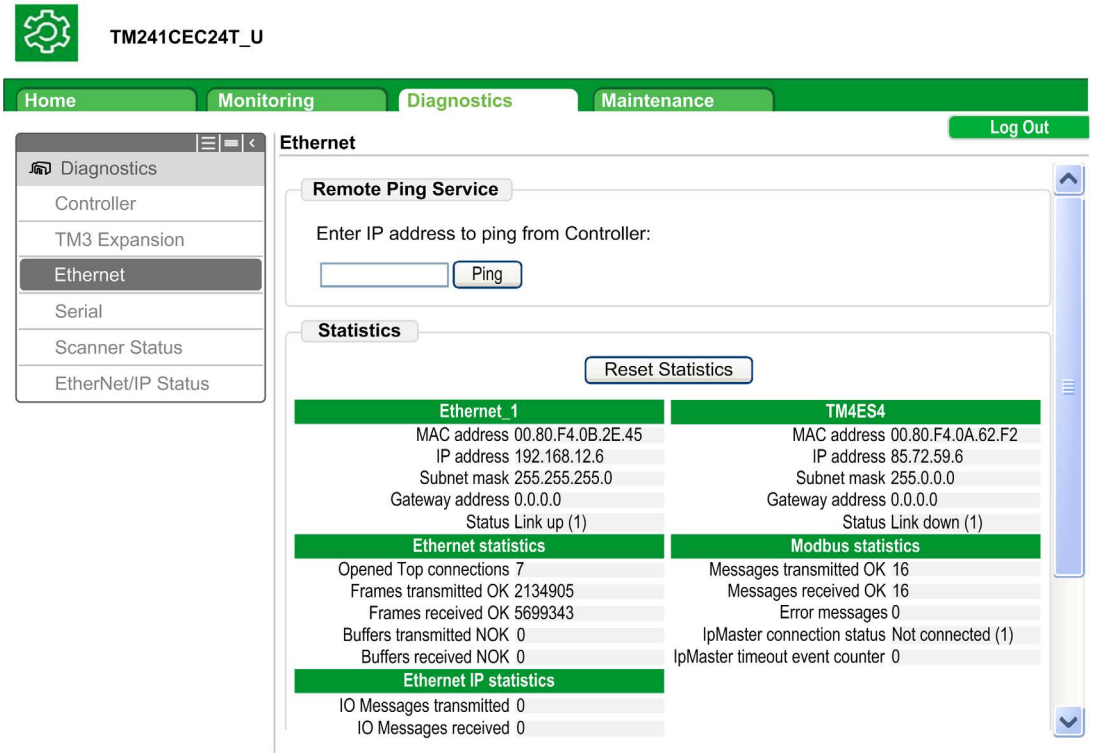
The **Oscilloscope** submenu can display up to 2 variables in the form of a recorder time chart:



Element	Description
Reset	Erases the memorization
Refresh	Starts/stops refreshing
Load	Loads parameter configuration of Item0 and Item1
Save	Saves parameter configuration of Item0 and Item1 in the controller
Item0	Variable to be displayed
Item1	Variable to be displayed
Min	Minimum value of the variable axis
Max	Maximum value of the variable axis
Period(ms)	Page refresh period in milliseconds

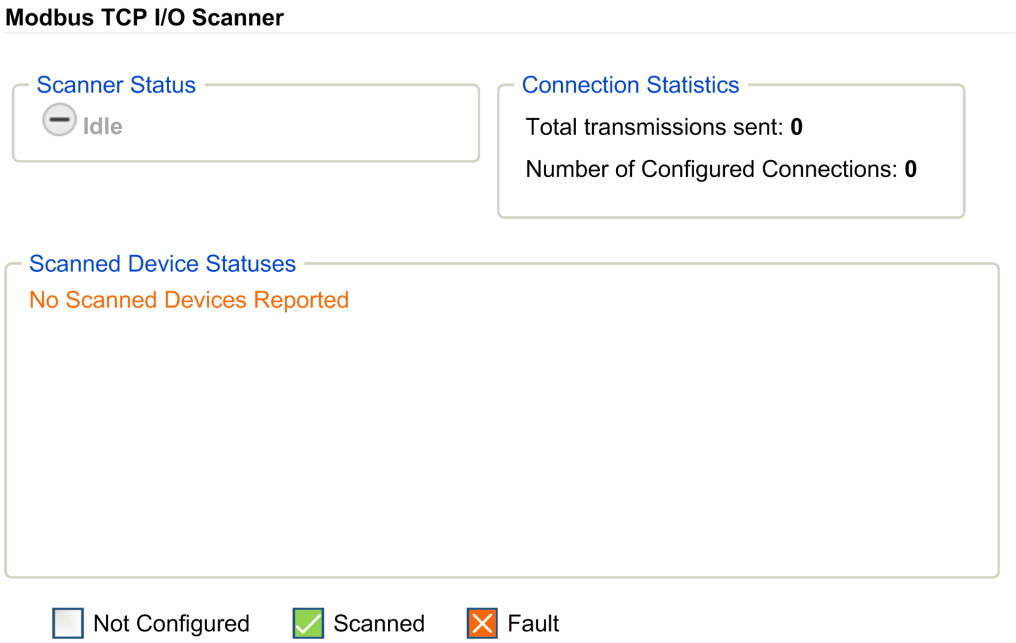
Diagnostics: Ethernet Submenu

This figure shows the remote ping service:



Diagnostics: Scanner Status Submenu

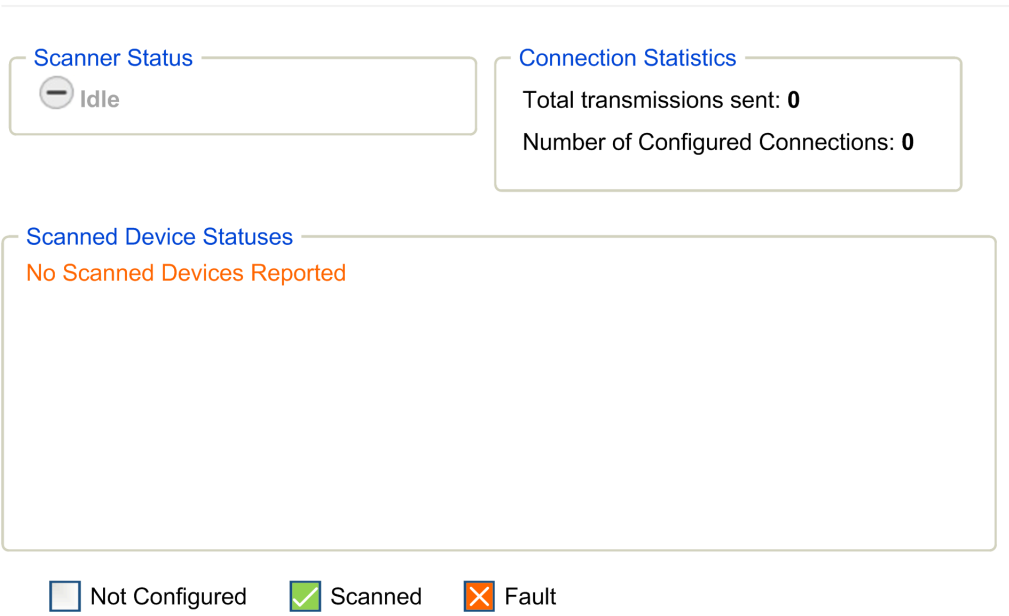
The **Scanner Status** submenu displays status of the Modbus TCP I/O Scanner (IDLE, STOPPED, OPERATIONAL) and the health bit of up to 64 Modbus slave devices:



For more information, refer to EcoStruxure Machine Expert Modbus TCP User guide.

Diagnostics: EtherNet/IP Status Submenu

The **EtherNet/IP Status** submenu displays the status of the EtherNet/IP Scanner (IDLE, STOPPED, OPERATIONAL) and the health bit of up to 16 EtherNet/IP target devices:



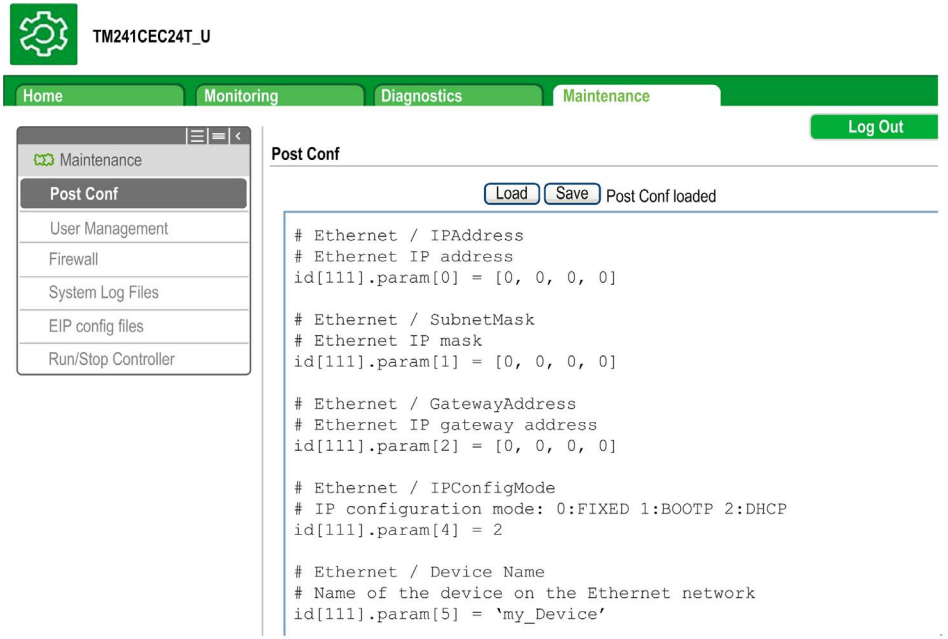
For more information, refer to EcoStruxure Machine Expert EtherNet/IP User guide.

Maintenance Page

The Maintenance page provides access to the controller data for maintenance capabilities.

Maintenance: Post Conf Submenu

The **Post Conf** submenu allows you to update the post configuration file *(see page 239)* saved on the controller:



Step	Action
1	Click **Load**.
2	Modify the parameters *(see page 243)*.
3	Click **Save**. **NOTE:** The new parameters will be considered at next Post Configuration file reading *(see page 240)*.

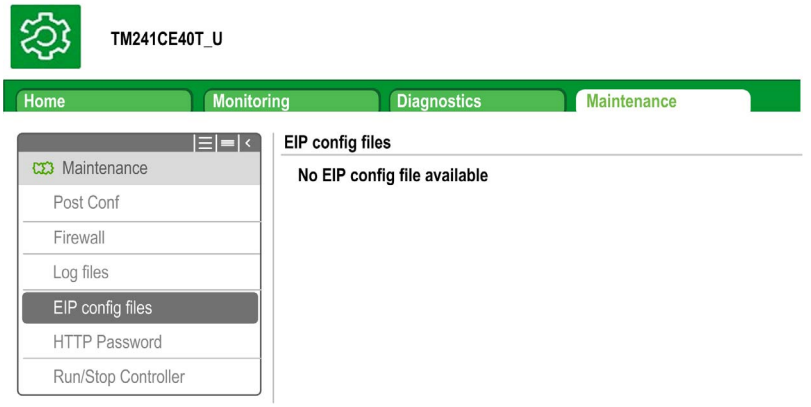
Log Files

This page provided access to the `/usr/Syslog/` folder of the controller flash memory *(see page 33)*.

Maintenance: EIP Config Files Submenu

The file tree only appears when the Ethernet IP service is configured on the controller.

Index of `/usr`:



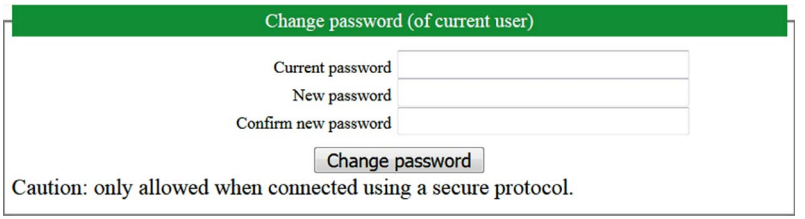
File	Description
My Machine Controller.gz	GZIP file
My Machine Controller.ico	Icon file
My Machine Controller.eds	Electronic Data Sheet file

Maintenance: User Management Submenu

The **User Management** submenu displays a screen that allows you to access four different actions, all restricted by using secure protocol (HTTPS):

- **Change password (of current user)**:

allows you to change your password.



- **User accounts management**:

Allows you to manage user accounts management, removing all password and returning all user accounts on the controller to default settings.



Click **Disable** to remove all passwords on the controller.

Click **OK** on the window that appears to confirm. As a result:
- Users no longer have to set and enter a password to connect to the controller.
- FTP, HTTP, and OPC UA Server connections accept anonymous user connections.
- Cloning the controller no longer requires authorization by using the FB_ControlClone function block *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*.

NOTE: The **Disable** button is only active if the current user has administrative privileges.

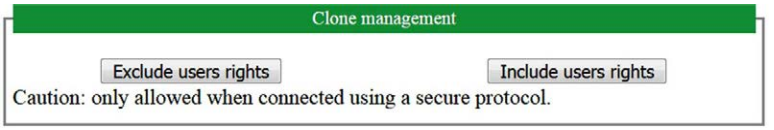
Click **Reset to default** to return all user accounts on the controller to their default setting state.

Click **OK** on the window that appears to confirm.

NOTE: Connections to FTP, HTTP, and the OPC UA Server are blocked until a new password is set.

- **Clone management**:

allows you to control whether user rights are copied and applied to the target controller when cloning a controller



Click **Exclude users rights** to exclude copying user rights to the target controller when cloning a controller.

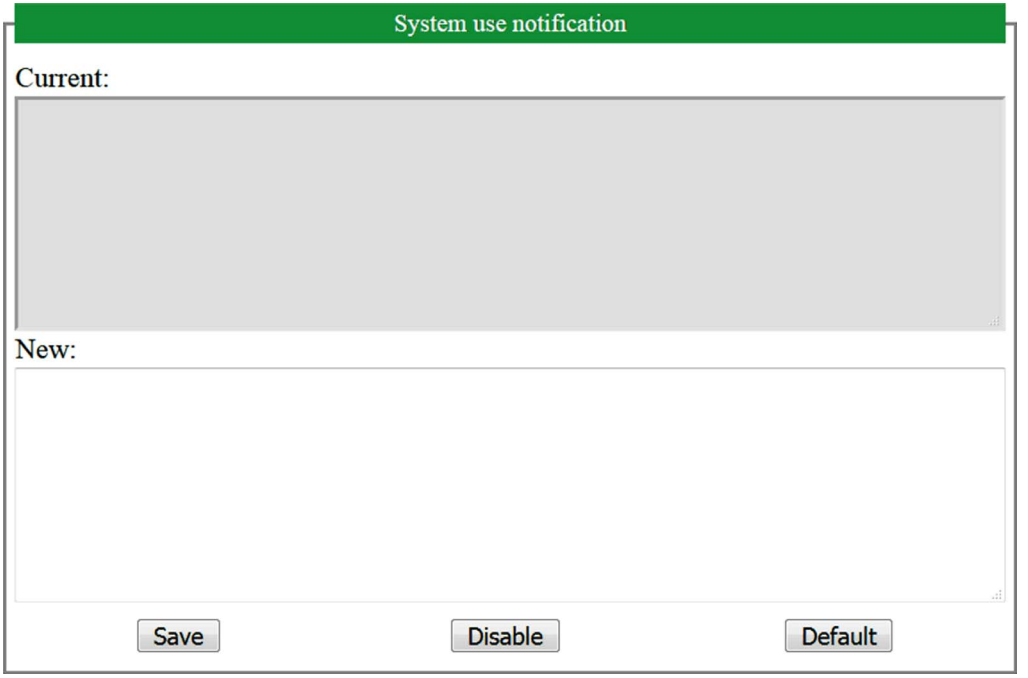
NOTE: By default, the users rights are excluded.

Click **Include users rights** to copy user rights to the target controller when cloning a controller. A popup prompts you to confirm copying the user rights. Click **OK** to continue.

NOTE: The **Exclude users rights** and **Include users rights** buttons are only active if the current user is connected to the controller using a secure protocol.

- **System use notification**:

allows you to customize a message which will be displayed at login.



FTP Server

Introduction

Any FTP client installed on a computer that is connected to the controller (Ethernet port), without EcoStruxure Machine Expert installed, can be used to transfer files to and from the data storage area of the controller.

NOTE: Schneider Electric adheres to industry best practices in the development and implementation of control systems. This includes a "Defense-in-Depth" approach to secure an Industrial Control System. This approach places the controllers behind one or more firewalls to restrict access to authorized personnel and protocols only.

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

- Evaluate whether your environment or your machines are connected to your critical infrastructure and, if so, take appropriate steps in terms of prevention, based on Defense-in-Depth, before connecting the automation system to any network.
- Limit the number of devices connected to a network to the minimum necessary.
- Isolate your industrial network from other networks inside your company.
- Protect any network against unintended access by using firewalls, VPN, or other, proven security measures.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Prepare a recovery plan including backup of your system and process information.

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

NOTE: Make use of the security-related commands *(see EcoStruxure Machine Expert, Menu Commands, Online Help)* which provide a way to add, edit, and remove a user in the online user management of the target device where you are currently logged in.

The FTP server is deactivated by default.

FTP Access

Access to the FTP server is controlled by User Rights when they are enabled in the controller. For more information, refer to **Users and Groups** Tab Description *(see page 78)*.

To access the FTP server you must first connect to the controller with EcoStruxure Machine Expert or Controller Assistant and modify the default user password.

Files Access

See File Organization *(see page 33)*.

FTP Client

Introduction

The FtpRemoteFileHandling library provides the following FTP client functionalities for remote file handling:

- Reading files
- Writing files
- Deleting files
- Listing content of remote directories
- Adding directories
- Removing directories

NOTE: Schneider Electric adheres to industry best practices in the development and implementation of control systems. This includes a "Defense-in-Depth" approach to secure an Industrial Control System. This approach places the controllers behind one or more firewalls to restrict access to authorized personnel and protocols only.

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

- Evaluate whether your environment or your machines are connected to your critical infrastructure and, if so, take appropriate steps in terms of prevention, based on Defense-in-Depth, before connecting the automation system to any network.
- Limit the number of devices connected to a network to the minimum necessary.
- Isolate your industrial network from other networks inside your company.
- Protect any network against unintended access by using firewalls, VPN, or other, proven security measures.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Prepare a recovery plan including backup of your system and process information.

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

For further information, refer to FtpRemoteFileHandling Library Guide.

SNMP

Introduction

The Simple Network Management Protocol (SNMP) is used to provide the data and services required for managing a network.

The data is stored in a Management Information Base (MIB). The SNMP protocol is used to read or write MIB data. Implementation of the Ethernet SNMP services is minimal, as only the compulsory objects are handled.

SNMP Server

This table presents the supported standard MIB-2 server objects:

Object	Description	Access	Value
sysDescr	Text description of the device	Read	SCHNEIDER M241-51 Fast Ethernet TCP/IP
sysName	Node administrative name	Read/Write	Controller reference

The size of these character strings is limited to 50 characters.

The values written are saved to the controller via SNMP client tool software. The Schneider Electric software for this is ConneXview. ConneXview is not supplied with the controller or bus coupler. For more details, refer to www.schneider-electric.com.

SNMP Client

The M241 Logic Controller supports an SNMP client library to allow you to query SNMP servers. For details, refer to the *SNMP Library Guide*.

Controller as a Target Device on EtherNet/IP

Introduction

This section describes the configuration of the M241 Logic Controller as an EtherNet/IP target device.

For further information about EtherNet/IP, refer to the *www.odva.org* website.

EtherNet/IP Target Configuration

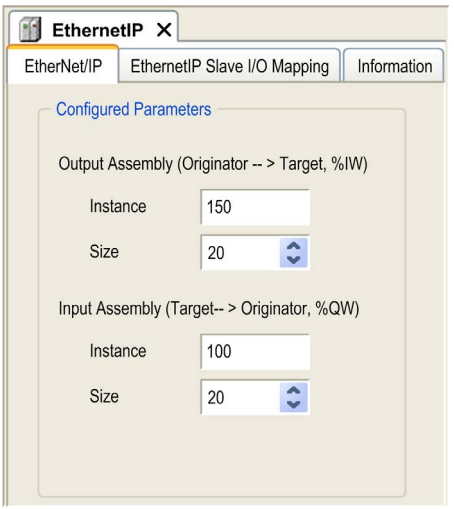
To configure your M241 Logic Controller as an EtherNet/IP target device, you must:

Step	Action
1	Select **EthernetIP** in the **Hardware Catalog**.
2	Drag and drop it to the **Devices tree** on one of the highlighted nodes. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)* • Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

EtherNet/IP Parameters Configuration

To configure the EtherNet/IP parameters, double-click **Ethernet_1 (Ethernet Network)** → **EthernetIP** in the Devices tree.

This dialog box is displayed:



The EtherNet/IP configuration parameters are defined as:

- **Instance**:
  Number referencing the input or output Assembly.
- **Size**:
  Number of channels of an input or output Assembly.
  The memory size of each channel is 2 bytes that stores the value of an %IWx or %QWx object, where x is the channel number.
  For example, if the **Size** of the **Output Assembly** is 20, it represents that there are 20 input channels (IW0...IW19) addressing %IWy...%IW(y+20-1), where y is the first available channel for the Assembly.

Element		Admissible Controller Range	EcoStruxure Machine Expert Default Value
Output Assembly	**Instance**	150...189	150
	Size	2...120	20
Input Assembly	**Instance**	100...149	100
	Size	2...120	20

EDS File Generation

You can generate an EDS file to configure EtherNet/IP cyclic data exchanges.

To generate the EDS file:

Step	Action
1	In the **Devices tree**, right-click the **EthernetIP** node and choose the **Export as EDS** command from the context menu.
2	Modify the default file name and location as required.
3	Click **Save**.

NOTE: The **Major Revision** and **Minor Revision** objects of the EDS file, defined in the file, are used to ensure uniqueness of the EDS file. The values of these objects do not reflect the actual controller revision level.

A generic EDS file for the M241 Logic Controller is also available on the Schneider website. You must adapt this file to your application by editing it and defining the required Assembly instances and sizes.

EthernetIP Slave I/O Mapping Tab

Variables can be defined and named in the **EthernetIP Slave I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

EthernetIP | EthernetIP Slave I/O Mapping | Information

Channels

Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
Input							Input
		IW0	%IW9	WORD			
		Bit0	%IX18.0	BOOL	FALSE		
		Bit1	%IX18.1	BOOL	FALSE		
		Bit2	%IX18.2	BOOL	FALSE		
		Bit3	%IX18.3	BOOL	FALSE		
		Bit4	%IX18.4	BOOL	FALSE		
		Bit5	%IX18.5	BOOL	FALSE		
		Bit6	%IX18.6	BOOL	FALSE		
		Bit7	%IX18.7	BOOL	FALSE		
		Bit8	%IX19.0	BOOL	FALSE		
		Bit9	%IX19.1	BOOL	FALSE		
		Bit10	%IX19.2	BOOL	FALSE		
		Bit11	%IX19.3	BOOL	FALSE		
		Bit12	%IX19.4	BOOL	FALSE		
		Bit13	%IX19.5	BOOL	FALSE		
		Bit14	%IX19.6	BOOL	FALSE		
		Bit15	%IX19.7	BOOL	FALSE		
		IW1	%IW10	WORD			
Output							Output
		QW0	%QW3	WORD			
		QW1	%QW4	WORD			
		QW2	%QW5	WORD			
		QW3	%QW6	WORD			
		QW4	%QW7	WORD			

The table below describes the **EthernetIP Slave I/O Mapping** configuration:

Channel		Type	Default Value	Description
Input	IW0	WORD	-	Command word of controller outputs (%QW)
	IWxxx			
Output	QW0	WORD	-	State of controller inputs (%IW)
	QWxxx			

The number of words depends on the size parameter configured in EtherNet/IP Target Configuration *(see page 149)*.

Output means OUTPUT from Originator controller (= %IW for the controller).

Input means INPUT from Originator controller (= %QW for the controller).

Connections on EtherNet/IP

To access a target device, an Originator opens a connection which can include several sessions that send requests.

One explicit connection uses one session (a session is a TCP or UDP connection).

One I/O connection uses 2 sessions.

The following table shows the EtherNet/IP connections limitations:

Characteristic	Maximum
Explicit connections	8 (Class 3)
I/O connections	1 (Class 1)
Connections	8
Sessions	16
Simultaneous requests	32

NOTE: The M241 Logic Controller supports cyclic connections only. If an Originator opens a connection using a change of state as a trigger, packets are sent at the RPI rate.

Profile

The controller supports the following objects:

Object class	Class ID (hex)	Cat.	Number of Instances	Effect on Interface Behavior
Identity Object *(see page 153)*	01	1	1	Supports the reset service
Message Router Object *(see page 156)*	02	1	1	Explicit message connection

Object class	Class ID (hex)	Cat.	Number of Instances	Effect on Interface Behavior
Assembly Object *(see page 157)*	04	2	2	Defines I/O data format
Connection Manager Object *(see page 159)*	06		1	–
TCP/IP Interface Object *(see page 162)*	F5	1	1	TCP/IP configuration
Ethernet Link Object *(see page 164)*	F6	1	1	Counter and status information
Interface Diagnostic Object *(see page 165)*	350	1	1	–
IOScanner Diagnostic Object *(see page 169)*	351	1	1	–
Connection Diagnostic Object *(see page 170)*	352	1	1	–
Explicit Connection Diagnostic Object *(see page 172)*	353	1	1	–
Explicit Connections Diagnostic List Object *(see page 173)*	354	1	1	–

Identity Object (Class ID = 01 hex)

The following table describes the class attributes of the Identity Object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Implementation revision of the Identity Object
2	Get	Max Instances	UINT	01	The largest instance number
3	Get	Number of Instances	UINT	01	The number of object instances
4	Get	Optional Instance Attribute List	UINT, UINT [ ]	00	The first 2 bytes contain the number of optional instance attributes. Each following pair of bytes represents the number of other optional instance attributes.
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	07	The largest instance attributes value

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
05	Reset (1)	Initializes EtherNet/IP component (controller reboot)
0E	Get Attribute Single	Returns the value of the specified attribute

(1) Reset Service description:

When the Identity Object receives a Reset request, it:
- determines whether it can provide the type of reset requested
- responds to the request
- attempts to perform the type of reset requested

The Reset common service has one specific parameter, Type of Reset (USINT), with the following values:

Value	Type of Reset
0	Reboots the controller **NOTE:** This is the default value if this parameter is omitted.
1	Not supported
2	Not supported
3...99	Reserved
100...199	Vendor specific
200...255	Reserved

The following table describes the Instance attributes:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Vendor ID	UINT	F3	Schneider Automation ID
2	Get	Device type	UINT	0E	Controller
3	Get	Product code	UINT	1001	Controller product code

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
4	Get	Revision	Struct of USINT, USINT	–	Product revision number of the controller (1). Equivalent to the 2 low bytes of the controller version
5	Get	Status	WORD	–	Status word(2)
6	Get	Serial number	UDINT	–	Serial number of the controller: XX + 3 LSB of MAC address
7	Get	Product name	Struct of USINT, STRING	–	–

(1) Mapped in a WORD:
- MSB: minor revision (second USINT)
- LSB: major revision (first USINT)

Example: 0205 hex means revision V5.2.

(2) Status word (Attribute 5):

Bit	Name	Description
0	Owned	Unused
1	Reserved	–
2	Configured	TRUE indicates the device application has been reconfigured.
3	Reserved	–
4...7	Extended Device Status	• 0: Self-testing or undetermined • 1: Firmware update in progress • 2: At least one invalid I/O connection detected • 3: No I/O connections established • 4: Non-volatile configuration invalid • 5: Unrecoverable error detected • 6: At least one I/O connection in RUNNING state • 7: At least one I/O connection established, all in idle mode • 8: Reserved • 9...15: Unused
8	Minor Recoverable Fault	TRUE indicates the device detected an error, which, under most circumstances, is recoverable. This type of event does not lead to a change in the device state.
9	Minor Unrecoverable Fault	TRUE indicates the device detected an error, which, under most circumstances, is unrecoverable. This type of event does not lead to a change in the device state.

Bit	Name	Description
10	Major Recoverable Fault	TRUE indicates the device detected an error, which requires the device to report an exception and enter into the HALT state. This type of event leads to a change in the device state, but, under most circumstances, is recoverable.
11	Major Unrecoverable Fault	TRUE indicates the device detected an error, which requires the device to report an exception and enter into the HALT state. This type of event leads to a change in the device state, but, under most circumstances, is not recoverable.
12...15	Reserved	–

Message Router Object (Class ID = 02 hex)

The following table describes the class attributes of the Message Router object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Implementation revision number of the Message Router Object
2	Get	Max Instances	UINT	02	The largest instance number
3	Get	Number of Instance	UINT	01	The number of object instances
4	Get	Optional Instance Attribute List	Struct of UINT, UINT [ ]	02	The first 2 bytes contain the number of optional instance attributes. Each following pair of bytes represents the number of other optional instance attributes (from 100 to 119).
5	Get	Optional Service List	UINT	0A	The number and list of any implemented optional services attribute (0: no optional services implemented)
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	02	The largest instance attributes value

The following table describes the Class services:

Service Code (hex)	Name	Description
01	Get_Attribute_All	Returns the value of all class attributes
0E	Get_Attribute_Single	Returns the value of the specified attribute

The following table describes the Instance services:

Service Code (hex)	Name	Description
01	Get_Attribute_All	Returns the value of all class attributes
0E	Get_Attribute_Single	Returns the value of the specified attribute

The following table describes the Instance attributes:

Attribute ID (hex)	Access	Name	Data Type	Value	Description
1	Get	Implemented Object List	Struct of UINT, UINT [ ]	–	Implemented Object list. The first 2 bytes contain the number of implemented objects. Each 2 bytes that follow represents another implemented class number. This list contains the following objects: ● Identity ● Message Router ● Assembly ● Connection Manager ● Parameter ● File Object ● Modbus ● Port ● TCP/IP ● Ethernet Link
2	Get	Number available	UINT	512	Maximum number of concurrent CIP (Class 1 or Class 3) connections supported

Assembly Object (Class ID = 04 hex)

The following table describes the class attributes of the Assembly object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	02	Implementation revision of the Assembly Object
2	Get	Max Instances	UINT	BE	The largest instance number
3	Get	Number of Instances	UINT	03	The number of object instances

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
4	Get	Optional Instance Attribute List	Struct of: UINT UINT [ ]	 01 04	The first 2 bytes contain the number of optional instance attributes. Each following pair of bytes represents the number of other optional instance attributes.
5	Get	Optional Service List	UINT	Not supported	The number and list of any implemented optional services attribute (0: no optional services implemented)
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	04	The largest instance attributes value

The following table describes the Class Services:

Service Code (hex)	Name	Description
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code (hex)	Name	Description
0E	Get Attribute Single	Returns the value of the specified attribute
10	Set Attribute Single	Modifies the value of the specified attribute

Instances Supported

Output means OUTPUT from Originator controller (= %IW for the controller).

Input means INPUT from Originator controller (= %QW for the controller).

The controller supports 2 Assemblies:

Name	Instance	Data Size
Controller Output (%IW)	Configurable: must be between 100 and 149	2...40 words
Controller Input (%QW)	Configurable: must be between 150 and 189	2...40 words

NOTE: The Assembly object binds together the attributes of multiple objects so that information to or from each object can be communicated over a single connection. Assembly objects are static. The Assemblies in use can be modified through the parameter access of the network configuration tool (RSNetWorx). The controller needs to recycle power to register a new Assembly assignment.

The following table describes the Instance attributes:

Attribute ID (hex)	Access	Name	Data Type	Value	Description
3	Get/Set	Instance Data	ARRAY of Byte	–	Data Set service only available for Controller output
4	Get	Instance Data Size	UINT	4...80	Size of data in byte

Access from a EtherNet/IP Scanner

When a EtherNet/IP Scanner needs to exchange assemblies with a M241 Logic Controller, it uses the following access parameters (`Connection path`):

- Class 4
- Instance xx where xx is the instance value (example: 2464 hex = instance 100).
- Attribute 3

In addition, a configuration assembly must be defined in the Originator.

For example: Class 4, Instance 3, Attribute 3, the resulting `Connection Path` will be:

- 2004 hex
- 2403 hex
- 2c<xx> hex

Connection Manager Object (Class ID = 06 hex)

The following table describes the class attributes of the Assembly Object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Implementation revision of the Connection Manager Object
2	Get	Max Instances	UINT	01	The largest instance number
3	Get	Number of Instances	UINT	01	The number of object instances

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
4	Get	Optional Instance Attribute List	Struct of: UINT UINT [ ]	–	The number and list of the optional attributes. The first word contains the number of attributes to follow and each following word contains another attribute code. Following optional attributes include: ● total number of incoming connection open requests ● the number of requests rejected due to non-conforming format of the Forward Open ● the number of requests rejected because of insufficient resources ● the number of requests rejected due to parameter value sent with the Forward Open ● the number of Forward Close requests received ● the number of Forward Close requests with an invalid format ● the number of Forward Close requests that could not be matched to an active connection ● the number of connections that have timed out because the other side stopped producing, or a network disconnection occurred
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	08	The largest instance attributes value

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all instance attributes
0E	Get Attribute Single	Returns the value of the specified attribute
4E	Forward Close	Closes an existing connection
52	Unconnected Send	Sends a multi-hop unconnected request
54	Forward Open	Opens a new connection

The following table describes the Instance attributes:

Attribute ID (hex)	Access	Name	Data Type	Value	Description
1	Get	Open Requests	UINT	–	Number of Forward Open service requests received
2	Get	Open Format Rejects	UINT	–	Number of Forward Open service requests which were rejected due to invalid format
3	Get	Open Resource Rejects	ARRAY of Byte	–	Number of Forward Open service requests which were rejected due to lack of resources
4	Get	Open Other Rejects	UINT	–	Number of Forward Open service requests which were rejected for reasons other than invalid format or lack of resources
5	Get	Close Requests	UINT	–	Number of Forward Close service requests received
6	Get	Close Format Requests	UINT	–	Number of Forward Close service requests which were rejected due to invalid format
7	Get	Close Other Requests	UINT	–	Number of Forward Close service requests which were rejected for reasons other than invalid format
8	Get	Connection Timeouts	UINT	–	Total number of connection timeouts that have occurred in connections controlled by this Connection Manager

TCP/IP Interface Object (Class ID = F5 hex)

This object maintains link specific counters and status information for an Ethernet 802.3 communications interface.

The following table describes the class attributes of the TCP/IP Interface Object:

Attribute ID (hex)	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	4	Implementation revision of the TCP/IP Interface Object
2	Get	Max Instances	UINT	2	The largest instance number
3	Get	Number of Instances	UINT	2	The number of object instances

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

Instance Codes

Only instance 1 is supported.

The following table describes the Instance Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all instance attributes
0E	Get Attribute Single	Returns the value of the specified instance attribute

The following table describes the Instance Attributes:

Attribute ID (hex)	Access	Name	Data Type	Value	Description
1	Get	Status	DWORD	Bit level	• 0: The interface configuration attribute has not been configured. • 1: The interface configuration contains a valid configuration. • 2...15: Reserved.

Attribute ID (hex)	Access	Name	Data Type	Value	Description
2	Get	Configuration Capability	DWORD	Bit level	• 0: BOOTP Client • 1: DNS Client • 2: DHCP Client • 5: Configured in EcoStruxure Machine Expert All other bits are reserved and set to 0.
3	Get	Configuration	DWORD	Bit level	• 0: The interface configuration is valid. • 1: The interface configuration is obtained with BOOTP. • 2: The interface configuration is obtained with DHCP. • 3: reserved • 4: DNS Enable All other bits are reserved and set to 0.
4	Get	Physical Link	UINT	Path size	Number of 16 bits word in the element Path
			Padded EPATH	Path	Logical segments identifying the physical link object. The path is restricted to one logical class segment and one logical instance segment. The maximum size is 12 bytes.
5	Get	Interface configuration	UDINT	IP Address	–
			UDINT	Network Mask	–
			UDINT	Gateway Address	–
			UDINT	Primary Name	–
			UDINT	Secondary Name	0: no secondary name server address has been configured.
			STRING	Default Domain Name	0: no Domain Name is configured
6	Get	Host Name	STRING	–	ASCII characters. 0: no host name is configured

Ethernet Link Object (Class ID = F6 hex)

This object provides the mechanism to configure a TCP/IP network interface device.

The following table describes the class attributes of the Ethernet Link object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	4	Implementation revision of the Ethernet Link Object
2	Get	Max Instances	UINT	3	The largest instance number
3	Get	Number of Instances	UINT	3	The number of object instances

The following table describes the class services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

Instance Codes

Only instance 1 is supported.

The following table describes the instance services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all instance attributes
0E	Get Attribute Single	Returns the value of the specified instance attribute

The following table describes the instance attributes:

Attribute ID (hex)	Access	Name	Data Type	Value	Description
1	Get	Interface Speed	UDINT	–	Speed in Mbit/s (10 or 100)
2	Get	Interface Flags	DWORD	Bit level	• 0: link status • 1: half/full duplex • 2...4: negotiation status • 5: manual setting / requires reset • 6: local hardware error detected All other bits are reserved and set to 0.

Attribute ID (hex)	Access	Name	Data Type	Value	Description
3	Get	Physical Address	ARRAY of 6 USINT	–	This array contains the MAC address of the product. Format: XX-XX-XX-XX-XX-XX

EtherNet/IP Interface Diagnostic Object (Class ID = 350 hex)

The following table describes the class attributes of the EtherNet/IP Interface Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Increased by 1 on each new update of the object
2	Get	Max Instance	UINT	01	Maximum instance number of the object

The following table describes the instance attributes of the EtherNet/IP Interface Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Details
1	Get	Protocols supported	UINT	Protocol(s) supported (0=not supported, 1=supported): • Bit 0: EtherNet/IP • Bit 1: Modbus TCP • Bit 2: Modbus Serial • Bits 3...15: Reserved, 0

Attribute ID (hex)	Access	Name	Data Type	Details
2	Get	Connection Diag	STRUCT of	
		Max CIP IO Connections opened	UINT	Maximum number of CIP I/O connections opened.
		Current CIP IO Connections	UINT	Number of CIP I/O connections currently opened.
		Max CIP Explicit Connections opened	UINT	Maximum number of CIP explicit connections opened.
		Current CIP Explicit Connections	UINT	Number of CIP explicit connections currently opened
		CIP Connections Opening Errors	UINT	Incremented on each unsuccessful attempt to open a CIP connection.
		CIP Connections Timeout Errors	UINT	Incremented when a CIP connection times out.
		Max EIP TCP Connections opened	UINT	Maximum number of TCP connections opened and used for EtherNet/IP communications.
		Current EIP TCP Connections	UINT	Number of TCP connections currently open and being used for EtherNet/IP communications.
3	Get Clear	IO Messaging Diag	STRUCT of	
		IO Production Counter	UDINT	Incremented each time a Class 0/1 CIP message is sent.
		IO Consumption Counter	UDINT	Incremented each time a Class 0/1 CIP message is received.
		IO Production Send Errors Counter	UINT	Incremented each Time a Class 0/1 message is not sent.
		IO Consumption Receive Errors Counter	UINT	Incremented each time a consumption is received that contains an error.
4	Get Clear	Explicit Messaging Diag	STRUCT of	
		Class3 Msg Send Counter	UDINT	Incremented each time a Class 3 CIP message is sent.
		Class3 Msg Receive Counter	UDINT	Incremented each time a Class 3 CIP message is received.
		UCMM Msg Send Counter	UDINT	Incremented each time a UCMM message is sent.
		UCMM Msg Receive Counter	UDINT	Incremented each time a UCMM message is received.

Attribute ID (hex)	Access	Name	Data Type	Details
5	Get	Com Capacity	STRUCT of	
		Max CIP Connections	UINT	Maximum number of supported CIP connections.
		Max TCP Connections	UINT	Maximum number of supported TCP connections.
		Max Urgent priority rate	UINT	Maximum number of CIP transport class 0/1 Urgent priority message packets per second.
		Max Scheduled priority rate	UINT	Maximum number of CIP transport class 0/1 Scheduled priority message packets per second.
		Max High priority rate	UINT	Maximum number of CIP transport class 0/1 High priority message packets per second.
		Max Low priority rate	UINT	Maximum number of CIP transport class 0/1 Low priority message packets per second.
		Max Explicit Messaging rate	UINT	Max CIP transport class 2/3 or other EtherNet/IP messages packets per second

Attribute ID (hex)	Access	Name	Data Type	Details
6	Get	Bandwidth Diag	STRUCT of	
		Current sending Urgent priority rate	UINT	CIP transport class 0/1 Urgent priority message packets sent per second.
		Current reception Urgent priority rate	UINT	CIP transport class 0/1 Urgent priority message packets received per second.
		Current sending Scheduled priority rate	UINT	CIP transport class 0/1 Scheduled priority message packets sent per second.
		Current reception Scheduled priority rate	UINT	CIP transport class 0/1 Scheduled priority message packets received per second.
		Current sending High priority rate	UINT	CIP transport class 0/1 High priority message packets sent per second.
		Current reception High priority rate	UINT	CIP transport class 0/1 High priority message packets received per second.
		Current sending Low priority rate	UINT	CIP transport class 0/1 Low priority message packets sent per second.
		Current reception Low priority rate	UINT	CIP transport class 0/1 Low priority message packets received per second.
		Current sending Explicit Messaging rate	UINT	CIP transport class 2/3 or other EtherNet/IP message packets sent per second.
		Current reception Explicit Messaging rate	UINT	CIP transport class 2/3 or other EtherNet/IP message packets received per second.
7	Get	Modbus Diag	STRUCT of	
		Max. Modbus TCP Connections opened	UINT	Maximum number of TCP connections opened and used for Modbus communications.
		Current Modbus TCP Connections	UINT	Number of TCP connections currently opened and used for Modbus communications.
		Modbus TCP Msg Send Counter	UDINT	Incremented each time a Modbus TCP message is sent.
		Modbus TCP Msg Receive Counter	UDINT	Incremented each time a Modbus TCP message is received.

The following table describes the class services:

Service Code (hex)	Name	Description
01	Get_Attributes_All	Returns the value of all class attributes.
0E	Get_Attribute_Single	Returns the value of a specified attribute.
4C	Get_and_Clear	Gets and clears a specified attribute.

IOScanner Diagnostic Object (Class ID = 351 hex)

The following table describes the class attributes of the IOScanner Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	1	Increased by 1 on each new update of the object.
2	Get	Max Instance	UINT	1	Maximum instance number of the object.

The following table describes the instance attributes of the IOScanner Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Details
1	Get	IO Status Table	STRUCT of	
		Size	UINT	Size in bytes of the Status attribute.
		Status	ARRAY of UINT	I/O status. Bit n, where n is instance n of the object, provides the status of the I/O exchanged on the I/O connection: ● 0: The input or output status of the I/O connection is in error, or no device. ● 1: The input or output status of the I/O connection is correct.

The following table describes the class services:

Service Code (hex)	Name	Description
01	Get_Attributes_All	Returns the value of all class attributes.

IO Connection Diagnostic Object (Class ID = 352 hex)

The following table describes the class attributes of the IO Connection Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Increased by 1 on each new update of the object.
2	Get	Max Instance	UINT	01	Maximum instance number of the object 0...n where n is the maximum number of CIP I/O connections. **NOTE:** There is an IO Connection Diagnostic object instance for both O->T and T->O paths.

The following table describes the instance attributes of the I/O Connection Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Details
1	Get Clear	IO Com Diag	STRUCT of	
		IO Production Counter	UDINT	Incremented each time a production is sent.
		IO Consumption Counter	UDINT	Incremented each time a consumption is received.
		IO Production Send Errors Counter	UINT	Incremented each time a production is not sent due to an error.
		IO Consumption Receive Errors Counter	UINT	Incremented each time a consumption is received that contains an error.
		CIP Connection TimeOut Errors	UINT	Incremented each time a connection times out.
		CIP Connection Opening Errors	UINT	Incremented on each unsuccessful attempt to open a connection.
		CIP Connection State	UINT	State of the CIP IO connection.
		CIP Last Error General Status	UINT	General status of the last error detected on the connection.
		CIP Last Error Extended Status	UINT	Extended status of the last error detected on the connection.
		Input Com Status	UINT	Communication status of the inputs.
		Output Com Status	UINT	Communication status of the outputs.

Attribute ID (hex)	Access	Name	Data Type	Details
2	Get	Connection Diag	STRUCT of	
		Production Connection ID	UDINT	Connection ID for production.
		Consumption Connection ID	UDINT	Connection ID for consumption.
		Production RPI	UDINT	Requested Packet Interval (RPI) for productions, in µs.
		Production API	UDINT	Actual Packet Interval (API) for productions.
		Consumption RPI	UDINT	RPI for consumptions.
		Consumption API	UDINT	API for consumptions.
		Production Connection Parameters	UDINT	Connection parameters for productions.
		Consumption Connection Parameters	UDINT	Connection parameters for consumptions.
		Local IP	UDINT	Local IP address for I/O communication.
		Local UDP Port	UINT	Local UDP port number for I/O communication.
		Remote IP	UDINT	Remote IP address for I/O communication.
		Remote UDP Port	UINT	Remote UDP port number for I/O communication.
		Production Multicast IP	UDINT	Multicast IP address for productions, or 0 if multicast is not used.
		Consumption Multicast IP	UDINT	Multicast IP address for consumptions, or 0 if multicast is not used.
		Protocols supported	UINT	Protocol(s) supported (0=not supported, 1=supported): ● Bit 0: EtherNet/IP ● Bit 1: Modbus TCP ● Bit 2: Modbus Serial ● Bits 3...15: Reserved, 0

Instance Attributes

The following table describes the class services:

Service Code (hex)	Name	Description
01	Get_Attributes_All	Returns the value of all class attributes.
0E	Get_Attribute_Single	Returns the value of the specified attribute.
4C	Get_and_Clear	Gets and clears a specified attribute.

Explicit Connection Diagnostic Object (Class ID = 353 hex)

The following table describes the class attributes of the Explicit Connection Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Increased by 1 at each new update of the object.
2	Get	Max Instance	UINT	0...n (maximum number of CIP IO connections)	Maximum instance number of the object.

The following table describes the instance attributes of the Explicit Connection Diagnostic object:

Attribute ID (hex)	Access	Name	Data Type	Details
1	Get	Originator Connection ID	UDINT	O to T Connection ID
2	Get	Originator IP	UDINT	
3	Get	Originator TCP Port	UINT	
4	Get	Target Connection ID	UDINT	T to O Connection ID
5	Get	Target IP	UDINT	
6	Get	Target TCP Port	UINT	
7	Get	Msg Send Counter	UDINT	Incremented each time a Class 3 CIP Message is sent on the connection
8	Get	Msg ReceiveCounter	UDINT	Incremented each time a Class 3 CIP Message is received on the connection

Explicit Connections Diagnostic List Object (Class ID = 354 hex)

The following table describes the class attributes of the Explicit Connections Diagnostic List object:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Increased by 1 at each new update of the object.
2	Get	Max Instance	UINT	0...n	n is the maximum number of concurrent list accesses supported.

The following table describes the instance attributes of the Explicit Connections Diagnostic List object:

Attribute ID (hex)	Access	Name	Data Type	Details
1	Get	Number of Connections	UINT	Total number of open Explicit connections
2	Get	Explicit Messaging Connections Diagnostic List	ARRAY of STRUCT	Contents of instantiated Explicit Connection Diagnostic objects
		Originator Connection ID	UDINT	Originator to Target connection ID
		Originator IP	UDINT	Originator to Target IP address
		Originator TCP Port	UINT	Originator to Target port number
		Target Connection ID	UDINT	Target to Originator connection ID
		Target IP	UDINT	Target to Originator IP address
		Target TCP Port	UINT	Target to Originator port number
		Msg Send Counter	UDINT	Incremented each time a Class 3 CIP message is sent on the connection
		Msg Receive Counter	UDINT	Incremented each time a Class 3 CIP message is sent on the connection

The following table describes the class services:

Service Code (hex)	Name	Description
08	Create	Creates an instance of the Explicit Connections Diagnostic List object.
09	Delete	Deletes an instance of the Explicit Connections Diagnostic List object.
33	Explicit_Connections_ Diagnostic_Read	Explicit corrections diagnostic read object.

Controller as a Slave Device on Modbus TCP

Overview

This section describes the configuration of the M241 Logic Controller as a **Modbus TCP Slave Device**.

The **Modbus TCP Slave Device** adds another Modbus server function to the controller. This server is addressed by the Modbus client application by specifying a configured Unit ID (Modbus address) in the range 1...247. The embedded Modbus server of the slave controller needs no configuration, and is addressed by specifying a Unit ID equal to 255. Refer to Modbus TCP Configuration *(see page 176)*.

To configure your M241 Logic Controller as a **Modbus TCP Slave Device**, you must add **Modbus TCP Slave Device** functionality to your controller (see Adding a Modbus TCP Slave Device thereafter). This functionality creates a specific I/O area in the controller that is accessible with the Modbus TCP protocol. This I/O area is used whenever an external master needs to access the `%IW` and `%QW` objects of the controller. This **Modbus TCP Slave Device** functionality allows you to furnish to this area the controller I/O objects which can then be accessed with a single Modbus read/write registers request.

Inputs/outputs are seen from the slave controller: inputs are written by the master, and outputs are read by the master.

The **Modbus TCP Slave Device** can define a privileged Modbus client application, whose connection is not forcefully closed (embedded Modbus connections may be closed when more than 8 connections are needed).

The watchdog associated to the privileged connection allows you to verify whether the controller is being polled by the privileged master. If no Modbus request is received within the timeout duration, the diagnostic information `i_byMasterIpLost` is set to 1 (TRUE). For more information, refer to the Ethernet Port Read-Only System Variables *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*.

For further information about Modbus TCP, refer to the *www.odva.org* website.

Adding a Modbus TCP Slave Device

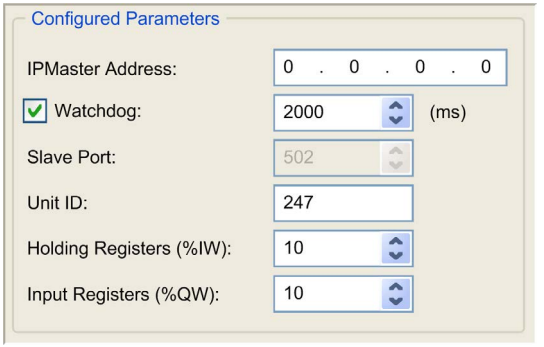
To configure your M241 Logic Controller as a Modbus TCP slave device, you must:

Step	Action
1	Add a TM4ES4 expansion module to your configuration. To do this, you must have added the **Industrial_Ethernet_manager** to your logic controller.
2	Select **Modbus TCP Slave Device** in the **Hardware Catalog**.
3	Drag and drop it to the **Devices tree** on one of the highlighted nodes. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)* • Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Modbus TCP Configuration

To configure the Modbus TCP slave device, double-click **Ethernet_1** → **ModbusTCP_Slave_Device** in the **Devices tree**.

This dialog box appears:



Element	Description
IP Master Address	IP address of the Modbus master The connections are not closed on this address.
Watchdog	Watchdog in 500 ms increments **NOTE:** The watchdog applies to the IP master Address unless the address is 0.0.0.0.
Slave Port	Modbus communication port (502) **NOTE:** The port number can be modified using the changeModbusPort script command *(see page 188)*.
Unit ID	Sends the requests to the Modbus TCP slave device (1...247), instead of to the embedded Modbus server (255).
Holding Registers (%IW)	Number of %IW registers to be used in the exchange (2...120) (each register is 2 bytes)
Input Registers (%QW)	Number of %QW registers to be used in the exchange (2...120) (each register is 2 bytes)

Modbus TCP Slave Device I/O Mapping Tab

The I/Os are mapped to Modbus registers from the master perspective as follows:

- %IWs are mapped from register 0 to n-1 and are R/W (n = Holding register quantity, each %IW register is 2 bytes).
- %QWs are mapped from register n to n+m -1 and are read only (m = Input registers quantity, each %QW register is 2 bytes).

Once a **Modbus TCP Slave Device** has been configured, Modbus commands sent to its Unit ID (Modbus address) are handled differently than the same command would be when addressed to any other Modbus device on the network. For example, when the Modbus command 3 (3 hex) is sent to a standard Modbus device, it reads and returns the value of one or more registers. When this same command is sent to the Modbus TCP *(see page 127)* Slave, it facilitates a read operation by the external I/O scanner.

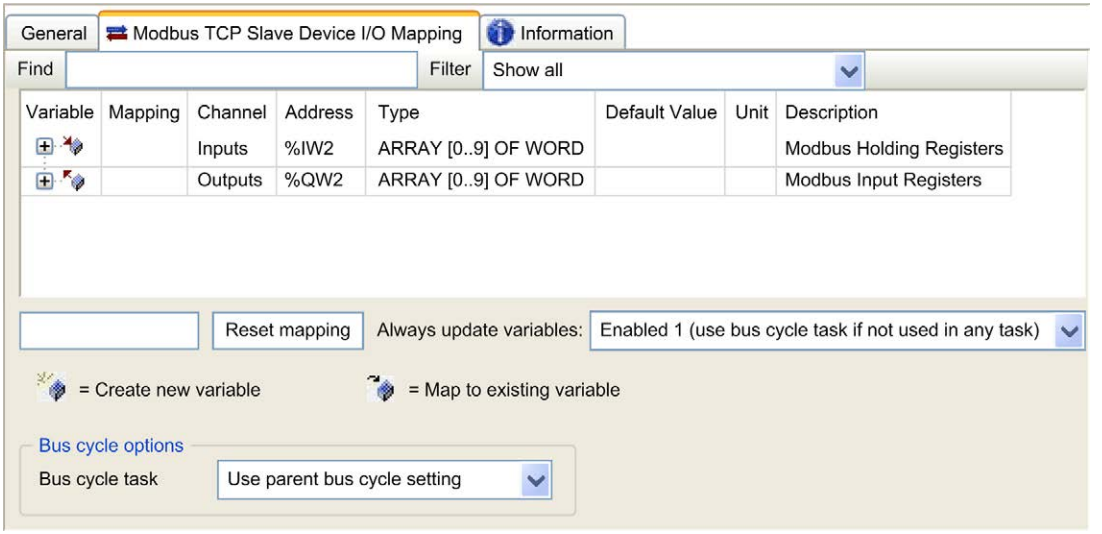
Once a **Modbus TCP Slave Device** has been configured, Modbus commands sent to its Unit ID (Modbus address) access the `%IW` and `%QW` objects of the controller instead of the regular Modbus words (accessed when the Unit ID is 255). This facilitates read/write operations by a Modbus TCP IOScanner application.

The **Modbus TCP Slave Device** responds to a subset of the Modbus commands with the purpose of exchanging data with the external I/O scanner. The following Modbus commands are supported by the Modbus TCP slave device:

Function Code Dec (Hex)	Function	Comment
3 (3)	Read holding register	Allows the master to read %IW and %QW objects of the device
6 (6)	Write single register	Allows the master to write %IW objects of the device
16 (10)	Write multiple registers	Allows the master to write %IW objects of the device
23 (17)	Read/write multiple registers	Allows the master to read %IW and %QW objects of the device and write %IW objects of the device
Other	Not supported	–

NOTE: Modbus requests that attempt to access registers above n+m-1 are answered by the 02 - ILLEGAL DATA ADDRESS exception code.

To link I/O objects to variables, select the **Modbus TCP Slave Device I/O Mapping** tab:



Channel		Type	Description
Input	IW0	WORD	Holding register 0
	...	...	...
	IWx	WORD	Holding register x
Output	QW0	WORD	Input register 0
	...	...	...
	QWy	WORD	Input register y

The number of words depends on the **Holding Registers (%IW)** and **Input Registers (%QW)** parameters of the **Modbus TCP** tab.

NOTE: Output means OUTPUT from Originator controller (= %IW for the controller). Input means INPUT from Originator controller (= %QW for the controller).

NOTE: The **Modbus TCP Slave Device** refreshes the `%IW` and `%QW` registers as a single time-consistent unit, synchronized with the IEC tasks (MAST task by default). By contrast, the embedded Modbus TCP server only ensures time-consistency for 1 word (2 bytes). If your application requires time-consistency for more than 1 word (2 bytes), use the **Modbus TCP Slave Device**.

The parameter **Always update variables** is set to **Enabled 1 (use bus cycle task if not used in any task)** and is not editable.

Bus Cycle Options

In the **Modbus TCP Slave Device I/O Mapping** tab, select the **Bus cycle task** to use:

- **Use parent bus cycle setting** (the default),
- **MAST**
- **An existing task of the project**: you can select an existing task and associate it to the scanner. For more information about the application tasks, refer to the EcoStruxure Machine Expert Programming Guide.

NOTE: There is a corresponding **Bus cycle task** parameter in the I/O mapping editor of the device that contains the **Modbus TCP Slave Device**. This parameter defines the task responsible for refreshing the `%IW` and `%QW` registers.

Changing the Modbus TCP Port

changeModbusPort Command

The `changeModbusPort` command can be used to change the port used for data exchanges with a Modbus TCP master.

The current Modbus **Slave Port** is displayed on the Modbus TCP configuration window *(see page 176)*.

The default Modbus port number is 502.

Command	Description
`changeModbusPort` *"portnum"*	*portnum* is the new Modbus port number to use is passed as a string of characters. Before running the command, refer to Used Ports *(see page 192)* to ensure that *portnum* is not being used by any other TCP/UDP protocols or processes. An error is logged in the */usr/Syslog/FWLog.txt* file if the specified port number is already in use.

To limit the number of open sockets, the `changeModbusPort` command can only be run twice.

A power cycle of the logic controller returns the Modbus port number to the default value (502). The `changeModbusPort` command must therefore be executed after each power cycle.

NOTE: After changing the port number, the **Modbus Server Active** checkbox on the Ethernet Configuration window *(see page 123)* is no longer taken into account, as the Modbus server always uses port 502.

Running the Command from an SD Card Script

Step	Action

1	Create a script file *(see page 252)*, for example:
	`; Change Modbus slave port`
	`changeModbusPort "1502";`

2	Name the script file *Script.cmd*.
3	Copy the script file to the SD card.
4	Insert the SD card in the controller.

Running the Command Using ExecuteScript

The `changeModbusPort` command can be run from within an application using the ExecuteScript function block *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*.

The following sample code changes the Modbus TCP slave port from the default (502) to 1502.

```
IF (myBExe = FALSE AND (PortNum <> 502)) THEN

   myExecSc( // falling edge for a second change
   xExecute:=FALSE ,
   sCmd:=myCmd ,
   xDone=>myBDone ,
   xBusy=> myBBusy,
   xError=> myBErr,
   eError=> myIerr);
   string1 := 'changeModbusPort "';
   string2 := WORD_TO_STRING(PortNum);
   myCmd := concat(string1,string2);
   myCmd := concat(myCmd,'"');
   myBExe := TRUE;
END_IF

myExecSc(
xExecute:=myBExe ,
sCmd:=myCmd ,
xDone=>myBDone ,
xBusy=> myBBusy,
xError=> myBErr,
eError=> myIerr);
```

Section 13.2
Firewall Configuration

Introduction

This section describes how to configure the firewall of the Modicon M241 Logic Controller.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction	183
Dynamic Changes Procedure	185
Firewall Behavior	186
Firewall Script Commands	188

Introduction

Firewall Presentation

In general, firewalls help protect network security zone perimeters by blocking unauthorized access and permitting authorized access. A firewall is a device or set of devices configured to permit, deny, encrypt, decrypt, or proxy traffic between different security zones based upon a set of rules and other criteria.

Process control devices and high-speed manufacturing machines require fast data throughput and often cannot tolerate the latency introduced by an aggressive security strategy inside the control network. Firewalls, therefore, play a significant role in a security strategy by providing levels of protection at the perimeters of the network. Firewalls are an important part of an overall, system level strategy. By default, firewall rules do not allow the transfer of incoming IP telegrams from a controller network to a fieldbus network.

NOTE: Schneider Electric adheres to industry best practices in the development and implementation of control systems. This includes a "Defense-in-Depth" approach to secure an Industrial Control System. This approach places the controllers behind one or more firewalls to restrict access to authorized personnel and protocols only.

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

- Evaluate whether your environment or your machines are connected to your critical infrastructure and, if so, take appropriate steps in terms of prevention, based on Defense-in-Depth, before connecting the automation system to any network.
- Limit the number of devices connected to a network to the minimum necessary.
- Isolate your industrial network from other networks inside your company.
- Protect any network against unintended access by using firewalls, VPN, or other, proven security measures.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Prepare a recovery plan including backup of your system and process information.

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

Firewall Configuration

There are three ways to manage the controller firewall configuration:

- Static configuration
- Dynamic changes
- Application settings

Script files are used in the static configuration and for dynamic changes.

Static Configuration

The static configuration is loaded at the controller boot.

The controller firewall can be statically configured by managing a default script file located in the controller. The path to this file is */usr/Cfg/FirewallDefault.cmd*.

Dynamic Changes

After the controller boot, the controller firewall configuration can be changed by the use of script files.

There are two ways to load these dynamic changes using:

- A physical SD card *(see page 185)*.
- A function block *(see page 185)* in the application.

Application Settings

See Ethernet Configuration *(see page 123)*.

Dynamic Changes Procedure

Using an SD Card

This table describes the procedure to execute a script file from an SD card:

Step	Action
1	Create a valid script file *(see page 188)*. For example, name the script file *FirewallMaintenance.cmd*.
2	Load the script file on the SD card. For example, load the script file in the *usr/Cfg* folder.

3	In the file *Sys/Cmd/Script.cmd*, add a code line with the command
	`Firewall_install "/pathname/FileName"`
	For example, the code line is
	`Firewall_install "/sd0/usr/Cfg/FirewallMaintenance.cmd"`

4	Insert the SD card on the controller.

Using a Function Block in the Application

This table describes the procedure to execute a script file from an application:

Step	Action
1	Create a valid script file *(see page 188)*. For example, name the script file *FirewallMaintenance.cmd*.
2	Load the script file in the controller memory. For example, load the script file in the *usr/Syslog* folder with FTP.
3	Use an ExecuteScript *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* function block. For example, the **[SCmd]** input is `'Firewall_install "/usr/Syslog/FirewallMaintenance.cmd"'`

Firewall Behavior

Introduction

The firewall configuration depends on the action done on the controller and the initial configuration state. There are five possible initial states:

- There is no default script file in the controller.
- A correct script file is present.
- An incorrect script file is present.
- There is no default script file and the application has configured the firewall.
- A dynamic script file configuration has already been executed.

No Default Script File

If...	Then ...
Boot of the controller	Firewall is not configured. No protection is activated.
Execute dynamic script file	Firewall is configured according to the dynamic script file.
Execute dynamic incorrect script file	Firewall is not configured. No protection is activated.
Download application	Firewall is configured according to the application settings.

Default Script File Present

If...	Then ...
Boot of the controller	Firewall is configured according to the default script file.
Execute dynamic script file	The whole configuration of the default script file is deleted. Firewall is configured according to the dynamic script file.
Execute dynamic incorrect script file	Firewall is configured according to the default script file. The dynamic script file is not taken into account.
Download application	The whole configuration of the application is ignored. Firewall is configured according to the default script file.

Incorrect Default Script File Present

If...	Then ...
Boot of the controller	Firewall is not configured. No protection is activated
Execute dynamic script file	Firewall is configured according to the dynamic script file.
Execute dynamic incorrect script file	Firewall is not configured. No protection is activated.
Download application	Firewall is configured according to the application settings.

Application Settings with No Default Script File

If...	Then ...
Boot of the controller	Firewall is configured according to the application settings.
Execute dynamic script file	The whole configuration of the application settings is deleted. Firewall is configured according to the dynamic script file.
Execute dynamic incorrect script file	Firewall is configured according to the application settings. The dynamic script file is not taken into account.
Download application	The whole configuration of the previous application is deleted. Firewall is configured according to the new application settings.

Execute Dynamic Script File Already Executed

If...	Then ...
Boot of the controller	Firewall is configured according to the dynamic script file configuration (see note).
Execute dynamic script file	The whole configuration of the previous dynamic script file is deleted. Firewall is configured according to the new dynamic script file.
Execute dynamic incorrect script file	Firewall is configured according to the previous dynamic script file configuration. The dynamic incorrect script file is not taken into account.
Download application	The whole configuration of the application is ignored Firewall is configured according to the dynamic script file.
NOTE: If an SD card containing a cybersecurity script is plugged into the controller, booting is blocked. First remove the SD card to correctly boot the controller.	

Firewall Script Commands

Overview

This section describes how script files (default script files or dynamic script files) are written so that they can be executed during the booting of the controller or during a specific command triggered.

NOTE: The MAC layer rules are managed separately and have more priority over other packet filter rules.

Script File Syntax

The syntax of script files is described in Script Syntax Guidelines *(see page 252)*.

General Firewall Commands

The following commands are available to manage the Ethernet firewall of the M241 Logic Controller:

Command	Description
`Firewall Enable`	Blocks the frames from the Ethernet interfaces. If no specific IP address is authorized, it is not possible to communicate on the Ethernet interfaces. **NOTE:** By default, when the firewall is enabled, the frames are rejected.
`Firewall Disable`	Firewall rules are not applied. Frames are not blocked
`Firewall Ethx Default Allow` (1)	Frames are accepted by the controller.
`Firewall Ethx Default Reject`(1)	Frames are rejected by the controller. **NOTE:** By default, if this line is not present, it corresponds to the command `Firewall Eth1 Default Reject`.
(1) Where Ethx = • Eth1: Ethernet_1 • Eth2: TM4ES4	

Specific Firewall Commands

The following commands are available to configure firewall rules for specific ports and addresses:

Command	Range	Description
`Firewall Eth1 Allow IP •.•.•.•`	• = 0...255	Frames from the specified IP address are allowed on all port numbers and port types.
`Firewall Eth1 Reject IP •.•.•.•`	• = 0...255	Frames from the specified IP address are rejected on all port numbers and port types.
`Firewall Eth1 Allow IPs •.•.•.• to •.•.•.•`	• = 0...255	Frames from the IP addresses in the specified range are allowed for all port numbers and port types.
`Firewall Eth1 Reject IPs •.•.•.• to •.•.•.•`	• = 0...255	Frames from the IP addresses in the specified range are rejected for all port numbers and port types.
`Firewall Eth1 Allow port_type port Y`	Y = (destination port numbers *(see page 192)*)	Frames with the specified destination port number are allowed.
`Firewall Eth1 Reject port_type port Y`	Y = (destination port numbers *(see page 192)*)	Frames with the specified destination port number are rejected. **NOTE:** When IP forwarding is activated, rules with reject port only filter frames with current controller as destination. They are not applied for the frames routed by the current controller.
`Firewall Eth1 Allow port_type ports Y1 to Y2`	Y = (destination port numbers *(see page 192)*)	Frames with a destination port number in the specified range are allowed.
`Firewall Eth1 Reject port_type ports Y1 to Y2`	Y = (destination port numbers *(see page 192)*)	Frames with a destination port number in the specified range are rejected.
`Firewall Eth1 Allow IP •.•.•.• on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from the specified IP address and with the specified destination port number are allowed.
`Firewall Eth1 Reject IP •.•.•.• on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from the specified IP address and with the specified destination port number are rejected.
`Firewall Eth1 Allow IP •.•.•.• on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from the specified IP address and with a destination port number in the specified range are allowed.

Command	Range	Description
`Firewall Eth1 Reject IP •.•.•.• on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from the specified IP address and with a destination port number in the specified range are rejected.
`Firewall Eth1 Allow IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from an IP address in the specified range and with the specified destination port number are allowed.
`Firewall Eth1 Reject IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from an IP address in the specified range and with the specified destination port number are rejected.
`Firewall Eth1 Allow IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from an IP address in the specified range and with a destination port number in the specified range are allowed.
`Firewall Eth1 Reject IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers *(see page 192)*)	Frames from an IP address in the specified range and with a destination port number in the specified range are rejected.
`Firewall Eth1 Allow MAC ••:••:••:••:••:••`	• = 0...F	Frames from the specified MAC address ••:••:••:••:•• are allowed. **NOTE:** When the rules to allow the MAC address are applied, only the listed MAC addresses can communicate with the controller, even if other rules are allowed.
`Firewall Eth1 Reject MAC ••:••:••:••:••:••`	• = 0...F	Frames with the specified MAC address ••:••:••:••:•• are rejected.

NOTE: The `port_type` can be TCP or UDP.

Script Example

```
; Enable FireWall. All frames are rejected;
FireWall Enable;
; Allow frames on Eth1
FireWall Eth1 Default Allow;
; Block all Modbus Requests on all IP address
Firewall Eth1 Reject tcp port 502;
; Reject frames on Eth2
FireWall Eth2 Default Reject;
; Allow FTP active connection for IP address 85.16.0.17
FireWall Eth2 Allow IP 85.16.0.17 on tcp ports 20 to 21;
```

NOTE: IP addresses are converted to CIDR format.
For example:
`"FireWall Eth2 Allow IPs 192.168.100.66 to 192.168.100.99 on tcp port 44818;"`, is separated into 7:

- `192.168.100.66/31`
- `192.168.100.68/30`
- `192.168.100.72/29`
- `192.168.100.80/28`
- `192.168.100.96/27`
- `192.168.100.128/26`
- `192.168.100.192/29`

To prevent a firewall error, use the entire subnet configuration.

NOTE: Characters are limited to 200 per line, including comments.

Ports Used

Protocol	Destination Port Numbers
Machine Expert	UDP 1740, 1741, 1742, 1743 TCP 1105
FTP	TCP 21, 20
HTTP	TCP 80
Modbus	TCP 502 (1)
Machine Expert Discovery	UDP 27126, 27127
SNMP	UDP 161, 162
NVL	UDP Default value: 1202
EtherNet/IP	UDP 2222 TCP 44818
TFTP	UDP 69 (used for FDR server only)
(1) The default value can be changed using the change ModbusPort command *(see page 180)*.	

Chapter 14
Industrial Ethernet Manager

Introduction

This chapter describes how to add and configure the Industrial Ethernet.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Industrial Ethernet	194
DHCP Server	198
Fast Device Replacement	199

Industrial Ethernet

Overview

Industrial Ethernet is the term used to represent the industrial protocols that use the standard Ethernet physical layer and standard Ethernet protocols.

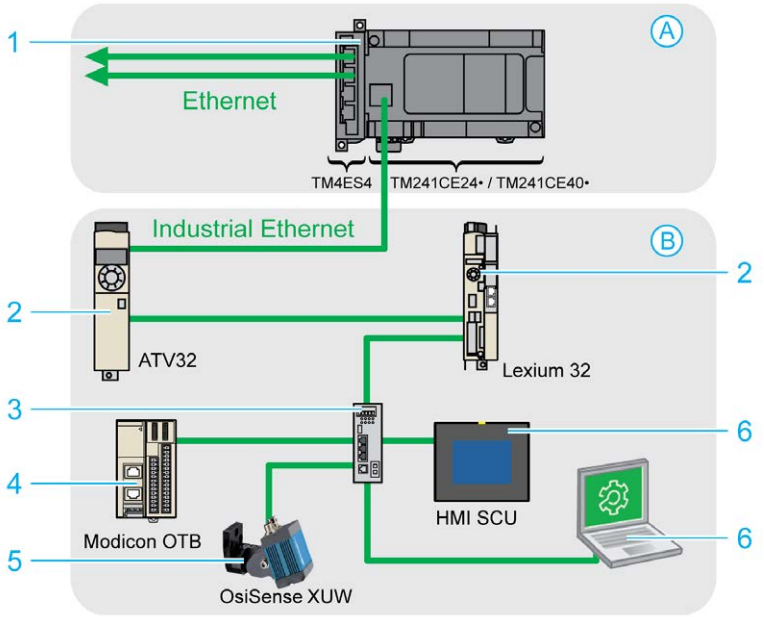
On an Industrial Ethernet network, you can connect:

- Industrial devices (industrial protocols)
- Non-industrial devices (other Ethernet protocols)

For more information, refer to Industrial Ethernet User Guide.

Industrial Ethernet Architecture

This figure presents a typical Industrial Ethernet architecture:



A Control network
B Device network
1 Logic controller *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)*
2 Daisy-chained devices
3 Ethernet switch
4 I/O island (Modbus TCP)
5 Vision sensor (EtherNet/IP)
6 PC and HMI (TCP/UDP)

2, 4, and 5 Industrial Ethernet slave devices (EtherNet/IP / Modbus TCP)

This architecture is configurable with EcoStruxure Machine Expert.

The M241 Logic Controller can be connected simultaneously to the control network and the device network. To use this functionality, you must make a second Ethernet port available by adding a TM4ES4 expansion module to your configuration. The Ethernet port embedded on the logic controller then connects to the device network and the Ethernet port on the TM4ES4 connects to the control network.

If no TM4ES4 expansion module is added, the embedded Ethernet port on the M241 Logic Controller can be connected to either the control network or the device network.

Industrial Ethernet Description

M241 Logic Controller	
Features	**Description**
Topology	Daisy chain and Star via switches
Bandwidth	10/100 Mbit/s
EtherNet/IP Scanner	
Performance	Up to 16 EtherNet/IP target devices managed by the logic controller, monitored within a timeslot of 10 ms
Number of connections	0...16
Number of input words	0...1024
Number of output words	0...1024
I/O communications	EtherNet/IP scanner service Function block for configuration and data transfer
	Originator/Target
Modbus TCP IOScanner	
Performance	Up to 64 Modbus TCP slave devices managed by the logic controller, monitored within a timeslot of 64 ms.
Number of connections	0...64
Number of input words	0...2048
Number of output words	0...2048
I/O communications	Modbus TCP IOScanner service Function block for data transfer
	Master/Slave

M241 Logic Controller	
Features	**Description**
Other services	FDT/DTM/EDS management
	FDR (Fast Device Replacement)
	DHCP server
	Security management (refer to Security Parameters *(see page 126)* and Firewall Configuration *(see page 182)*)
	Modbus TCP server
	Modbus TCP client
	Web server
	FTP Server (FTP and TFTP protocols)
	SNMP
	EtherNet/IP adapter (controller as a target on EtherNet/IP)(1)
	EtherNet/IP Originator
	Modbus TCP server (controller as a slave on Modbus TCP)(1)
	IEC VAR ACCESS
Additional features	Possible to mix up to 16 EtherNet/IP and Modbus TCP server devices. Devices can be directly accessed for configuration, monitoring, and management purposes. Network transparency between control network and device network (logic controller can be used as a gateway). **NOTE:** Using the logic controller as a gateway can impact the performance of the logic controller.
(1) You must add a TM4ES4 expansion module to your logic controller to use this service in addition to the EtherNet/IP Scanner or Modbus TCP IOScanner features.	

EtherNet/IP Overview

EtherNet/IP is the implementation of the CIP protocol over standard Ethernet.

The EtherNet/IP protocol uses an originator/target architecture for data exchange.

Originators are devices that initiate data exchanges with target devices on the network. This applies to both I/O communications and service messaging. This is the equivalent of the role of a client in a Modbus network.

Targets are devices that respond to data requests generated by originators. This applies to both I/O communications and service messaging. This is the equivalent of the role of a server in a Modbus network.

EtherNet/IP Adapter is an end-device in an EtherNet/IP network. I/O blocks and drives can be EtherNet/IP Adapter devices.

The communication between an EtherNet/IP originator and target is accomplished using an EtherNet/IP connection.

Modbus TCP Overview

The Modbus TCP protocol uses a client/server architecture for data exchange.

Modbus TCP explicit (non-cyclic) data exchanges are managed by the application.

Modbus TCP implicit (cyclic) data exchanges are managed by the Modbus TCP IOScanner. The Modbus TCP IOScanner is a service based on Ethernet that polls slave devices continuously to exchange data, status, and diagnostic information. This process monitors inputs and controls outputs of slave devices.

Clients are devices that initiate data exchange with other devices on the network. This applies to both I/O communications and service messaging.

Servers are devices that address any data requests generated by a Client. This applies to both I/O communications and service messaging.

The communication between the Modbus TCP IOScanner and the slave device is accomplished using Modbus TCP channels.

Adding the Industrial Ethernet Manager

The **Industrial_Ethernet_manager** must be present on the **Ethernet_1 (Ethernet Network)** node of the device tree to activate these functions and services:

- EtherNet/IP Scanner
- Modbus TCP IOScanner

If **Ethernet_1 (Ethernet Network)** is already in use, you must add a TM4ES4 expansion module to your controller and move the **EthernetIP** or **Modbus TCP slave device** nodes from **Ethernet_1 (Ethernet Network)** to the **TM4ES4** node.

The **Industrial_Ethernet_manager** is automatically added when a slave device is added on the **Ethernet_1 (Ethernet Network)** node.

To manually add the **Industrial_Ethernet_manager** to the **Ethernet_1 (Ethernet Network)**:

Step	Action
1	In the **Devices Tree**, select **Ethernet_1** (**Ethernet Network**) and click the green plus button of the node or right-click **Ethernet_1** (**Ethernet Network**) and execute the **Add Device** command from the contextual menu. **Result**: The **Add Device** dialog box opens.
2	In the **Add Device** dialog box, select **Protocol Managers → Industrial Ethernet manager**.
3	Click the **Add Device** button.
4	Click the **Close** button.

For more information, refer to Industrial Ethernet Manager Configuration, EtherNet/IP Target Settings and Modbus TCP Settings *(see EcoStruxure Machine Expert Modbus TCP, User Guide)*.

DHCP Server

Overview

It is possible to configure a DHCP server on the **Ethernet_1** network of the M241 Logic Controller.

The DHCP server offers addresses to the devices connected on the **Ethernet_1** network. The DHCP server only delivers static addresses. Each slave identified is assigned a unique address. DHCP slave devices are identified either by their MAC address or their DHCP device name. The DHCP server configuration table defines the relation between addresses and identified slave devices.

The DHCP server addresses are given with an infinite lease time. There is no need for the slave devices to refresh the leased IP address.

For more information, refer to IP Addressing Methods *(see EcoStruxure Machine Expert Modbus TCP, User Guide)*.

Fast Device Replacement

Overview

The Fast Device Replacement (FDR) helps facilitate replacing and reconfiguring a network device. This function is available on the **Ethernet_1** port of the M241 Logic Controller.

For more information, refer to Slave Device Replacement with FDR *(see EcoStruxure Machine Expert Modbus TCP, User Guide)*.

Chapter 15
Serial Line Configuration

Introduction

This chapter describes how to configure the serial line communication of the Modicon M241 Logic Controller.

The Modicon M241 Logic Controller has 2 Serial Line ports. These ports are configured to use the following protocols when new or after a controller firmware update:

- Serial Line 1: Machine Expert Network Manager.
- Serial Line 2: Modbus Manager.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Serial Line Configuration	202
Machine Expert Network Manager	204
Modbus Manager	205
ASCII Manager	209
Modbus Serial IOScanner	211
Adding a Device on the Modbus Serial IOScanner	213
Adding a Modem to a Manager	220

Serial Line Configuration

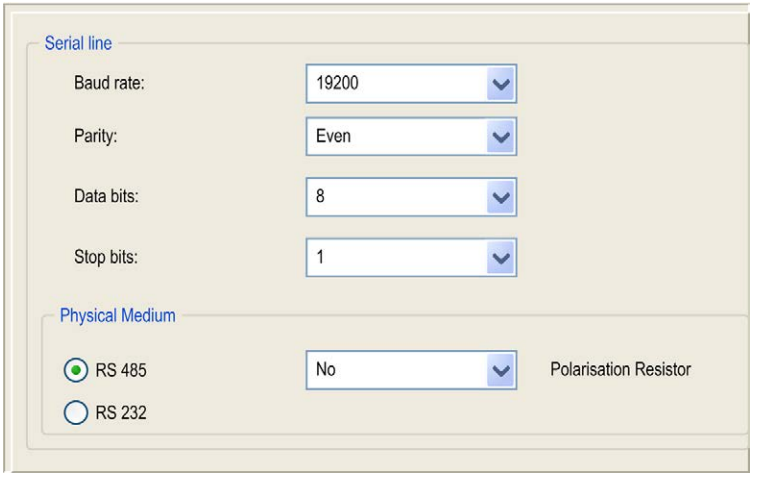
Introduction

The Serial Line configuration window allows you to configure the physical parameters of a serial line (baud rate, parity, and so on).

Serial Line Configuration

To configure a Serial Line, double-click **Serial line** in the **Devices tree**.

The **Configuration** window is displayed as below:



The following parameters must be identical for each serial device connected to the port.

Element	Description
Baud rate	Transmission speed in bits/s
Parity	Used for error detection
Data bits	Number of bits for transmitting data
Stop bits	Number of stop bits
Physical Medium	Specify the medium to use: ● RS485 (using polarisation resistor or not) ● RS232 (only available on Serial Line 1)
Polarization Resistor	Polarization resistors are integrated in the controller. They are switched on or off by this parameter.

The serial line ports of your controller are configured for the Machine Expert protocol by default when new or when you update the controller firmware. The Machine Expert protocol is incompatible with that of other protocols such as Modbus Serial Line. Connecting a new controller to, or updating the firmware of a controller connected to, an active Modbus configured serial line can cause the other devices on the serial line to stop communicating. Make sure that the controller is not connected to an active Modbus serial line network before first downloading a valid application having the concerned port or ports properly configured for the intended protocol.

NOTICE
INTERRUPTION OF SERIAL LINE COMMUNICATIONS Be sure that your application has the serial line ports properly configured for Modbus before physically connecting the controller to an operational Modbus Serial Line network. **Failure to follow these instructions can result in equipment damage.**

This table indicates the maximum baud rate value of the managers:

Manager	Maximum Baud Rate (Bits/S)
Machine Expert Network Manager	115200
Modbus Manager	
ASCII Manager	
Modbus IOScanner	

Machine Expert Network Manager

Introduction

Use the Machine Expert Network Manager to exchange variables with a XBTGT/XBTGK Advanced Panel with Machine Expert software protocol, or when the Serial Line is used for EcoStruxure Machine Expert programming.

Adding the Manager

To add a Machine Expert Network Manager to your controller, select the **Machine Expert-Network Manager** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Configuring the Manager

There is no configuration for Machine Expert Network Manager.

Adding a Modem

To add a modem to the Machine Expert Network Manager, refer to Adding a Modem to a Manager *(see page 220)*.

Modbus Manager

Introduction

The Modbus Manager is used for Modbus RTU or ASCII protocol in master or slave mode.

Adding the Manager

To add a Modbus manager to your controller, select the **Modbus Manager** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

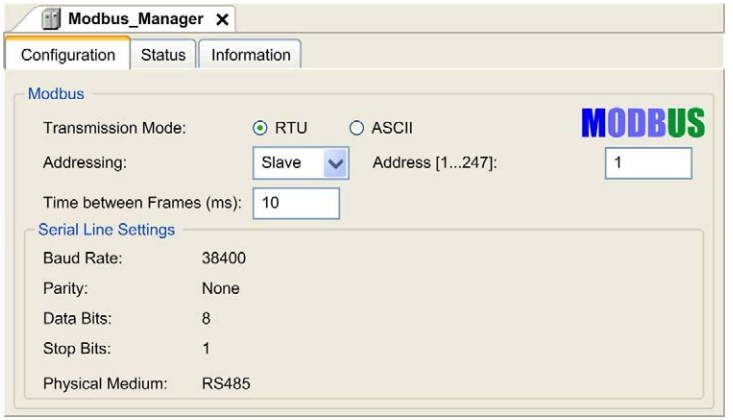
For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Modbus Manager Configuration

To configure the Modbus Manager of your controller, double-click **Modbus Manager** in the **Devices tree**.

The Modbus Manager configuration window is displayed as below:



Set the parameters as described in this table:

Element	Description
Transmission Mode	Specify the transmission mode to use: ● RTU: uses binary coding and CRC error-checking (8 data bits) ● ASCII: messages are in ASCII format, LRC error-checking (7 data bits) Set this parameter identical for each Modbus device on the link.
Addressing	Specify the device type: ● Master ● Slave
Address	Modbus address of the device, when slave is selected.
Time between Frames (ms)	Time to avoid bus-collision. Set this parameter identical for each Modbus device on the link.
Serial Line Settings	Parameters specified in the Serial Line configuration window.

Modbus Master

When the controller is configured as a Modbus Master, the following function blocks are supported from the PLCCommunication Library:

- ADDM
- READ_VAR
- SEND_RECV_MSG
- SINGLE_WRITE
- WRITE_READ_VAR
- WRITE_VAR

For further information, see Function Block Descriptions *(see EcoStruxure Machine Expert, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)* of the PLCCommunication Library.

Modbus Slave

When the controller is configured as Modbus Slave, the following Modbus requests are supported:

Function Code Dec (Hex)	Sub-Function Dec (Hex)	Function
1 (1 hex)	–	Read digital outputs (%Q)
2 (2 hex)	–	Read digital inputs (%I)
3 (3 hex)	–	Read multiple register (%MW)
6 (6 hex)	–	Write single register (%MW)
8 (8 hex)	–	Diagnostic
15 (F hex)	–	Write multiple digital outputs (%Q)
16 (10 hex)	–	Write multiple registers (%MW)
23 (17 hex)	–	Read/write multiple registers (%MW)
43 (2B hex)	14 (E hex)	Read device identification

This table contains the sub-function codes supported by the diagnostic Modbus request 08:

Sub-Function Code		Function
Dec	Hex	
10	0A	Clears Counters and Diagnostic Register
11	0B	Returns Bus Message Count
12	0C	Returns Bus Communication Error Count
13	0D	Returns Bus Exception Error Count
14	0E	Returns Slave Message Count
15	0F	Returns Slave No Response Count
16	10	Returns Slave NAK Count
17	11	Returns Slave Busy Count
18	12	Returns Bus Character Overrun Count

This table lists the objects that can be read with a read device identification request (basic identification level):

Object ID	Object Name	Type	Value
00 hex	Vendor code	ASCII String	Schneider Electric
01 hex	Product code	ASCII String	Controller reference eg: TM241CE24T
02 hex	Major / Minor revision	ASCII String	aa.bb.cc.dd (same as device descriptor)

The following section describes the differences between the Modbus memory mapping of the controller and HMI Modbus mapping. If you do not program your application to recognize these differences in mapping, your controller and HMI will not communicate correctly. Thus it will be possible for incorrect values to be written to memory areas responsible for output operations.

⚠ WARNING

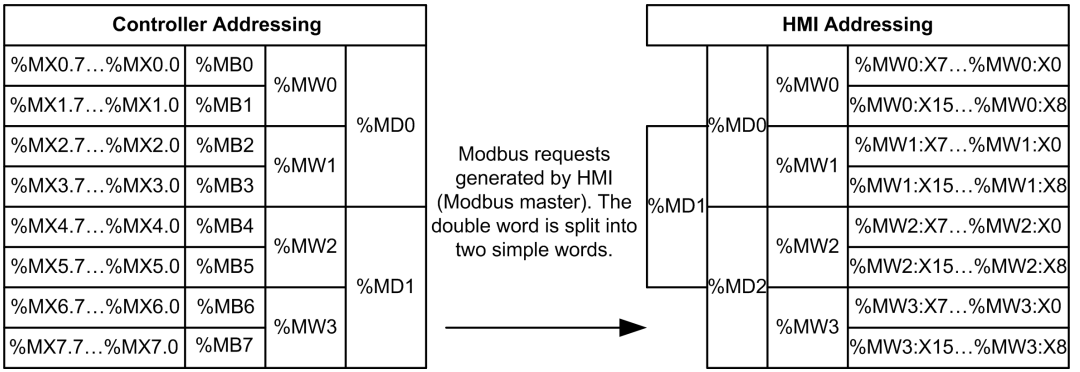
UNINTENDED EQUIPMENT OPERATION

Program your application to translate between the Modbus memory mapping used by the controller and that used by any attached HMI devices.

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

When the controller and the Magelis HMI are connected via Modbus (HMI is master of Modbus requests), the data exchange uses simple word requests.

There is an overlap on simple words of the HMI memory while using double words but not for the controller memory (see following diagram). In order to have a match between the HMI memory area and the controller memory area, the ratio between double words of HMI memory and the double words of controller memory has to be 2.



The following gives examples of memory match for the double words:

- %MD2 memory area of the HMI corresponds to %MD1 memory area of the controller because the same simple words are used by the Modbus request.
- %MD20 memory area of the HMI corresponds to %MD10 memory area of the controller because the same simple words are used by the Modbus request.

The following gives examples of memory match for the bits:

- %MW0:X9 memory area of the HMI corresponds to %MX1.1 memory area of the controller because the simple words are split in 2 distinct bytes in the controller memory.

Adding a Modem

To add a Modem to the Modbus Manager, refer to Adding a Modem to a Manager *(see page 220)*.

ASCII Manager

Introduction

The ASCII manager is used on a Serial Line, to transmit and/or receive data with a simple device.

Adding the Manager

To add an ASCII manager to your controller, select the **ASCII Manager** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

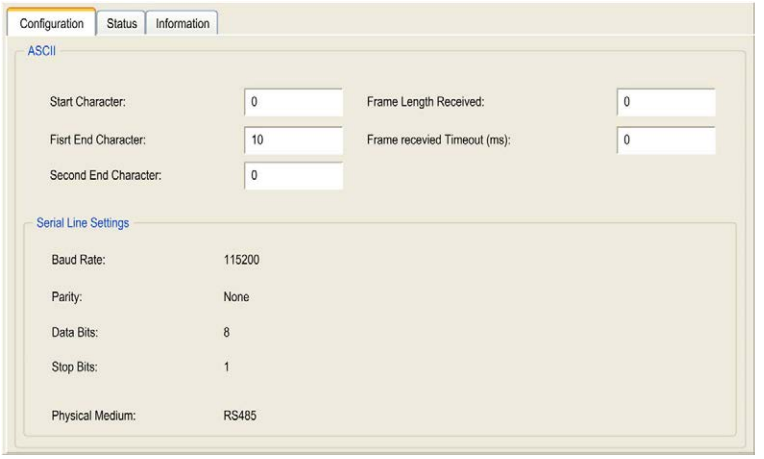
For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

ASCII Manager Configuration

To configure the ASCII manager of your controller, double-click **ASCII Manager** in the **Devices tree**.

The ASCII Manager configuration window is displayed as below:



Set the parameters as described in this table:

Parameter	Description
Start Character	If 0, no start character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the beginning of a frame. In **Sending Mode**, this character is added at the beginning of the frame.
First End Character	If 0, no first end character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Second End Character	If 0, no second end character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Frame Length Received	If 0, this parameter is not used. This parameter allows the system to conclude an end of frame at reception when the controller received the specified number of characters. **Note:** This parameter cannot be used simultaneously with **Frame Received Timeout (ms)**.
Frame Received Timeout (ms)	If 0, this parameter is not used. This parameter allows the system to conclude the end of frame at reception after a silence of the specified number of ms.
Serial Line Settings	Parameters specified in the Serial Line configuration window *(see page 202)*.

NOTE: In the case of using several frame termination conditions, the first condition to be TRUE terminates the exchange.

Adding a Modem

To add a Modem to the ASCII manager, refer to Adding a Modem to a Manager *(see page 220)*.

Modbus Serial IOScanner

Introduction

The Modbus IOScanner is used to simplify exchanges with Modbus slave devices.

Add a Modbus IOScanner

To add a Modbus IOScanner on a Serial Line, select the **Modbus_IOScanner** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

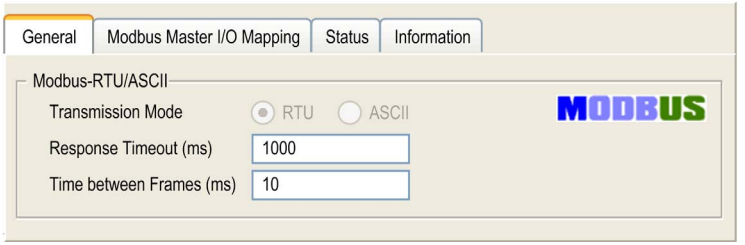
• Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Modbus IOScanner Configuration

To configure a Modbus IOScanner on a Serial Line, double-click **Modbus IOScanner** in the **Devices tree**.

The configuration window is displayed as below:



Set the parameters as described in this table:

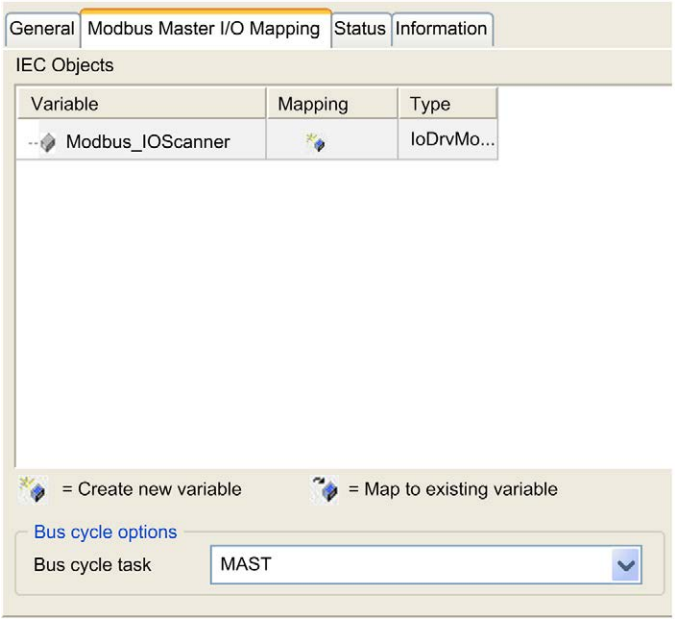
Element	Description
Transmission Mode	Specifies the transmission mode to use: ● RTU: uses binary coding and CRC error-checking (8 data bits) ● ASCII: messages are in ASCII format, LRC error-checking (7 data bits) Set this parameter identical for each Modbus device on the network.
Response Timeout (ms)	Timeout used in the exchanges.
Time between Frames (ms)	Delay to reduce data collision on the bus. Set this parameter identical for each Modbus device on the network.

NOTE: Do not use function blocks of the PLCCommunication library on a serial line with a Modbus IOScanner configured. This disrupts the Modbus IOScanner exchange.

Bus Cycle Task Selection

The Modbus IOScanner and the devices exchange data at each cycle of the chosen application task.

To select this task, select the **Modbus Master IO Mapping** tab. The configuration window is displayed as below:



The **Bus cycle task** parameter allows you to select the application task that manages the scanner:

- **Use parent bus cycle setting**: associate the scanner with the application task that manages the controller.
- **MAST**: associate the scanner with the MAST task.
- Another existing task: you can select an existing task and associate it to the scanner. For more information about the application tasks, refer to the EcoStruxure Machine Expert Programming Guide *(see EcoStruxure Machine Expert, Programming Guide)*.

The scan time of the task associated with the scanner must be less than 500 ms.

Adding a Device on the Modbus Serial IOScanner

Introduction

This section describes how to add a device on the Modbus IOScanner.

Adding a Device on the Modbus IOScanner

To add a device on the Modbus IOScanner, select the **Generic Modbus Slave** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the **Modbus_IOScanner** node of the **Devices tree**.

For more information on adding a device to your project, refer to:

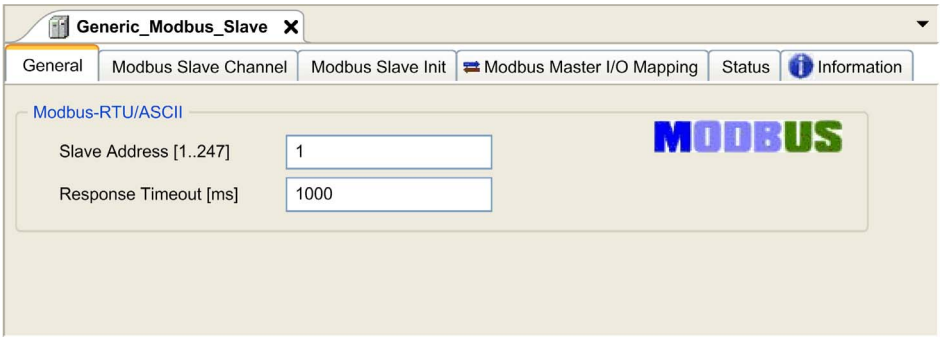
• Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

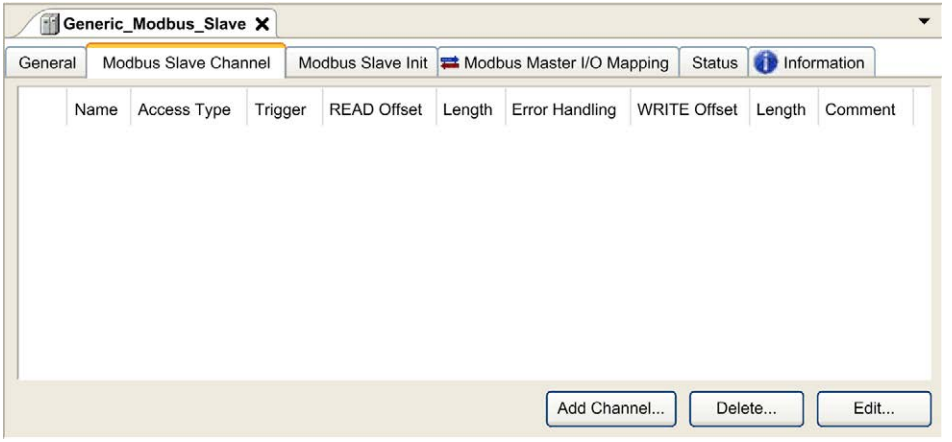
NOTE: The variable for the exchange is automatically created in the `%IWx` and `%QWx` of the **Modbus Serial Master I/O Mapping** tab.

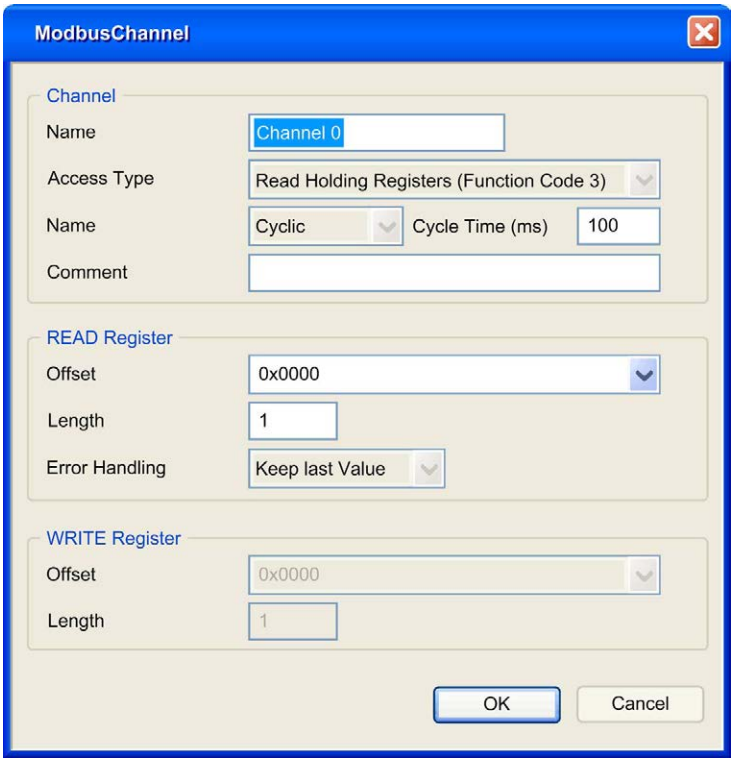
Configuring a Device Added on the Modbus IOScanner

To configure the device added on the Modbus IOScanner, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **Generic Modbus Slave**. **Result:** The configuration window is displayed. 
2	Enter a **Slave Address** value for your device (choose a value from 1 to 247).
3	Choose a value for the **Response Timeout** (in ms).

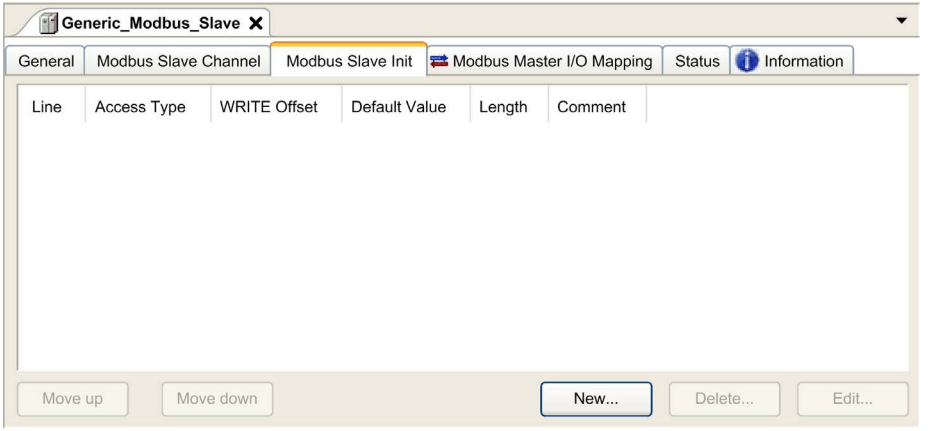
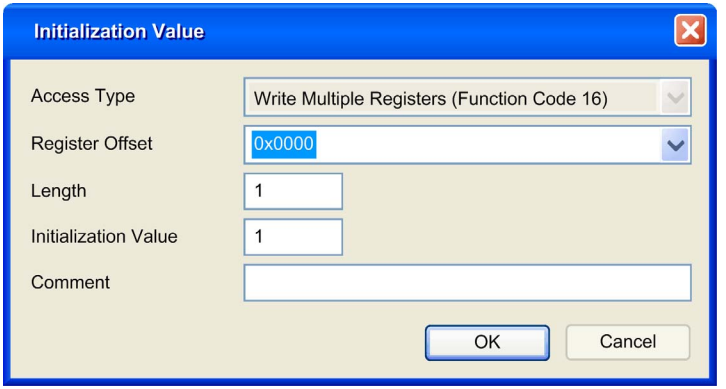
To configure the **Modbus Channels**, proceed as follows:

Step	Action
1	Click the **Modbus Slave Channel** tab: 

Step	Action
2	Click the **Add Channel** button: 

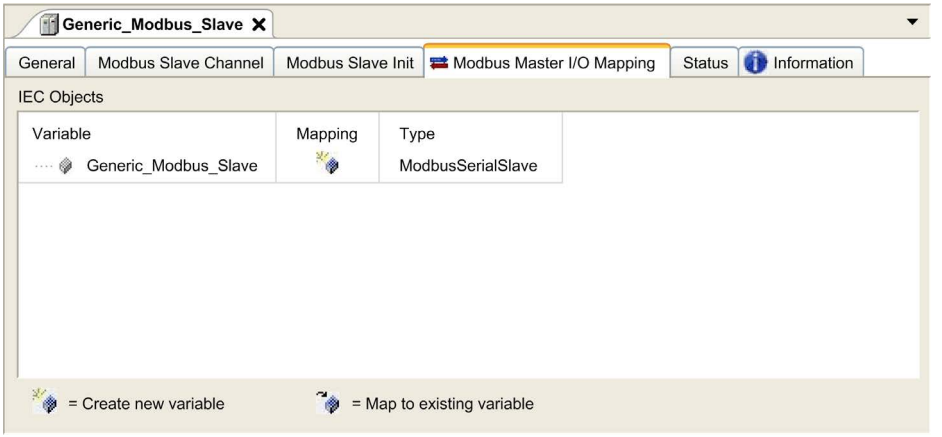
Step	Action
3	Configure an exchange: In the field **Channel**, you can add the following values: • **Channel**: Enter a name for your channel. • **Access Type**: Choose the exchange type: Read or Write or Read/Write multiple requests *(see page 219)*. • **Trigger**: Choose the trigger of the exchange. It can be **CYCLIC** with the period defined in **Cycle Time (ms)** field, started by a **RISING EDGE** on a boolean variable (this boolean variable is then created in the **Modbus Master I/O Mapping** tab), or by the **Application**. • **Comment**: Add a comment about this channel. In the field **READ Register** (if your channel is Read or Read/Write one), you can configure the %MW to be read on the Modbus slave. Those are mapped on %IW (see Modbus Master I/O Mapping tab): • **Offset**: Offset of the %MW to read. 0 means that the first object that is read is %MW0. • **Length**: Number of %MW to be read. For example, if 'Offset' = 2 and 'Length' = 3, the channel reads %MW2, %MW3 and %MW4. • **Error Handling**: choose the behavior of the related %IW in case of loss of communication. In the field **WRITE Register** (if your channel is Write or Read/Write one), you can configure the %MW to be written to the Modbus slave. Those are mapped on %QW (see Modbus Master I/O Mapping tab): • **Offset**: Offset of the %MW to write. 0 means that the first object that is written is %MW0. • **Length**: Number of %MW to be written. For example, if 'Offset' = 2 and 'Length' = 3, the channel writes %MW2, %MW3 and %MW4.
4	Click **OK** to validate the configuration of this channel. **NOTE:** You can also: • Click the **Delete** button to remove a channel. • Click the **Edit** button to change the parameters of a channel.

To configure your **Modbus Initialization Value**, proceed as follows:

Step	Action
1	Click the **Modbus Slave Init** tab: 
2	Click **New** to create a new initialization value:  The **Initialization Value** window contains the following parameters: • **Access Type**: Enter the exchange type: Write requests *(see page 219)*. • **Register Offset**: Register number of register to be initialized. • **Length**: Number of %MW to be read. For example, if 'Offset' = 2 and 'Length' = 3, the channel reads %MW2, %MW3 and %MW4. • **Initialization Value**: Value the registers are initialized with. • **Comment**: Add a comment about this channel.

Step	Action
3	Click **OK** to create a new **Initialization Value**. **NOTE:** You can also: ● Click **Move up** or **Move down** to change the position of a value in the list. ● Click **Delete** to remove a value in the list. ● Click **Edit** to change the parameters of a value.

To configure your **Modbus Master I/O Mapping**, proceed as follows:

Step	Action
1	Click the **Modbus Master I/O Mapping** tab: 
2	Double-click in a cell of the **Variable** column to open a text field. Enter the name of a variable or click the browse button **[...]** and chose a variable with the **Input Assistant**.
3	For more information on I/O mapping, refer to EcoStruxure Machine Expert Programming Guide.

Access Types

This table describes the different access types available:

Function	Function Code	Availability
`Read Coils`	1	**ModbusChannel**
`Read Discrete Inputs`	2	**ModbusChannel**
`Read Holding Registers` (default setting for the channel configuration)	3	**ModbusChannel**
`Read Input Registers`	4	**ModbusChannel**
`Write Single Coil`	5	**ModbusChannel** **Initialization Value**
`Write Single Register`	6	**ModbusChannel** **Initialization Value**
`Write Multiple Coils`	15	**ModbusChannel** **Initialization Value**
`Write Multiple Registers` (default setting for the slave initialization)	16	**ModbusChannel** **Initialization Value**
`Read/Write Multiple Registers`	23	**ModbusChannel**

Adding a Modem to a Manager

Introduction

A modem can be added to the following managers:

- ASCII Manager
- Modbus Manager
- Machine Expert Network Manager

NOTE: Use a modem which implements Hayes commands if you need a modem connection with Machine Expert Network Manager.

Adding a Modem to a Manager

To add a modem to your controller, select the modem you want in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the manager node.

For more information on adding a device to your project, refer to:

• Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

For further information, refer to Modem Library Guide *(see EcoStruxure Machine Expert, Modem Functions, Modem Library Guide)*.

Chapter 16
CANopen Configuration

CANopen Interface Configuration

CAN Bus Configuration

To configure the **CAN** bus of your controller, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **CAN_1**.
2	Configure the baudrate (by default: 250000 bits/s):  **NOTE:** The **Online Bus Access** option allows you to block SDO, DTM, and NMT sending through the status screen.

When connecting a DTM to a device using the network, the DTM communicates in parallel with the running application. The overall performance of the system is impacted and may overload the network, and therefore have consequences for the coherency of data across devices under control.

⚠ WARNING

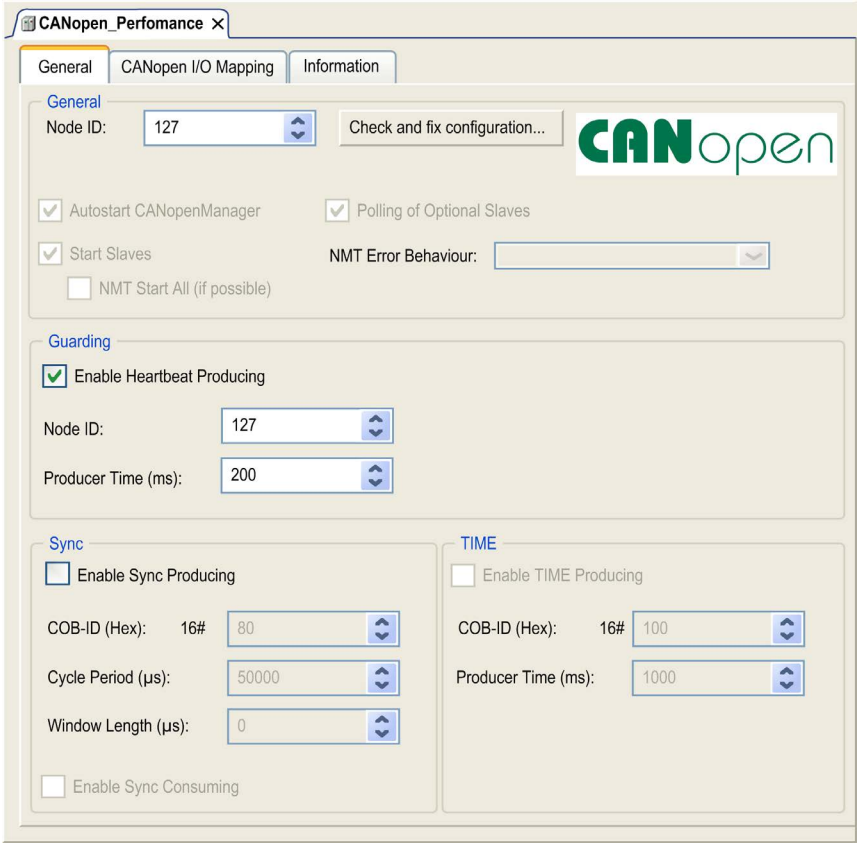
UNINTENDED EQUIPMENT OPERATION

Place your machine or process in a state such that DTM communications will not impact its performance.

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

CANopen Manager Creation and Configuration

If the **CANopen Manager** is not already present below the **CAN** node, proceed as follows to create and configure it:

Step	Action
1	Click the **Plus Button**  next to the **CAN_1** node in the **Devices Tree**. In the **Add Device** window, select **CANopen Performance** and click the **Add Device** button. For more information on adding a device to your project, refer to: ● Using the Drag-and-Drop Method *(see EcoStruxure Machine Expert, Programming Guide)* ● Using the Contextual Menu or Plus button *(see EcoStruxure Machine Expert, Programming Guide)*
2	Double-click **CANopen_Performance**. **Result:** The **CANopen Manager** configuration window appears: 

NOTE: If **Enable Sync Producing** is checked, the **CAN_x_Sync** task is added to the **Application → Task Configuration** node in the **Applications tree** tab.
Do not delete or change the **Type** or **External event** attributes of **CAN_x_Sync** tasks. If you do so, EcoStruxure Machine Expert will detect an error when you attempt to build the application, and you will not be able to download it to the controller.
If you uncheck the **Enable Sync Producing** option on the **CANopen Manager** subtab of the **CANopen_Performance** tab, the **CAN0_Sync** task is automatically deleted from your program.

Adding a CANopen Device

Refer to the EcoStruxure Machine Expert Programming Guide for more information on Adding Communication Managers and Adding Slave Devices to a Communication Manager.

CANopen Operating Limits

The Modicon M241 Logic Controller CANopen master has the following operating limits:

Maximum number of slave devices	63
Maximum number of Received PDO (RPDO)	252
Maximum number of Transmitted PDO (TPDO)	252

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not connect more than 63 CANopen slave devices to the controller.
- Program your application to use 252 or fewer Transmit PDO (TPDO).
- Program your application to use 252 or fewer Receive PDO (RPDO).

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

CAN Bus Format

The CAN bus format is CAN2.0A for CANopen.

Chapter 17
J1939 Configuration

J1939 Interface Configuration

CAN Bus Configuration

To configure the **CAN** bus of your controller, refer to CAN Bus Configuration *(see page 221)*.

The CAN bus format is CAN2.0B for J1939.

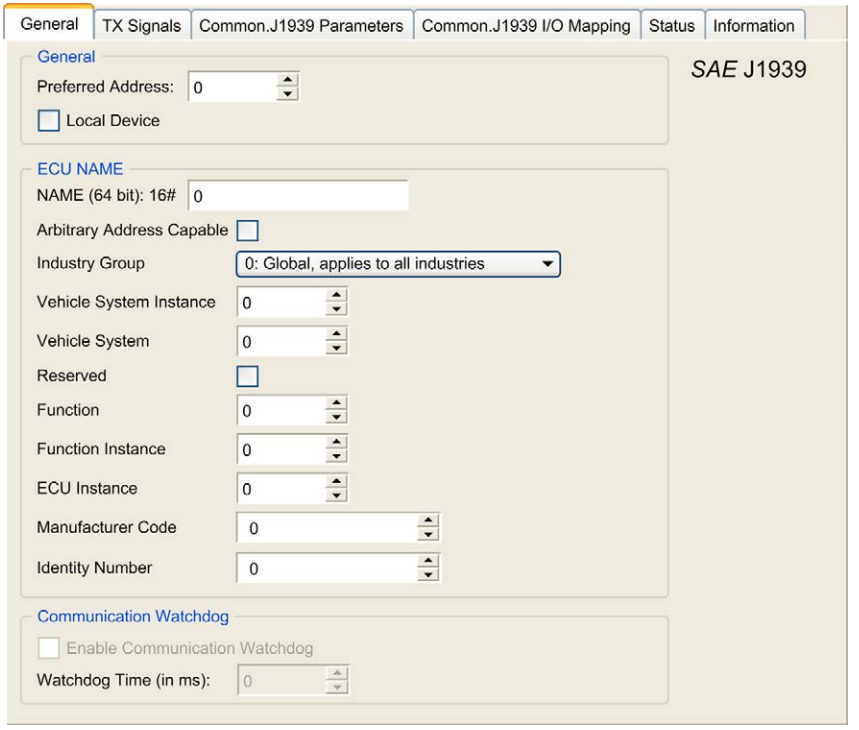
J1939 Manager Creation and Configuration

Proceed as follows to create and configure a J1939 Manager, if not already present, below the **CAN_1** node:

Step	Action
1	Click the **Plus** button  next to the **CAN_1** node in the **Devices** tree.
2	In the **Add Device** window, select **J1939_Manager** and click the **Add Device** button. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)* • Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*
3	Close the **Add Device** window.
4	Double-click **J1939_Manager (J1939_Manager)**. **Result:** The **J1939_Manager** configuration window appears: 
5	To configure the **J1939_Manager**, refer to *Programming with EcoStruxure Machine Expert / Device Editors / J1939 Configuration Editor / J1939 Manager Editor / Manager Editor* found in the EcoStruxure Machine Expert online help.

ECU Creation and Configuration

Proceed as follows to create and configure Electronic Control Units (ECUs):

Step	Action
1	Click the Plus button next to the **J1939_Manager (J1939_Manager)** node in the **Devices** tree.
2	In the **Add Device** window, select **J1939_ECU** and click the **Add Device** button. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)* • Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*
3	Close the **Add Device** window.
4	Double-click **J1939_ECU (J1939_ECU)**. **Result:** The **J1939_ECU** configuration window appears: 
5	To configure the **J1939_ECU**, refer to Configuring J1939 ECUs *(see page 227)*.

Configuring J1939 ECUs

As an overview, the following tasks must be generally accomplished:

- Add one **J1939_ECU** node for each physical J1939 device connected on the CAN bus.
- For each J1939 device, specify a unique **Preferred Address** in the range 1...253.
- For each J1939 device, configure the signals (SPNs) in the **TX Signals** tab. These signals are broadcast by the J1939 device to the other J1939 devices.
  Refer to the device documentation for information on the supported SPNs.
- Associate the SPN signals with variables in the **J1939 I/O Mapping** tab so that they can be processed by the application.
- When signals have been added, verify their settings in the **Conversion** window of the **TX signals** tab, for example, **Scaling**, **Offset**, and **Unit**. The J1939 protocol does not directly support REAL values, which are instead encoded in the protocol and so must be converted in the application. Similarly, in J1939 units are defined according to the International System of Units (SI) and therefore may need to be converted to values of other unit systems.
  Examples:
  - The **Engine Speed** signal of parameter group **EEC1** has a property `Scaling=0.125` that is encoded into a raw variable of type `ARRAY[0..1] OF BYTE`. Use the following ST code to convert this to a REAL variable:
    ```
    rRPM:=(Engine_Speed[1]*256 + Engine_Speed[0])*0.125;
    ```
  - The **Total Vehicle Distance** signal has properties `Scaling=0.125` and `Unit=km`, which are received in a (raw) variable of type `ARRAY[0..3] OF BYTE`. Use the following ST code to convert this to a REAL variable in mile units:
    ```
    rTVD := (Total_Vehicle_Distance[3]*EXPT(256,3) +
    Total_Vehicle_Distance[2]*EXPT(256,2) + Total_Vehicle_Distance[1]*2
    56 +
    Total_Vehicle_Distance[0])*0.125*0.621371;
    ```
  - The **Engine Coolant Temperature** signal of parameter group **ET1** has properties `Offset=-40` and `Unit=C(Celsius)`, which are received in a (raw) variable of type `BYTE`. Use the following ST code to convert it to a REAL variable in Fahrenheit units:
    ```
    rEngineCoolantTemperature := (Engine_Coolant_Temperature -
     40)*1.8 + 32;
    ```

For more details on how to configure the **J1939_ECU**, refer to *Programming with EcoStruxure Machine Expert / Device Editors / J1939 Configuration Editor / J1939 ECU Editor / ECU Editor* found in the EcoStruxure Machine Expert online help.

Configuring the M241 Logic Controller as an ECU Device

The controller can also be configured as a J1939 ECU device:

Step	Action
1	Add a **J1393_ECU** node to the **J1939_Manager**. Refer to ECU Creation and Configuration *(see page 226)*.
2	Select the **Local Device** option in the **General** tab.
3	Configure signals sent from the controller to other J1939 devices in the **TX Signals** tab. Parameter groups are either of type **Broadcast**, that is, sent to all devices, or **P2P** (Peer-to-Peer), that is, sent to one specified device.
4	For **P2P** signals, configure the **Destination Address** of the receiving J1939 ECU device in the parameter group properties window.
5	Add **P2P** signals sent by another J1939 device to the controller in the **RX Signals (P2P)** tab of the J1939 (local) device representing the controller.
6	Configure the **Source Address** of the parameter group by specifying the address of the sending J1939 device.

Chapter 18
OPC UA Server Configuration

Introduction

This chapter describes how to configure the OPC UA server of the M241 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
OPC UA Server Overview	230
OPC UA Server Configuration	231
OPC UA Server Symbols Configuration	234
OPC UA Server Performance	236

OPC UA Server Overview

Overview

The OPC Unified Architecture server (OPC UA server) allows the M241 Logic Controller to exchange data with OPC UA clients. Server and client communicate through sessions.

The monitored items of data (also referred to as symbols) to be shared by the OPC UA server are manually selected from a list of the IEC variables used in the application.

OPC UA uses a subscription model; clients subscribe to symbols. The OPC UA server reads the values of symbols from devices at a fixed sampling rate, places the data in a queue, then sends them to clients as notifications at a regular publishing interval. The sampling interval can be shorter than the publishing interval, in which case notifications may be queued until the publishing interval elapses.

Symbols that have not changed value since the previous sample are not re-published. Instead, the OPC UA server sends regular KeepAlive messages to indicate to the client that the connection is still active.

User and Group Access Rights

Access to the OPC UA server is controlled by user rights. Refer to Users and Groups in the EcoStruxure Machine Expert Programming Guide.

OPC UA Services

The following table describes the supported OPC UA services:

OPC UA Service	Description
Address Space Model	Yes
Session services	Yes
Attribute services	Yes
Monitored item services	Yes
Queued items	Yes
Subscription services	Yes
Publishing method	Yes

OPC UA Server Configuration

Introduction

The OPC UA Server Configuration window allows you to configure the OPC UA server.

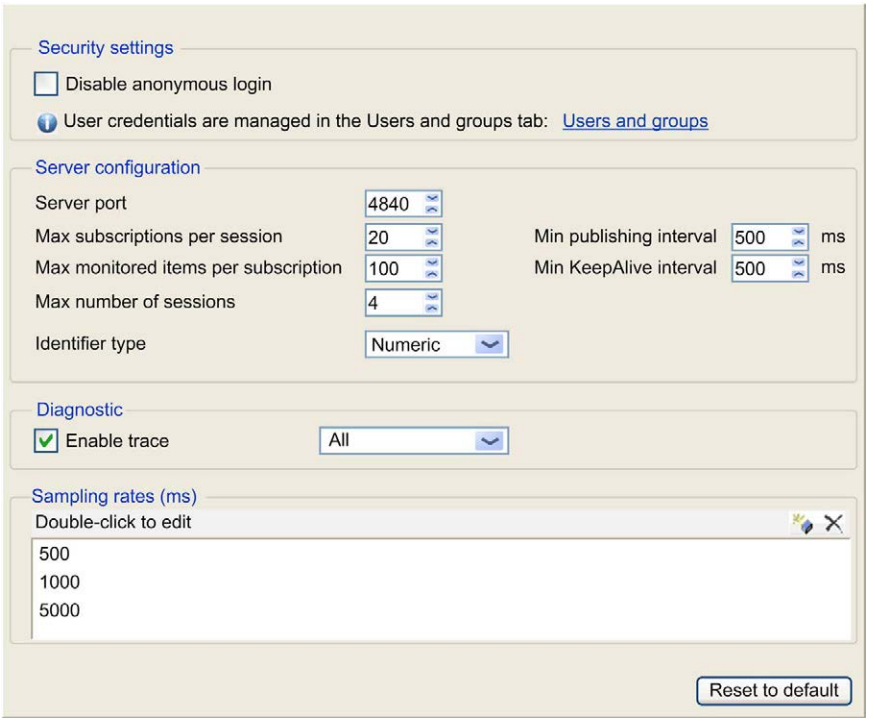
Accessing the OPC UA Server Configuration Tab

To configure the OPC UA Server:

Step	Action
1	In the **Devices tree**, double-click **MyController**.
2	Select the **OPC UA Server Configuration** tab.

OPC UA Server Configuration Tab

The following figure shows the OPC UA Server Configuration window:



OPC UA Server Configuration Description

This table describes the OPC UA Server Configuration parameters:

Parameter	Value	Default value	Description
Security Settings			
Disable anonymous login	Enabled/ Disabled	Disabled	By default, this checkbox is cleared, meaning that OPC UA clients can connect to the server anonymously. Select this checkbox to require that clients provide a valid user name and password to connect to the OPC UA server.
Server Configuration			
Server port	0...65535	4840	The port number of the OPC UA server. OPC UA clients must append this port number to the TCP URL of the controller to connect to the OPC UA server.
Max. subscriptions per session	1...100	20	Specify the maximum number of subscriptions allowed within each session.
Min. publishing interval	200...5000	1000	The publishing interval defines how frequently the OPC UA server sends notification packages to clients. Specify the minimum time that must elapse between notifications, in ms.
Max. monitored items per subscription	1...1000	100	The maximum number of *monitored items* in each subscription that the server assembles into a notification package.
Min. KeepAlive interval	500...5000	500	The OPC UA server only sends notifications when the values of monitored items of data are modified. A *KeepAlive* notification is an empty notification sent by the server to inform the client that although no data has been modified, the subscription is still active. Specify the minimum interval between KeepAlive notifications, in ms.
Max. number of sessions	1...4	2	The maximum number of clients that can connect simultaneously to the OPC UA server.
Identifier type	**Numeric** **String**	**Numeric**	Certain OPC UA clients require a specific format of unique symbol identifier (node ID). Select the format of the identifiers: ● Numeric values ● Text strings

Parameter	Value	Default value	Description
Diagnostic			
Enable trace	Enabled/disabled	Enabled	Select this checkbox to include OPC UA diagnostic messages in the controller log file *(see EcoStruxure Machine Expert, Programming Guide)*. Traces are available from the **Log** tab or from the System Log File of the Web Server. You can select the category of events to write to the log file: • **None** • **Error** • **Warning** • **System** • **Information** • **Debug** • **Content** • **All** (default)
Sampling rates (ms)	**200...5000**	**500** **1000** **2000**	The sampling rate indicates a time interval, in milliseconds (ms). When this interval has elapsed, the server sends the notification package to the client. The sampling rate can be shorter than the publishing interval, in which case notifications are queued until the publishing interval has elapsed. Sampling rates must be in the range 200...5000 (ms). Up to 3 different sampling rates can be configured. Double-click on a sampling rate to edit its value. To add a sampling rate to the list, right-click and choose **Add a new rate**. To remove a sample rate from the list, select the value and click ✕

Click **Reset to default** to return the configuration parameters on this window to their default values.

OPC UA Server Symbols Configuration

Introduction

Symbols are the items of data shared with OPC UA clients. Symbols are selected from a list of all the IEC variables used in the application. The selected symbols are then sent to the logic controller as part of the application download.

Each symbol is assigned a unique identifier. As certain client types may require a specific format, identifiers can be configured to be in either string or numeric format.

The OPC UA server supports the following IEC variable types:

- Boolean
- Byte
- Int16, Int32, Int64
- UInt16, UInt32, UInt64
- Float
- Double
- String (255 bytes)
- Sbyte

Bit memory variables (%MX) cannot be selected.

Displaying the List of Variables

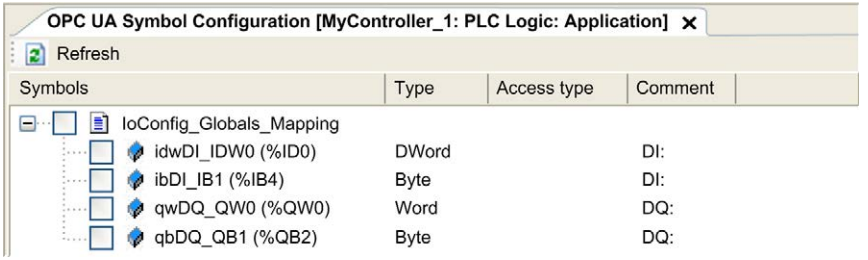
To display the list of variables:

Step	Action
1	On the **Applications tree** tab, right-click **Application** and choose **Add object → OPC UA Symbol Configuration**. **Result:** The OPC UA Symbols window is displayed. The logic controller starts the OPC UA server.
2	Click **Add**.

NOTE: The IEC objects `%MX`, `%IX`, `%QX` are not directly accessible. To access IEC objects you must first group their contents in located registers (refer to Relocation Table *(see page 37)*).

Selecting OPC UA Server Symbols

The **OPC UA Symbols** window displays the variables available for selection as symbols:



Select **IoConfig_Globals_Mapping** to select all the available variables. Otherwise, select individual symbols to share with OPC UA clients. A maximum of 1000 symbols can be selected.

Each symbol has the following properties:

Name	Description
Symbols	The variable name followed by the address of the variable.
Type	The data type of the variable.
Access type	Click repeatedly to specify the access rights of the symbol: read-only ( ) (default), write-only ( ), or read/write ( ). **NOTE:** Click in the **Access type** column of **IoConfig_Globals_Mapping** to set the access rights of all the symbols at once.
Comment	An optional comment.

Click **Refresh** to update the list of available variables.

OPC UA Server Performance

Overview

The following provides capacity and performance information for the OPC UA server of the M241 Logic Controller. Design considerations are also provided to help optimize the performance of the OPC UA server.

System Configurations Used to Evaluate Performance

OPC UA server performance is determined by the system configuration, the number of symbols being published, and the percentage of symbols being refreshed.

The following table presents the number of elements in small, medium, and large sample configurations used for evaluating OPC UA server performance:

Elements	Small	Medium	Large
EtherNet/IP adapters	0	7	0
Expansion modules	0	5	7
CANopen slave devices	0	1	63
PTO functions	0	4	4
HSC functions	0	8	8
Profibus connections	0	0	1
Modbus TCP slave devices	0	6	64

This table presents average read/write request times for each of the sample configurations and for different numbers of symbols:

Average Read/Write Request Times						
	Number of Symbols					
Configuration	50	100	250	400	500	1000
Small	42 ms	70 ms	151 ms	232 ms	284 ms	554 ms
Medium	73 ms	121 ms	265 ms	412 ms	514 ms	1024 ms
Large	520 ms	895 ms	2045 ms	3257 ms	4071 ms	7153 ms

The following tables present the average time required to refresh a monitored set of symbols using a sampling rate of 200 ms and a publishing interval of 200 ms.

This table presents the average time required to refresh 100% of symbols for each of the sample configurations:

Average Time to Refresh 100% of Symbols			
	Number of Symbols		
Configuration	100	400	1000
Small	214 ms	227 ms	254 ms
Medium	224 ms	250 ms	292 ms
Large	234 ms	330 ms	800 ms

This table presents the average time required to refresh 50% of symbols for each of the sample configurations:

Average Time to Refresh 50% of Symbols			
	Number of Symbols		
Configuration	100	400	1000
Small	211 ms	220 ms	234 ms
Medium	219 ms	234 ms	254 ms
Large	284 ms	300 ms	660 ms

This table presents the average time required to refresh 1% of symbols for each of the sample configurations:

Average Time to Refresh 1% of Symbols			
	Number of Symbols		
Configuration	100	400	1000
Small	210 ms	210 ms	212 ms
Medium	215 ms	217 ms	220 ms
Large	270 ms	277 ms	495 ms

Optimizing OPC UA Server Performance

The OPC UA server functionality is dependent on external communication networks, external device performance, and other external parameters. Data transmitted may be delayed or other possible communication errors may arise that impose practical limits on machine control. Do not use the OPC UA server functionality for safety-related data or other time-dependent purposes.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not allow safety-related data in OPC UA server data exchanges.
- Do not use OPC UA server data exchanges for any critical or time-dependent purposes.
- Do not use OPC UA server data exchanges to change equipment states without having done a risk analysis and implementing appropriate safety-related measures.

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

The above tables can be useful in determining whether OPC UA server performance is within acceptable limits. Be aware, however, that other external factors influence overall system performance, such as the volume of Ethernet traffic, or the use of jitter *(see page 92)*.

To optimize OPC UA server performance, consider the following:

- Minimize Ethernet traffic by setting the **Min. publishing interval** to the lowest value that yields an acceptable response time.
- The task cycle time *(see page 45)* configured for the M241 Logic Controller must be less than the configured **Min. publishing interval** value.
- Configuring a **Max. number of sessions** (the number of OPC UA clients that can simultaneously connect to the OPC UA server) value of greater than 1 decreases the performance of all sessions.
- The sampling rate determines the frequency at which data is exchanged. Tune the **Sampling rates (ms)** value to product the lowest response time that does not adversely affect the overall performance of the logic controller.

Chapter 19
Post Configuration

Introduction

This chapter describes how to generate and configure the post configuration file of the Modicon M241 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Post Configuration Presentation	240
Post Configuration File Management	242
Post Configuration Example	244

Post Configuration Presentation

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 parameters are set in the application.The parameters defined in the Post Configuration file are used instead of the corresponding parameters defined in the application. Not all parameters have to be specified in the Post Configuration file (for example: one parameter can change the IP address without changing the Gateway Address).

Parameters

The Post Configuration file allows you to change network parameters.

Ethernet parameters:
- IP Address
- Subnet Mask
- Gateway Address
- Transfer Rate
- IP Config Mode
- Device Name
- IP Master Address *(see Modicon TM4, Expansion Modules, Programming Guide)*

Serial Line parameters, for each serial line in the application (embedded port or PCI module):
- Baud rate
- Parity
- Data bits
- Stop bit

Profibus parameters, for each Profibus in the application (TM4 module):
- Station address
- Baud rate

NOTE: Parameter updates with a Post Configuration file that impacts parameters used by other devices via a communication port are not updated in the other devices.

For example, if the IP address used by an HMI is updated in the configuration with a Post Configuration file, the HMI uses the previous address. You must update the address used by the HMI independently.

Operating Mode

The Post Configuration file is read after:

- A Reset Warm command *(see page 68)*
- A Reset Cold command *(see page 68)*
- A reboot *(see page 70)*
- An application download *(see page 72)*

Refer to Controller States and Behaviors *(see page 53)* for further details on controller states and transitions.

Post Configuration File Management

Introduction

The file **Machine.cfg** is located in the directory */usr/cfg*.

Each parameter is specified by a variable type, variable ID, and value. The format is:

```
id[moduleType].pos[param1Id].id[param2Id].param[param3Id].paramField=value
```

where you have to modify only value.

Each parameter is defined on three lines in the Post Configuration file:

- The first line describes the internal 'path' for this parameter.
- The second line is a comment describing the parameter in a comprehensive way.
- The third line is the definition of the parameter (as described above) with its value.

Post Configuration File Generation

The Post Configuration file (**Machine.cfg**) is generated by EcoStruxure Machine Expert.

To generate the file, proceed as follows:

Step	Action
1	In the menu bar, choose **Build → Post Configuration → Generate...** **Result:** An explorer window is displayed.
2	Select the destination folder of the Post Configuration file.
3	Click **OK**.

When you use EcoStruxure Machine Expert to create a Post Configuration file (**Generate**), it reads the value of each parameter assigned in your application program and then writes the values to the **Machine.cfg** Post Configuration file. After generating a Post Configuration file, review the file and remove any parameter assignments that you wish to remain under the control of your application. Keep only those parameter assignments that you wish changed by the Post Configuration function that are necessary to make your application portable and then modify those values appropriately.

Post Configuration File Transfer

After creating and modifying your Post Configuration file, transfer it to the */usr/cfg* directory of the controller. The controller does not read the **Machine.cfg** file unless it is in this directory.

You can transfer the Post Configuration file by the following methods:

- SD card *(see page 252)* (with the proper script)
- Download through the FTP server *(see page 145)*
- Download with EcoStruxure Machine Expert controller device editor *(see page 78)*

Modifying a Post Configuration File

If the Post Configuration file is located in the PC, use a text editor to modify it.

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

To modify the Post Configuration file directly in the controller, use the **Setup** menu of the Web server *(see page 129)*.

To modify the Post Configuration file in the controller with EcoStruxure Machine Expert in online mode:

Step	Action
1	In the **Devices tree**, click the controller name.
2	Click **Build** → **Post Configuration** → **Edit...** **Result:** The Post Configuration file opens in a text editor.
3	Edit the file.
4	If you want to apply the modifications after saving them, select **Reset device after sending**.
5	Click **Save as**.
6	Click **Close**.

NOTE: If the parameters are invalid, they are ignored.

Deleting the Post Configuration File

You can delete the Post Configuration file by the following methods:

- SD card (with the delete script)
- Through the FTP server *(see page 145)*
- Online with EcoStruxure Machine Expert controller device editor *(see page 78)*, **Files** tab

For more information on **Files** tab of the Device Editor, refer to EcoStruxure Machine Expert Programming Guide.

NOTE:
The parameters defined in the application are used instead of the corresponding parameters defined in the Post Configuration file after:

- A Reset Warm command *(see page 68)*
- A Reset Cold command *(see page 68)*
- A reboot *(see page 70)*
- An application download *(see page 72)*

Post Configuration Example

Post Configuration File Example

```
# TM241CEC24T / Ethernet_1 / IPAddress
# Ethernet IP address
id[45000].pos[7].id[111].param[0] = [172, 30, 3, 99]]

# TM241CEC24T / Ethernet_1 / SubnetMask
# Ethernet IP mask
id[45000].pos[7].id[111].param[1] = [255, 255, 0, 0]]

# TM241CEC24T / Ethernet_1 / GatewayAddress
# Ethernet IP gateway address
id[45000].pos[7].id[111].param[2] = [0, 0, 0, 0]]

# TM241CEC24T / Ethernet_1 / IPConfigMode
# IP configuration mode: 0:FIXED 1:BOOTP 2:DHCP
id[45000].pos[7].id[111].param[4] = 0

# TM241CEC24T / Ethernet_1 / DeviceName
# Name of the device on the Ethernet network
id[45000].pos[7].id[111].param[5] = 'my_Device'

# TM241CEC24T / Serial_Line_1 / Serial Line Configuration / Baudrate
# Serial Line Baud Rate in bit/s
id[45000].pos[8].id[40101].param[10000].Bauds = 115200

# TM241CEC24T / Serial_Line_1 / Serial Line Configuration / Parity
# Serial Line Parity (0=None, 1=Odd, 2=Even)
id[45000].pos[8].id[40101].param[10000].Parity = 0
```

```
# TM241CEC24T / Serial_Line_1 / Serial Line Configuration / DataBits
# Serial Line Data bits (7 or 8)
id[45000].pos[8].id[40101].param[10000].DataFormat = 8

# TM241CEC24T / Serial_Line_1 / Serial Line Configuration / StopBits
# Serial Line Stop bits (1 or 2)
id[45000].pos[8].id[40101].param[10000].StopBit = 1

# TM241CEC24T / Serial_Line_2 / Serial Line Configuration / Baudrate
# Serial Line Baud Rate in bit/s
id[45000].pos[9].id[40102].param[10000].Bauds = 19200

# TM241CEC24T / Serial_Line_2 / Serial Line Configuration / Parity
# Serial Line Parity (0=None, 1=Odd, 2=Even)
id[45000].pos[9].id[40102].param[10000].Parity = 2

# TM241CEC24T / Serial_Line_2 / Serial Line Configuration / DataBits
# Serial Line Data bits (7 or 8)
id[45000].pos[9].id[40102].param[10000].DataFormat = 8

# TM241CEC24T / Serial_Line_2 / Serial Line Configuration / StopBits
# Serial Line Stop bits (1 or 2)
id[45000].pos[9].id[40102].param[10000].StopBit = 1
```

Chapter 20
Connecting a Modicon M241 Logic Controller to a PC

Connecting the Controller to a PC

Overview

To transfer, run, and monitor the applications, connect the controller to a computer, that has EcoStruxure Machine Expert installed, using either a USB cable or an Ethernet connection (for those references that support an Ethernet port).

NOTICE
INOPERABLE EQUIPMENT Always connect the communication cable to the PC before connecting it to the controller. **Failure to follow these instructions can result in equipment damage.**

USB Powered Download

In order to execute limited operations, the M241 Logic Controller has the capability to be powered through the USB Mini-B port. A diode mechanism avoids having the logic controller both powered by USB and by the normal power supply, or to supply voltage on the USB port.

When powered only by USB, the logic controller executes the firmware and the boot project (if any) and the I/O board is not powered during boot (same duration as a normal boot). USB powered download initializes the internal flash memory with some firmware or some application and parameters when the controller is powered by USB. The preferred tool to connect to the controller is the **Controller Assistant**. Refer to the *EcoStruxure Machine Expert Controller Assistant User Guide*.

The controller packaging allows easy access to USB Mini-B port with minimum opening of the packaging. You can connect the controller to the PC with a USB cable. Long cables are not suitable for the USB powered download.

⚠ WARNING
INSUFFICENT POWER FOR USB DOWNLOAD Do not use a USB cable longer than 3m (9.8 ft) for USB powered download. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: It is not intended that you use the USB Powered Download on an installed controller. Depending on the number of I/O expansion modules in the physical configuration of the installed controller, there may be insufficient power from your PC USB port to accomplish the download.

USB Mini-B Port Connection

TCSXCNAMUM3P: This USB cable is suitable for short duration connections such as quick updates or retrieving data values.

BMXXCAUSBH018: Grounded and shielded, this USB cable is suitable for long duration connections.

NOTE: You can only connect 1 controller or any other device associated with EcoStruxure Machine Expert and its component to the PC at any one time.

The USB Mini-B Port is the programming port you can use to connect a PC with a USB host port using EcoStruxure Machine Expert software. Using a typical USB cable, this connection is suitable for quick updates of the program or short duration connections to perform maintenance and inspect data values. It is not suitable for long-term connections such as commissioning or monitoring without the use of specially adapted cables to help minimize electromagnetic interference.

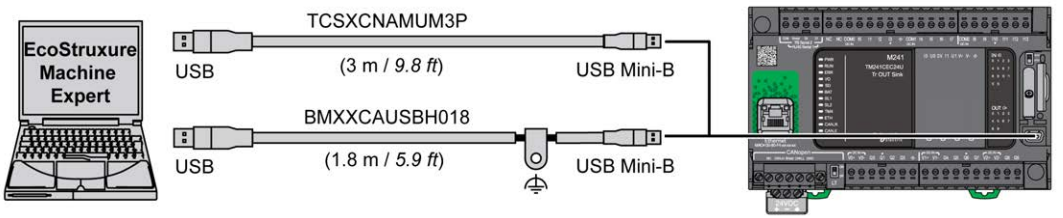
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- You must use a shielded USB cable such as a BMX XCAUSBH0•• secured to the functional ground (FE) of the system for any long-term connection.
- Do not connect more than one controller or bus coupler at a time using USB connections.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

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

The communication cable should be connected to the PC first to minimize the possibility of electrostatic discharge affecting the controller.

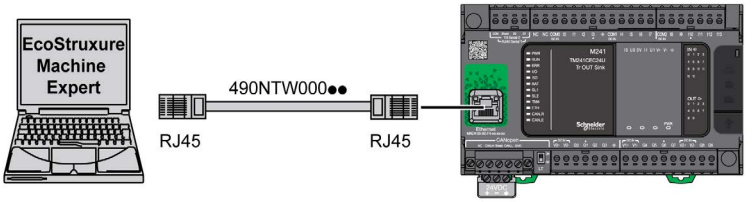


To connect the USB cable to your controller, follow the steps below:

Step	Action
1	**1a** If making a long-term connection using the cable BMXXCAUSBH018, or other cable with a ground shield connection, be sure to securely connect the shield connector to the functional ground (FE) or protective ground (PE) of your system before connecting the cable to your controller and your PC. **1b** If making a short-term connection using the cable TCSXCNAMUM3P or other non-grounded USB cable, proceed to step 2.
2	Connect your USB cable to the computer.
3	Open the hinged access cover.
4	Connect the Mini connector of your USB cable to the controller USB connector.

Ethernet Port Connection

You can also connect the controller to a PC using an Ethernet cable.



To connect the controller to the PC, do the following:

Step	Action
1	Connect the Ethernet cable to the PC.
2	Connect the Ethernet cable to the Ethernet port on the controller.

Chapter 21
SD Card

Introduction

This chapter describes how to transfer firmware, application, using an SD card to the Modicon M241 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Script Files	252
SD Card Commands	253
Updating Modicon M241 Logic Controller Firmware	260

Script Files

Overview

The following describes how to write script files (default script file or dynamic script file) to be executed from an SD card or by an application using the ExecScript function block *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*.

Script files can be used to:

- Configure the Ethernet firewall *(see page 188)*.
- Perform file transfer operations. The script files for these commands can be generated automatically and the necessary files copied to the SD card using the **Mass Storage (USB or SD Card)** command.
- Change the Modbus slave port *(see page 180)* for Modbus TCP data exchanges.

Script Syntax Guidelines

The following describes the script syntax guidelines:

- End every line of a command in the script with a ";".
- If the line begins with a ";", the line is a comment.
- The maximum number of lines in a script file is 50.
- The syntax is not case-sensitive.
- If the syntax is not respected in the script file, the script file is not executed. This means, for example, that the firewall configuration remains in the previous state.

NOTE: If the script file is not executed, a log file is generated. The log file location in the controller is */usr/Syslog/FWLog.txt*.

SD Card Commands

Introduction

The Modicon M241 Logic Controller allows file transfers with an SD card.

To upload or download files to the controller with an SD card, use one of the following methods:

- The clone function *(see page 254)* (use of an empty SD card)
- A script stored in the SD card

When an SD card is inserted into the SD card slot of the controller, the firmware searches and executes the script contained in the SD card (/sys/cmd/Script.cmd).

NOTE: The controller operation is not modified during file transfer.

For file transfer commands, the **Mass Storage (USB or SDCard)** editor lets you generate and copy the script and all necessary files into the SD card.

NOTE: The Modicon M241 Logic Controller accepts only SD cards formatted in FAT or FAT32.

The SD card must have a label. To add a label, insert the SD card in your PC, right-click on the drive in Windows Explorer and choose **Properties**.

⚠ 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 you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE
INOPERABLE EQUIPMENT • Do not interrupt the transfer of the application program or a firmware change once the transfer has begun. • Re-initiate the transfer if the transfer is interrupted for any reason. • Do not attempt to place the device into service until the file transfer has completed successfully. **Failure to follow these instructions can result in equipment damage.**

Clone Function

The clone function allows you to upload the application from one controller and to download it only to a same controller reference.

This function clones every parameter of the controller (for example applications, firmware, data file, post configuration). Refer to Memory Mapping *(see page 29)*.

NOTE: User access rights can only be copied if the **Include User Rights** button has previously been clicked on the **Clone Management** subpage of the Web server.

By default, clone is allowed without using the function block **FB_ControlClone**. If you want to restrict access to the clone feature, you can remove the access rights of the `USBExecCommand object` on **Everyone** group *(see page 86)*. As a result, cloning will be not allowed without using **FB_ControlClone**. For more details about this function block, refer to the M262 System Library Guide. For more details about Access Rights, refer to the EcoStruxure Machine Expert Programming Guide.

If you wish to control access to the cloned application in the target controller, you must use the **Include users rights** button (on the **Clone Management** subpage of the Web Server *(see page 142)*) of the source controller before doing the clone operation. For more details about Access Rights, refer to the EcoStruxure Machine Expert Programming Guide.

This procedure describes how to upload the application stored in the controller to your SD card:

Step	Action
1	Erase an SD card and set the card label as follows: **CLONExxx** **NOTE:** The label must begin with '**CLONE**' (not case sensitive), followed by any normal character.
2	Select if you want to clone the **Users Rights**. Refer to the **Clone Management** subpage *(see page 142)* of the web server.

Step	Action
3	Remove power from the controller.
4	Insert the prepared SD card in the controller.
5	Restore power to the controller. **Result:** The clone procedure starts automatically. During the clone procedure, the **PWR** and **I/O** LEDs are ON and the **SD** LED flashes regularly. **NOTE:** The clone procedure lasts 2 or 3 minutes. **Result:** At the end of the clone procedure, the SD LED is ON and the controller starts in normal application mode. If an error was detected, the ERR LED is ON and the controller is in STOPPED state.
6	Remove the SD card from the controller.

This procedure describes how to download the application stored in the SD card to your controller:

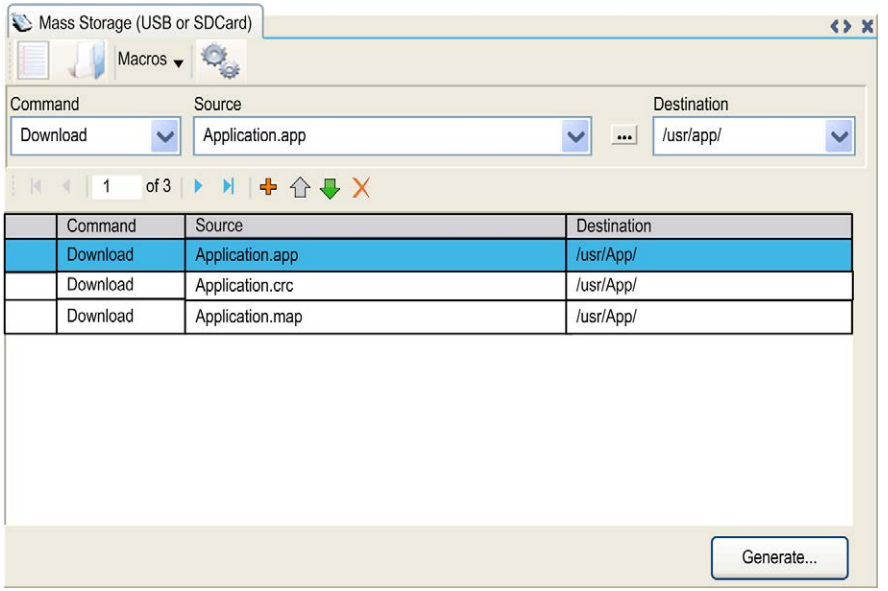
Step	Action
1	Remove power from the controller.
2	Insert the SD card into the controller.
3	Restore power to the controller. **Result:** The download procedure starts and the SD LED is flashing during this procedure.
4	Wait until the end of the download: ● If the **SD** LED (green) is ON, and the **ERR** LED (red) flashes regularly, the download ended successfully. ● If the **SD** LED (green) is OFF, and the **ERR** and **I/O** LEDs (red) flash regularly, an error is detected.
5	Remove the SD card to restart the controller.

NOTE: If you wish to control access to the cloned application in the target controller, you will need to enable and establish user access-rights, and any Web Server/FTP passwords, which are controller-specific. For more details about Access Rights, refer to the EcoStruxure Machine Expert Programming Guide.

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

Script and Files Generation with Mass Storage

Click **Project → Mass Storage (USB or SDCard)** in the main menu:



Element	Description
New	Create a new script.
Open	Open a script.
Macros	Insert a Macro. A macro is a sequence of unitary commands. A macro helps to perform many common operations such as upload application, download application, and so on.
Generate	Generate the script and all necessary files on the SD card.
Command	Basic instructions.
Source	Source file path on the PC or the controller.
Destination	Destination directory on the PC or the controller.
Add New	Add a script command.
Move Up/Down	Change the script commands order.
Delete	Delete a script command.

Commands descriptions:

Command	Description	Source	Destination	Syntax
Download	Downloads a file from the SD card to the controller.	Select the file to download.	Select the controller destination directory.	`'Download "/usr/Cfg/*"'`
SetNodeName	Sets the node name of the controller.	New node name.	Controller node name	`'SetNodeName "Name_PLC"'`
	Resets the node name of the controller.	Default node name.	Controller node name	`'SetNodeName ""'`
Upload	Uploads files contained in a controller directory to the SD card.	Select the directory.	-	`'Upload "/usr/*"'`
Delete	Deletes files contained in a controller directory. **NOTE:** Delete "*" does not delete system files.	Select the directory and enter a specific file name **Important:** by default, all directory files are selected.	-	`'Delete "/usr/SysLog/*"'`
	Removes the UserRights from the controller.	-	-	`'Delete "/usr/*"'`
	Deletes the files contained in the SD card or a folder of the SD card	-	-	`'Delete "/sd0/*"'` or `'Delete "/sd0/folder name"'`
Reboot	Restarts the controller (only available at the end of the script).	-	-	`'Reboot'`

NOTE: When User Rights are activated on a controller and if the user is not allowed to read/write/delete file system, scripts used to **Upload**/**Download**/**Delete** files are disabled. It includes the clone operation. For more details about User Rights, refer to the EcoStruxure Machine Expert Programming Guide.

This table describes the macros:

Macros	Description	Directory/Files

Download App	Download the application from the SD card to the controller.	`/usr/App/*.app`
		`/usr/App/*.crc`
		`/usr/App/*.map`
		`/usr/App/*.conf`(1)

Upload App	Upload the application from the controller to the SD card.	
(1) If OPC UA *(see page 231)* is configured.		

Macros	Description	Directory/Files
Download Sources	Download the project archive from the SD card to the controller.	`/usr/App/*.prj`
Upload Sources	Upload the project archive from the controller to the SD card.	
Download Multi-files	Download multiple files from the SD card to a controller directory.	Defined by user
Upload Log	Upload the log files from the controller to the SD card.	`/usr/Log/*.log`
(1) If OPC UA *(see page 231)* is configured.		

Reset the User Rights to Default

You can manually create a script to remove the user rights, along with the application, from the controller. This script must contain this command:

```
delete /usr/*
Reboot
```

NOTE: This command also removes user application and data.

Step	Action
1	Remove power from the controller.
2	Insert the prepared SD card in the source controller.
3	Restore power to the source controller. **Result:** The copy starts automatically. During the copy, the **PWR** and **I/O** LEDs are ON and the **SD** LED flashes regularly.
4	Wait until the copy is completed. **Result:** The SD LED is ON and the controller reboots with default user rights. If an error was detected, the ERR LED is ON and the controller is in STOPPED state.

Transfer Procedure

⚠ 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.

Step	Action
1	Create the script with the **Mass Storage (USB or SDCard)** editor.
2	Click **Generate...** and select the SD card root directory. **Result**: The script and files are transferred on the SD card.
3	Insert the SD card into the controller. **Result:** The transfer procedure starts and the **SD** LED is flashing during this procedure.
4	Wait until the end of the download: ● If the **SD** LED (green) is ON, and the **ERR** LED (red) flashes regularly, the download ended successfully. ● If the **SD** LED (green) is OFF, and the **ERR** and **I/O** LEDs (red) flash regularly, an error is detected.
5	Remove the SD card from the controller. **NOTE:** Changes will be applied after next restart.

When the controller has executed the script, the result is logged on the SD card (file `/sys/cmd/Cmd.log`).

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Consult the controller state and behavior diagram in this document to understand the state that will be assumed by the controller after you cycle power.

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

Updating Modicon M241 Logic Controller Firmware

Introduction

The firmware updates for Modicon M241 Logic Controller are available on the http://www.schneider-electric.com website (in .zip format).

Updating the firmware is possible by:
- Using an SD card with a compatible script file
- Using the **Controller Assistant**

Performing a firmware update deletes the current application program in the device, including the Boot Application in Flash memory.

NOTICE
LOSS OF APPLICATION DATA ● Perform a backup of the application program to the hard disk of the PC before attempting a firmware update. ● Restore the application program to the device after a successful firmware update. **Failure to follow these instructions can result in equipment damage.**

If you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE
INOPERABLE EQUIPMENT ● Do not interrupt the transfer of the application program or a firmware change once the transfer has begun. ● Re-initiate the transfer if the transfer is interrupted for any reason. ● Do not attempt to place the device into service until the file transfer has completed successfully. **Failure to follow these instructions can result in equipment damage.**

The serial line ports of your controller are configured for the Machine Expert protocol by default when new or when you update the controller firmware. The Machine Expert protocol is incompatible with that of other protocols such as Modbus Serial Line. Connecting a new controller to, or updating the firmware of a controller connected to, an active Modbus configured serial line can cause the other devices on the serial line to stop communicating. Make sure that the controller is not connected to an active Modbus serial line network before first downloading a valid application having the concerned port or ports properly configured for the intended protocol.

NOTICE
INTERRUPTION OF SERIAL LINE COMMUNICATIONS Be sure that your application has the serial line ports properly configured for Modbus before physically connecting the controller to an operational Modbus Serial Line network. **Failure to follow these instructions can result in equipment damage.**

Updating Firmware by SD Card

Follow these steps to update the firmware by an SD card:

Step	Action
1	Extract the .zip file to the root of the SD card. **NOTE:** The SD card folder \sys\cmd\ contains the download script file.
2	Remove power from the controller.
3	Insert the SD card into the controller.
4	Restore power to the controller. **NOTE:** The SD LED (green) is flashing during the operation.
5	Wait until the end of the download: ● If the **SD** LED (green) is ON, and the **ERR** LED (red) flashes regularly, the download ended successfully. ● If the **SD** LED (green) is OFF, and the **ERR** and **I/O** LEDs (red) flash regularly, an error is detected.
6	Remove the SD card from the controller. **Result**: The controller restarts automatically with new firmware if the download ended successfully.

Updating Firmware by Controller Assistant

To update the firmware, you must open the **Controller Assistant**. Click **Tools** → **External Tools** → **Open Controller Assistant**.

To execute a complete firmware update of a controller without replacing the Boot application and data, proceed as follows:

Step	Action
1	On the **Home** dialog, click the **Read from....** controller button. **Result**: The **Controller selection** dialog opens.
2	Select the required connection type and controller and click the **Reading** button. **Result**: The image is transmitted from the controller to the computer. After this has been accomplished successfully, you are automatically redirected to the **Home** dialog.
3	Click the button **New / Process...** and then **Update firmware...**. **Result**: The dialog for updating the firmware opens.
4	Execute individual steps for updating the firmware in the current image (Changes are only effected in the image on your computer). In the final step, you can decide whether you want to create a backup copy of the image read by the controller. **Result**: Following the update of the firmware, you are automatically returned to the **Home** dialog.
5	On the **Home** dialog, click the **Write on....** controller button. **Result**: The **Controller selection** dialog opens.
6	Select the required connection type and controller and click the **Write** button. **Result**: The image is transmitted from your computer to the controller. After the transmission, you are automatically returned to the **Home** dialog.

For more information about the firmware update and creating a new flash disk with firmware, refer to Project Settings - Firmware Update and Flash Memory Organization *(see page 33)*.

Chapter 22
Firmware Management

Updating TM3 Expansion Modules Firmware

Overview

The firmware update for the controller and the expansion modules are available on the *Schneider Electric* website (in .zip format).

Downloading Firmware to TM3 Expansion Modules

The firmware can be updated in:
- TM3X•HSC•
- TM3DI16 and TM3DI16G with firmware version (SV) ≥ 2.0
- TM3A• and TM3T• with firmware version (SV) ≥ 2.0

NOTE: The firmware version (SV) is found on the packaging and product labels.

Firmware updates are performed if, during a power on, at least one firmware file is present in the `/usr/TM3fwupdate/` directory of controller. You can download the file(s) to the controller using the SD card, an FTP file transfer or through EcoStruxure Machine Expert.

The controller updates the firmware of the TM3 expansion modules on the I/O bus, including those that are:
- Connected remotely, using a TM3 Transmitter/Receiver module.
- In configurations comprising a mix of TM3 and TM2 expansion modules.

The following table describes how to download firmware to one or more TM3 expansion modules using an SD card:

Step	Action
1	Insert an empty SD card into the PC.
2	Create the folder path `/sys/Cmd` and create a file called `Script.cmd`.
3	Edit the file and insert the following command for each firmware file you wish to transfer to the controller: `Download "usr/TM3fwupdate/<filename>"`
4	Create the folder path `/usr/TM3fwupdate/` in the SD card root directory and copy the firmware files to the `TM3fwupdate` folder.
5	Ensure that power is removed from controller.
6	Remove the SD card from the PC and insert it into the SD card slot of the controller.

Step	Action
7	Restore power to the controller. Wait until the end of the operation (until the **SD** LED is green ON). **Result:** The controller begins transferring the firmware file(s) from the SD card to the `/usr/TM3fwupdate` in the controller. During this operation, the **SD** LED on the controller is flashing. A `SCRIPT.log` file is created on the SD card and contains the result of the file transfer. If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `SCRIPT.log` file.
8	Remove power from the controller.
9	Remove SD card from the controller.
10	Restore power to the controller. **Result:** The controller transfers the firmware file(s) to the appropriate TM3 I/O module(s). **NOTE:** The TM3 update process adds approximatively 15 seconds to the controller boot duration.
11	Verify in the message logger of the controller that the firmware is successfully updated: `Your TM3 Module X successfully updated`. X corresponds to the position of the module on the bus. **NOTE:** You can also obtain the logger information in the `PlcLog.txt` file in the `/usr/Syslog/` directory of the controller file system. **NOTE:** If the controller encounters an error during the update, the update terminates with that module.
12	If all targeted modules were successfully updated, delete the firmware file(s) from `/usr/TM3fwupdate/` folder on the controller. You can delete the files directly using EcoStruxure Machine Expert or by creating and executing a script containing the following command: `Delete "usr/TM3fwupdate/*"` **NOTE:** If a targeted module was not updated successfully, or there are no message logger messages for all the targeted modules, see the Recovery Procedure *(see page 265)* below.

Recovery Procedure

If you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE
INOPERABLE EQUIPMENT • Do not interrupt the transfer of the application program or a firmware change once the transfer has begun. • Re-initiate the transfer if the transfer is interrupted for any reason. • Do not attempt to place the device into service until the file transfer has completed successfully. **Failure to follow these instructions can result in equipment damage.**

If, during the reattempted firmware update, the update prematurely terminates with an error, it means that the communication interruption or power outage had damaged the firmware of one of your modules in your configuration, and that module must be reinitialized.

NOTE: Once the firmware update process detects an error with the firmware in the destination module, the update process is terminated. After you have reinitialized the damaged module following the recovery procedure, any modules that followed the damaged module remain unchanged and will need to have their firmware updated.

The following table describes how to reinitialize the firmware on TM3 expansion modules:

Step	Action
1	Ensure that the correct firmware is present in the `/usr/TM3fwupdate/` directory of the controller.
2	Remove power from the controller.
3	Disassemble from the controller all TM3 expansion modules that are functioning normally, up to the first module to recover. Refer to the hardware guides of the modules for disassembly instructions.
4	Apply power to the controller. **NOTE:** The TM3 update process adds approximatively 15 seconds to the controller boot duration.
5	Verify in the message logger of the controller that the firmware is successfully updated: `Your TM3 Module X successfully updated`. X corresponds to the position of the module on the bus.
6	Remove power from the controller.

Step	Action
7	Reassemble the TM3 expansion module configuration to the controller. Refer to the hardware guides of the modules for assembly instructions.
8	Restore power to the controller. **Result:** The controller transfers the firmware file(s) to the appropriate and yet to be updated TM3 I/O module(s). **NOTE:** The TM3 update process adds approximatively 15 seconds to the controller boot duration.
9	Verify in the message logger of the controller that the firmware is successfully updated: `Your TM3 Module X successfully updated`. X corresponds to the position of the module on the bus. **NOTE:** You can also obtain the logger information in the `Sys.log` file in the `/usr/Log` directory of the controller file system.
10	Delete the firmware file(s) from `/usr/TM3fwupdate/` folder on the controller.

Appendices

Overview

This appendix lists the documents necessary for technical understanding of the Modicon M241 Logic Controller Programming Guide.

What Is in This Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	How to Change the IP Address of the Controller	269
B	Functions to Get/Set Serial Line Configuration in User Program	273
C	Controller Performance	279

Appendix A
How to Change the IP Address of the Controller

changeIPAddress: Change the IP address of the controller

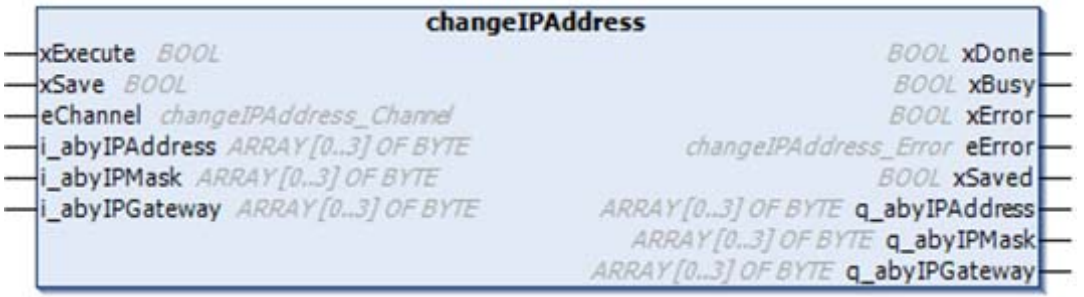
Function Block Description

The `changeIPAddress` function block provides the capability to change dynamically a controller IP address, its subnet mask and its gateway address. The function block can also save the IP address so that it is used in subsequent reboots of the controller.

NOTE: Changing the IP addresses is only possible if the IP mode is configured to **fixed IP address**. For more details, refer to IP Address Configuration *(see page 121)*.

NOTE: For more information on the function block, use the **Documentation** tab of EcoStruxure Machine Expert Library Manager Editor. For the use of this editor, refer EcoStruxure Machine Expert Programming Guide *(see EcoStruxure Machine Expert, Functions and Libraries User Guide)*.

Graphical Representation



Parameter Description

Input	Type	Comment
xExecute	BOOL	• Rising edge: action starts. • Falling edge: resets outputs. If a falling edge occurs before the function block has completed its action, the outputs operate in the usual manner and are only reset if either the action is completed or in the event that an error is detected. In this case, the corresponding output values (xDone, xError, iError) are present at the outputs for exactly one cycle.
xSave	BOOL	TRUE: save configuration for subsequent reboots of the controller.
eChannel	changeIPAd-dress_Channel	The input eChannel is the Ethernet port to be configured. Depending on the number of the ports available on the controller, it is one of 5 values *(see page 271)* in changeIPAddress_Channel (0 or 1).
i_abyIPAddress	ARRAY[0..3] OF BYTE	The new IP Address to be configured. Format: 0.0.0.0. **NOTE:** If this input is set to 0.0.0.0 then the controller default IP addresses *(see page 124)* is configured.
i_abyIPMask	ARRAY[0..3] OF BYTE	The new subnet mask. Format: 0.0.0.0
i_abyIPGateway	ARRAY[0..3] OF BYTE	The new gateway IP address. Format: 0.0.0.0

Output	Type	Comment
xDone	BOOL	TRUE: if IP Addresses have been successfully configured or if default IP Addresses have been successfully configured because input i_abyIPAddress is set to 0.0.0.0.
xBusy	BOOL	Function block active.
xError	BOOL	• TRUE: error detected, function block aborts action. • FALSE: no error has been detected.
eError	changeIPAd-dress_Error	Error code of the detected error *(see page 271)*.
xSaved	BOOL	Configuration saved for the subsequent reboots of the controller.
q_abyIPAddress	ARRAY[0..3] OF BYTE	Current controller IP address. Format: 0.0.0.0.
q_abyIPMask	ARRAY[0..3] OF BYTE	Current subnet mask. Format: 0.0.0.0.
q_abyIPGateway	ARRAY[0..3] OF BYTE	Current gateway IP address. Format: 0.0.0.0.

changeIPAddress_Channel: Ethernet port to be configured

The changeIPAddress_Channel enumeration data type contains the following values:

Enumerator	Value	Description
CHANNEL_ETHERNET_NETWORK	0	M241, M251MESC, M258, LMC058, LMC078: **Ethernet port** M251MESE: **Ethernet_2 port**
CHANNEL_DEVICE_NETWORK	1	M241: **TM4ES4 Ethernet port** M251MESE: **Ethernet_1 port**

changeIPAddress_Error: Error Codes

The changeIPAddress_Error enumeration data type contains the following values:

Enumerator	Value	Description
ERR_NO_ERROR	00 hex	No error detected.
ERR_UNKNOWN	01 hex	Internal error detected.
ERR_INVALID_MODE	02 hex	IP address is not configured as a fixed IP address.
ERR_INVALID_IP	03 hex	Invalid IP address.
ERR_DUPLICATE_IP	04 hex	The new IP address is already used in the network.
ERR_WRONG_CHANNEL	05 hex	Incorrect Ethernet communication port.
ERR_IP_BEING_SET	06 hex	IP address is already being changed.
ERR_SAVING	07 hex	IP addresses not saved due to a detected error or no non-volatile memory present.
ERR_DHCP_SERVER	08 hex	A DHCP server is configured on this Ethernet communication port.

Appendix B
Functions to Get/Set Serial Line Configuration in User Program

Overview

This section describes the functions to get/set the serial line configuration in your program.

To use these functions, add the **M2xx Communication** library.

For further information on adding a library, refer to the EcoStruxure Machine Expert Programming Guide.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
GetSerialConf: Get the Serial Line Configuration	274
SetSerialConf: Change the Serial Line Configuration	275
SERIAL_CONF: Structure of the Serial Line Configuration Data Type	277

GetSerialConf: Get the Serial Line Configuration

Function Description

`GetSerialConf` returns the configuration parameters for a specific serial line communication port.

Graphical Representation



Parameter Description

Input	Type	Comment
`Link`	LinkNumber *(see EcoStruxure Machine Expert, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	`Link` is the communication port number.
`PointerToSerialConf`	POINTER TO `SERIAL_CONF` *(see page 277)*	`PointerToSerialConf` is the address of the configuration structure (variable of `SERIAL_CONF` type) in which the configuration parameters are stored. The `ADR` standard function must be used to define the associated pointer. (See the example below.)

Output	Type	Comment
`GetSerialConf`	`WORD`	This function returns: ● 0: The configuration parameters are returned ● 255: The configuration parameters are not returned because: ❍ the function was not successful ❍ the function is in progress

Example

Refer to the `SetSerialConf` *(see page 276)* example.

SetSerialConf: Change the Serial Line Configuration

Function Description

`SetSerialConf` is used to change the serial line configuration.

Graphical Representation



NOTE: Changing the configuration of the Serial Line(s) port(s) during programming execution can interrupt ongoing communications with other connected devices.

⚠ WARNING

LOSS OF CONTROL DUE TO CONFIGURATION CHANGE

Validate and test all the parameters of the `SetSerialConf` function before putting your program into service.

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

Parameter Description

Input	Type	Comment
`Link`	`LinkNumber` *(see SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	`LinkNumber` is the communication port number.
`PointerToSerialConf`	POINTER TO `SERIAL_CONF` *(see page 277)*	`PointerToSerialConf` is the address of the configuration structure (variable of `SERIAL_CONF` type) in which the new configuration parameters are stored. The `ADR` standard function must be used to define the associated pointer. (See the example below.) If 0, set the application default configuration to the serial line.

Output	Type	Comment
SetSerialConf	WORD	This function returns: ● 0: The new configuration is set ● 255: The new configuration is refused because: ❍ the function is in progress ❍ the input parameters are not valid

Example

```
VAR
  MySerialConf: SERIAL_CONF
  result: WORD;
END_VAR

(*Get current configuration of serial line 1*)
GetSerialConf(1, ADR(MySerialConf));

(*Change to modbus RTU slave address 9*)
MySerialConf.Protocol := 0;         (*Modbus RTU/Machine Expert protocol
(in this case CodesysCompliant selects the protocol)*)
MySerialConf.CodesysCompliant := 0; (*Modbus RTU*)
MySerialConf.address := 9;          (*Set modbus address to 9*)

(*Reconfigure the serial line 1*)
result := SetSerialConf(1, ADR(MySerialConf));
```

SERIAL_CONF: Structure of the Serial Line Configuration Data Type

Structure Description

The SERIAL_CONF structure contains configuration information about the serial line port. It contains these variables:

Variable	Type	Description
Bauds	DWORD	baud rate
InterframeDelay	WORD	minimum time (in ms) between 2 frames in Modbus (RTU, ASCII)
FrameReceivedTimeout	WORD	In the ASCII protocol, FrameReceivedTimeout allows the system to conclude the end of a frame at reception after a silence of the specified number of ms. If 0 this parameter is not used.
FrameLengthReceived	WORD	In the ASCII protocol, FrameLengthReceived allows the system to conclude the end of a frame at reception, when the controller received the specified number of characters. If 0, this parameter is not used.
Protocol	BYTE	0: Modbus RTU or Machine Expert (see CodesysCompliant)
		1: Modbus ASCII
		2: ASCII
Address	BYTE	Modbus address 0 to 255 (0 for Master)
Parity	BYTE	0: none
		1: odd
		2: even
Rs485	BYTE	0: RS232
		1: RS485
ModPol (polarizartion resistor)	BYTE	0: no
		1: yes
DataFormat	BYTE	7 bits or 8 bits
StopBit	BYTE	1: 1 stop bit
		2: 2 stop bits
CharFrameStart	BYTE	In the ASCII protocol, 0 means there is no start character in the frame. Otherwise, the corresponding ASCII character is used to detect the beginning of a frame in receiving mode. In sending mode, this character is added at the beginning of the user frame.
CharFrameEnd1	BYTE	In the ASCII protocol, 0 means there is no second end character in the frame. Otherwise, the corresponding ASCII character is used to detect the end of a frame in receiving mode. In sending mode, this character is added at the end of the user frame.

Variable	Type	Description
CharFrameEnd2	BYTE	In the ASCII protocol, 0 means there is no second end character in the frame. Otherwise, the corresponding ASCII character is used (along with CharFrameEnd1) to detect the end of a frame in receiving mode. In sending mode, this character is added at the end of the user frame.
CodesysCompliant	BYTE	0: Modbus RTU
		1: Machine Expert (when Protocol = 0)
CodesysNetType	BYTE	not used

Appendix C
Controller Performance

Processing Performance

Introduction

This chapter provides information about the M241 processing performance.

Logic Processing

This table presents logic processing performance for various logical instructions:

IL Instruction Type	Duration for 1000 Instructions
Addition/subtraction/multiplication of INT	42 µs
Addition/subtraction/multiplication of DINT	41 µs
Addition/subtraction/multiplication of REAL	336 µs
Division of REAL	678 µs
Operation on BOOLEAN, for example, Status:= Status and value	75 µs
LD INT + ST INT	64 µs
LD DINT + ST DINT	49 µs
LD REAL + ST REAL	50 µs

Communication and System Processing Time

The communication processing time varies, depending on the number of sent/received requests.

Response Time on Event

The response time presented in the following table represents the time between a signal rising edge on an input triggering an external task and the edge of an output set by this task. The event task also process 100 IL instructions before setting the output:

Minimum	Typical	Maximum
120 µs	200 µs	500 µs

Glossary

A

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.

application source

The collection of human-readable controller instructions, configuration data, HMI instructions, symbols, and other program documentation. The application source file is saved on the PC and you can download the application source file to most logic controllers. The application source file is used to build the executable program that runs in the logic controller.

ARP

(*address resolution protocol*) An IP network layer protocol for Ethernet that maps an IP address to a MAC (hardware) address.

B

BCD

(*binary coded decimal*) The format that represents decimal numbers between 0 and 9 with a set of 4 bits (a nybble/nibble, also titled as half byte). In this format, the 4 bits used to encode decimal numbers have an unused range of combinations.

For example, the number 2,450 is encoded as `0010 0100 0101 0000`.

BOOL

(*boolean*) A basic data type in computing. A `BOOL` variable can have one of these values: 0 (`FALSE`), 1 (`TRUE`). A bit that is extracted from a word is of type `BOOL`; for example, `%MW10.4` is a fifth bit of memory word number 10.

Boot application

(*boot application*) The binary file that contains the application. Usually, it is stored in the controller and allows the controller to boot on the application that the user has generated.

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.

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

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.

continuous function chart language

A graphical programming language (an extension of the IEC61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to inputs of other blocks to create complex expressions.

control network

A network containing logic controllers, SCADA systems, PCs, HMI, switches, ...

Two kinds of topologies are supported:

- flat: all modules and devices in this network belong to same subnet.
- 2 levels: the network is split into an operation network and an inter-controller network.

These two networks can be physically independent, but are generally linked by a routing device.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

CRC

(*cyclical redundancy check*) A method used to determine the validity of a communication transmission. The transmission contains a bit field that constitutes a checksum. The message is used to calculate the checksum by the transmitter according to the content of the message. Receiving nodes, then recalculate the field in the same manner. Any discrepancy in the value of the 2 CRC calculations indicates that the transmitted message and the received message are different.

D

data log

The controller logs events relative to the user application in a *data log*.

device network

A network that contains devices connected to a specific communication port of a logic controller. This controller is seen as a master from the devices point of view.

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.)

DINT

(*double integer type*) Encoded in 32-bit format.

DNS

(*domain name system*) The naming system for computers and devices connected to a LAN or the Internet.

DTM

(*device type manager*) Classified into 2 categories:

- Device DTMs connect to the field device configuration components.
- CommDTMs connect to the software communication components.

The DTM provides a unified structure for accessing device parameters and configuring, operating, and diagnosing the devices. DTMs can range from a simple graphical user interface for setting device parameters to a highly sophisticated application capable of performing complex real-time calculations for diagnosis and maintenance purposes.

DWORD

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

E

EDS

(*electronic data sheet*) A file for fieldbus device description that contains, for example, the properties of a device such as parameters and settings.

encoder
A device for length or angular measurement (linear or rotary encoders).

equipment
A part of a machine including sub-assemblies such as conveyors, turntables, and so on.

Ethernet
A physical and data link layer technology for LANs, also known as IEEE 802.3.

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

F

FBD
(*function block diagram*) One of 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks, where each network contains a graphical structure of boxes and connection lines, which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

FE
(functional Earth) A common grounding connection to enhance or otherwise allow normal operation of electrically sensitive equipment (also referred to as functional ground in North America).

In contrast to a protective Earth (protective ground), a functional earth connection serves a purpose other than shock protection, and may normally carry current. Examples of devices that use functional earth connections include surge suppressors and electromagnetic interference filters, certain antennas, and measurement instruments.

firmware
Represents the BIOS, data parameters, and programming instructions that constitute the operating system on a controller. The firmware is stored in non-volatile memory within the controller.

flash memory
A non-volatile memory that can be overwritten. It is stored on a special EEPROM that can be erased and reprogrammed.

freewheeling
When a logic controller is in freewheeling scan mode, a new task scan starts as soon as the previous scan has been completed. Contrast with *periodic scan mode*.

FreqGen
(*frequency generator*) A function that generates a square wave signal with programmable frequency.

FTP

(*file transfer protocol*) A standard network protocol built on a client-server architecture to exchange and manipulate files over TCP/IP based networks regardless of their size.

H

HE10

Rectangular connector for electrical signals with frequencies below 3 MHz, complying with IEC 60807-2.

I

I/O

(*input/output*)

ICMP

(*Internet control message protocol*) Reports errors detected and provides information related to datagram processing.

IEC

(*international electrotechnical commission*) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

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).

INT

(*integer*) A whole number encoded in 16 bits.

IP

(*Internet protocol* Part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

K

KeepAlive

Messages sent by the OPC UA server to keep a subscription active. This is necessary when none of the monitored items of data have been updated since the previous publication.

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).

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).

LINT

(*long integer*) A whole number encoded in a 64-bit format (4 times `INT` or 2 times `DINT`).

LRC

(*longitudinal redundancy checking*) An error-detection method for determining the correctness of transmitted and stored data.

LWORD

(*long word*) A data type encoded in a 64-bit format.

M

MAC address

(*media access control address*) A unique 48-bit number associated with a specific piece of hardware. The MAC address is programmed into each network card or device when it is manufactured.

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.

MIB

(*management information base*) An object database that is monitored by a network management system like SNMP. SNMP monitors devices are defined by their MIBs. Schneider Electric has obtained a private MIB, groupeschneider (3833).

monitored items

In OPC UA, the items of data (samples) made available by the OPC UA server that clients subscribe to.

ms

(*millisecond*)

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.

N

network

A system of interconnected devices that share a common data path and protocol for communications.

NMT

(*network management*) CANopen protocols that provide services for network initialization, detected error control, and device status control.

node

An addressable device on a communication network.

notifications

In OPC UA, messages sent by the OPC UA server to inform clients that new items of data are available.

O

open loop

Open loop control refers to a motion control system with no external sensors to provide position or velocity correction signals.

See also: *closed loop*.

P

PDO

(*process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PE

(*Protective Earth*) A common grounding connection to help avoid the hazard of electric shock by keeping any exposed conductive surface of a device at earth potential. To avoid possible voltage drop, no current is allowed to flow in this conductor (also referred to as *protective ground* in North America or as an equipment grounding conductor in the US national electrical code).

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.

PTO

(*pulse train outputs*) A fast output that oscillates between off and on in a fixed 50-50 duty cycle, producing a square wave form. PTO is especially well suited for applications such as stepper motors, frequency converters, and servo motor control, among others.

publishing interval

In OPC UA, the frequency at which the OPC_UA server sends notifications to clients informing them that data updates are available.

PWM

(*pulse width modulation*) A fast output that oscillates between off and on in an adjustable duty cycle, producing a rectangular wave form (though you can adjust it to produce a square wave).

R

REAL

A data type that is defined as a floating-point number encoded in a 32-bit format.

RJ45

A standard type of 8-pin connector for network cables defined for Ethernet.

RPDO

(*receive process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

RPI

(*requested packet interval)* The time period between cyclic data exchanges requested by the scanner. EtherNet/IP devices publish data at the rate specified by the RPI assigned to them by the scanner, and they receive message requests from the scanner with a period equal to RPI.

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.

run

A command that causes the controller to scan the application program, read the physical inputs, and write to the physical outputs according to solution of the logic of the program.

S

sampling rate

In OPC UA, the frequency at which the OPC UA server reads items of data from connected devices.

scan

A function that includes:

- reading inputs and placing the values in memory
- executing the application program 1 instruction at a time and storing the results in memory
- using the results to update outputs

SDO

(*service data object*) A message used by the field bus master to access (read/write) the object directories of network nodes in CAN-based networks. SDO types include service SDOs (SSDOs) and client SDOs (CSDOs).

SFC

(*sequential function chart*) A language that is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

SINT

(*signed integer*) A 15-bit value plus sign.

SNMP

(*simple network management protocol*) A protocol that can control a network remotely by polling the devices for their status and viewing information related to data transmission. You can also use it to manage software and databases remotely. The protocol also permits active management tasks, such as modifying and applying a new configuration.

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

STOP

A command that causes the controller to stop running an application program.

string

A variable that is a series of ASCII characters.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

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.

terminal block

(*terminal block*) The component that mounts in an electronic module and provides electrical connections between the controller and the field devices.

TPDO

(*transmit process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

U

UDINT

(*unsigned double integer*) Encoded in 32 bits.

UDP

(*user datagram protocol*) A connectionless mode protocol (defined by IETF RFC 768) in which messages are delivered in a datagram (data telegram) to a destination computer on an IP network. The UDP protocol is typically bundled with the Internet protocol. UDP/IP messages do not expect a response, and are therefore ideal for applications in which dropped packets do not require retransmission (such as streaming video and networks that demand real-time performance).

UINT

(*unsigned integer*) Encoded in 16 bits.

V

variable

A memory unit that is addressed and modified by a program.

W

watchdog

A watchdog is a special timer used to ensure that programs do not overrun their allocated scan time. The watchdog timer is usually set to a higher value than the scan time and reset to 0 at the end of each scan cycle. If the watchdog timer reaches the preset value, for example, because the program is caught in an endless loop, an error is declared and the program stopped.

WORD

A type encoded in a 16-bit format.

Index

A

ASCII Manager, *209*

C

changeIPAddress, *269*
  changing the controller IP address, *269*
changeModbusPort
  command syntax, *180*
  script example, *181*
Controller Configuration
  Communication Settings, *80*
  PLC Settings, *81*
  Services, *83*
cyclic data exchanges, generating EDS file for, *150*

D

DHCP server, *198*
Download application, *72*

E

ECU, creating for J1939, *226*
EDS file, generating, *150*
Embedded Functions Configuration
  Embedded HSC Configuration, *97*
  Embedded I/O Configuration, *87*
embedded functions configuration
  embedded pulse generators configuration, *99*
Ethernet
  changeIPAddress function block, *269*
EtherNet
  EtherNet/IP device, *149*
Ethernet
  FTP Server, *145*
  Modbus TCP Client/Server, *127*
  Modbus TCP slave device, *175*
  Services, *119*
  SNMP, *148*
  Web server, *129*
EtherNet/IP Adapter, *149*
ExecuteScript example, *181*
External Event, *47*

F

Fast Device Replacement, *199*
features
  key features, *15*
file transfer with SD card, *253*
firewall
  configuration, *186*
  default script file, *186*
  script commands, *188*
firmware
  downloading to TM3 expansion modules, *263*
FTP client, *147*
FTP Server
  Ethernet, *145*
FTPRemoteFileHandling library, *147*

G

GetSerialConf
  getting the serial line configuration, *274*

H

Hardware Initialization Values, *65*

I

I/O bus configuration, *110*

I/O configuration general information
general practices, *104*
Industrial Ethernet
overview, *194*
IP address
changeIPAddress, *269*

J

J1939
creating ECU for, *226*
interface configuration, *225*

K

KeepAlive (OPC UA), *230*
KeepAlive interval (OPC UA), *232*

L

libraries, *25*
Libraries
FTPRemoteFileHandling, *147*

M

M2•• communication
GetSerialConf, *274*
SetSerialConf, *275*
Memory Mapping, *29*
Modbus
Protocols, *127*
Modbus Ioscanner, *211*
Modbus Manager, *205*
Modbus TCP Client/Server
Ethernet, *127*
Modbus TCP port, changing, *180*
monitored items (OPC UA), *230*

O

OPC UA server
configuration, *231*
KeepAlive interval, *232*
overview, *230*
publishing interval, *232*
sampling interval, *232*
selecting symbols, *235*
symbols configuration, *234*
Output Behavior, *65*, *65*, *66*
Output Forcing, *66*

P

Post Configuration, *239*
baud rate, *240*, *240*
data bits, *240*
device name, *240*
Example, *244*
file management, *242*
gateway address, *240*
IP address, *240*
IP configuration mode, *240*
IP master name, *240*
parity, *240*
presentation, *240*
station address, *240*
stop bit, *240*
subnet mask, *240*
transfer rate, *240*
programming languages
IL, LD, Grafcet, *15*
Protocols, *119*
IP, *121*
Modbus, *127*
protocols
SNMP, *148*
publishing interval (OPC UA), *230*, *232*

R

Reboot, *70*
Remanent variables, *75*
Reset cold, *68*
Reset origin, *69*

Reset warm, *68*
Run command, *67*

S

sampling interval (OPC UA), *230*, *232*
script commands
  firewall, *188*
script file
  syntax rules, *252*
SD card
  commands, *253*
serial line
  ASCII Manager, *209*
  GetSerialConf, *274*
  Modbus Manager, *205*
  SetSerialConf, *275*
SERIAL_CONF, *277*
SetSerialConf, *275*
  setting the serial line configuration, *275*
SNMP
  Ethernet, *148*
  protocols, *148*
Software Initialization Values, *65*
State diagram, *54*
Stop command, *67*
symbols (OPC UA), *234*

T

Task
  Cyclic task, *45*
  Event task, *47*
  External Event Task, *47*
  Freewheeling task, *46*
  Types, *45*
  Watchdogs, *48*
TM3 analog I/O modules
  downloading firmware to, *263*

U

updating the firmware of TM3 expansion modules, *263*

W

Web server
  Ethernet, *129*

Modicon M241 Logic Controller

System Functions and Variables PLCSystem Library Guide

12/2019



EIO0000003065.01

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.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

© 2019 Schneider Electric. All rights reserved.

Table of Contents

Safety Information 7
About the Book 9
Chapter 1 M241 System Variables 11
1.1 System Variables: Definition and Use 12
Understanding System Variables 13
Using System Variables 15
1.2 PLC_R and PLC_W Structures 17
PLC_R: Controller Read-Only System Variables 18
PLC_W: Controller Read/Write System Variables 23
1.3 SERIAL_R and SERIAL_W Structures 24
SERIAL_R[0...1]: Serial Line Read-Only System Variables 25
SERIAL_W[0...1]: Serial Line Read/Write System Variables 26
1.4 ETH_R and ETH_W Structures 27
ETH_R: Ethernet Port Read-Only System Variables 28
ETH_W: Ethernet Port Read/Write System Variables 33
1.5 TM3_MODULE_R Structure 34
TM3_MODULE_R[0...13]: TM3 Modules Read-Only System Variables 34
1.6 TM3_BUS_W Structure 35
TM3_BUS_W: TM3 Bus System Variables 35
1.7 PROFIBUS_R Structure 36
PROFIBUS_R: PROFIBUS Read-Only System Variables 36
1.8 CART_R Structure 37
CART_R_STRUCT: Cartridge Read-Only System Variables 37
Chapter 2 M241 System Functions 39
2.1 M241 Read Functions 40
GetImmediateFastInput: Read Input of an Embedded Expert I/O 41
GetRtc: Get Real Time Clock 42
IsFirstMastColdCycle: Indicate if this Cycle is the First MAST Cold Start Cycle 43
IsFirstMastCycle: Indicate if this Cycle is the First MAST Cycle . 44
IsFirstMastWarmCycle: Indicate if this Cycle is the First MAST Warm Start Cycle 46

2.2 M241 Write Functions 47
PhysicalWriteFastOutputs: Write Fast Output of an Embedded Expert I/O 48
SetRTCDrift: Set Compensation Value to the RTC 49
2.3 M241 User Functions 51
FB_ControlClone: Clone the Controller 52
DataFileCopy: Copy File Commands 53
ExecuteScript: Run Script Commands 56
2.4 M241 Disk Space Functions 58
FC_GetFreeDiskSpace: Gets the Free Memory Space 59
FC_GetLabel: Gets the Label of Memory 60
FC_GetTotalDiskSpace: Gets the Size of Memory 61
2.5 TM3 Read Functions 62
TM3_GetModuleBusStatus: Get TM3 Module Bus Status 63
TM3_GetModuleFWVersion: Get TM3 Module Firmware Version . . 64
TM3_GetModuleInternalStatus: Get TM3 Module Internal Status 65
Chapter 3 M241 Library Data Types 67
3.1 PLC_RW System Variables Data Types 68
PLC_R_APPLICATION_ERROR: Detected Application Error Status Codes 69
PLC_R_BOOT_PROJECT_STATUS: Boot Project Status Codes 71
PLC_R_IO_STATUS: I/O Status Codes 72
PLC_R_SDCARD_STATUS: SD Card Slot Status Codes 73
PLC_R_STATUS: Controller Status Codes 74
PLC_R_STOP_CAUSE: From RUN State to Other State Transition Cause Codes 75
PLC_R_TERMINAL_PORT_STATUS: Programming Port Connection Status Codes 77
PLC_R_TM3_BUS_STATE: TM3 Bus Status Codes 78
PLC_W_COMMAND: Control Command Codes 79
3.2 DataFileCopy System Variables Data Types 80
DataFileCopyError: Detected Error Codes 81
DataFileCopyLocation: Location Codes 82
3.3 ExecScript System Variables Data Types 83
ExecuteScriptError: Detected Error Codes 83
3.4 ETH_RW System Variables Data Types 84
ETH_R_FRAME_PROTOCOL: Frame Transmission Protocol Codes . 85
ETH_R_IP_MODE: IP Address Source Codes 86
ETH_R_PORT_DUPLEX_STATUS: Transmission Mode Codes 87

ETH_R_PORT_IP_STATUS: Ethernet TCP/IP Port Status Codes . . . 88
ETH_R_PORT_LINK_STATUS: Communication Link Status Codes . 89
ETH_R_PORT_SPEED: Communication Speed of the Ethernet Port Codes 90
ETH_R_RUN_IDLE: Ethernet/IP Run and Idle States Codes 91
3.5 TM3_MODULE_RW System Variables Data Types 92
TM3_ERR_CODE: TM3 Expansion Module Detected Error Codes . . 93
TM3_MODULE_R_ARRAY_TYPE: TM3 Expansion Module Read Array Type 94
TM3_MODULE_STATE: TM3 Expansion Module State Codes 95
TM3_BUS_W_IOBUSERRMOD: TM3 bus error mode 96
3.6 Cartridge System Variables Data Types 97
CART_R_ARRAY_TYPE: Cartridge Read Array Type 98
CART_R_MODULE_ID: Cartridge Read Module Identifier 99
CART_R_STATE: Cartridge Read State 100
3.7 System Function Data Types 101
IMMEDIATE_ERR_TYPE: `GetImmediateFastInput` Read Input of Embedded Expert I/O Codes. 102
RTCSETDRIFT_ERROR: `SetRTCDrift` Function Detected Error Codes 103
Appendices 105
Appendix A Function and Function Block Representation 107
Differences Between a Function and a Function Block 108
How to Use a Function or a Function Block in IL Language 109
How to Use a Function or a Function Block in ST Language 112
Glossary 115
Index 123

Safety Information

Important Information

NOTICE

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



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



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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This document will acquaint you with the system functions and variables offered within the Modicon M241 Logic Controller. The M241 PLCSystem library contains functions and variables to get information from and send commands to the controller system.

This document describes the data type functions and variables of the M241 PLCSystem library.

The following knowledge is required:

- Basic information on the functionality, structure, and configuration of the M241 Logic Controller.
- Programming in the FBD, LD, ST, IL, or CFC language.
- system variables (global variables).

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.2.

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert Programming Guide	*EIO0000002854 (ENG)*; *EIO0000002855 (FRE)*; *EIO0000002856 (GER)*; *EIO0000002858 (SPA)*; *EIO0000002857 (ITA)*; *EIO0000002859 (CHS)*
Modicon M241 Logic Controller Hardware Guide	*EIO0000003083 (ENG)*; *EIO0000003084 (FRE)*; *EIO0000003085 (GER)*; *EIO0000003086 (SPA)*; *EIO0000003087 (ITA)*; *EIO0000003088 (CHS)*
Modicon M241 Logic Controller Programming Guide	*EIO0000003059 (ENG)*; *EIO0000003060 (FRE)*; *EIO0000003061 (GER)*; *EIO0000003062 (SPA)*; *EIO0000003063 (ITA)*; *EIO0000003064 (CHS)*

You can download these technical publications and other technical information from our website at https://www.se.com/ww/en/download/ .

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Chapter 1
M241 System Variables

Overview

This chapter:

- gives an introduction to the system variables *(see page 12)*
- describes the system variables *(see page 18)* included with the M241 PLCSystem library

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
1.1	System Variables: Definition and Use	12
1.2	PLC_R and PLC_W Structures	17
1.3	SERIAL_R and SERIAL_W Structures	24
1.4	ETH_R and ETH_W Structures	27
1.5	TM3_MODULE_R Structure	34
1.6	TM3_BUS_W Structure	35
1.7	PROFIBUS_R Structure	36
1.8	CART_R Structure	37

Section 1.1
System Variables: Definition and Use

Overview

This section defines system variables and how to implement them in the Modicon M241 Logic Controller.

What Is in This Section?

This section contains the following topics:

Topic	Page
Understanding System Variables	13
Using System Variables	15

Understanding System Variables

Introduction

This section describes how system variables are implemented. System variables:

- allow you to access general system information, perform system diagnostics, and command simple actions.
- are structured variables conforming to IEC 61131-3 definitions and naming conventions. You can access the system variables using the IEC symbolic name PLC_GVL. Some of the PLC_GVL variables are read-only (for example, PLC_R) and some are read/write (for example, PLC_W).
- are automatically declared as global variables. They have system-wide scope and can be accessed by any Program Organization Unit (POU) in any task.

Naming Convention

The system variables are identified by:

- a structure name that represents the category of system variable. For example, PLC_R represents a structure name of read-only variables used for the controller diagnostic.
- a set of component names that identifies the purpose of the variable. For example, i_wVendorID represents the controller vendor ID.

You can access the system variables by typing the structure name of the variables followed by the name of the component.

Here is an example of system variable implementation:

```
VAR
    myCtr_Serial : DWORD;
    myCtr_ID : DWORD;
    myCtr_FramesRx : UDINT;
END_VAR
```

```
myCtr_Serial := PLC_GVL.PLC_R.i_dwSerialNumber;
myCtr_ID := PLC_GVL.PLC.R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK
```

NOTE: The fully-qualified name of the system variable in the example above is PLC_GVL.PLC.R. The PLC_GVL is implicit when declaring a variable using the **Input Assistant**, but it may also be entered in full. Good programming practice often dictates the use of the fully-qualified variable name in declarations.

System Variables Location

2 kinds of system variables are defined for use when programming the controller:

- located variables
- unlocated variables

The located variables:

- have a fixed location in a static %MW area: %MW60000 to %MW60199 for read-only system variables.
- are accessible through Modbus TCP, Modbus serial, and EtherNet/IP requests both in RUNNING and STOPPED states.
- are used in EcoStruxure Machine Expert programs according to the `structure_name.component_name` convention explained previously. %MW addresses from 0 to 59999 can be accessed directly. Addresses greater than this are considered out of range by EcoStruxure Machine Expert and can only be accessed through the `structure_name.component_name` convention.

The unlocated variables:

- are not physically located in the %MW area.
- are not accessible through any fieldbus or network requests unless you locate them in the relocation table, and only then these variables can be accessed in RUNNING and STOPPED states. The relocation table uses the following dynamic %MW areas:
  - %MW60200 to %MW61999 for read-only variables
  - %MW62200 to %MW63999 for read/write variables
- are used in EcoStruxure Machine Expert programs according to the `structure_name.component_name` convention explained previously. %MW addresses from 0 to 59999 can be accessed directly. Addresses greater than this are considered out of range by EcoStruxure Machine Expert and can only be accessed through the `structure_name.component_name` convention.

Using System Variables

Introduction

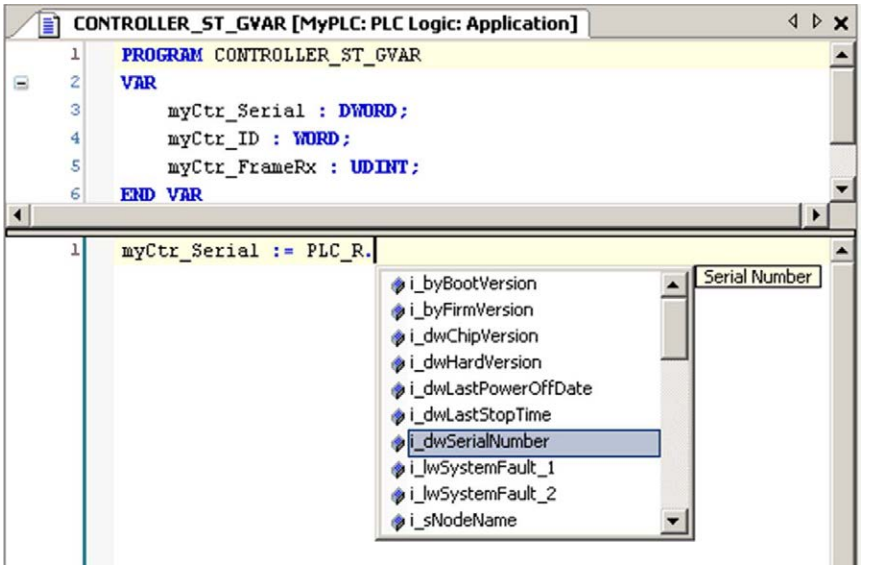
This section describes the steps required to program and to use system variables in EcoStruxure Machine Expert.

System variables are global in scope, and you can use them in all the Program Organization Units (POUs) of the application.

System variables do not need to be declared in the Global Variable List (GVL). They are automatically declared from the controller system library.

Using System Variables in a POU

EcoStruxure Machine Expert has an auto-completion feature. In a **POU**, start by entering the system variable structure name (`PLC_R`, `PLC_W`...) followed by a dot. The system variables appear in the **Input Assistant**. You can select the desired variable or enter the full name manually.



NOTE: In the example above, after you type the structure name `PLC_R.`, EcoStruxure Machine Expert offers a pop-up menu of possible component names/variables.

Example

The following example shows the use of some system variables:

```
VAR
    myCtr_Serial : DWORD;
    myCtr_ID : WORD;
    myCtr_FramesRx : UDINT;
END_VAR

myCtr_Serial := PLC_R.i_dwSerialNumber;
myCtr_ID := PLC_R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK;
```

Section 1.2
PLC_R and PLC_W Structures

Overview

This section lists and describes the different system variables included in the PLC_R and PLC_W structures.

What Is in This Section?

This section contains the following topics:

Topic	Page
PLC_R: Controller Read-Only System Variables	18
PLC_W: Controller Read/Write System Variables	23

PLC_R: Controller Read-Only System Variables

Variable Structure

This table describes the parameters of the PLC_R system variable (PLC_R_STRUCT type):

Modbus Address (1)	Var Name	Type	Comment
60000	i_wVendorID	WORD	Controller Vendor ID. 101A hex = Schneider Electric
60001	i_wProductID	WORD	Controller Reference ID. **NOTE:** Vendor ID and Reference ID are the components of the Target ID of the controller displayed in the communication settings view (Target ID = 101A XXXX hex).
60002	i_dwSerialNumber	DWORD	Controller Serial Number
60004	i_byFirmVersion	ARRAY[0..3] OF BYTE	Controller Firmware Version [aa.bb.cc.dd]: • i_byFirmVersion[0] = aa • ... • i_byFirmVersion[3] = dd
60006	i_byBootVersion	ARRAY[0..3] OF BYTE	Controller Boot Version [aa.bb.cc.dd]: • i_byBootVersion[0] = aa • ... • i_byBootVersion[3] = dd
60008	i_dwHardVersion	DWORD	Controller Hardware Version.
60010	i_dwChipVersion	DWORD	Controller Coprocessor Version.
60012	i_wStatus	PLC_R_STATUS *(see page 74)*	State of the controller.
60013	i_wBootProjectStatus	PLC_R_BOOT_PROJECT_STATUS *(see page 71)*	Returns information about the boot application stored in FLASH memory.
60014	i_wLastStopCause	PLC_R_STOP_CAUSE *(see page 75)*	Cause of the last transition from RUN to another state.
60015	i_wLastApplicationError	PLC_R_APPLICATION_ERROR *(see page 69)*	Cause of the last controller exception.

Modbus Address (1)	Var Name	Type	Comment
60016	i_lwSystemFault_1	LWORD	Bit field FFFF FFFF FFFF FFFF hex indicates no error detected. A bit at low level means that an error has been detected: ● bit 0 = Expert I/O error detected ● bit 1 = TM3 error detected ● bit 2 = Ethernet IF1 error detected ● bit 3 = Ethernet IF2 error detected ● bit 4 = Serial 1 in overcurrent error detected ● bit 5 = Serial 2 error detected ● bit 6 = CAN 1 error detected ● bit 7 = Cartridge 1 error detected ● bit 8 = Cartridge 2 error detected ● bit 9 = TM4 error detected ● bit 10 = SD Card error detected ● bit 11 = Firewall error detected ● bit 12 = DHCP server error detected ● bit 13 = OPC UA server error detected
60020	i_lwSystemFault_2	LWORD	Bit field FFFF hex indicates no error detected. If i_wIOStatus1 = PLC_R_IO_SHORTCUT_FAULT, the meaning of i_lwSystemFault_2 is: ● bit 0 = 0: short-circuit detected in output group 0 (Q0...Q1) ● bit 1 = 0: short-circuit detected in output group 1 (Q2...Q3) ● bit 2 = 0: short-circuit detected in output group 2 (Q4...Q7) ● bit 3 = 0: short-circuit detected in output group 3 (Q8...Q11) ● bit 4 = 0: short-circuit detected in output group 4 (Q12...Q15)

Modbus Address (1)	Var Name	Type	Comment
60024	i_wIOStatus1	PLC_R_IO_STATUS *(see page 72)*	Embedded Expert I/O status.
60025	i_wIOStatus2	PLC_R_IO_STATUS *(see page 72)*	TM3 I/O status.
60026	i_wClockBatterystatus	WORD	Status of the battery of the RTC: ● 0 = Battery change needed ● 100 = Battery fully charged Other values (1...99) represents the percentage of charge. For example, if the value is 75, it represents that the battery charge is 75%.
60028	i_dwAppliSignature1	DWORD	First DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.
60030	i_dwAppliSignature2	DWORD	Second DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.
60032	i_dwAppliSignature3	DWORD	Third DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.
60034	i_dwAppliSignature4	DWORD	Fourth DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.
(1) Not accessible through the application.			

n/a	i_sVendorName	STRING(31)	Name of the vendor: “Schneider Electric”.
n/a	i_sProductRef	STRING(31)	Reference of the controller.
n/a	i_sNodeName	STRING(99)	Node name on EcoStruxure Machine Expert Network.
n/a	i_dwLastStopTime	DWORD	The time of the last detected STOP in seconds beginning with January 1, 1970 at 00:00 UTC.

n/a	i_dwLastPowerOffDate	DWORD	The date and time of the last detected power off in seconds beginning with January 1, 1970 at 00:00 UTC. **NOTE:** Convert this value into date and time by using the function SysTimeRtcConvertUtcToDate. For more information about Time and Date conversion, refer to the Systime Library Guide *(see EcoStruxure Machine Expert, Getting & Setting Real Time Clock, SysTimeRtc and SysTimeCore Library Guide)*.
n/a	i_uiEventsCounter	UINT	Number of external events detected on inputs configured for external event detection since the last cold start. Reset by a Cold Start or by the PLC_W.q_wResetCounterEvent command.
n/a	i_wTerminalPort Status	PLC_R_ TERMINAL_PORT_ STATUS *(see page 77)*	Status of the USB Programming Port (USB Mini-B).
n/a	i_wSdCardStatus	PLC_R_SDCARD_ STATUS *(see page 73)*	Status of the SD card.
n/a	i_wUsrFreeFileHdl	WORD	Number of available File Handles. A File Handle is the resource allocated by the system when you open a file.
n/a	i_udiUsrFsTotalBytes	UDINT	User FileSystem total memory size (in bytes). It is the size of the flash memory for the directory "/usr/".
n/a	i_udiUsrFsFreeBytes	UDINT	User FileSystem free memory size (in bytes).

n/a	`i_uiTM3BusState`	PLC_R_TM3_BUS_STATE *(see page 78)*	TM3 bus state. `i_uiTM3BusState` can have the following values: ● 1: TM3_CONF_ERROR Configuration mismatch between physical configuration and EcoStruxure Machine Expert configuration. ● 3: TM3_OK Physical configuration matches EcoStruxure Machine Expert configuration. ● 4: TM3_POWER_SUPPLY_ERROR TM3 bus is not powered (for example when the Logic Controller is powered by USB).
n/a	`i_ExpertIO_RunStop_Input`	`BYTE`	Run/Stop input location is: ● 16...FF hex if the expert I/O is not configured ● 0 for `%IX0.0` ● 1 for `%IX0.1` ● 2 for `%IX0.2` ● ...and so on.
n/a	`i_x10msClk`	`BOOL`	TimeBase bit of 10 ms. This variable is toggling On/Off with period = 10 ms. The value toggles when the logic controller is in Stop and in Run state.
n/a	`i_x100msClk`	`BOOL`	TimeBase bit of 100 ms. This variable is toggling On/Off with period = 100 ms. The value toggles when the logic controller is in Stop and in Run state.
n/a	`i_x1sClk`	`BOOL`	TimeBase bit of 1 s. This variable is toggling On/Off with period = 1 s. The value toggles when the logic controller is in Stop and in Run state.

NOTE: n/a means that there is no pre-defined Modbus address mapping for this system variable.

PLC_W: Controller Read/Write System Variables

Variable Structure

This table describes the parameters of the PLC_W system variable (PLC_W_STRUCT type):

%MW	Var Name	Type	Comment
n/a	q_wResetCounterEvent	WORD	Transition from 0 to 1 resets the events counter (PLC_R.i_uiEventsCounter). To reset the counter again, it is necessary to write 0 to this variable before another transition from 0 to 1 can take place.
n/a	q_uiOpenPLCControl	UINT	When Value passes from 0 to 6699, the command previously written in the following PLC_W.q_wPLCControl is executed.
n/a	q_wPLCControl	PLC_W_COMMAND *(see page 79)*	Controller RUN / STOP command executed when the system variable PLC_W.q_uiOpenPLCControl value passes from 0 to 6699.

NOTE: n/a means that there is no pre-defined %MW mapping for this system variable.

Section 1.3
SERIAL_R and SERIAL_W Structures

Overview

This section lists and describes the different system variables included in the SERIAL_R and SERIAL_W structures.

What Is in This Section?

This section contains the following topics:

Topic	Page
SERIAL_R[0...1]: Serial Line Read-Only System Variables	25
SERIAL_W[0...1]: Serial Line Read/Write System Variables	26

SERIAL_R[0...1]: Serial Line Read-Only System Variables

Introduction

SERIAL_R is an array of 2 SERIAL_R_STRUCT type. Each element of the array returns diagnostic system variables for the corresponding serial line.

For the M241 Logic Controller:

- Serial_R[0] refers to the serial line 1
- Serial_R[1] refers to the serial line 2

Variable Structure

This table describes the parameters of the SERIAL_R[0...1] system variables:

%MW	Var Name	Type	Comment
Serial Line			
n/a	i_udiFramesTransmittedOK	UDINT	Number of frames successfully transmitted.
n/a	i_udiFramesReceivedOK	UDINT	Number of frames received without any errors detected.
n/a	i_udiRX_MessagesError	UDINT	Number of frames received with errors detected (checksum, parity).
Modbus Specific			
n/a	i_uiSlaveExceptionCount	UINT	Number of Modbus exception responses returned by the logic controller.
n/a	i_udiSlaveMsgCount	UINT	Number of messages received from the Master and addressed to the logic controller.
n/a	i_uiSlaveNoRespCount	UINT	Number of Modbus broadcast requests received by the logic controller.
n/a	i_uiSlaveNakCount	UINT	Not used
n/a	i_uiSlaveBusyCount	UINT	Not used
n/a	i_uiCharOverrunCount	UINT	Number of character overruns.
n/a means that there is no predefined %MW mapping for this system variable. **Not used** means that the variable is not maintained by the system, and that if the value of the variable is non-zero, it should be considered extraneous.			

The SERIAL_R counters are reset on:

- Download.
- Controller reset.
- SERIAL_W[x].q_wResetCounter command.
- Reset command by Modbus request function code number 8.

SERIAL_W[0...1]: Serial Line Read/Write System Variables

Introduction

SERIAL_W is an array of 2 SERIAL_W_STRUCT type. Each element of the array resets the SERIAL_R system variables for the corresponding serial line to be reset.

For the M241 Logic Controller:

- Serial_W[0] refers to the serial line 1
- Serial_W[1] refers to the serial line 2

Variable Structure

This table describes the parameters of the SERIAL_W[0...1] system variable:

%MW	Var Name	Type	Comment
n/a	q_wResetCounter	WORD	Transition from 0 to 1 resets all SERIAL_R[0...1] counters. To reset the counters again, it is necessary to write 0 to this variable before another transition from 0 to 1 can take place.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Section 1.4
ETH_R and ETH_W Structures

Overview

This section lists and describes the different system variables included in the ETH_R and ETH_W structures.

What Is in This Section?

This section contains the following topics:

Topic	Page
ETH_R: Ethernet Port Read-Only System Variables	28
ETH_W: Ethernet Port Read/Write System Variables	33

ETH_R: Ethernet Port Read-Only System Variables

Variable Structure

This table describes the parameters of the ETH_R system variable (ETH_R_STRUCT type):

%MW	Var Name	Type	Comment
60050	i_byIPAddress	ARRAY[0..3] OF BYTE	IP address [aaa.bbb.ccc.ddd]: • i_byIPAddress[0] = aaa • ... • i_byIPAddress[3] = ddd
60052	i_bySubNetMask	ARRAY[0..3] OF BYTE	Subnet Mask [aaa.bbb.ccc.ddd]: • i_bySub-netMask[0] = aaa • ... • i_bySub-netMask[3] = ddd
60054	i_byGateway	ARRAY[0..3] OF BYTE	Gateway address [aaa.bbb.ccc.ddd]: • i_byGateway[0] = aaa • ... • i_byGateway[3] = ddd
60056	i_byMACAddress	ARRAY[0..5] OF BYTE	MAC address [aa.bb.cc.dd.ee.ff]: • i_byMACAddress[0] = aa • ... • i_byMACAddress[5] = ff
60059	i_sDeviceName	STRING(15)	Name used to get IP address from server.
n/a	i_wIpMode	ETH_R_IP_MODE *(see page 86)*	Method used to obtain an IP address.
n/a	i_byFDRServerIPAddress	ARRAY[0..3] OF BYTE	The IP address [aaa.bbb.ccc.ddd] of the DHCP or BootP server: • i_byFDRServerIPAddress[0] = aaa • ... • i_byFDRServerIPAddress[3] = ddd Equals 0.0.0.0 if Stored IP or Default IP used.
n/a	i_udiOpenTcpConnections	UDINT	Number of open TCP connections.
n/a means that there is no predefined %MW mapping for this system variable.			

%MW	Var Name	Type	Comment
n/a	i_udiFramesTransmittedOK	UDINT	Number of frames successfully transmitted. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiFramedReceivedOK	UDINT	Number of frames successfully received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiTransmitBufferErrors	UDINT	Numbers of frames transmitted with detected errors. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiReceiveBufferErrors	UDINT	Numbers of frames received with detected errors. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_wFrameSendingProtocol	ETH_R_FRAME_ PROTOCOL *(see page 85)*	Ethernet protocol configured for frames sending (IEEE 802.3 or Ethernet II).
n/a	i_wPortALinkStatus	ETH_R_PORT_LINK_ STATUS *(see page 89)*	Link of the Ethernet Port (0 = No Link, 1 = Link connected to another Ethernet device).
n/a	i_wPortASpeed	ETH_R_PORT_SPEED *(see page 90)*	Ethernet Port network speed (10Mb/s, 100Mb/s).
n/a	i_wPortADuplexStatus	ETH_R_PORT_DUPLEX_ STATUS *(see page 87)*	Ethernet Port duplex status (0 = Half or 1 = Full duplex).
n/a	i_udiPortACollisions	UDINT	Number of frames involved in one or more collisions and subsequently transmitted successfully. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_byIPAddress_If2	ARRAY[0..3] OF BYTE	IP address of the TM4 expansion module.
n/a	i_bySubNetMask_If2	ARRAY[0..3] OF BYTE	Subnet Mask of the TM4 expansion module.
n/a	i_byGateway_If2	ARRAY[0..3] OF BYTE	Gateway address of the TM4 expansion module.
n/a	i_byMACAddress_If2	ARRAY[0..5] OF BYTE	MAC address of theTM4 expansion module.
n/a means that there is no predefined %MW mapping for this system variable.			

%MW	Var Name	Type	Comment
n/a	i_sDeviceName_If2	STRING(15)	Name used to get IP address of the TM4 expansion module.
n/a	i_wIpMode_If2	ETH_R_IP_MODE *(see page 86)*	Method used to obtain the IP address of the TM4 expansion module.
n/a	i_wPortALinkStatus_If2	ETH_R_PORT_LINK_ STATUS *(see page 89)*	Link of the TM4 expansion module Ethernet Port: ● 0: No link ● 1: Link connected to another Ethernet device
n/a	i_wPortASpeed_If2	ETH_R_PORT_SPEED *(see page 90)*	Ethernet Port network speed of the TM4 expansion module (10Mb/s or 100Mb/s).
n/a	i_wPortADuplexStatus_If2	ETH_R_PORT_DUPLEX_ STATUS *(see page 87)*	Ethernet Port duplex status of the TM4 expansion module: ● 0: Half ● 1: Full duplex
n/a	i_wPortAIpStatus_If2	ETH_R_PORT_IP_ STATUS *(see page 88)*	Ethernet TCP/IP port stack status of the TM4 expansion module.
Modbus TCP/IP Specific			
n/a	i_udiModbusMessage Transmitted	UDINT	Number of Modbus messages transmitted. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiModbusMessage Received	UDINT	Number of Modbus messages received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiModbusErrorMessage	UDINT	Modbus detected error messages transmitted and received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a means that there is no predefined %MW mapping for this system variable.			

%MW	Var Name	Type	Comment
EtherNet/IP Specific			
n/a	i_udiETHIP_IOMessaging Transmitted	UDINT	EtherNet/IP Class 1 frames transmitted. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiETHIP_IOMessagingReceived	UDINT	EtherNet/IP Class 1 frames received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiUCMM_Request	UDINT	EtherNet/IP Unconnected Messages received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiUCMM_Error	UDINT	EtherNet/IP invalid Unconnected Messages received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiClass3_Request	UDINT	EtherNet/IP Class 3 requests received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiClass3_Error	UDINT	EtherNet/IP invalid class 3 requests received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_uiAssemblyInstanceInput	UINT	Input Assembly Instance number. See the appropriate Programming Guide of your controller for more information.
n/a	i_uiAssemblyInstanceInputSize	UINT	Input Assembly Instance size. See the appropriate Programming Guide of your controller for more information.
n/a	i_uiAssemblyInstanceOutput	UINT	Output Assembly Instance number. See the appropriate Programming Guide of your controller for more information.

n/a means that there is no predefined %MW mapping for this system variable.
Not used means that the variable is not maintained by the system, and that if the value of the variable is non-zero, it should be considered extraneous.

%MW	Var Name	Type	Comment
n/a	i_uiAssemblyInstanceOutputSize	UINT	Output Assembly Instance size. See the appropriate Programming Guide of your controller for more information.
n/a	i_uiETHIP_ConnectionTimeouts	UINT	Number of connection timeouts. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_ucEipRunIdle	ETH_R_ RUN_IDLE *(see page 91)*	Run (value = 1)/Idle(value = 0) flag for EtherNet/IP class 1 connection.
n/a	i_byMasterIpTimeouts	BYTE	Ethernet Modbus TCP Master timeout events counter. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_byMasterIpLost	BYTE	Ethernet Modbus TCP Master link status: 0 = link OK, 1 = link lost.
n/a	i_wPortAIpStatus	ETH_R_ PORT_IP_ STATUS *(see page 88)*	Ethernet TCP/IP port stack status.
n/a means that there is no predefined %MW mapping for this system variable. **Not used** means that the variable is not maintained by the system, and that if the value of the variable is non-zero, it should be considered extraneous.			

NOTE: **n/a** means that there is no predefined %MW mapping for this system variable.

ETH_W: Ethernet Port Read/Write System Variables

Variable Structure

This table describes the parameters of the ETH_W system variable (ETH_W_STRUCT type):

%MW	Var Name	Type	Comment
n/a	q_wResetCounter	WORD	Transition from 0 to 1 resets all ETH_R counters. To reset again, it is necessary to write 0 to this variable before another transition from 0 to 1 can take place.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Section 1.5
TM3_MODULE_R Structure

TM3_MODULE_R[0...13]: TM3 Modules Read-Only System Variables

Introduction

The TM3_MODULE_R is an array of 14 TM3_MODULE_R_STRUCT type. Each element of the array returns diagnostic system variables for the corresponding TM3 expansion module.

For the Modicon M241 Logic Controller:

- TM3_MODULE_R[0] refers to the TM3 expansion module 0
- ...
- TM3_MODULE_R[13] refers to the TM3 expansion module 13

Variable Structure

The following table describes the parameters of the TM3_MODULE_R[0...13] system variable:

%MW	Var Name	Type	Comment
n/a	i_wProductID	WORD	TM3 expansion module ID.
n/a	i_wModuleState	TM3_MODULE_STATE *(see page 95)*	Describes the state of the TM3 module.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Section 1.6
TM3_BUS_W Structure

TM3_BUS_W: TM3 Bus System Variables

Variable Structure

This table describes the parameters of the TM3_BUS_W system variable (TM3_BUS_W_STRUCT type):

Var Name	Type	Comment
q_wIOBusErrPassiv	TM3_BUS_W_IOBUSERRMOD	When set to ERR_ACTIVE (the default), bus errors detected on TM3 expansion modules stop all I/O exchanges. When set to ERR_PASSIVE, passive I/O error handing is used: the controller attempts to continue data bus exchanges.
q_wIOBusRestart	TM3_BUS_W_IOBUSINIT	When set to 1, restarts the I/O expansion bus. This is only necessary when q_wIOBusErrPassiv is set to ERR_ACTIVE and at least one bit of TM3_MODULE_R[i].i_wModuleState is set to TM3_BUS_ERROR.

For more information, refer to I/O Configuration General Description *(see Modicon M241 Logic Controller, Programming Guide)*.

Section 1.7
PROFIBUS_R Structure

PROFIBUS_R: PROFIBUS Read-Only System Variables

Variable Structure

This table describes the parameters of the PROFIBUS_R system variable (PROFIBUS_R_STRUCT type):

%MW	Var Name	Type	Comment
n/a	i_wPNOIdentifier	WORD	Slave identification code.
n/a	i_wBusAdr	UINT	PROFIBUS slave address.
n/a	i_CommState	UDINT	Value representing the state of the PROFIBUS module: • 0x00: Unknown • 0x01: Not configured • 0x02: Stop • 0x03: Idle • 0x04: Operate
n/a	i_CommError	UDINT	Communication error code.
n/a	i_ErrorCount	UDINT	Communication error counter.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Section 1.8
CART_R Structure

CART_R_STRUCT: Cartridge Read-Only System Variables

Variable Structure

The following table describes the parameters of the CART_R_STRUCT system variable:

%MW	Var Name	Type	Comment
n/a	i_uiModuleId	CART_R_MODULE_ID *(see page 99)*	Module ID
n/a	i_uifirmwareVersion	UINT	Firmware version
n/a	i_udiCartState	CART_R_STATE *(see page 100)*	Cartridge state

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Chapter 2
M241 System Functions

Overview

This chapter describes the system functions included in the M241 PLCSystem library.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	M241 Read Functions	40
2.2	M241 Write Functions	47
2.3	M241 User Functions	51
2.4	M241 Disk Space Functions	58
2.5	TM3 Read Functions	62

Section 2.1
M241 Read Functions

Overview

This section describes the read functions included in the M241 PLCSystem library.

What Is in This Section?

This section contains the following topics:

Topic	Page
`GetImmediateFastInput`: Read Input of an Embedded Expert I/O	41
`GetRtc`: Get Real Time Clock	42
`IsFirstMastColdCycle`: Indicate if this Cycle is the First MAST Cold Start Cycle	43
`IsFirstMastCycle`: Indicate if this Cycle is the First MAST Cycle	44
`IsFirstMastWarmCycle`: Indicate if this Cycle is the First MAST Warm Start Cycle	46

GetImmediateFastInput: Read Input of an Embedded Expert I/O

Function Description

This function returns the current physical value of the input, which may be different from the current logical value of that input. The value is read immediately from the hardware at function call time. Only I0 to I7 can be accessed through this function.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
Block	INT	Not used.
Input	INT	Input Index to read from 0...7.

The following table describes the output variable:

Output	Type	Comment
GetImmediateFastInput	BOOL	Value of the input <Input> – FALSE/TRUE.

The following table describes the input/output variables:

Input/Output	Type	Comment
Error	BOOL	FALSE= operation is successful. TRUE= operation error, the function returns an invalid value.
ErrID	IMMEDIATE_ERR_TYPE *(see page 102)*	Operation error code when Error is TRUE.

GetRtc: Get Real Time Clock

Function Description

This function returns RTC time in seconds in UNIX format (time expired in seconds since January 1, 1970 at 00:00 UTC).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

The following table describes the I/O variable:

Output	Type	Comment
GetRtc	DINT	RTC in seconds in UNIX format.

Example

The following example describes how to get the RTC value:

```
VAR
    MyRTC : DINT := 0;
END_VAR
MyRTC := GetRtc();
```

IsFirstMastColdCycle: Indicate if this Cycle is the First MAST Cold Start Cycle

Function Description

This function returns TRUE during the first MAST cycle after a cold start (first cycle after download or reset cold).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

The table describes the output variable:

Output	Type	Comment
`IsFirstMastColdCycle`	`BOOL`	TRUE during the first MAST task cycle after a cold start.

Example

Refer to the function `IsFirstMastCycle` *(see page 44)*.

IsFirstMastCycle: Indicate if this Cycle is the First MAST Cycle

Function Description

This function returns TRUE during the first MAST cycle after a start.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

Output	Type	Comment
IsFirstMastCycle	BOOL	TRUE during the first MAST task cycle after a start.

Example

This example describes the three functions IsFirstMastCycle, IsFirstMastColdCycle and IsFirstMastWarmCycle used together.

Use this example in MAST task. Otherwise, it may run several times or possibly never (an additional task might be called several times or not called during 1 MAST task cycle):

```
VAR
MyIsFirstMastCycle : BOOL;
MyIsFirstMastWarmCycle : BOOL;
MyIsFirstMastColdCycle : BOOL;
END_VAR

MyIsFirstMastWarmCycle := IsFirstMastWarmCycle();
MyIsFirstMastColdCycle := IsFirstMastColdCycle();
MyIsFirstMastCycle := IsFirstMastCycle();

IF (MyIsFirstMastWarmCycle) THEN

(*This is the first Mast Cycle after a Warm Start: all variables are set to their initialization values except the Retain variables*)

(*=> initialize the needed variables so that your application runs as expected in this case*)
```

```
END_IF;

IF (MyIsFirstMastColdCycle) THEN

(*This is the first Mast Cycle after a Cold Start: all variables are set
to their initialization values including the Retain Variables*)

(*=> initialize the needed variables so that your application runs as
expected in this case*)

END_IF;

IF (MyIsFirstMastCycle) THEN

(*This is the first Mast Cycle after a Start, i.e. after a Warm or Cold
Start as well as STOP/RUN commands*)

(*=> initialize the needed variables so that your application runs as
expected in this case*)

END_IF;
```

IsFirstMastWarmCycle: Indicate if this Cycle is the First MAST Warm Start Cycle

Function Description

This function returns TRUE during the first MAST cycle after a warm start.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

This table describes the output variable:

Output	Type	Comment
IsFirstMastWarmCycle	BOOL	TRUE during the first MAST task cycle after a warm start.

Example

Refer to the function IsFirstMastCycle *(see page 44)*.

Section 2.2
M241 Write Functions

Overview

This section describes the write functions included in the M241 PLCSystem library.

What Is in This Section?

This section contains the following topics:

Topic	Page
`PhysicalWriteFastOutputs`: Write Fast Output of an Embedded Expert I/O	48
`SetRTCDrift`: Set Compensation Value to the RTC	49

PhysicalWriteFastOutputs: Write Fast Output of an Embedded Expert I/O

Function Description

This function writes a physical state to the Q0 to Q3 outputs at function call time.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
Q0Value	BOOL	Requested value for the output 0.
Q1Value	BOOL	Requested value for the output 1.
Q2Value	BOOL	Requested value for the output 2.
Q3Value	BOOL	Requested value for the output 3.

The following table describes the output variable:

Output	Type	Comment
PhysicalWriteFastOutputs	WORD	Output value of the function.

NOTE: Only the first 4 bits of the output value are significant and used as a bit field to indicate if the output is written.
If the bit corresponding to the output is 1, the output is written successfully.
If the bit corresponding to the output is 0, the output is not written because it is already used by an expert function.
If the bit corresponding to the output is 1111 bin, all of the 4 outputs are written correctly.
If the bit corresponding to the output is 1110 bin, Q0 is not written because it is used by a frequency generator.

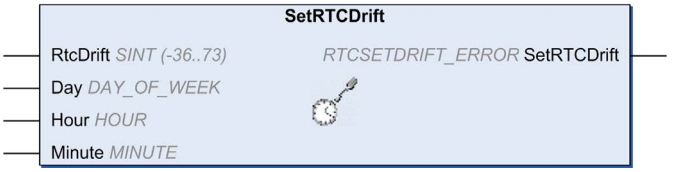
SetRTCDrift: Set Compensation Value to the RTC

Function Description

This function accelerates or slows down the frequency of the RTC to give control to the application for RTC compensation, depending on the operating environment (temperature, …). The compensation value is given in seconds per week. It can be positive (accelerate) or negative (slow down).

NOTE: The `SetRTCDrift` function must be called only once. Each new call replaces the compensation value by the new one. The value is kept in the controller hardware while the RTC is powered by the main supply or by the battery. If both battery and power supply are removed, the RTC compensation value is not available.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variables Description

This table describes the input parameters:

Inputs	Type	Comment
`RtcDrift`	`SINT(-36...+73)`	Correction in seconds per week (-36 ... +73).

NOTE: The parameters `Day`, `Hour`, and `Minute` are used only to ensure backwards compatibility.

NOTE: If the value entered for `RtcDrift` exceeds the limit value, the controller firmware sets the value to its maximum value.

This table describes the output variable:

Output	Type	Comment
`SetRTCDrift`	RTCSETDRIFT_ERROR *(see page 103)*	Returns `RTC_OK` (00 hex) if command is correct otherwise returns the ID code of the detected error.

Example

In this example, the function is called only once during the first MAST task cycle. It accelerates the RTC by 4 seconds a week (18 seconds a month).

```
VAR
    MyRTCDrift : SINT (-36...+73) := 0;
    MyDay : DAY_OF_WEEK;
    MyHour : HOUR;
    MyMinute : MINUTE;
END_VAR
```

```
IF IsFirstMastCycle() THEN
    MyRTCDrift := 4;
    MyDay := 0;
    MyHour := 0;
    MyMinute := 0;
    SetRTCDrift(MyRTCDrift, MyDay, MyHour, MyMinute);
END_IF
```

Section 2.3
M241 User Functions

Overview

This section describes the `FB_Control_Clone`, `DataFileCopy` and `ExecuteScript` functions included in the M241 PLCSystem library.

What Is in This Section?

This section contains the following topics:

Topic	Page
`FB_ControlClone`: Clone the Controller	52
`DataFileCopy`: Copy File Commands	53
`ExecuteScript`: Run Script Commands	56

FB_ControlClone: Clone the Controller

Function Block Description

Cloning is possible by SD card or **Controller Assistant**. When user rights are enabled, the cloning function is not allowed, and the function block enables cloning functionality one time on the next controller power on.

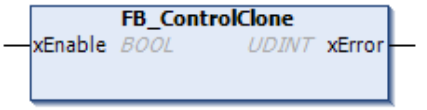
NOTE: You can choose whether user rights are included in the clone on the **Clone Management** page of the Web server *(see Modicon M241 Logic Controller, Programming Guide)*.

This table shows how to set the function block and the user rights:

Function block setting	When user rights enabled	When user rights disabled
xEnable = 1	Cloning is allowed	Cloning is still allowed
xEnable = 0	Cloning is not allowed	Cloning is not allowed

Read from controller with **Controller Assistant** is also affected by FB_ControlClone.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation (see page 107)*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
xEnable	BOOL	If TRUE, enables the cloning functionality one time. If FALSE, disables the cloning functionality.

The following table describes the output variables:

Output	Type	Comment
xError	BOOL	TRUE indicates that an error is detected and the function block aborted the action.

DataFileCopy: Copy File Commands

Function Block Description

This function block copies memory data to a file and vice versa. The file is located either within the internal file system or an external file system (SD card).

The DataFileCopy function block can:

- Read data from a formatted file or
- Copy data from memory to a formatted file. For further information, refer to Flash Memory Organization *(see Modicon M241 Logic Controller, Programming Guide)*.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation (see page 107)*.

I/O Variable Description

This table describes the input variables:

Input	Type	Comment
xExecute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when any ongoing execution terminates. **NOTE:** With the falling edge, the function continues until it concludes its execution and updates its outputs. The outputs are retained for one cycle and reset.
sFileName	STRING	File name without extension (the extension *.DTA* is automatically added). Use only a...z, A...Z, 0...9 alphanumeric characters.
xRead	BOOL	TRUE: copy data from the file identified by sFileName to the internal memory of the controller. FALSE: copy data from the internal memory of the controller to the file identified by sFileName.

Input	Type	Comment
xSecure	BOOL	TRUE: The MAC address is always stored in the file. Only a controller with the same MAC address can read from the file. FALSE: Another controller with the same type of memory can read from the file.
iLocation	INT	0: the file location is */usr/DTA* in internal file system. 1: the file location is */usr/DTA* in external file system (SD card). **NOTE:** If the file does not already exist in the directory, the file is created.

uiSize	UINT	Indicates the size in bytes. Maximum is 65534 bytes.
		Only use addresses of variables conforming to IEC 61131-3 (variables, arrays, structures), for example:
		`Variable : int;`
		`uiSize := SIZEOF (Variable);`

dwAdd	DWORD	Indicates the address in the memory that the function will read from or write to. Only use addresses of variables conforming to IEC 61131-3 (variables, arrays, structures), for example:
		`Variable : int;`
		`dwAdd := ADR (Variable);`

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Verify that the memory location is of the correct size and the file is of the correct type before copying the file to memory.

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

This table describes the output variables:

Output	Type	Comment
xDone	BOOL	TRUE = indicates that the action is successfully completed.
xBusy	BOOL	TRUE = indicates that the function block is running.
xError	BOOL	TRUE = indicates that an error is detected and the function block aborted the action.
eError	DataFileCopyError *(see page 81)*	Indicates the type of the data file copy detected error.

NOTE: If you write to memory variable within the area of the file write, a CRC integrity error results.

Example

This example describes how to copy file commands:

```
VAR
LocalArray : ARRAY [0..29] OF BYTE;
myFileName: STRING := 'exportfile';
EXEC_FLAG: BOOL;
DataFileCopy: DataFileCopy;
END_VAR
DataFileCopy(
xExecute:= EXEC_FLAG,
sFileName:= myFileName,
xRead:= FALSE,
xSecure:= FALSE,
iLocation:= DFCL_INTERNAL,
uiSize:= SIZEOF(LocalArray),
dwAdd:= ADR(LocalArray),
xDone=> ,
xBusy=> ,
xError=> ,
eError=> );
```

ExecuteScript: Run Script Commands

Function Block Description

This function block can run the following SD card script commands:

- `Download`
- `Upload`
- `SetNodeName`
- `Delete`
- `Reboot`
- `ChangeModbusPort`

For information on the required script file format, refer to Script Files for SD Cards *(see Modicon M241 Logic Controller, Programming Guide)*.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation (see page 107)*.

I/O Variable Description

This table describes the input variables:

Input	Type	Comment
xExecute	BOOL	On detection of a rising edge, starts the function block execution. On detection of a falling edge, resets the outputs of the function block when any on-going execution terminates. **NOTE:** With the falling edge, the function continues until it concludes its execution and updates its outputs. The outputs are retained for one cycle and reset.
sCmd	STRING	SD card script command syntax. Simultaneous command executions are not allowed: if a command is being executed from another function block or from an SD card script then the function block queues the command and does not execute it immediately. **NOTE:** An SD card script executed from an SD card is considered as being executed until the SD card has been removed.

This table describes the output variables:

Output	Type	Comment
xDone	BOOL	TRUE indicates that the action is successfully completed.
xBusy	BOOL	TRUE indicates that the function block is running.
xError	BOOL	TRUE indicates error detection; the function block aborts the action.
eError	ExecuteScriptError *(see page 83)*	Indicates the type of the execute script detected error.

Example

This example describes how to execute an Upload script command:

```
VAR
EXEC_FLAG: BOOL;
ExecuteScript: ExecuteScript;
END_VAR
ExecuteScript(
xExecute:= EXEC_FLAG,
sCmd:= 'Upload "/usr/Syslog/*"',
xDone=> ,
xBusy=> ,
xError=> ,
eError=> );
```

Section 2.4
M241 Disk Space Functions

Overview

This section describes the disk space functions included in the SystemInterface library.

What Is in This Section?

This section contains the following topics:

Topic	Page
`FC_GetFreeDiskSpace`: Gets the Free Memory Space	59
`FC_GetLabel`: Gets the Label of Memory	60
`FC_GetTotalDiskSpace`: Gets the Size of Memory	61

FC_GetFreeDiskSpace: Gets the Free Memory Space

Function Description

This function retrieves the amount of free memory space of a memory medium (flash disk, RAM disk, SD card) in bytes. The name of the memory medium is transferred:

- Flash disk = "ide0:"
- RAM disk = "ram0:"
- SD card = "sd0:"

The free memory space of a remote device cannot be accessed. If a remote device is specified as parameter, then the function returns "-1".

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

This table describes the input variables:

Input	Data type	Description
`i_sVolumeName`	STRING[80]	Name of the device whose free memory space must be accessed
`iq_uliFreeDiskSpace`	ULINT	Free memory space in bytes

This table describes the output variables:

Output	Data type	Description
`FC_GetFreeDiskSpace`	DINT	0: The amount of free memory space was retrieved successfully -1: Error when attempting to access the amount of free memory. For example, an invalid device or remote device was selected -318: Invalid parameter (`i_sVolumeName`)

FC_GetLabel: Gets the Label of Memory

Function Description

This function retrieves the label of a memory medium. If a device has no label, then an empty string is returned.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

This table describes the input variables:

Input	Data type	Description
i_sVolumeName	STRING[80]	Name of the device whose label must be accessed
iq_sLabel	STRING[11]	Label of the device

This table describes the output variables:

Output	Data type	Description
FC_GetLabel	DINT	0: The label was retrieved successfully -1: Error when accessing the label -318: Invalid parameter

FC_GetTotalDiskSpace: Gets the Size of Memory

Function Description

This function retrieves the size of a memory medium (flash disk, RAM disk, SD card) in bytes.

The name of the memory medium is transferred:

- Flash disk = "ide0:"
- RAM disk = "ram0:"
- SD card = "sd0:"

The size of a remote device cannot be accessed. If a remote device is specified as parameter, then the function returns "-1".

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

This table describes the input variables:

Input	Data type	Description
i_sVolumeName	STRING[80]	Name of the device whose memory size must be accessed
iq_uliTotalDiskSpace	ULINT	Size of the memory medium in byte

This table describes the output variables:

Output	Data type	Description
FC_GetTotalDiskSpace	DINT	0: Size was retrieved successfully -1: Error when reading the size -318: At least one of the parameters is invalid

Section 2.5
TM3 Read Functions

Overview

This section describes the TM3 read functions included in the M241 PLCSystem library.

What Is in This Section?

This section contains the following topics:

Topic	Page
`TM3_GetModuleBusStatus`: Get TM3 Module Bus Status	63
`TM3_GetModuleFWVersion`: Get TM3 Module Firmware Version	64
`TM3_GetModuleInternalStatus`: Get TM3 Module Internal Status	65

TM3_GetModuleBusStatus: Get TM3 Module Bus Status

Function Description

This function returns the bus status of the module. The index of the module is given as an input parameter.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
ModuleIndex	BYTE	Index of the expansion module (0 for the module closest to the controller, 1 for the second closest, and so on).

The following table describes the output variable:

Output	Type	Comment
TM3_GetModuleBusStatus	TM3_ERR_CODE *(see page 93)*	Returns TM3_NO_ERR (00 hex) if command is correct otherwise returns the ID code of the detected error.

TM3_GetModuleFWVersion: Get TM3 Module Firmware Version

Function Description

This function returns the firmware version of a specified TM3 module.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
ModuleIndex	BYTE	Index of the module (0 for the first expansion, 1 for the second, and so on).

The following table describes the output variable:

Output	Type	Comment
TM3_GetModuleFWVersion	UINT	Returns the firmware version of the module, or FFFF hex if the information cannot be read. For example, 001A hex indicates firmware version 26.

TM3_GetModuleInternalStatus: Get TM3 Module Internal Status

Function Description

This function fills the pStatusBuffer with the status table of the module ModuleIndex.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 107)*.

I/O Variable Description

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that pStatusBuffer is sufficiently allocated for the number of bytes to be read.

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

The following table describes the input variables:

Input	Type	Comment
ModuleIndex	BYTE	Index of the expansion module (0 for the module closest to the controller, 1 for the second closest, and so on)
StatusOffset	BYTE	Offset of the first status to be read in the status table.
StatusSize	BYTE	Number of bytes to be read in the status table.
pStatusBuffer	POINTER TO BYTE	Buffer containing the read status table.

The following table describes the output variable:

Output	Type	Comment
TM3_GetModuleInternalStatus	TM3_ERR_CODE *(see page 93)*	Returns TM3_NO_ERR (00 hex) if command is correct otherwise returns the ID code of the error.

Example

The following example describes how to get the module internal status:

```
VAR
AMM3HT_Channel1_Input_Status: BYTE;
END_VAR

TM3_GetModuleInternalStatus(0, 1, 1, ADR(AMM3HT_Channel1_Input_Status));
```

Chapter 3
M241 Library Data Types

Overview

This chapter describes the data types of the M241 PLCSystem library.

There are 2 kinds of data types available:

- System variable data types are used by the system variables *(see page 11)* of the M241 PLCSystem Library (PLC_R, PLC_W,...).
- System function data types are used by the read/write system functions *(see page 39)* of the M241 PLCSystem Library.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	PLC_RW System Variables Data Types	68
3.2	DataFileCopy System Variables Data Types	80
3.3	ExecScript System Variables Data Types	83
3.4	ETH_RW System Variables Data Types	84
3.5	TM3_MODULE_RW System Variables Data Types	92
3.6	Cartridge System Variables Data Types	97
3.7	System Function Data Types	101

Section 3.1
PLC_RW System Variables Data Types

Overview

This section lists and describes the system variable data types included in the PLC_R and PLC_W structures.

What Is in This Section?

This section contains the following topics:

Topic	Page
PLC_R_APPLICATION_ERROR: Detected Application Error Status Codes	69
PLC_R_BOOT_PROJECT_STATUS: Boot Project Status Codes	71
PLC_R_IO_STATUS: I/O Status Codes	72
PLC_R_SDCARD_STATUS: SD Card Slot Status Codes	73
PLC_R_STATUS: Controller Status Codes	74
PLC_R_STOP_CAUSE: From RUN State to Other State Transition Cause Codes	75
PLC_R_TERMINAL_PORT_STATUS: Programming Port Connection Status Codes	77
PLC_R_TM3_BUS_STATE: TM3 Bus Status Codes	78
PLC_W_COMMAND: Control Command Codes	79

PLC_R_APPLICATION_ERROR: Detected Application Error Status Codes

Enumerated Type Description

The PLC_R_APPLICATION_ERROR enumeration data type contains the following values:

Enumerator	Value	Comment	What to do
PLC_R_APP_ERR_UNKNOWN	FFFF hex	Undefined error detected.	Contact your Schneider Electric service representative.
PLC_R_APP_ERR_NOEXCEPTION	0000 hex	No error detected.	–
PLC_R_APP_ERR_WATCHDOG	0010 hex	Task watchdog expired.	Check your application *(see Modicon M241 Logic Controller, Programming Guide)*. A reset is needed to enter Run mode.
PLC_R_APP_ERR_ HARDWAREWATCHDOG	0011 hex	System watchdog expired.	If the problem is reproducible, verify that there are no configured but disconnected communication ports. If the problem persists, update the firmware. If the problem still persists, contact your Schneider Electric service representative.
PLC_R_APP_ERR_IO_CONFIG_ ERROR	0012 hex	Incorrect I/O configuration parameters detected.	Your application might be corrupted. To resolve this issue, use one of the methods: 1. **Build → Clean All** 2. Export/Import your application. 3. Upgrade EcoStruxure Machine Expert to the latest version.
PLC_R_APP_ERR_UNRESOLVED_ EXTREFS	0018 hex	Undefined functions detected.	Delete the unresolved functions from the application.

Enumerator	Value	Comment	What to do
PLC_R_APP_ERR_IEC_TASK_ CONFIG_ERROR	0025 hex	Incorrect Task configuration parameters detected.	Your application might be corrupted. To resolve this issue, use one of the methods: 1. **Build → Clean All** 2. Export/Import your application. 3. Upgrade EcoStruxure Machine Expert to the latest version.
PLC_R_APP_ERR_ILLEGAL_ INSTRUCTION	0050 hex	Undefined instruction detected.	Debug your application to resolve the problem.
PLC_R_APP_ERR_ACCESS_ VIOLATION	0051 hex	Attempted access to reserved memory area.	Debug your application to resolve the problem.
PLC_R_APP_ERR_DIVIDE_BY_ ZERO	0102 hex	Integer division by zero detected.	Debug your application to resolve the problem.
PLC_R_APP_ERR_PROCESSORLOAD_ WATCHDOG	0105 hex	Processor overloaded by Application Tasks.	Reduce the application workload by improving the application architecture. Increase the task cycle duration. Reduce event frequency.
PLC_R_APP_ERR_DIVIDE_REAL_ BY_ZERO	0152 hex	Real division by zero detected.	Debug your application to resolve the problem.
PLC_R_APP_ERR_EXPIO_EVENTS_ COUNT_EXCEEDED	4E20 hex	Too many events on expert I/Os are detected.	Reduce the number of event tasks.
PLC_R_APP_ERR_APPLICATION_ VERSION_MISMATCH	4E21 hex	Mismatch in the application version detected.	The application version in the logic controller does not match the version in EcoStruxure Machine Expert. Refer to Applications *(see EcoStruxure Machine Expert, Programming Guide)*.

PLC_R_BOOT_PROJECT_STATUS: Boot Project Status Codes

Enumerated Type Description

The PLC_R_BOOT_PROJECT_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_R_NO_BOOT_PROJECT	0000 hex	Boot project does not exist in non-volatile memory.
PLC_R_BOOT_PROJECT_CREATION_IN_PROGRESS	0001 hex	Boot project is being created.
PLC_R_DIFFERENT_BOOT_PROJECT	0002 hex	Boot project in non-volatile is different from the project loaded in memory.
PLC_R_VALID_BOOT_PROJECT	FFFF hex	Boot project in non-volatile memory is the same as the project loaded in memory.

PLC_R_IO_STATUS: I/O Status Codes

Enumerated Type Description

The PLC_R_IO_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_R_IO_OK	FFFF hex	Inputs/Outputs are operational.
PLC_R_IO_NO_INIT	0001 hex	Inputs/Outputs are not initialized.
PLC_R_IO_CONF_FAULT	0002 hex	Incorrect I/O configuration parameters detected.
PLC_R_IO_SHORTCUT_FAULT	0003 hex	Inputs/Outputs short-circuit detected.
PLC_R_IO_POWER_SUPPLY_FAULT	0004 hex	Inputs/Outputs power supply error detected.

PLC_R_SDCARD_STATUS: SD Card Slot Status Codes

Enumerated Type Description

The PLC_R_SDCARD_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
NO_SDCARD	0000 hex	No SD card detected in the slot or the slot is not connected.
SDCARD_READONLY	0001 hex	SD card is in read-only mode.
SDCARD_READWRITE	0002 hex	SD card is in read/write mode.
SDCARD_ERROR	0003 hex	Error detected in the SD card. More details on the error that occurred are written to the file FwLog.txt.

PLC_R_STATUS: Controller Status Codes

Enumerated Type Description

The PLC_R_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_R_EMPTY	0000 hex	Controller does not contain an application.
PLC_R_STOPPED	0001 hex	Controller is stopped.
PLC_R_RUNNING	0002 hex	Controller is running.
PLC_R_HALT	0004 hex	Controller is in a HALT state (see the controller state diagram in your controller programming guide *(see Modicon M241 Logic Controller, Programming Guide)*).
PLC_R_BREAKPOINT	0008 hex	Controller has paused at a breakpoint.

PLC_R_STOP_CAUSE: From RUN State to Other State Transition Cause Codes

Enumerated Type Description

The PLC_R_STOP_CAUSE enumeration data type contains the following values:

Enumerator	Value	Comment	What to do
PLC_R_STOP_REASON_UNKNOWN	00 hex	Initial value or stop cause is indeterminable.	Contact your local Schneider Electric representative.
PLC_R_STOP_REASON_HW_WATCHDOG	01 hex	Stopped after hardware watchdog timeout.	Contact your local Schneider Electric representative.
PLC_R_STOP_REASON_RESET	02 hex	Stopped after reset.	See reset possibilities in Controller State Diagram.
PLC_R_STOP_REASON_EXCEPTION	03 hex	Stopped after exception.	Verify your application, and correct if necessary. See System and Task Watchdogs *(see Modicon M241 Logic Controller, Programming Guide)*. A reset is needed to enter Run mode.
PLC_R_STOP_REASON_USER	04 hex	Stopped after a user request.	Refer to Stop Command in Commanding State Transitions *(see Modicon M241 Logic Controller, Programming Guide)*.
PLC_R_STOP_REASON_IECPROGRAM	05 hex	Stopped after a program command request (for example: control command with parameter PLC_W.q_wPLCControl:=PLC_W_COMMAND.PLC_W_STOP;).	–
PLC_R_STOP_REASON_DELETE	06 hex	Stopped after a remove application command.	See the **Applications** tab of the Controller Device Editor *(see Modicon M241 Logic Controller, Programming Guide)*.
PLC_R_STOP_REASON_DEBUGGING	07 hex	Stopped after entering debug mode.	–
PLC_R_STOP_FROM_NETWORK_REQUEST	0A hex	Stopped after a request from the network, the controller Web server, or PLC_W command.	–
PLC_R_STOP_FROM_INPUT	0B hex	Stop required by a controller input.	–

Enumerator	Value	Comment	What to do
PLC_R_STOP_FROM_RUN_STOP_SWITCH	0C hex	Stop required by the controller switch.	–
PLC_R_STOP_REASON_RETAIN_MISMATCH	0D hex	Stopped after an unsuccessful check context test during rebooting.	There are retained variables in non-volatile memory that do not exist in the executing application. Verify your application, correct if necessary, then reestablish the boot application.
PLC_R_STOP_REASON_BOOT_APPLI_MISMATCH	0E hex	Stopped after an unsuccessful compare between the boot application and the application that was in the memory before rebooting.	Create a valid boot application.
PLC_R_STOP_REASON_POWERFAIL	0F hex	Stopped after a power interruption.	–

For more information for reasons the controller has stopped, refer to the Controller State Description.

PLC_R_TERMINAL_PORT_STATUS: Programming Port Connection Status Codes

Enumerated Type Description

The PLC_R_TERMINAL_PORT_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
TERMINAL_NOT_CONNECTED	00 hex	No PC is connected to the programming port.
TERMINAL_CONNECTION_IN_PROGRESS	01 hex	Connection is in progress.
TERMINAL_CONNECTED	02 hex	PC is connected to the programming port.
TERMINAL_ERROR	0F hex	Error detected during connection.

PLC_R_TM3_BUS_STATE: TM3 Bus Status Codes

Enumerated Type Description

The PLC_R_TM3_BUS_STATE enumeration data type contains the following values:

Enumerator	Value	Comment
TM3_CONF_ERROR	01 hex	Error detected due to mismatch in the physical configuration and the configuration in EcoStruxure Machine Expert.
TM3_OK	03 hex	The physical configuration and the configuration in EcoStruxure Machine Expert match.
TM3_POWER_SUPPLY_ERROR	04 hex	Error detected in power supply.

PLC_W_COMMAND: Control Command Codes

Enumerated Type Description

The PLC_W_COMMAND enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_W_STOP	0001 hex	Command to stop the controller.
PLC_W_RUN	0002 hex	Command to run the controller.
PLC_W_RESET_COLD	0004 hex	Command to initiate a Controller cold reset.
PLC_W_RESET_WARM	0008 hex	Command to initiate a Controller warm reset.

Section 3.2
DataFileCopy System Variables Data Types

Overview

This section lists and describes the system variable data types included in the `DataFileCopy` structures.

What Is in This Section?

This section contains the following topics:

Topic	Page
DataFileCopyError: Detected Error Codes	81
DataFileCopyLocation: Location Codes	82

DataFileCopyError: Detected Error Codes

Enumerated Type Description

The `DataFileCopyError` enumeration data type contains the following values:

Enumerator	Value	Description
`ERR_NO_ERR`	00 hex	No error detected.
`ERR_FILE_NOT_FOUND`	01 hex	The file does not exist.
`ERR_FILE_ACCESS_REFUSED`	02 hex	The file cannot be opened.
`ERR_INCORRECT_SIZE`	03 hex	The request size is not the same as size read from file.
`ERR_CRC_ERR`	04 hex	The CRC is not correct and the file is assumed to be corrupted.
`ERR_INCORRECT_MAC`	05 hex	The controller attempting to read from the file does not have the same MAC address as that contained in the file.

DataFileCopyLocation: Location Codes

Enumerated Type Description

The `DataFileCopyLocation` enumeration data type contains the following values:

Enumerator	Value	Description
`DFCL_INTERNAL`	00 hex	Data file with DTA extension is located in */usr/Dta* directory.
`DFCL_EXTERNAL`	01 hex	Data file with DTA extension is located in */sd0/usr/Dta* directory.
`DFCL_TBD`	02 hex	Not used.

Section 3.3
ExecScript System Variables Data Types

ExecuteScriptError: Detected Error Codes

Enumerated Type Description

The `ExecuteScriptError` enumeration data type contains the following values:

Enumerator	Value	Description
`CMD_OK`	00 hex	No error detected.
`ERR_CMD_UNKNOWN`	01 hex	The command is invalid.
`ERR_SD_CARD_MISSING`	02 hex	SD card is not present.
`ERR_SEE_FWLOG`	03 hex	There was an error detected during command execution, see `FwLog.txt`. For more information, refer to File Type *(see Modicon M241 Logic Controller, Programming Guide)*.
`ERR_ONLY_ONE_COMMAND_ALLOWED`	04 hex	An attempt was made to execute several scripts simultaneously.
`CMD_BEING_EXECUTED`	05 hex	A script is already in progress.

Section 3.4
ETH_RW System Variables Data Types

Overview

This section lists and describes the system variable data types included in the ETH_R and ETH_W structures.

What Is in This Section?

This section contains the following topics:

Topic	Page
ETH_R_FRAME_PROTOCOL: Frame Transmission Protocol Codes	85
ETH_R_IP_MODE: IP Address Source Codes	86
ETH_R_PORT_DUPLEX_STATUS: Transmission Mode Codes	87
ETH_R_PORT_IP_STATUS: Ethernet TCP/IP Port Status Codes	88
ETH_R_PORT_LINK_STATUS: Communication Link Status Codes	89
ETH_R_PORT_SPEED: Communication Speed of the Ethernet Port Codes	90
ETH_R_RUN_IDLE: Ethernet/IP Run and Idle States Codes	91

ETH_R_FRAME_PROTOCOL: Frame Transmission Protocol Codes

Enumerated Type Description

The ETH_R_FRAME_PROTOCOL enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_802_3	00 hex	The protocol used for frame transmission is IEEE 802.3.
ETH_R_ETHERNET_II	01 hex	The protocol used for frame transmission is Ethernet II.

ETH_R_IP_MODE: IP Address Source Codes

Enumerated Type Description

The ETH_R_IP_MODE enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_STORED	00 hex	Stored IP address is used.
ETH_R_BOOTP	01 hex	Bootstrap protocol (BOOTP) is used to get an IP address.
ETH_R_DHCP	02 hex	DHCP protocol is used to get an IP address.
ETH_DEFAULT_IP	FF hex	Default IP address is used.

ETH_R_PORT_DUPLEX_STATUS: Transmission Mode Codes

Enumerated Type Description

The ETH_R_PORT_DUPLEX_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_PORT_HALF_DUPLEX	00 hex	Half duplex transmission mode is used.
ETH_R_FULL_DUPLEX	01 hex	Full duplex transmission mode is used.
ETH_R_PORT_NA_DUPLEX	03 hex	No duplex transmission mode is used.

ETH_R_PORT_IP_STATUS: Ethernet TCP/IP Port Status Codes

Enumerated Type Description

The ETH_R_PORT_IP_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
WAIT_FOR_PARAMS	00 hex	Waiting for parameters.
WAIT_FOR_CONF	01 hex	Waiting for configuration.
DATA_EXCHANGE	02 hex	Ready for data exchange.
ETH_ERROR	03 hex	Ethernet TCP/IP port error detected (cable disconnected, invalid configuration, and so on).
DUPLICATE_IP	04 hex	IP address already used by another device.

ETH_R_PORT_LINK_STATUS: Communication Link Status Codes

Enumerated Type Description

The ETH_R_PORT_LINK_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_LINK_DOWN	00 hex	Communication link not available to another device.
ETH_R_LINK_UP	01 hex	Communication link available to another device.

ETH_R_PORT_SPEED: Communication Speed of the Ethernet Port Codes

Enumerated Type Description

The ETH_R_PORT_SPEED enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_SPEED_NA	0 dec	Network speed is 0 megabits per second.
ETH_R_SPEED_10_MB	10 dec	Network speed is 10 megabits per second.
ETH_R_100_MB	100 dec	Network speed is 100 megabits per second.

ETH_R_RUN_IDLE: Ethernet/IP Run and Idle States Codes

Enumerated Type Description

The ETH_R_RUN_IDLE enumeration data type contains the following values:

Enumerator	Value	Comment
IDLE	00 hex	EtherNet/IP connection is idle.
RUN	01 hex	EtherNet/IP connection is running.

Section 3.5
TM3_MODULE_RW System Variables Data Types

Overview

This section lists and describes the system variable data types included in the `TM3_MODULE_R` and `TM3_MODULE_W` structures.

What Is in This Section?

This section contains the following topics:

Topic	Page
TM3_ERR_CODE: TM3 Expansion Module Detected Error Codes	93
TM3_MODULE_R_ARRAY_TYPE: TM3 Expansion Module Read Array Type	94
TM3_MODULE_STATE: TM3 Expansion Module State Codes	95
TM3_BUS_W_IOBUSERRMOD: TM3 bus error mode	96

TM3_ERR_CODE: TM3 Expansion Module Detected Error Codes

Enumerated Type Description

The TM3_ERR_CODE enumeration data type contains the following values:

Enumerator	Value	Comment
TM3_NO_ERR	00 hex	Last bus exchange with the expansion module was successful.
TM3_ERR_FAILED	01 hex	Error detected due to the last bus exchange with the expansion module was unsuccessful.
TM3_ERR_PARAMETER	02 hex	Parameter error detected in the last bus exchange with the module.
TM3_ERR_COK	03 hex	Temporary or permanent hardware error detected on one of the TM3 expansion modules.
TM3_ERR_BUS	04 hex	Bus error detected in the last bus exchange with the expansion module.

TM3_MODULE_R_ARRAY_TYPE: TM3 Expansion Module Read Array Type

Description

The TM3_MODULE_R_ARRAY_TYPE is an array of 0...13 TM3_MODULE_R_STRUCT.

TM3_MODULE_STATE: TM3 Expansion Module State Codes

Enumerated Type Description

The TM3_MODULE_STATE enumeration data type contains the following values:

Enumerator	Value	Comment
TM3_EMPTY	00 hex	No module.
TM3_CONF_ERROR	01 hex	Physical expansion module does not match with the one configured in EcoStruxure Machine Expert.
TM3_BUS_ERROR	02 hex	Bus error detected in the last exchange with the module.
TM3_OK	03 hex	Last bus exchange with this module was successful.
TM3_MISSING_OPT_MOD	05 hex	Optional module is not physically present.

TM3_BUS_W_IOBUSERRMOD: TM3 bus error mode

Enumerated Type Description

The TM3_BUS_W_IOBUSERRMOD enumeration data type contains the following values:

Enumerator	Value	Comment
IOBUS_ERR_ACTIVE	00 hex	Active mode. The logic controller stops all I/O exchanges on the TM3 bus on detection of a permanent error. Refer to I/O Configuration General Description *(see Modicon M241 Logic Controller, Programming Guide)*.
IOBUS_ERR_PASSIVE	01 hex	Passive mode. I/O exchanges continue on the TM3 bus even if an error is detected.

Section 3.6
Cartridge System Variables Data Types

Overview

This section lists and describes the system variable data types included in the `Cartridge` structure.

What Is in This Section?

This section contains the following topics:

Topic	Page
CART_R_ARRAY_TYPE: Cartridge Read Array Type	98
CART_R_MODULE_ID: Cartridge Read Module Identifier	99
CART_R_STATE: Cartridge Read State	100

CART_R_ARRAY_TYPE: Cartridge Read Array Type

Description

The CART_R_ARRAY_TYPE is an array of 0..1 CART_R_STRUCT.

CART_R_MODULE_ID: Cartridge Read Module Identifier

Enumerated Type Description

The CART_R_MODULE_ID enumeration data type contains the following values:

Enumerator	Value	Description
CART_R_MODULE_ID	40 hex	TMC4AI2
CART_R_MODULE_ID	41 hex	TMC4AQ2
CART_R_MODULE_ID	42 hex	TMC4TI2
CART_R_MODULE_ID	48 hex	TMC4HOIS01
CART_R_MODULE_ID	49 hex	TMC4PACK01
CART_R_MODULE_ID	FF hex	None

CART_R_STATE: Cartridge Read State

Enumerated Type Description

The CART_R_STATE enumeration data type contains the following values:

Enumerator	Value	Comment
CONFIGURED	00 hex	Cartridge is configured.
INITIALIZED_NOT_CONFIGURED	01 hex	Cartridge is initialized but not configured.
NOT_INITIALIZED	02 hex	Cartridge is not initialized.

Section 3.7
System Function Data Types

Overview

This section describes the different system function data types of the M241 PLCSystem library.

What Is in This Section?

This section contains the following topics:

Topic	Page
IMMEDIATE_ERR_TYPE: `GetImmediateFastInput` Read Input of Embedded Expert I/O Codes	102
RTCSETDRIFT_ERROR: `SetRTCDrift` Function Detected Error Codes	103

IMMEDIATE_ERR_TYPE: `GetImmediateFastInput` Read Input of Embedded Expert I/O Codes

Enumerated Type Description

The enumeration data type contains the following values:

Enumerator	Type	Comment
`IMMEDIATE_NO_ERROR`	Word	No errors detected.
`IMMEDIATE_UNKNOWN`	Word	The reference of `Immediate` function is incorrect or not configured.
`IMMEDIATE_UNKNOWN_PARAMETER`	Word	A parameter reference is incorrect.

RTCSETDRIFT_ERROR: `SetRTCDrift` Function Detected Error Codes

Enumerated Type Description

The RTCSETDRIFT_ERROR enumeration data type contains the following values:

Enumerator	Value	Comment
RTC_OK	00 hex	RTC drift correctly configured.
RTC_BAD_DAY	01 hex	Not used.
RTC_BAD_HOUR	02 hex	Not used.
RTC_BAD_MINUTE	03 hex	Not used.
RTC_BAD_DRIFT	04 hex	RTC Drift parameter out of range.
RTC_INTERNAL_ERROR	05 hex	RTC Drift settings rejected on internal error detected.

Appendices



Appendix A
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Differences Between a Function and a Function Block	108
How to Use a Function or a Function Block in IL Language	109
How to Use a Function or a Function Block in ST Language	112

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

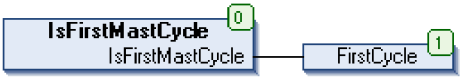
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

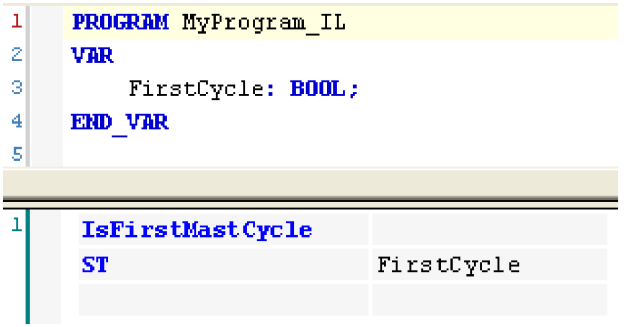
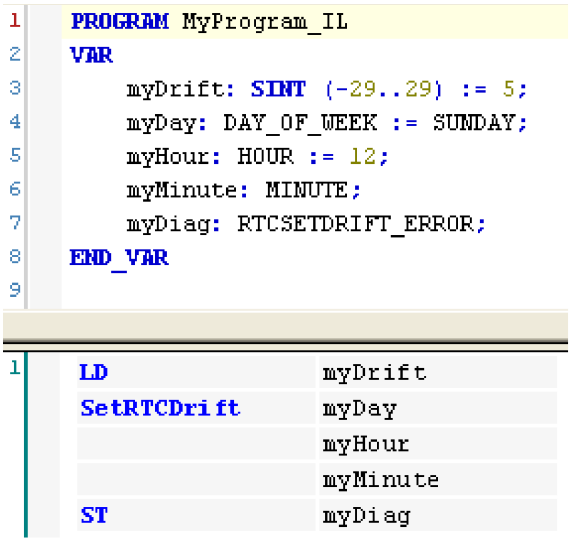
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

This procedure describes how to insert a function block in IL language:

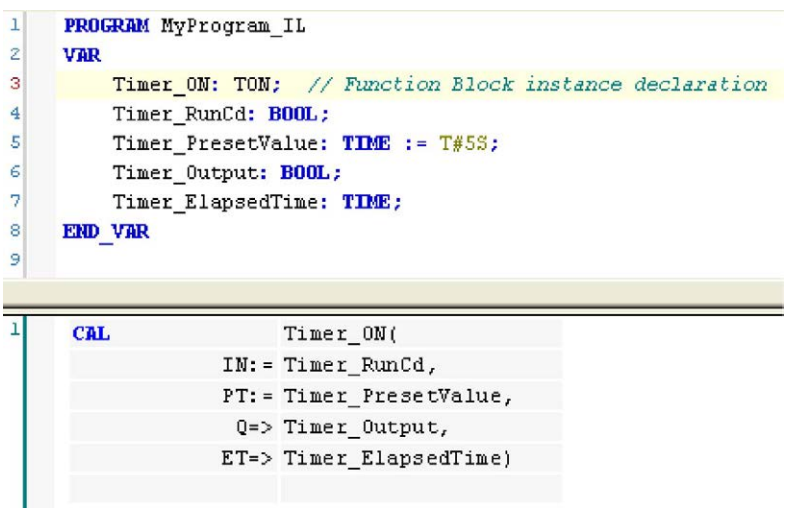
Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.

Step	Action
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a CAL instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the TON Function Block graphically presented below:

Function Block	Graphical Representation
TON	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

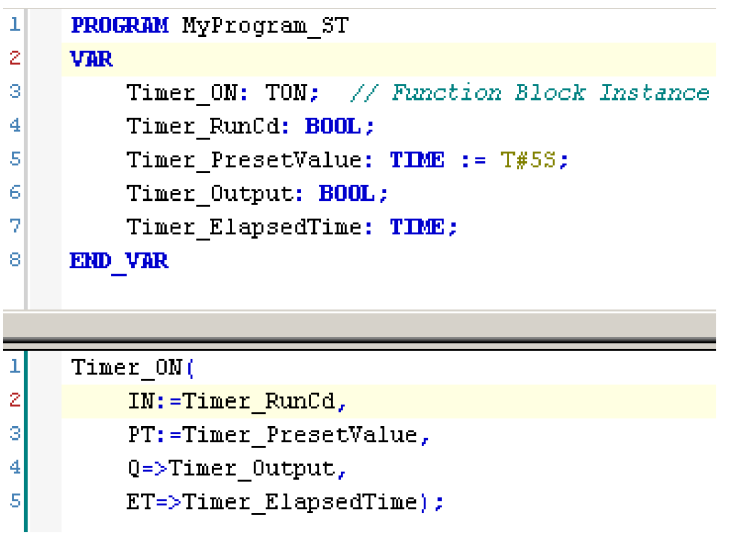
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

!

%MW

According to the IEC standard, %MW represents a memory word register (for example, a language object of type memory word).

A

application

A program including configuration data, symbols, and documentation.

ARRAY

The systematic arrangement of data objects of a single type in the form of a table defined in logic controller memory. The syntax is as follows: `ARRAY [<dimension>] OF <Type>`

Example 1: `ARRAY [1..2] OF BOOL` is a 1-dimensional table with 2 elements of type `BOOL`.

Example 2: `ARRAY [1..10, 1..20] OF INT` is a 2-dimensional table with 10 x 20 elements of type `INT`.

B

BOOL

(*boolean*) A basic data type in computing. A `BOOL` variable can have one of these values: 0 (FALSE), 1 (TRUE). A bit that is extracted from a word is of type `BOOL`; for example, `%MW10.4` is a fifth bit of memory word number 10.

Boot application

(*boot application*) The binary file that contains the application. Usually, it is stored in the controller and allows the controller to boot on the application that the user has generated.

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.

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

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.

control network

A network containing logic controllers, SCADA systems, PCs, HMI, switches, ...

Two kinds of topologies are supported:

- flat: all modules and devices in this network belong to same subnet.
- 2 levels: the network is split into an operation network and an inter-controller network.

These two networks can be physically independent, but are generally linked by a routing device.

CRC

(*cyclical redundancy check*) A method used to determine the validity of a communication transmission. The transmission contains a bit field that constitutes a checksum. The message is used to calculate the checksum by the transmitter according to the content of the message. Receiving nodes, then recalculate the field in the same manner. Any discrepancy in the value of the 2 CRC calculations indicates that the transmitted message and the received message are different.

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.)

DWORD

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

E

element

The short name of the ARRAY element.

Ethernet

A physical and data link layer technology for LANs, also known as IEEE 802.3.

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.

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

firmware

Represents the BIOS, data parameters, and programming instructions that constitute the operating system on a controller. The firmware is stored in non-volatile memory within the controller.

flash memory

A non-volatile memory that can be overwritten. It is stored on a special EEPROM that can be erased and reprogrammed.

function

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

Examples: boolean (AND) operators, calculations, conversions (BYTE_TO_INT)

function block

A programming unit that has 1 or more inputs and returns 1 or more outputs. FBs are called through an instance (function block copy with dedicated name and variables) and each instance has a persistent state (outputs and internal variables) from 1 call to the other.

Examples: timers, counters

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

G

GVL

(*global variable list*) Manages global variables within an EcoStruxure Machine Expert project.

H

hex

(*hexadecimal*)

I

I/O

(*input/output*)

ID

(*identifier/identification*)

IEC

(*international electrotechnical commission*) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

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.

IEEE 802.3

A collection of IEEE standards defining the physical layer, and the media access control sublayer of the data link layer, of wired Ethernet.

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).

INT

(*integer*) A whole number encoded in 16 bits.

IP

(*Internet protocol* Part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

L

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).

LWORD

(*long word*) A data type encoded in a 64-bit format.

M

MAC address

(*media access control address*) A unique 48-bit number associated with a specific piece of hardware. The MAC address is programmed into each network card or device when it is manufactured.

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.

N

network

A system of interconnected devices that share a common data path and protocol for communications.

P

PLC

(*programmable logic controller*) An industrial computer used to automate manufacturing, industrial, and other electromechanical processes. PLCs are different from common computers in that they are designed to have multiple input and output arrays and adhere to more robust specifications for shock, vibration, temperature, and electrical interference among other things.

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

protocol

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

R

run

A command that causes the controller to scan the application program, read the physical inputs, and write to the physical outputs according to solution of the logic of the program.

S

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

STOP

A command that causes the controller to stop running an application program.

string

A variable that is a series of ASCII characters.

system variable

A variable that provides controller data and diagnostic information and allows sending commands to the controller.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

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.

U

UDINT

(*unsigned double integer*) Encoded in 32 bits.

UINT

(*unsigned integer*) Encoded in 16 bits.

unlocated variable

A variable that does not have an address (refer to *located variable*).

V

variable

A memory unit that is addressed and modified by a program.

W

watchdog

A watchdog is a special timer used to ensure that programs do not overrun their allocated scan time. The watchdog timer is usually set to a higher value than the scan time and reset to 0 at the end of each scan cycle. If the watchdog timer reaches the preset value, for example, because the program is caught in an endless loop, an error is declared and the program stopped.

WORD

A type encoded in a 16-bit format.

Index

Specials

C

CART_R_ARRAY_TYPE
    Data Types, *98*
CART_R_MODULE_ID
    Data Types, *99*
CART_R_STATE
    Data Types, *100*
CART_R_STRUCT
    System Variable, *37*
cycle
    IsFirstMastColdCycle, *43*
    IsFirstMastCycle, *44*
    IsFirstMastWarmCycle, *46*

D

Data Types
    CART_R_ARRAY_TYPE, *98*
    CART_R_MODULE_ID, *99*
    CART_R_STATE, *100*
    DataFileCopyError, *81*
    DataFileCopyLocation, *82*
    ETH_R_FRAME_PROTOCOL, *85*
    ETH_R_IP_MODE, *86*
    ETH_R_PORT_DUPLEX_STATUS, *87*
    ETH_R_PORT_IP_STATUS, *88*
    ETH_R_PORT_LINK_STATUS, *89*
    ETH_R_PORT_SPEED, *90*
    ETH_R_RUN_IDLE, *91*
    ExecuteScriptError, *83*
    IMMEDIATE_ERR_TYPE, *102*
    PLC_R_APPLICATION_ERROR, *69*
    PLC_R_BOOT_PROJECT_STATUS, *71*
    PLC_R_IO_STATUS, *72*
    PLC_R_SDCARD_STATUS, *73*
    PLC_R_STATUS, *74*
    PLC_R_STOP_CAUSE, *75*
    PLC_R_TERMINAL_PORT_STATUS, *77*
    PLC_R_TM3_BUS_STATE, *78*
    PLC_W_COMMAND, *79*
    RTCSETDRIFT_ERROR, *103*
    TM3_BUS_W_IOBUSERRMOD, *96*
    TM3_ERR_CODE, *93*
    TM3_MODULE_R_ARRAY_TYPE, *94*
    TM3_MODULE_STATE, *95*
DataFileCopy
    copying data to or from a file, *53*
DataFileCopyError
    Data Types, *81*
DataFileCopyLocation
    Data Types, *82*

E

embedded I/O
GetImmediateFastInput, *41*
PhysicalWriteFastOutputs, *48*
ETH_R
System Variable, *28*
ETH_R_FRAME_PROTOCOL
Data Types, *85*
ETH_R_IP_MODE
Data Types, *86*
ETH_R_PORT_DUPLEX_STATUS
Data Types, *87*
ETH_R_PORT_LINK_STATUS
Data Types, *89*
ETH_R_PORT_SPEED
Data Types, *90*
ETH_W
System Variable, *33*
ExecuteScript
running script commands, *56*
ExecuteScriptError
Data Types, *83*

F

FB_ControlClone
function block, *52*
FC_GetFreeDiskSpace, *59*
FC_GetLabel, *60*
FC_GetTotalDiskSpace, *61*
file copy commands
DataFileCopy, *53*
functions
differences between a function and a function block, *108*
how to use a function or a function block in IL language, *109*
how to use a function or a function block in ST language, *112*

G

GetImmediateFastInput
getting the value of a fast input, *41*
GetRtc
getting real time clock (RTC) value, *42*

I

IMMEDIATE_ERR_TYPE
Data Types, *102*
IsFirstMastColdCycle
first cold start cycle, *43*
IsFirstMastCycle
first mast cycle, *44*
IsFirstMastWarmCycle
first warm start cycle, *46*

M

M241 PLCSystem
DataFileCopy, *53*
ExecuteScript, *56*
GetImmediateFastInput, *41*
GetRtc, *42*
IsFirstMastColdCycle, *43*
IsFirstMastCycle, *44*
IsFirstMastWarmCycle, *46*
PhysicalWriteFastOutputs, *48*
SetRTCDrift, *49*
TM3_GetModuleBusStatus, *63*, *65*
TM3_GetModuleFWVersion, *64*

P

PhysicalWriteFastOutputs
writing output of an embedded expert I/O, *48*
PLC_R
System Variable, *18*
PLC_R_APPLICATION_ERROR
Data Types, *69*
PLC_R_BOOT_PROJECT_STATUS
Data Types, *71*
PLC_R_IO_STATUS
Data Types, *72*
PLC_R_SDCARD_STATUS
Data Types, *73*

PLC_R_STATUS
Data Types, *74*
PLC_R_STOP_CAUSE
Data Types, *75*
PLC_R_TERMINAL_PORT_STATUS
Data Types, *77*
PLC_R_TM3_BUS_STATE
Data Types, *78*
PLC_W
System Variable, *23*
PLC_W_COMMAND
Data Types, *79*
PROFIBUS_R
System Variable, *36*

R

real time clock
GetRtc, *42*
SetRTCDrift, *49*
RTC
GetRtc, *42*
SetRTCDrift, *49*
RTCSETDRIFT_ERROR
Data Types, *103*

S

script commands
ExecuteScript, *56*
SERIAL_R
System Variable, *25*
SERIAL_W
System Variable, *26*
SetRTCDrift
accelerating or slowing the RTC frequency, *49*
System Variable
CART_R_STRUCT, *37*
ETH_R, *28*
ETH_W, *33*
PLC_R, *18*
PLC_W, *23*
System variable
PROFIBUS_R, *36*
System Variable
SERIAL_R, *25*
SERIAL_W, *26*
TM3_BUS_W, *35*
TM3_MODULE_R, *34*
System Variables
Definition, *13*
Using, *15*

T

TM3 module bus status
TM3_GetModuleBusStatus, *63*
TM3 module firmware version
TM3_GetModuleFWVersion, *64*
TM3 module internal status
TM3_GetModuleInternalStatus, *65*
TM3_BUS_W
system variable, *35*
TM3_BUS_W_IOBUSERRMOD
Data Types, *96*
TM3_ERR_CODE
Data Types, *93*
TM3_GetModuleBusStatus
getting the bus status of a TM3 module, *63*
TM3_GetModuleFWVersion
getting the firmware version of a TM3 module, *64*
TM3_GetModuleInternalStatus
getting the internal status of a TM3 module, *65*
TM3_MODULE_R
System Variable, *34*
TM3_MODULE_R_ARRAY_TYPE
Data Types, *94*
TM3_MODULE_STATE
Data Types, *95*

U

unction blocks
FB_ControlClone, *52*

Modicon M241 Logic Controller

High Speed Counting HSC Library Guide

12/2019



EIO0000003071.01

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.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

© 2019 Schneider Electric. All rights reserved.

Table of Contents

	Safety Information	7
	About the Book	9
Part I	Introduction	11
Chapter 1	Expert Function Introduction	13
	Expert Functions Overview	14
	Embedded Expert I/O Assignment	17
Chapter 2	High Speed Counter Types	21
	Choosing Your Counter	22
	Simple Type Overview	25
	Main Type Overview	26
	Frequency Meter Type Overview	27
	Period Meter Type Overview	28
Part II	One-shot Mode	29
Chapter 3	One-shot Mode Principle	31
	One-shot Mode Principle Description	31
Chapter 4	One-shot with a Simple Type	33
	Synopsis Diagram	34
	Configuration of the Simple Type in One-Shot Mode	35
	Programming the **Simple** Type	36
	Adjusting Parameters	38
Chapter 5	One-shot with a Main Type	39
	Synopsis Diagram	40
	Configuration of the Main Type Single Phase in One-Shot Mode	41
	Programming the **Main** Type	42
	Adjusting Parameters	45
Part III	Modulo-loop Mode	47
Chapter 6	Modulo-loop Principle	49
	Modulo-loop Mode Principle Description	49
Chapter 7	Modulo-loop with a Simple Type	53
	Synopsis Diagram	54
	Configuration of the Simple Type in Modulo-Loop Mode	55
	Programming the **Simple** Type	56
	Adjusting Parameters	58

Chapter 8 Modulo-loop with a Main Type 59
Synopsis Diagram 60
Configuration of the Main Type Single Phase in Modulo-Loop Mode . 61
Configuration of the Main Type Dual Phase in Modulo-Loop Mode. . . 62
Programming the **Main** Type 63
Adjusting Parameters. 66
Part IV Free-large Mode. 67
Chapter 9 Free-large Mode Principle 69
Free-large Mode Principle Description. 70
Limits Management 73
Chapter 10 Free-large with a Main Type. 75
Synopsis Diagram 76
Configuration of the Main Type Dual Phase in Free-Large Mode 77
Programming the **Main** Type 78
Adjusting Parameters. 81
Part V Event Counting Mode. 83
Chapter 11 Event Counting Principle 85
Event Counting Mode Principle Description. 85
Chapter 12 Event Counting with a Main Type. 87
Synopsis Diagram 88
Configuration of the Main Type Single Phase in Event Counting Mode 89
Programming the **Main** Type 90
Adjusting Parameters. 93
Part VI Frequency Meter Type 95
Chapter 13 Frequency Meter Principle 97
Description 97
Chapter 14 Frequency Meter with a Main Type 99
Synopsis Diagram 100
Configuration of the Frequency Meter Type. 101
Programming 102
Part VII Period Meter Type 105
Chapter 15 Period Meter Type Principle. 107
Description 107

Chapter 16 Period Meter with a Main Type 109
Synopsis Diagram 110
Configuration of the Period Meter Type in Edge to Edge Mode 111
Configuration of the Period Meter Type in Edge to Opposite Mode . . 112
Programming 113
Adjusting Parameters 116
Part VIII Optional Functions 117
Chapter 17 Comparison Function 119
Comparison Principle with a **Main** type 120
Configuration of the Comparison on a **Main** Type 125
External Event Configuration 126
Chapter 18 Capture Function 129
Capture Principle with a **Main** Type 130
Configuration of the Capture on a **Main** Type 132
Chapter 19 Preset and Enable Functions 133
Preset Function 134
Free-large or Period Meter Preset Conditions 136
Enable: Authorize Counting Operation 137
Appendices 139
Appendix A General Information 141
Dedicated Features 142
General Information on Administrative and Motion Function Block Management 143
Appendix B Data Types 145
EXPERT_DIAG_TYPE: Type for EXPERTGetDiag Diagnostics 146
EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block 147
EXPERT_FREQMETER_TIMEBASE_TYPE: Type for Frequency Meter Time Base Variable 148
EXPERT_HSCMAIN_TIMEBASE_TYPE: Type for HSC Main Time Base Variable 149
EXPERT_IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block 150
EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERT 151
EXPERT_PERIODMETER_RESOLUTION_TYPE: Type for Period Meter Time Base Variable 152
EXPERT_REF: EXPERT Reference Value 153

Appendix C Function Blocks 155
EXPERTGetCapturedValue: Read Value of Capture Registers 156
EXPERTGetDiag: Return Detail of a Detected HSC Error 158
EXPERTGetImmediateValue: Read Counter Value of HSC 160
EXPERTGetParam: Returns Parameters of HSC 162
EXPERTSetParam: Adjust Parameters of a HSC 164
HSCMain_M241: Control a Main Type Counter for M241 166
HSCSimple_M241: Control a Simple Type Counter for M241 170
Appendix D Function and Function Block Representation 173
Differences Between a Function and a Function Block 174
How to Use a Function or a Function Block in IL Language 175
How to Use a Function or a Function Block in ST Language. 178
Glossary 181
Index 185

Safety Information

Important Information

NOTICE

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



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



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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This documentation will acquaint you with the High Speed Counter (HSC) functions and variables offered within the M241 logic controller.

This documentation describes the functions and variables of the M241 HSC library.

In order to use this manual, you must:

- Have a thorough understanding of the M241, including its design, functionality, and implementation within control systems.
- Be proficient in the use of the following IEC 61131-3 PLC programming languages:
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - Structured Text (ST)
  - Instruction List (IL)
  - Sequential Function Chart (SFC)

EcoStruxure Machine Expert software can also be used to program these controllers using CFC (Continuous Function Chart) language.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.2.

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert Programming Guide	*EIO0000002854 (ENG)*, *EIO0000002855 (FRE)*, *EIO0000002856 (GER)*, *EIO0000002858 (SPA)*, *EIO0000002857 (ITA)*, *EIO0000002859 (CHS)*
Modicon M241 Logic Controller Programming Guide	*EIO0000003059 (ENG)*, *EIO0000003060 (FRE)*, *EIO0000003061 (GER)*, *EIO0000003062 (SPA)*, *EIO0000003063 (ITA)*, *EIO0000003064 (CHS)*

You can download these technical publications and other technical information from our website at https://www.se.com/ww/en/download/ .

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Part I
Introduction

Overview

This part provides an overview description, available modes, functionality and performances of the different functions.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Expert Function Introduction	13
2	High Speed Counter Types	21

Chapter 1
Expert Function Introduction

Overview

This chapter provides an overview description, functionality, and performances of:

- High Speed Counter (HSC)
- Pulse Train Output (PTO)
- Pulse Width Modulation (PWM)
- Frequency Generator (FreqGen)

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Expert Functions Overview	14
Embedded Expert I/O Assignment	17

Expert Functions Overview

Introduction

The inputs and outputs available on the M241 logic controller can be connected to expert functions.

The M241 logic controller supports the following expert functions:

Functions		Description
Counters	HSC Simple	The HSC functions can execute fast counts of pulses from sensors, switches, etc. that are connected to the fast or regular inputs. HSC functions connected to regular inputs operate at a maximum frequency of 1 kHz. For more information about the HSC functions, refer to High Speed Counter types *(see page 21)*.
	HSC Main Single Phase	
	HSC Main Dual Phase	
	Frequency Meter	
	Period Meter	
Pulse Generators	PTO	The PTO function provides 2 pulse train output channels to control 2 independent linear single-axis stepper or servo drives in open loop mode. The PTO function connected to regular transistor outputs operates at a maximum frequency of 1 kHz.
	PWM	The PWM function generates a square wave signal on dedicated output channels with a variable duty cycle. The PWM function connected to regular transistor outputs operates at a maximum frequency of 1 kHz.
	Frequency Generator	The frequency generator function generates a square wave signal on dedicated output channels with a fixed duty cycle (50%). The Frequency Generator function connected to regular transistor outputs operates at a maximum frequency of 1 kHz.

As of the release of EcoStruxure Machine Expert, any regular I/O not already in use can be configured for use by any of the expert function types, in the same way as fast I/Os.

NOTE:

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

For more details, refer to Embedded Functions Configuration.

Maximum Number of Expert Functions

The maximum number of expert functions that can be configured depends on:

1. The logic controller reference.
2. The expert function types and number of optional functions *(see page 117)* configured. Refer to Embedded Expert I/O Assignment *(see page 17)*.
3. The number of I/Os that are available.

Maximum number of expert functions by logic controller reference:

Expert Function Type		24 I/O References (TM241•24•)	40 I/O References (TM241•40•)
Total number of HSC functions		14	16
HSC	Simple	14	16
	Main Single Phase	4	
	Main Dual Phase		
	Frequency Meter(1)		
	Period Meter		
PTO			
PWM			
FreqGen			
(1) When the maximum number is configured, only 12 additional HSC Simple functions can be added.			

The maximum number of expert functions possible may be further limited by the number of I/Os used by each expert function.

Example configurations:

- 4 PTO(2) + 14 HSC Simple on 24 I/O controller references
- 4 FreqGen(2) + 16 HSC Simple on 40 I/O controller references
- 4 HSC Main Single Phase + 10 HSC Simple on 24 I/O controller references
- 4 HSC Main Dual Phase + 8 HSC Simple on 40 I/O controller references
- 2 PTO(2) + 2 HSC Main Single Phase + 14 HSC Simple on 40 I/O controller references

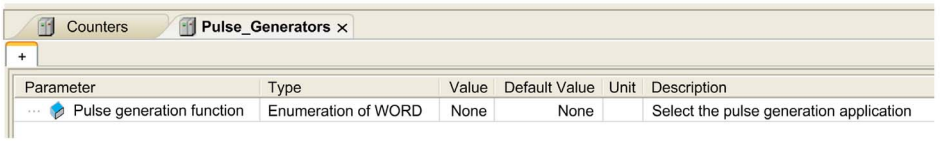
(2) With no optional I/O configured

The performance of the expert function is limited by the I/Os used:

- HSC with fast inputs: 100 kHz/200 kHz
- HSC with regular inputs: 1 kHz

Configuring an Expert Function

To configure an expert function, proceed as follows:

Step	Description
1	Double-click the **Counters** or **Pulse_Generators** node in the **Devices Tree**. **Result:** The **Counters** or **Pulse_Generators** configuration window appears: 
2	Double-click **None** in the **Value** column and choose the expert function type to assign. **Result:** The default configuration of the expert function appears when you click anywhere in the configuration window.
3	Configure the expert function parameters, as described in the following chapters.
4	To configure an additional expert function, click the **+** tab. **NOTE:** If the maximum number of expert functions is already configured, a message appears at the bottom of the configuration window informing you that you can now add only HSC Simple functions.

Regular I/O Configured as Expert Function

When regular I/Os are configured as expert functions, note the following:

- Inputs can be read through memory variables.
- An input cannot be configured as an expert function if it has already been configured as a Run/Stop input.
- An output cannot be configured in an expert function if it has already been configured as an alarm.
- Short-Circuit management applies on the outputs. Status of outputs are available.
- The I/O that are not used by expert functions can be used as any other regular I/O.
- When inputs are used in expert functions (Latch, HSC,…), integrator filter is replaced by anti-bounce filter. Filter value is configured in the configuration screen.

Embedded Expert I/O Assignment

I/O Assignment

The following regular or fast I/Os can be configured for use by expert functions:

	24 I/O References		40 I/O References	
	TM241•24T, TM241•24U	TM241•24R	TM241•40T, TM241•40U	TM241•40R
Inputs	8 fast inputs (I0...I7) 6 regular inputs (I8...I13)		8 fast inputs (I0...I7) 8 regular inputs (I8...I15)	
Outputs	4 fast outputs (Q0...Q3) 4 regular outputs (Q4...Q7)	4 fast outputs (Q0...Q3)	4 fast outputs (Q0...Q3) 4 regular outputs (Q4...Q7)	4 fast outputs (Q0...Q3)

When an I/O has been assigned to an expert function, it is no longer available for selection with other expert functions.

NOTE: All I/Os are by default disabled in the configuration window.

The following table shows the I/Os that can be configured for expert functions:

Expert Function	Name	Input (Fast or Regular)	Output (Fast or Regular)
HSC Simple	Input	M	
HSC Main	Input A	M	
	Input B/EN	C	
	SYNC	C	
	CAP	C	
	Reflex 0		C
	Reflex 1		C
Frequency Meter/Period Meter	Input A	M	
	EN	C	
PWM/FreqGen	Output A		M
	SYNC	C	
	EN	C	
M Mandatory **C** Optionally configurable			

Expert Function	Name	Input (Fast or Regular)	Output (Fast or Regular)
PTO	Output A/CW/Pulse		M
	Output B/CCW/Dir		C
	REF (Origin)	C	
	INDEX (Proximity)	C	
	PROBE	C	
M Mandatory **C** Optionally configurable			

Using Regular I/O with Expert Functions

Expert function I/O within regular I/O:

- Inputs can be read through standard memory variables even if configured as expert functions.
- All I/Os that are not used by expert functions can be used as regular I/Os.
- An I/O can only be used by one expert function; once configured, the I/O is no longer available for other expert functions.
- If no more fast I/Os are available, a regular I/O can be configured instead. In this case, however, the maximum frequency of the expert function is limited to 1 kHz.
- You cannot configure an input in an expert function and use it as a Run/Stop, Event, or Latch input at a same time.
- An output cannot be configured in an expert function if it has already been configured as an alarm.
- Short-circuit management still applies on all outputs. Status of outputs are available. For more information, refer to Output Management *(see Modicon M241 Logic Controller, Hardware Guide)*.
- When inputs are used in expert functions (PTO, HSC,…), the integrator filter is replaced by an anti-bounce filter *(see page 142)*. The filter value is configured in the configuration window.

For more details, refer to Embedded Functions Configuration *(see Modicon M241 Logic Controller, Programming Guide)*.

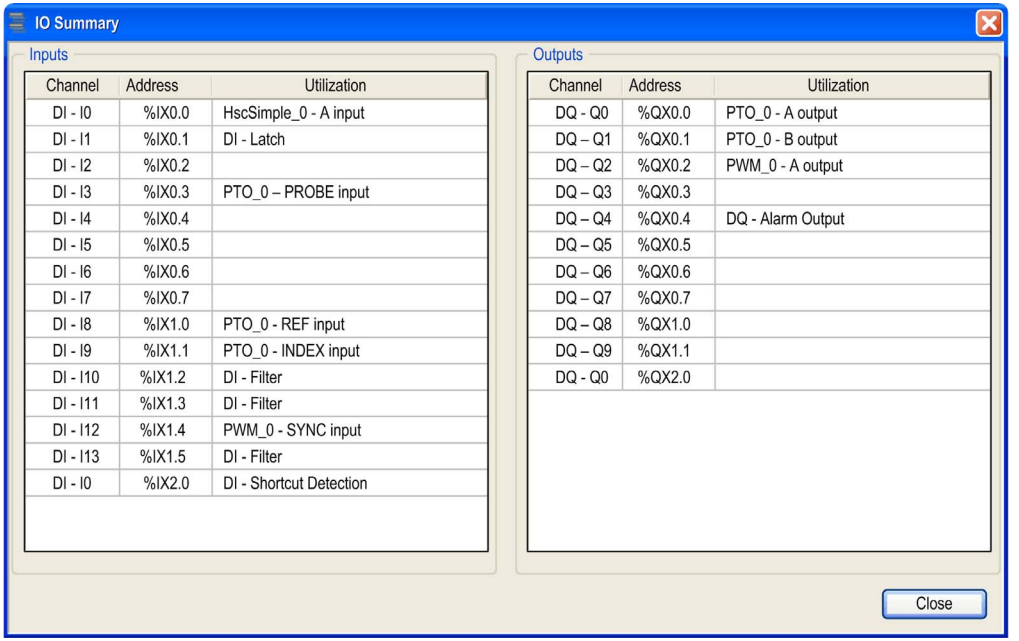
I/O Summary

The **IO Summary** window displays the I/Os used by the expert functions.

To display the **IO Summary** window:

Step	Action
1	In the **Devices tree** tab, right-click the **MyController** node and choose **IO Summary**.

Example of IO Summary window:



Chapter 2
High Speed Counter Types

Overview

This chapter provides an overview of the different types of HSC.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Choosing Your Counter	22
Simple Type Overview	25
Main Type Overview	26
Frequency Meter Type Overview	27
Period Meter Type Overview	28

Choosing Your Counter

Overview

Start the HSC configuration by choosing a counter type according to the type of sensor you are using and the application need.

In the **Counters** editor, select a **Counting function** from the list that offers the following types of counters (for more information, refer to the Counter Function *(see Modicon M241 Logic Controller, Programming Guide)*):

- **HSC Simple**
- **HSC Main Single Phase**
- **HSC Main Dual Phase**
- **Frequency Meter**
- **Period Meter**

The **Frequency Meter** type and the **Period Meter** type are both based on an **HSC Main** type.

For each counter defined in the **Counters** editor, a default **Instance name** is assigned by EcoStruxure Machine Expert. This default **Instance name** is editable. You must use exactly the same instance name as an input to the function blocks dealing with the counter.

Type and Mode Matrix

This table presents the different types and modes available:

Type Mode	HSC Simple	HSC Main Single Phase	HSC Main Dual Phase	Frequency Meter	Period Meter
One-shot	X	X	–	–	–
Modulo-loop	X	X	X	–	–
Event Counting	–	X	–	–	–
Free-large	–	–	X	–	–
Edge to Edge	–	–	–	–	X
Edge to Opposite	–	–	–	–	X

HSC Simple

This table presents an overview of the specifications available in **HSC Simple** type according to the mode requested:

Feature	Function	
	One-shot Mode	**Modulo-loop Mode**
Counting mode	Count down	Count up
Enable with an HSC physical input	No	No

Feature	Function	
	One-shot Mode	Modulo-loop Mode
Synchronization / preset with an HSC physical input	No	No
Comparison function	No	No
Capture function	No	No

HSC Main Single Phase

This table presents an overview of the specifications available in **HSC Main Single Phase** type according to the mode requested:

Feature	Function		
	One-shot Mode	Modulo-loop Mode	Event Counting Mode
Counting mode	Count down	Count up	Pulse counting during given time base (10 ms, 100 ms, or 1000 ms)
Enable with an HSC physical input	Yes	Yes	No
Synchronization / preset with an HSC physical input	Yes	Yes	Yes
Comparison function	Yes, 4 thresholds, 2 outputs, and 4 events	Yes, 4 thresholds, 2 outputs, and 4 events	No
Capture function	Yes, 1 capture register	Yes, 1 capture register	No

HSC Main Dual Phase

This table presents an overview of the specifications available in **HSC Main Dual Phase** type according to the mode requested:

Feature	Function	
	Modulo-Loop Mode	Free-Large Mode
Counting mode	Count up / down Pulse / direction Quadrature	Count up / down Pulse / direction Quadrature
Enable with an HSC physical input	No	No
Synchronization / preset with an HSC physical input	Yes	Yes
Comparison function	Yes, 4 thresholds, 2 outputs, and 4 events	Yes, 4 thresholds, 2 outputs, and 4 events
Capture function	Yes, 1 capture register	Yes, 1 capture register

Frequency Meter

This table presents an overview of the specifications available in **Frequency Meter** type:

Feature	Function
Counting mode	Pulse frequency in Hz with updated value available every time base value (10 ms, 100 ms, or 1000 ms)
Enable with an HSC physical input	Yes
Synchronization / preset with an HSC physical input	No
Comparison function	No
Capture function	No

Period Meter

This table presents an overview of the specifications available in **Period Meter** type according to the mode requested:

Feature	Function
Counting modes	Edge to edge: Measure the time between two events Edge to opposite: Measure the duration of an event
Enable with an HSC physical input	Yes
Synchronization / preset with an HSC physical input	No
Comparison function	No
Capture function	No
Resolution	Duration counting with configurable resolution (0.1 µs, 1 µs, 100 µs, or 1000 µs)
Timeout	0...858993459, calculated using resolution units 0 means no timeout

Simple Type Overview

Overview

The **Simple** type is a single input counter.

Any operation on the counter (enable, sync) and any action triggered (when count value is reached) is executed in the context of a task.

With the **Simple** type, you cannot trigger an event or a reflex output.

Simple Type Modes

The **Simple** type supports 2 configurable counting modes on single-phase pulses:

One-shot *(see page 33)*. In this mode, the counter current value register decrements (from a user-defined value) for each pulse applied to A input, until the counter reaches 0.

Modulo-loop *(see page 53)*. In this mode, the counter repeatedly counts from 0 to a user-defined modulo value then returns to 0 and restarts counting.

Performance

The maximum frequency admissible on a fast input is 100 kHz if the bounce filter value is 0.005 ms (default value for configuration). If the bounce filter value is 0.002 ms, the maximum frequency is 200 kHz.

The maximum frequency admissible on a regular input is 1 kHz if the bounce filter value is 0.5 ms. If the bounce filter value is 1 ms, the maximum frequency is 500 Hz.

For more information about the bounce filter, refer to Dedicated Features *(see page 142)*.

Main Type Overview

Overview

The **Main** type is a counter that uses up to 4 fast or regular inputs and 2 reflex outputs. The M241 Logic Controller can have up to 4 **Main** type High Speed Counters.

Main Type Modes

The **Main** type supports the following counting modes on single phase (1 input) or dual-phase (2 inputs) pulses:

One-shot *(see page 39)*: In this mode, the counter current value register decrements (from a user-defined value) for each pulse applied to the A input until the counter reaches 0.

Modulo-loop *(see page 59)*: In this mode, the counter repeatedly counts up from 0 to a user-defined modulo value, 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.

Free-large *(see page 75)*: In this mode, the counter behaves like a high range up and down counter.

Event Counting *(see page 87)*: In this mode, the counter accumulates the number of events that are received during a user-configured time base.

Optional Features

Optional features can be configured depending on the selected mode:

- Hardware inputs to operate the counter (enable, preset) or capture the current counting value
- Up to 4 thresholds, the values of which can be compared.
- Up to 4 events (1 for each threshold) can be associated with external tasks
- Up to 2 reflex outputs

Performance

The maximum frequency admissible on an **Expert I/O** interface is 100 kHz if the bounce filter value is 0.005 ms (default value for configuration). If the bounce filter value is 0.002 ms, the maximum frequency is 200 kHz.

If the expert function is configured with a regular I/O, the minimum period admissible is 0.4 ms.

Frequency Meter Type Overview

Overview

The **Frequency Meter** type is a counter that uses up to 2 fast or regular inputs. The M241 Logic Controller can have up to 4 **Frequency Meter** type High Speed Counters.

Frequency Meter Type Mode

The **Frequency meter** *(see page 99)* counter measures the frequency of events. Frequency is the number of events per second (Hz).

Performance

The maximum frequency admissible on a fast input is 100 kHz if the bounce filter value is 0.005 ms (default value for configuration). If the bounce filter value is 0.002 ms, the maximum frequency is 200 kHz.

The maximum frequency admissible on a regular input is 1 kHz if the bounce filter value is 0.5 ms. If the bounce filter value is 1 ms, the maximum frequency is 500 Hz.

Period Meter Type Overview

Overview

The **Period Meter** type is a counter that uses up to 2 fast or regular inputs.

The M241 Logic Controller can have up to 4 **Period Meter** type High Speed Counters.

Period Meter Type Mode

Use the **Period meter** counting mode to:

- Determine the duration of an event
- Measure the time between 2 events
- Set and measure the execution time for a process

Performance

The minimum period admissible on a fast input is 0.005 ms.

If the expert function is configured with a regular I/O, the minimum period admissible is 0.4 ms.

For more information about the bounce filter, refer to Dedicated Features *(see page 142)*.

Part II
One-shot Mode

Overview

This part describes the use of a HSC in **One-shot** Mode.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	**One-shot** Mode Principle	31
4	**One-shot** with a **Simple** Type	33
5	**One-shot** with a **Main** Type	39

Chapter 3
One-shot Mode Principle

One-shot Mode Principle Description

Overview

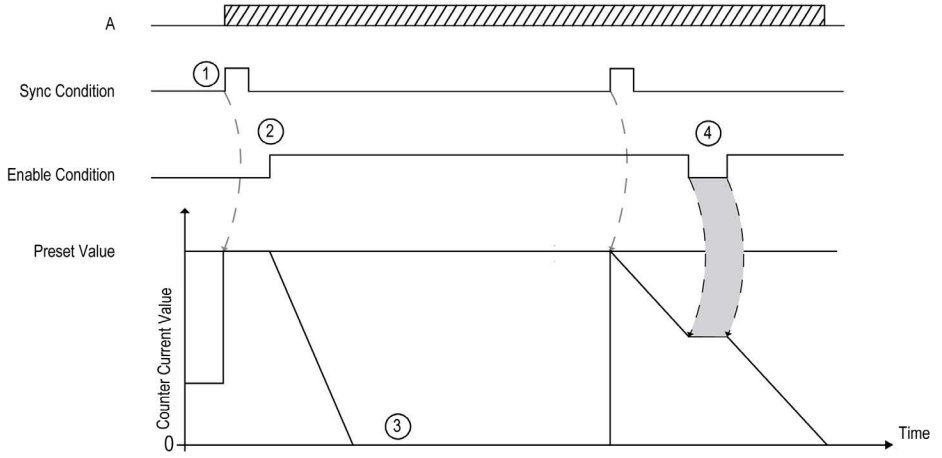
The counter is activated by a synchronization edge, and the preset value is loaded.

When counting is enabled, each pulse applied to the input decrements the current value. The counter stops when its current value reaches 0.

The counter value remains at 0 even if new pulses are applied to the input.

A new synchronization is needed to activate the counter again.

Principle Diagram



This table explains the stages from the preceding graphic:

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

NOTE: Enable and Sync conditions depends on configuration. These are described in the Enable *(see page 137)* and Preset *(see page 134)* function.

Chapter 4
One-shot with a Simple Type

Overview

This chapter describes how to implement a High Speed Counter in **One-shot** mode using a **Simple** type.

What Is in This Chapter?

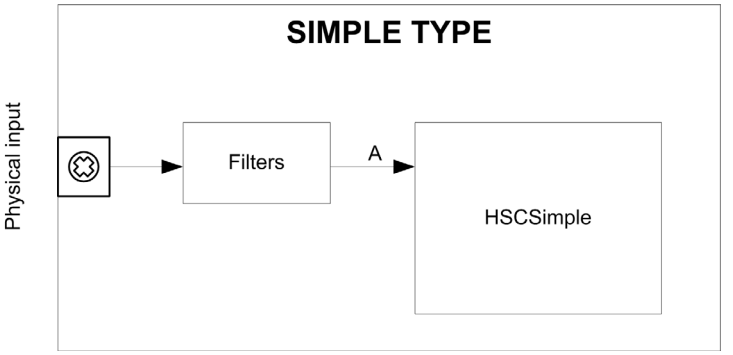
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	34
Configuration of the Simple Type in One-Shot Mode	35
Programming the **Simple** Type	36
Adjusting Parameters	38

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Simple** type in **One-shot** mode:



A is the counting input of the High Speed Counter. **Simple** type counting for **One-shot** mode always counts down.

Configuration of the Simple Type in One-Shot Mode

Procedure

Follow this procedure to configure a **Simple** type in **One-shot** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: The **Counters** editor tab opens for HSC configuration.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Simple**, then click anywhere in the configuration area. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, modify the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **One-shot**.
5	In **Counting inputs → A input → Location** select the fast or regular input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
7	Enter the value of the **Range → Preset** parameter to set the counting initial value.
8	With a expansion module, you can specify the name of an external event. When this event is triggered in a task, the counter is stopped. Set the value of **Stop → Stop event** to **Yes**, then modify the **Stop Event Name** to the name of the external event.

Programming the Simple Type

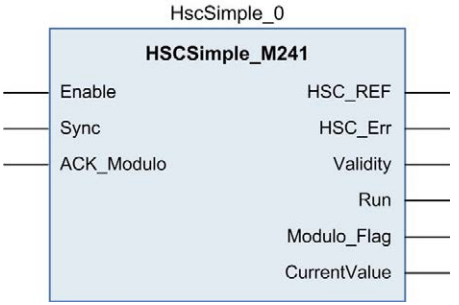
Overview

A **Simple** type counter is always managed by an HSCSimple_M241 *(see page 170)* function block.

NOTE: At build time, an error is detected if the `HSCSimple_M241` function block is used to manage a different HSC type.

Adding an HSCSimple Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCSimple_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Simple** type instance name (defined in configuration) or select the function block instance by clicking:  Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **One-shot** mode.

This table describes the input variables:

Input	Type	Comment
Sync	BOOL	On rising edge, presets and starts the counter
ACK_Modulo	BOOL	Not used in one-shot mode.

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC. To be used as input of **Administrative** function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 158)* function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	Set to 1 when the counter is running. Switches to 0 when CurrentValue reaches 0. A synchronization is needed to restart the counter.
Modulo_Flag	BOOL	Not used in one-shot mode.
CurrentValue	DWORD	Current count value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 162)* or EXPERTSetParam *(see page 164)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M241 Logic Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 151)* that can be read or modified while the program is running:

Parameter	Description
EXPERT_PRESET	to get or set the Preset value of an HSC

Chapter 5
One-shot with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **One-shot** mode using a **Main** type.

What Is in This Chapter?

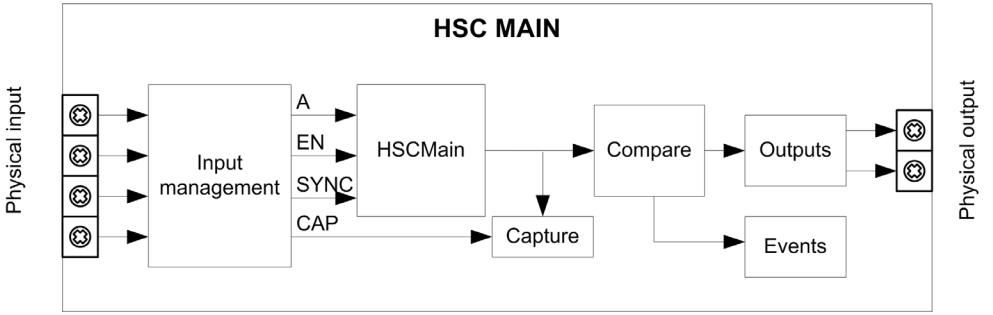
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	40
Configuration of the Main Type Single Phase in One-Shot Mode	41
Programming the **Main** Type	42
Adjusting Parameters	45

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **One-shot** mode:



A is the counting input of the counter.

EN is the enable input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Function

In addition to the **One-shot** mode, the **Main** type can provide the following functions:

- Preset function *(see page 134)*
- Enable function *(see page 137)*
- Capture function *(see page 129)*
- Comparison function *(see page 119)*

Configuration of the Main Type Single Phase in One-Shot Mode

Procedure

Follow this procedure to configure a **Main** type single phase in **One-shot** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: The **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Single Phase** and click anywhere in the configuration window. **Result**: The configuration parameters appear in the **Counters** tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE:** **Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **One-shot**.
5	In **Counting Inputs → A input → Location** select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
7	Enter the value of the **Range → Preset** parameter to set the initial counting value of the Preset function *(see page 134)*.
8	Optionally, you can enable these functions: • Preset function *(see page 134)* • Enable function *(see page 137)* • Capture function *(see page 129)* • Comparison function *(see page 119)*
9	Optionally, set the value of the **Events → Stop Event** parameter to **Yes** to enable the External Event function *(see page 126)*. **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.

Programming the Main Type

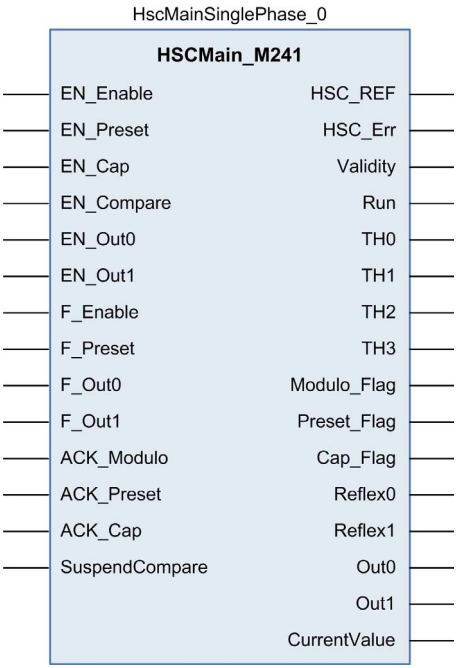
Overview

The **Main** type is always managed by an `HSCMain_M241` function block.

NOTE: At build time, an error is detected if the `HSCMain_M241` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCMain_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **One-shot** mode.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes enabling of the counter with the Enable input *(see page 137)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter Preset via the Sync input *(see page 134)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input.
EN_Compare	BOOL	TRUE = enables the comparator operation *(see page 119)* (using Thresholds 0, 1, 2, 3): ● basic comparison (TH0, TH1, TH2, TH3 output bits) ● reflex (Reflex0, Reflex1 output bits) ● events (to trigger external tasks on threshold crossing) **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.
EN_Out0	BOOL	TRUE = enables physical output Out_R0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Out_R1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, presets and starts the counter.
F_Out0	BOOL	TRUE = forces Out_R0 to 1 (if Reflex0 is configured in HSC Embedded Function. Takes priority over EN_Out0.
F_Out1	BOOL	TRUE = forces Out_R1 to 1 (if Reflex1 is configured in HSC Embedded Function. Takes priority over EN_Out1.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: ● TH0, TH1, TH2, TH3 , Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. ● Hardware Outputs 0, 1 maintain their last value. ● Events are masked. **NOTE:** EN_Compare, EN_ReflexO, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC. To be used as input of **Administrative** function blocks.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	TRUE = counter is running. Set to False when CurrentValue reaches 0.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 *(see page 119)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 *(see page 119)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 *(see page 119)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 *(see page 119)*.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 134)*.
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register. This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0 *(see page 120)*. Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1 *(see page 120)*. Only active when EN_Compare is set.
Out0	BOOL	State of physical output Out_R0 (if Reflex0 configured).
Out1	BOOL	State of physical output Out_R1 (if Reflex1 configured).
CurrentValue	DINT	Current value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 162)* or EXPERTSetParam *(see page 164)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M241 Logic Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 151)* which can be read or modified while the program is running:

Parameter	Description
EXPERT_PRESET	to get or set the Preset value of an HSC
EXPERT_THRESHOLD0	to get or set the Threshold 0 value of an HSC
EXPERT_THRESHOLD1	to get or set the Threshold 1 value of an HSC
EXPERT_THRESHOLD2	to get or set the Threshold 2 value of an HSC
EXPERT_THRESHOLD3	to get or set the Threshold 3 value of an HSC
EXPERT_REFLEX0	to get or set output 0 reflex mode of an EXPERT function
EXPERT_REFLEX1	to get or set output 1 reflex mode of an EXPERT function

Part III
Modulo-loop Mode

Overview

This part describes the use of a HSC in **Modulo-loop** mode.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
6	**Modulo-loop** Principle	49
7	**Modulo-loop** with a **Simple** Type	53
8	**Modulo-loop** with a **Main** Type	59

Chapter 6
Modulo-loop Principle

Modulo-loop Mode Principle Description

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 Sync condition *(see page 134)*, the counter is activated and the current value is reset to 0.

When counting is enabled *(see page 137)*:

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

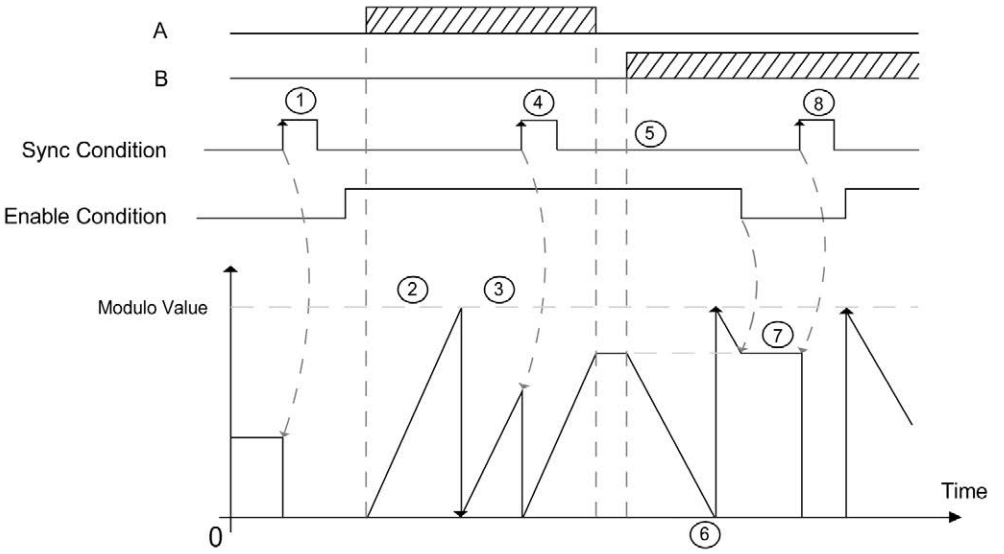
Decrementing direction: the counter decrements until it reaches 0. At the next pulse, the counter is set to the modulo value, a modulo flag is set to 1, and the counting continues.

Input Modes

This table shows the 8 types of input modes available:

Input Mode	Comment
A = Up, B = Down	default mode The counter increments on A and decrements on B.
A = Impulse, B = Direction	If there is a rising edge on A and B is true, then the counter decrements. If there is a rising edge on A and B is false, then the counter increments.
Normal Quadrature X1	A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction: ● X1: 1 count by Encoder cycle ● X2: 2 counts by Encoder cycle ● X4: 4 counts by Encoder cycle
Normal Quadrature X2	
Normal Quadrature X4	
Reverse Quadrature X1	
Reverse Quadrature X2	
Reverse Quadrature X4	

Up Down Principle Diagram

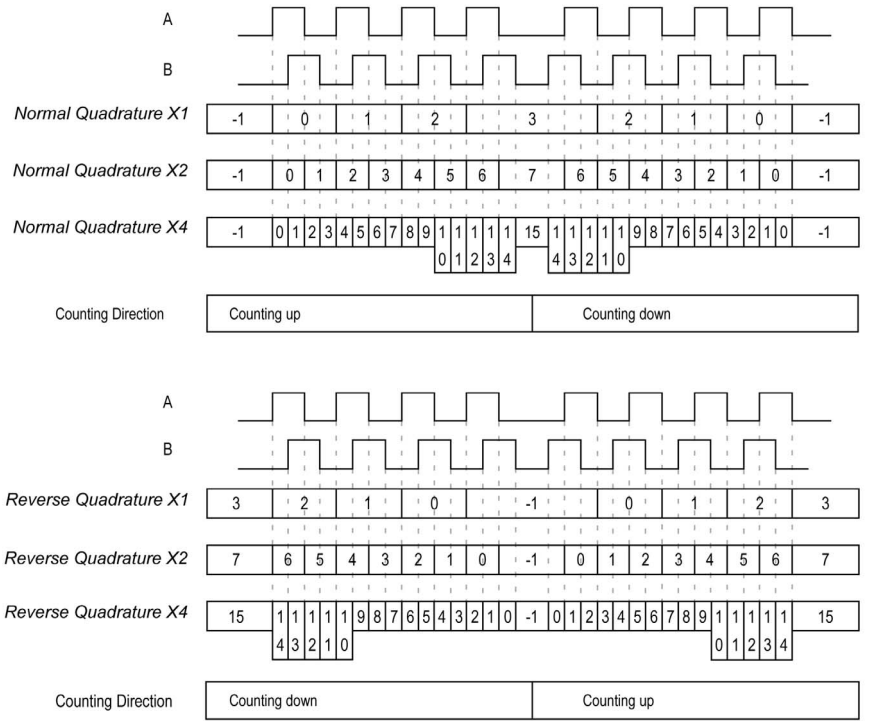


Stage	Action
1	On the rising edge of Sync condition, the current value is reset to 0 and the counter is activated.
2	When Enable condition = 1, each pulses on A increments the counter value.
3	When the counter reaches the (modulo-1) value, the counter loops to 0 at the next pulse and the counting continues.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

NOTE: Enable and Sync conditions depends on configuration. These are described in the Enable *(see page 137)* and Preset *(see page 134)* function.

Quadrature Principle Diagram

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



Chapter 7
Modulo-loop with a Simple Type

Overview

This chapter describes how to implement a High Speed Counter in **Modulo-loop** mode using a **Simple** type.

What Is in This Chapter?

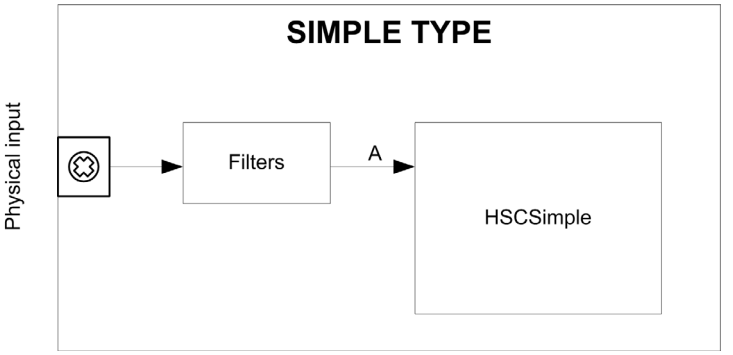
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	54
Configuration of the Simple Type in Modulo-Loop Mode	55
Programming the **Simple** Type	56
Adjusting Parameters	58

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Simple** type in **Modulo-loop** mode:



A **Simple** type counting for **Modulo-loop** mode only counts up.

Configuration of the Simple Type in Modulo-Loop Mode

Procedure

Follow this procedure to configure a **Simple** type in **Modulo-loop** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: The **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Simple**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE:** **Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
5	In **Counting Inputs → A input → Location** select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
7	Enter the value of the **Range → Modulo** parameter to set the counting modulo value.

Programming the Simple Type

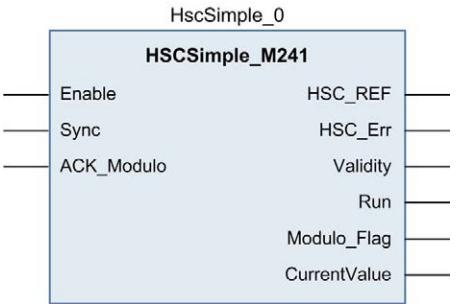
Overview

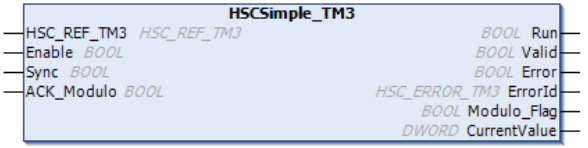
A **Simple** type is always managed by an HSCSimple_M241 *(see page 170)* function block.

NOTE: At build time, an error is detected if the `HSCSimple_M241` function block is used to manage a different HSC type.

Adding a HSCSimple Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCSimple_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Simple** type instance name (defined in configuration) or select the function block instance by clicking:  Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.





I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Modulo-loop** mode.

This table describes the input variables:

Input	Type	Comment
`Enable`	`BOOL`	`TRUE` = authorizes changes to the current counter value.
`Sync`	`BOOL`	On rising edge, resets and starts the counter.
`ACK_Modulo`	`BOOL`	On rising edge, resets `Modulo_Flag`.

This table describes the output variables:

Output	Type	Comment
`HSC_REF`	`EXPERT_REF` *(see page 153)*	Reference to the HSC. To be used as input of the **Administrative** function blocks.
`HSC_Err`	`BOOL`	`TRUE` = indicates that an error was detected. Use the `EXPERTGetDiag` *(see page 158)* function block to get more information about this detected error.
`Validity`	`BOOL`	`TRUE` = indicates that the output values on the function block are valid.
`Run`	`BOOL`	Not relevant
`Modulo_Flag`	`BOOL`	Set to `TRUE` when the counter rolls over the `Modulo` value.
`CurrentValue`	`DWORD`	Current value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 162)* or EXPERTSetParam *(see page 164)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M241 Logic Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 151)* that can be read or modified while the program is running:

Parameter	Description
EXPERT_MODULO	to get or set the modulo value of an HSC

Chapter 8
Modulo-loop with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Modulo-loop** mode using a **Main** type.

What Is in This Chapter?

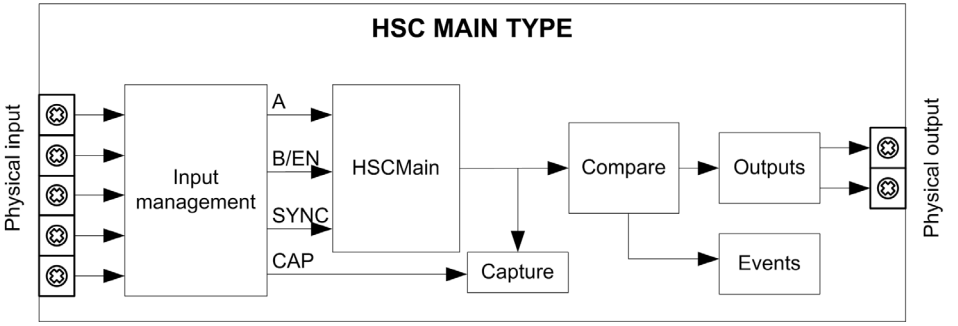
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	60
Configuration of the Main Type Single Phase in Modulo-Loop Mode	61
Configuration of the Main Type Dual Phase in Modulo-Loop Mode	62
Programming the **Main** Type	63
Adjusting Parameters	66

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Modulo-loop** mode:



A and B are the counting inputs of the counter.

EN not configurable when B input is used.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Functions

In addition to the **Modulo-loop** mode, the **Main** type can provide the following functions:

- Enable function *(see page 137)*
- Capture function *(see page 129)*
- Comparison function *(see page 119)*

NOTE: The Preset value is 0 and cannot be modified.

Configuration of the Main Type Single Phase in Modulo-Loop Mode

Procedure

Follow this procedure to configure a **Main** type single phase in **Modulo-loop** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Single Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
5	In **Counting Inputs → A input → Location** select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
7	Enter the value of the **Range → Modulo** parameter to set the counting modulo value.
8	Optionally, you can enable these control functions: ● Enable function *(see page 137)* ● Capture function *(see page 129)* ● Comparison function *(see page 119)*

Configuration of the Main Type Dual Phase in Modulo-Loop Mode

Procedure

Follow this procedure to configure a **Main** type dual phase in **Modulo-loop** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Dual Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
5	Set the value of the **General → Input mode** parameter to select the modulo loop input mode *(see page 49)*.
6	In **Counting Inputs → A input → Location** select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
7	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
8	In **Counting Inputs → B input → Location** select the regular or fast input to use as the B input.
9	Set the value of the **Counting inputs → B input → Bounce filter** parameter to reduce the bounce effect on the input..
10	Enter the value of the **Range → Modulo** parameter to set the counting modulo value.
11	Optionally, you can enable these control functions: ● Capture function *(see page 129)* ● Comparison function *(see page 119)*

Programming the Main Type

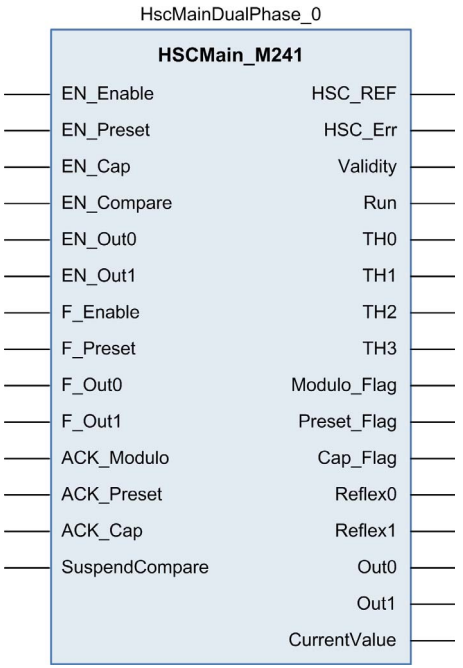
Overview

The **Main** type is always managed by an `HSCMain_M241` function block.

NOTE: At build time, an error is detected if the `HSCMain_M241` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCMain_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking: `???` `...` Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Modulo-loop** mode.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes the counter enable via the Enable input *(see page 137)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter Preset via the Sync input *(see page 134)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input.
EN_Compare	BOOL	TRUE = enables the comparison function *(see page 119)* using Threshold 0, 1, 2, 3: • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = enables physical output Out_R0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Out_R1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, resets, and starts the counter.
F_Out0	BOOL	TRUE = forces Out_R0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Out_R1 to 1 (if Reflex1 is configured).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: • TH0, TH1, TH2, TH3 , Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. • Physical Outputs 0, 1 maintain their last value. • Events are masked. **NOTE:** EN_Compare, EN_ReflexO, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC. To be used as input of **Administrative** function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 158)* function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	TRUE = counter is running. The Run bit switches to 0 when CurrentValue reaches 0. A synchronization is needed to restart the counter.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 *(see page 119)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 *(see page 119)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 *(see page 119)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 *(see page 119)*.
Modulo_Flag	BOOL	Set to 1 when the counter roll overs the modulo or 0.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 134)*.
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 130)*. This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0 *(see page 122)*. Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1 *(see page 122)*. Only active when EN_Compare is set.
Out0	BOOL	State of physical output Out_R0 (if Reflex0 is configured).
Out1	BOOL	State of physical output Out_R1 (if Reflex1 is configured).
CurrentValue	DINT	Current value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 162)* or EXPERTSetParam *(see page 162)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M241 Logic Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 151)* that can be read or modified while the program is running:

Parameter	Description
EXPERT_MODULO	to get or set the Modulo value of an HSC
EXPERT_THRESHOLD0	to get or set the Threshold 0 value of an HSC
EXPERT_THRESHOLD1	to get or set the Threshold 1 value of an HSC
EXPERT_THRESHOLD2	to get or set the Threshold 2 value of an HSC
EXPERT_THRESHOLD3	to get or set the Threshold 3 value of an HSC
EXPERT_REFLEX0	to get or set output 0 reflex mode of an EXPERT function
EXPERT_REFLEX1	to get or set output 1 reflex mode of an EXPERT function

Part IV
Free-large Mode

Overview

This part describes the use of an HSC in **Free-large** mode.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
9	Free-large Mode Principle	69
10	Free-large with a Main Type	75

Chapter 9
Free-large Mode Principle

Overview

This chapter describes the principle of the **Free-large** mode.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Free-large Mode Principle Description	70
Limits Management	73

Free-large Mode Principle Description

Overview

The **Free-large** mode can be used for axis monitoring or labeling in cases where the incoming position of each part has to be known.

Principle

In the **Free-large** mode, the module behaves like a standard up and down counter.

When counting is enabled *(see page 137)*, the counter counts as follows in:
Incrementing direction: the counter increments.
Decrementing direction: the counter decrements.

The counter is activated by a preset edge *(see page 136)* which loads the preset value.

The current counter is stored in the capture register by using the Capture *(see page 129)* function.

If the counter reaches the counting limits, the counter will react according to the Limits Management *(see page 73)* configuration.

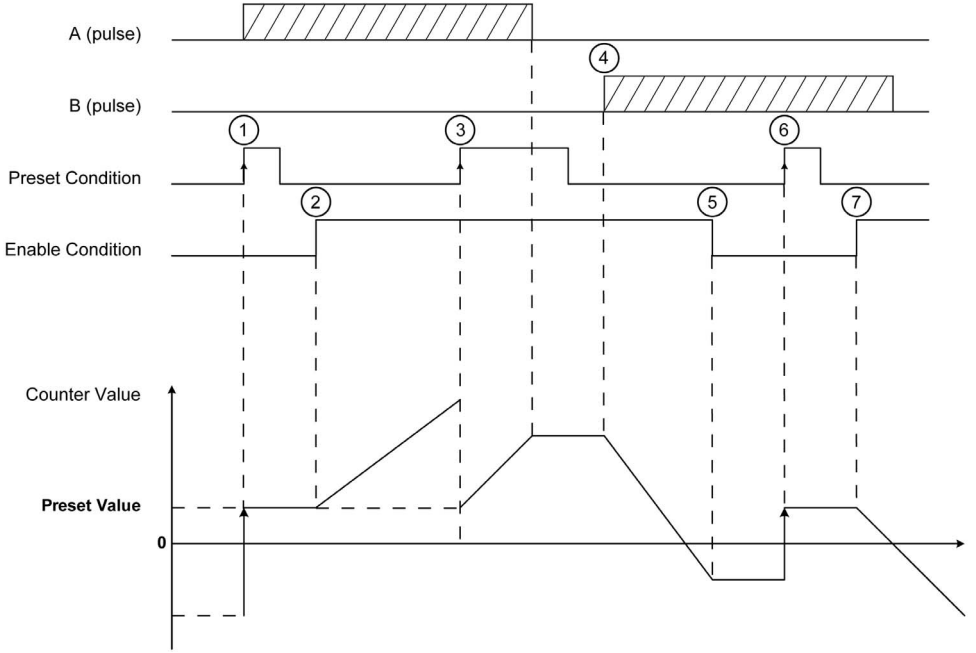
Input Modes

This table shows the 8 types of input modes available:

Input Mode	Comment
A = Up, B = Down	default mode The counter increments on A and decrements on B.
A = Pulse, B = Direction	If there is a rising edge on A and B is true, then the counter decrements. If there is a rising edge on A and B is false, then the counter increments.
Normal Quadrature X1	A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction: ● X1: 1 count for each Encoder cycle ● X2: 2 counts for each Encoder cycle ● X4: 4 counts for each Encoder cycle
Normal Quadrature X2	
Normal Quadrature X4	
Reverse Quadrature X1	
Reverse Quadrature X2	
Reverse Quadrature X4	

Up Down Principle Diagram

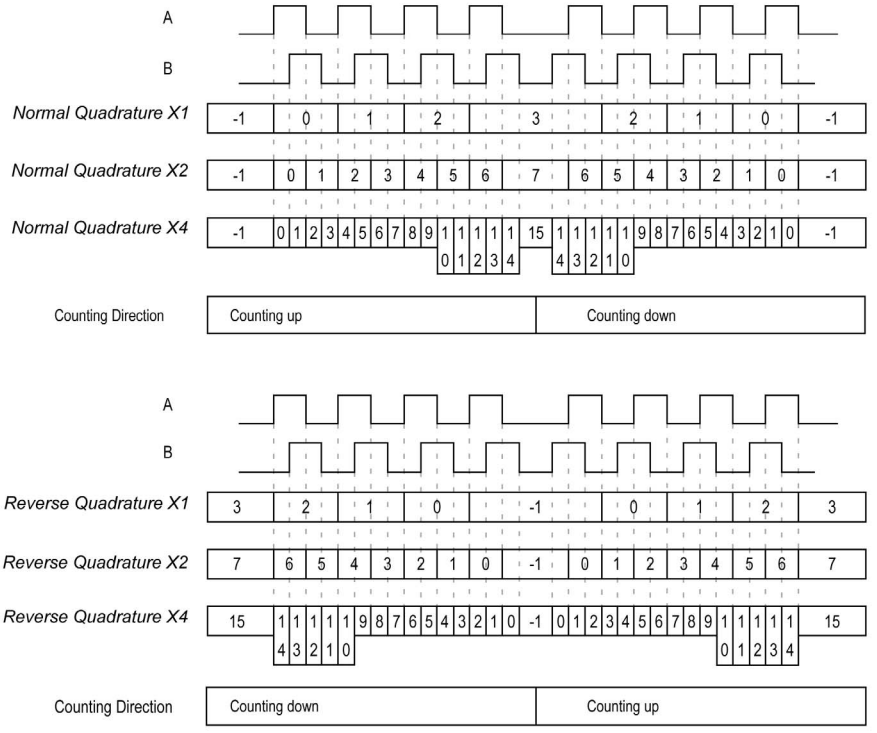
The figures shows the **A = Up, B = Down** mode:



Stage	Action
1	On the rising edge of Preset condition, the current value is set to the preset value and the counter is activated.
2	When Enable condition = 1, each pulse on A increment the counter value.
3	On the rising edge of Preset condition, the current value is set to the preset value.
4	When Enable condition = 1, each pulse on B decrements the counter value.
5	When Enable condition = 0, the pulses on A or B are ignored.
6	On the rising edge of Preset condition, the current value is set to the preset value.
7	When Enable condition = 1, the pulses on B decrements the counter value.

Quadrature Principle Diagram

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



Limits Management

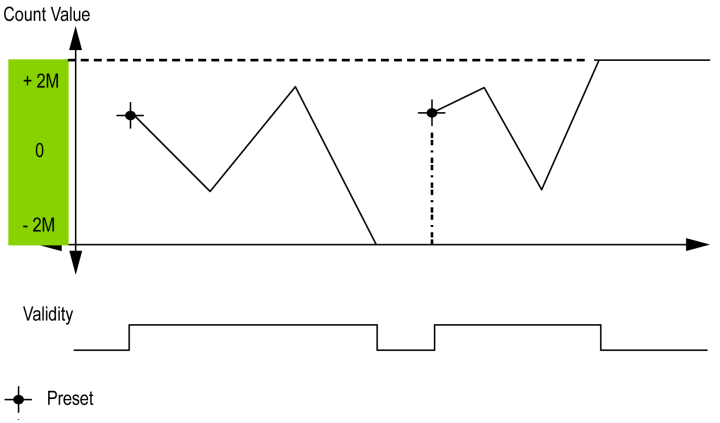
Overview

When the counter limit is reached, the counter can have 2 behaviors depending on configuration:

- Lock on limits
- Rollover

Lock on Limits

In the case of an overflow or underflow counter, the current counter value is maintained at the limit value, the validity bit goes to 0, and the `Error` bit indicates that this detected error until the counter is preset again.



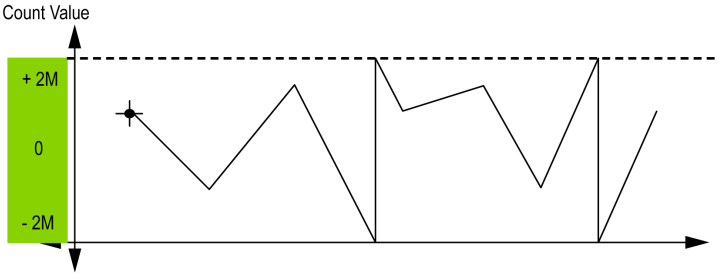
2M value is given as:

- +2M = 2 (exp 31) -1
- -2M = -2 (exp 31)

Rollover

In the case of overflow or underflow of the counter, the current counter value goes automatically to the opposite limit value.

`Modulo_Flag` output is set to



1.

Chapter 10
Free-large with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Free-large** mode using a **Main** type.

What Is in This Chapter?

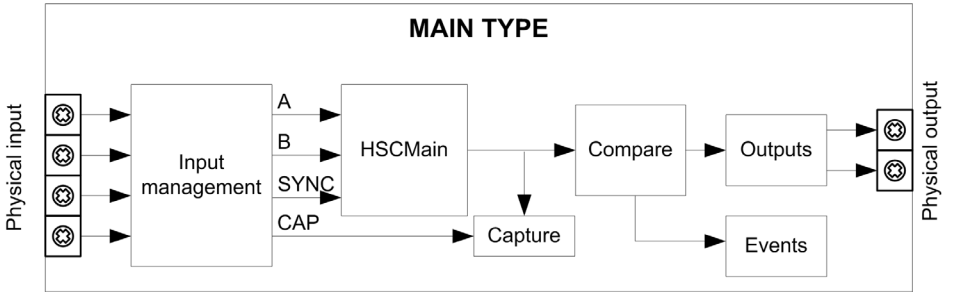
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	76
Configuration of the Main Type Dual Phase in Free-Large Mode	77
Programming the **Main** Type	78
Adjusting Parameters	81

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Free-large** mode:



A and B are the counting inputs of the counter.

EN is the enable input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Function

In addition to the **Free-large** mode, the **Main** type can provide the following functions:

- Preset function *(see page 134)*
- Enable function *(see page 137)*
- Capture function *(see page 129)*
- Comparison function *(see page 119)*

Configuration of the Main Type Dual Phase in Free-Large Mode

Procedure

Follow this procedure to configure a **Main** type dual phase in **Free-large** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Dual Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Free-large**.
5	Set the value of the **General → Input mode** parameter to select the input mode *(see page 70)*.
6	In **Counting Inputs → A input → Location** select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
7	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
8	In **Counting Inputs → B input → Location** select the regular or fast input to use as the B input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
9	Set the value of the **Counting inputs → B input → Bounce filter** parameter.
10	Enter the value of the **Range → Preset** parameter to set the counting initial value.
11	Enter the value of the **Range → Limits** for limits management *(see page 73)*.
12	Optionally, you can enable these functions: ● Preset function *(see page 134)* ● Enable function *(see page 137)* ● Capture function *(see page 129)* ● Comparison function *(see page 119)*

Programming the Main Type

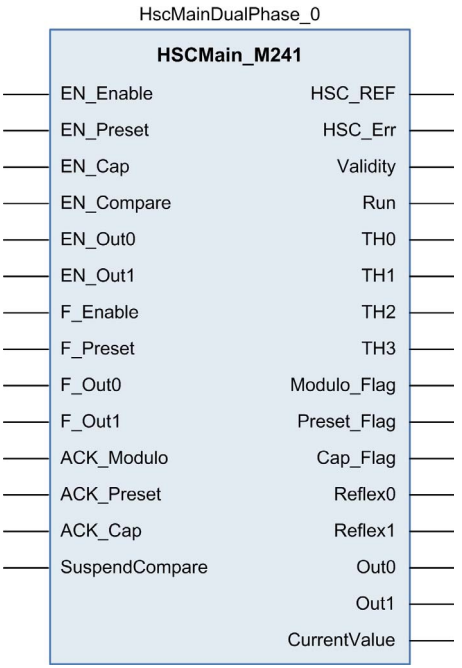
Overview

The **Main** type is always managed by an `HSCMain_M241` function block.

NOTE: At build time, an error is detected if the `HSCMain_M241` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCMain_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Free-large** mode.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes the counter enable via the Enable input *(see page 137)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter Preset via the Sync input *(see page 134)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input *(see page 132)*.
EN_Compare	BOOL	TRUE = enables the comparison operation *(see page 119)* (using Thresholds 0, 1, 2, 3): • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = enables physical output Out_R0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Out_R1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, presets and starts the counter.
F_Out0	BOOL	TRUE = forces Out_R0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Out_R1 to 1 (if Reflex1 is configured).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: • TH0, TH1, TH2, TH3 , Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. • Physical outputs 0, 1 maintain their last value. • Events are masked. **NOTE:** EN_Compare, EN_ReflexO, EN_Reflex1,F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC. To be used as input of **Administrative** function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 158)* function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	Not used.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 *(see page 119)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 *(see page 119)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 *(see page 119)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 *(see page 119)*.
Modulo_Flag	BOOL	Set to 1 when the counter rolls over its limits.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 134)*
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 129)*. This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0. Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1. Only active when EN_Compare is set.
Out0	BOOL	State of physical outputs Out_R0 (if Reflex0 is configured in HSC Embedded Function, otherwise FALSE if not configured).
Out1	BOOL	State of physical outputs Out_R1 (if Reflex1 is configured in HSC Embedded Function, otherwise FALSE if not configured).

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 162)* or EXPERTSetParam *(see page 164)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M241 Logic Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 151)* enumeration which can be read or modified while the program is running:

Parameter	Description
EXPERT_PRESET	to get or set the Preset value of the HSC
EXPERT_THRESHOLD0	to get or set the Threshold 0 value of an HSC
EXPERT_THRESHOLD1	to get or set the Threshold 1 value of an HSC
EXPERT_THRESHOLD2	to get or set the Threshold 2 value of an HSC
EXPERT_THRESHOLD3	to get or set the Threshold 3 value of an HSC
EXPERT_REFLEX0	to get or set output 0 reflex mode of an expert function
EXPERT_REFLEX1	to get or set output 0 reflex mode of an expert function

Part V
Event Counting Mode

Overview

This part describes the use of an HSC in **Event Counting** mode.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
11	**Event Counting** Principle	85
12	**Event Counting** with a **Main** Type	87

Chapter 11
Event Counting Principle

Event Counting Mode Principle Description

Overview

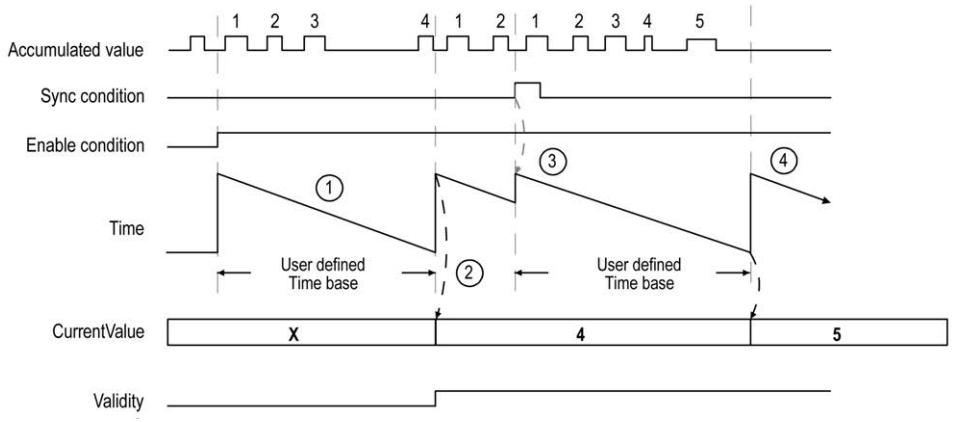
The **Event Counting** mode allows you to count the number of events that occur during a given period of time.

Principle

The counter assesses the number of pulses applied to the input for a predefined period of time. At the end of each period, the counting register is updated with the number of events received.

Synchronization can be used over the time period. This restarts the counting event for a new predefined time period. The counting restarts at the edge Sync condition *(see page 134)*.

Principle Diagram



Stage	Action
1	When Enable condition = 1, the counter accumulates the number of events (pulses) on the physical input during a predefined period of time. If Validity = 0, the current value is not relevant.
2	Once the first period of time has elapsed, the counter value is set to the number of events counted over the period and Validity is set to 1. The counting restarts for a new period of time.
3	On the rising edge of the Sync condition: ● the accumulated value is reset to 0 ● the current value is not updated ● the counting restarts for a new period of time
4	Once the period of time has elapsed, the counter value is set to the number of events counted over the period. The counting restarts for a new period of time.

NOTE:
On the **Main** type, when the Enable condition is:
- Set to 0: the current counting is aborted and `CurrentValue` is maintained at the previous valid value.
- Set to 1: the accumulated value is reset to 0, the `CurrentValue` remains unchanged, and the counting restarts for a new period of time.

Chapter 12
Event Counting with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Event Counting** mode using a **Main** type.

What Is in This Chapter?

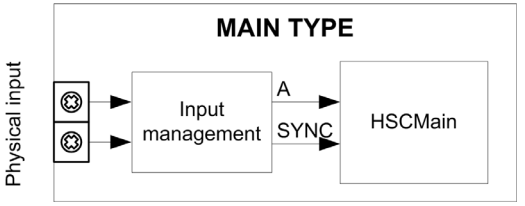
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	88
Configuration of the Main Type Single Phase in Event Counting Mode	89
Programming the **Main** Type	90
Adjusting Parameters	93

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Event Counting** mode.



A is the counting input of the counter.

SYNC is the synchronization input of the counter.

Optional Function

In addition to the **Event Counting** mode, the **Main** type provides the Preset function *(see page 134)*.

Configuration of the Main Type Single Phase in Event Counting Mode

Procedure

Follow this procedure to configure a **Main** type single phase in **Event Counting** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Single Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE:** **Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Event Counting**.
5	In **Counting Inputs → A input → Location** select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
7	Set the value of the **Range → Time base** parameter to determine the period during which the number of events is counted. Select the measurement of the update cycle time: ● 0.1 s ● 1 s (default value) ● 10 s ● 60 s
8	Optionally, set the value of the **Control inputs → SYNC input → Location** parameter to enable the Preset Function *(see page 134)*.

Programming the Main Type

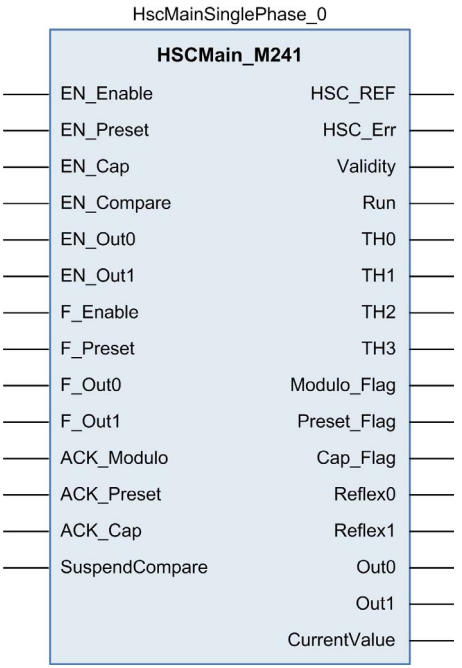
Overview

The **Main** type is always managed by an `HSCMain_M241` function block.

NOTE: At build time, an error is detected if the `HSCMain_M241` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCMain_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.



I/O Variables Usage

These tables describe how the different pins of the function block are used in the mode **Event**.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	Not used.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter Preset via the Sync input *(see page 134)*.
EN_Cap	BOOL	Not used.
EN_Compare	BOOL	Not used.
EN_Out0	BOOL	Not used.
EN_Out1	BOOL	Not used.
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, restarts the internal timer relative to the time base.
F_Out0	BOOL	Not used.
F_Out1	BOOL	Not used.
ACK_Modulo	BOOL	Not used.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	Not used.
SuspendCompare	BOOL	Not used.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the **Administrative** function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. EXPERTGetDiag *(see page 158)* function block may be used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	Counter is running
TH0	BOOL	Not used.
TH1	BOOL	Not used.
TH2	BOOL	Not used.
TH3	BOOL	Not used.
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 134)*.
Cap_Flag	BOOL	Not used.
Reflex0	BOOL	Not used.
Reflex1	BOOL	Not used.
Out0	BOOL	Not used.
Out1	BOOL	Not used.
CurrentValue	DINT	Current value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 162)* or EXPERTSetParam *(see page 164)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M241 Logic Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 151)* which can be read or modified while the program is running:

Parameter	Type	Description
EXPERT_TIMEBASE	EXPERT_HSCMAIN_TIMEBASE_TYPE For more information, refer to Type for HSC *(see page 149)*.	To get or set the Timebase value of the HSC.

Part VI
Frequency Meter Type

Overview

This part describes the use of an HSC in **Frequency meter** type.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
13	Frequency Meter Principle	97
14	Frequency Meter with a Main Type	99

Chapter 13
Frequency Meter Principle

Description

Overview

The **Frequency meter** type measures an event frequency in Hz.

The **Frequency meter** type calculates the number of pulses in time intervals of 1 s. An updated value in Hz is available for each time base value (10, 100, or 1000 ms).

When there is a variation in the frequency, the value restoration time is 1 s with a value precision of 1 Hz.

Operation Limits

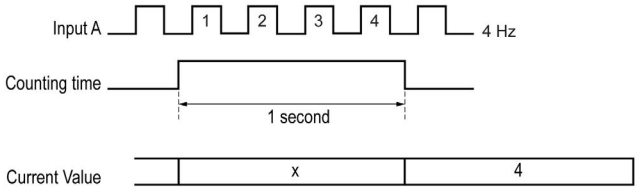
The maximum frequency that the module can measure on the A input is 200 kHz. Beyond 200 kHz, the counting register value may decrease until it reaches 0.

If the expert function is configured with a regular I/O, the minimum period admissible is 0.4 ms.

The maximum duty cycle at 200 kHz is 60%.

Synopsis Diagram

This diagram provides an overview of the **Frequency meter** principle:



Chapter 14
Frequency Meter with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Frequency meter** mode with a **Main** type.

What Is in This Chapter?

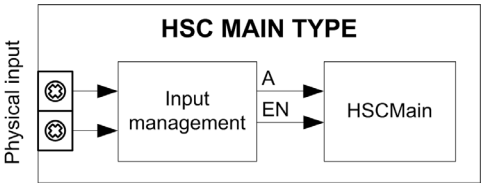
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	100
Configuration of the Frequency Meter Type	101
Programming	102

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Frequency meter** type:



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

In addition to the **Frequency meter** type, the **Main** type can provide the following function:

- Enable function *(see page 137)*

Configuration of the Frequency Meter Type

Procedure

Follow this procedure to configure a **Frequency Meter** type:

Step	Action
1	Double-click **MyController → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **Frequency Meter**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE:** **Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	In **Counting Inputs → A input → Location** select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
5	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
6	Set the value of the **Range → Time base** parameter to determine the period during which the number of events is counted. Select the measurement of the update cycle time: ● 10 ms ● 100 ms ● 1000 ms (default value)
7	Optionally, set the value of the **Control inputs → EN input → Location** parameter to enable the Enable Function *(see page 137)*.

Programming

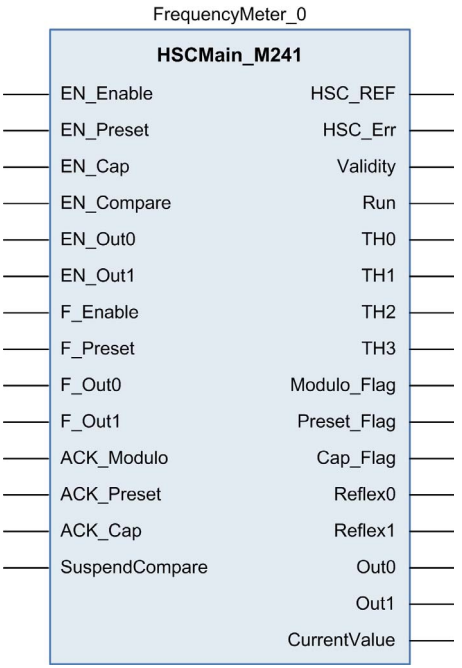
Overview

The **Main** type is always managed by an `HSCMain_M241` function block.

NOTE: At build time, an error is detected if the `HSCMain_M241` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCMain_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Frequency meter** type.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	If TRUE and the **EN** input is configured, authorizes the counter to be enabled using the Enable input *(see page 137)*.
EN_Preset	BOOL	Not used.
EN_Cap	BOOL	Not used.
EN_Compare	BOOL	Not used.
EN_Out0	BOOL	Not used.
EN_Out1	BOOL	Not used.
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, restarts the internal timer relative to the time base.
F_Out0	BOOL	Not used.
F_Out1	BOOL	Not used.
ACK_Modulo	BOOL	Not used.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	Not used.
SuspendCompare	BOOL	Not used

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the **Administrative** function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 158)* function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	Counter is running
TH0	BOOL	Not used.
TH1	BOOL	Not used.
TH2	BOOL	Not used.
TH3	BOOL	Not used.

Outputs	Type	Comment
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 134)*
Cap_Flag	BOOL	Not used.
Reflex0	BOOL	Not used.
Reflex1	BOOL	Not used.
Out0	BOOL	Not used.
Out1	BOOL	Not used.
CurrentValue	DINT	Current value of the counter.

Part VII
Period Meter Type

Overview

This part describes the use of an HSC in **Period meter** type.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
15	Period Meter Type Principle	107
16	Period Meter with a Main Type	109

Chapter 15
Period Meter Type Principle

Description

Overview

Use the **Period meter** type to:
- Determine the duration of an event
- Determine the time between two events
- Set and measure the execution time for a process.

The **Period meter** can be used in two ways:
- Edge to opposite: Allows measurement of the duration of an event.
- Edge to edge: Allows measurement of the time between two events.

The measurement is expressed in the units defined by the **Resolution** parameter (0.1 µs, 1 µs, 100 µs, 1000 µs).

For example, if the current value `CurrentValue` = 100 and the **Resolution** parameter is:
0.0001 (0.1 µs) measurement = 0.01 ms
0.001 (1 µs) measurement = 0.1 ms
0.1 (100 µs) measurement = 10 ms
1 (1000 µs) measurement = 100 ms

A timeout value can be specified in the configuration screen. Measurement is stopped if this timeout value is exceeded. In this case, the counting register is not valid until the next complete measurement.

Edge to Opposite Mode

The Edge to Opposite mode measures the duration of an event.

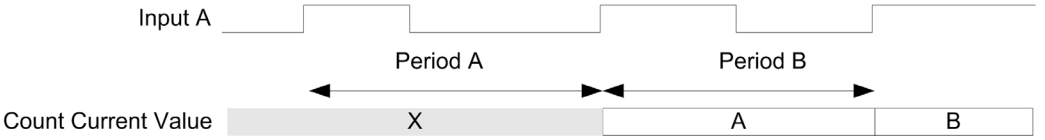
When the Enable condition = 1, the measurement is taken between the rising edge and the falling edge of the A input. The counting register is updated as soon as the falling edge is detected.



Edge to Edge Mode

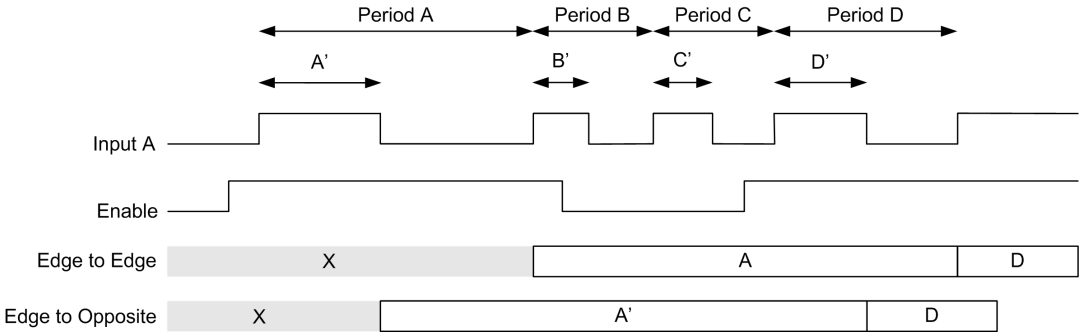
The Edge to Edge mode measures the elapsed time between two events.

When the Enable condition = 1, the measurement is taken between two rising edges of the A input. The counting register is updated as soon as the second rising edge is detected.



Enable Condition Interruption Behavior

The trend diagram below describes the behavior of the counting register when the Enable condition is interrupted:



Operating Limits

The module can perform a maximum of one measurement every 5 ms.

The shortest pulse that can be measured is 100 µs, even if the unit defined in the configuration is 1 µs.

The maximum duration that can be measured is 1,073,741,823 units.

Chapter 16
Period Meter with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Period meter** mode with a **Main** type.

What Is in This Chapter?

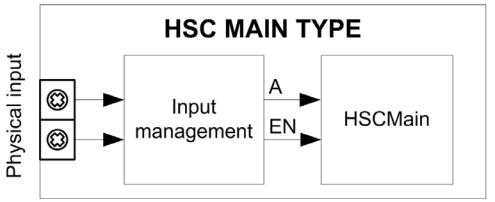
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	110
Configuration of the Period Meter Type in Edge to Edge Mode	111
Configuration of the Period Meter Type in Edge to Opposite Mode	112
Programming	113
Adjusting Parameters	116

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main type** in **Period meter** type:



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

In addition to the **Period meter** type, the **Main** type can provide the following function:

- Enable function *(see page 137)*

Configuration of the Period Meter Type in Edge to Edge Mode

Procedure

Follow this procedure to configure a **Period Meter** type in **Edge to Edge** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **Period Meter**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → PeriodMeter Mode** parameter to **Edge to Edge**.
5	In **Counting Inputs → A input → Location**, select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the inputs. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
7	Set the value of the **Range → Resolution** parameter. Select the unit of measurement: ● 0.1 µs ● 1 µs (default value) ● 100 µs ● 1000 µs
8	Enter the value of the **Range → Timeout** parameter to set the time value that a measured period must not exceed.
9	Optionally, you can enable these functions: ● Enable function *(see page 137)*

Configuration of the Period Meter Type in Edge to Opposite Mode

Procedure

Follow this procedure to configure a **Period Meter** type in **Edge to Opposite** mode:

Step	Action
1	Double-click **MyController → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **Period Meter**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → PeriodMeter Mode** parameter to **Edge to Opposite**.
5	In **Counting Inputs → A input → Location**, select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Bounce filter** parameter to reduce the bounce effect on the inputs. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
7	Set the value of the **Range → Resolution** parameter. Select the unit of measurement: ● 0.1 µs ● 1 µs (default value) ● 100 µs ● 1000 µs
8	Enter the value of the **Range → Timeout** parameter to set the time value that a measured period must not exceed.
9	Optionally, you can enable these functions: ● Enable function *(see page 137)*

Programming

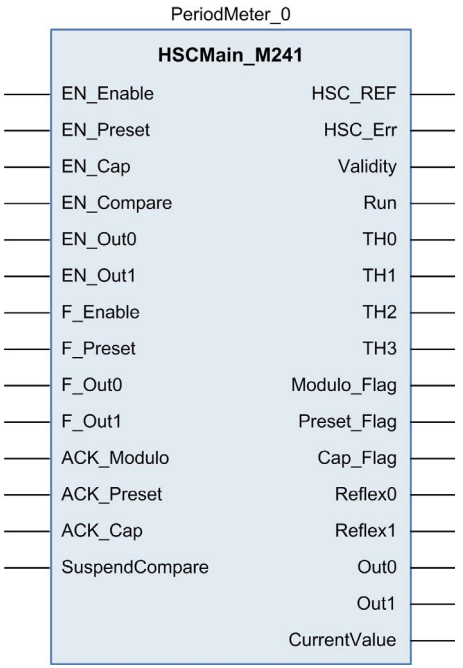
Overview

The **Main** type is always managed by an `HSCMain_M241` function block.

NOTE: At build time, an error is detected if the `HSCMain_M241` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → HSC → HSCMain_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking: ??? ... Using the input assistant, the HSC instance can be selected at the following path: **<MyController> → Counters**.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Period meter** type.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes the counter enable via the Enable input *(see page 137)*.
EN_Preset	BOOL	Not used.
EN_Cap	BOOL	Not used.
EN_Compare	BOOL	Not used.
EN_Out0	BOOL	Not used
EN_Out1	BOOL	Not used
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	Not used.
F_Out0	BOOL	Not used.
F_Out1	BOOL	Not used.
ACK_Modulo	BOOL	Not used.
ACK_Preset	BOOL	Not used.
ACK_Cap	BOOL	Not used.
SuspendCompare	BOOL	Not used

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the **Administrative** function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 158)* function block used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid. If the time-out value is exceeded, Validity = FALSE.
Run	BOOL	TRUE = Counter is running.
TH0	BOOL	Not used.
TH1	BOOL	Not used.
TH2	BOOL	Not used.

Outputs	Type	Comment
TH3	BOOL	Not used.
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Not used.
Cap_Flag	BOOL	Not used.
Reflex0	BOOL	Not used.
Reflex1	BOOL	Not used.
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
CurrentValue	DINT	Current value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by using the EXPERTGetParam *(see page 162)* or EXPERTSetParam *(see page 164)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M241 Logic Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 151)* which can be read or modified while the program is running:

Parameter	Description
EXPERT_TIMEBASE	To get or set the Resolution value of the HSC.
EXPERT_PERIODMETER_RESOLUTION_TYPE	To dynamically read or modify the time base. For more information, refer to Type for period meter *(see page 152)*.

Part VIII
Optional Functions

Overview

This part provides information on optional functions for HSC.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
17	Comparison Function	119
18	Capture Function	129
19	Preset and Enable Functions	133

Chapter 17
Comparison Function

Overview

This chapter provides information on the comparison function for the HSC.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Comparison Principle with a **Main** type	120
Configuration of the Comparison on a **Main** Type	125
External Event Configuration	126

Comparison Principle with a Main type

Overview

The compare block with the **Main** type manages thresholds, reflex outputs and events in the following modes:

- One-shot *(see page 33)*
- Modulo-loop *(see page 47)*
- Free-Large *(see page 67)*

Comparison is configured in the Configuration screen *(see page 125)* by activating at least one threshold.

Comparison can be used to trigger:

- a programming action on thresholds *(see page 122)*
- an event on a threshold associated with an external task *(see page 121)*
  NOTE: This option is only available for TM3XF• expansion modules, which support external events.
- reflex outputs *(see page 122)*.

Principle of a Comparison

The **Main** type can manage up to four thresholds.

A threshold is a configured value that is compared to the current counting value. Thresholds are used to define up to five zones or to react to a value crossing the threshold value.

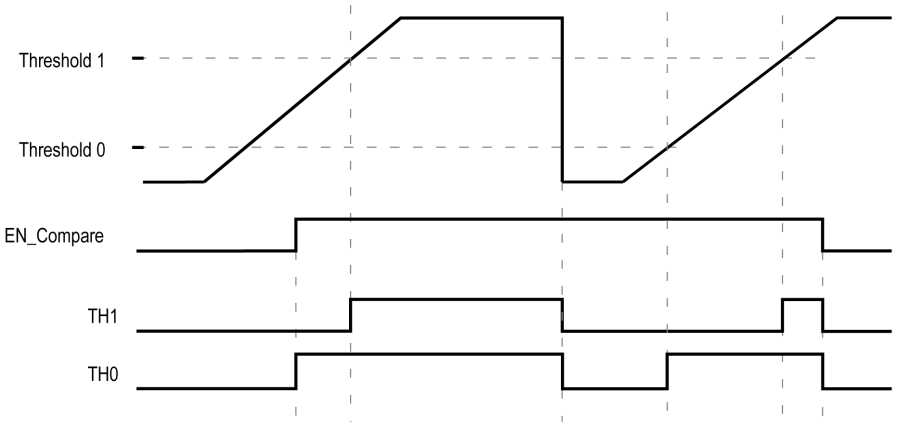
Threshold values are defined in the configuration window and can also be adjusted in the application program by using the EXPERTSetParam *(see page 164)* function block.

If Thresholdx (x= 0, 1, 2, 3) is configured and comparison is enabled (`EN_Compare` = 1), output pin THx of the `HSCMain_M241` function block is:

- set when counter value >= Thresholdx
- reset when counter value < Thresholdx

NOTE: When `EN_Compare` is set to 0 on `HSCMain_M241` function block, comparison functions are disabled, including external tasks triggered by a threshold event and Reflex outputs.

The following example for Modulo loop with two thresholds shows comparison in the `HSCMain_M241` function block:



Configuring Event Triggering in HSC Main Single or Dual Phase

Configuring an event on threshold crossing allows to trigger an external task *(see page 126)*. You can choose to trigger an event when a configured threshold is crossed as follows:

- **Upward Cross**. The event is triggered when the measured value goes above the threshold value.
- **Downward Cross**. The event is triggered when the measured value goes below the threshold value.
- **Both Cross**. The event is triggered when the measured value goes above the threshold value and when the measured value goes below the threshold value.

Configuring Event Triggering in Period Meter Mode

Configuring an event allows to trigger an external task *(see page 126)*. You can choose to trigger an event as follows:

- **Below threshold value**. The event is triggered when the measured value is lower than the threshold value.
- **Above threshold value**. The event is triggered when the measured value is higher than the threshold value.
- **Between threshold values**. The event is triggered when the measured value is between two threshold values.

Threshold Behavior

Using thresholds comparison status available in the task context (`TH0` to `TH2` output pins of the function block) is suitable for an application with a low time constant.

It can be used, for example, to monitor the liquid level in a tank.

Reflex Output Behavior

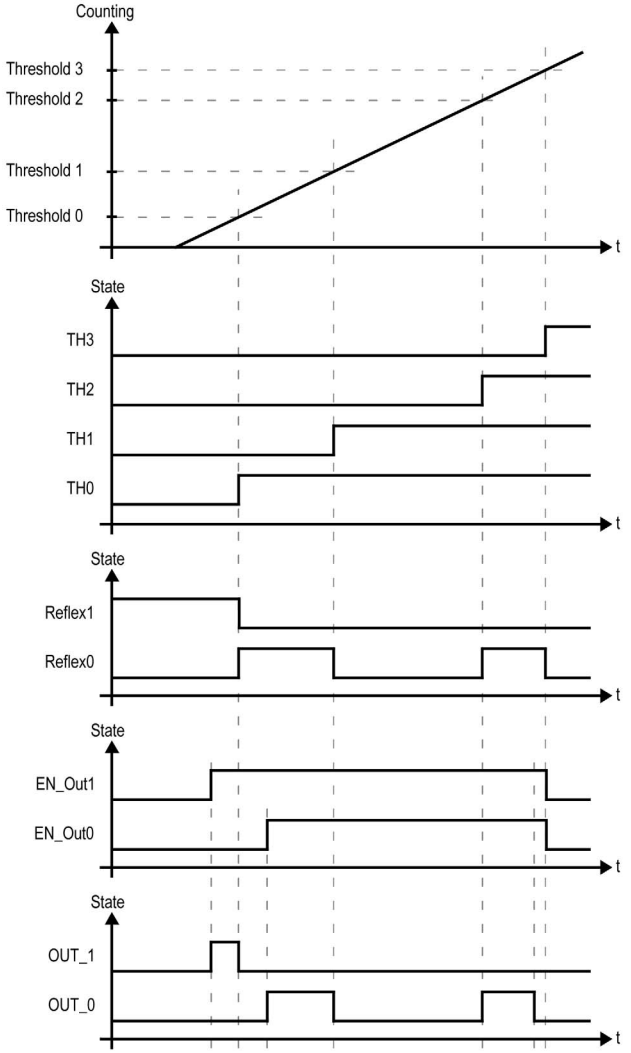
Configuring reflex outputs allows to trigger physical reflex outputs.

These outputs are not controlled in the task context, reducing the reaction time to a minimum. This is convenient for operations that need fast execution.

Outputs used by the High Speed Counter can only be accessed through the function block. They cannot be read or written directly within the application.

The performance is directly linked with the type of output used: fast or regular. For more information, refer to Embedded Expert I/O Assignment *(see page 17)*.

Example of the reflex outputs triggered by threshold:



NOTE: The state of the reflex outputs depends on the configuration.

Changing the Threshold Values

Care must be exercised when threshold compares are active to avoid unintended or unexpected results from the outputs or from sudden Event task execution. If the compare function is disabled, threshold values can be modified freely. However, if the compare function is enabled, suspend at least the threshold compare function while modifying the threshold values.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not change the Threshold values without using the `SuspendCompare` input if `EN_Compare` is equal to 1.
- Verify that `TH0` is less than `TH1`, that `TH1` is less than `TH2`, and that `TH2` is less than `TH3` before reactivating the threshold compare function.

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

While `EN_Compare` = 1, the comparison is active, and it is necessary to follow this procedure to apply changes to threshold values:

Step	Action
1	Set `SuspendCompare` to 1. The comparison is frozen at the current value: ● The `TH0`, `TH1`, `Reflex0`, `Reflex1`, `Out0`, and `Out1` output bits of the function block maintain their last value. ● Physical outputs 0, 1 maintain their last value ● Events are masked **NOTE:** `EN_Compare`, `EN_Out0`, `EN_Out1`, `F_Out0`, and `F_Out1` remain operational while `SuspendCompare` is set.
2	Modify the threshold values as needed using the `EXPERTSetParam` *(see page 162)* function block. **NOTE:** Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3.
3	Set `SuspendCompare` to 0. The new threshold values are applied and the comparison is resumed.

Configuration of the Comparison on a Main Type

Configuration Procedure

Follow this procedure to configure the comparison function on a **Main** type:

Step	Action
1	In the **Devices tree**, double-click **MyController → Counters**.
2	Set the value of the **Counting function** parameter to **HSC Main Single Phase** or **HSC Main Dual Phase**.
3	In the **Number of thresholds** parameter, select the number of thresholds to use.
4	Set the value of each threshold. **NOTE:** Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3
5	Optionally, define event conditions for the thresholds: 1. Configure external events *(see page 126)* associated with tasks. 2. In **Events → Threshold x**, set a trigger type (**Upward Cross**, **Downward Cross**, **Both Cross**) 3. In **HSC Main Id**, select the group of external events (HSC0...HSC3) containing the external event. **Result:** External events in the selected group (HSCx_TH0, HSCx_TH1, HSCx_TH2, HSCx_TH3, HSCx_STOP) appear below **Threshold x External Event**.

External Event Configuration

Procedure

The following procedure describes how to configure an external event *(see Modicon M241 Logic Controller, Programming Guide)* to activate a task:

Step	Action
1	In the **Applications tree** tab, add a task.
2	Double-click the task node to associate it with to an external event.
3	In the **Type** dropdown menu, select **External**.
4	In the **External event** dropdown menu, select the event to associate to the task (see the list below).

External Events

This table provides a description of the possible external events to associate to a task:

Event Name	Description
I0	Task is activated when the input I0 is set to 1.
I1	Task is activated when the input I1 is set to 1.
I2	Task is activated when the input I2 is set to 1.
I3	Task is activated when the input I3 is set to 1.
I4	Task is activated when the input I4 is set to 1.
I5	Task is activated when the input I5 is set to 1.
I6	Task is activated when the input I6 is set to 1.
I7	Task is activated when the input I7 is set to 1.
HSC0_TH0	Task is activated when the threshold TH0 of the HSC0 is set to 1.
HSC0_TH1	Task is activated when the threshold TH1 of the HSC0 is set to 1.
HSC0_TH2	Task is activated when the threshold TH2 of the HSC0 is set to 1.
HSC0_TH3	Task is activated when the threshold TH3 of the HSC0 is set to 1.
HSC0_STOP	Task is activated when the HSC0.Value is set to 0.
HSC1_TH0	Task is activated when the threshold TH0 of the HSC1 is set to 1.
HSC1_TH1	Task is activated when the threshold TH1 of the HSC1 is set to 1.
HSC1_TH2	Task is activated when the threshold TH2 of the HSC1 is set to 1.
HSC1_TH3	Task is activated when the threshold TH3 of the HSC1 is set to 1.
HSC1_STOP	Task is activated when the HSC1.Value is set to 0.
HSC2_TH0	Task is activated when the threshold TH0 of the HSC2 is set to 1.
HSC2_TH1	Task is activated when the threshold TH1 of the HSC2 is set to 1.

Event Name	Description
HSC2_TH2	Task is activated when the threshold TH2 of the HSC2 is set to 1.
HSC2_TH3	Task is activated when the threshold TH3 of the HSC2 is set to 1.
HSC2_STOP	Task is activated when the HSC2.Value is set to 0.
HSC3_TH0	Task is activated when the threshold TH0 of the HSC3 is set to 1.
HSC3_TH1	Task is activated when the threshold TH1 of the HSC3 is set to 1.
HSC3_TH2	Task is activated when the threshold TH2 of the HSC3 is set to 1.
HSC3_TH3	Task is activated when the threshold TH3 of the HSC3 is set to 1.
HSC3_STOP	Task is activated when the HSC3.Value is set to 0.

NOTE: The Stop event is only available on HSC Main Single Phase, One-shot mode.

Chapter 18
Capture Function

Overview

This chapter provides information on capture function for HSC.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Capture Principle with a **Main** Type	130
Configuration of the Capture on a **Main** Type	132

Capture Principle with a Main Type

Overview

The capture function stores the current counter value when an external input signal is detected.

The capture function is available in **Main** type with the following modes:

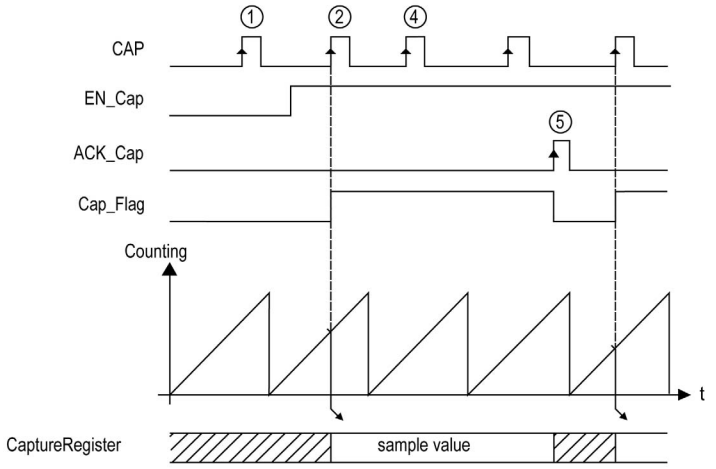
- One-shot *(see page 39)*
- Modulo-loop *(see page 59)*
- Free-large *(see page 75)*

To use this function:

- configure the optional Capture input **CAP**
- use the `EXPERTGetCapturedValue` *(see page 156)* function block to retrieve the captured value in your application.

Principle of a Capture

This graphic illustrates how the capture works in **Modulo-loop** mode:



Stage	Action
1	When EN_Cap = 0, the function is not operational.
2	When EN_Cap = 1, the edge on **CAP** captures the current counter value, puts it into the Capture register, and triggers the rising edge of Cap_Flag.
3	Get the stored value using EXPERTGetCapturedValue *(see page 156)*.
4	While Cap_Flag = 1, any new edge on the physical input CAP is ignored.
5	The rising edge of HSCMain_M241 *(see page 166)* function block input ACK_Cap triggers the falling edge Cap_Flag output. A new capture is authorized.

Configuration of the Capture on a Main Type

Configuration Procedure

Follow this procedure to configure the capture function on a **Main** type:

Step	Action
1	In the **Devices tree**, double-click **MyController → Counters**.
2	Set the value of the **Counting function** parameter to **HSC Main Single Phase** or **HSC Main Dual Phase**.
3	Select a value for the **Capture → CAP input → Location**.
4	Select a value for the **Capture → CAP input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.
5	Select a triggering mode for the **Capture → Mode** parameter: ● Preset *(see page 134)* (default value) ● **CAP Rising** ● **CAP Falling** ● **CAP Both**

Chapter 19
Preset and Enable Functions

Overview

This chapter provides information on preset and enable functions for an HSC.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Preset Function	134
Free-large or Period Meter Preset Conditions	136
Enable: Authorize Counting Operation	137

Preset Function

Overview

The preset function is used to set/reset the counter operation.

The preset function authorizes counting function, synchronization, and start in the following counting modes:

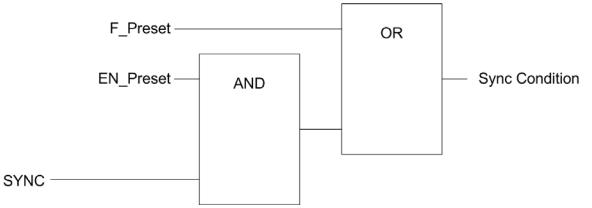
- **One shot** counter: preset and start the counter
- **Modulo-loop** counter: reset and start the counter
- **Event counting**: restart the internal time base at the beginning

NOTE: Sync condition for a **Simple** HSC type corresponds to the function block input `Sync`.

Description

This function is used to synchronize the counter depending on the status and the configuration of the optional SYNC physical input and the function block inputs `F_Preset` and `EN_Preset`.

This diagram illustrates the Sync conditions of the HSC:



EN_Preset input of the HSC function block
F_Preset input of the HSC function block
SYNC physical input SYNC

The function block output `Preset_Flag` is set 1 when the Sync Condition is reached.

Either of the following events trigger the capturing of the Sync Condition:

- Rising edge of the `F_Preset` input
- Rising edge, falling edge, or rising and falling edge, of the SYNC physical input (if the SYNC input is configured, and the `EN_Preset` input is TRUE).

Configuration

This procedure describes how to configure a preset function:

Step	Action
1	In the **Devices tree**, double-click **MyController → Counters**.
2	Set the value of the **Counting function** parameter to **HSC Main Single Phase** or **HSC Main Dual Phase**.
3	Select the value of the **Control inputs → SYNC input → Location** parameter.
4	Select the value of the **Control inputs → SYNC input → Bounce filter** parameter.
5	Select the value of the **Control inputs → SYNC input → Preset condition** parameter to specify the transition type of the SYNC physical input: • **SYNC Rising**. Rising edge of the SYNC input • **SYNC Falling**. Falling edge of the SYNC input • **SYNC Both**. Both edges of the SYNC input

Free-large or Period Meter Preset Conditions

Overview

In **Free-large** mode, the Preset condition is created by using one physical input:

- SYNC

Preset condition available:

- At the edge of the input SYNC (rising)

At the Edge of the Input SYNC (Rising)

The counter synchronizes upon the encoder reference point.

Enable: Authorize Counting Operation

Overview

The enable function is used to authorize the counting operation.

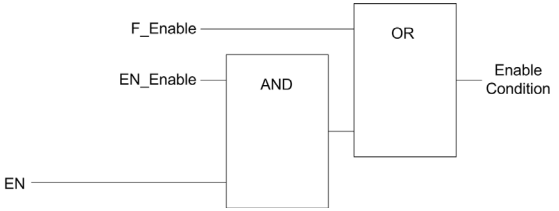
The enable function is available in the following HSC modes:

- HSC Main Single Phase (One-shot)
- HSC Main Single Phase (Modulo Loop)
- Frequency Meter
- Period Meter

Description

This function is used to authorize changes to the current counter value depending on the status of the optional `EN` physical input and the function block inputs `F_Enable` and `EN_Enable`.

The following diagram illustrates the enable conditions:



EN_Enable input of the HSC function block
F_Enable input of the HSC function block
EN physical input Enable

As long as the function is not enabled, the counting pulses are ignored.

NOTE: Enable condition for a **Simple** type corresponds to the function block input `Enable`.

Configuration

This procedure describes how to configure an Enable function:

Step	Action
1	In the **Devices tree**, double-click **MyController → Counters**.
2	Select the **Counters** tab.
3	Select a **Counting function** that supports the Enable function: • **HSC Main Single Phase** (**One-shot** or **Modulo-loop**) • **Frequency Meter** • **Period Meter**
4	Set the value of the **Control inputs → EN input → Location** parameter.
5	Select the value of the **Control inputs → EN input → Bounce filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 142)*.

Appendices

Overview

This appendix extracts parts of the programming guide for technical understanding of the library documentation.

What Is in This Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	General Information	141
B	Data Types	145
C	Function Blocks	155
D	Function and Function Block Representation	173

Appendix A
General Information

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Dedicated Features	142
General Information on Administrative and Motion Function Block Management	143

Dedicated Features

Bounce Filter

This table shows the maximum counter frequencies determined by the filtering values used to reduce the bounce effect on the input:

Input	Bounce Filter Value (ms)	Maximum Counter Frequency Expert	Maximum Counter Frequency Regular
A B	0.000	200 kHz	1 kHz
	0.001	200 kHz	1 kHz
	0.002	200 kHz	1 kHz
	0.005	100 kHz	1 kHz
	0.01	50 kHz	1 kHz
	0.05	25 kHz	1 kHz
	0.1	5 kHz	1 kHz
	0.5	1 kHz	1 kHz
	1	500 Hz	500 Hz
	5	100 Hz	100 Hz
A is the counting input of the counter. B is the counting input of the dual phase counter.			

Dedicated Outputs

Outputs used by the high speed expert functions can only be accessed through the function block. They cannot be read or written directly within the application.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use the same function block instance in different program tasks.
- Do not modify or otherwise 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.

General Information on Administrative and Motion Function Block Management

Management of Input Variables

At the Execute input rising edge, the function block starts.

Any further modifications of the input variables are not taken into account.

Following the IEC 61131-3 standards, if any variable input to a function block is missing, that is, left open or unconnected, then the value from the previous invocation of the instance of the function block will be used. In the first invocation, the initial, configured value is applied in this case. Therefore, it is best that a function block always has known values attributed to its inputs to help avoid difficulties in debugging your program. For HSC and PTO function blocks, it is best to use the instance only once, and preferably the instance be in the main task.

Management of Output Variables

The Done, InVelocity, or InFrequency output is mutually exclusive with Busy, CommandAborted, and Error outputs: only one of them can be TRUE on one function block. If the Execute input is TRUE, one of these outputs is TRUE.

At the rising edge of the Execute input, the Busy output is set. This Busy output remains set during the function block execution, and is reset at the rising edge of one of the other outputs (Done, InVelocity, InFrequency, CommandAborted, and Error).

The Done, InVelocity, or InFrequency output is set when the function block execution has been completed successfully.

When a function block execution is interrupted by another one, the CommandAborted output is set instead.

When a function block execution ends due to a detected error, the Error output is set and the detected error number is given through the ErrId output.

The Done, InVelocity, InFrequency, Error, ErrID, and CommandAborted outputs are reset with the falling edge of Execute. If Execute input is reset before the execution is finished, then the outputs are set for one task cycle at the execution ending.

When an instance of a function block receives a new Execute before it is finished, the function block does not return any feedback, such as Done, for the previous action.

Handling a Detected Error

All blocks have 2 outputs that can report a detected error during the execution of the function block:

- Error = TRUE when an error is detected.
- ErrID When Error = TRUE, returns the detected error ID.

Appendix B
Data Types

Overview

This chapter describes the data types of the HSC Library.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
EXPERT_DIAG_TYPE: Type for EXPERTGetDiag Diagnostics	146
EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block	147
EXPERT_FREQMETER_TIMEBASE_TYPE: Type for Frequency Meter Time Base Variable	148
EXPERT_HSCMAIN_TIMEBASE_TYPE: Type for HSC Main Time Base Variable	149
EXPERT_IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block	150
EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERT	151
EXPERT_PERIODMETER_RESOLUTION_TYPE: Type for Period Meter Time Base Variable	152
EXPERT_REF: EXPERT Reference Value	153

EXPERT_DIAG_TYPE: Type for EXPERTGetDiag Diagnostics

Enumerated Type Description

This enumeration describes the different counter errors that can be read by the `EXPERTGetDiag` function block:

Name	Value	Comment
`EXPERT_NO_ERROR`	0	No error has occurred.
`EXPERT_PERIODMETER_TIMEOUT_REACHED`	1	Timeout on period measure is reached.
`EXPERT_SHORTCUT_DETECTED`	4	Shortcut detected on HSC Main reflex output
`EXPERT_CONFIGURATION_FAULT`	128	Counter is incorrectly configured.

EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block

Enumerated Type Description

The enumeration data type ENUM contains the different types of detected error with the following values:

Enumerator	Value	Description
EXPERT_NO_ERROR	00 hex	No error detected.
EXPERT_UNKNOWN	01 hex	The reference EXPERT is incorrect or not configured.
EXPERT_UNKNOWN_PARAMETER	02 hex	The parameter reference is incorrect. See PARAMETER_TYPE section for valid parameters *(see page 151)*.
EXPERT_INVALID_PARAMETER	03 hex	The value of the parameter is incorrect. For example, Preset Value is <TH1 or <TH0.
EXPERT_COM_ERROR	04 hex	Communication error was detected with the EXPERT module.
EXPERT_CAPTURE_NOT_CONFIGURED	05 hex	Capture is not configured. It is impossible to get a captured value.

EXPERT_FREQMETER_TIMEBASE_TYPE: Type for Frequency Meter Time Base Variable

Enumerated Type Description

The enumeration data type ENUM contains the different time base values allowed for use with an EXPERT function block:

Name	Value
EXPERT_FREQMETER_10ms	10
EXPERT_FREQMETER_100ms	100
EXPERT_FREQMETER_1000ms	1000

EXPERT_HSCMAIN_TIMEBASE_TYPE: Type for HSC Main Time Base Variable

Enumerated Type Description

The enumeration data type ENUM contains the different time base values allowed for use with an EXPERT Main function block:

Name	Value
EXPERT_HSCMAIN_100ms	00 hex
EXPERT_HSCMAIN_1s	01 hex
EXPERT_HSCMAIN_10s	02 hex
EXPERT_HSCMAIN_60s	03 hex

EXPERT_IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block

Enumerated Type Description

The enumeration data type ENUM contains the different types of detected error with the following values:

Enumerator	Value	Description
EXPERT_IMMEDIATE_FUNC_NO_ERROR	00 hex	No error detected
EXPERT_IMMEDIATE_FUNC_UNKNOWN	01 hex	The reference of IMMEDIATE function is incorrect or not configured
EXPERT_IMMEDIATE_FUNC_UNKNOWN_PARAMETER	02 hex	A parameter reference is incorrect

EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERT

Enumerated Type Description

The enumeration data type ENUM contains the following values:

Enumerator	Value	Description
EXPERT_PRESET	00 hex	To get or set the Preset value of an EXPERT function.
EXPERT_MODULO	01 hex	To get or set the Modulo value of an EXPERT function.
EXPERT_TIMEBASE	03 hex	To get or set the Timebase value *(see page 149)* of an EXPERT function.
EXPERT_THRESHOLD0	06 hex	To get or set the Threshold 0 value of an EXPERT function.
EXPERT_THRESHOLD1	07 hex	To get or set the Threshold 1 value of an EXPERT function.
EXPERT_THRESHOLD2	08 hex	To get or set the Threshold 2 value of an EXPERT function.
EXPERT_THRESHOLD3	09 hex	To get or set the Threshold 3 value of an EXPERT function.
EXPERT_REFLEX0	0A hex	To get or set output 0 reflex mode of an EXPERT function
EXPERT_REFLEX1	0B hex	To get or set output 1 reflex mode of an EXPERT function

EXPERT_PERIODMETER_RESOLUTION_TYPE: Type for Period Meter Time Base Variable

Enumerated Type Description

The enumeration data type ENUM contains the different time base values allowed for use with an EXPERT function block:

Name	Value
EXPERT_PERIODMETER_100ns	FFFFFFFF hex (-1 decimal)
EXPERT_PERIODMETER_1µs	00 hex (0 decimal)
EXPERT_PERIODMETER_100µs	01 hex (1 decimal)
EXPERT_PERIODMETER_1000µs	02 hex (2 decimal)

EXPERT_REF: EXPERT Reference Value

Data Type Description

The EXPERT_REF is a byte used to identify the EXPERT function associated with the administrative block.

Appendix C
Function Blocks

Overview

This chapter describes the functions and the function blocks of the HSC Library.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
EXPERTGetCapturedValue: Read Value of Capture Registers	156
EXPERTGetDiag: Return Detail of a Detected HSC Error	158
EXPERTGetImmediateValue: Read Counter Value of HSC	160
EXPERTGetParam: Returns Parameters of HSC	162
EXPERTSetParam: Adjust Parameters of a HSC	164
HSCMain_M241: Control a Main Type Counter for M241	166
HSCSimple_M241: Control a Simple Type Counter for M241	170

EXPERTGetCapturedValue: Read Value of Capture Registers

Function Block Description

This administrative function block returns the content of a capture register.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
`EXPERT_REF_IN`	`EXPERT_REF` *(see page 153)*	Reference to the EXPERT function block. Must not be changed during block execution.
`Execute`	`BOOL`	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`CaptureNumber`	`BYTE`	Index of the capture register: 0

This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that CaptureValue is valid. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE *(see page 147)*	When Error is TRUE: type of the detected error.
CaptureValue	DINT	When Done is TRUE: Capture register value is valid.

NOTE: In case of detected error, variables take the last value captured.

NOTE: For more information about Done, Busy and Execution pins, refer to General Information on Function Block Management *(see page 143)*.

Adding the EXPERTGetCapturedValue Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → Administrative → EXPERTGetCapturedValue** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

EXPERTGetDiag: Return Detail of a Detected HSC Error

Function Block Description

This administrative function block returns the details of a detected HSC error.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block. Must not be changed during block execution.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.

This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that HSCDiag is valid. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE *(see page 147)*	When Error is TRUE: type of the detected error.
EXPERTDiag	DWORD	When Done is TRUE: diagnostic value is valid, refer to the table below.

NOTE: For more information about Done, Busy and Execution pins, refer to General Information on Function Block Management *(see page 143)*.

This table indicates the diagnostic values:

Bit	BASE (HSCMain or HSCSimple)	Description
0	–	No error detected
1	–	Timeout reached on period meter
2	–	Shortcut detected on HSC Main expert output
7	–	Error detected in the configuration of the counter

Adding the EXPERTGetDiag Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → Administrative → EXPERTGetDiag** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

EXPERTGetImmediateValue: Read Counter Value of HSC

Function Block Description

This administrative function block permits to read the counter value of an HSC bypassing the controller cycle.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.

This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that ExpertDiag is valid. Function block execution is finished.
Error	BOOL	TRUE = indicates that an error was detected.
ErrID	IMMEDIATE_FUNC_ERR_TYPE *(see page 150)*	When Error is TRUE: type of the detected error.
ImmediateValue	DINT	Contains the counter value.

Adding the EXPERTGetImmediateValue Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → Administrative → EXPERTGetImmediateValue** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

EXPERTGetParam: Returns Parameters of HSC

Function Block Description

This administrative function block returns a parameter value of an HSC.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block. Must not be changed during block execution.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
Param	EXPERT_PARAMETER_ TYPE *(see page 151)*	Parameter to read.

This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that ParamValue is valid. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE *(see page 147)*	When Error is TRUE: type of the detected error.
ParamValue	DINT	Value of the parameter that has been read.

NOTE: For more information about Done, Busy and Execution pins, refer to General Information on Function Block Management *(see page 143)*.

Adding the EXPERTGetParam Function Block

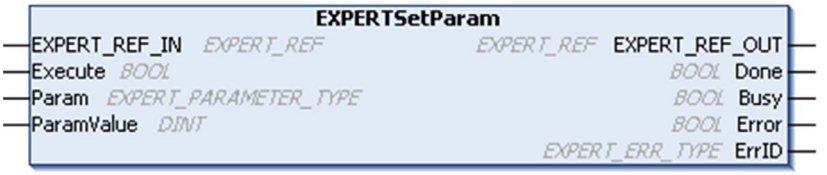
Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → Administrative → EXPERTGetParam** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

EXPERTSetParam: Adjust Parameters of a HSC

Function Block Description

This administrative function block modifies the value of a parameter of an HSC.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation (see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block. Must not be changed during block execution.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
Param	EXPERT_PARAMETER_TYPE *(see page 151)*	Parameter to read.
ParamValue	DINT	Parameter value to write.

This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF *(see page 153)*	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that the parameter was successfully written. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE *(see page 147)*	When Error is TRUE: type of the detected error.

NOTE: For more information about Done, Busy, and Execution pins, refer to General Information on Function Block Management *(see page 143)*.

Adding the EXPERTSetParam Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 HSC → Administrative → EXPERTSetParam** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

HSCMain_M241: Control a Main Type Counter for M241

Function Block Description

This function block controls a **Main** type counter with the following functions:

- up/down counting
- frequency meter
- thresholds
- events
- period meter
- dual phase

The HSC Main function block is mandatory when using **Main** counter.

The function block instance name must match the name defined by configuration. Hardware related information managed by this function block is synchronized with the MAST task cycle.

⚠ WARNING

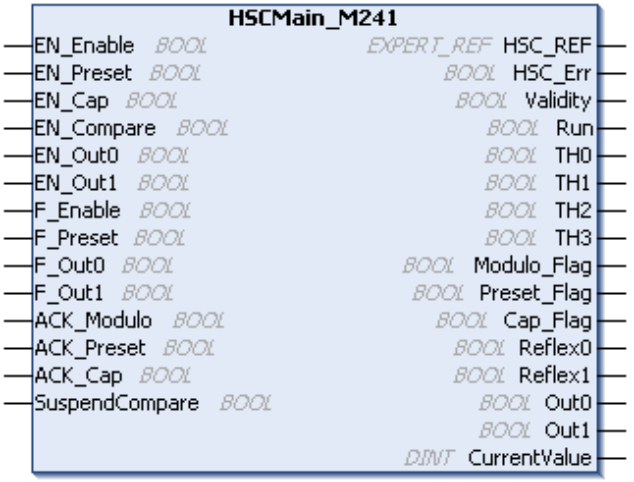
UNINTENDED OUTPUT VALUES

- Only use the Function Block instance in the MAST task.
- Do not use the same Function Block instance in a different task.

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

NOTE: Forcing the logical output values of the FB is allowed by EcoStruxure Machine Expert but it will have no impact on hardware related outputs if the function is active (executing).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Input	Type	Description
`EN_Enable`	`BOOL`	`TRUE` = authorizes enabling of the counter using the `Enable` input.
`EN_Preset`	`BOOL`	`TRUE` = authorizes counter synchronization and start using the Sync input.
`EN_Cap`	`BOOL`	`TRUE` = enables the Capture input (if configured in **One shot**, **Modulo loop**, **Free large** modes).
`EN_Compare`	`BOOL`	`TRUE` = enables the comparator operation (using Thresholds 0, 1, 2, 3): • basic comparison (`TH0`, `TH1`, `TH2`, `TH3` output bits) • reflex (`Reflex0`, `Reflex1` output bits) • events (to trigger external tasks on threshold crossing)
`EN_Out0`	`BOOL`	`TRUE` = enables Output0 to echo the `Reflex0` value (if configured in **One shot**, **Modulo loop**, **Free large** modes).
`EN_Out1`	`BOOL`	`TRUE` = enables Output1 to echo the `Reflex1` value (if configured in **One shot**, **Modulo loop**, **Free large** modes).
`F_Enable`	`BOOL`	`TRUE` = authorizes changes to the current counter value.

Input	Type	Description
F_Preset	BOOL	On rising edge, authorizes counting function synchronization and start in the following counting modes: **One-shot counter:** to preset and start the counter **Modulo loop counter:** to reset and start the counter **Free large counter:** to preset and start the counter **Event counter:** to restart the internal time base at the beginning **Frequency meter:** to restart the internal timer relative to the time base.
F_Out0	BOOL	TRUE = forces Output0 to 1 (if configured in **One-shot**, **Modulo loop**, **Free large** modes).
F_Out1	BOOL	TRUE = forces Output1 to TRUE (if configured in **One-shot**, **Modulo loop**, **Free large** modes).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag (**Modulo loop** and **Free large** modes).
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets the Cap_Flag (**One-shot**, **Modulo loop**, **Free large** modes).
SuspendCompare	BOOL	TRUE = compare results are suspended: • TH0, TH1, TH2, TH3 , Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. • Physical Outputs 0, 1 maintain their last value. • Compare events are masked. **NOTE:** EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC.
Validity	BOOL	TRUE = indicates that output values on the function block are valid. In the **Period Meter** Type, if the time-out value is exceeded, Validity = FALSE. In **One-Shot** mode, Validity is set to TRUE when a rising edge of **Preset** is detected.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the HSCGetDiag *(see page 158)* function block to get more information about this detected error.

Outputs	Type	Comment
Run	BOOL	TRUE = counter is running. In One-shot mode, the Run bit switches to 0 when CurrentValue reaches 0.
TH0	BOOL	TRUE = current counter value > Threshold 0 (if configured in **One shot**, **Modulo loop**, **Free large** modes). Only active when EN_Compare is set.
TH1	BOOL	TRUE = current counter value > Threshold 1 (if configured in **One shot**, **Modulo loop**, **Free large** modes). Only active when EN_Compare is set.
TH2	BOOL	TRUE = current counter value > Threshold 2 (if configured in **One-shot**, **Modulo loop**, **Free large** modes). Only active when EN_Compare is set.
TH3	BOOL	TRUE = current counter value > Threshold 3 (if configured in **One-shot**, **Modulo loop**, **Free large** modes). Only active when EN_Compare is set.
Modulo_Flag	BOOL	Set to TRUE when the counter rolls over its limits in the following modes: • **Modulo loop counter**: when the counter rolls over to the modulo or 0 • **Free large** counter: when the counter roll overs its limits
Preset_Flag	BOOL	Set to TRUE by the synchronization of: • **One-shot** counter: when the counter presets and starts • **Modulo loop** counter: when the counter resets • **Free large** counter: when the counter presets • **Event** counter: when the internal timer relative to the time base restarts • **Frequency meter**: when the internal timer relative to the time base restarts
Cap_Flag	BOOL	TRUE = indicates that a value has been latched in the capture register. This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0 (if configured in **One shot**, **Modulo loop**, **Free large** modes). Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1 (if configured in **One shot**, **Modulo loop**, **Free large** modes). Only active when EN_Compare is set.
Out0	BOOL	Indicates the state of Output0.
Out1	BOOL	Indicates the state of Output1.
CurrentValue	DINT	Current value of the counter.

HSCSimple_M241: Control a Simple Type Counter for M241

Function Block Description

This function block controls a **Simple** type counter with the following reduced functions:

- one-channel counting
- no threshold
- no event
- no capture
- no reflex

The `HSCSimple` function block is mandatory when using a **Simple** counter type.

The function block instance name must match the name defined by configuration. Hardware related information managed by this function block is synchronized with the MAST task cycle.

⚠ WARNING

UNINTENDED OUTPUT VALUES

- Only use the Function Block instance in the MAST task.
- Do not use the same Function Block instance in a different task.

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

NOTE: Forcing the logical output values of the FB is allowed by EcoStruxure Machine Expert but it will have no impact on hardware related outputs if the function is active (executing).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
Enable	BOOL	TRUE = authorizes changes to the current counter value.
Sync	BOOL	On rising edge, presets and starts the counter.
ACK_Modulo	BOOL	Modulo loop mode: On rising edge, resets the modulo flag Modulo_Flag.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 153)*	Reference to the HSC.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 158)* function block used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	TRUE = counter is running. In One-shot mode, switches to 0 when CurrentValue reaches 0. A rising edge on Sync is needed to restart the counter.
Modulo_Flag	BOOL	Module loop mode: Set to TRUE when the counter rolls over the modulo value.
CurrentValue	DWORD	Current count value of the counter.

Appendix D
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Differences Between a Function and a Function Block	174
How to Use a Function or a Function Block in IL Language	175
How to Use a Function or a Function Block in ST Language	178

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

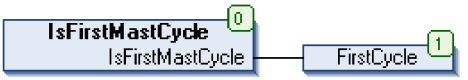
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

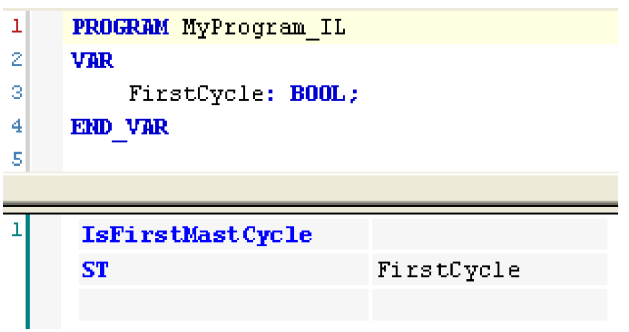
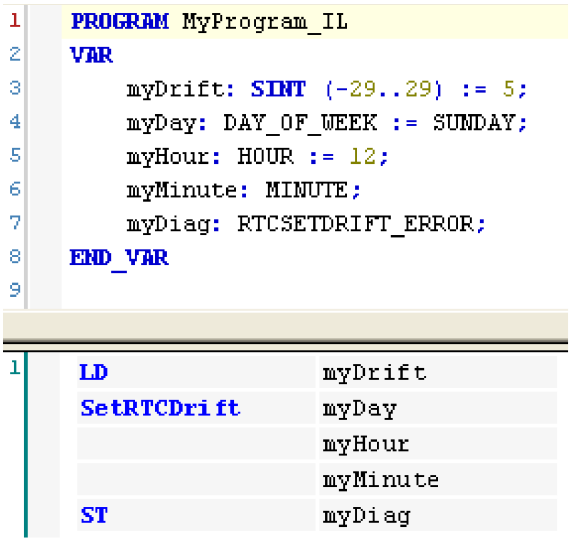
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

This procedure describes how to insert a function block in IL language:

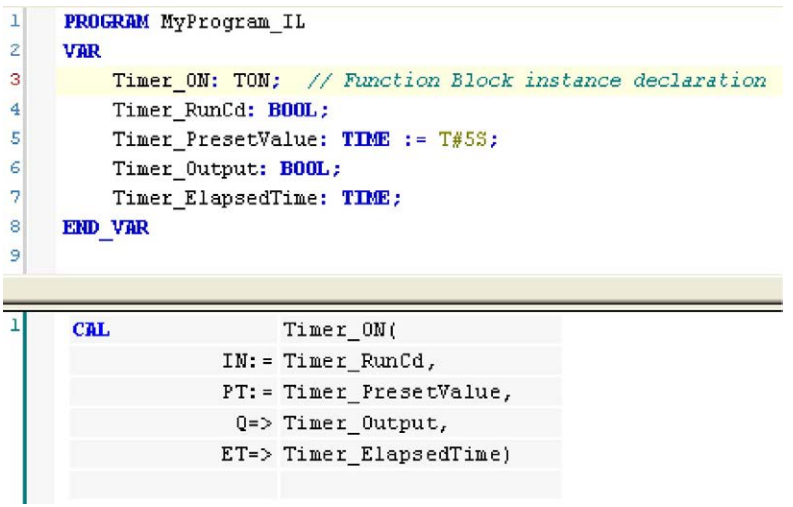
Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function block requires, including the instance name.

Step	Action
3	Function Blocks are called using a CAL instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the TON Function Block graphically presented below:

Function Block	Graphical Representation
TON	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

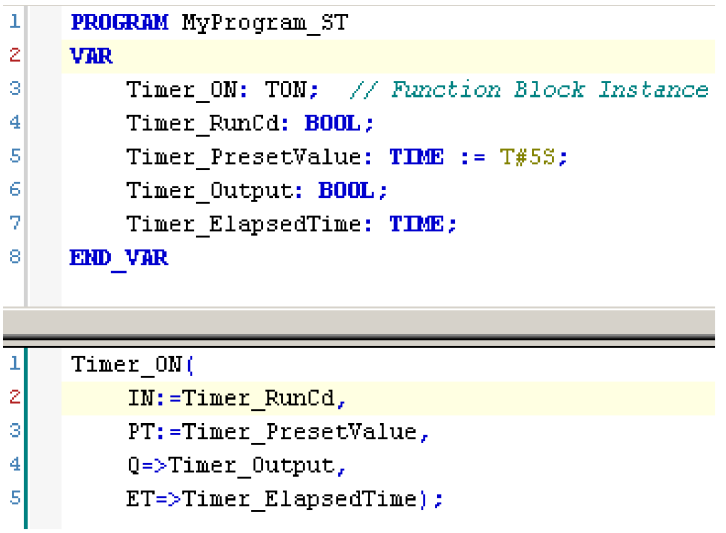
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

A

application

A program including configuration data, symbols, and documentation.

B

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

I

ID

(*identifier/identification*)

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).

INT

(*integer*) A whole number encoded in 16 bits.

L

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).

N

node

An addressable device on a communication network.

P

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

PTO

(*pulse train outputs*) A fast output that oscillates between off and on in a fixed 50-50 duty cycle, producing a square wave form. PTO is especially well suited for applications such as stepper motors, frequency converters, and servo motor control, among others.

S

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

V

variable

A memory unit that is addressed and modified by a program.

Index

B

Busy
management of status variables, *143*

C

Capture
HSCMain, *130*
capture register of HSC
EXPERTGetCapturedValue, *156*
CommandAborted
management of status variables, *143*
Comparison
HSCMain, *120*

D

data types
EXPERT_DIAG_TYPE, *146*
EXPERT_ERR_TYPE, *147*
EXPERT_FREQMETER_TIMEBASE_-TYPE, *148*
EXPERT_HSCMAIN_TIMEBASE_TYPE, *149*
EXPERT_IMMEDIATE_ERR_TYPE, *150*
EXPERT_PARAMETER_TYPE, *151*
EXPERT_PERIODMETER_RESOLU-TION_TYPE, *152*
HSC_REF, *153*
dedicated features, *142*
Done
management of status variables, *143*

E

Enable
authorize counting operation, *137*
ErrID
handling a detected error, *143*
management of status variables, *143*
Error
handling a detected error, *143*
management of status variables, *143*
Event Counting
HSC Modes of Embedded HSC, *85*
Execute
management of status variables, *143*
EXPERT_DIAG_TYPE
data types, *146*
EXPERT_ERR_TYPE, *147*
EXPERT_FREQMETER_TIMEBASE_TYPE
data types, *148*
EXPERT_HSCMAIN_TIMEBASE_TYPE
data types, *149*
EXPERT_IMMEDIATE_ERR_TYPE, *150*
EXPERT_PARAMETER_TYPE, *151*
EXPERT_PERIODMETER_RESOLUTION_-TYPE
data types, *152*
EXPERTGetCapturedValue
getting a capture register value, *156*
EXPERTGetDiag
getting the detected error on EXPERT I/O function, *158*
EXPERTGetImmediateValue
getting the counter value of an HSC, *160*
EXPERTGetParam
getting parameters values of an HSC, *162*
EXPERTSetParam
setting parameters values of an HSC, *164*

F

Free-large
HSC Modes of Embedded HSC, *70*
frequency meter
description, *97*
programming, *102*
synopsis, *100*
functions
differences between a function and a

function block, *174*
Enable, *137*
how to use a function or a function block in IL language, *175*
how to use a function or a function block in ST language, *178*

H

handling a detected error
ErrID, *143*
Error, *143*
high speed counter
EXPERTGetDiag, *158*
EXPERTGetImmediateValue, *160*
EXPERTGetParam, *162*
EXPERTSetParam, *164*
HSCMain_M241, *166*
HSCSimple_M241, *170*
HSC
EXPERTGetDiag, *158*
EXPERTGetImmediateValue, *160*
EXPERTGetParam, *162*
EXPERTSetParam, *164*
HSCMain_M241, *166*
HSCSimple_M241, *170*
HSC Modes of Embedded HSC
Event Counting, *85*
Free-large, *70*
Modulo-loop, *49*
HSC_REF, *153*
HSCMain
Capture, *130*
Comparison, *120*
HSCMain_M241
controlling a main type high speed counter (M241), *166*
HSCSimple_M241
controlling a simple type high speed counter (M241), *170*

M

M241 HSC
EXPERTGetCapturedValue, *156*
EXPERTGetDiag, *158*
EXPERTGetImmediateValue, *160*
EXPERTGetParam, *162*
EXPERTSetParam, *164*
HSCMain_M241, *166*
HSCSimple_M241, *170*
management of status variables
Busy, *143*
CommandAborted, *143*
Done, *143*
ErrID, *143*
Error, *143*
Execute, *143*
Modulo-loop
HSC Modes of Embedded HSC, *49*

P

period meter
description, *107*
parameters, *116*
programming, *113*
synopsis, *110*

Modicon M241 Logic Controller

PTOPWM

Library Guide

12/2019



EIO0000003077.01

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.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

© 2019 Schneider Electric. All rights reserved.

Table of Contents

	Safety Information	7
	About the Book	11
Part I	Introduction	15
Chapter 1	Expert Function Introduction	17
	Expert Functions Overview	18
	Embedded Expert I/O Assignment	21
Chapter 2	Generalities	25
	Dedicated Features	27
	General Information on Function Block Management	28
Part II	Pulse Train Output (PTO)	29
Chapter 3	Overview	31
	Pulse Train Output (PTO)	31
Chapter 4	Configuration	35
4.1	Configuration	36
	PTO Configuration	37
	Pulse Output Modes	42
	Acceleration / Deceleration Ramp	44
	Probe Event	48
	Backlash Compensation (Only Available in Quadrature Mode)	51
	Positioning Limits	53
4.2	Home Modes	56
	Homing Modes	57
	Position Setting	60
	Long Reference	61
	Long Reference & Index	63
	Short Reference Reversal	65
	Short Reference No Reversal	67
	Short Reference & Index Outside	69
	Short Reference & Index Inside	73
	Home Offset	77

Chapter 5 Data Unit Types 79
AXIS_REF_PTO Data Type 80
MC_BUFFER_MODE 81
MC_DIRECTION 83
PTO_HOMING_MODE 84
PTO_PARAMETER 85
PTO_ERROR 86
Chapter 6 Motion Function Blocks 89
6.1 Operation Modes 90
Motion State Diagram 91
Buffer Mode 93
Timing Diagram Examples 95
6.2 MC_Power_PTO Function Block 104
Description 105
MC_Power_PTO: Manage the Power of the Axis State 106
6.3 MC_MoveVelocity_PTO Function Block 109
Description 110
MC_MoveVelocity_PTO: Control the Speed of the Axis 111
6.4 MC_MoveRelative_PTO Function Block 115
Description 116
MC_MoveRelative_PTO: Command Relative Axis Movement 117
6.5 MC_MoveAbsolute_PTO Function Block 122
Description 123
MC_MoveAbsolute_PTO: Command Movement to Absolute Position . 124
6.6 MC_Home_PTO Function Block 128
Description 129
MC_Home_PTO: Command the Axis to Move to a Reference Position 130
6.7 MC_SetPosition_PTO Function Block 133
Description 134
MC_SetPosition_PTO: Force the Reference Position of the Axis 135
6.8 MC_Stop_PTO Function Block 136
Description 137
MC_Stop_PTO: Command a Controlled Motion Stop 138
6.9 MC_Halt_PTO Function Block 141
Description 142
MC_Halt_PTO: Command a Controlled Motion Stop until the Velocity equals Zero 143

6.10 Adding a Motion Function Block . . . 146
Adding a Motion Function Block . . . 146
Chapter 7 Administrative Function Blocks . . . 147
7.1 Status Function Blocks . . . 148
MC_ReadActualVelocity_PTO: Get the Commanded Velocity of the Axis . . . 149
MC_ReadActualPosition_PTO: Get the Position of the Axis . . . 151
MC_ReadStatus_PTO: Get the State of the Axis . . . 153
MC_ReadMotionState_PTO: Get the Motion Status of the Axis . . . 155
7.2 Parameters Function Blocks . . . 157
MC_ReadParameter_PTO: Get Parameters from the PTO . . . 158
MC_WriteParameter_PTO: Write Parameters to the PTO . . . 160
MC_ReadBoolParameter_PTO: Get B00L Parameters from the PTO 162
MC_WriteBoolParameter_PTO: Write B00L Parameters to the PTO . 164
7.3 Probe Function Blocks . . . 166
MC_TouchProbe_PTO: Activate a Trigger Event . . . 167
MC_AbortTrigger_PTO: Abort/Deactivate Function Blocks . . . 169
7.4 Error Handling Function Blocks . . . 170
MC_ReadAxisError_PTO: Get the Axis Control Error . . . 171
MC_Reset_PTO: Reset All Axis-Related Errors . . . 173
7.5 Adding an Administrative Function Block . . . 174
Adding an Administrative Function Block . . . 174
Part III Pulse Width Modulation (PWM) . . . 175
Chapter 8 Introduction . . . 177
Description . . . 178
FreqGen/PWM Naming Convention . . . 180
Synchronization and Enable Functions . . . 181
Chapter 9 Configuration and Programming . . . 183
Configuration . . . 184
PWM_M241: Command a Pulse Width Modulation Signal . . . 187
Programming the PWM Function Block . . . 189
Chapter 10 Data Types . . . 191
FREQGEN_PWM_ERR_TYPE . . . 191
Part IV Frequency Generator (FreqGen) . . . 193
Chapter 11 Introduction . . . 195
Description . . . 196
FreqGen Naming Convention . . . 197
Synchronization and Enable Functions . . . 198

Chapter 12 Configuration and Programming. 199
Configuration 200
FrequencyGenerator_M241: Commanding a Square Wave Signal . . 203
Programming 205
Appendices 207
Appendix A Function and Function Block Representation 209
Differences Between a Function and a Function Block 210
How to Use a Function or a Function Block in IL Language 211
How to Use a Function or a Function Block in ST Language. 215
Glossary 219
Index 223

Safety Information

Important Information

NOTICE

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



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



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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

BEFORE YOU BEGIN

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

⚠ WARNING

UNGUARDED EQUIPMENT

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

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

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

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

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

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

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

START-UP AND TEST

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

⚠ WARNING

EQUIPMENT OPERATION HAZARD

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

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

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

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

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

Before energizing equipment:

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

OPERATION AND ADJUSTMENTS

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

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

About the Book

At a Glance

Document Scope

This documentation acquaints you with the pulse train output (PTO), pulse width modulation (PWM) and frequency generator (FreqGen) functions offered within the Modicon M241 Logic Controller.

This document describes the data types and functions of the M241 PTOPWM Library.

In order to use this manual, you must:

- Have a thorough understanding of the M241, including its design, functionality, and implementation within control systems.
- Be proficient in the use of the following IEC 61131-3 PLC programming languages:
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - Structured Text (ST)
  - Instruction List (IL)
  - Sequential Function Chart (SFC)
  - Continuous Function Chart (CFC)

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.2.

Related Documents

Title of Documentation	Reference Number
Modicon M241 Logic Controller Programming Guide	*EIO0000003059 (ENG)*, *EIO0000003060 (FRE)*, *EIO0000003061 (GER)*, *EIO0000003062 (SPA)*, *EIO0000003063 (ITA)*, *EIO0000003064 (CHS)*

You can download these technical publications and other technical information from our website at https://www.se.com/ww/en/download/ .

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Part I
Introduction

Overview

This part provides an overview description, available modes, functionality and performances of the different functions.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Expert Function Introduction	17
2	Generalities	25

Chapter 1
Expert Function Introduction

Overview

This chapter provides an overview description, functionality, and performances of:

- High Speed Counter (HSC)
- Pulse Train Output (PTO)
- Pulse Width Modulation (PWM)
- Frequency Generator (FreqGen)

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Expert Functions Overview	18
Embedded Expert I/O Assignment	21

Expert Functions Overview

Introduction

The inputs and outputs available on the M241 logic controller can be connected to expert functions.

As of the release of EcoStruxure Machine Expert, any regular I/O not already in use can be configured for use by any of the expert function types, in the same way as fast I/Os.

NOTE:

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

For more details, refer to Embedded Functions Configuration *(see Modicon M241 Logic Controller, Programming Guide)*.

Maximum Number of Expert Functions

The maximum number of expert functions that can be configured depends on:

1. The logic controller reference.
2. The expert function types and number of optional functions *(see Modicon M241 Logic Controller, High Speed Counting, HSC Library Guide)* configured. Refer to Embedded Expert I/O Assignment *(see page 21)*.
3. The number of I/Os that are available.

Maximum number of expert functions by logic controller reference:

Expert Function Type		24 I/O References (TM241•24•)	40 I/O References (TM241•40•)
Total number of HSC functions		14	16
HSC	Simple	14	16
	Main Single Phase	4	
	Main Dual Phase		
	Frequency Meter(1)		
	Period Meter		
PTO			
PWM			
FreqGen			
(1) When the maximum number is configured, only 12 additional HSC Simple functions can be added.			

The maximum number of expert functions possible may be further limited by the number of I/Os used by each expert function.

Example configurations:

- 4 PTO(2) + 14 HSC Simple on 24 I/O controller references
- 4 FreqGen(2) + 16 HSC Simple on 40 I/O controller references
- 4 HSC Main Single Phase + 10 HSC Simple on 24 I/O controller references
- 4 HSC Main Dual Phase + 8 HSC Simple on 40 I/O controller references
- 2 PTO(2) + 2 HSC Main Single Phase + 14 HSC Simple on 40 I/O controller references

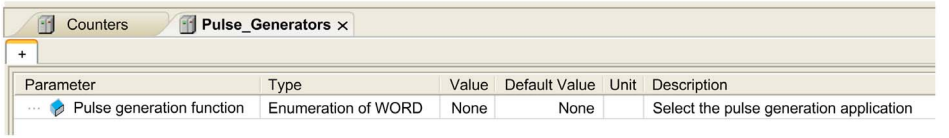
(2) With no optional I/O configured

The performance of the expert function is limited by the I/Os used:

- HSC with fast inputs: 100 kHz/200 kHz
- HSC with regular inputs: 1 kHz

Configuring an Expert Function

To configure an expert function, proceed as follows:

Step	Description
1	Double-click the **Counters** or **Pulse_Generators** node in the **Devices Tree**. **Result:** The **Counters** or **Pulse_Generators** configuration window appears: 
2	Double-click **None** in the **Value** column and choose the expert function type to assign. **Result:** The default configuration of the expert function appears when you click anywhere in the configuration window.
3	Configure the expert function parameters, as described in the following chapters.
4	To configure an additional expert function, click the **+** tab. **NOTE:** If the maximum number of expert functions is already configured, a message appears at the bottom of the configuration window informing you that you can now add only HSC Simple functions.

Regular I/O Configured as Expert Function

When regular I/Os are configured as expert functions, note the following:

- Inputs can be read through memory variables.
- An input cannot be configured as an expert function if it has already been configured as a Run/Stop input.
- An output cannot be configured in an expert function if it has already been configured as an alarm.
- Short-Circuit management applies on the outputs. Status of outputs are available.
- The I/O that are not used by expert functions can be used as any other regular I/O.
- When inputs are used in expert functions (Latch, HSC,…), integrator filter is replaced by anti-bounce filter. Filter value is configured in the configuration screen.

Embedded Expert I/O Assignment

I/O Assignment

The following regular or fast I/Os can be configured for use by expert functions:

	24 I/O References		40 I/O References	
	TM241•24T, TM241•24U	TM241•24R	TM241•40T, TM241•40U	TM241•40R
Inputs	8 fast inputs (I0...I7) 6 regular inputs (I8...I13)		8 fast inputs (I0...I7) 8 regular inputs (I8...I15)	
Outputs	4 fast outputs (Q0...Q3) 4 regular outputs (Q4...Q7)	4 fast outputs (Q0...Q3)	4 fast outputs (Q0...Q3) 4 regular outputs (Q4...Q7)	4 fast outputs (Q0...Q3)

When an I/O has been assigned to an expert function, it is no longer available for selection with other expert functions.

NOTE: All I/Os are by default disabled in the configuration window.

The following table shows the I/Os that can be configured for expert functions:

Expert Function	Name	Input (Fast or Regular)	Output (Fast or Regular)
HSC Simple	Input	M	
HSC Main	Input A	M	
	Input B/EN	C	
	SYNC	C	
	CAP	C	
	Reflex 0		C
	Reflex 1		C
Frequency Meter/Period Meter	Input A	M	
	EN	C	
PWM/FreqGen	Output A		M
	SYNC	C	
	EN	C	
M Mandatory **C** Optionally configurable			

Expert Function	Name	Input (Fast or Regular)	Output (Fast or Regular)
PTO	Output A/CW/Pulse		M
	Output B/CCW/Dir		C
	REF (Origin)	C	
	INDEX (Proximity)	C	
	PROBE	C	
M Mandatory **C** Optionally configurable			

Using Regular I/O with Expert Functions

Expert function I/O within regular I/O:

- Inputs can be read through standard memory variables even if configured as expert functions.
- All I/Os that are not used by expert functions can be used as regular I/Os.
- An I/O can only be used by one expert function; once configured, the I/O is no longer available for other expert functions.
- If no more fast I/Os are available, a regular I/O can be configured instead. In this case, however, the maximum frequency of the expert function is limited to 1 kHz.
- You cannot configure an input in an expert function and use it as a Run/Stop, Event, or Latch input at a same time.
- An output cannot be configured in an expert function if it has already been configured as an alarm.
- Short-circuit management still applies on all outputs. Status of outputs are available. For more information, refer to Output Management.
- When inputs are used in expert functions (PTO, HSC,…), the integrator filter is replaced by an anti-bounce filter *(see page 27)*. The filter value is configured in the configuration window.

For more details, refer to Embedded Functions Configuration.

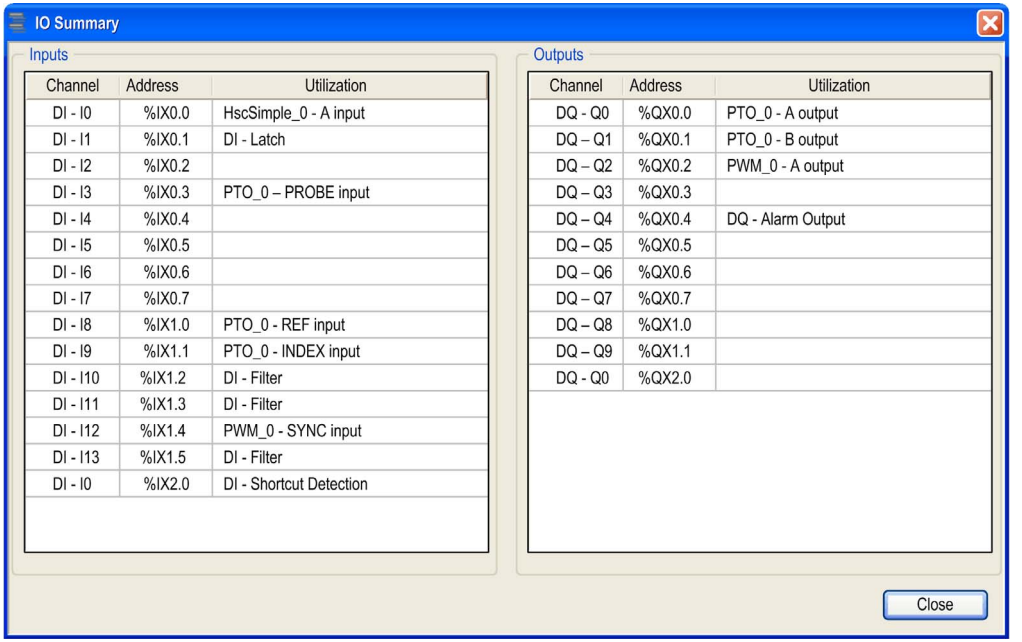
I/O Summary

The **IO Summary** window displays the I/Os used by the expert functions.

To display the **IO Summary** window:

Step	Action
1	In the **Devices tree** tab, right-click the **MyController** node and choose **IO Summary**.

Example of IO Summary window:



Chapter 2
Generalities

Overview

This chapter provides general information of the Frequency Generator (FreqGen), Pulse Train Output (PTO), and Pulse Width Modulation (PWM) functions.

The functions provide simple, yet powerful solutions for your application. In particular, they are useful for controlling movement. However, the use and application of the information contained herein require expertise in the design and programming of automated control systems. Only you, the user, machine builder or integrator, can be aware of all the conditions and factors present during installation and setup, operation, and maintenance of the machine or related processes, and can therefore determine the automation and associated equipment and the related safeties and interlocks which can be effectively and properly used. When selecting automation and control equipment, and any other related equipment or software, for a particular application, you must also consider any applicable local, regional, or national standards and/or regulations.

⚠ WARNING

REGULATORY INCOMPATIBILITY

Ensure that all equipment applied and systems designed comply with all applicable local, regional, and national regulations and standards.

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Dedicated Features	27
General Information on Function Block Management	28

Dedicated Features

Bounce Filter

This table shows the maximum counter frequencies determined by the filtering values used to reduce the bounce effect on the input:

Input	Bounce Filter Value (ms)	Maximum Counter Frequency Expert	Maximum Counter Frequency Regular
A B	0.000	200 kHz	1 kHz
	0.001	200 kHz	1 kHz
	0.002	200 kHz	1 kHz
	0.005	100 kHz	1 kHz
	0.01	50 kHz	1 kHz
	0.05	25 kHz	1 kHz
	0.1	5 kHz	1 kHz
	0.5	1 kHz	1 kHz
	1	500 Hz	500 Hz
	5	100 Hz	100 Hz
A is the counting input of the counter. B is the counting input of the dual phase counter.			

Dedicated Outputs

Outputs used by the high speed expert functions can only be accessed through the function block. They cannot be read or written directly within the application.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use the same function block instance in different program tasks.
- Do not modify or otherwise 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.

General Information on Function Block Management

Management of Input Variables

The variables are used with the rising edge of the `Execute` input. To modify any variable, it is necessary to change the input variables and to trigger the function block again.

The function blocks managed by an `Enable` input are executed when this input is true. The values of the function block inputs can be modified continuously, and the outputs are updated continuously. When the `Enable` input is false, the function block execution is terminated and its outputs are reseted.

According to IEC 61131-3, if any variable of a function block input is missing (= open), then the value from the previous invocation of this instance will be used. In the first invocation the initial value is applied.

Management of Output Variables

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

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

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

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

The `Done`, `Error`, `ErrID`, and `CommandAborted` outputs are set or reset with the falling edge of `Execute` input:

- reset if the function block execution is finished.
- set for at least one task cycle if the function block execution is not finished.

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

Error Handling

All blocks have two outputs that can report error detection during the execution of the function block:

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

When an `Error` occurs, the other output signals, such as `Done` are reset.

Part II
Pulse Train Output (PTO)

Overview

This part describes the `Pulse Train Output` function.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	Overview	31
4	Configuration	35
5	Data Unit Types	79
6	Motion Function Blocks	89
7	Administrative Function Blocks	147

Chapter 3
Overview

Pulse Train Output (PTO)

Introduction

The PTO function provides up to four 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 up to four independent linear single-axis stepper or servo drives in open loop mode (for example, with Lexium 28).

The PTO function does not have any position feedback information from the process.

The PTO function can be configured on any output channel of the logic controller not already configured for use by another expert function.

Each PTO channel can use up to:

- Six inputs, if optional interface signals for homing (ref/index), event (probe), limits (limP, limN), or drive interface (driveReady) are used,
- Three physical outputs, if optional drive interface signal is used (driveEnable).

Automatic origin offset and backlash compensation are also managed to improve positioning accuracy. Diagnostics are available for status monitoring, providing comprehensive and quick troubleshooting.

Supported Functions

The four PTO channels support the following functions:

- Four output modes, including quadrature
- Single axis moves (velocity and position)
- Relative and absolute positioning
- Automatic trapezoidal and S-curve acceleration and deceleration
- Homing (seven modes with offset compensation)
- Dynamic acceleration, deceleration, velocity, and position modification
- Switch from velocity to position mode and vice versa
- Move queuing (buffer of one move)
- Position capture and move trigger on event (using probe input)
- Backlash compensation (in quadrature mode)
- Limits (hardware and software)
- Diagnostics

PTO Function Blocks

The PTO function is programmed in EcoStruxure Machine Expert using the following function blocks, available in the **M241 PTOPWM** library:

Category	Subcategory	Function Block
Motion (single axis)	Power	`MC_Power_PTO` *(see page 104)*
	Discrete	`MC_MoveAbsolute_PTO` *(see page 122)*
		`MC_MoveRelative_PTO` *(see page 115)*
		`MC_Halt_PTO` *(see page 141)*
		`MC_SetPosition_PTO` *(see page 133)*
	Continuous	`MC_MoveVelocity_PTO` *(see page 109)*
	Homing	`MC_Home_PTO` *(see page 128)*
	Stopping	`MC_Stop_PTO` *(see page 136)*
Administrative	Status	`MC_ReadActualVelocity_PTO` *(see page 149)*
		`MC_ReadActualPosition_PTO` *(see page 151)*
		`MC_ReadStatus_PTO` *(see page 153)*
		`MC_ReadMotionState_PTO` *(see page 155)*
	Parameters	`MC_ReadParameter_PTO` *(see page 158)*
		`MC_WriteParameter_PTO` *(see page 160)*
		`MC_ReadBoolParameter_PTO` *(see page 162)*
		`MC_WriteBoolParameter_PTO` *(see page 164)*
	Probe	`MC_TouchProbe_PTO` *(see page 167)*
		`MC_AbortTrigger_PTO` *(see page 169)*
	Error handling	`MC_ReadAxisError_PTO` *(see page 171)*
		`MC_Reset_PTO` *(see page 173)*

NOTE: The motion function blocks act on the position of the axis according to the motion state diagram *(see page 91)*. The administrative function blocks do not influence the motion state.

NOTE: `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

The PTO function has the following characteristics:

Characteristic	Value
Number of channels	4
Number of axes	1 per channel
Position range	-2,147,483,648...2,147,483,647 (32 bits)
Minimum velocity	1 Hz
Maximum velocity	For a 40/60 duty cycle and max. 200 mA: ● Fast outputs (Q0...Q3): 100 kHz ● Regular outputs (Q4...Q7): 1 kHz
Minimum step	1 Hz
Acceleration / deceleration min	1 Hz/ms
Acceleration / deceleration max	100,000 Hz/ms
Start move IEC	300 µs + 1 pulse output time
Start move on probe event	
Change move parameter	
Accuracy on velocity	0.5 %
Accuracy in position	Depends on the pulse output time

Chapter 4
Configuration

Overview

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

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
4.1	Configuration	36
4.2	Home Modes	56

Section 4.1
Configuration

Overview

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

What Is in This Section?

This section contains the following topics:

Topic	Page
PTO Configuration	37
Pulse Output Modes	42
Acceleration / Deceleration Ramp	44
Probe Event	48
Backlash Compensation (Only Available in Quadrature Mode)	51
Positioning Limits	53

PTO Configuration

Hardware Configuration

There are up to six inputs for a PTO channel:

- Three physical inputs are associated to the PTO function through configuration and are taken into account immediately on a rising edge on the input:
  - REF input
  - INDEX input
  - PROBE input
- Three inputs are associated with the `MC_Power_PTO` function block. They have no fixed assignment (they are freely assigned; that is, they are not configured in the configuration screen), and are read as any other input:
  - Drive ready input
  - Limit positive input
  - Limit negative input

NOTE: These inputs are managed as any other input, but are used by the PTO controller when used by the `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 outputs for a PTO channel:

- Either one physical output to manage pulse only, or two physical outputs to manage both pulse and direction; they must be enabled by configuration:
  - A / CW / Pulse
  - B / CCW / Direction
- The other output, `DriveEnable`, is used through the `MC_Power_PTO` function block.

Configuration Window Description

The figure provides an example of a configuration window on channel **PTO_0**:

PTO_0 (PTO) +

Parameter	Type	Value	Default Value	Unit	Description
Pulse generation function	Enumeration of WORD	PTO	None		Select the pulse gen
General					
Instance name	STRING	'PTO_0'	''		Name the Axis contr
Output Mode	Enumeration of BYTE	Quadrature	A ClockWise / B CounterClockWise		Select the pulse out
A output location	Enumeration of SINT	Q0	Disabled		Select the PLC outp
B output location	Enumeration of SINT	Q1	Disabled		Select the PLC outp
Mechanics					
Backlash Compensation	DWORD(0..255)	0	0		Amount of motion ne
Position Limits					
Software Limits					
Enable Software Limits	Enumeration of BYTE	Enabled	Enabled		Select whether or no
SW Low Limit	DINT(–2147483648...21474...	–2147483648	–2147483648		Set the software limi
SW High Limit	DINT(–2147483648...21474...	2147483647	2147483647		Set the software limi
Motion					
General					
Maximum Velocity	DWORD(0..100000)	100000	100000	Hz	Set the pulse output
Start Velocity	DWORD(0...100000)	0	0	Hz	Set the pulse output
Stop Velocity	DWORD(0...100000)	0	0	Hz	Set the pulse output
Acc./Dec. Unit	Enumeration of BYTE	Hz/ms	Hz/ms		Set acceleration/dec
Maximum Acceleration	DWORD(1...100000)	100000	100000		Set the acceleration
Maximum Deceleration	DWORD(1...100000)	100000	100000		Set the deceleration
Fast Stop					
Fast Stop Deceleration	DWORD(1...100000)	5000	5000		Set the deceleration
Homing					
REF input					
Location	Enumeration of SINT	I8	Disabled		Select the PLC input
Bounce filter	Enumeration of BYTE	0.005	0.005	ms	Set the filtering valu
Type	Enumeration of WORD	Normally opened	Normally opened		Select whether the s
INDEX input					
Location	Enumeration of SINT	I9	Disabled		Select the PLC inpu
Bounce filter	Enumeration of BYTE	0.005	0.005	ms	Set the filtering valu
Type	Enumeration of WORD	Normally opened	Normally opened		Select whether the s
Registration					
PROBE input					
Location	Enumeration of SINT	I10	Disabled		Select the PLC inpu
Bounce filter	Enumeration of BYTE	0.005	0.005	ms	Set the filtering valu

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

Parameter		Value	Default	Description
General	Instance name	-	PTO_0...PTO_3	Name of the axis controlled by this PTO channel. It is used as input of the PTO function blocks.
	Output Mode *(see page 42)*	A ClockWise / B CounterClockWise A Pulse / B Direction A Pulse Quadrature	A ClockWise / B CounterClockWise	Select the pulse output mode.
	A output location	Disabled Q0...Q3 (fast outputs) Q4...Q7 (regular outputs)(1)	Disabled	Select the controller output used for the signal A.
	B output location	Disabled Q0...Q3 (fast outputs) Q4...Q7 (regular outputs)(1)	Disabled	Select the controller output used for the signal B.
Mechanics	Backlash Compensation *(see page 51)*	0...255	0	In quadrature mode, amount of motion needed to compensate the mechanical clearance when movement is reversed.
Position Limits / Software Limits	Enable Software Limits *(see page 54)*	Enabled Disabled	Enabled	Select whether to use the software limits.
	SW 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.
	SW 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.

(1) Not available for M241 Logic Controller references with relay outputs.

Parameter		Value	Default	Description
Motion / General	Maximum Velocity	0...100000 (fast outputs) 0...1000 (regular outputs)	100000 (fast outputs) 1000 (regular outputs)	Set the pulse output maximum velocity (in Hz).
	Start Velocity *(see page 44)*	Start Velocity...100000 (fast outputs) Start Velocity...1000 (regular outputs)	0	Set the pulse output start velocity (in Hz). 0 if not used.
	Stop Velocity *(see page 44)*	0...100000 (fast outputs) 0...1,000 (regular outputs)	0	Set the pulse output stop velocity (in Hz). 0 if not used.
	Acc./Dec. Unit *(see page 45)*	Hz/ms ms	Hz/ms	Set acceleration/deceleration as rates (Hz/ms) or as time constants from 0 to **Maximum Velocity** (ms).
	Maximum Acceleration	1...100000	100000	Set the acceleration maximum value (in **Acc./Dec. Unit**).
	Maximum Deceleration	1...100000	100000	Set the deceleration maximum value (in **Acc./Dec. Unit**).
Motion / Fast Stop	Fast Stop Deceleration	1...100000	5000	Set the deceleration value in case an error is detected (in **Acc./Dec. Unit**)
Homing / REF input	Location	Disabled I0...I7 (fast inputs) I8...I15 (regular inputs)	Disabled	Select the controller input used for the REF signal *(see page 56)*.
	Bounce filter	0.000 0.001 0.002 0.005 0.010 0.05 0.1 0.5 1 5	0.005	Set the filtering value to reduce the bounce effect on the REF input (in ms).
	Type	Normally opened Normally closed	Normally opened	Select whether the switch contact default state is open or closed.
(1) Not available for M241 Logic Controller references with relay outputs.				

Parameter		Value	Default	Description
Homing / INDEX input	Location	Disabled I0...I7 (fast inputs) I8...I15 (regular inputs)	Disabled	Select the controller input used for the INDEX signal *(see page 56)*.
	Bounce filter	0.000 0.001 0.002 0.005 0.010 0.05 0.1 0.5 1 5	0.005	Set the filtering value to reduce the bounce effect on the INDEX input (in ms).
	Type	Normally opened Normally closed	Normally opened	Select whether the switch contact default state is open or closed.
Registration / PROBE input	Location	Disabled I0...I7 (fast inputs) I8...I15 (regular inputs)	Disabled	Select the controller input used for the PROBE signal *(see page 48)*.
	Bounce filter	0.000 0.001 0.002 0.005 0.010 0.05 0.1 0.5 1 5	0.005	Set the filtering value to reduce the bounce effect on the PROBE input (in ms).
(1) Not available for M241 Logic Controller references with relay outputs.				

Pulse Output Modes

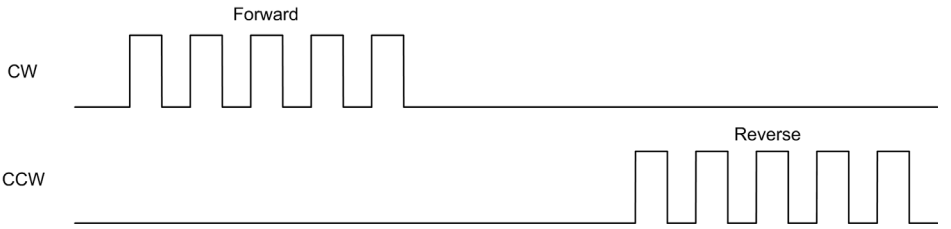
Overview

There are four possible output modes:

- A ClockWise / B CounterClockwise
- A Pulse
- A Pulse / B direction
- Quadrature

A ClockWise (CW) / B CounterClockwise (CCW) Mode

This mode generates a signal that defines the motor operating speed and direction. This signal is implemented either on the PTO output A or on PTO output B depending on the motor rotation direction.

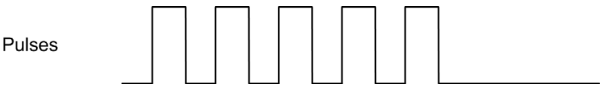


A Pulse Mode

This mode generates one signal on the PTO outputs:

- Output A: pulse which provides the motor operating speed.

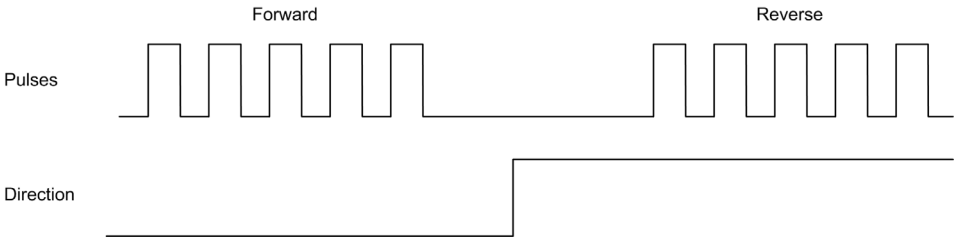
NOTE: The corresponding function block generates an "Invalid Direction" error if you specify a negative direction value.



A Pulse / B Direction Mode

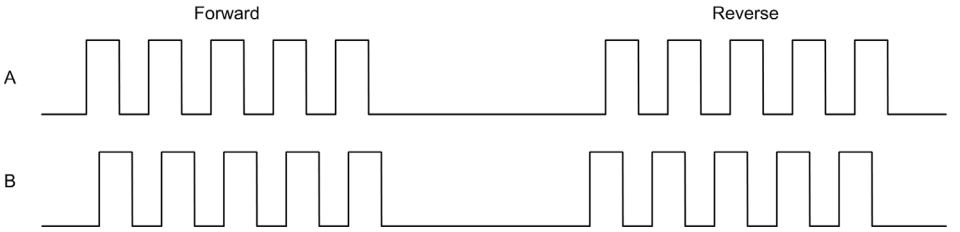
This mode generates two signals on the PTO outputs:

- Output A: pulse which provides the motor operating speed.
- Output B: direction which provides the motor rotation direction.



Quadrature Mode

This mode generates two signals in quadrature phase on the PTO outputs (the phase sign depends on motor direction).



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` *(see page 85)*.

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` *(see page 85)*.

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.

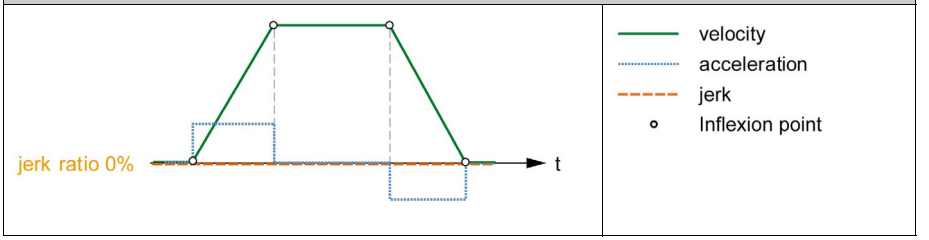
Acceleration / Deceleration Ramp with a Trapezoidal Profile

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.

Expressed in ms, they represent the time to go from 0 to **Maximum velocity**.

`JerkRatio` 0%: Constant acceleration / deceleration.

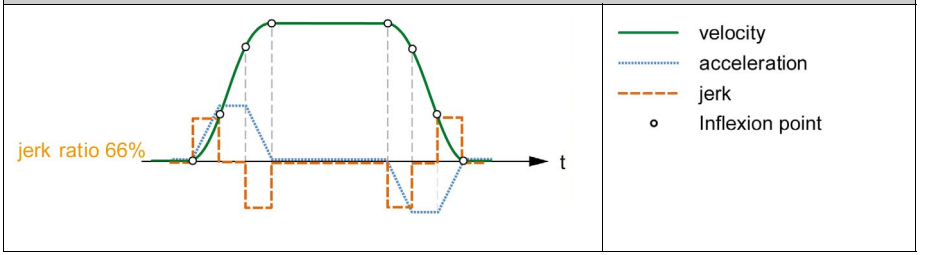


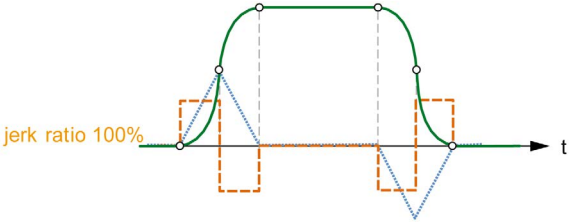
Acceleration / Deceleration Ramp with an S-curve Profile

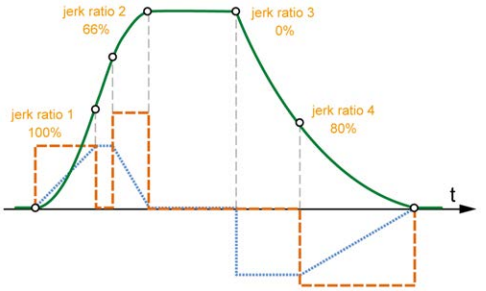
When the jerk ratio parameter is greater than 0, the acceleration / deceleration ramp has an S-curve profile.

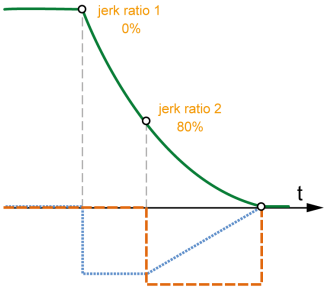
The S-curve ramp is used in applications controlling high inertia, or in those that manipulate fragile objects or liquids. The S-curve ramp enables a smoother and progressive acceleration / deceleration, as demonstrated in the following graphics:

`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.	
	velocity acceleration jerk Inflexion point

Example using 4 `JerkRatio` phases with a variable length.	
	velocity acceleration jerk Inflexion point

Example using 2 `JerkRatio` phases with a variable length.	
	velocity acceleration jerk Inflexion point

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

Affect of the S-Curve Ramp on Acceleration / Deceleration

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

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

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

NOTE: If the `JerkRatio` parameter value is invalid, the value is re-calculated to respect the `MaxAccelerationAppl` and `MaxDecelerationAppl` parameters.
`JerkRatio` is invalid when:

- its value is greater than 100. In this case, a `Jerkratio` of 100 is applied.
- its value is less than 0. In this case, a `Jerkratio` of 0 is applied.

Probe Event

Description

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

The Probe input is used as an event to:

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

Both functions can be active at the same time, that is, the same event captures the position and start a motion function block *(see page 89)*.

The Probe input event can be defined to be enabled within a predefined window that is demarcated by position limits (refer to `MC_TouchProbe_PTO` *(see page 167)*.

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

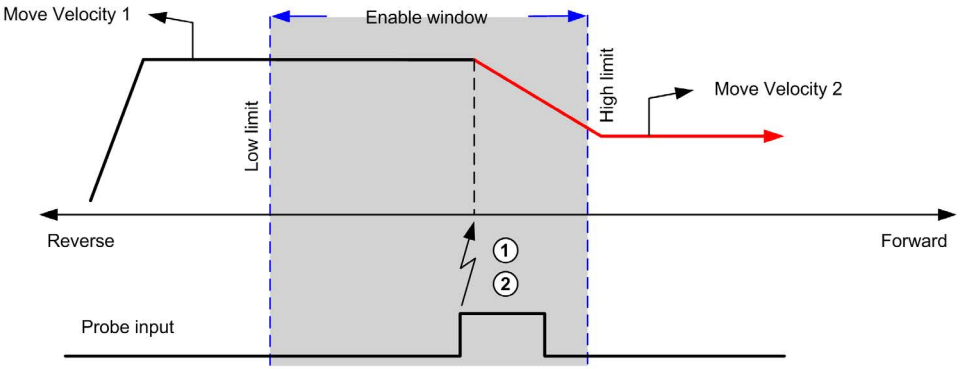
Position Capture

The position captured is available in `MC_TouchProbe_PTO.RecordedPosition`.

Motion Trigger

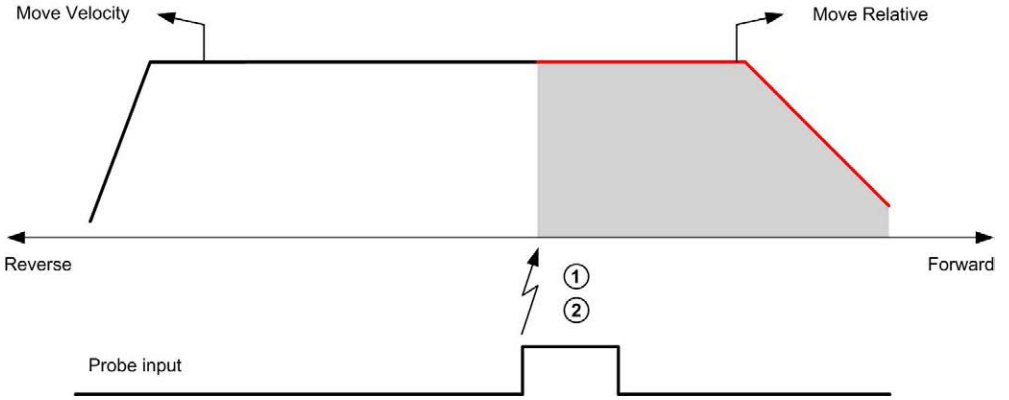
The `BufferMode` input of a motion function block must be set to `seTrigger`.

This example illustrates a change target velocity with enable window:



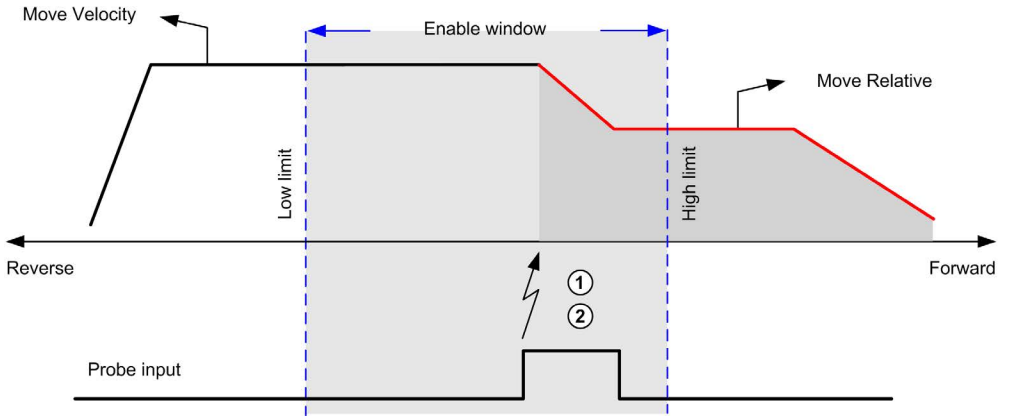
1 Capture the position counter value
2 Trigger `Move Velocity` function block

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



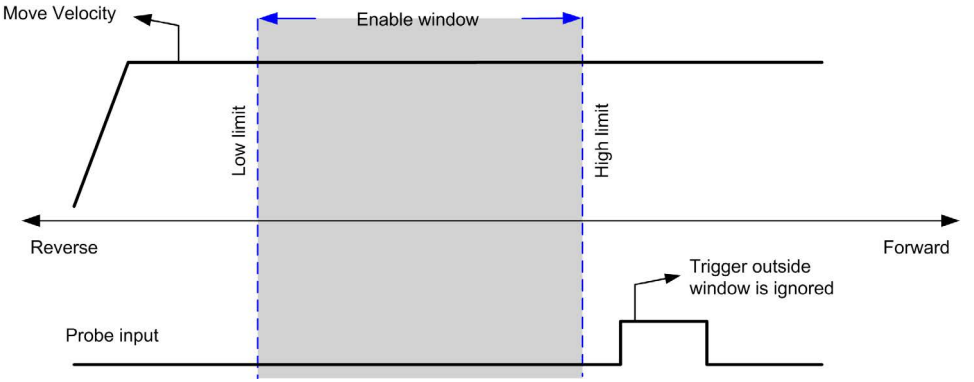
1 Capture the position counter value
2 Trigger `Move Relative` function block

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



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

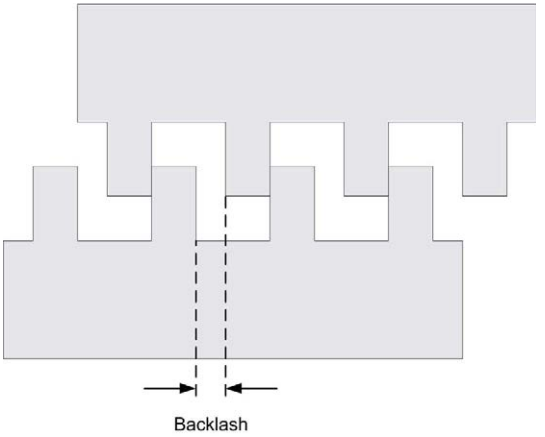
This example illustrates a trigger event out of enable window:



Backlash Compensation (Only Available in Quadrature Mode)

Description

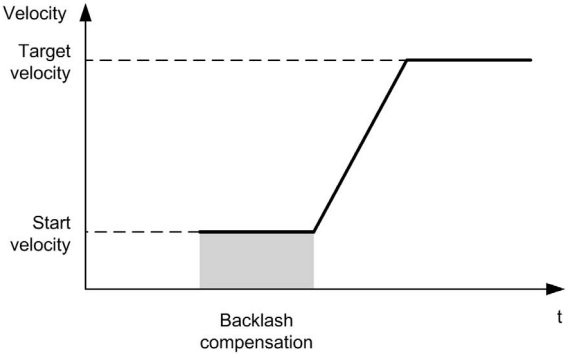
The **Backlash Compensation** parameter is defined as the amount of motion needed to compensate for the mechanical clearance in gears, when movement is reversed and the axis is homed:



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

Backlash compensation is set in number of pulses (0...255, default value is 0). When set, at each direction reversal, the specified number of pulses is first output at start velocity, and then the programmed movement is executed. The backlash compensation pulses are not added to the position counter.

This figure illustrates the backlash compensation:



NOTE:

- Before the initial movement is started, the function cannot determine the amount of backlash to compensate for. Therefore, the backlash compensation is only active after a homing is successfully performed. If the homing is performed without movement, it is assumed that the initial movement applies no compensation, and the compensation is applied at the first direction reversal.
- Once started, the compensation pulses are output until completion, even if an aborting command is received in the meantime. In this case, the aborting command is buffered and will start as soon as compensation pulses are output. No additional buffered command is accepted in this case.
- If the axis is stopped by an error detected before all the compensation pulses are output, the backlash compensation is reset. A new homing procedure is needed to reinitialize the backlash compensation.
- Backlash timeout of 80 s: The system does not accept to configure a movement of more than 80 s. So if a backlash is configured, it may for example not be more than 80 pulses to 1 Hz. The error detected in case of this timeout is "Internal error" (code 1000).

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

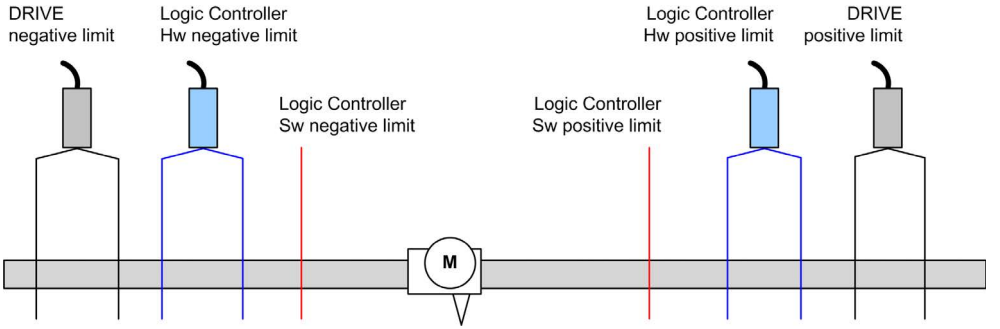
⚠ WARNING

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.

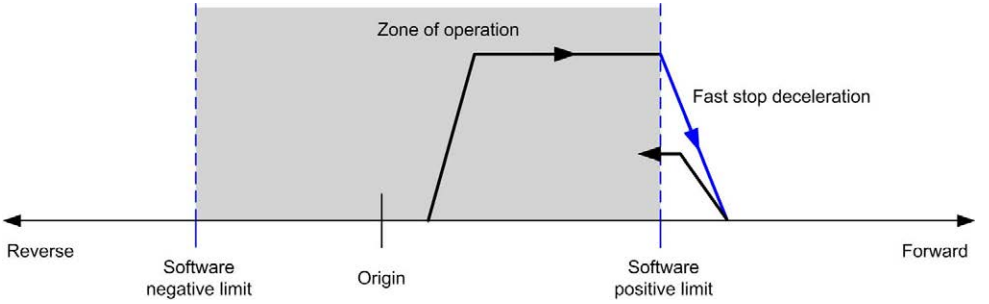
The figure illustrates hardware and software limit switches:



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 `ErrorId` 1002 to 1005 (`PTO_ERROR` *(see page 86)*),
- the function block under execution detects the error state,
- status bits on other applicable function blocks are set to `CommandAborted`.

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



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_WriteParameter_PTO` *(see page 160)* and PTO_PARAMETER *(see page 85)*).

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 128)*).

NOTE: An error is only detected when the software limit is physically reached, not at the initiation of the movement.

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` input bits. 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 4.2
Home Modes

Overview

This section describes the PTO home modes.

What Is in This Section?

This section contains the following topics:

Topic	Page
Homing Modes	57
Position Setting	60
Long Reference	61
Long Reference & Index	63
Short Reference Reversal	65
Short Reference No Reversal	67
Short Reference & Index Outside	69
Short Reference & Index Inside	73
Home Offset	77

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 M241 PTO channels provide several standard homing movement types:

- position setting *(see page 60)*,
- long reference *(see page 61)*,
- long reference and index *(see page 63)*,
- short reference reversal *(see page 65)*,
- short reference no reversal *(see page 67)*,
- short reference and index outside *(see page 69)*,
- short reference and index inside *(see page 73)*.

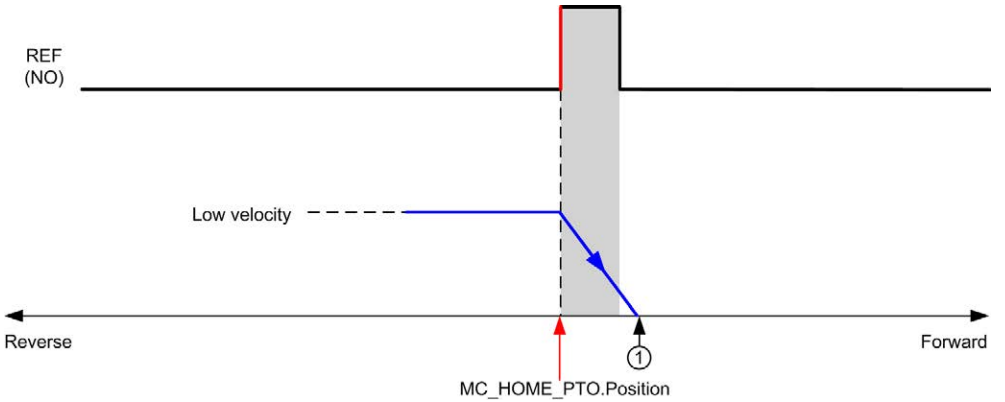
A homing movement must be terminated without interruption for the new reference point to be valid. If the reference movement is interrupted, it needs to be started again.

Refer to `MC_Home_PTO` *(see page 128)* and PTO_HOMING_MODE *(see page 84)*.

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.

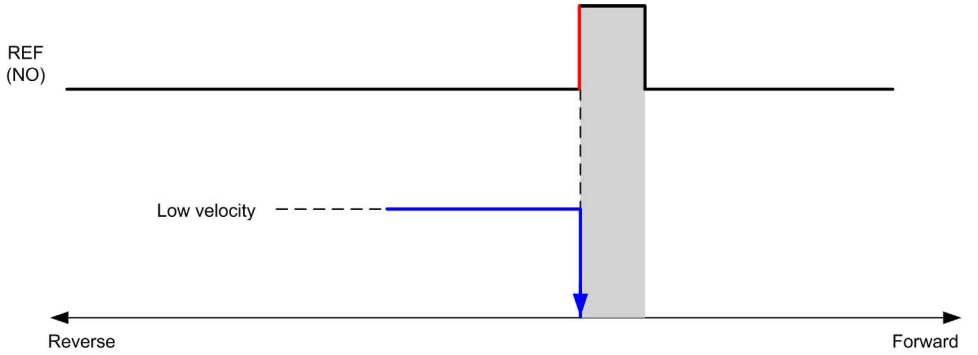
The actual position of the axis at the end of the motion sequence may therefore differ from the position parameter set on the function block:



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 53)* and `MC_Power_PTO` *(see page 104)*). 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

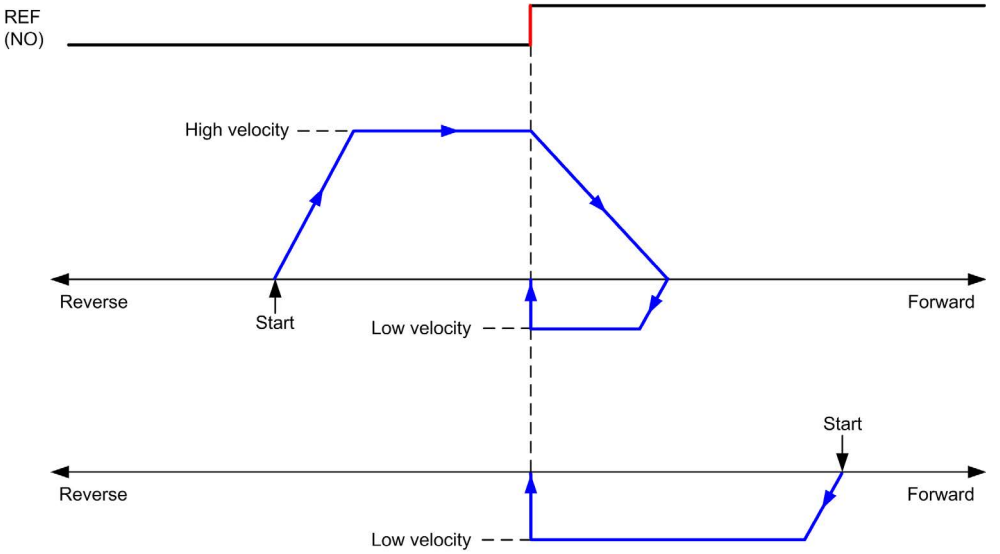
In the case of position setting, the current position is set to the specified position value. No move is performed.

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:

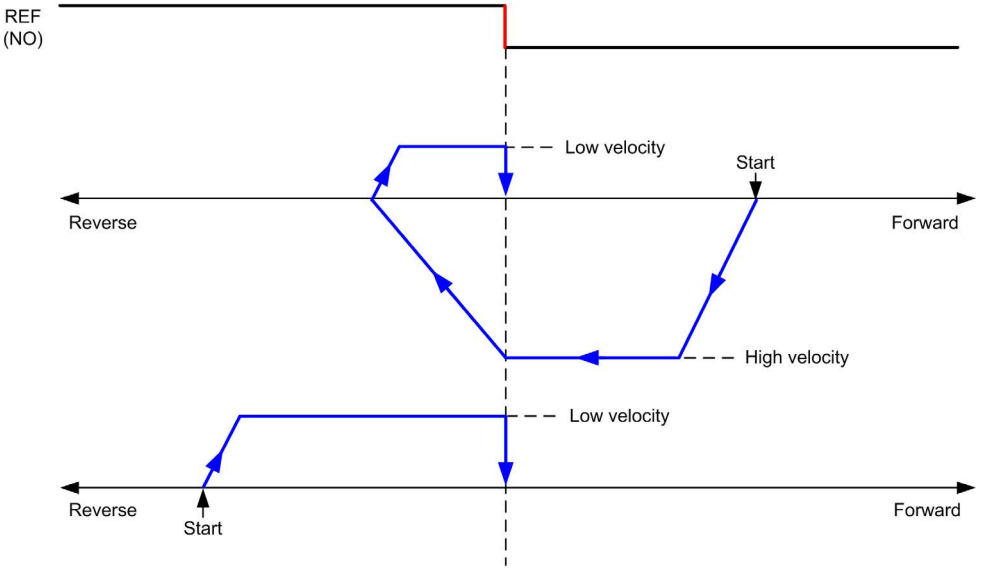


REF (NO) Reference point (Normally Open)

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:



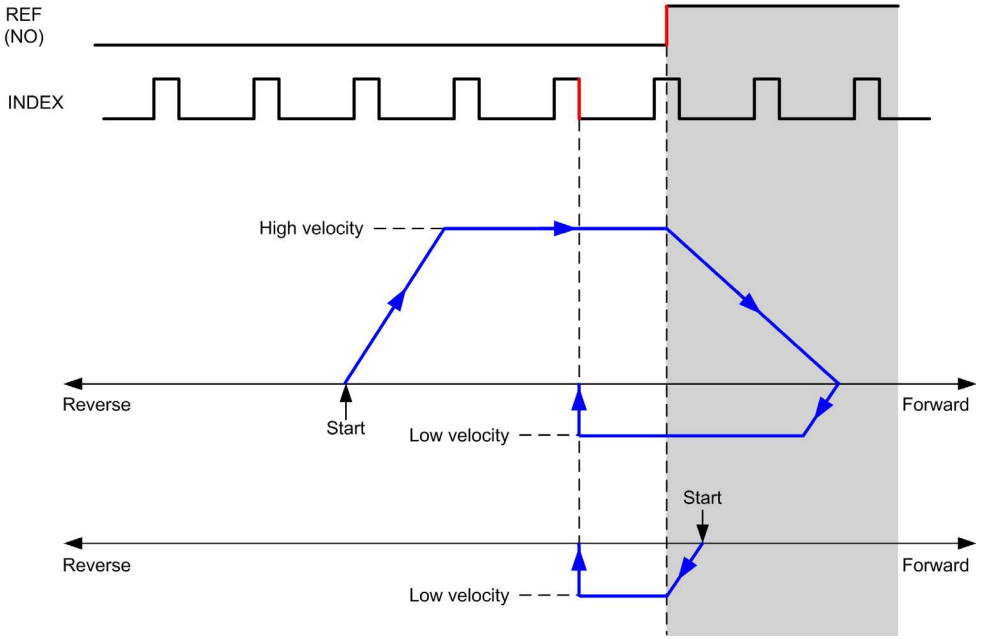
REF (NO) Reference point (Normally Open)

Long Reference & Index

Long Reference & Index: Positive Direction

Homes to the first index, after the reference switch falling edge in reverse direction.

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

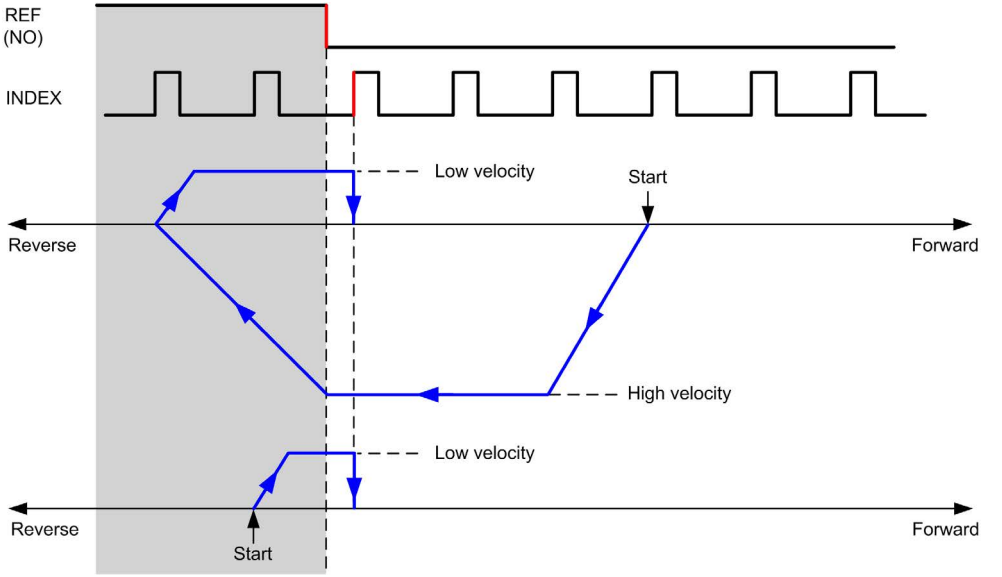


REF (NO) Reference point (Normally Open)

Long Reference & Index: Negative Direction

Homes to the first index, after the reference switch falling edge in forward direction.

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



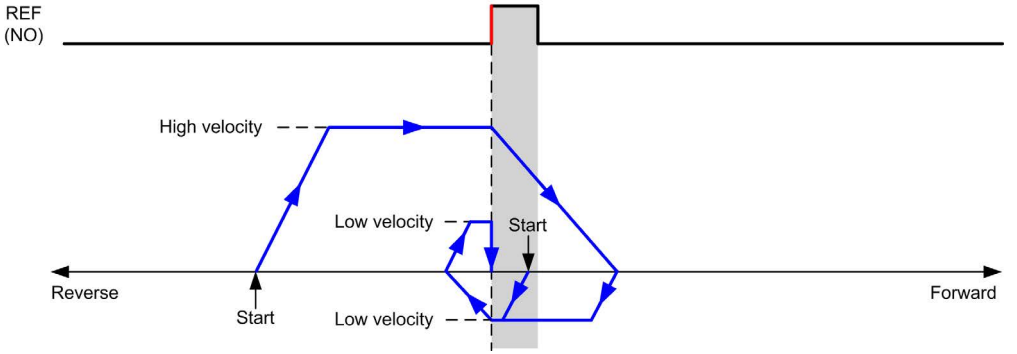
REF (NO) Reference point (Normally Open)

Short Reference Reversal

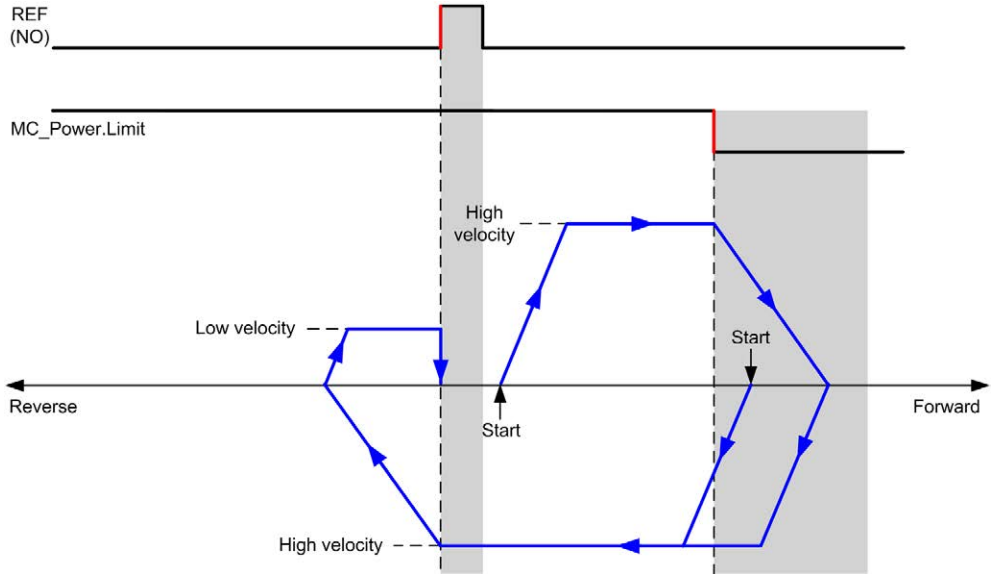
Short Reference Reversal: Positive Direction

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)

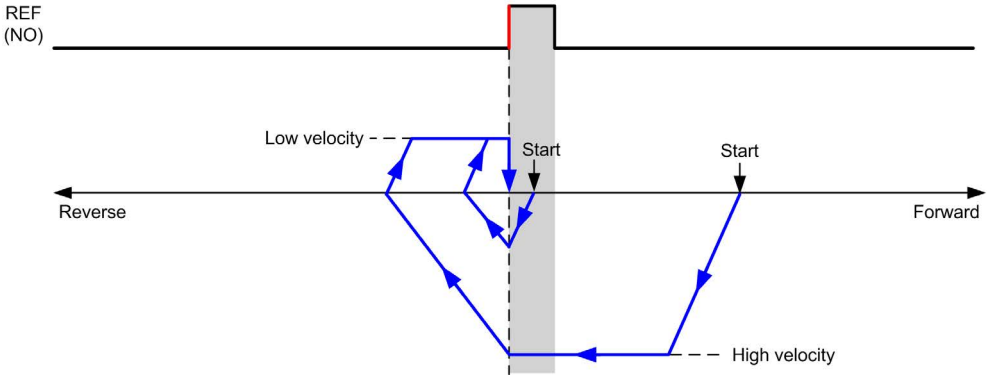


REF (NO) Reference point (Normally Open)

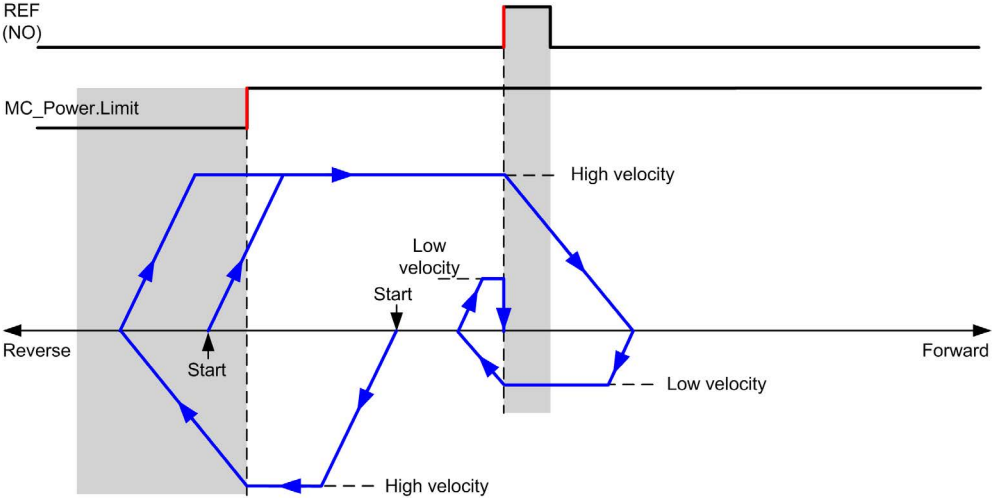
Short Reference Reversal: Negative Direction

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)

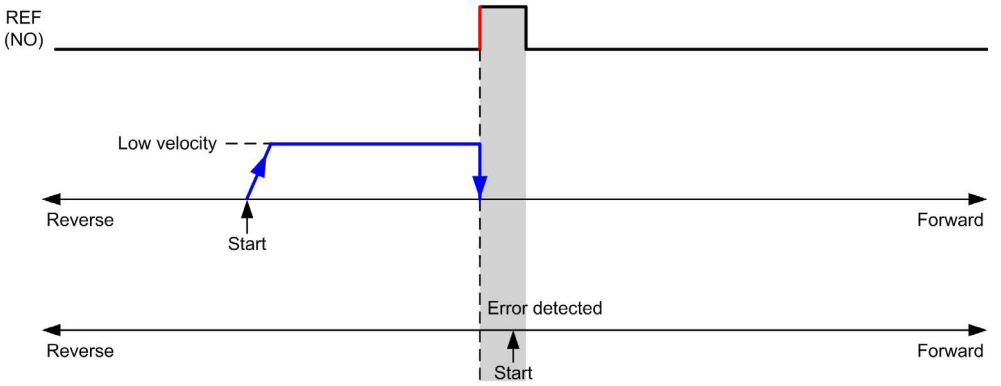


REF (NO) Reference point (Normally Open)

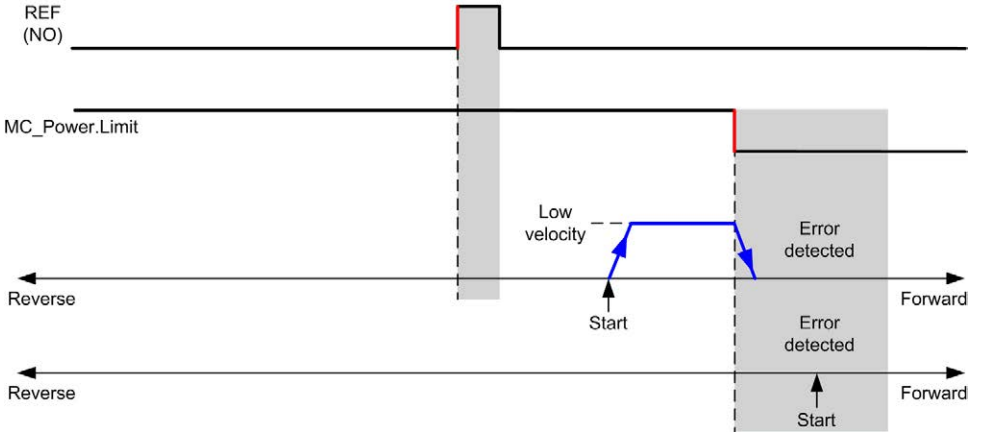
Short Reference No Reversal

Short Reference No Reversal: Positive Direction

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



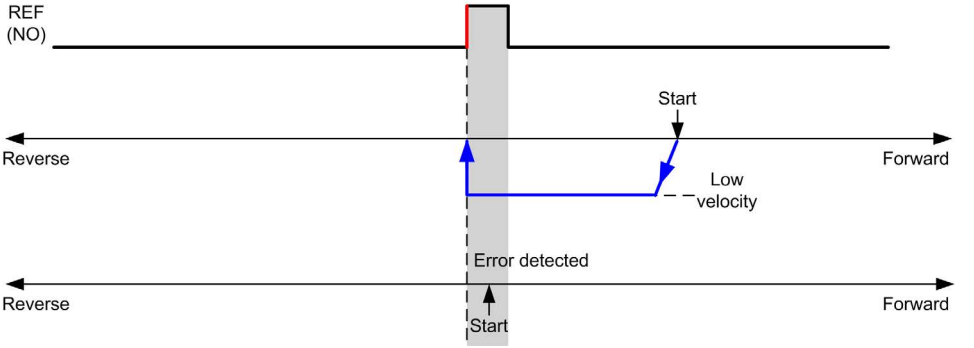
REF (NO) Reference point (Normally Open)



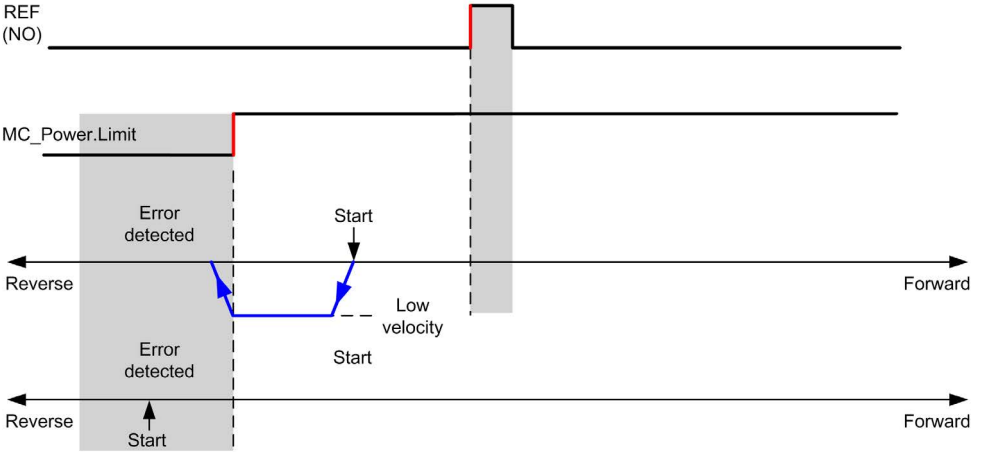
REF (NO) Reference point (Normally Open)

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)



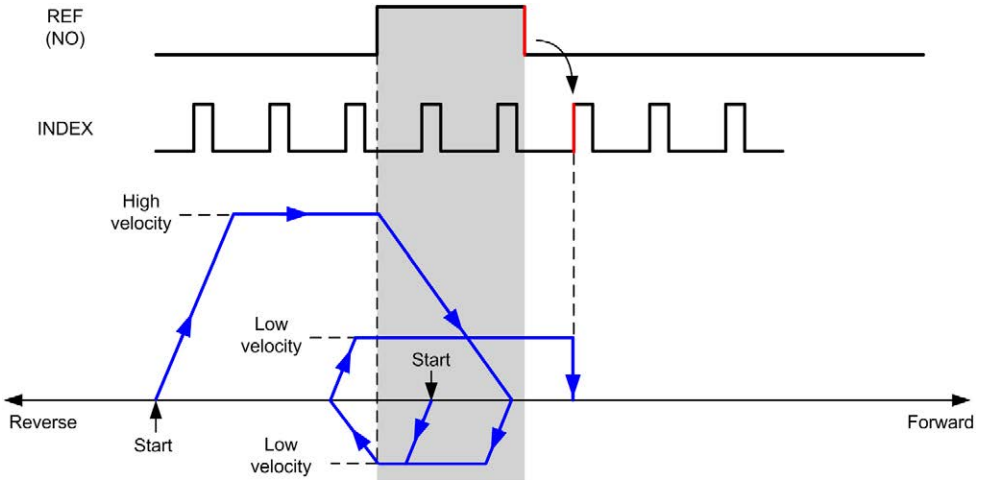
REF (NO) Reference point (Normally Open)

Short Reference & Index Outside

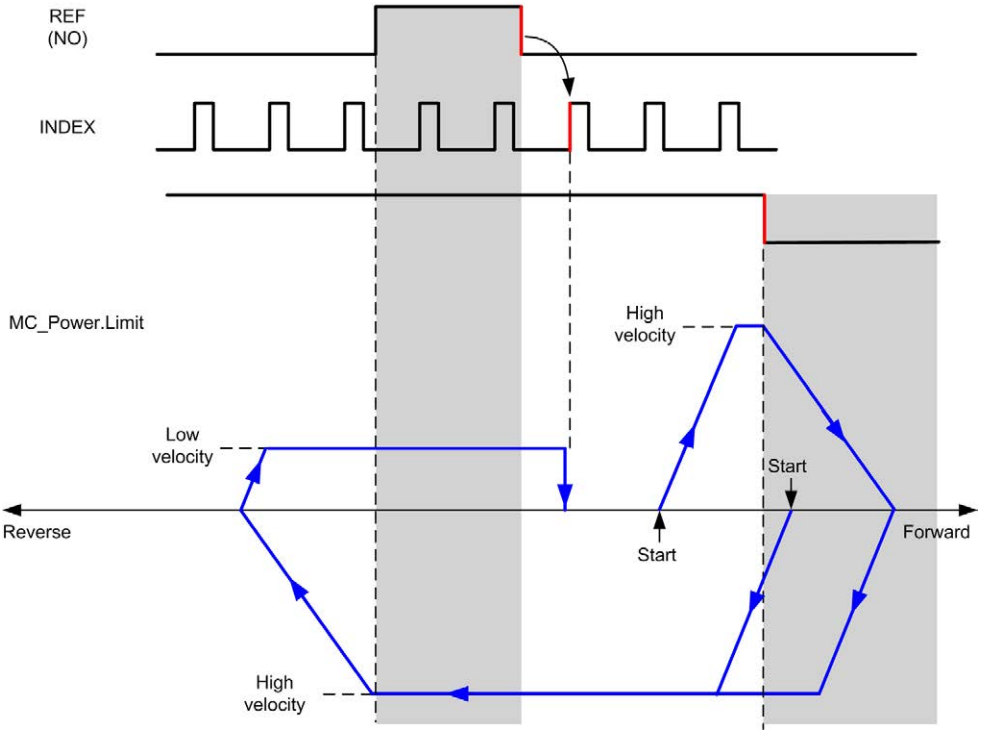
Short Reference & Index Outside: Positive Direction

Homes to the first index, after the reference switch transitions on and off in forward direction.

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



REF (NO) Reference point (Normally Open)

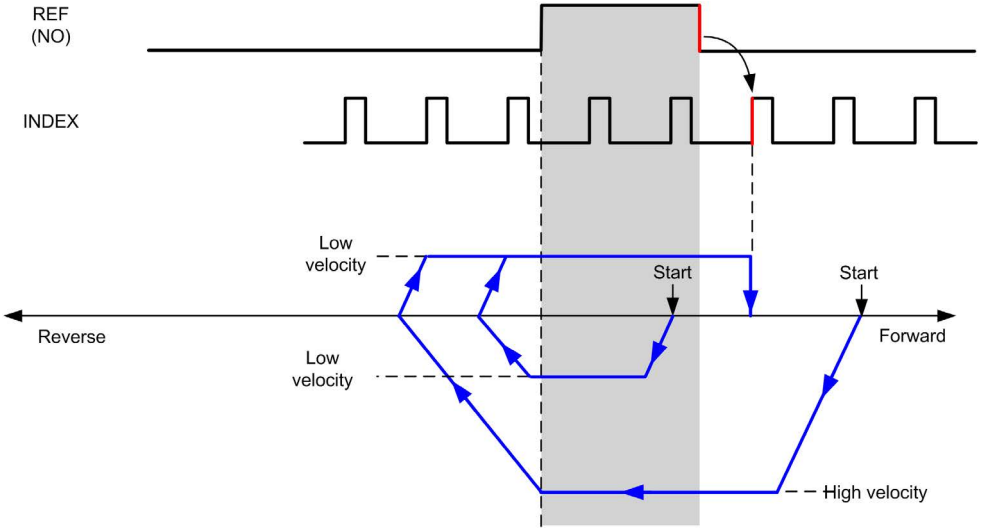


REF (NO) Reference point (Normally Open)

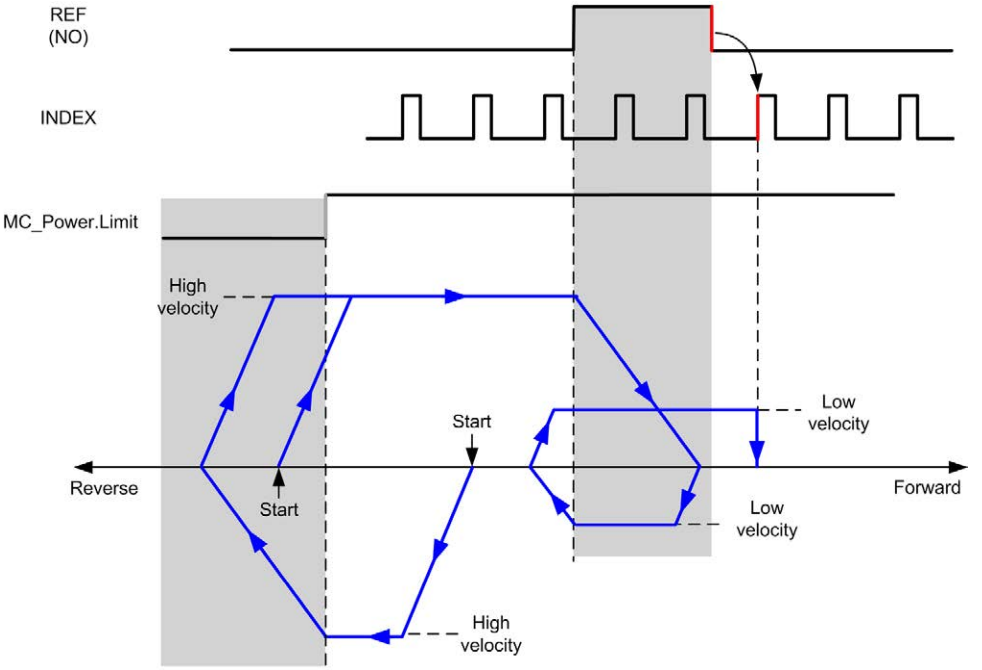
Short Reference & Index Outside: Negative Direction

Homes to the first index, after the reference switch transitions on and off in forward direction.

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



REF (NO) Reference point (Normally Open)



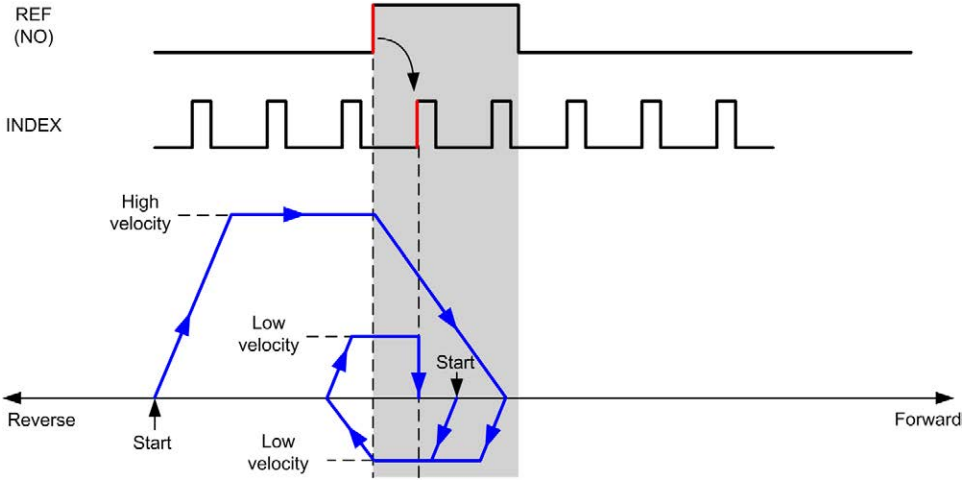
REF (NO) Reference point (Normally Open)

Short Reference & Index Inside

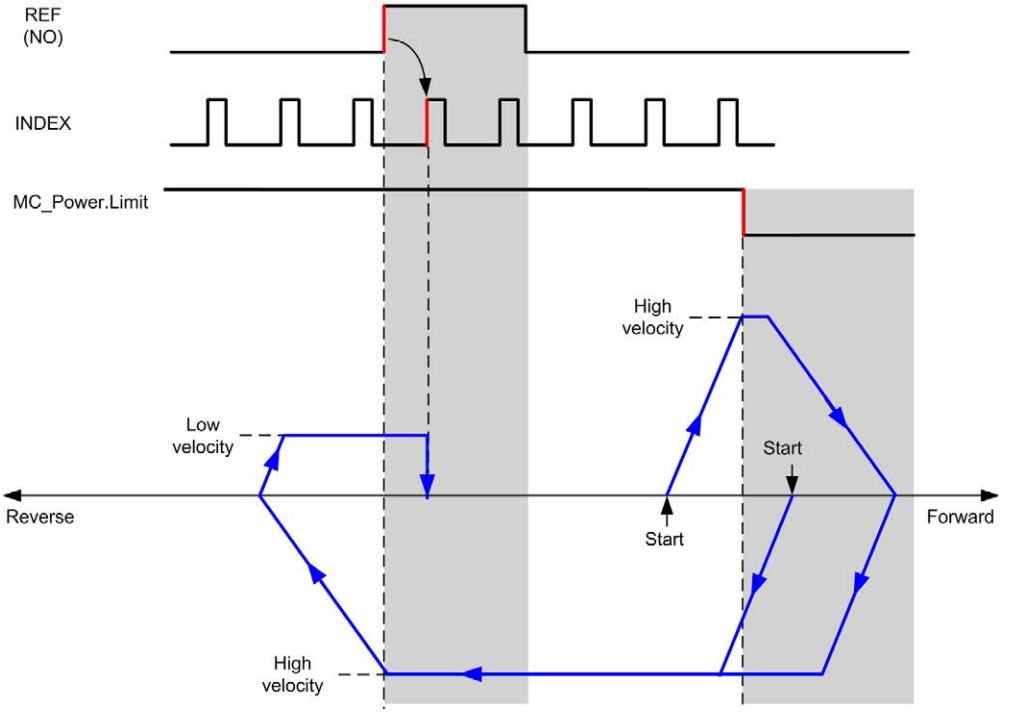
Short Reference & Index Inside: Positive Direction

Homes to the first index, after 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)

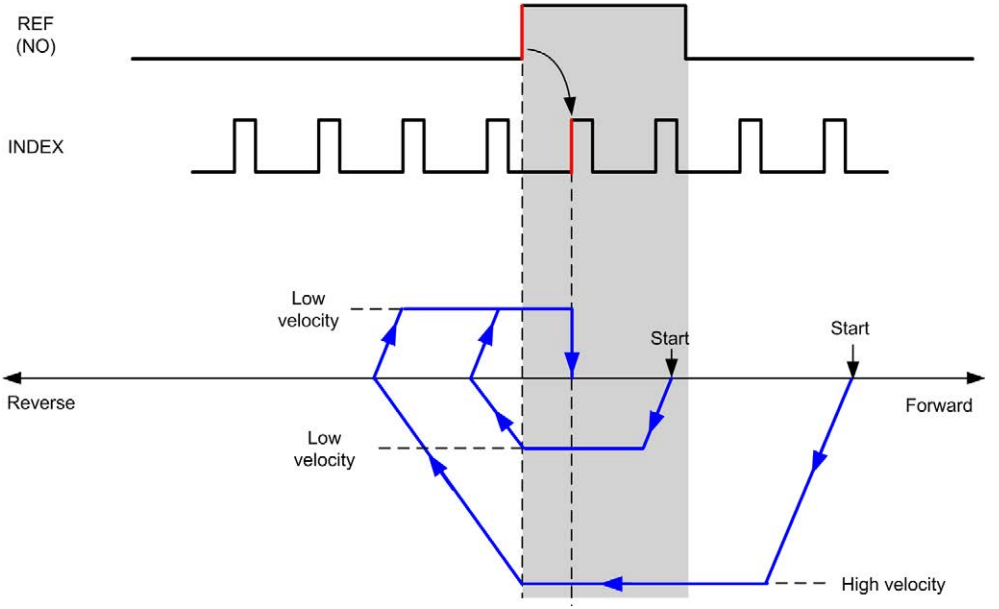


REF (NO) Reference point (Normally Open)

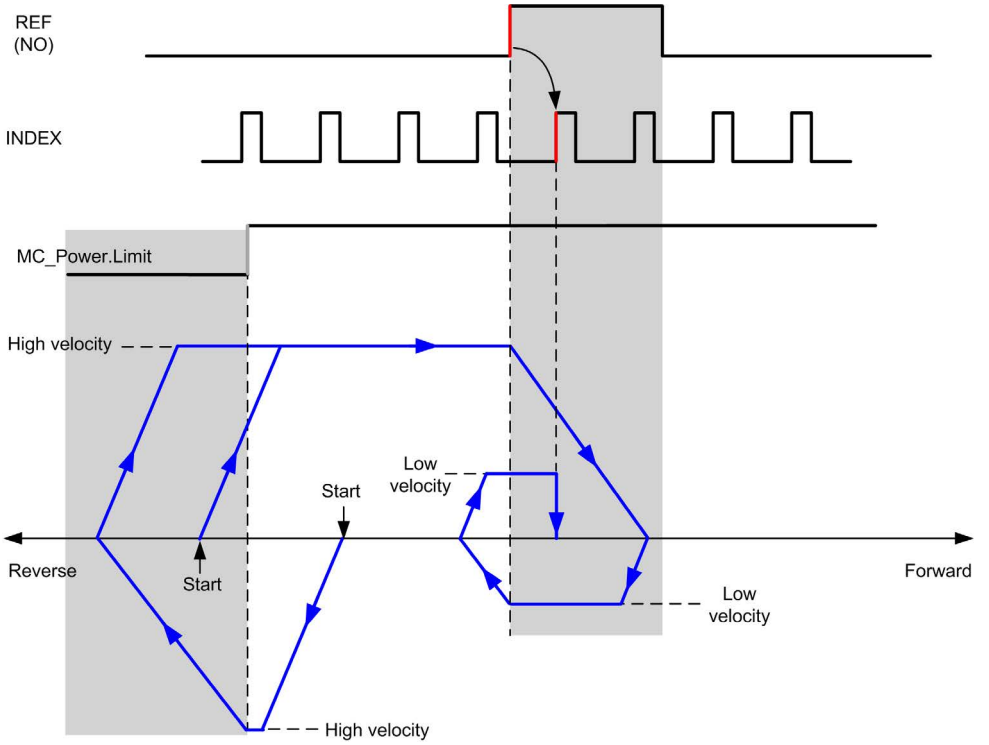
Short Reference & Index Inside: Negative Direction

Homes to the first index, after the reference switch rising edge in forward direction.

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



REF (NO) Reference point (Normally Open)



REF (NO) Reference point (Normally Open)

Home Offset

Description

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

Home offset is set in number of pulses (-2,147,483,648...2,147,483,647, default value 0). When set by configuration, the `MC_Home_PTO` *(see page 128)* command is executed first, and then the specified number of pulses is output at the home low velocity in the specified direction. The parameter is only effective during a reference movement without index pulse.

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

Chapter 5
Data Unit Types

Overview

This chapter describes the data unit types of the M241 PTO Library.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
AXIS_REF_PTO Data Type	80
MC_BUFFER_MODE	81
MC_DIRECTION	83
PTO_HOMING_MODE	84
PTO_PARAMETER	85
PTO_ERROR	86

AXIS_REF_PTO Data Type

Data Type Description

The AXIS_REF_PTO type is a data type that contains information on the corresponding axis. It is used as a VAR_IN_OUT in all function blocks of the PTO library.

MC_BUFFER_MODE

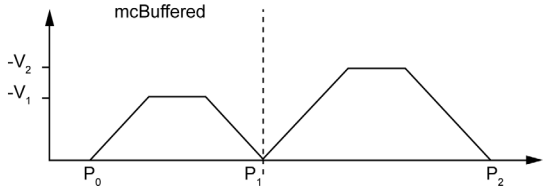
Buffer Mode Enumeration

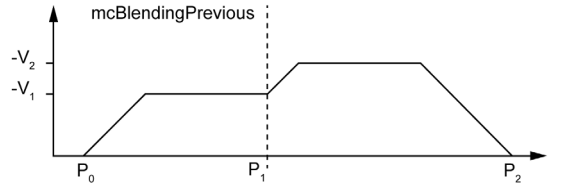
This table lists the values for the MC_BUFFER_MODE enumeration:

Enumerator	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 move has finished (`Done` or `InVelocity` bit is set). 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 `InVelocity` bit is set) and the time delay has elapsed. There is no blending. The `Delay` parameter is set using `MC_WriteParameter_PTO` *(see page 160)*, with `ParameterNumber` 1000.

Examples

The examples below show a movement executed by two motion commands. The axis moves from the position P_0 to P_1 and then P_2. The second command is passed while the axis is executing the first command but before the stopping ramp is reached. For each motion profile below, P_1 is the reference point for the blending calculation. The buffer mode determines whether velocity V_1 or V_2 is reached at position P_1.





MC_DIRECTION

Move Direction Enumeration

This table lists the values for the MC_DIRECTION enumeration:

Enumerator	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).
`mcCurrentDirection`	2	Move in the last used direction.

PTO_HOMING_MODE

Homing Mode Enumeration

This table lists the values for the PTO_HOMING_MODE enumeration:

Enumerator	Value	Description
PositionSetting	0	Position.
LongReference	1	Long reference.
LongReferenceAndIndex	10	Long reference and index.
ShortReference_Reversal	20	Short reference.
ShortReference_NoReversal	21	Short reference no reversal.
ShortReferenceAndIndex_Outside	30	Short reference and index outside.
ShortReferenceAndIndex_Inside	31	Short reference and index inside.

PTO_PARAMETER

PTO Parameter Enumeration

This table lists the values for the PTO_PARAMETER enumeration:

Parameter Name	Parameter Number	Type	Standard	R/W	Description
CommandedPosition	1	DINT	Mandatory	R	Commanded position.
SWLimitPos	2	DINT	Optional	R/W	Positive software limit switch position.
SWLimitNeg	3	DINT	Optional	R/W	Negative software limit switch position.
EnableLimitPos	4	BOOL	Optional	R/W	Enable positive software limit switch.
EnableLimitNeg	5	BOOL	Optional	R/W	Enable negative software limit switch.
MaxVelocityAppl	9	DINT	Mandatory	R/W	Maximal allowed velocity of the axis in the application.
ActualVelocity	10	DINT	Mandatory	R	Actual velocity.
CommandedVelocity	11	DINT	Mandatory	R	Commanded velocity.
MaxAccelerationAppl	13	DINT	Optional	R/W	Maximal allowed acceleration of the axis in the application.
MaxDecelerationAppl	15	DINT	Optional	R/W	Maximal allowed deceleration of the axis in the application.
Reserved	to 999	-	-	-	Reserved for the PLCopen standard.
Delay	1000	DINT	Optional	R/W	Time in ms (0...65,535) Default value: 0

PTO_ERROR

PTO Error Enumeration

This table lists the values for the PTO_ERROR enumeration:

Enumerator	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.
Axis Control Advisories		
`WarningVelocityValue`	1100	Commanded Velocity parameter is out of range.
`WarningAccelerationValue`	1101	Commanded Acceleration parameter is out of range.
`WarningDecelerationValue`	1102	Commanded Deceleration parameter is out of range.
`WarningDelayedMove`	1103	Not enough time to stop the active move, so the requested move is delayed.
`WarningJerkRatioValue`	1104	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.
Motion State Advisories		
`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 high).
`InvalidTransition`	2002	Transition not allowed, refer to the Motion State Diagram *(see page 91)*.

Enumerator	Value	Description
InvalidSetPosition	2003	MC_SetPosition_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 home inputs (Ref, Index) 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 128)*).
MotionQueueFull	2008	The move could not be buffered because the motion queue is full.
Range Advisories		
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. The value must be greater than the start velocity and less than the maximum velocity.
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	Home mode is not applicable.
InvalidParameter	3008	The parameter number does not exist for the specified axis.
InvalidParameterValue	3009	Parameter value is out of range.
ReadOnlyParameter	3010	Parameter is read-only.

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_ReadStatus_PTO and MC_ReadAxisError_PTO.

A **Motion State Advisory** or a **Range Advisory** does not affect the axis state, nor any ongoing move, nor the move queue. In this case, the error is only local to the applicable function block: the Error output is set, and the ErrorId pin is set to the appropriate PTO_ERROR value.

Chapter 6
Motion Function Blocks

Overview

This chapter describes the motion function blocks.

A motion function block acts on the diagram of axis state, to modify the motion of the axis. These function blocks can return a status to the application before the move is complete. The application program uses these status bits to determine the move status (`Done`, `Busy`, `Active`, `CommandAborted`, and detected `Error`). For axis status, you can use the `MC_ReadStatus_PTO` function block.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Operation Modes	90
6.2	MC_Power_PTO Function Block	104
6.3	MC_MoveVelocity_PTO Function Block	109
6.4	MC_MoveRelative_PTO Function Block	115
6.5	MC_MoveAbsolute_PTO Function Block	122
6.6	MC_Home_PTO Function Block	128
6.7	MC_SetPosition_PTO Function Block	133
6.8	MC_Stop_PTO Function Block	136
6.9	MC_Halt_PTO Function Block	141
6.10	Adding a Motion Function Block	146

Section 6.1
Operation Modes

Overview

This section describes the operation modes.

What Is in This Section?

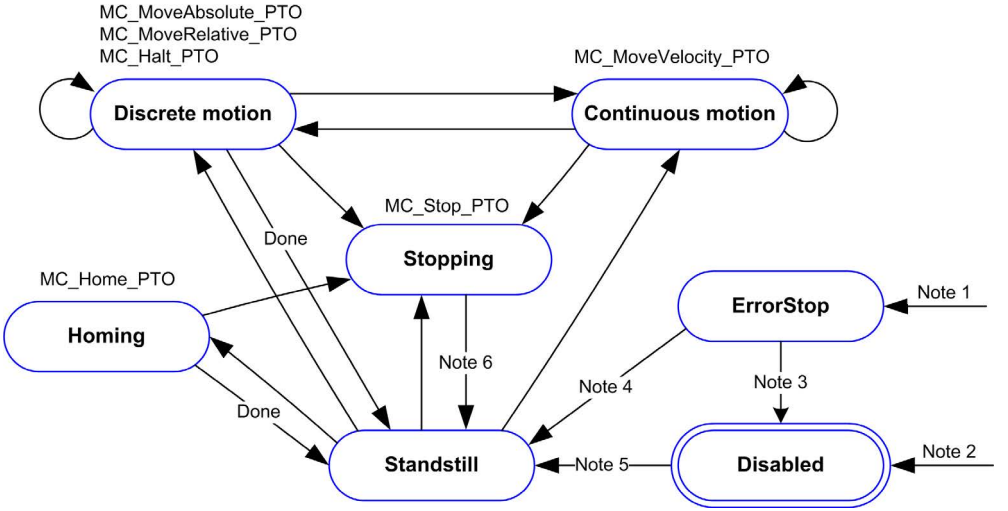
This section contains the following topics:

Topic	Page
Motion State Diagram	91
Buffer Mode	93
Timing Diagram Examples	95

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, there is no error detected, and there are no motion commands 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` pin is set on applicable function blocks, and an `ErrorId` sets the error code. 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.

State	Description
Discrete	Applicable when `MC_MoveRelative_PTO`, `MC_MoveAbsolute_PTO`, or `MC_Halt_PTO` controls the axis.
Continuous	Applicable when `MC_MoveVelocity_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 ongoing command according to the following table:

Command		Next					
		Home	**MoveVelocity**	**MoveRelative**	**MoveAbsolute**	**Halt**	**Stop**
Ongoing	**Standstill**	Allowed	Allowed (1)	Allowed (1)	Allowed (1)	Allowed	Allowed
	Home	Rejected	Rejected	Rejected	Rejected	Rejected	Allowed
	MoveVelocity	Rejected	Allowed	Allowed	Allowed	Allowed	Allowed
	MoveRelative	Rejected	Allowed	Allowed	Allowed	Allowed	Allowed
	MoveAbsolute	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

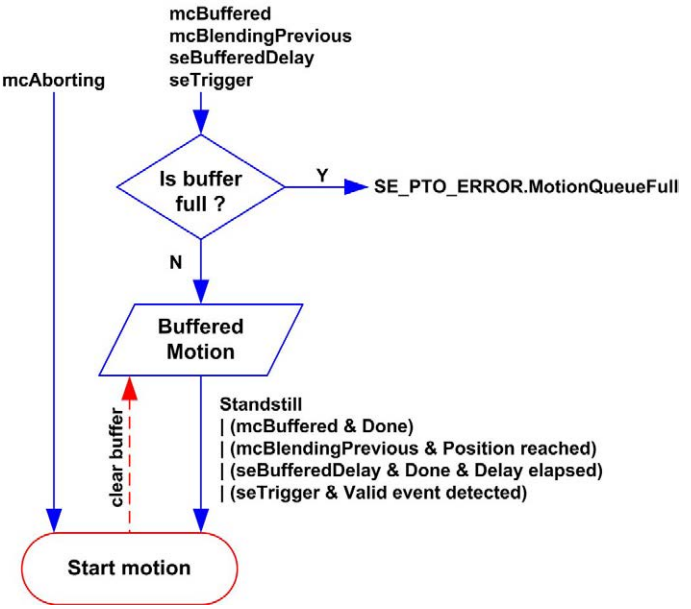
Some of the motion function blocks have an input called `BufferMode`. With this input, the function block can either start immediately, start on probe event, or be buffered.

The available options are defined in the enumeration of type MC_BUFFER_MODE *(see page 81)*:

- An aborting motion (`mcAborting`) starts immediately, aborting any ongoing move, and clearing the motion queue.
- A buffered motion (`mcBuffered`, `mcBlendingPrevious`, `seBufferedDelay`) is queued, that is, appended to any moves currently executing or waiting to execute, and will start when the previous motion is done.
- An event motion (`seTrigger`) is a buffered motion, starting on probe event *(see page 48)*.

Motion Queue Diagram

The figure illustrates the motion queue diagram:



The buffer can contain only one motion function block.

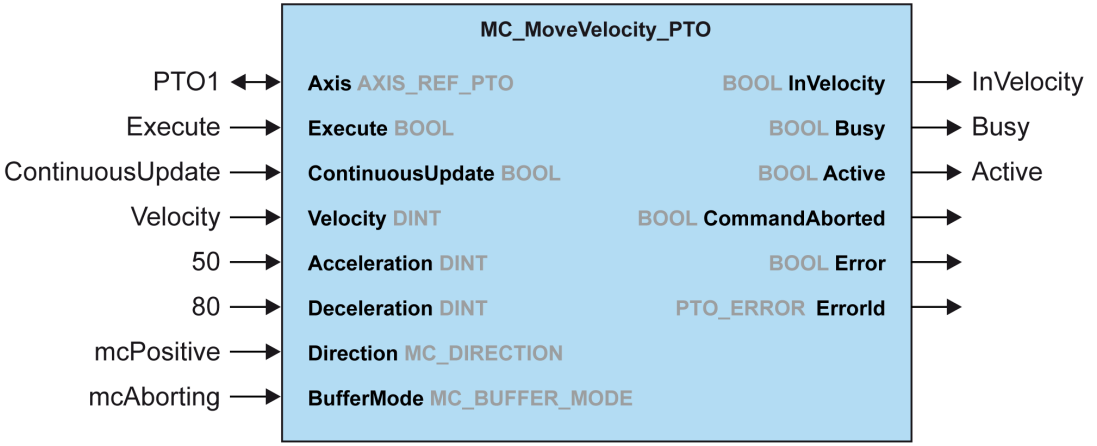
The execution condition of the motion function block present in the buffer is:
- `mcBuffered`: when the current continuous motion is `InVelocity`, resp. when the current discrete motion stops.
- `seBufferedDelay`: when the specified delay has elapsed, from the current continuous motion is `InVelocity`, resp. from the current discrete motion stops.
- `mcBlendingPrevious`: when the position and velocity targets of current function block are reached.
- `seTrigger`: when a valid event is detected on the probe input.

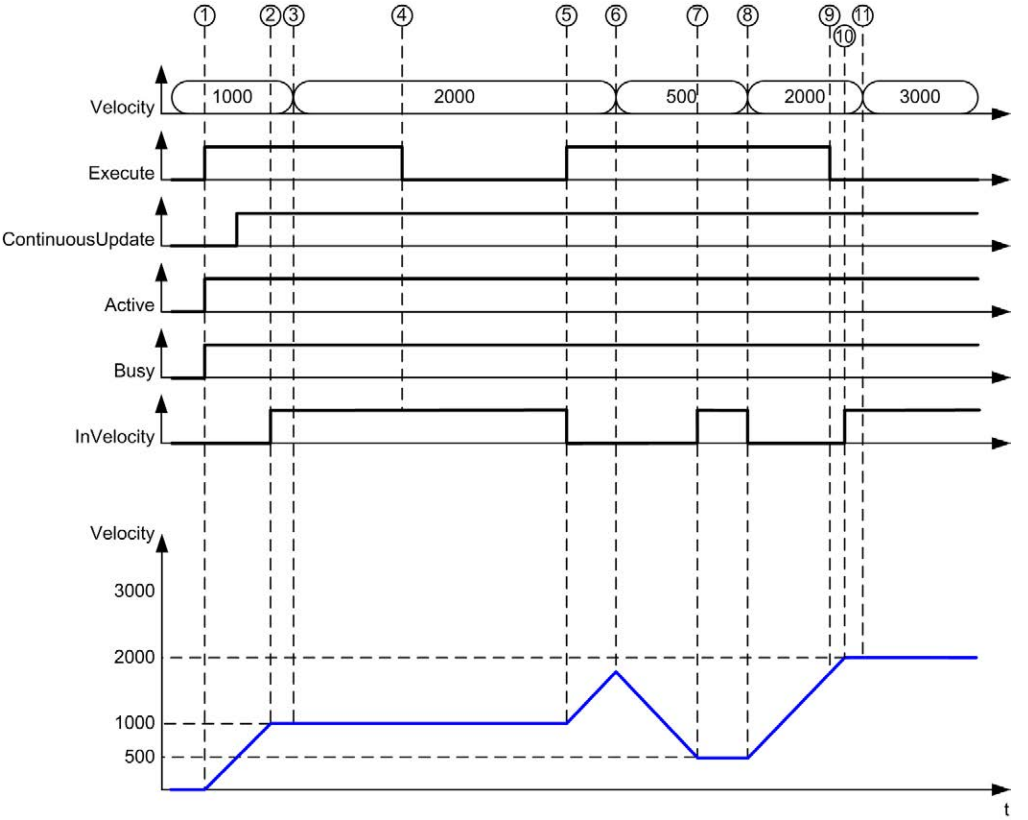
The motion queue is cleared (all buffered motions are deleted):
- When an aborting move is triggered (`mcAborting`): `CommandAborted` pin is set on buffered function blocks.
- When a `MC_Stop_PTO` function is executed: `Error` pin is set on cleared buffered function blocks, with `ErrorId=StoppingActive` *(see page 86)*.
- When a transition to **ErrorStop** state is detected: `Error` pin is set on buffered function blocks, with `ErrorId=ErrorStopActive` *(see page 86)*.

NOTE:
- Only a valid motion can be queued. If the function block execution terminates with the `Error` output set, 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 on the applicable function block, and `ErrorId` output returns the error `MotionQueueFull` *(see page 86)*.

Timing Diagram Examples

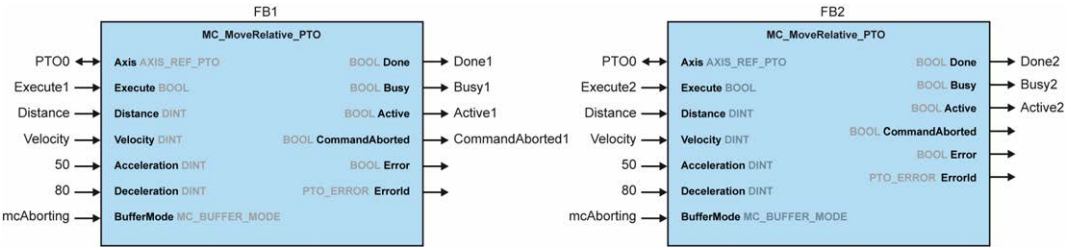
Move Velocity to Move Velocity with mcAborting

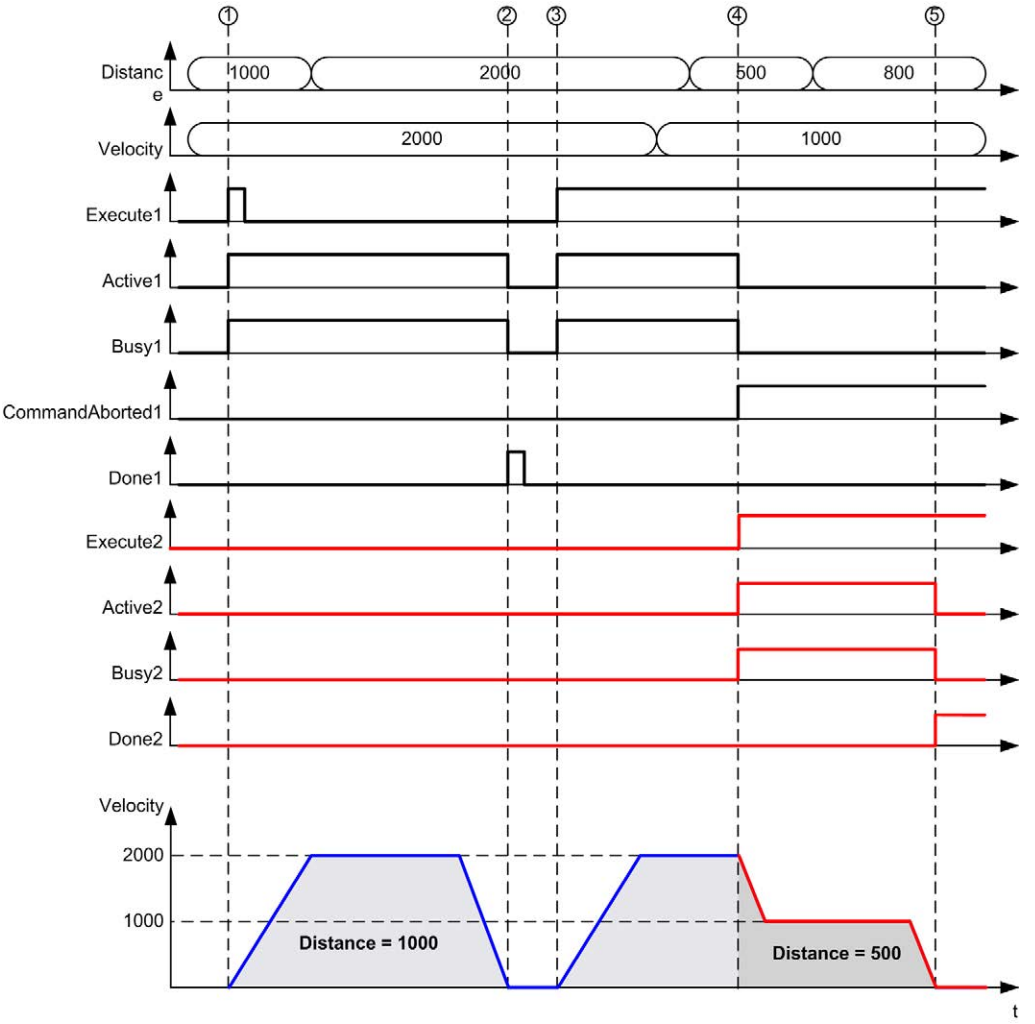




1 `Execute` rising edge: command parameters are latched, movement is started with target `velocity` 1000.
2 Target `velocity` 1000 is reached.
3 `Velocity` parameter changed to 2000: not applied (no rising edge on `Execute` input, and `ContinuousUpdate` was latched with value 0 at start of the movement).
4 `Execute` falling edge: status bits are cleared.
5 `Execute` rising edge: command parameters are latched, movement is started with target `velocity` 2000 and `ContinuousUpdate` active.
6 Velocity parameter changed to 500: applied `ContinuousUpdate` is true). Note: previous target `velocity` 2000 is not reached.
7 Target `velocity` 500 is reached.
8 `Velocity` parameter changed to 2000: applied `ContinuousUpdate` is true).
9 `Execute` falling edge: status bits are cleared.
10 Target `velocity` 2000 is reached, `InVelocity` is set for 1 cycle (`Execute` pin is reset).
11 `Velocity` parameter changed to 3000: not applied (movement is still active, but no longer busy).

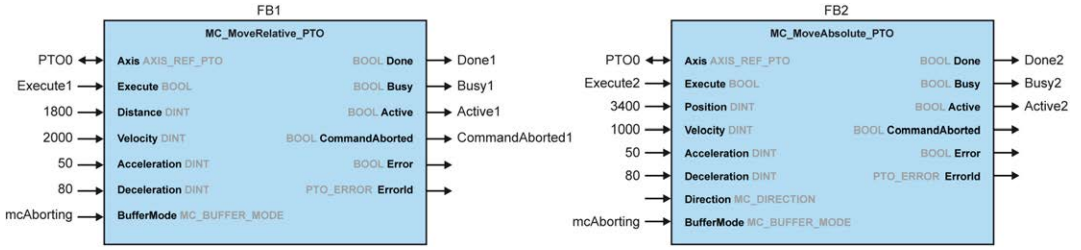
Move Relative to Move Relative with mcAborting

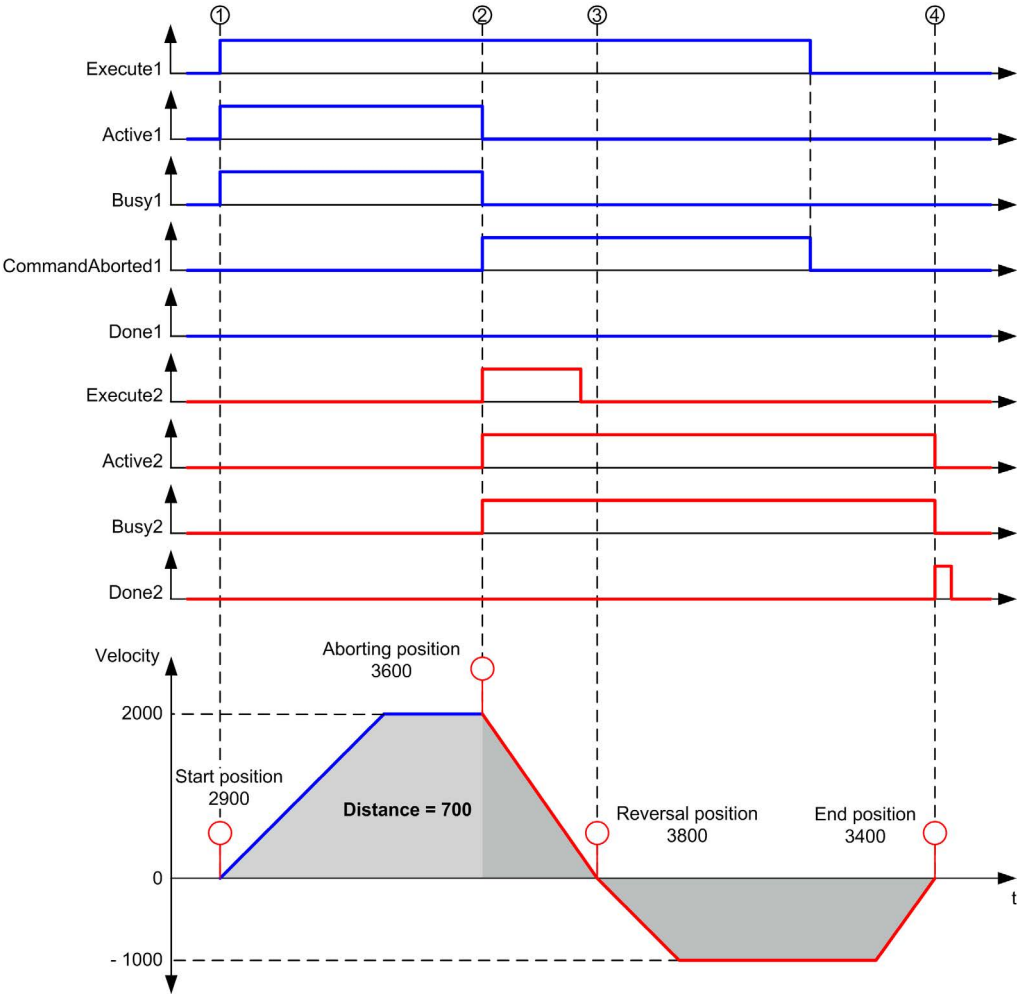




1 FB1 `Execute` rising edge: command parameters are latched, movement is started with target `velocity` 2000 and `distance` 1000.
2 Movement ends: distance traveled is 1000.
3 FB1 `Execute` rising edge: command parameters are latched, movement is started with target `velocity` 2000 and `distance` 2000.
4 FB2 `Execute` rising edge: command parameters are latched, movement is started with target `velocity` 1000 and `distance` 500. Note: FB1 is aborted.
5 Movement ends.

Move Relative to Move Absolute with mcAborting





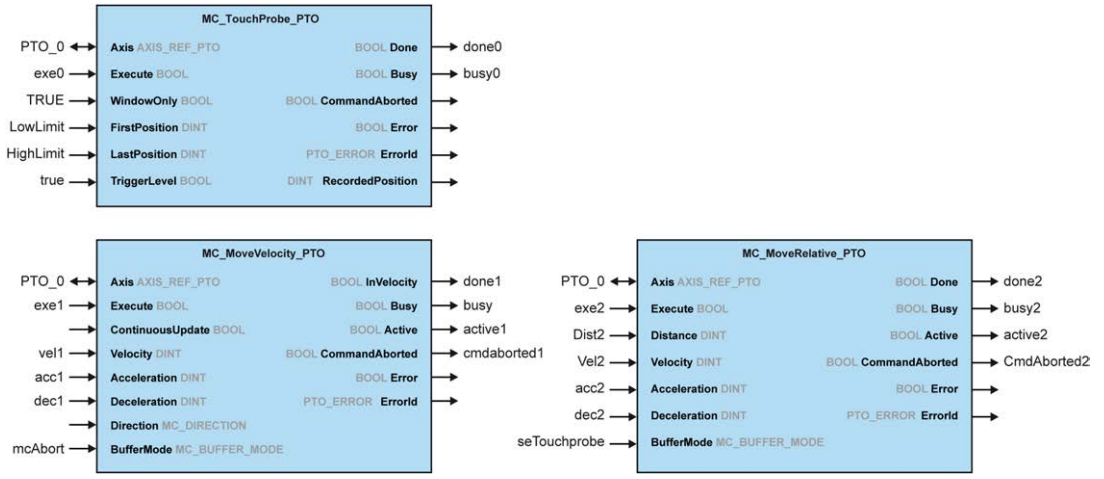
1 FB1 `Execute` rising edge: command parameters are latched, movement is started with target `velocity` 2000 and `distance` 1800.

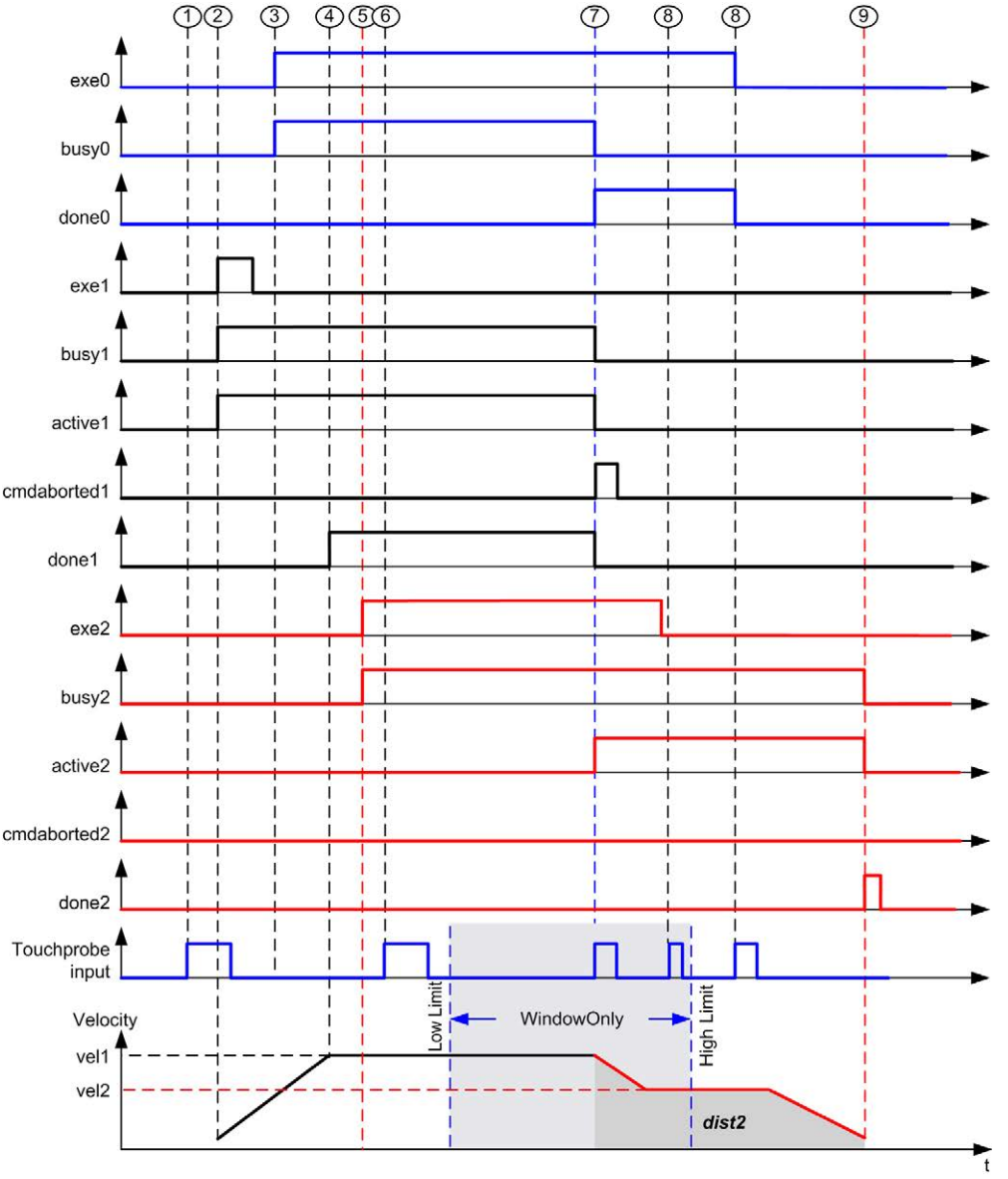
2 FB2 `Execute` rising edge: command parameters are latched, FB1 is aborted, and movement continues with target `velocity` 1000 and target`position` 3400. Automatic direction management: direction reversal is needed to reach target position, move to stop at `deceleration` of FB2.

3 Velocity 0, direction reversal, movement resumes with target `velocity` 1000 and target `position` 3400.

4 Movement ends: target position 3400 reached.

Move Velocity to Move Relative with seTrigger





1 `MC_TouchProbe_PTO` not executed yet: probe input is not active.
2 `MC_MoveVelocity_PTO` `Execute` rising edge: command parameters are latched, movement is started with target `velocity` vel1.
3 `MC_TouchProbe_PTO` `Execute` rising edge: probe input is active.
4 vel1 is reached.

5 `MC_MoveRelative_PTO` `Execute` rising edge: command parameters are latched, waiting for probe event to start.
6 Probe event outside of enable windows: event is ignored.
7 A valid event is detected. `MC_MoveRelative_PTO` aborts `MC_MoveVelocity_PTO`, and probe input is deactivated.
8 Following events are ignored.
9 Movement ends.

Section 6.2
MC_Power_PTO Function Block

Overview

This section describes the `MC_Power_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	105
MC_Power_PTO: Manage the Power of the Axis State	106

Description

Overview

The MC_Power_PTO function block is mandatory for execution of the other PTO function blocks. It allows enabling power and control to the axis, switching the axis state from **Disabled** to **Standstill**.

This function block must always be the first PTO function block called.

No motion function block is allowed to affect the axis until the MC_Power_PTO.Status bit is TRUE.

Disabling power (MC_Power_PTO.Enable = FALSE) switches the axis:

- from **Standstill**, back to **Disabled** state.
- from any ongoing move, to **ErrorStop**, and then **Disabled** when the error is reset.

If DriveReady input is reset, the axis state switches to **ErrorStop**.

MC_Power_PTO: Manage the Power of the Axis State

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared under the controller configuration.
`Enable`	BOOL	FALSE	When TRUE, the function block is executed. The values of the function block inputs can be modified and the outputs updated continuously. When FALSE, terminates the function block execution and resets its outputs.
`DriveReady`(1)	BOOL	FALSE	Drive ready information from the drive. Must be TRUE when the drive is ready to start executing motion. If the drive signal is connected to the controller, use the appropriate %Ix input. If the drive does not provide this signal, you can select the value TRUE for this input.
`LimP`(1)	BOOL	TRUE	Hardware limit switch information, in positive direction. It must be FALSE when the hardware limit switch is reached. If the hardware limit switch signal is connected to the controller, use the appropriate %Ix input. If this signal is not available, you can leave this input unused or set to TRUE.

Input	Type	Initial Value	Description
LimN(1)	BOOL	TRUE	Hardware limit switch information, in negative direction. It must be FALSE when the hardware limit switch is reached. If the hardware limit switch signal is connected to the controller, use the appropriate %Ix. If this signal is not available, you can leave this input unused or set to TRUE.

(1) DriveReady, LimP, and LimN are read at the task cycle time.

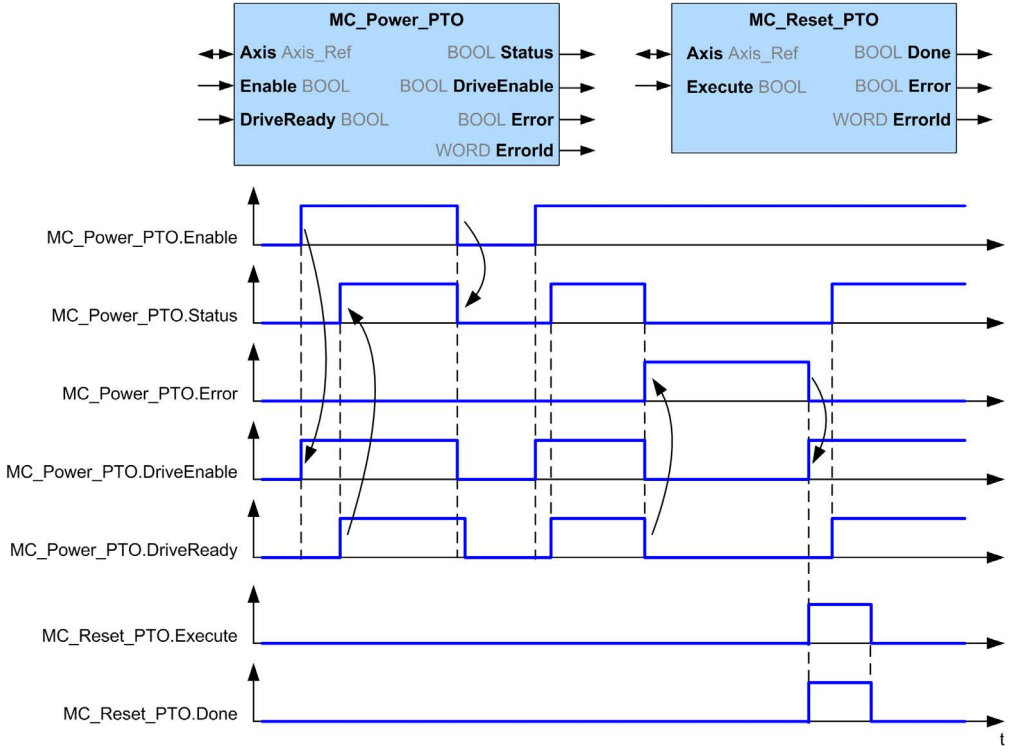
Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
Status	BOOL	FALSE	When TRUE, power is enabled, motion commands are possible.
DriveEnable	BOOL	FALSE	Enables the drive to accept commands. If the drive does not use this signal, you can leave this output unused.
Error	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
ErrorId	PTO_ERROR	PTO_ERROR. No Error	When Error is TRUE: code of the error detected *(see page 86)*.

Timing Diagram Example

The diagram illustrates the function block operation:



Section 6.3
MC_MoveVelocity_PTO Function Block

Overview

This section describes the `MC_MoveVelocity_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	110
MC_MoveVelocity_PTO: Control the Speed of the Axis	111

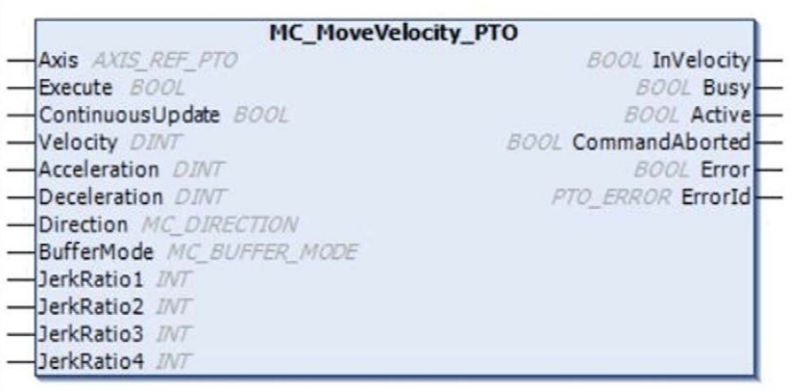
Description

Overview

This function causes the specified axis to move at the specified speed, and transfers 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_MoveVelocity_PTO: Control the Speed of the Axis

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates. Later changes in the function block input parameters do not affect the ongoing command, unless the input `ContinuousUpdate` is used. If a second rising edge is detected during the execution of the function block, the ongoing execution is aborted and the function block is restarted with the values of the parameters at the time.

Input	Type	Initial Value	Description
ContinuousUpdate	BOOL	FALSE	At TRUE, makes the function block use the values of the input variables (Velocity, Acceleration, Deceleration, and Direction), and apply it to the ongoing command regardless of their original values. The impact of the input ContinuousUpdate begins when the function block is triggered by a rising edge on the Execute pin, and ends as soon as the function block is no longer Busy or the input ContinuousUpdate is set to FALSE.
Velocity	DINT	0	Target velocity in Hz, not necessarily reached. Range: 0...MaxVelocityAppl *(see page 85)*
Acceleration	DINT	0	Acceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...MaxAccelerationAppl *(see page 85)* Range (ms): MaxAccelerationAppl *(see page 85)*...100,000
Deceleration	DINT	0	Deceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...MaxDecelerationAppl *(see page 85)* Range (ms): MaxDecelerationAppl *(see page 85)*...100,000
Direction	MC_DIRECTION	mcPositiveDirection	Direction of the movement *(see page 83)*.
BufferMode	MC_BUFFER_MODE	mcAborting	Transition mode from ongoing move *(see page 81)*.
JerkRatio1	INT	0	Percentage of acceleration from standstill used to create the S-curve profile *(see page 45)*.
JerkRatio2	INT	0	Percentage of acceleration to constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio3	INT	0	Percentage of deceleration from constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio4	INT	0	Percentage of deceleration to standstill used to create the S-curve profile *(see page 45)*.

Output Variables

This table describes the output variables:

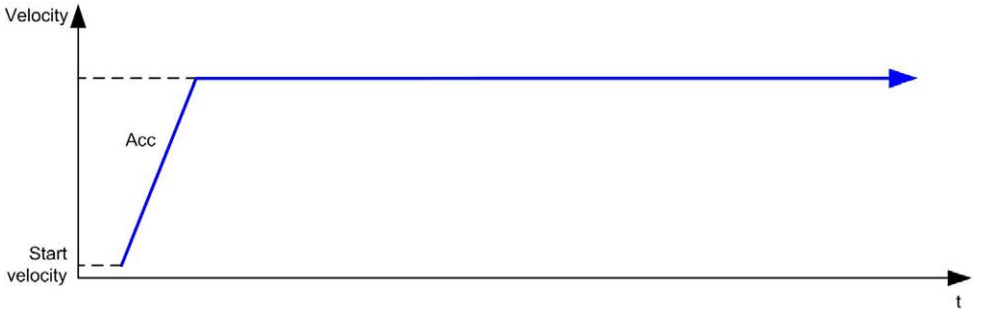
Output	Type	Initial Value	Description
`InVelocity`	BOOL	FALSE	If TRUE, indicates that the target velocity is reached.
`Busy`	BOOL	FALSE	If TRUE, indicates that the function block execution is in progress.
`Active`	BOOL	FALSE	The function block controls the `Axis`. Only one function block at a time can set `Active` TRUE for a defined `Axis`.
`CommandAborted`	BOOL	FALSE	Function block execution is finished, by aborting due to another move command or an error detected.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.

NOTE:

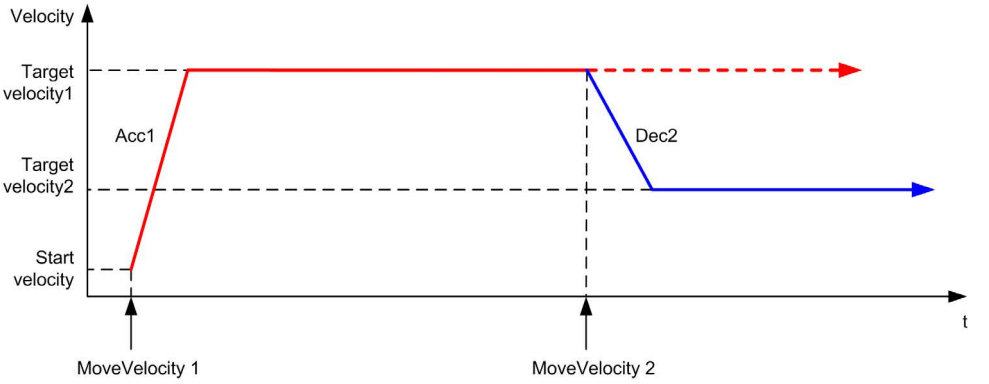
- To stop the motion, the function block has to be interrupted by another function block issuing a new command.
- If a motion is ongoing, and the direction is reversed, first the motion is halted with the deceleration of the `MC_MoveVelocity_PTO` function block, and then the motion resumes backwards.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

Timing Diagram Example

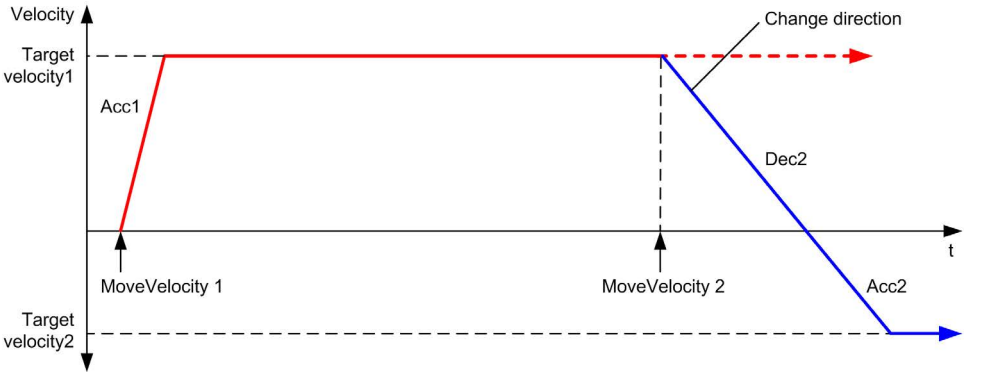
The diagram illustrates a simple profile from **Standstill** state:



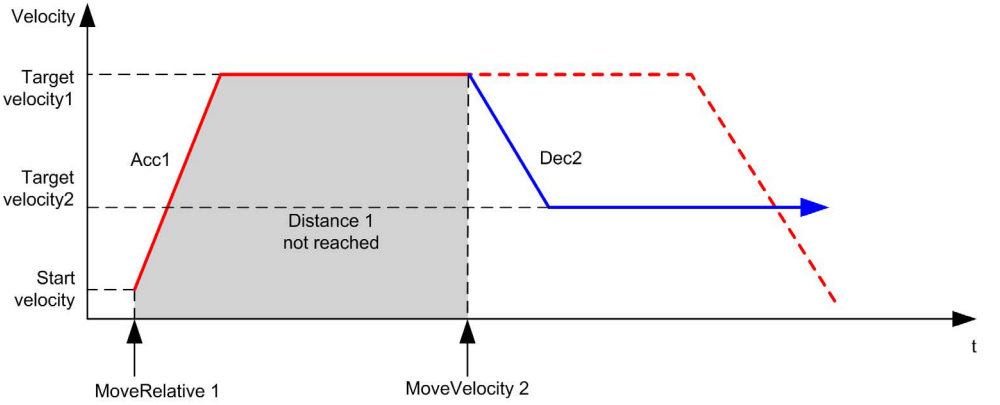
The diagram illustrates a complex profile from **Continuous** state:



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



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



Section 6.4
MC_MoveRelative_PTO Function Block

Overview

This section describes the `MC_MoveRelative_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	116
MC_MoveRelative_PTO: Command Relative Axis Movement	117

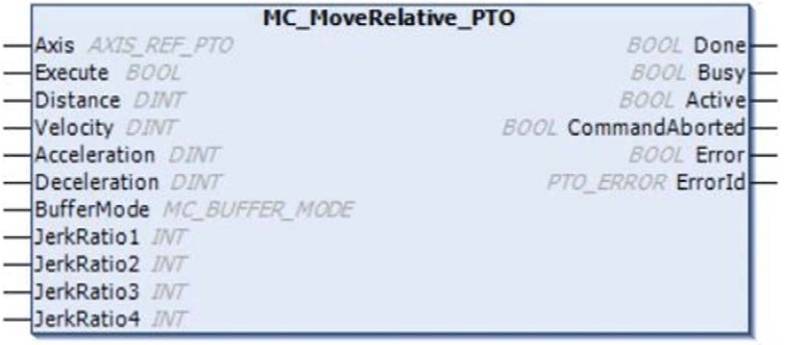
Description

Overview

This function causes the specified axis to move of an incremental distance, and transfers the axis to the state **Discrete**. The target position is referenced from the current position at execution time, incremented by a distance.

MC_MoveRelative_PTO: Command Relative Axis Movement

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`Distance`	DINT	0	Relative distance of the motion in number of pulses. The sign specifies the direction.
`Velocity`	DINT	0	Target velocity in Hz, not necessarily reached. Range: 1...`MaxVelocityAppl` *(see page 85)*
`Acceleration`	DINT	0	Acceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...`MaxAccelerationAppl` *(see page 85)* Range (ms): `MaxAccelerationAppl` *(see page 85)*...100,000

Input	Type	Initial Value	Description
Deceleration	DINT	0	Deceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...MaxDecelerationAppl *(see page 85)* Range (ms): MaxDecelerationAppl *(see page 85)*...100,000
BufferMode	MC_BUFFER_MODE	mcAborting	Transition mode from ongoing move *(see page 81)*.
JerkRatio1	INT	0	Percentage of acceleration from standstill used to create the S-curve profile *(see page 45)*.
JerkRatio2	INT	0	Percentage of acceleration to constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio3	INT	0	Percentage of deceleration from constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio4	INT	0	Percentage of deceleration to standstill used to create the S-curve profile *(see page 45)*.

Output Variables

This table describes the output variables:

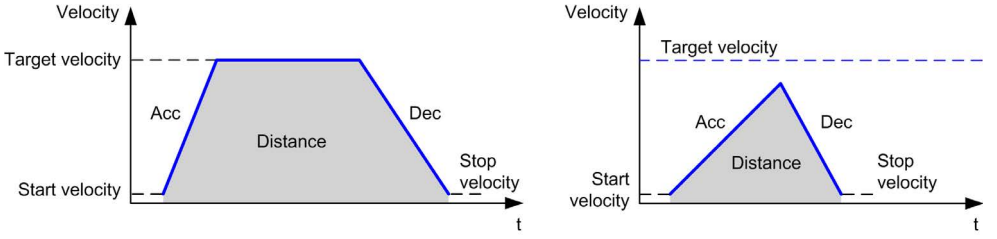
Output	Type	Initial Value	Description
Done	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected.
Busy	BOOL	FALSE	If TRUE, indicates that the function block execution is in progress.
Active	BOOL	FALSE	The function block controls the Axis. Only one function block at a time can set Active TRUE for a defined Axis.
CommandAborted	BOOL	FALSE	Function block execution is finished, by aborting due to another move command or an error detected.
Error	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
ErrorId	PTO_ERROR	PTO_ERROR.NoError	When Error is TRUE: code of the error detected *(see page 86)*.

NOTE:

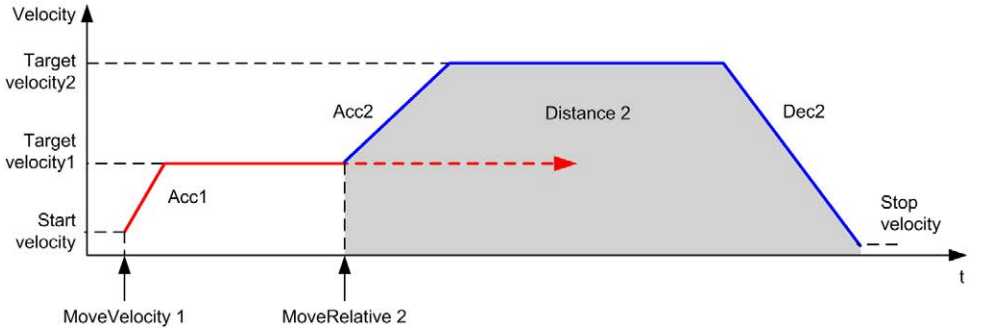
- The function block completes with velocity zero if no further blocks are pending.
- If the distance is too short for the target velocity to be reached, the movement profile is triangular, rather than trapezoidal.
- If a motion is ongoing, and the commanded distance is exceeded due to the motion parameters,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

Timing Diagram Example

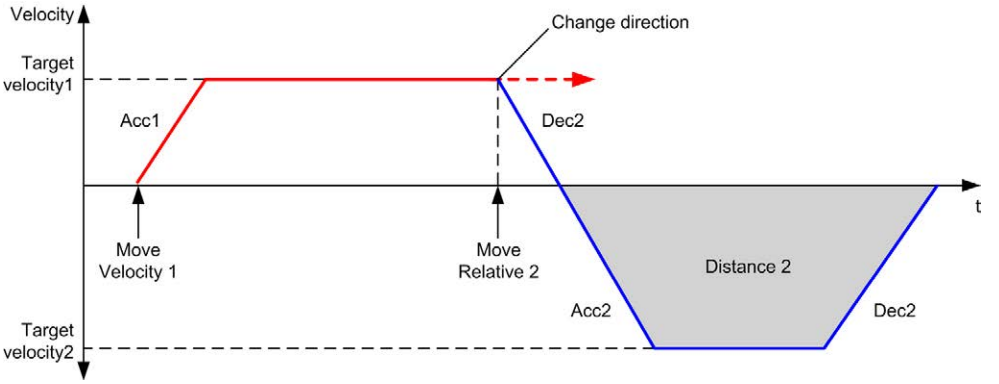
The diagram illustrates a simple profile from **Standstill** state:



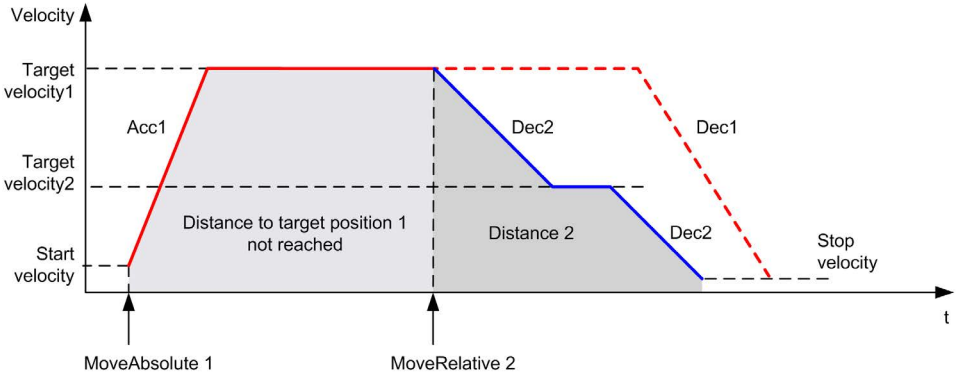
The diagram illustrates a complex profile from **Continuous** state:



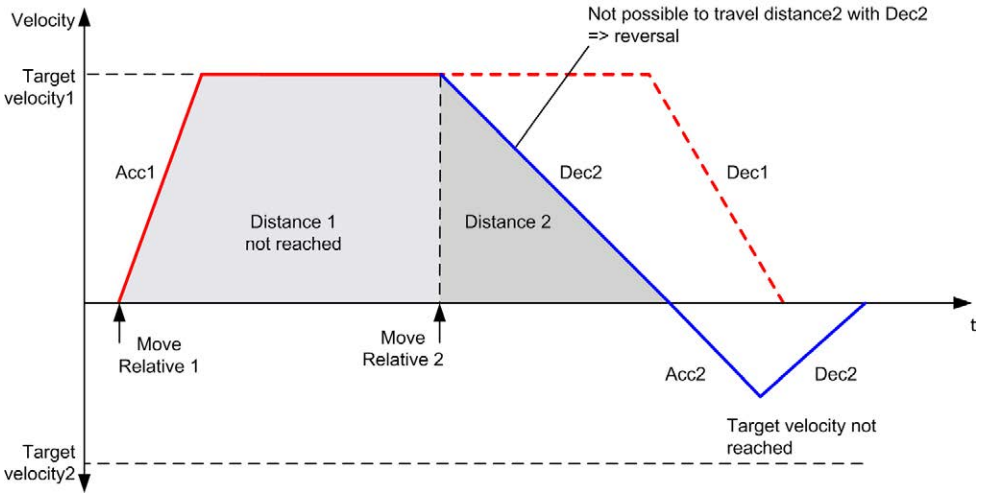
The diagram illustrates a complex profile from **Continuous** state with change of direction:



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



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



Section 6.5
MC_MoveAbsolute_PTO Function Block

Overview

This section describes the `MC_MoveAbsolute_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	123
MC_MoveAbsolute_PTO: Command Movement to Absolute Position	124

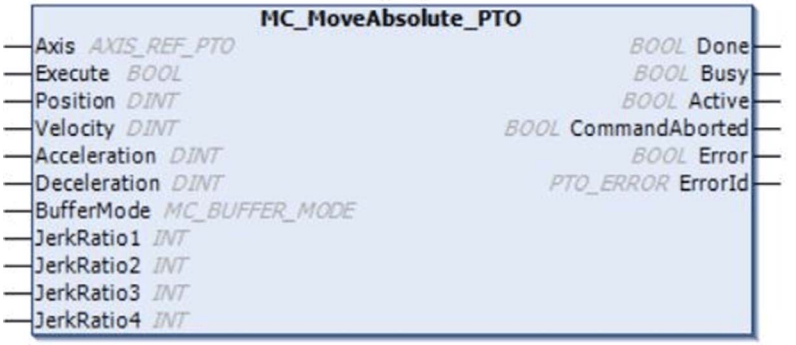
Description

Overview

This function causes the specified axis to move towards a given position at the specified speed, and transfers the axis to the state **Discrete**. To use the `MC_MoveAbsolute_PTO` function block, you must first home the axis. If not the function block will terminate in error (`Error` set to 1 and `ErrorId` set to `InvalidAbsolute`).

MC_MoveAbsolute_PTO: Command Movement to Absolute Position

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`Position`	DINT	0	Target absolute position.
`Velocity`	DINT	0	Target velocity in Hz, not necessarily reached. Range: 1...`MaxVelocityAppl` *(see page 85)*
`Acceleration`	DINT	0	Acceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...`MaxAccelerationAppl` *(see page 85)* Range (ms): `MaxAccelerationAppl` *(see page 85)*...100,000

Input	Type	Initial Value	Description
Deceleration	DINT	0	Deceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...MaxDecelerationAppl *(see page 85)* Range (ms): MaxDecelerationAppl *(see page 85)*...100,000
Direction	MC_DIRECTION	mcPositiveDirection	Direction of the movement.
BufferMode	MC_BUFFER_MODE	mcAborting	Transition mode from ongoing move *(see page 81)*.
JerkRatio1	INT	0	Percentage of acceleration from standstill used to create the S-curve profile *(see page 45)*.
JerkRatio2	INT	0	Percentage of acceleration to constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio3	INT	0	Percentage of deceleration from constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio4	INT	0	Percentage of deceleration to standstill used to create the S-curve profile *(see page 45)*.

Output Variables

This table describes the output variables:

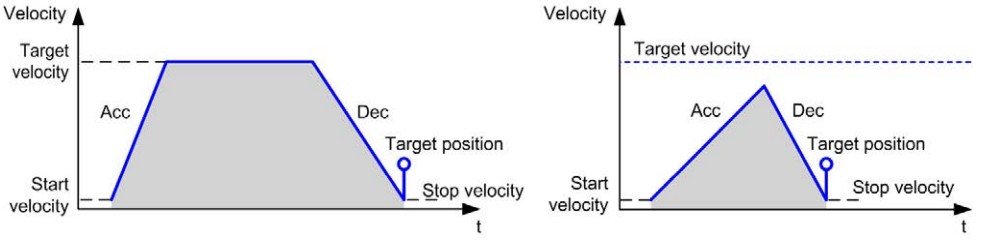
Output	Type	Initial Value	Description
Done	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected.
Busy	BOOL	FALSE	If TRUE, indicates that the function block execution is in progress.
Active	BOOL	FALSE	The function block controls the Axis. Only one function block at a time can set Active TRUE for a defined Axis.
CommandAborted	BOOL	FALSE	Function block execution is finished, by aborting due to another move command or an error detected.
Error	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
ErrorId	PTO_ERROR	PTO_ERROR.NoError	When Error is TRUE: code of the error detected *(see page 86)*.

NOTE:

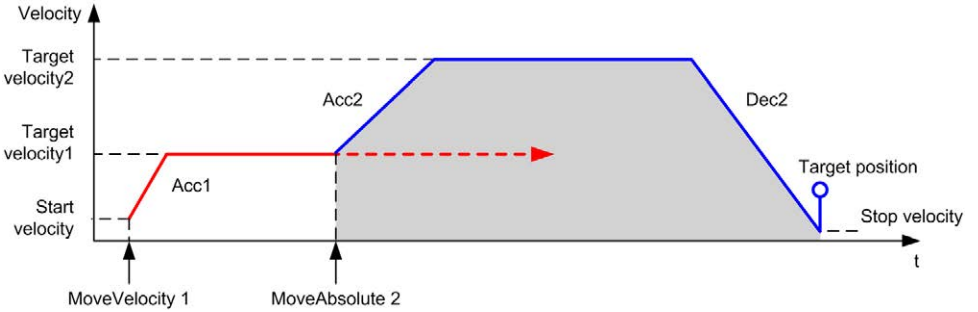
- The function block completes with velocity zero if no further blocks are pending.
- The motion direction is automatically set, according to the present and targeted positions.
- If the distance is too short for the target velocity to be reached, the movement profile is triangular, rather than trapezoidal.
- If the position cannot be reached with the ongoing 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 backwards.
- The acceleration/deceleration duration of the segment block must not exceed 80 seconds.

Timing Diagram Example

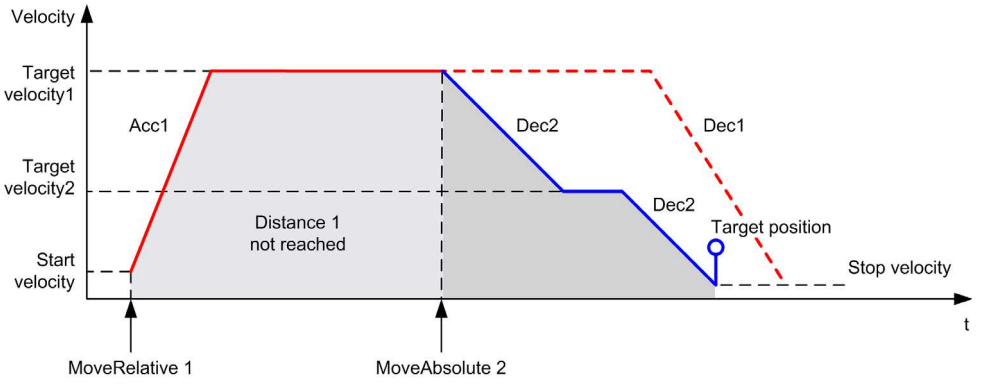
The diagram illustrates a simple profile from **Standstill** state:



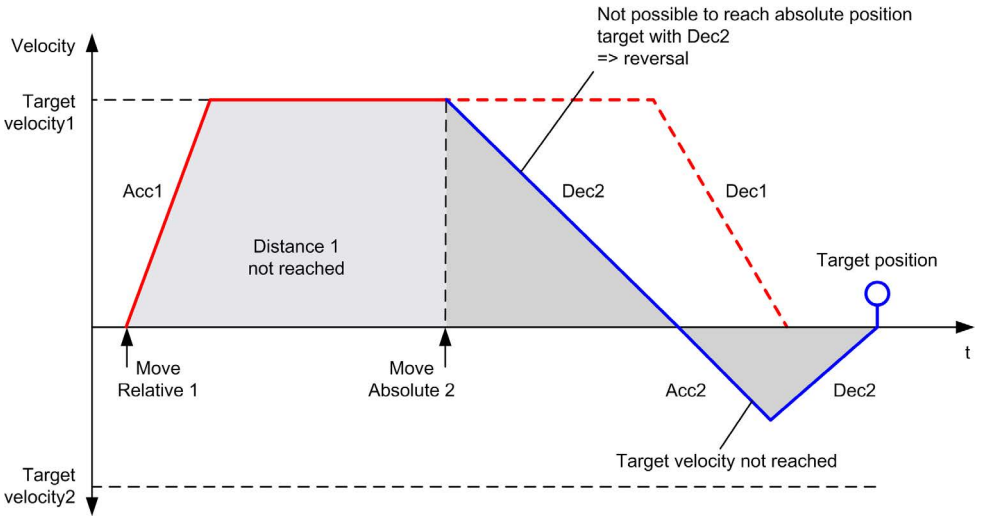
The diagram illustrates a complex profile from **Continuous** state:



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



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



Section 6.6
MC_Home_PTO Function Block

Overview

This section describes the `MC_Home_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	129
MC_Home_PTO: Command the Axis to Move to a Reference Position	130

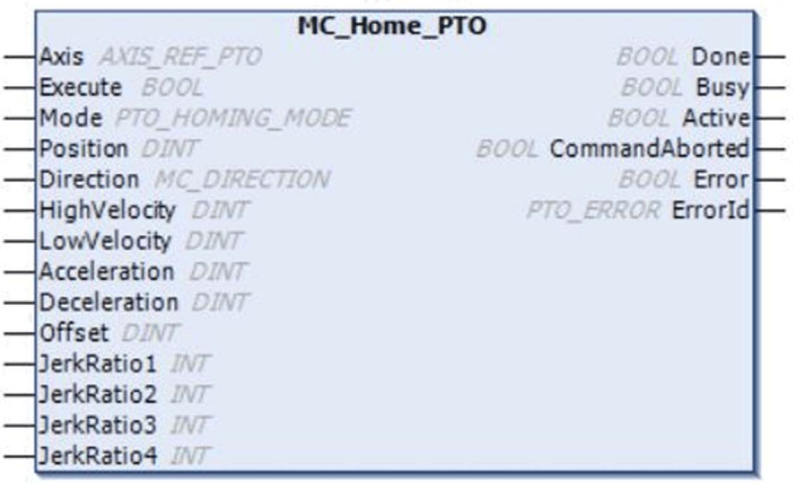
Description

Overview

This function block commands the axis to move to the reference absolute position, and transfers the axis to the state **Homing**. The details of this sequence depend on homing configuration parameter settings.

MC_Home_PTO: Command the Axis to Move to a Reference Position

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`Mode`	PTO_HOMING_MODE	`mcPositionSetting`	Predefined home mode *(see page 84)* type.
`Position`	DINT	0	Position value is set as absolute position at the reference point switch detection, when the homing has been successfully executed.

Input	Type	Initial Value	Description
Direction	MC_DIRECTION	mcPositiveDirection	Starting direction. For Homing, only mcPositiveDirection and mcNegativeDirection are valid.
HighVelocity	DINT	0	Target homing velocity for searching the limit or reference switch. Range Hz: 1...MaxVelocityAppl *(see page 85)*
LowVelocity	DINT	0	Target homing velocity for searching the reference switch or index signal. The movement stop when switching point is detected. Range Hz: 1...HighVelocity
Acceleration	DINT	0	Acceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...MaxAccelerationAppl *(see page 85)* Range (ms): MaxAccelerationAppl *(see page 85)*...100,000
Deceleration	DINT	0	Deceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...MaxDecelerationAppl *(see page 85)* Range (ms): MaxDecelerationAppl *(see page 85)*...100,000
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 77)*). Range: -2,147,483,648...2,147,483,647
JerkRatio1	INT	0	Percentage of acceleration from standstill used to create the S-curve profile *(see page 45)*.
JerkRatio2	INT	0	Percentage of acceleration to constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio3	INT	0	Percentage of deceleration from constant velocity used to create the S-curve profile *(see page 45)*.
JerkRatio4	INT	0	Percentage of deceleration to standstill used to create the S-curve profile *(see page 45)*.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected .
`Busy`	BOOL	FALSE	If TRUE, indicates that the function block execution is in progress.
`Active`	BOOL	FALSE	The function block controls the `Axis`. Only one function block at a time can set `Active` TRUE for a defined `Axis`.
`CommandAborted`	BOOL	FALSE	Function block execution is finished, by aborting due to another move command or an error detected .
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ ERROR	`PTO_ERROR. NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.

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

Timing Diagram Example

Home modes *(see page 57)*

Section 6.7
MC_SetPosition_PTO Function Block

Overview

This section describes the `MC_SetPosition_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	134
MC_SetPosition_PTO: Force the Reference Position of the Axis	135

Description

Overview

This function block modifies the coordinates of the actual position of the axis without any physical movement. This function block can only be used while the axis is a **Standstill** state.

MC_SetPosition_PTO: Force the Reference Position of the Axis

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`Position`	DINT	0	New value of absolute position of the `Axis`.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected .
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: type of the error detected *(see page 86)*.

Section 6.8
MC_Stop_PTO Function Block

Overview

This section describes the `MC_Stop_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	137
MC_Stop_PTO: Command a Controlled Motion Stop	138

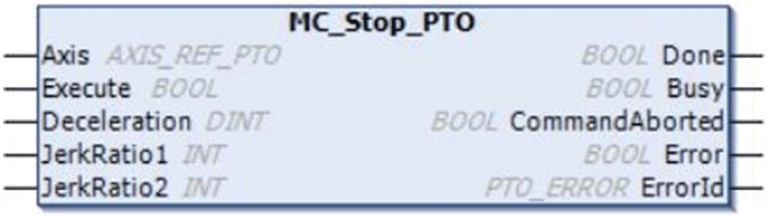
Description

Overview

This function block commands a controlled motion stop and transfers the axis to the state **Stopping**. It aborts any ongoing move execution and the move queue is cleared. While the axis is in state **Stopping**, no other function block can perform any motion on the same axis. This function block is primarily intended for exception situations, or fast stop functionality.

MC_Stop_PTO: Command a Controlled Motion Stop

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`Deceleration`	DINT	20	Deceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...`MaxDecelerationAppl` *(see page 85)* Range (ms): `MaxDecelerationAppl` *(see page 85)*...100,000
`JerkRatio1`	INT	0	Percentage of deceleration from constant velocity used to create the S-curve profile *(see page 45)*.
`JerkRatio2`	INT	0	Percentage of deceleration to standstill used to create the S-curve profile *(see page 45)*.

Output Variables

This table describes the output variables:

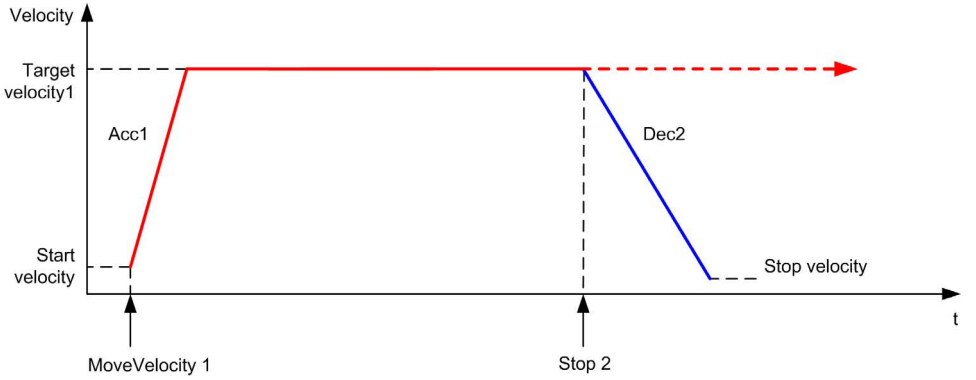
Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected .
`Busy`	BOOL	FALSE	If TRUE, indicates that the function block execution is in progress.
`CommandAborted`	BOOL	FALSE	Function block execution is finished, by aborting due to another move command or an error detected .
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: type of the error detected *(see page 86)*.

NOTE:

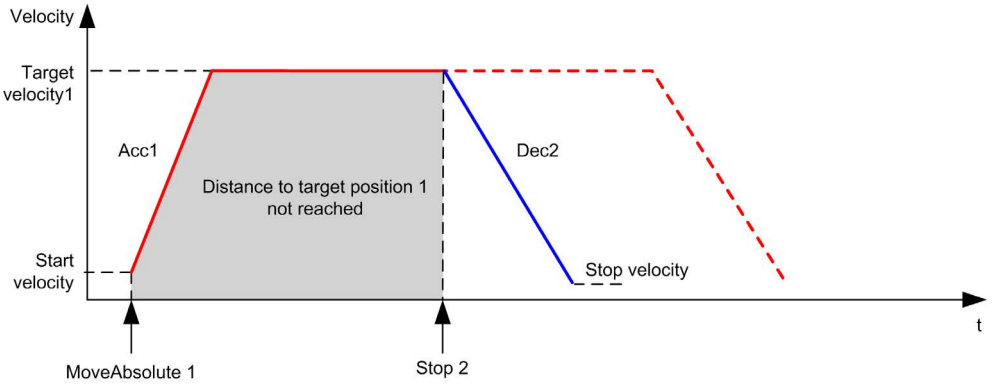
- Calling this function block in state **Standstill** changes the state to **Stopping**, and back to **Standstill** when `Execute` is FALSE.
- The state **Stopping** is kept as long as the input `Execute` is true.
- The`Done` output is set when the stop ramp is finished.
- If `Deceleration` = 0, the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



Section 6.9
MC_Halt_PTO Function Block

Overview

This section describes the `MC_Halt_PTO` function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description	142
MC_Halt_PTO: Command a Controlled Motion Stop until the Velocity equals Zero	143

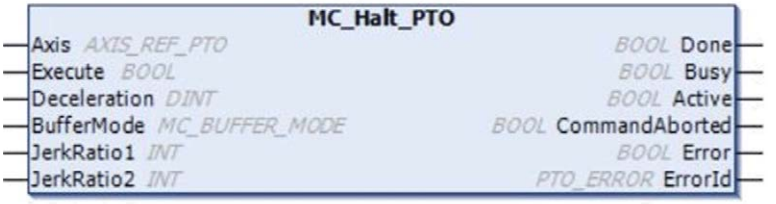
Description

Overview

This function block commands a controlled motion stop until the velocity is zero, and transfers the axis to the state **Discrete**. With the Done output set, the state is transferred to **Standstill**.

MC_Halt_PTO: Command a Controlled Motion Stop until the Velocity equals Zero

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`Deceleration`	DINT	20	Deceleration in Hz/ms or in ms (according to configuration). Range (Hz/ms): 1...`MaxDecelerationAppl` *(see page 85)* Range (ms): `MaxDecelerationAppl` *(see page 85)*...100,000
`BufferMode`	MC_BUFFER_MODE	`mcAborting`	Transition mode from ongoing move *(see page 81)*.
`JerkRatio1`	INT	0	Percentage of deceleration from constant velocity used to create the S-curve profile *(see page 45)*.
`JerkRatio2`	INT	0	Percentage of deceleration to standstill used to create the S-curve profile *(see page 45)*.

Output Variables

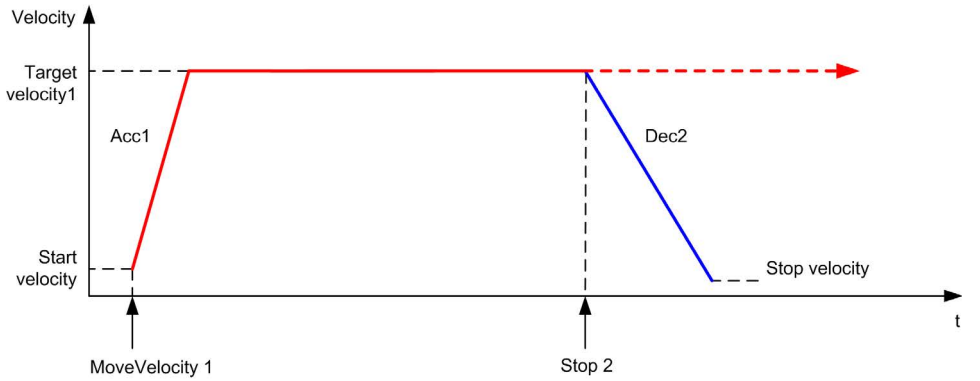
This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected .
`Busy`	BOOL	FALSE	If TRUE, indicates that the function block execution is in progress.
`Active`	BOOL	FALSE	The function block controls the `Axis`. Only one function block at a time can set `Active` TRUE for a defined `Axis`.
`CommandAborted`	BOOL	FALSE	Function block execution is finished, by aborting due to another move command or an error detected .
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: type of the error detected *(see page 86)*.

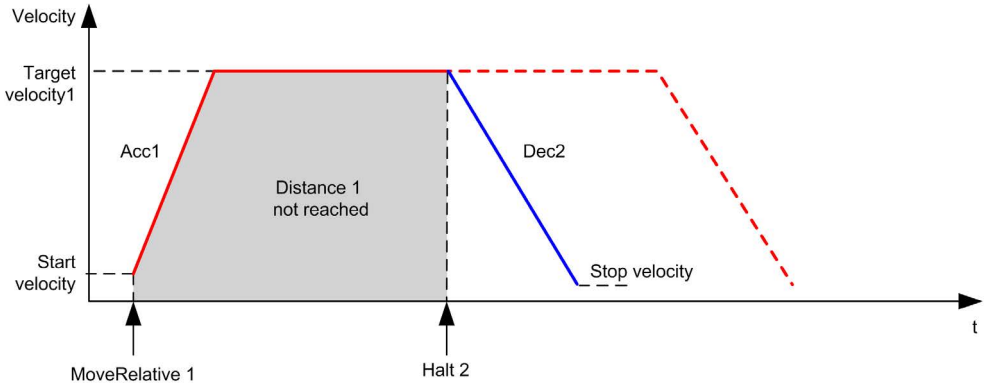
NOTE: The function block completes with velocity zero.

Timing Diagram Example

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



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



Section 6.10
Adding a Motion Function Block

Adding a Motion Function Block

Procedure

Follow these steps to add and create the instance of a motion function block:

Step	Action
1	Add a POU *(see EcoStruxure Machine Expert, Programming Guide)* in the **Applications tree**.
2	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 PTOPWM → PTO → Motion → MC_xxxxxx_PTO** in the list, drag-and-drop the item onto the **POU** window.
3	Create the function block instance by clicking: 
4	Associate the input/output variables *(see page 89)* of the function block.

Chapter 7
Administrative Function Blocks

Overview

This chapter describes the administrative function blocks.

Administrative function blocks do not influence the state diagram *(see page 91)*.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	Status Function Blocks	148
7.2	Parameters Function Blocks	157
7.3	Probe Function Blocks	166
7.4	Error Handling Function Blocks	170
7.5	Adding an Administrative Function Block	174

Section 7.1
Status Function Blocks

Overview

This section describes the status function blocks.

What Is in This Section?

This section contains the following topics:

Topic	Page
MC_ReadActualVelocity_PTO: Get the Commanded Velocity of the Axis	149
MC_ReadActualPosition_PTO: Get the Position of the Axis	151
MC_ReadStatus_PTO: Get the State of the Axis	153
MC_ReadMotionState_PTO: Get the Motion Status of the Axis	155

MC_ReadActualVelocity_PTO: Get the Commanded Velocity of the Axis

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration,.
`Enable`	BOOL	FALSE	When TRUE, the function block is executed. The values of the function block inputs can be modified continuously, and the outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Valid`	BOOL	FALSE	Valid data is available at the function block output pin.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`Velocity`	DINT	0	Actual velocity of the axis (in Hz).

MC_ReadActualPosition_PTO: Get the Position of the Axis

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration,.
`Enable`	BOOL	FALSE	When TRUE, the function block is executed. The values of the function block inputs can be modified continuously, and the outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

Output Variables

This table describes the output variables:

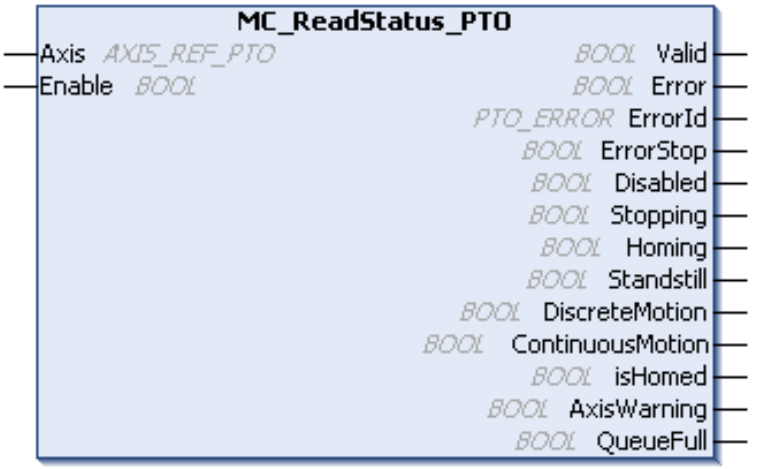
Output	Type	Initial Value	Description
`Valid`	BOOL	FALSE	Valid data is available at the function block output pin.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`Position`	DINT	0	Actual position of the axis.

MC_ReadStatus_PTO: Get the State of the Axis

Function Block Description

This function block returns the state diagram *(see page 91)* status of the axis.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
Axis	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration,.
Enable	BOOL	FALSE	When TRUE, the function block is executed. The values of the function block inputs can be modified continuously, and the outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Valid`	BOOL	FALSE	The set of outputs is valid.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`ErrorStop`	BOOL	FALSE	If TRUE, the state is active (Motion state diagram *(see page 91)*).
`Disabled`	BOOL	FALSE	
`Stoppping`	BOOL	FALSE	
`Homing`	BOOL	FALSE	
`Stanstill`	BOOL	FALSE	
`DiscreteMotion`	BOOL	FALSE	
`ContinuousMotion`	BOOL	FALSE	
`IsHomed`	BOOL	FALSE	If TRUE, the reference point is valid, absolute motion is allowed.
`AxisWarning`	BOOL	FALSE	If TRUE, an alert is present on the axis (call `MC_ReadAxisError_PTO` *(see page 171)* for detailed information).
`QueueFull`	BOOL	FALSE	If TRUE, the motion queue is full, no additional move is allowed in the buffer.

MC_ReadMotionState_PTO: Get the Motion Status of the Axis

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration,.
`Enable`	BOOL	FALSE	When TRUE, the function block is executed. The values of the function block inputs can be modified continuously, and the outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Valid`	BOOL	FALSE	Valid data is available at the function block output pin.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`ConstantVelocity`	BOOL	FALSE	The actual velocity is constant.
`Accelerating`	BOOL	FALSE	The actual velocity is increasing.
`Decelerating`	BOOL	FALSE	The actual velocity is decreasing.

Section 7.2
Parameters Function Blocks

Overview

This section describes the parameters function blocks.

What Is in This Section?

This section contains the following topics:

Topic	Page
MC_ReadParameter_PTO: Get Parameters from the PTO	158
MC_WriteParameter_PTO: Write Parameters to the PTO	160
MC_ReadBoolParameter_PTO: Get B00L Parameters from the PTO	162
MC_WriteBoolParameter_PTO: Write B00L Parameters to the PTO	164

MC_ReadParameter_PTO: Get Parameters from the PTO

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration,.
`Enable`	BOOL	FALSE	When TRUE, the function block is executed. The values of the function block inputs can be modified continuously, and the outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.
`ParameterNumber`	INT	0	ID of the requested parameter (PTO_PARAMETER *(see page 85)*)

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Valid`	BOOL	FALSE	Valid data is available at the function block output pin.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`Value`	DINT	0	Value of the requested parameter.

MC_WriteParameter_PTO: Write Parameters to the PTO

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration,.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`ParameterNumber`	INT	0	ID of the requested parameter (PTO_PARAMETER *(see page 85)*)
`Value`	DINT	0	Value to be written to the requested parameter.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	`BOOL`	FALSE	If TRUE, indicates that the function block execution is finished with no error detected..
`Error`	`BOOL`	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	`PTO_ERROR`	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.

MC_ReadBoolParameter_PTO: Get BOOL Parameters from the PTO

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Enable`	BOOL	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.
`ParameterNumber`	INT	0	ID of the requested parameter (PTO_PARAMETER *(see page 85)*)

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Valid`	BOOL	FALSE	Valid data is available at the function block output pin.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`Value`	BOOL	FALSE	Value of the requested parameter.

MC_WriteBoolParameter_PTO: Write BOOL Parameters to the PTO

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`ParameterNumber`	INT	0	ID of the requested parameter (PTO_PARAMETER *(see page 85)*)
`Value`	BOOL	FALSE	Value to be written to the requested parameter.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.

Section 7.3
Probe Function Blocks

Overview

This section describes the probe function blocks.

What Is in This Section?

This section contains the following topics:

Topic	Page
MC_TouchProbe_PTO: Activate a Trigger Event	167
MC_AbortTrigger_PTO: Abort/Deactivate Function Blocks	169

MC_TouchProbe_PTO: Activate a Trigger Event

Function Block 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



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
`WindowOnly`	BOOL	FALSE	If TRUE, only use the window defined by `FirstPosition` and `LastPosition` to accept trigger events.
`FirstPosition`	DINT	0	Start absolute position from where (positive direction) trigger events are accepted (value included in window).
`LastPosition`	DINT	0	Stop absolute position until where (positive direction) trigger events are accepted (value included in window).
`TriggerLevel`	BOOL	FALSE	If FALSE, position capture at falling edge. If TRUE, position capture at rising edge.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected.
`Busy`	BOOL	FALSE	If TRUE, indicates that the function block execution is in progress.
`CommandAborted`	BOOL	FALSE	Function block execution is finished, by aborting due to another move command or a error detected.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`RecordedPosition`	DINT	0	Position where trigger event was detected.

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

MC_AbortTrigger_PTO: Abort/Deactivate Function Blocks

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.

Section 7.4
Error Handling Function Blocks

Overview

This section describes the error handling function blocks.

What Is in This Section?

This section contains the following topics:

Topic	Page
MC_ReadAxisError_PTO: Get the Axis Control Error	171
MC_Reset_PTO: Reset All Axis-Related Errors	173

MC_ReadAxisError_PTO: Get the Axis Control Error

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Enable`	BOOL	FALSE	When TRUE, the function block is executed. The values of the function block inputs can be modified continuously, and the outputs are updated continuously. When FALSE, terminates the function block execution and resets its outputs.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Valid`	BOOL	FALSE	Valid data is available at the function block output pin.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.
`AxisErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	Index 1000 of PTO_ERROR *(see page 86)*.

MC_Reset_PTO: Reset All Axis-Related Errors

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter Function and Function Block Representation *(see page 209)*.

Input Variables

This table describes the input variables:

Input	Type	Initial Value	Description
`Axis`	AXIS_REF_PTO	-	Name of the axis (instance) for which the function block is to be executed. In the devices tree, the name is declared in the controller configuration.
`Execute`	BOOL	FALSE	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.

Output Variables

This table describes the output variables:

Output	Type	Initial Value	Description
`Done`	BOOL	FALSE	If TRUE, indicates that the function block execution is finished with no error detected.
`Error`	BOOL	FALSE	If TRUE, indicates that an error was detected. Function block execution is finished.
`ErrorId`	PTO_ERROR	`PTO_ERROR.NoError`	When `Error` is TRUE: code of the error detected *(see page 86)*.

Section 7.5
Adding an Administrative Function Block

Adding an Administrative Function Block

Procedure

Follow these steps to add and create the instance of an administrative function block:

Step	Action
1	Add a POU *(see EcoStruxure Machine Expert, Programming Guide)* in the **Applications tree**.
2	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 PTOPWM → PTO → Administrative → MC_xxxxxx_PTO** in the list, drag-and-drop the item onto the **POU** window.
3	Create the function block instance by clicking: 
4	Associate the input/output variables *(see page 147)* of the function block.

Part III
Pulse Width Modulation (PWM)

Overview

This part describes the `Pulse Width Modulation` function.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
8	Introduction	177
9	Configuration and Programming	183
10	Data Types	191

Chapter 8
Introduction

Overview

This chapter provides a description of the PWM functions.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	178
FreqGen/PWM Naming Convention	180
Synchronization and Enable Functions	181

Description

Overview

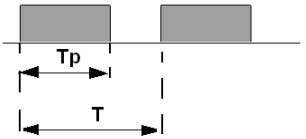
The pulse width modulation function generates a programmable pulse wave signal on a dedicated output with adjustable duty cycle and frequency.

Signal Form

The signal form depends on the following input parameters:

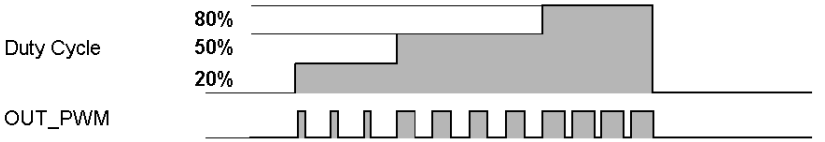
- **Frequency** configurable:
  - from 0.1 Hz to 20 kHz with a 0.1 Hz step (fast outputs: Q0...Q3)
  - from 0.1 Hz to 1 kHz with a 0.1 Hz step (regular outputs: Q4...Q7)
- **Duty Cycle** of the output signal from 0% to 100% with 1% step or 0.1% step with `HighPrecision`.

Duty Cycle=Tp/T

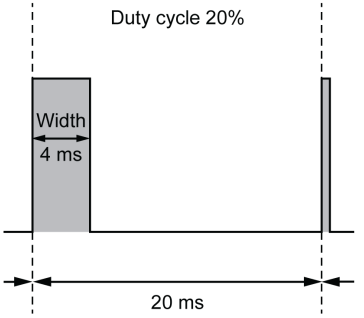


Tp pulse width
T pulse period (1/Frequency)

Modifying the duty cycle in the program modulates the width of the signal. Below is an illustration of an output signal with varying duty cycles.



The following illustration shows a duty cycle of 20%:



FreqGen/PWM Naming Convention

Definition

Frequency Generator and Pulse Width Modulation uses 1 fast physical output and up to 2 physical inputs.

In this document, use the following naming convention:

Name	Description
SYNC	Synchronization function *(see page 181)*.
EN	Enable function *(see page 181)*.
IN_SYNC	Physical input dedicated to the SYNC function.
IN_EN	Physical input dedicated to the EN function.
OUT_PWM	Physical output dedicated to the FreqGen or PWM.

Synchronization and Enable Functions

Introduction

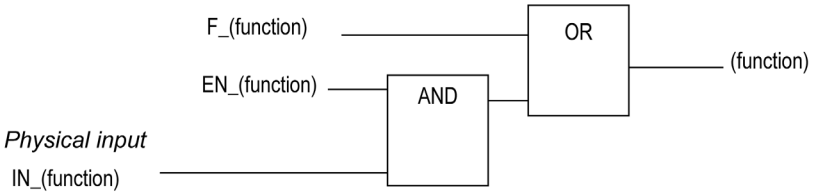
This section presents the functions used by the FreqGen/PWM:

- **Synchronization** function
- **Enable** function

Each function uses the 2 following function block bits:

- **EN_(function) bit:** Setting this bit to 1 allows the (function) to operate on an external physical input if configured.
- **F_(function) bit:** Setting this bit to 1 forces the (function).

The following diagram explains how the function is managed:



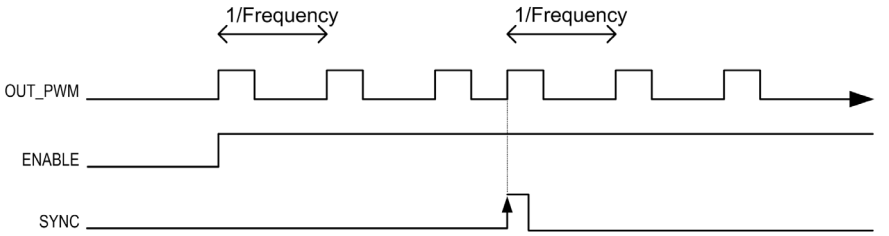
NOTE: (function) stands either for **Enable** (for Enable function) or **Sync** (for Synchronization function).
If the physical input is required, enable it in the configuration screen *(see page 184)*.

Synchronization Function

The **Synchronization** function is used to interrupt the current FreqGen/PWM cycle and then restart a new cycle.

Enable Function

The **Enable** function is used to activate the FreqGen/PWM:



Chapter 9
Configuration and Programming

Overview

This chapter provides configuration and programming guidelines for using PWM functions.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Configuration	184
PWM_M241: Command a Pulse Width Modulation Signal	187
Programming the PWM Function Block	189

Configuration

Overview

Four pulse width modulation functions can be configured on the controller.

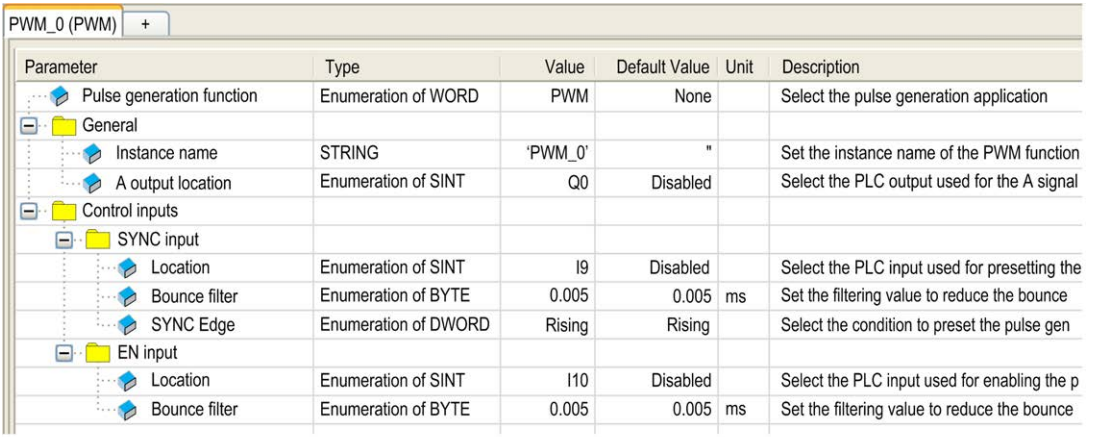
Adding a Pulse Width Modulation Function

To add a pulse width modulation function, proceed as follows:

Step	Action
1	Double-click the **Pulse Generators** node of your controller in the **Devices Tree**.
2	Double-click the **Pulse generation function** value and select **PWM**. **Result:** The PWM configuration parameters appear.

Parameters

The figure provides an example of a PWM configuration window:



The pulse width modulation function has the following parameters:

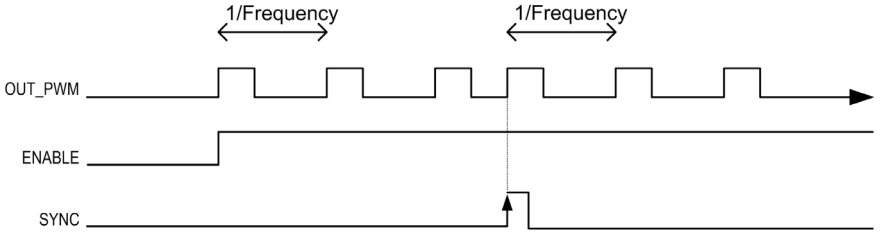
Parameter		Value	Default	Description
General	Instance name	-	PWM_0...PWM_3	Set the instance name of the PWM function.
	A output location	Disabled Q0...Q3 (fast outputs) Q4...Q7 (regular outputs)(1)	Disabled	Select the controller output used for the A signal.
Control inputs / SYNC input	Location	Disabled I0...I7 (fast inputs) I8...I13 (TM241•24• regular inputs) I8...I15 (TM241•40• regular inputs)	Disabled	Select the controller input used for presetting the PWM function.
	Bounce filter	0.000 0.001 0.002 0.005 0.010 0.1 1.5 1 5	0.005	Set the filtering value to reduce the bounce effect on the SYNC input (in ms).
	SYNC Edge	Rising Falling Both	Rising	Select the condition to preset the PWM function with the SYNC input.
(1) Not available for M241 Logic Controller references with relay outputs.				

Parameter		Value	Default	Description
Control inputs / EN input	Location	Disabled I0...I7 (fast inputs) I8...I15 (TM241•40• regular inputs) I8...I13 (TM241•24• regular inputs)	Disabled	Select the controller input used for enabling the PWM function.
	Bounce filter	0.000 0.001 0.002 0.005 0.010 0.1 1.5 1 5	0.005	Set the filtering value to reduce the bounce effect on the EN input (in ms).
(1) Not available for M241 Logic Controller references with relay outputs.				

Synchronizing with an External Event

On a rising edge on the IN_SYNC physical input (with EN_Sync = 1), the current cycle is interrupted and the PWM restarts a new cycle.

This illustration provides a pulse diagram for the `Pulse Width Modulation` function block with use of IN_SYNC input:



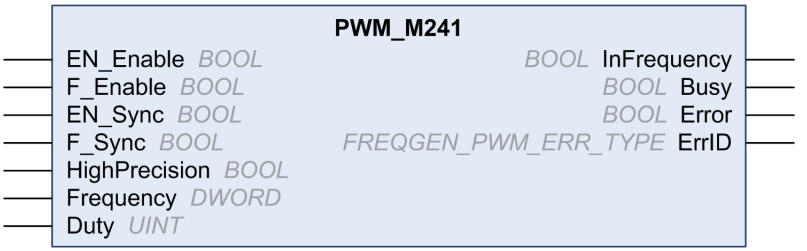
PWM_M241: Command a Pulse Width Modulation Signal

Overview

The `Pulse Width Modulation` function block commands a pulse width modulated signal output at the specified frequency and duty cycle.

Graphical Representation

This illustration is a `Pulse Width Modulation` function block:



IL and ST Representation

To see the general representation in IL or ST language, refer to the *Differences Between a Function and a Function Block (see page 210)* chapter.

Input Variables

This table describes the input variables:

Inputs	Type	Comment
`EN_Enable`	`BOOL`	`TRUE` = authorizes the PWM enable via the `IN_EN` input (if configured).
`F_Enable`	`BOOL`	`TRUE` = enables the `Pulse Width Modulation`.
`EN_SYNC`	`BOOL`	`TRUE` = authorizes the restart via the `IN_Sync` input of the internal timer relative to the time base (if configured).
`F_SYNC`	`BOOL`	On a rising edge, forces a restart of the internal timer relative to the time base.
`HighPrecision`	`BOOL`	If `FALSE` (the default), the duty cycle is specified in units of 1%. See `Duty` below. If `TRUE`, the duty cycle *(see page 178)* is specified in units of 0.1%. **NOTE:** The value of the `Duty` parameter is automatically updated to 0...100 or 0...1000 according to the value selected.

Inputs	Type	Comment
Frequency	DWORD	Frequency of the Pulse Width Modulation output signal in tenths of Hz (range: 1 (0.1 Hz)...200,000 (20 kHz)).
Duty	UINT	Duty cycle of the Pulse Width Modulation output signal, in units of 1% (range: 0...100 (0%...100%)). **NOTE:** If the HighPrecision input is set to TRUE, the duty cycle is in units of 0.1% (range: 0...1000 (0%...100%)).

Output Variables

This table describes the output variables:

Outputs	Type	Comment
InFrequency	BOOL	TRUE = the Pulse Width Modulation signal is currently being output at the specified frequency and duty cycle. FALSE = ● The required frequency cannot be reached for any reason. ● F_Enable is set to False. ● EN_Enable is set to False or no signal detected on the physical input **EN Input** (if configured).
Busy	BOOL	Busy is used to indicate that a command change is in progress: the frequency is changed. Set to TRUE when the Enable command is set and the frequency or duty is changed. Reset to FALSE when InFrequency or Error is set, or when the Enable command is reset.
Error	BOOL	TRUE = indicates that an error was detected.
ErrID	FREQGEN_PWM_ERR_TYPE *(see page 191)*	When Error is set: type of the detected error.

NOTE: When the required frequency cannot be reached for any reason, the InFrequency output is not set to TRUE, but Error stays to FALSE.

Programming the PWM Function Block

Procedure

Follow these steps to program a **PWM** function block:

Step	Action
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 PTOPWM → PWM → PWM_M241** in the list, drag-and-drop the item onto the **POU** window.
2	Select the function block instance by clicking . The **Input Assistant** dialog is displayed. Select the global variable which references to the added PWM *(see page 184)* during the configuration and confirm. **NOTE:** If the function block instance is not visible, verify if the PWM is configured.
3	The inputs/outputs are detailed in the function block *(see page 187)*.

Chapter 10
Data Types

FREQGEN_PWM_ERR_TYPE

Error Type Enumeration

This table lists the values for the FREQGEN_PWM_ERR_TYPE enumeration:

Enumerator	Value	Description
FREQGEN_PWM_NO_ERROR	0	No error detected.
FREQGEN_PWM_UNKNOWN_REF	1	The reference to the FreqGen / PWM is not valid.
FREQGEN_PWM_UNKNOWN_PARAMETER	2	The parameter type is unknown in the current mode.
FREQGEN_PWM_INVALID_PARAMETER	3	A parameter value is not valid or the combination of parameter values is not valid.
FREQGEN_PWM_COM_ERROR	4	Communication error with the FreqGen / PWM.
FREQGEN_PWM_AXIS_ERROR	5	PWM is in error state (“PWMError" is set on PTOSimple instance). No move is possible until the error is reset.

Part IV
Frequency Generator (FreqGen)

Overview

This part describes the `Frequency Generator` function.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
11	Introduction	195
12	Configuration and Programming	199

Chapter 11
Introduction

Overview

This chapter provides a description of the `FreqGen` functions.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Description	196
FreqGen Naming Convention	197
Synchronization and Enable Functions	198

Description

Overview

The frequency generator function generates a square wave signal on dedicated output channels with a fixed duty cycle (50%).

Frequency is configurable from 0.1 Hz to 100 kHz with a 0.1 Hz step.

FreqGen Naming Convention

Description

FreqGen/PWM Naming Convention *(see page 180)*

Synchronization and Enable Functions

Description

Synchronization and Enable Functions *(see page 181)*

Chapter 12
Configuration and Programming

Overview

This chapter provides configuration and programming guidelines for using `FreqGen` functions.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Configuration	200
FrequencyGenerator_M241: Commanding a Square Wave Signal	203
Programming	205

Configuration

Overview

Up to 4 frequency generator functions can be configured on the controller.

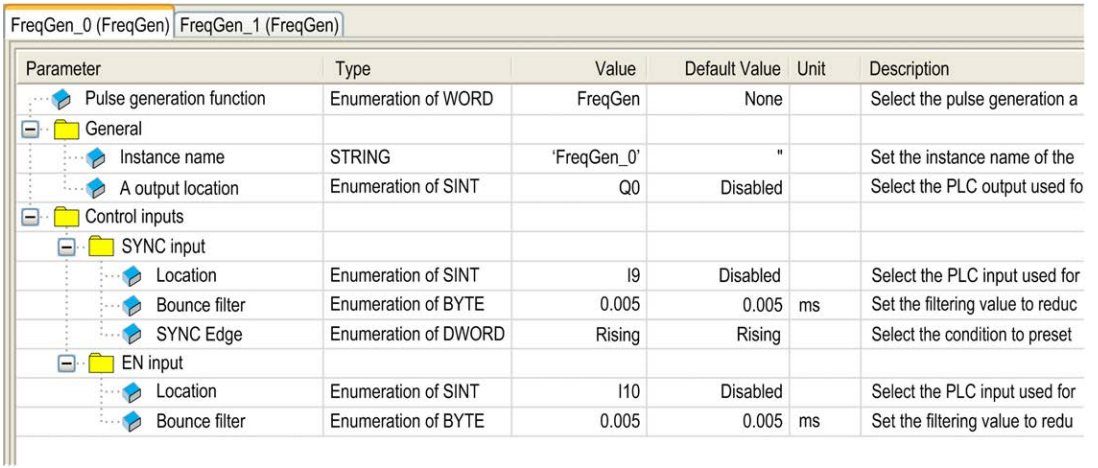
Adding a Frequency Generator Function

To add a frequency generator function, proceed as follows:

Step	Action
1	Double-click the **Pulse Generators** node of your controller in the **Devices Tree**.
2	Double-click the **Pulse generation function** value and select **FreqGen**. **Result:** The frequency generator configuration parameters are displayed.

Parameters

The figure provides an example of a frequency generator configuration window:



Parameter	Type	Value	Default Value	Unit	Description
Pulse generation function	Enumeration of WORD	FreqGen	None		Select the pulse generation a
General					
Instance name	STRING	'FreqGen_0'	''		Set the instance name of the
A output location	Enumeration of SINT	Q0	Disabled		Select the PLC output used fo
Control inputs					
SYNC input					
Location	Enumeration of SINT	I9	Disabled		Select the PLC input used for
Bounce filter	Enumeration of BYTE	0.005	0.005	ms	Set the filtering value to reduc
SYNC Edge	Enumeration of DWORD	Rising	Rising		Select the condition to preset
EN input					
Location	Enumeration of SINT	I10	Disabled		Select the PLC input used for
Bounce filter	Enumeration of BYTE	0.005	0.005	ms	Set the filtering value to redu

The frequency generator function has the following parameters:

Parameter		Value	Default	Description
General	Instance name	-	FreqGen0...FreqGen3	Set the instance name of the frequency generator function.
	A output location	Disabled Q0...Q3 (fast outputs) Q4...Q7 (regular outputs)(1)	Disabled	Select the controller output used for the A signal.
Control inputs / SYNC input	Location	Disabled I0...I7 (fast inputs) I8...I13 (TM241•24• regular inputs) I8...I15 (TM241•40• regular inputs)	Disabled	Select the controller input used for presetting the frequency generator function.
	Bounce filter	0.000 0.001 0.002 0.005 0.010 0.1 1.5 1 5	0.005	Set the filtering value to reduce the bounce effect on the SYNC input (in ms).
	SYNC Edge	Rising Falling Both	Rising	Select the condition to preset the frequency generator function with the SYNC input.

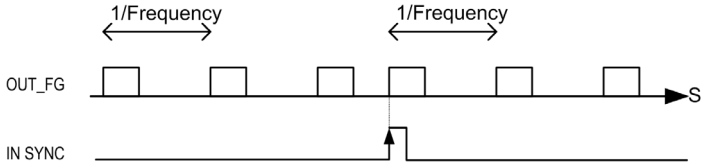
(1) Not available for M241 Logic Controller references with relay outputs.

Parameter		Value	Default	Description
Control inputs / EN input	Location	Disabled I0...I7 (fast inputs) I8...I15 (TM241•40• regular inputs) I8...I13 (TM241•24• regular inputs)	Disabled	Select the controller input used for enabling the frequency generator function.
	Bounce filter	0.000 0.001 0.002 0.005 0.010 0.1 1.5 1 5	0.005	Set the filtering value to reduce the bounce effect on the EN input (in ms).
(1) Not available for M241 Logic Controller references with relay outputs.				

Synchronizing with an External Event

On a rising edge on the IN_SYNC physical input (with EN_Sync = 1), the current cycle is interrupted and the FreqGen restarts a new cycle.

This illustration provides a pulse diagram for the frequency generator function block with use of IN_SYNC input:



FrequencyGenerator_M241: Commanding a Square Wave Signal

Overview

The `Frequency Generator` function block commands a square wave signal output at the specified frequency.

Graphical Representation (LD/FBD)

This illustration is a `Frequency Generator` function block:



IL and ST Representation

To see the general representation in IL or ST language, refer to the *Differences Between a Function and a Function Block* *(see page 210)* chapter.

Input Variables

This table describes the input variables:

Inputs	Type	Comment
`EN_Enable`	`BOOL`	TRUE = authorizes the `Frequency Generator` enable via the IN_EN input (if configured).
`F_Enable`	`BOOL`	TRUE = enables the `Frequency Generator`.
`EN_SYNC`	`BOOL`	TRUE = authorizes the restart via the IN_SYNC input of the internal timer relative to the time base (if configured).
`F_SYNC`	`BOOL`	On rising edge, forces a restart of the internal timer relative to the time base.
`Frequency`	`DWORD`	Frequency of the `Frequency Generator` output signal in tenths of Hz. (Range: min 1 (0.1Hz)...max 1,000,000 (100kHz)

Output Variables

This table describes the output variables:

Outputs	Type	Comment
InFrequency	BOOL	TRUE = the Frequency Generator signal is output at the specified Frequency. FALSE = • The required frequency cannot be reached for any reason. • F_Enable is set to False. • EN_Enable is set to False or no signal detected on the physical input **EN Input** (if configured).
Busy	BOOL	Busy is used to indicate that a command change is in progress: the frequency is changed. Set to TRUE when the Enable command is set and the frequency is changed. Reset to FALSE when InFrequency or Error is set, or when the Enable command is reset.
Error	BOOL	TRUE = indicates that an error was detected.
ErrID	FREQGEN_PWM_ERR_TYPE *(see page 191)*	When Error is set: type of the detected error.

NOTE: When the required frequency cannot be reached for any reason, the InFrequency output is not set to TRUE, but Error stays to FALSE.

NOTE: Outputs are forced to 0 when the logic controller is in the STOPPED state.

Programming

Procedure

Follow these steps to program a `Frequency Generator` function block:

Step	Action
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → M241 → M241 PTOPWM → Frequency Generator → FrequencyGenerator_M241** in the list; drag-and-drop the item onto the **POU** window.
2	Select the function block instance by clicking . The Input Assistant screen appears. Select the global variable which references to the added FreqGen *(see page 200)* during the configuration and confirm. **NOTE:** If the function block instance is not visible, verify if the frequency generator is configured.
3	The inputs/outputs are detailed in the function block *(see page 203)*.

Appendices



Appendix A
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Differences Between a Function and a Function Block	210
How to Use a Function or a Function Block in IL Language	211
How to Use a Function or a Function Block in ST Language	215

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

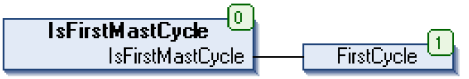
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

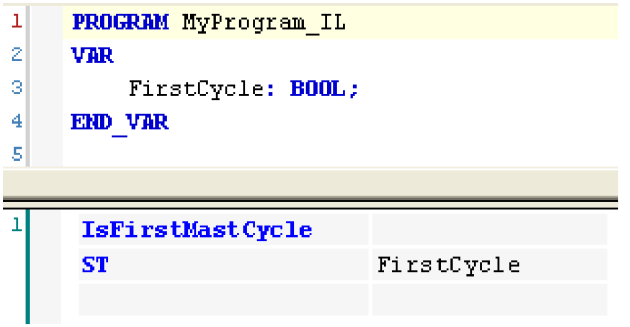
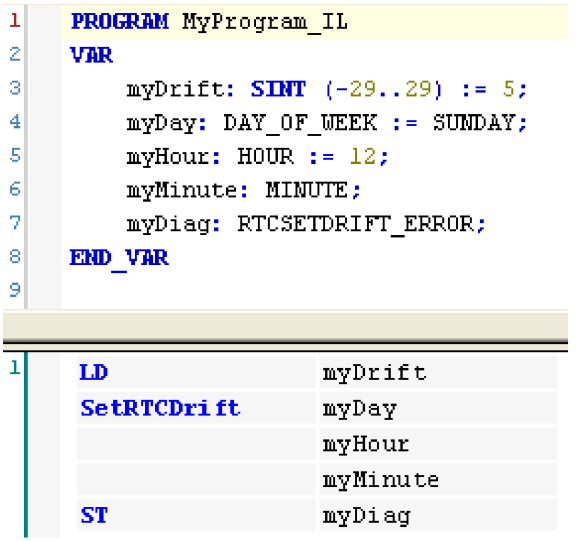
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

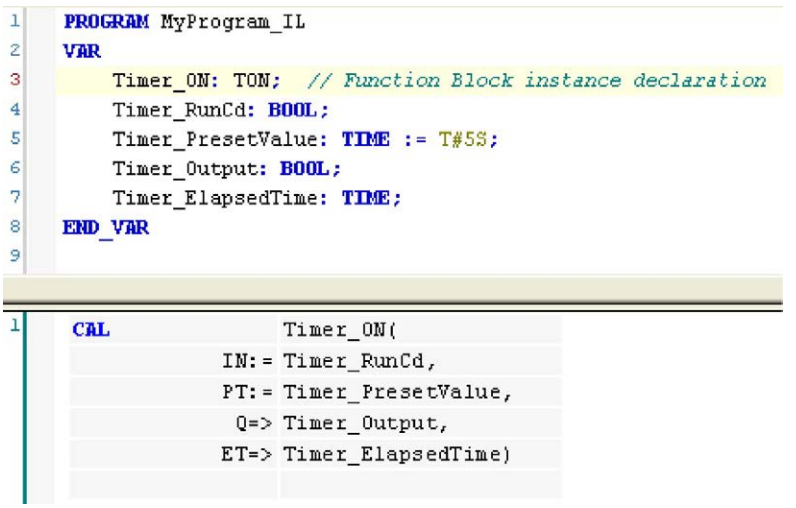
This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a CAL instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the TON Function Block graphically presented below:

Function Block	Graphical Representation
TON	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	

```
PROGRAM MyProgram_IL
VAR
    Timer_ON: TON;  // Function Block instance declaration
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
CAL             Timer_ON(
            IN:= Timer_RunCd,
            PT:= Timer_PresetValue,
             Q=> Timer_Output,
            ET=> Timer_ElapsedTime)
```

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

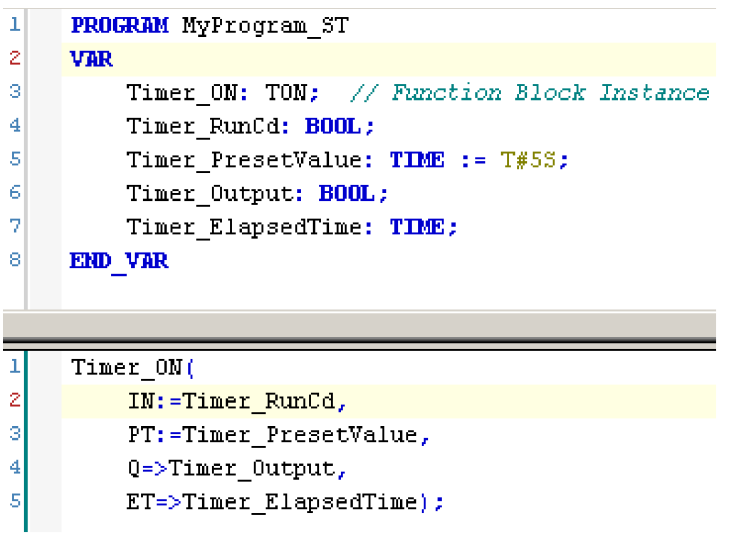
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

A

absolute movement

A movement to a position defined from a reference point.

acceleration / deceleration

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

application

A program including configuration data, symbols, and documentation.

B

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

function

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

Examples: boolean (AND) operators, calculations, conversions (BYTE_TO_INT)

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

H

homing

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

I

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).

INT

(*integer*) A whole number encoded in 16 bits.

J

jerk ratio

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

L

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).

P

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

S

S-curve ramp

An acceleration / deceleration ramp with a `JerkRatio` parameter greater than 0%.

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

start velocity

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

stop velocity

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

T

trapezoidal ramp

An acceleration / deceleration ramp with a `JerkRatio` parameter set to 0%.

V

variable

A memory unit that is addressed and modified by a program.

Index

A

acceleration ramp, *44*
axis
  MC_AbortTrigger_PTO, *169*
  MC_Halt_PTO, *143*
  MC_Home_PTO, *130*
  MC_MoveAbsolute_PTO, *124*
  MC_MoveRelative_PTO, *117*
  MC_MoveVelocity_PTO, *111*
  MC_Power_PTO, *106*
  MC_ReadActualPosition_PTO, *151*
  MC_ReadActualVelocity_PTO, *149*
  MC_ReadAxisError_PTO, *171*
  MC_ReadBoolParameter_PTO, *162*
  MC_ReadMotionState_PTO, *155*
  MC_ReadParameter_PTO, *158*
  MC_ReadStatus_PTO, *153*
  MC_Reset_PTO, *173*
  MC_SetPosition_PTO, *135*
  MC_Stop_PTO, *138*
  MC_TouchProbe_PTO, *167*
  MC_WriteBoolParameter_PTO, *164*
  MC_WriteParameter_PTO, *160*
AXIS_REF_PTO, *80*

D

data unit types
  AXIS_REF_PTO, *80*
  FREQGEN_PWM_ERR_TYPE, *191*
  MC_BUFFER_MODE, *81*
  MC_DIRECTION, *83*
  PTO_ERROR, *86*
  PTO_HOMING_MODE, *84*
  PTO_PARAMETER, *85*
deceleration ramp, *44*
dedicated features, *27*

E

error handling
  ErrID, *28*
  Error, *28*

F

FreqGen
  FrequencyGenerator_M241, *203*
  programming FrequencyGenerator_M241, *205*
FREQGEN_PWM_ERR_TYPE, *191*
frequency generator
  configuration, *200*
  description, *196*
  programming FrequencyGenerator_M241, *205*
FrequencyGenerator_M241
  commanding a square wave signal , *203*
  programming, *205*
function blocks
  FrequencyGenerator_M241, *203*
  PWM_M241, *187*
functionalities
  PTO, *31*
functions
  differences between a function and a function block, *210*
  enable, *181*
  how to use a function or a function block in IL language, *211*
  how to use a function or a function block in ST language, *215*
  synchronization, *181*

J

JerkRatio, *44*

M

M241 PTOPWM
FrequencyGenerator_M241, *203*
MC_AbortTrigger_PTO, *169*
MC_Halt_PTO, *143*
MC_Home_PTO, *130*
MC_MoveAbsolute_PTO, *124*
MC_MoveRelative_PTO, *117*
MC_MoveVelocity_PTO, *111*
MC_Power_PTO, *106*
MC_ReadActualPosition_PTO, *151*
MC_ReadActualVelocity_PTO, *149*
MC_ReadAxisError_PTO, *171*
MC_ReadBoolParameter_PTO, *162*
MC_ReadMotionState_PTO, *155*
MC_ReadParameter_PTO, *158*
MC_ReadStatus_PTO, *153*
MC_Reset_PTO, *173*
MC_SetPosition_PTO, *135*
MC_Stop_PTO, *138*
MC_TouchProbe_PTO, *167*
MC_WriteBoolParameter_PTO, *164*
MC_WriteParameter_PTO, *160*
programming FrequencyGenerator_M241, *205*
programming PWM_M241, *189*
management of status variables
Busy, *28*
CommandAborted, *28*
Done, *28*
ErrID, *28*
Error, *28*
Execute, *28*
MC_AbortTrigger_PTO
aborting or deactivating PTO function blocks, *169*
MC_BUFFER_MODE, *81*
MC_DIRECTION, *83*
MC_Halt_PTO
commanding a controlled PTO motion halt, *143*
MC_Home_PTO
commanding the axis to move to a reference position, *130*
MC_MoveAbsolute_PTO
commanding the axis to absolute position, *124*
MC_MoveRelative_PTO
commanding the relative axis movement, *117*
MC_MoveVelocity_PTO
controlling the speed of the axis, *111*
MC_Power_PTO
managing the power of the axis state, *106*
MC_ReadActualPosition_PTO
getting the position of the axis, *151*
MC_ReadActualVelocity_PTO
getting the velocity of the axis, *149*
MC_ReadAxisError_PTO
getting the axis control error, *171*
MC_ReadBoolParameter_PTO
getting boolean parameters from the PTO, *162*
MC_ReadMotionState_PTO
getting the motion status of the axis, *155*
MC_ReadParameter_PTO
getting parameters from the PTO, *158*
MC_ReadStatus_PTO
getting the motion status of the axis, *153*
MC_Reset_PTO
resetting axis-related errors, *173*
MC_SetPosition_PTO
forcing the reference position of the axis, *135*
MC_Stop_PTO
commanding a controlled motion stop, *138*
MC_TouchProbe_PTO
activating a trigger event on the PTO probe input, *167*
MC_WriteBoolParameter_PTO
setting boolean parameters to the PTO, *164*
MC_WriteParameter_PTO
setting parameters to the PTO, *160*

P

Programming
    PWM, *189*
PTO
    configuration, *37*
    functionalities, *31*
    MC_AbortTrigger_PTO, *169*
    MC_Halt_PTO, *143*
    MC_Home_PTO, *130*
    MC_MoveAbsolute_PTO, *124*
    MC_MoveRelative_PTO, *117*
    MC_MoveVelocity_PTO, *111*
    MC_Power_PTO, *106*
    MC_ReadActualPosition_PTO, *151*
    MC_ReadActualVelocity_PTO, *149*
    MC_ReadAxisError_PTO, *171*
    MC_ReadBoolParameter_PTO, *162*
    MC_ReadMotionState_PTO, *155*
    MC_ReadParameter_PTO, *158*
    MC_ReadStatus_PTO, *153*
    MC_Reset_PTO, *173*
    MC_SetPosition_PTO, *135*
    MC_Stop_PTO, *138*
    MC_TouchProbe_PTO, *167*
    MC_WriteBoolParameter_PTO, *164*
    MC_WriteParameter_PTO, *160*
PTO_ERROR, *86*
PTO_HOMING_MODE, *84*
PTO_PARAMETER, *85*
pulse width modulation
    configuration, *184*
    description, *178*
    programming PWM_M241, *189*
    PWM_M241, *187*
PWM
    programming PWM_M241, *189*
    PWM_M241, *187*
PWM_M241
    commanding a pulse width modulation signal, *187*
    programming, *189*

Modicon M241 Logic Controller

Hardware Guide

12/2019



EIO0000003083.01

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.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

© 2019 Schneider Electric. All rights reserved.

Table of Contents

	Safety Information	7
	About the Book	9
Part I	**Modicon M241 Logic Controller Introduction**	15
Chapter 1	**M241 General Overview**	17
	M241 Logic Controller Description	18
	Maximum Hardware Configuration	23
	TMC4 Cartridges	26
	TM2 Expansion Modules	27
	TM3 Expansion Modules	31
	TM3 Bus Couplers	40
	TM4 Expansion Modules	41
	TM5 Fieldbus Interfaces	42
	TM5 CANopen Fieldbus Interfaces	43
	TM7 CANopen Fieldbus Interfaces	44
	Accessories	45
Chapter 2	**M241 Features**	47
	Real Time Clock (RTC)	48
	Input Management	52
	Output Management	56
	Run/Stop	61
	SD Card	63
Chapter 3	**M241 Installation**	67
3.1	M241 Logic Controller General Rules for Implementing	68
	Environmental Characteristics	69
	Certifications and Standards	72
3.2	M241 Logic Controller Installation	73
	Installation and Maintenance Requirements	74
	M241 Logic Controller Mounting Positions and Clearances	77
	Top Hat Section Rail (DIN rail)	80
	Installing and Removing the Controller with Expansions	84
	Direct Mounting on a Panel Surface	86

3.3 M241 Electrical Requirements 87
Wiring Best Practices 88
DC Power Supply Characteristics and Wiring 94
AC Power Supply Characteristics and Wiring 98
Grounding the M241 System 100
Part II Modicon M241 Logic Controller 105
Chapter 4 TM241C24R 107
TM241C24R Presentation 107
Chapter 5 TM241CE24R 113
TM241CE24R Presentation 113
Chapter 6 TM241CEC24R 119
TM241CEC24R Presentation 119
Chapter 7 TM241C24T 125
TM241C24T Presentation 125
Chapter 8 TM241CE24T 131
TM241CE24T Presentation 131
Chapter 9 TM241CEC24T 137
TM241CEC24T Presentation 137
Chapter 10 TM241C24U 143
TM241C24U Presentation 143
Chapter 11 TM241CE24U 149
TM241CE24U Presentation 149
Chapter 12 TM241CEC24U 155
TM241CEC24U Presentation 155
Chapter 13 TM241C40R 161
TM241C40R Presentation 161
Chapter 14 TM241CE40R 167
TM241CE40R Presentation 167
Chapter 15 TM241C40T 173
TM241C40T Presentation 173
Chapter 16 TM241CE40T 179
TM241CE40T Presentation 179
Chapter 17 TM241C40U 185
TM241C40U Presentation 185
Chapter 18 TM241CE40U 191
TM241CE40U Presentation 191

Chapter 19 Embedded I/O Channels 197
Digital Inputs 198
Relay Outputs 206
Regular Transistor Outputs 212
Fast Transistor Outputs 218
Part III Modicon M241 Logic Controller Communication . . . 225
Chapter 20 Integrated Communication Ports 227
CAN Port 228
Ethernet Port 232
USB Mini-B Programming Port 235
Serial Line 1 237
Serial Line 2 240
Chapter 21 Connecting the M241 Logic Controller to a PC 243
Connecting the Controller to a PC 243
Glossary 247
Index 253

Safety Information

Important Information

NOTICE

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



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



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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

QUALIFICATION OF PERSONNEL

Only appropriately trained persons who are familiar with and understand the contents of this manual and all other pertinent product documentation are authorized to work on and with this product.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

INTENDED USE

The products described or affected by this document, together with software, accessories, and options, are programmable logic controllers (referred to herein as “logic controllers”), intended for industrial use according to the instructions, directions, examples, and safety information contained in the present document and other supporting documentation.

The product may only be used in compliance with all applicable safety regulations and directives, the specified requirements, and the technical data.

Prior to using the product, you must perform a risk assessment in view of the planned application. Based on the results, the appropriate safety-related measures must be implemented.

Since the product is used as a component in an overall machine or process, you must ensure the safety of persons by means of the design of this overall system.

Operate the product only with the specified cables and accessories. Use only genuine accessories and spare parts.

Any use other than the use explicitly permitted is prohibited and can result in unanticipated hazards.

About the Book

At a Glance

Document Scope

Use this document to:

- Install and operate your M241 Logic Controller.
- Connect the M241 Logic Controller to a programming device equipped with EcoStruxure Machine Expert software.
- Interface the M241 Logic Controller with I/O expansion modules, HMI, and other devices.
- Familiarize yourself with the M241 Logic Controller features.

NOTE: Read and understand this document and all related documents *(see page 10)* before installing, operating, or maintaining your controller.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.2.

For product compliance and environmental information (RoHS, REACH, PEP, EOLI, etc.), go to *www.schneider-electric.com/green-premium*.

The technical characteristics of the devices described in this manual also appear online (*https://www.se.com/*).

The characteristics that are described in the present document 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 document and online information, use the online information as your reference.

Related Documents

Title of Documentation	Reference Number
Modicon M241 Logic Controller - Programming Guide	*EIO0000003059 (ENG)* *EIO0000003060 (FRE)* *EIO0000003061 (GER)* *EIO0000003062 (SPA)* *EIO0000003063 (ITA)* *EIO0000003064 (CHS)*
Modicon TMC4 Cartridges - Hardware Guide	*EIO0000003113 (ENG)* *EIO0000003114 (FRE)* *EIO0000003115 (GER)* *EIO0000003116 (SPA)* *EIO0000003117 (ITA)* *EIO0000003118 (CHS)*
Modicon TM4 Expansion Modules - Hardware Guide	*EIO0000003155 (ENG)* *EIO0000003156 (FRE)* *EIO0000003157 (GER)* *EIO0000003158 (SPA)* *EIO0000003159 (ITA)* *EIO0000003160 (CHS)*
Modicon TM3 Digital I/O Modules - Hardware Guide	*EIO0000003125 (ENG)* *EIO0000003126 (FRE)* *EIO0000003127 (GER)* *EIO0000003128 (SPA)* *EIO0000003129 (ITA)* *EIO0000003130 (CHS)* *EIO0000003425 (TUR)* *EIO0000003424 (POR)*
Modicon TM3 Analog I/O Modules - Hardware Guide	*EIO0000003131 (ENG)* *EIO0000003132 (FRE)* *EIO0000003133 (GER)* *EIO0000003134 (SPA)* *EIO0000003135 (ITA)* *EIO0000003136 (CHS)* *EIO0000003427 (TUR)* *EIO0000003426 (POR)*

Title of Documentation	Reference Number
Modicon TM3 Expert I/O Modules - Hardware Guide	*EIO0000003137 (ENG)* *EIO0000003138 (FRE)* *EIO0000003139 (GER)* *EIO0000003140 (SPA)* *EIO0000003141 (ITA)* *EIO0000003142 (CHS)* *EIO0000003429 (TUR)* *EIO0000003428 (POR)*
Modicon TM3 Safety Modules - Hardware Guide	*EIO0000003353 (ENG)* *EIO0000003354 (FRE)* *EIO0000003355 (GER)* *EIO0000003356 (SPA)* *EIO0000003357 (ITA)* *EIO0000003358 (CHS)* *EIO0000003359 (POR)* *EIO0000003360 (TUR)*
Modicon TM3 Transmitter and Receiver Modules - Hardware Guide	*EIO0000003143 (ENG)* *EIO0000003144 (FRE)* *EIO0000003145 (GER)* *EIO0000003146 (SPA)* *EIO0000003147 (ITA)* *EIO0000003148 (CHS)* *EIO0000003431 (TUR)* *EIO0000003430 (POR)*
Modicon TM3 Bus Coupler - Hardware Guide	*EIO0000003635 (ENG)* *EIO0000003636 (FRE)* *EIO0000003637 (GER)* *EIO0000003638 (SPA)* *EIO0000003639 (ITA)* *EIO0000003640 (CHS)* *EIO0000003641 (POR)* *EIO0000003642 (TUR)*
Modicon TM5 Fieldbus Interface - Hardware Guide	*EIO0000003715 (ENG)* *EIO0000003716 (FRE)* *EIO0000003717 (GER)* *EIO0000003718 (SPA)* *EIO0000003719 (ITA)* *EIO0000003720 (CHS)*
M241 DC Logic Controller - Instruction Sheet	*HRB59603*
M241 AC Logic Controller - Instruction Sheet	*EAV48551*

You can download these technical publications and other technical information from our website at https://www.se.com/ww/en/download/ .

Product Related Information

⚠ ⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

⚠ DANGER

POTENTIAL FOR EXPLOSION

- Only use this equipment in non-hazardous locations, or in locations that comply with Class I, Division 2, Groups A, B, C and D.
- Do not substitute components which would impair compliance to Class I, Division 2.
- Do not connect or disconnect equipment unless power has been removed or the location is known to be non-hazardous.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.

Standard	Description
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Part I
Modicon M241 Logic Controller Introduction

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	M241 General Overview	17
2	M241 Features	47
3	M241 Installation	67

Chapter 1
M241 General Overview

Overview

This chapter provides general information about the M241 Logic Controller system architecture and its components.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
M241 Logic Controller Description	18
Maximum Hardware Configuration	23
TMC4 Cartridges	26
TM2 Expansion Modules	27
TM3 Expansion Modules	31
TM3 Bus Couplers	40
TM4 Expansion Modules	41
TM5 Fieldbus Interfaces	42
TM5 CANopen Fieldbus Interfaces	43
TM7 CANopen Fieldbus Interfaces	44
Accessories	45

M241 Logic Controller Description

Overview

The M241 Logic Controller has various powerful features and can service a wide range of applications.

Software configuration, programming, and commissioning is accomplished with the EcoStruxure Machine Expert software described in detail in the EcoStruxure Machine Expert Programming Guide *(see EcoStruxure Machine Expert, Programming Guide)* and the M241 Logic Controller Programming Guide *(see Modicon M241 Logic Controller, Programming Guide)*.

Programming Languages

The M241 Logic Controller is configured and programmed with the EcoStruxure Machine Expert software, which supports the following IEC 61131-3 programming languages:

- IL: Instruction List
- ST: Structured Text
- FBD: Function Block Diagram
- SFC: Sequential Function Chart
- LD: Ladder Diagram

EcoStruxure Machine Expert software can also be used to program these controllers using CFC (Continuous Function Chart) language.

Power Supply

The power supply of the M241 Logic Controller is 24 Vdc *(see page 94)* or 100...240 Vac *(see page 98)*.

Real Time Clock

The M241 Logic Controller includes a Real Time Clock (RTC) system *(see page 48)*.

Run/Stop

The M241 Logic Controller can be operated externally by the following:

- A hardware Run/Stop switch *(see page 61)*.
- A Run/Stop *(see page 52)* operation by a dedicated digital input, defined in the software configuration. For more information, refer to Configuration of Digital Inputs *(see Modicon M241 Logic Controller, Programming Guide)*.
- An EcoStruxure Machine Expert software command.

Memory

This table describes the different types of memory:

Memory Type	Size	Used to
RAM	64 Mbytes, of which 8 Mbytes available for the application	execute the application.
Non-volatile	128 Mbytes	save the program and data in case of a power interruption.

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/source transistor outputs
- Fast sink/source transistor outputs associated with pulse generators
- Relay outputs

Removable Storage

The M241 Logic Controllers include an embedded SD card slot *(see page 63)*.

The main uses of the SD card are:

- Initializing the controller with a new application
- Updating the controller firmware
-
- Storing recipes files
- Receiving data logging files

Embedded Communication Features

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

- CANopen Master *(see page 228)*
- Ethernet *(see page 232)*
- USB Mini-B *(see page 235)*
- Serial Line 1 *(see page 237)*
- Serial Line 2 *(see page 240)*

Expansion Module and Bus Coupler Compatibility

Refer to the compatibility tables in the EcoStruxure Machine Expert - Compatibility and Migration User Guide.

M241 Logic Controller

Reference	Digital Inputs	Digital Outputs	Communication Ports	Terminal Type	Power supply
TM241C24R *(see page 107)*	6 regular inputs(1) 8 fast inputs (counters)(2)	6 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	100...240 Vac
TM241CE24R *(see page 191)*	6 regular inputs(1) 8 fast inputs (counters)(2)	6 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	100...240 Vac
TM241CEC24R *(see page 119)*	6 regular inputs(1) 8 fast inputs (counters)(2)	6 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 Ethernet port 1 CANopen master port 1 USB programming port	Removable screw terminal blocks	100...240 Vac
TM241C24T *(see page 125)*	6 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc
TM241CE24T *(see page 131)*	6 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc
TM241CEC24T *(see page 137)*	6 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port 1 CANopen master port	Removable screw terminal blocks	24 Vdc
TM241C24U *(see page 143)*	6 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc

(1) The regular inputs have a maximum frequency of 1 kHz.
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.
(3) The fast transistor outputs can be used either as regular transistor outputs, as reflex outputs for counting function (HSC), or as fast transistor outputs for pulse generator functions (FreqGen / PTO / PWM).

Reference	Digital Inputs	Digital Outputs	Communication Ports	Terminal Type	Power supply
TM241CE24U *(see page 149)*	6 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc
TM241CEC24U *(see page 155)*	6 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 6 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port 1 CANopen master port	Removable screw terminal blocks	24 Vdc
TM241C40R *(see page 161)*	16 regular inputs(1) 8 fast inputs (counters)(2)	12 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	100...240 Vac
TM241CE40R *(see page 167)*	16 regular inputs(1) 8 fast inputs (counters)(2)	12 2A relay outputs 4 source fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	100...240 Vac
TM241C40T *(see page 173)*	16 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc
TM241CE40T *(see page 179)*	16 regular inputs(1) 8 fast inputs (counters)(2)	Source outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc
TM241C40U *(see page 185)*	16 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port	Removable screw terminal blocks	24 Vdc

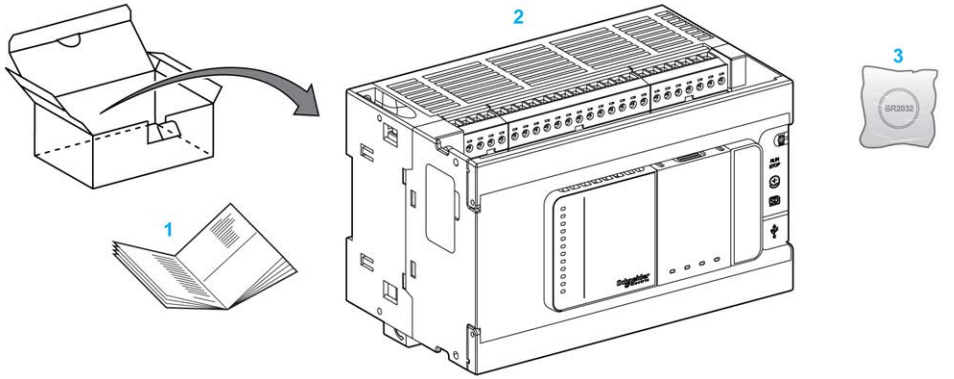
(1) The regular inputs have a maximum frequency of 1 kHz.
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.
(3) The fast transistor outputs can be used either as regular transistor outputs, as reflex outputs for counting function (HSC), or as fast transistor outputs for pulse generator functions (FreqGen / PTO / PWM).

Reference	Digital Inputs	Digital Outputs	Communication Ports	Terminal Type	Power supply
TM241CE40U *(see page 191)*	16 regular inputs(1) 8 fast inputs (counters)(2)	Sink outputs 12 regular transistor outputs 4 fast outputs (pulse generators)(3)	2 serial line ports 1 USB programming port 1 Ethernet port	Removable screw terminal blocks	24 Vdc

(1) The regular inputs have a maximum frequency of 1 kHz.
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.
(3) The fast transistor outputs can be used either as regular transistor outputs, as reflex outputs for counting function (HSC), or as fast transistor outputs for pulse generator functions (FreqGen / PTO / PWM).

Delivery Content

The following figure presents the content of the delivery for a M241 Logic Controller:



1 M241 Logic Controller Instruction Sheet
2 M241 Logic Controller
3 Lithium carbon monofluoride battery, type Panasonic BR2032.

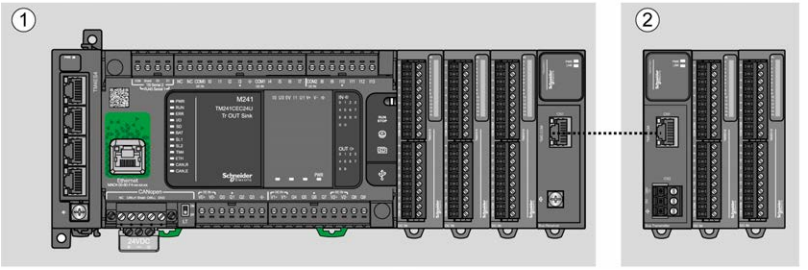
Maximum Hardware Configuration

Introduction

The M241 Logic Controller is a control system that offers an all-in-one solution with optimized configurations and an expandable architecture.

Local and Remote Configuration Principle

The following figure defines the local and remote configurations:



(1) Local configuration
(2) Remote configuration

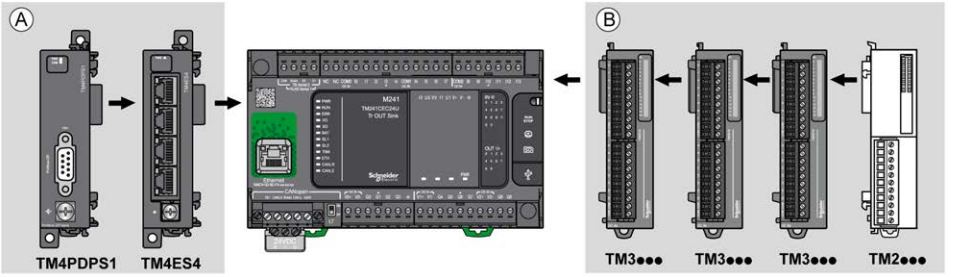
M241 Logic Controller Local Configuration Architecture

Optimized local configuration and flexibility are provided by the association of:

- M241 Logic Controller
- TM4 expansion modules
- TM3 expansion modules
- TM2 expansion modules

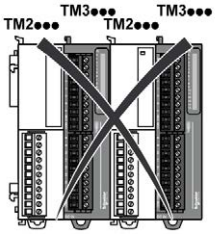
Application requirements determine the architecture of your M241 Logic Controller configuration.

The following figure represents the components of a local configuration:



(A) Expansion modules (3 maximum)
(B) Expansion modules (7 maximum)

NOTE: It is prohibited to mount a TM2 module before any TM3 module as indicated in the following figure:



M241 Logic Controller Remote Configuration Architecture

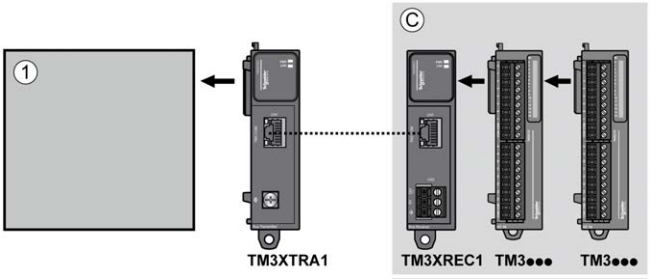
Optimized remote configuration and flexibility are provided by the association of:

- M241 Logic Controller
- TM4 expansion modules
- TM3 expansion modules
- TM3 transmitter and receiver modules

Application requirements determine the architecture of your M241 Logic Controller configuration.

NOTE: You cannot use TM2 modules in configurations that include the TM3 transmitter and receiver modules.

The following figure represents the components of a remote configuration:



(1) Logic controller and modules
(C) TM3 expansion modules (7 maximum)

Maximum Number of Modules

The following table shows the maximum configuration supported:

References	Maximum	Type of Configuration
TM241••••	7 TM3 / TM2 expansion modules	Local
TM241••••	3 TM4 expansion modules	Local
TM3XREC1	7 TM3 expansion modules	Remote
NOTE: TM3 transmitter and receiver modules are not included in a count of the maximum number of expansion modules.		

NOTE: The configuration with its TM4, TM3, and TM2 expansion modules is validated by EcoStruxure Machine Expert software in the **Configuration** window.

NOTE: In some environments, the maximum configuration populated by high consummation modules, coupled with the maximum distance allowable between the TM3 transmitter and receiver modules, may present bus communication issues although the EcoStruxure Machine Expert software allowed for the configuration. In such a case you will need to analyze the consummation of the modules chosen for your configuration, as well as the minimum cable distance required by your application, and possibly seek to optimize your choices.

TMC4 Cartridges

Overview

You can expand the number of I/Os of your Modicon M241 Logic Controller by adding TMC4 cartridges.

For more information, refer to the TMC4 Cartridges Hardware Guide.

TMC4 Standard Cartridges

The following table presents the general-purpose TMC4 cartridges with the corresponding channel type, voltage/current range, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type
TMC4AI2	2	Analog inputs (voltage or current)	0...10 Vdc 0...20 mA or 4...20 mA	3.81 mm (0.15 in.) pitch, removable spring terminal block
TMC4TI2	2	Analog temperature inputs	Thermocouple type K, J, R, S, B, E, T, N,C 3 wires RTD type Pt100, Pt1000, Ni100, Ni1000	3.81 mm (0.15 in.) pitch, removable spring terminal block
TMC4AQ2	2	Analog outputs (voltage or current)	0...10 Vdc 4...20 mA	3.81 mm (0.15 in.) pitch, removable spring terminal block

TMC4 Application Cartridges

The following table presents the applicative TMC4 cartridges with the corresponding channel type, voltage/current range, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type
TMC4HOIS01	2	Analog inputs (voltage or current)	0...10 Vdc 0...20 mA or 4...20 mA	3.81 mm (0.15 in.) pitch, removable spring terminal block
TMC4PACK01	2	Analog inputs (voltage or current)	0...10 Vdc 0...20 mA or 4...20 mA	3.81 mm (0.15 in.) pitch, removable spring terminal block

TM2 Expansion Modules

Overview

You can expand the number of I/Os of your M241 Logic Controller by adding TM2 I/O expansion modules.

The following types of electronic modules are supported:
- TM2 digital I/O expansion modules
- TM2 analog I/O expansion modules

For more information, refer to the following documents:
- TM2 Digital I/O Expansion Modules Hardware Guide
- TM2 Analog I/O Expansion Modules Hardware Guide

NOTE: TM2 modules can only be used in the local configuration, and only if there is no TM3 transmitter and receiver modules present in the configuration.

NOTE: It is prohibited to mount a TM2 module before any TM3 module. The TM2 modules must be mounted and configured at the end of the local configuration.

TM2 Digital Input Expansion Modules

The following table shows the compatible TM2 digital input expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type
TM2DAI8DT	8	Regular inputs	120 Vac 7.5 mA	Removable screw terminal block
TM2DDI8DT	8	Regular inputs	24 Vdc 7 mA	Removable screw terminal block
TM2DDI16DT	16	Regular inputs	24 Vdc 7 mA	Removable screw terminal block
TM2DDI16DK	16	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector
TM2DDI32DK	32	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector

TM2 Digital Output Expansion Modules

The following table shows the compatible TM2 digital output expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal type
TM2DRA8RT	8	Relay outputs	30 Vdc / 240 Vac 2 A max	Removable screw terminal block
TM2DRA16RT	16	Relay outputs	30 Vdc / 240 Vac 2 A max	Removable screw terminal block
TM2DDO8UT	8	Regular transistor outputs (sink)	24 Vdc 0.3 A max per output	Removable screw terminal block
TM2DDO8TT	8	Regular transistor outputs (source)	24 Vdc 0.5 A max per output	Removable screw terminal block
TM2DDO16UK	16	Regular transistor outputs (sink)	24 Vdc 0.1 A max per output	HE10 (MIL 20) connector
TM2DDO16TK	16	Regular transistor outputs (source)	24 Vdc 0.4 A max per output	HE10 (MIL 20) connector
TM2DDO32UK	32	Regular transistor outputs (sink)	24 Vdc 0.1 A max per output	HE10 (MIL 20) connector
TM2DDO32TK	32	Regular transistor outputs (source)	24 Vdc 0.4 A max per output	HE10 (MIL 20) connector

TM2 Digital Mixed Input/Output Expansion Modules

The following table shows the compatible TM2 digital mixed I/O expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal type
TM2DMM8DRT	4	Regular inputs	24 Vdc 7 mA	Removable screw terminal block
	4	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM2DMM24DRF	16	Regular inputs	24 Vdc 7 mA	Non-removable spring terminal block
	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	

TM2 Analog Input Expansion Modules

The following table shows the compatible TM2 analog input expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal Type
TM2AMI2HT	2	High-level inputs	0...10 Vdc 4...20 mA	Removable screw terminal block
TM2AMI2LT	2	Low-level inputs	Thermocouple type J,K,T	Removable screw terminal block
TM2AMI4LT	4	Analog inputs	0...10 Vdc 0...20 mA PT100/1000 Ni100/1000	Removable screw terminal block
TM2AMI8HT	8	Analog inputs	0...20 mA 0...10 Vdc	Removable screw terminal block
TM2ARI8HT	8	Analog inputs	NTC / PTC	Removable screw terminal block
TM2ARI8LRJ	8	Analog inputs	PT100/1000	RJ11 connector
TM2ARI8LT	8	Analog inputs	PT100/1000	Removable screw terminal block

TM2 Analog Output Expansion Modules

The following table shows the compatible TM2 analog output expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal Type
TM2AMO1HT	1	Analog outputs	0...10 Vdc 4...20 mA	Removable screw terminal block
TM2AVO2HT	2	Analog outputs	+/- 10 Vdc	Removable screw terminal block

TM2 Analog Mixed Input/Output Expansion Modules

The following table shows the compatible TM2 analog mixed I/O expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal Type
TM2AMM3HT	2	Analog inputs	0...10 Vdc 4...20 mA	Removable screw terminal block
	1	Analog outputs	0...10 Vdc 4...20 mA	
TM2AMM6HT	4	Analog inputs	0...10 Vdc 4...20 mA	Removable screw terminal block
	2	Analog outputs	0...10 Vdc 4...20 mA	
TM2ALM3LT	2	Low-level inputs	Thermo J,K,T, PT100	Removable screw terminal block
	1	Analog outputs	0...10 Vdc 4...20 mA	

TM3 Expansion Modules

Introduction

The range of TM3 expansion modules includes:

- Digital modules, classified as follows:
  - Input modules *(see page 31)*
  - Output modules *(see page 32)*
  - Mixed input/output modules *(see page 34)*
- Analog modules, classified as follows:
  - Input modules *(see page 35)*
  - Output modules *(see page 36)*
  - Mixed input/output modules *(see page 37)*
- Expert modules *(see page 38)*
- Safety modules *(see page 38)*
- Transmitter and Receiver modules *(see page 39)*

For more information, refer to the following documents:

- TM3 Digital I/O Modules Hardware Guide
- TM3 Analog I/O Modules Hardware Guide
- TM3 Expert I/O Modules Hardware Guide
- TM3 Safety Modules Hardware Guide
- TM3 Transmitter and Receiver Modules Hardware Guide

TM3 Digital Input Modules

The following table shows the TM3 digital input expansion modules, with corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DI8A	8	Regular inputs	120 Vac 7.5 mA	Removable screw terminal block / 5.08 mm
TM3DI8	8	Regular inputs	24 Vdc 7 mA	Removable screw terminal block / 5.08 mm
TM3DI8G	8	Regular inputs	24 Vdc 7 mA	Removable spring terminal block / 5.08 mm
TM3DI16	16	Regular inputs	24 Vdc 7 mA	Removable screw terminal blocks / 3.81 mm

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DI16G	16	Regular inputs	24 Vdc 7 mA	Removable spring terminal blocks / 3.81 mm
TM3DI16K	16	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector
TM3DI32K	32	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector

TM3 Digital Output Modules

The following table shows the TM3 digital output expansion modules, with corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DQ8R	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	Removable screw terminal block / 5.08 mm
TM3DQ8RG	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	Removable spring terminal block / 5.08 mm
TM3DQ8T	8	Regular transistor outputs (source)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable screw terminal block / 5.08 mm
TM3DQ8TG	8	Regular transistor outputs (source)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable spring terminal block / 5.08 mm
TM3DQ8U	8	Regular transistor outputs (sink)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable screw terminal block / 5.08 mm
TM3DQ8UG	8	Regular transistor outputs (sink)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable spring terminal block / 5.08 mm
TM3DQ16R	16	Relay outputs	24 Vdc / 240 Vac 8 A maximum per common line / 2 A maximum per output	Removable screw terminal blocks / 3.81 mm

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DQ16RG	16	Relay outputs	24 Vdc / 240 Vac 8 A maximum per common line / 2 A maximum per output	Removable spring terminal blocks / 3.81 mm
TM3DQ16T	16	Regular transistor outputs (source)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable screw terminal blocks / 3.81 mm
TM3DQ16TG	16	Regular transistor outputs (source)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable spring terminal blocks / 3.81 mm
TM3DQ16U	16	Regular transistor outputs (sink)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable screw terminal blocks / 3.81 mm
TM3DQ16UG	16	Regular transistor outputs (sink)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable spring terminal blocks / 3.81 mm
TM3DQ16TK	16	Regular transistor outputs (source)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connector
TM3DQ16UK	16	Regular transistor outputs (sink)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connector
TM3DQ32TK	32	Regular transistor outputs (source)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connectors
TM3DQ32UK	32	Regular transistor outputs (sink)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connectors

TM3 Digital Mixed Input/Output Modules

This following table shows the TM3 mixed I/O modules, with corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DM8R	4	Regular inputs	24 Vdc 7 mA	Removable screw terminal block / 5.08 mm
	4	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM3DM8RG	4	Regular inputs	24 Vdc 7 mA	Removable spring terminal block /5.08 mm
	4	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM3DM24R	16	Regular inputs	24 Vdc 7 mA	Removable screw terminal blocks / 3.81 mm
	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM3DM24RG	16	Regular inputs	24 Vdc 7 mA	Removable spring terminal blocks / 3.81 mm
	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	

TM3 Analog Input Modules

The following table shows the TM3 analog input expansion modules, with corresponding resolution, channel type, nominal voltage/current, and terminal type:

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3AI2H	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 5.08 mm
TM3AI2HG	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 5.08 mm
TM3AI4	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 3.81 mm
TM3AI4G	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal blocks / 3.81 mm
TM3AI8	12 bit, or 11 bit + sign	8	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA 0...20 mA extended 4...20 mA extended	Removable screw terminal block / 3.81 mm
TM3AI8G	12 bit, or 11 bit + sign	8	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA 0...20 mA extended 4...20 mA extended	Removable spring terminal blocks / 3.81 mm
TM3TI4	16 bit, or 15 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable screw terminal block / 3.81 mm

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3TI4G	16 bit, or 15 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable spring terminal blocks / 3.81 mm
TM3TI4D	16 bit, or 15 bit + sign	4	inputs	Thermocouple	Removable screw terminal block / 3.81 mm
TM3TI4DG	16 bit, or 15 bit + sign	4	inputs	Thermocouple	Removable spring terminal blocks / 3.81 mm
TM3TI8T	16 bit, or 15 bit + sign	8	inputs	Thermocouple NTC/PTC Ohmmeter	Removable screw terminal block / 3.81 mm
TM3TI8TG	16 bit, or 15 bit + sign	8	inputs	Thermocouple NTC/PTC Ohmmeter	Removable spring terminal blocks / 3.81 mm

TM3 Analog Output Modules

The following table shows the TM3 analog output modules, with corresponding resolution, channel type, nominal voltage/current, and terminal type:

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3AQ2	12 bit, or 11 bit + sign	2	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 5.08 mm
TM3AQ2G	12 bit, or 11 bit + sign	2	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 5.08 mm
TM3AQ4	12 bit, or 11 bit + sign	4	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 5.08 mm
TM3AQ4G	12 bit, or 11 bit + sign	4	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 5.08 mm

TM3 Analog Mixed Input/Output Modules

This following table shows the TM3 analog mixed I/O modules, with corresponding resolution, channel type, nominal voltage/current, and terminal type:

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3AM6	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 3.81 mm
		2	outputs		
TM3AM6G	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 3.81 mm
		2	outputs		
TM3TM3	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable screw terminal block / 5.08 mm
	12 bit, or 11 bit + sign	1	outputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	
TM3TM3G	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable spring terminal block / 5.08 mm
	12 bit, or 11 bit + sign	1	outputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	

TM3 Expert Modules

The following table shows the TM3 expert expansion modules, with corresponding terminal types:

Reference	Description	Terminal Type / Pitch
TM3XTYS4	TeSys module	4 front connectors RJ-45 1 removable power supply connector / 5.08 mm
TM3XHSC202	High Speed Counting (HSC) module	Removable screw terminal blocks / 3.81 mm
TM3XHSC202G	High Speed Counting (HSC) module	Removable spring terminal blocks / 3.81 mm

TM3 Safety Modules

This table contains the TM3 safety modules, with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Function Category	Channels	Channel type	Voltage Current	Terminal type
TM3SAC5R	1 function, up to category 3	1 or 2 (1)	Safety input	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start (2)	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAC5RG	1 function, up to category 3	1 or 2 (1)	Safety input	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start (2)	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAF5R	1 function, up to category 4	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAF5RG	1 function, up to category 4	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	

(1) Depending on external wiring
(2) Non-monitored start

Reference	Function Category	Channels	Channel type	Voltage Current	Terminal type
TM3SAFL5R	2 functions, up to category 3	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAFL5RG	2 functions, up to category 3	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAK6R	3 functions, up to category 4	1 or 2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAK6RG	3 functions, up to category 4	1 or 2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	

(1) Depending on external wiring
(2) Non-monitored start

TM3 Transmitter and Receiver Modules

The following table shows the TM3 transmitter and receiver expansion modules:

Reference	Description	Terminal Type / Pitch
TM3XTRA1	Data transmitter module for remote I/O	1 front connector RJ-45 1 screw for functional ground connection
TM3XREC1	Data receiver module for remote I/O	1 front connector RJ-45 Power supply connector / 5.08 mm

TM3 Bus Couplers

Introduction

The TM3 bus coupler is a device designed to manage fieldbus communication when using TM2 and TM3 expansion modules in a distributed architecture.

For more information, refer to the Modicon TM3 Bus Coupler Hardware Guide.

Modicon TM3 Bus Couplers

The following table shows the TM3 bus couplers, with ports and terminal types:

Reference	Port	Communication type	Terminal type
TM3BCEIP	2 isolated switched Ethernet ports	EtherNet/IP Modbus TCP	RJ45
	1 USB mini-B port	USB 2.0	USB mini-B
TM3BCSL	2 isolated ports	Serial Line Modbus	RJ45
	1 USB mini-B port	USB 2.0	USB mini-B

TM4 Expansion Modules

Introduction

The range of TM4 expansion modules includes communication modules.

For more information, refer to the TM4 Expansion Modules Hardware Guide.

TM4 Expansion Modules

The following table shows the TM4 expansion module features:

Module reference	Type	Terminal type
TM4ES4	Ethernet communication	4 RJ45 connectors 1 screw for functional ground connection
TM4PDPS1	PROFIBUS DP slave communication	1 SUB-D 9 pins female connector 1 screw for functional ground connection
NOTE: The TM4ES4 module has two applications: expansion or standalone. For more information, refer to TM4 Compatibility.		

TM5 Fieldbus Interfaces

Introduction

The TM5 fieldbus interfaces are devices designed to manage EtherNet/IP communication when using TM5 System and TM7 expansion modules with a controller in a distributed architecture.

For more information, refer to the Modicon TM5 System Interface – Hardware Guide.

TM5 Fieldbus Interfaces

The following table shows the TM5 fieldbus interfaces with ports and terminal type:

Reference	Port	Communication type	Terminal type
TM5NEIP1	2 Ethernet switched ports	EtherNet/IP	RJ45

TM5 CANopen Fieldbus Interfaces

Introduction

The TM5 fieldbus module is a CANopen interface with built-in power distribution and is the first TM5 distributed I/O island.

For more information, refer to the Modicon TM5 CANopen Interface Hardware Guide.

Modicon TM5 CANopen Fieldbus Interfaces

The following table shows the TM5 CANopen fieldbus interfaces:

Reference	Communication type	Terminal type
TM5NCO1	CANopen	1 SUB-D 9, male

TM7 CANopen Fieldbus Interfaces

Introduction

The TM7 fieldbus modules are CANopen interfaces with 24 Vdc digital configurable input or output on 8 or 16 channels.

For more information, refer to the Modicon TM7 CANopen Interface I/O Blocks Hardware Guide.

Modicon TM7 CANopen Fieldbus Interfaces

The following table shows the TM7 CANopen fieldbus interfaces:

Reference	Number of channels	Voltage/Current	Communication type	Terminal type
TM7NCOM08B	8 inputs 8 outputs	24 Vdc / 4 mA 24 Vdc / 500 mA	CANopen	M8 Connector
TM7NCOM16A	16 inputs 16 outputs	24 Vdc / 4 mA 24 Vdc / 500 mA	CANopen	M8 Connector
TM7NCOM16B	16 inputs 16 outputs	24 Vdc / 4 mA 24 Vdc / 500 mA	CANopen	M12 Connector

Accessories

Overview

This section describes the accessories and cables.

Accessories

Reference	Description	Use	Quantity
TMASD1	SD Card *(see page 63)*	Use to update the controller firmware, initialize a controller with a new application or clone a controller, manage user files, etc.,.	1
TMAT2CSET	Set of 5 removable screw terminal block	Connects M241 Logic Controller embedded I/Os.	1
TMAT2PSET	Set of 5 removable screw terminal block	Connects 24 Vdc power supply.	1
NSYTRAAB35	End brackets	Helps secure the controller or receiver module and their expansion modules on a top hat section rail (DIN rail).	1
TM2XMTGB	Grounding Bar	Connects the cable shield and the module to the functional ground.	1
TM200RSRCEMC	Shielding take-up clip	Mounts and connects the ground to the cable shielding.	25 pack

Cables

Reference	Description	Details	Length
TCSXCNAMUM3P	Terminal port/USB port cordset	From the USB mini-B port on the M241 Logic Controller to USB port on the PC terminal.	3 m (10 ft)
BMXXCAUSBH018	Terminal port/USB port cordset	From the USB mini-B port on the M241 Logic Controller to USB port on the PC terminal. **NOTE:** Grounded and shielded, this USB cable is suitable for long-duration connections.	1.8 m (5.9 ft)
490NTW000••	Ethernet shielded cable for DTE connections	Standard cable, equipped with RJ45 connectors at each end for DTE. CE compliant.	2, 5, 12, 40, or 80 m (6.56, 16.4, 39.37, 131.23 or 262.47 ft)
490NTW000••U		Standard cable, equipped with RJ45 connectors at each end for DTE. UL compliant.	2, 5, 12, 40, or 80 m (6.56, 16.4, 39.37, 131.23, or 262.47 ft)
TCSECE3M3M••S4		Cable for harsh environment, equipped with RJ45 connectors at each end. CE compliant.	1, 2, 3, 5, or 10 m (3.28, 6.56, 9.84, 16.4, 32.81 ft)
TCSECU3M3M••S4		Cable for harsh environment, equipped with RJ45 connectors at each end. UL compliant.	1, 2, 3, 5, or 10 m (3.28, 6.56, 9.84, 16.4, 32.81 ft)

Chapter 2
M241 Features

Overview

This chapter describes the Modicon M241 Logic Controller features.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Real Time Clock (RTC)	48
Input Management	52
Output Management	56
Run/Stop	61
SD Card	63

Real Time Clock (RTC)

Overview

The M241 Logic Controller includes an RTC to provide system date and time information, and to support related functions requiring a real-time clock. To continue keeping time when power is off, a non-rechargeable battery is required (see reference below). A battery LED on the front panel of the controller indicates if the battery is depleted or absent.

This table shows how RTC drift is managed:

RTC Characteristics	Description
RTC drift	Less than 60 seconds per month without any user calibration at 25 °C (77 °F)

Battery

The controller has one battery.

In the event of a power interruption, the backup battery maintains the RTC for the controller.

This table shows the characteristics of the battery:

Characteristics	Description
Use	In the event of a transient power outage, the battery powers the RTC.
Backup life	At least 2 years at 25 °C maximum (77 °F). At higher temperatures, the time is reduced.
Battery monitoring	Yes
Replaceable	Yes
Controller battery type	Lithium carbon monofluoride, type Panasonic BR2032

Installing and Replacing the Battery

While lithium batteries are preferred due to their slow discharge and long life, they can present hazards to personnel, equipment and the environment and must be handled properly.

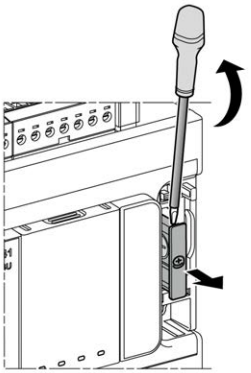
⚠ DANGER

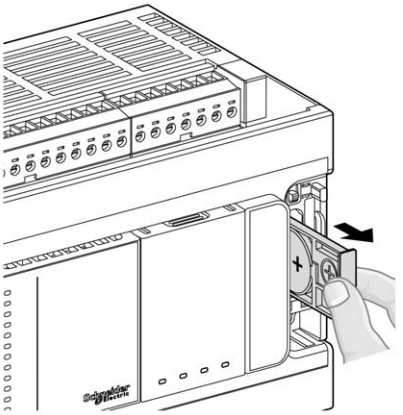
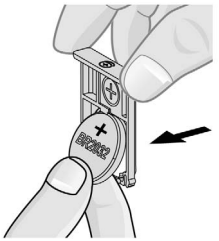
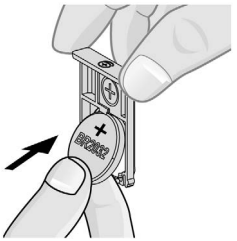
EXPLOSION, FIRE, OR CHEMICAL BURNS

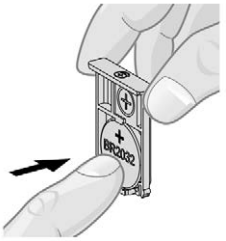
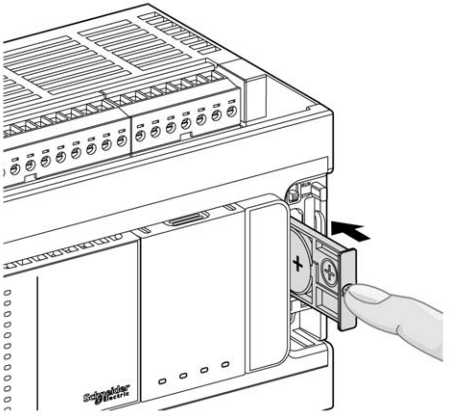
- Replace with identical battery type.
- Follow all the instructions of the battery manufacturer.
- Remove all replaceable batteries before discarding unit.
- Recycle or properly dispose of used batteries.
- Protect battery from any potential short-circuit.
- Do not recharge, disassemble, heat above 100 °C (212 °F), or incinerate.
- Use your hands or insulated tools to remove or replace the battery.
- Maintain proper polarity when inserting and connecting a new battery.

Failure to follow these instructions will result in death or serious injury.

To install or replace the battery, follow these steps:

Step	Action
1	Remove power from your controller.
2	Use an insulated screw-driver to pull out the battery holder. 

Step	Action
3	Slide out the battery holder of the controller. 
4	Remove the battery from the battery holder. 
5	Insert the new battery into the battery holder in accordance with the polarity markings on the battery. 

Step	Action
6	Replace the battery holder on the controller and verify that the latch clicks into place. 
7	Slide in the battery holder of the controller. 
8	Power up your M241 Logic Controller.
9	Set the internal clock. For further details on the internal clock, refer to Programming Guide *(see Modicon M241 Logic Controller, Programming Guide)*.

NOTE: Replacement of the battery in the controllers other than with the type specified in this documentation may present a risk of fire or explosion.

⚠ WARNING
IMPROPER BATTERY CAN PROVOKE FIRE OR EXPLOSION
Replace battery only with identical type: Panasonic Type BR2032.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Input Management

Overview

The M241 Logic Controller features digital inputs, including 8 fast inputs.

The following functions are configurable:

- Filters (depends on the function associated with the input).
- All inputs can be used for the Run/Stop function.
- 8 fast inputs can be either latched or used for events (rising edge, falling edge, or both) and thus be linked to an external task.

NOTE: All inputs can be used as regular inputs.

Input Management Functions Availability

Embedded digital inputs can be configured as functions (Run/Stop, events, HSC).

Inputs not configured as functions are used as regular inputs.

The following table shows the possible usage of the M241 Logic Controller digital inputs:

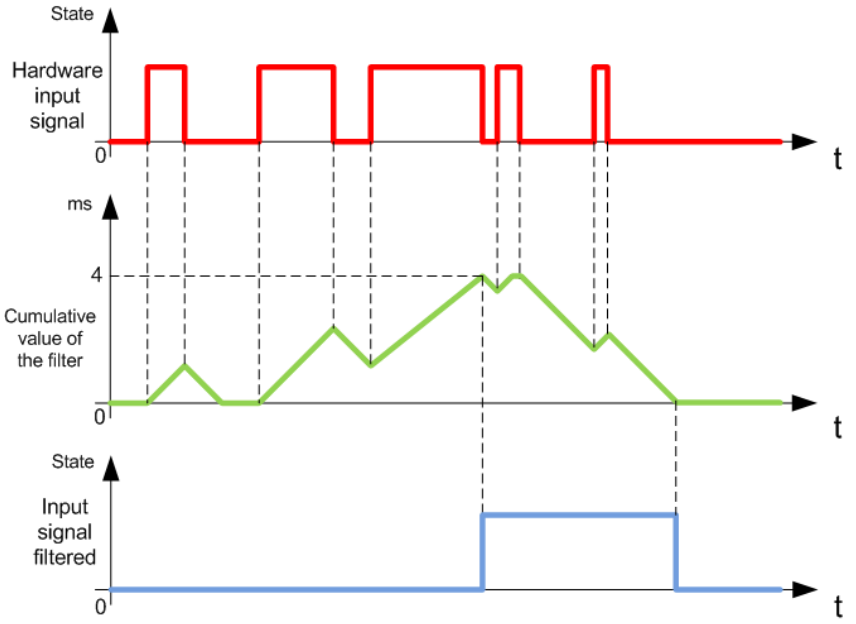
Function	Input Function				HSC
	None	RUN/STOP	Latch	Event	
Filter type	Integrator	Integrator	Bounce	Bounce	
Fast inputs[1]	I0...I7				
Regular inputs	I8...I13[2] I8...I23[3]	I8...I13[2] I8...I23[3]	–	–	I8...I13[2] [4] I8...I15[3] [4]

– No
1 Can also be used as regular inputs
2 For M241 with 24 I/O channels
3 For M241 with 40 I/O channels
4 Limited to 1 kHz

Integrator Filter Principle

The integrator filter is designed to reduce the effect of noise. Setting a filter value allows the logic controller to ignore some sudden changes of input levels caused by noise.

The following timing diagram illustrates the integrator filter effects for a value of 4 ms:

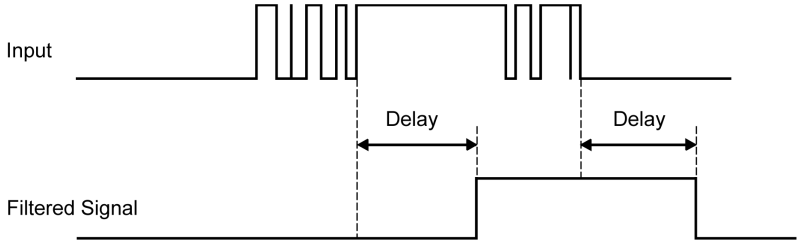


NOTE: The value selected for the filter's time parameter specifies the cumulative time in ms that must elapse before the input can be 1.

Bounce Filter Principle

The bounce filter is designed to reduce the bouncing effect at the inputs. Setting a bounce filter value allows the controller to ignore some sudden changes of input levels caused by electrical noise. The bounce filter is only available on the fast inputs.

The following timing diagram illustrates the anti-bounce filter effects:



Bounce Filter Availability

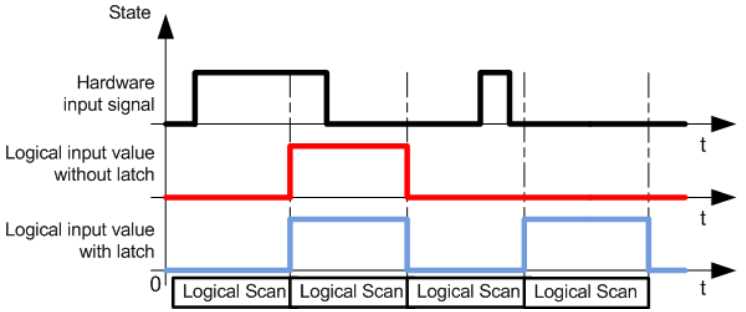
The bounce filter can be used on a fast input when:

- Using a latch or event
- HSC is enabled

Latching

Latching is a function that can be assigned to the M241 Logic Controller fast inputs. This function is used to memorize (or latch) any pulse with a duration that is less than the M241 Logic Controller scan time. When a pulse is shorter than one scan, the controller latches the pulse, which is then updated in the next scan. This latching mechanism only recognizes rising edges. Falling edges cannot be latched. Assigning inputs to be latched is done in the **I/O Configuration** tab in EcoStruxure Machine Expert.

The following timing diagram illustrates the latching effects:



Event

An input configured for Event can be associated with an External Task *(see Modicon M241 Logic Controller, Programming Guide)*.

Run/Stop

The Run/Stop function is used to start or stop an application program using an input. In addition to the embedded Run/Stop switch, it is allowed to configure one (and only one) input as an additional Run/Stop command.

For more information, refer to Run/Stop *(see page 61)*.

⚠ WARNING

UNINTENDED MACHINE OR PROCESS START-UP

- Verify the state of security of your machine or process environment before applying power to the Run/Stop input.
- Use the Run/Stop input to help prevent the unintentional start-up from a remote location.

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Use the sensor and actuator power supply only for supplying power to sensors or actuators connected to the module.

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

Output Management

Introduction

The M241 Logic Controller features both regular and fast transistor outputs (PTO / PWM / FreqGen).

The following output functions are configurable on the transistor outputs:

- Alarm output
- HSC (reflex features on HSC threshold)
- PTO
- PWM
- FreqGen

NOTE: All outputs can be used as regular outputs.

Output Management Availability

The following table shows the possible usage of the M241 Logic Controller digital outputs on references with transistor outputs:

Reference		Function		Alarm Output	HSC	FreqGen	PWM	PTO
TM241C•40T / TM241C•40U	TM241C••24T / TM241C••24U	Fast output	Q0	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q1	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q2	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q3	X	Reflex output 0 or 1	Output A	Output A	Output A or B
		Regular output	Q4	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q5	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q6	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q7	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q8	X	–	–	–	–
			Q9	X	–	–	–	–
			Q10	X	–	–	–	–
			Q11	X	–	–	–	–
			Q12	X	–	–	–	–
			Q13	X	–	–	–	–
			Q14	X	–	–	–	–
			Q15	X	–	–	–	–

The following table shows the possible usage of the M241 Logic Controller on references with relay outputs:

Reference		Function		Alarm Output	HSC	FreqGen	PWM	PTO
TM241C•40R	TM241C••24R	Fast output	Q0	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q1	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q2	X	Reflex output 0 or 1	Output A	Output A	Output A or B
			Q3	X	Reflex output 0 or 1	Output A	Output A	Output A or B
		Regular output	Q4	X	Reflex output 0 or 1	–	–	–
			Q5	X	Reflex output 0 or 1	–	–	–
			Q6	X	Reflex output 0 or 1	–	–	–
			Q7	X	Reflex output 0 or 1	–	–	–
			Q8	X	–	–	–	–
			Q9	X	–	–	–	–
			Q10	X	–	–	–	–
			Q11	X	–	–	–	–
			Q12	X	–	–	–	–
			Q13	X	–	–	–	–
			Q14	X	–	–	–	–
			Q15	X	–	–	–	–

Fallback Modes (Behavior for Outputs in Stop)

When the controller enters the STOPPED or one of the exception states for any reason, the local (embedded and expansion) outputs are set to **Default Value** defined in the application.

In case of PTO outputs, the fallback values are forced to 0 logic (0 Vdc) and these values cannot be modified.

Short-circuit or Over-current on Source Transistor Outputs

Outputs are clustered in packs of 4 outputs maximum (less when the total number of outputs of the controller is not a multiple of 4):

- **Q0**...**Q3**
- **Q4**...**Q7**
- **Q8**...**Q11**
- **Q12**...**Q15**

When a short-circuit or overload is detected, the cluster of 4 outputs is set to 0. An automatic rearming is done periodically (about 1 s).

The following table describes the actions taken on short-circuits or overload of transistor outputs Q0 to Q3:

If...	then...
If you have short-circuit at 0 V on transistor outputs	Transistor outputs automatically go into over-current protection or thermal protection mode. For more information, refer to transistor output wiring diagrams.
If you have short-circuit at 24 V on transistor outputs	Transistor outputs automatically go into over-current protection mode. For more information, refer to transistor output wiring diagrams.

The following table describes the actions taken on short-circuits or overload of transistor outputs from Q4 to Q15:

If...	then...
If you have short-circuit at 0 V on transistor outputs	Transistor outputs automatically go into thermal protection mode. For more information, refer to transistor output wiring diagrams.
If you have short-circuit at 24 V on transistor outputs	No action is taken and no error is detectable. A short-circuit or overvoltage over 24 V may result in equipment damage.

In the case of a short-circuit or current overload, the common group of outputs automatically enters into thermal protection mode (all outputs in the group are set to 0), and are then periodically rearmed (each second) to test the connection state. However, you must be aware of the effect of this rearming on the machine or process being controlled.

⚠ WARNING

UNINTENDED MACHINE START-UP

Inhibit the automatic rearming of outputs if this feature is an undesirable behavior for your machine or process.

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

NOTE: The automatic rearming feature can be inhibited. Refer to the Programming Guide of your controller for more information.

Short-circuit or Over-Current on Sink Transistor Outputs

Sink transistor outputs are not internally protected against overloads or short-circuits.

The following table describes the actions taken on overloads or short-circuits on sink transistor outputs from Q0 to Q3:

If...	then...
If you have short-circuit at 0 V on transistor outputs	Transistor outputs automatically go into over-current protection or thermal protection mode. For more information, refer to transistor output wiring diagrams.
If you have short-circuit at 24 V on transistor outputs	Transistor outputs automatically go into over-current protection mode. For more information, refer to transistor output wiring diagrams.

The following table describes the actions taken on overloads or short-circuits on sink transistor outputs from Q4 to Q15:

If...	then...
If you have short-circuit at 0 V on transistor outputs	no action is taken and no error is detectable. A short-circuit or undervoltage less than 0 V may result in equipment damage.
If you have short-circuit at 24 V on transistor outputs	Transistor outputs automatically go into thermal protection mode. For more information, refer to transistor output wiring diagrams.

Short-circuit or Over-Current on Relay Outputs

Relay outputs are not internally protected against overloads or short-circuits.

The following table describes the actions taken on overloads or short-circuits on relay outputs:

If...	then...
If you have short-circuit or overload at 0 V or 24 V on relay outputs	No action is taken and no error is detectable. For more information, refer to relay output wiring diagrams.

Relay outputs are electromechanical switches capable of carrying significant levels of current and voltage. All electromechanical devices have a limited operational life and must be installed so as to minimize the potential for unintended consequences.

⚠ WARNING

INOPERABLE OUTPUTS

Use appropriate, external safety interlocks on outputs where personnel and/or equipment hazards exist.

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

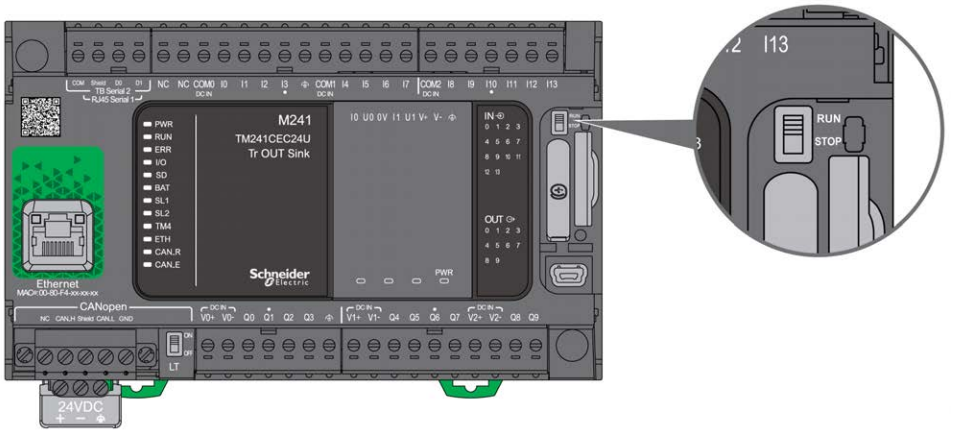
Run/Stop

Run/Stop

The M241 Logic Controller can be operated externally by the following:

- a hardware Run/Stop switch
- a Run/Stop operation by a dedicated digital input, defined in the software configuration (For more information, refer to Embedded I/Os Configuration *(see Modicon M241 Logic Controller, Programming Guide)*.)
- an EcoStruxure Machine Expert software command.

The M241 Logic Controller has a Run/Stop hardware switch, which puts the controller in a RUN or STOP state.



The interaction of the 2 external operators on the controller state behavior is summarized in the table below:

		Embedded hardware Run/Stop switch		
		Switch on Stop	**Stop to Run transition**	**Switch on Run**
Software configurable Run/Stop digital input	**None**	STOP Ignores external Run/Stop commands.	Commands a transition to RUN state[1].	Allows external Run/Stop commands.
	State 0	STOP Ignores external Run/Stop commands.	STOP Ignores external Run/Stop commands.	STOP Ignores external Run/Stop commands.
	Rising edge	STOP Ignores external Run/Stop commands.	Commands a transition to RUN state[1].	Commands a transition to RUN state.
	State 1	STOP Ignores external Run/Stop commands.	Commands a transition to RUN state[1].	Allows external Run/Stop commands.

[1] For more information, refer to the Controller States and Behaviors *(see Modicon M241 Logic Controller, Programming Guide)*.

⚠ WARNING

UNINTENDED MACHINE OR PROCESS START-UP

- Verify the state of security of your machine or process environment before applying power to the Run/Stop input or engaging the Run/Stop switch.
- Use the Run/Stop input to help prevent the unintentional start-up from a remote location, or from accidentally engaging the Run/Stop switch.

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

SD Card

Overview

When handling the SD card, follow the instructions below to help prevent internal data on the SD card from being corrupted or lost or an SD card malfunction from occurring:

NOTICE

LOSS OF APPLICATION DATA

- Do not store the SD card where there is static electricity or probable electromagnetic fields.
- Do not store the SD card in direct sunlight, near a heater, or other locations where high temperatures can occur.
- Do not bend the SD card.
- Do not drop or strike the SD card against another object.
- Keep the SD card dry.
- Do not touch the SD card connectors.
- Do not disassemble or modify the SD card.
- Use only SD cards formatted using FAT or FAT32.

Failure to follow these instructions can result in equipment damage.

The M241 Logic Controller does not recognize NTFS formatted SD cards. Format the SD card on your computer using FAT or FAT32.

When using the M241 Logic Controller and an SD card, observe the following to avoid losing valuable data:

- Accidental data loss can occur at any time. Once data is lost it cannot be recovered.
- If you forcibly extract the SD card, data on the SD card may become corrupted.
- Removing an SD card that is being accessed could damage the SD card, or corrupt its data.
- If the SD card is not positioned correctly when inserted into the controller, the data on the card and the controller could become damaged.

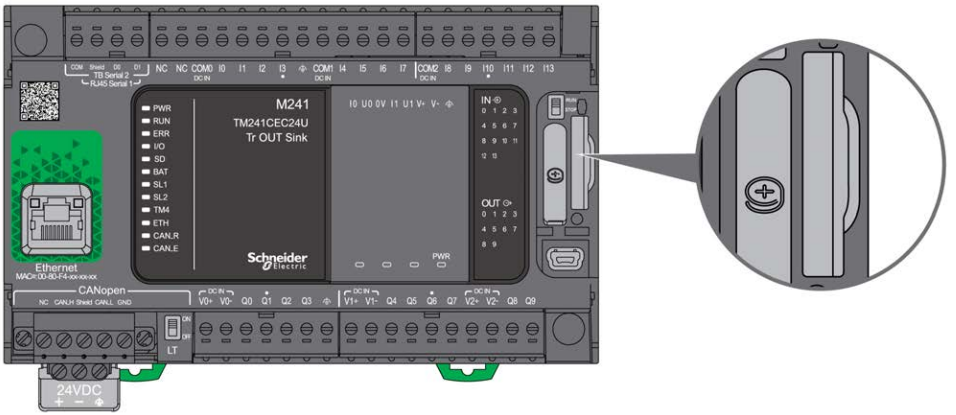
NOTICE

LOSS OF APPLICATION DATA

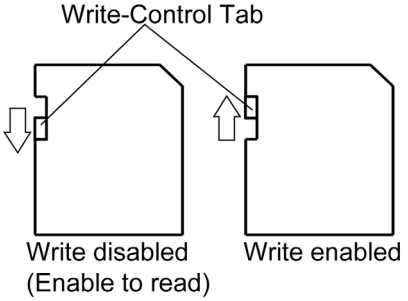
- Backup SD card data regularly.
- Do not remove power or reset the controller, and do not insert or remove the SD card while it is being accessed.

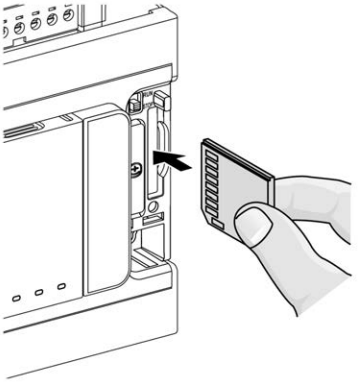
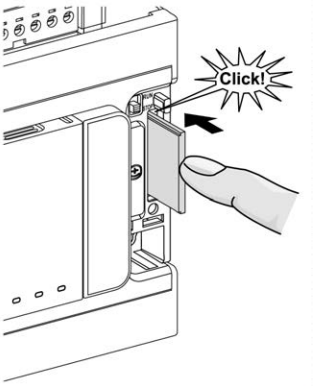
Failure to follow these instructions can result in equipment damage.

The following figure shows the SD card slot:



It is possible to set the Write-Control Tab to prevent write operations to the SD card. Push the tab up, as shown in the example on the right-hand side, to release the lock and enable writing to the SD card. Before using an SD card, read the manufacturer's instructions.



Step	Action
1	Insert the SD card into the SD card slot: 
2	Push until you hear it “click”: 

SD Card Slot Characteristics

Topic	Characteristics	Description
Supported type	Standard Capacity	SD (SDSC)
	High Capacity	SDHC
Global memory	Size	16 GB max.

TMASD1 Characteristics

Characteristics	Description
Card removal durability	Minimum 1000 times
File retention time	10 years @ 25 °C (77 °F)
Flash type	SLC NAND
Memory size	256 MB
Ambient operation temperature	–10 … +85°C (14...185 °F)
Storage temperature	–25 … +85°C (–13...185 °F)
Relative humidity	95% max. non-condensing
Write/Erase cycles	3,000,000 (approximately)

NOTE: The TMASD1 has been rigorously tested in association with the logic controller. For other commercially available cards, consult your local sales representative.

NOTE: The SD card can be used directly on your PC.

Status LED

The following figure shows the status LEDs:



The following table describes the SD card status LED:

Label	Description	LED		
		Color	Status	Description
SD	SD card	Green	On	Indicates that the SD card is being accessed.
			Off	Indicates no access.

Chapter 3
M241 Installation

Overview

This chapter provides installation safety guidelines, device dimensions, mounting instructions, and environmental specifications.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	M241 Logic Controller General Rules for Implementing	68
3.2	M241 Logic Controller Installation	73
3.3	M241 Electrical Requirements	87

Section 3.1
M241 Logic Controller General Rules for Implementing

What Is in This Section?

This section contains the following topics:

Topic	Page
Environmental Characteristics	69
Certifications and Standards	72

Environmental Characteristics

Enclosure Requirements

M241 Logic Controller system components are designed as Zone B, Class A industrial equipment according to IEC/CISPR Publication 11. If they are used in environments other than those described in the standard, or in environments that do not meet the specifications in this manual, the ability to meet electromagnetic compatibility requirements in the presence of conducted and/or radiated interference may be reduced.

All M241 Logic Controller system components meet European Community (CE) requirements for open equipment as defined by IEC/EN 61131-2. You must install them in an enclosure designed for the specific environmental conditions and to minimize the possibility of unintended contact with hazardous voltages. Use metal enclosures to improve the electromagnetic immunity of your M241 Logic Controller system. Use enclosures with a keyed locking mechanism to minimize unauthorized access.

Environmental Characteristics

All the M241 Logic Controller module components are electrically isolated between the internal electronic circuit and the input/output channels within the limits set forth and described by these environmental characteristics. For more information on electrical isolation, see the technical specifications of your particular controller found later in the current document. This equipment meets CE requirements as indicated in the table below. This equipment is intended for use in a Pollution Degree 2 industrial environment.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

The following table shows the general environmental characteristics:

Characteristic	Minimum Specification	Tested Range	
Standard compliance	IEC/EN 61131-2 IEC/EN 61010-2-201	–	
Ambient operating temperature	–	Horizontal installation	–10...55 °C (14...131 °F)
	–	Vertical installation	–10...50 °C (14...122 °F)
Storage temperature	–	–25...70 °C (- 13...158 °F)	
Relative humidity	–	Transport and storage	10...95 % (non-condensing)
		Operation	10...95 % (non-condensing)
Degree of pollution	IEC/EN 60664-1	2	
Degree of protection	IEC/EN 61131-2	IP20 with protective covers in place	
Corrosion immunity	–	Atmosphere free from corrosive gases	
Operating altitude	–	0...2000 m (0...6560 ft)	
Storage altitude	–	0...3000 m (0...9843 ft)	
Vibration resistance	IEC/EN 61131-2	Panel mounting or mounted on a top hat section rail (DIN rail)	3.5 mm (0.13 in) fixed amplitude from 5...8.4 Hz 9.8 m/s^2 (32.15 ft/s^2) (1 g_n) fixed acceleration from 8.4...150 Hz 10 mm (0.39 in) fixed amplitude from 5...8.7 Hz 29.4 m/s^2 (96.45 ft/s^2) (3 g_n) fixed acceleration from 8.7...150 Hz
Mechanical shock resistance	–	147 m/s^2 or 482.28 ft/s^2 (15 g_n) for a duration of 11 ms	

NOTE: The tested ranges may indicate values beyond that of the IEC Standard. However, our internal standards define what is necessary for industrial environments. In all cases, we uphold the minimum specification if indicated.

Electromagnetic Susceptibility

The M241 Logic Controller system meets electromagnetic susceptibility specifications as indicated in the following table:

Characteristic	Minimum Specification	Tested Range		
Electrostatic discharge	IEC/EN 61000-4-2	8 kV (air discharge) 4 kV (contact discharge)		
Radiated electromagnetic field	IEC/EN 61000-4-3	10 V/m (80...1000 MHz) 3 V/m (1.4...2 GHz) 1 V/m (2...3 GHz)		
Fast transients burst	IEC/EN 61000-4-4	24 Vdc main power lines	2 kV (CM[1] and DM[2])	
		24 Vdc I/Os	2 kV (clamp)	
		Relay output	1 kV (clamp)	
		Digital I/Os	1 kV (clamp)	
		Communication line	1 kV (clamp)	
Surge immunity	IEC/EN 61000-4-5 IEC/EN 61131-2	–	CM[1]	DM[2]
		DC Power lines	0.5 kV	0.5 kV
		Relay Outputs	–	–
		24 Vdc I/Os	–	–
		Shielded cable (between shield and ground)	1 kV	–
Induced electromagnetic field	IEC/EN 61000-4-6	10 Vrms (0.15...80 MHz)		
Conducted emission	IEC 61000-6-4	• 10...150 kHz: 120...69 dBµV/m QP • 150...1500 kHz: 79...63 dBµV/m QP • 1.5...30 MHz: 63 dBµV/m QP		
Radiated emission	IEC 61000-6-4	30...230 MHz: 40 dBµV/m QP 230...1000 MHz: 47 dBµV/m QP		

1 Common Mode
2 Differential Mode

NOTE: The tested ranges may indicate values beyond that of the IEC Standard. However, our internal standards define what is necessary for industrial environments. In all cases, we uphold the minimum specification if indicated.

Certifications and Standards

Introduction

The M241 Logic Controllers are designed to conform to the main national and international standards concerning electronic industrial control devices:

- IEC/EN 61131-2
- UL 508

The M241 Logic Controllers have obtained the following conformity marks:

- CE
- cULus
- CSA

For product compliance and environmental information (RoHS, REACH, PEP, EOLI, etc.), go to *www.schneider-electric.com/green-premium*.

Section 3.2
M241 Logic Controller Installation

What Is in This Section?

This section contains the following topics:

Topic	Page
Installation and Maintenance Requirements	74
M241 Logic Controller Mounting Positions and Clearances	77
Top Hat Section Rail (DIN rail)	80
Installing and Removing the Controller with Expansions	84
Direct Mounting on a Panel Surface	86

Installation and Maintenance Requirements

Before Starting

Read and understand this chapter before beginning the installation of your system.

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

Pay particular attention in conforming to any safety information, different electrical requirements, and normative standards that would apply to your machine or process in the use of this equipment.

Disconnecting Power

All options and modules should be assembled and installed before installing the control system on a mounting rail, onto a mounting plate or in a panel. Remove the control system from its mounting rail, mounting plate or panel before disassembling the equipment.

⚠ ⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

Programming Considerations

⚠ 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.

Operating Environment

In addition to the **Environmental Characteristics**, refer to **Product Related Information** in the beginning of the present document for important information regarding installation in hazardous locations for this specific equipment.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Install and operate this equipment according to the conditions described in the Environmental Characteristics.

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

Installation Considerations

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use appropriate safety interlocks where personnel and/or equipment hazards exist.
- Install and operate this equipment in an enclosure appropriately rated for its intended environment and secured by a keyed or tooled locking mechanism.
- Use the sensor and actuator power supplies only for supplying power to the sensors or actuators connected to the module.
- Power line and output circuits must be wired and fused in compliance with local and national regulatory requirements for the rated current and voltage of the particular equipment.
- Do not use this equipment in safety-critical machine functions unless the equipment is otherwise designated as functional safety equipment and conforming to applicable regulations and standards.
- Do not disassemble, repair, or modify this equipment.
- Do not connect any wiring to reserved, unused connections, or to connections designated as No Connection (N.C.).

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

NOTE: JDYX2 or JDYX8 fuse types are UL-recognized and CSA approved.

M241 Logic Controller Mounting Positions and Clearances

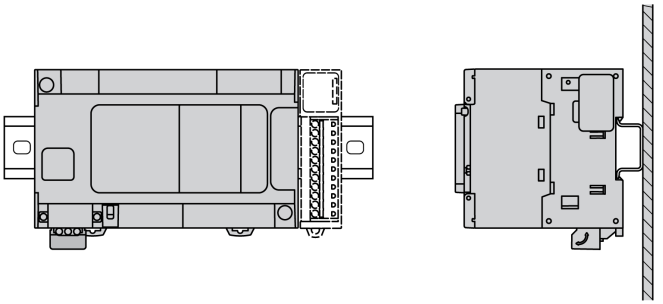
Introduction

This section describes the mounting positions for the M241 Logic Controller .

NOTE: Keep adequate spacing for proper ventilation and to maintain the operating temperature specified in the Environmental Characteristics *(see page 69)*.

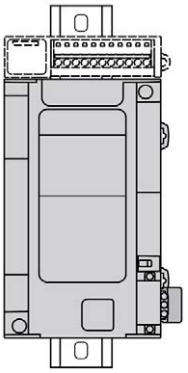
Correct Mounting Position

Whenever possible, the M241 Logic Controller should be mounted horizontally on a vertical plane as shown in the figure below:



Acceptable Mounting Positions

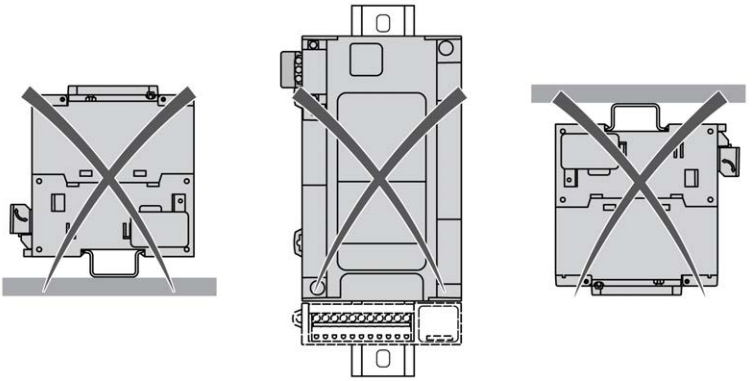
The M241 Logic Controller can also be mounted vertically with a temperature derating on a vertical plane as shown below.



NOTE: Expansion modules must be mounted above the logic controller.

Incorrect Mounting Position

The M241 Logic Controller should only be positioned as shown in Correct Mounting Position *(see page 77)* figure. The figures below show the incorrect mounting positions.



Minimum Clearances

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Place devices dissipating the most heat at the top of the cabinet and ensure adequate ventilation.
- Avoid placing this equipment next to or above devices that might cause overheating.
- Install the equipment in a location providing the minimum clearances from all adjacent structures and equipment as directed in this document.
- Install all equipment in accordance with the specifications in the related documentation.

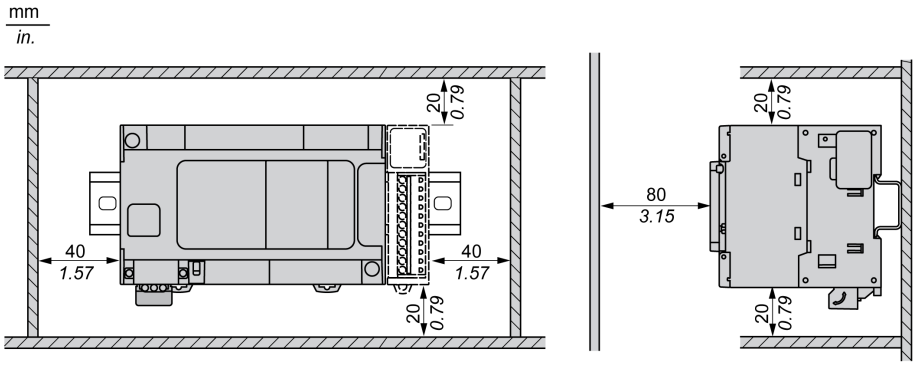
Failure to follow these instructions can result in death, serious injury, or equipment damage.

The M241 Logic Controller has been designed as an IP20 product and must be installed in an enclosure. Clearances must be respected when installing the product.

There are 3 types of clearances between:

- The M241 Logic Controller and all sides of the cabinet (including the panel door).
- The M241 Logic Controller terminal blocks and the wiring ducts. This distance reduces electromagnetic interference between the controller and the wiring ducts.
- The M241 Logic Controller and other heat generating devices installed in the same cabinet.

The following figure shows the minimum clearances that apply to all M241 Logic Controller references:



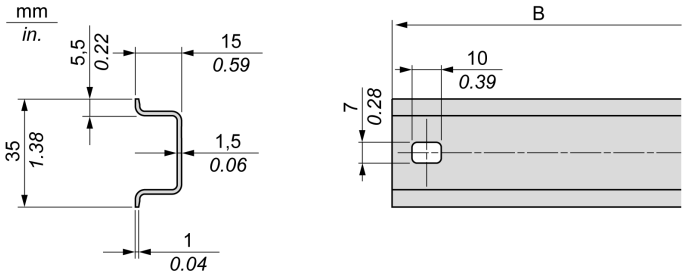
Top Hat Section Rail (DIN rail)

Dimensions of Top Hat Section Rail DIN Rail

You can mount the controller or receiver and its expansions on a 35 mm (1.38 in.) top hat section rail (DIN rail). It can be attached to a smooth mounting surface or suspended from a EIA rack or mounted in a NEMA cabinet.

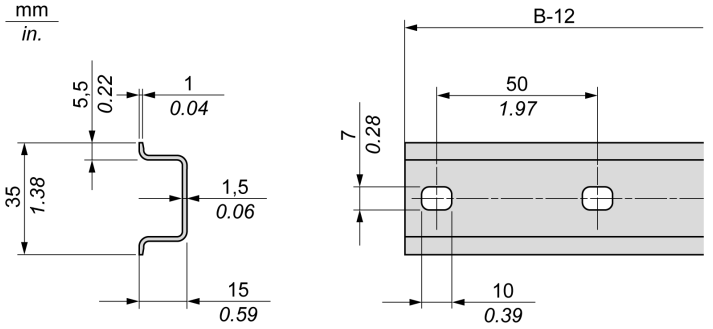
Symmetric Top Hat Section Rails (DIN Rail)

The following illustration and table show the references of the top hat section rails (DIN rail) for the wall-mounting range:



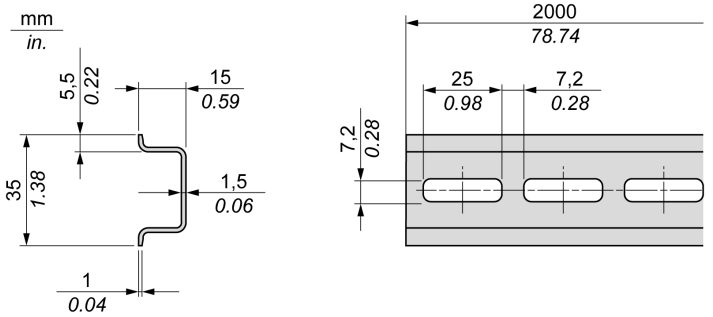
Reference	Type	Rail Length (B)
NSYSDR50A	A	450 mm (17.71 in.)
NSYSDR60A	A	550 mm (21.65 in.)
NSYSDR80A	A	750 mm (29.52 in.)
NSYSDR100A	A	950 mm (37.40 in.)

The following illustration and table show the references of the symmetric top hat section rails (DIN rail) for the metal enclosure range:



Reference	Type	Rail Length (B-12 mm)
NSYSDR60	A	588 mm (23.15 in.)
NSYSDR80	A	788 mm (31.02 in.)
NSYSDR100	A	988 mm (38.89 in.)
NSYSDR120	A	1188 mm (46.77 in.)

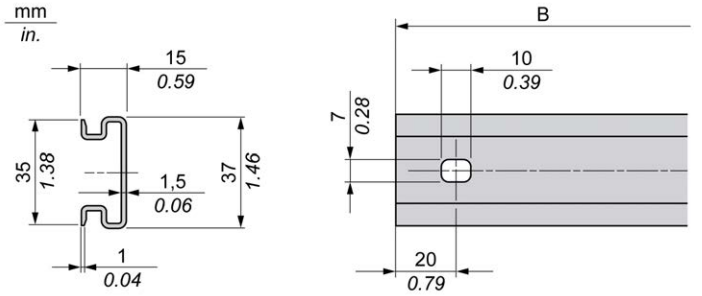
The following illustration and table shows the references of the symmetric top hat section rails (DIN rail) of 2000 mm (78.74 in.):



Reference	Type	Rail Length
NSYSDR200[1]	A	2000 mm (78.74 in.)
NSYSDR200D[2]	A	
1 Unperforated galvanized steel **2** Perforated galvanized steel		

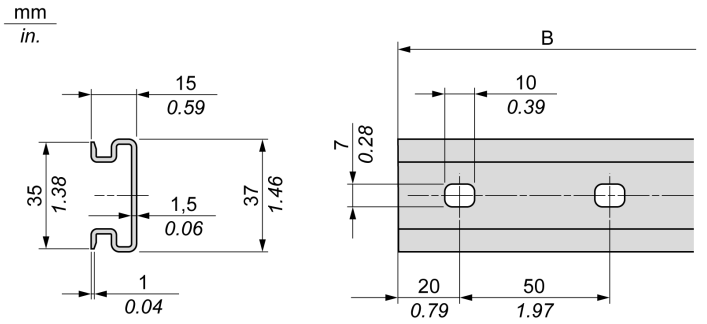
Double-Profile Top Hat Section Rails (DIN rail)

The following illustration and table show the references of the double-profile top hat section rails (DIN rails) for the wall-mounting range:



Reference	Type	Rail Length (B)
NSYDPR25	W	250 mm (9.84 in.)
NSYDPR35	W	350 mm (13.77 in.)
NSYDPR45	W	450 mm (17.71 in.)
NSYDPR55	W	550 mm (21.65 in.)
NSYDPR65	W	650 mm (25.60 in.)
NSYDPR75	W	750 mm (29.52 in.)

The following illustration and table show the references of the double-profile top hat section rails (DIN rail) for the floor-standing range:



Reference	Type	Rail Length (B)
NSYDPR60	F	588 mm (23.15 in.)
NSYDPR80	F	788 mm (31.02 in.)
NSYDPR100	F	988 mm (38.89 in.)
NSYDPR120	F	1188 mm (46.77 in.)

Installing and Removing the Controller with Expansions

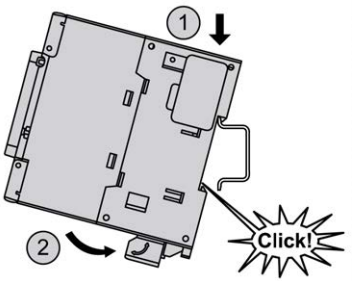
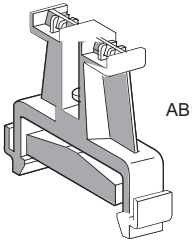
Overview

This section describes how to install and remove the controller with its expansion modules from a top hat section rail (DIN rail).

To assemble expansion modules to a controller or receiver module, or to other modules, refer to the respective expansion modules hardware guide(s).

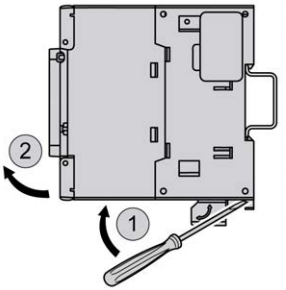
Installing a Controller with its Expansions on a DIN Rail

The following procedure describes how to install a controller with its expansion modules on a top hat section rail (DIN rail):

Step	Action
1	Fasten the top hat section rail (DIN rail) to a panel surface using screws.
2	Position the top groove of the controller and its expansion modules on the top edge of the DIN rail and press the assembly against the top hat section rail (DIN rail) until you hear the top hat section rail (DIN rail) clip snap into place. 
3	Place 2 terminal block end clamps on both sides of the controller and expansion module assembly.  AB1AB8P35 **NOTE:** Type ABB8P35 or equivalent terminal block end clamps help minimize sideways movement and improve the shock and vibration characteristics of the controller and expansion module assembly.

Removing a Controller with its Expansions from a Top Hat Section Rail (DIN Rail)

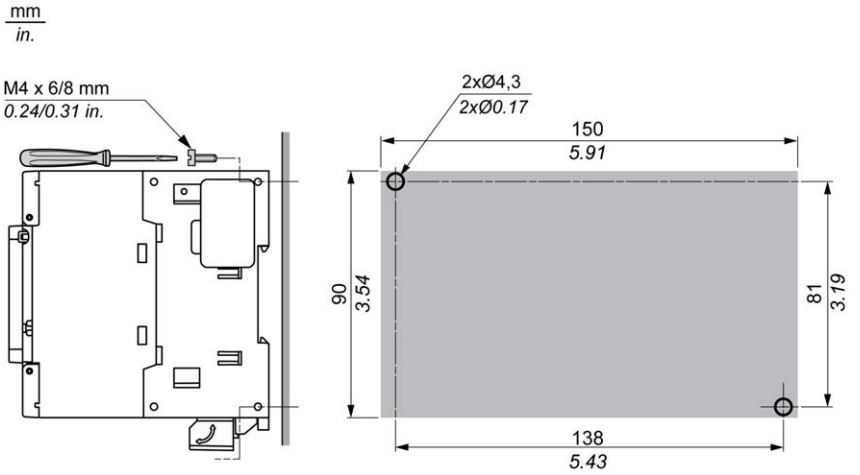
The following procedure describes how to remove a controller with its expansion modules from a top hat section rail (DIN rail):

Step	Action
1	Remove all power from your controller and expansion modules.
2	Insert a flat screwdriver into the slot of the top hat section rail (DIN rail) clip. 
3	Pull down the DIN rail clip.
4	Pull the controller and its expansion modules from the top hat section rail (DIN rail) from the bottom.

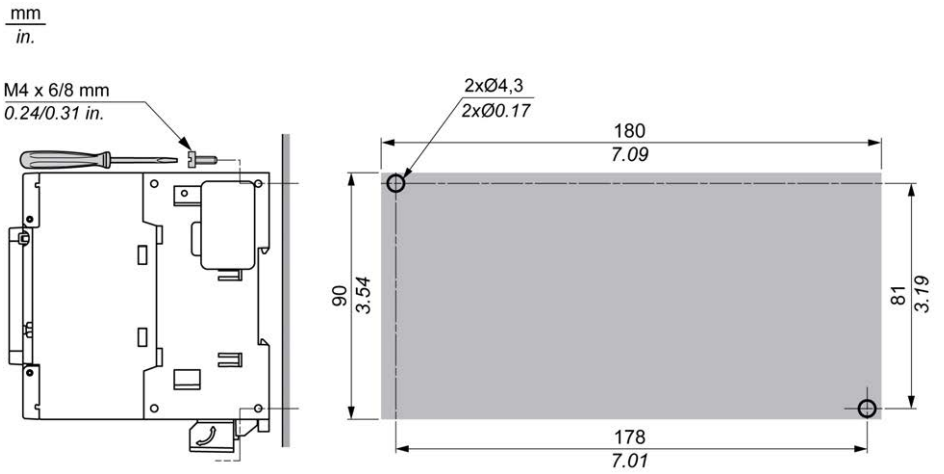
Direct Mounting on a Panel Surface

Mounting Hole Layout

The following diagram shows the mounting hole layout for M241 Logic Controller with 24 I/O channels:



The following diagram shows the mounting hole layout for M241 Logic Controller with 40 I/O channels:



Section 3.3
M241 Electrical Requirements

What Is in This Section?

This section contains the following topics:

Topic	Page
Wiring Best Practices	88
DC Power Supply Characteristics and Wiring	94
AC Power Supply Characteristics and Wiring	98
Grounding the M241 System	100

Wiring Best Practices

Overview

This section describes the wiring guidelines and associated best practices to be respected when using the M241 Logic Controller system.

⚠ ⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

⚠ 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.

Wiring Guidelines

The following rules must be applied when wiring a M241 Logic Controller system:

- I/O and communication wiring must be kept separate from the power wiring. Route these 2 types of wiring in separate cable ducting.
- Verify that the operating conditions and environment are within the specification values.
- Use proper wire sizes to meet voltage and current requirements.
- Use copper conductors (required).
- Use twisted pair, shielded cables for analog, and/or fast I/O.
- Use twisted pair, shielded cables for networks, and fieldbus.

Use shielded, properly grounded cables for all analog and high-speed inputs or outputs and communication connections. If you do not use shielded cable for these connections, electromagnetic interference can cause signal degradation. Degraded signals can cause the controller or attached modules and equipment to perform in an unintended manner.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use shielded cables for all fast I/O, analog I/O and communication signals.
- Ground cable shields for all analog I/O, fast I/O and communication signals at a single point[1].
- Route communication and I/O cables separately from power cables.

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

[1]Multipoint grounding is permissible if connections are made to an equipotential ground plane dimensioned to help avoid cable shield damage in the event of power system short-circuit currents.

For more details, refer to Grounding Shielded Cables *(see page 100)*.

NOTE: Surface temperatures may exceed 60 °C (140 °F).
To conform to IEC 61010 standards, route primary wiring (wires connected to power mains) separately and apart from secondary wiring (extra low voltage wiring coming from intervening power sources). If that is not possible, double insulation is required such as conduit or cable gains.

Rules for Removable Screw Terminal Block

The following tables show the cable types and wire sizes for a **5.08 pitch** removable screw terminal block (I/Os and power supply):

mm / in. 7 / 0.28								
mm^2	0.2…2.5	0.2…2.5	0.25…2.5	0.25…2.5	2 x 0.2…1	2 x 0.2…1.5	2 x 0.25…1	2 x 0.5...1.5
AWG	*24…14*	*24…14*	*23...14*	*23...14*	*2 x 24…17*	*2 x 24…16*	*2 x 23…17*	*2 x 20...16*

Ø 3,5 mm (*0.14 in.*)	C	N•m	0.5...0.6
		lb-in	*4.42...5.31*

The use of copper conductors is required.

DANGER

LOOSE WIRING CAUSES ELECTRIC SHOCK

Tighten connections in conformance with the torque specifications.

Failure to follow these instructions will result in death or serious injury.

DANGER

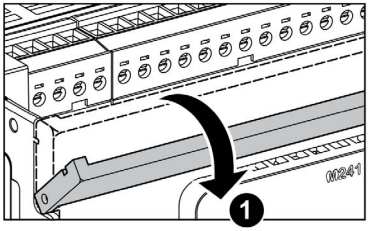
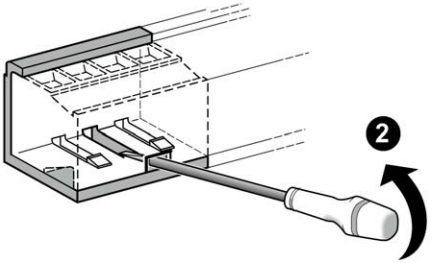
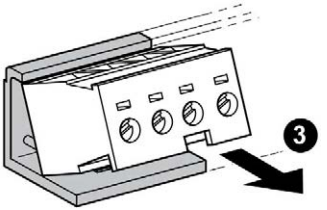
FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

Removing the I/O Terminal Block

The following figure shows the removal of the I/O terminal block from the M241 Logic Controller:

Step	Action
1	Remove power from your controller.
2	Pull down the protective cache: 
3	Press with a screwdriver through the terminal block front hole: 
4	Remove the terminal block: 

Protecting Outputs from Inductive Load Damage

Depending on the load, a protection circuit may be needed for the outputs on the controllers and certain modules. Inductive loads using DC voltages may create voltage reflections resulting in overshoot that will damage or shorten the life of output devices.

⚠ CAUTION

OUTPUT CIRCUIT DAMAGE DUE TO INDUCTIVE LOADS

Use an appropriate external protective circuit or device to reduce the risk of inductive direct current load damage.

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

If your controller or module contains relay outputs, these types of outputs can support up to 240 Vac. Inductive damage to these types of outputs can result in welded contacts and loss of control. Each inductive load must include a protection device such as a peak limiter, RC circuit or flyback diode. Capacitive loads are not supported by these relays.

⚠ WARNING

RELAY OUTPUTS WELDED CLOSED

- Always protect relay outputs from inductive alternating current load damage using an appropriate external protective circuit or device.
- Do not connect relay outputs to capacitive loads.

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

AC-driven contactor coils are, under certain circumstances, inductive loads that generate pronounced high-frequency interference and electrical transients when the contactor coil is de-energized. This interference may cause the logic controller to detect an I/O bus error.

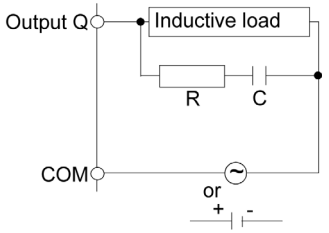
⚠ WARNING

CONSEQUENTIAL LOSS OF CONTROL

Install an RC surge suppressor or similar means, such as an interposing relay, on each TM3 expansion module relay output when connecting to AC-driven contactors or other forms of inductive loads.

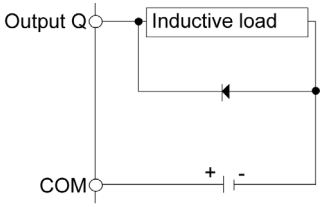
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Protective circuit A: this protection circuit can be used for both AC and DC load power circuits.



C Value from 0.1 to 1 µF
R Resistor of approximately the same resistance value as the load

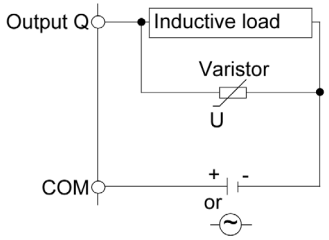
Protective circuit B: this protection circuit can be used for DC load power circuits.



Use a diode with the following ratings:

- Reverse withstand voltage: power voltage of the load circuit x 10.
- Forward current: more than the load current.

Protective circuit C: this protection circuit can be used for both AC and DC load power circuits.



In applications where the inductive load is switched on and off frequently and/or rapidly, ensure that the continuous energy rating (J) of the varistor exceeds the peak load energy by 20 % or more.

DC Power Supply Characteristics and Wiring

Overview

This section provides the characteristics and the wiring diagrams of the DC power supply.

DC Power Supply Voltage Range

If the specified voltage range is not maintained, outputs may not switch as expected. Use appropriate safety interlocks and voltage monitoring circuits.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

DC Power Supply Requirements

The M241 Logic Controllerand associated I/O (TM2, TM3, and embedded I/O) require power supplies with a nominal voltage of 24 Vdc. The 24 Vdc power supplies must be rated Safety Extra Low Voltage (SELV) or Protective Extra Low Voltage (PELV) according to IEC 61140. These power supplies are isolated between the electrical input and output circuits of the power supply.

⚠ WARNING

POTENTIAL OF OVERHEATING AND FIRE

- Do not connect the equipment directly to line voltage.
- Use only isolating PELV power supplies and circuits to supply power to the equipment[1].

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

[1] For compliance to UL (Underwriters Laboratories) requirements, the power supply must also conform to the various criteria of NEC Class 2, and be inherently current limited to a maximum power output availability of less than 100 VA (approximately 4 A at nominal voltage), or not inherently limited but with an additional protection device such as a circuit breaker or fuse meeting the requirements of clause 9.4 Limited-energy circuit of UL 61010-1. In all cases, the current limit should never exceed that of the electric characteristics and wiring diagrams for the equipment described in the present documentation. In all cases, the power supply must be grounded, and you must separate Class 2 circuits from other circuits. If the indicated rating of the electrical characteristics or wiring diagrams are greater than the specified current limit, multiple Class 2 power supplies may be used.

Controller DC Characteristics

The following table shows the DC power supply characteristics required for the controller:

Characteristic		Value	
Rated voltage		24 Vdc	
Power supply voltage range		20.4...28.8 Vdc	
Power interruption time		1 ms at 24 Vdc	
Maximum inrush current		50 A	
Power consumption		32.6 W	max. 40.4 W(1)
Isolation	between DC power supply and internal logic	Not isolated	
	between DC power supply and protective earth ground (PE)	500 Vac	
(1) Controller + 7 TM3 expansion modules			

Power interruption

The TM241C••24T / TM241C•40T / TM241C••24U and TM241C•40U must be supplied by an external 24 V power supply equipment. During power interruptions, the M241 Logic Controller, associated to the suitable power supply, is able to continue normal operation for a minimum of 10 ms as specified by IEC standards.

The TM241C••24T / TM241C•40T / TM241C••24U and TM241C•40U must be supplied by an external 24 V power supply equipment. During power interruptions, the M241 Logic Controller, associated to the suitable power supply, is able to continue normal operation for a minimum of 10 ms as specified by IEC standards.

When planning the management of the power supplied to the controller, you must consider the power interruption duration due to the fast cycle time of the controller.

There could potentially be many scans of the logic and consequential updates to the I/O image table during the power interruption, while there is no external power supplied to the inputs, the outputs or both depending on the power system architecture and power interruption circumstances.

⚠ WARNING

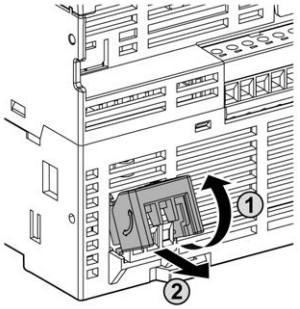
UNINTENDED EQUIPMENT OPERATION

- Individually monitor each source of power used in the controller system including input power supplies, output power supplies and the power supply to the controller to allow appropriate system shutdown during power system interruptions.
- The inputs monitoring each of the power supply sources must be unfiltered inputs.

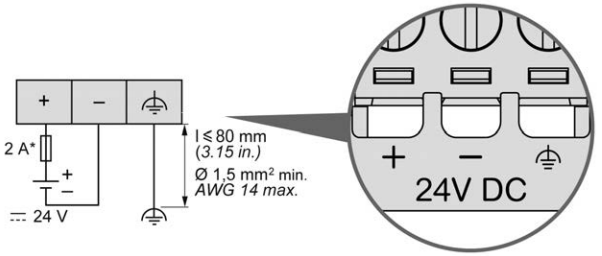
Failure to follow these instructions can result in death, serious injury, or equipment damage.

DC Power Supply Wiring Diagram

The following figure shows the power supply terminal block removal procedure:



The following figure shows the wiring of the DC power supply:



* Type T fuse

For more information, refer to the 5.08 pitch Rules for Removable Screw Terminal block *(see page 90)*.

AC Power Supply Characteristics and Wiring

Overview

This section provides the wiring diagrams and the characteristics of the AC power supply.

AC Power Supply Voltage Range

If the specified voltage range is not maintained, outputs may not switch as expected. Use appropriate safety interlocks and voltage monitoring circuits.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

Controller AC Characteristics

The following table shows the AC power supply characteristics:

Characteristic		Value
Voltage	rated	100...240 Vac
	limit (including ripple)	85...264 Vac
Frequency		50/60 Hz
Power interruption time	at 100 Vac	10 ms
Maximum inrush current	at 240 Vac	42.5 A

Characteristic		Value
Typical power consumption	at 100 Vac	78 VA
	at 240 Vac	98.4 VA
Isolation	between AC power supply and internal logic	1780 Vac
	between AC power supply and protective earth ground (PE)	2500 Vdc

Power interruption

The duration of power interruptions where the M241 Logic Controller is able to continue normal operation varies depending upon the load to the power supply of the controller, but generally a minimum of 10 ms is maintained as specified by IEC standards.

If there is a minimum load on the controller power supply, the interruption can be as long as 400 ms.

When planning the management of the power supplied to the controller, you must consider the duration due to the fast cycle time.

There could potentially be many scans of the logic and consequential updates to the I/O image table during the power interruption, while there is no external power supplied to the inputs, the outputs or both depending on the power system architecture and power interruption circumstances.

⚠ WARNING

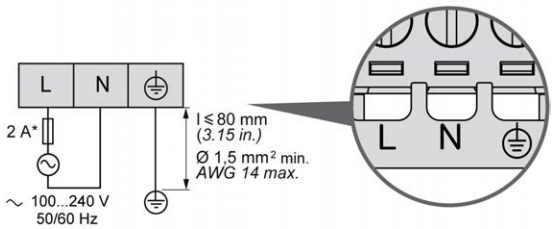
UNINTENDED EQUIPMENT OPERATION

- Individually monitor each source of power used in the Modicon M241 Logic Controller system including input power supplies, output power supplies and the power supply to the controller to allow appropriate system shutdown during power system interruptions.
- The inputs monitoring each of the power supply sources must be unfiltered inputs.

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

AC Power Supply Wiring Diagram

The following figure shows the wiring of the AC power supply:



* Use an external, slow-blow, type T fuse.

Grounding the M241 System

Overview

To help minimize the effects of electromagnetic interference, cables carrying the fast I/O, analog I/O, and field bus communication signals must be shielded.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use shielded cables for all fast I/O, analog I/O, and communication signals.
- Ground cable shields for all fast I/O, analog I/O, and communication signals at a single point[1].
- Route communications and I/O cables separately from power cables.

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

[1]Multipoint grounding is permissible if connections are made to an equipotential ground plane dimensioned to help avoid cable shield damage in the event of power system short-circuit currents.

The use of shielded cables requires compliance with the following wiring rules:

- For protective ground connections (PE), metal conduit or ducting can be used for part of the shielding length, provided there is no break in the continuity of the ground connections. For functional ground (FE), the shielding is intended to attenuate electromagnetic interference and the shielding must be continuous for the length of the cable. If the purpose is both functional and protective, as is often the case for communication cables, the cable must have continuous shielding.
- Wherever possible, keep cables carrying one type of signal separate from the cables carrying other types of signals or power.

Protective Ground (PE) on the Backplane

The protective ground (PE) should be connected to the conductive backplane by a heavy-duty wire, usually a braided copper cable with the maximum allowable cable section.

Shielded Cables Connections

Cables carrying the fast I/O, analog I/O, and field bus communication signals must be shielded. The shielding must be securely connected to ground. The fast I/O and analog I/O shields may be connected either to the functional ground (FE) or to the protective ground (PE) of your M241 Logic Controller. The field bus communication cable shields must be connected to the protective ground (PE) with a connecting clamp secured to the conductive backplane of your installation.

⚠ WARNING

ACCIDENTAL DISCONNECTION FROM PROTECTIVE GROUND (PE)

- Do not use the TM2XMTGB Grounding Plate to provide a protective ground (PE).
- Use the TM2XMTGB Grounding Plate only to provide a functional ground (FE).

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

The shielding of the Modbus cable must be connected to the protective ground (PE).

⚠⚠ DANGER

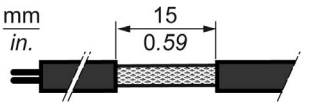
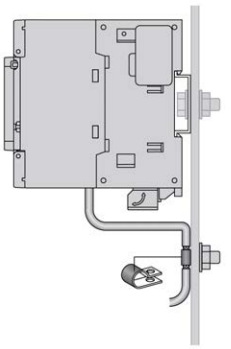
HAZARD OF ELECTRIC SHOCK

- The grounding terminal connection (PE) must be used to provide a protective ground at all times.
- Make sure that an appropriate, braided ground cable is attached to the PE/PG ground terminal before connecting or disconnecting the network cable to the equipment.

Failure to follow these instructions will result in death or serious injury.

Protective Ground (PE) Cable Shielding

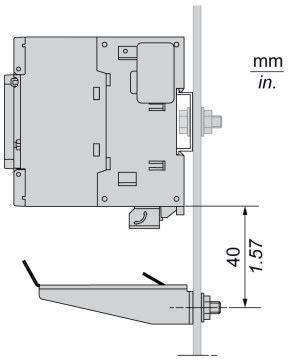
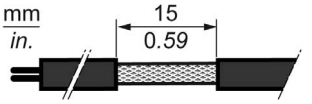
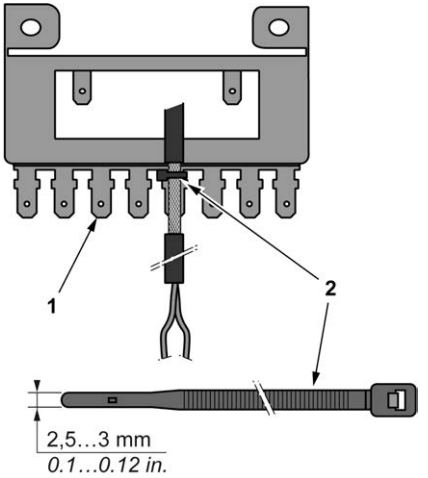
To ground the shield of a cable through a grounding clamp:

Step	Description	
1	Strip the shielding for a length of 15 mm (0.59 in.)	
2	Attach the cable to the conductive backplane plate by attaching the grounding clamp to the stripped part of the shielding as close as possible to the M241 Logic Controller system base.	

NOTE: The shielding must be clamped securely to the conductive backplane to ensure a good contact.

Functional Ground (FE) Cable Shielding

To connect the shield of a cable through the Grounding Bar:

Step	Description	
1	Install the Grounding Bar *(see Modicon TM2, Digital I/O Modules, Hardware Guide)* directly on the conductive backplane below the M241 Logic Controller system as illustrated.	
2	Strip the shielding for a length of 15 mm (0.59 in.	
3	Tightly clamp on the blade connector **(1)** using nylon fastener **(2)**(width 2.5...3 mm (0.1...0.12 in.)) and appropriate tool.	

NOTE: Use the TM2XMTGB Grounding Bar for Functional Ground (FE) connections.

Part II
Modicon M241 Logic Controller

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	TM241C24R	107
5	TM241CE24R	113
6	TM241CEC24R	119
7	TM241C24T	125
8	TM241CE24T	131
9	TM241CEC24T	137
10	TM241C24U	143
11	TM241CE24U	149
12	TM241CEC24U	155
13	TM241C40R	161
14	TM241CE40R	167
15	TM241C40T	173
16	TM241CE40T	179
17	TM241C40U	185
18	TM241CE40U	191
19	Embedded I/O Channels	197

Chapter 4
TM241C24R

TM241C24R Presentation

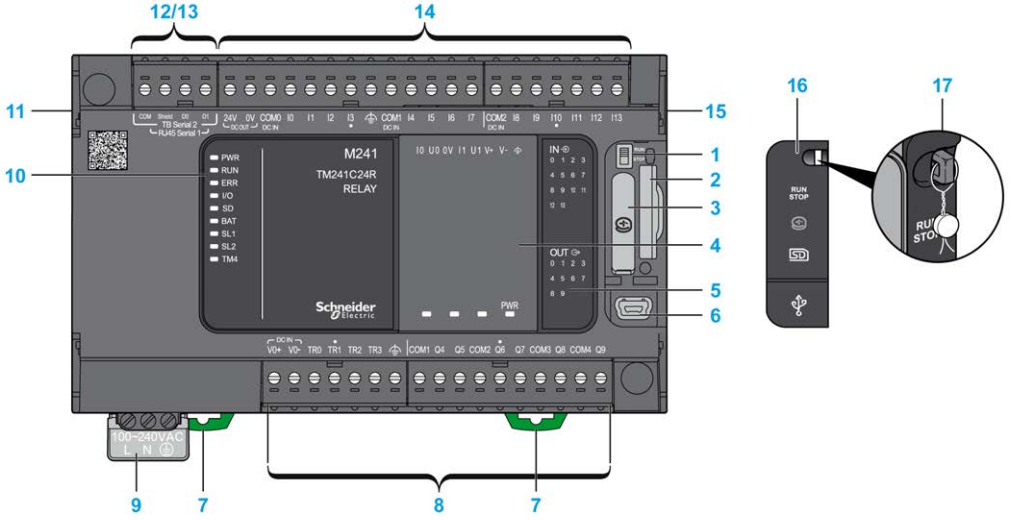
Overview

TM241C24R logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 relay outputs (2 A)
- Communication port
  - 2 serial line ports
  - 1 USB mini-B programming port

Description

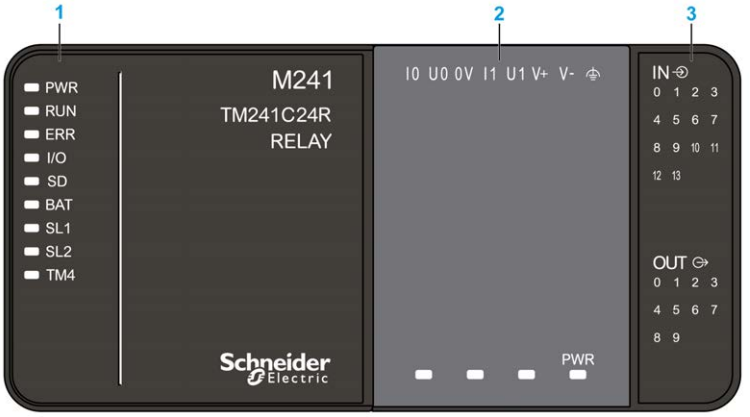
The following figure shows the different components of the TM241C24R logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Relay Outputs Status LEDs *(see page 207)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded relay outputs	Relay Outputs *(see page 206)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	100...240 Vac 50/60 Hz power supply	AC Power Supply Characteristics and Wiring *(see page 98)*
10	Status LEDs	–
11	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
12	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
13	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
14	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
15	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
16	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
17	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

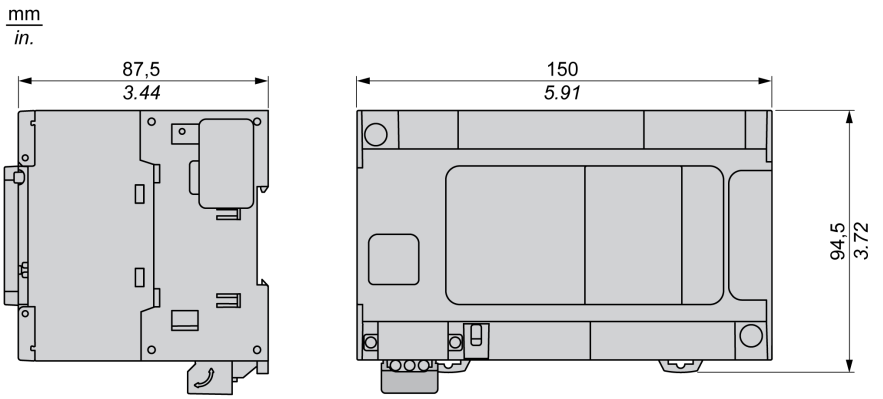
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	Indicates that an operating system error has been detected.	Restricted	No
			Fast flashing	Indicates that the controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 5
TM241CE24R

TM241CE24R Presentation

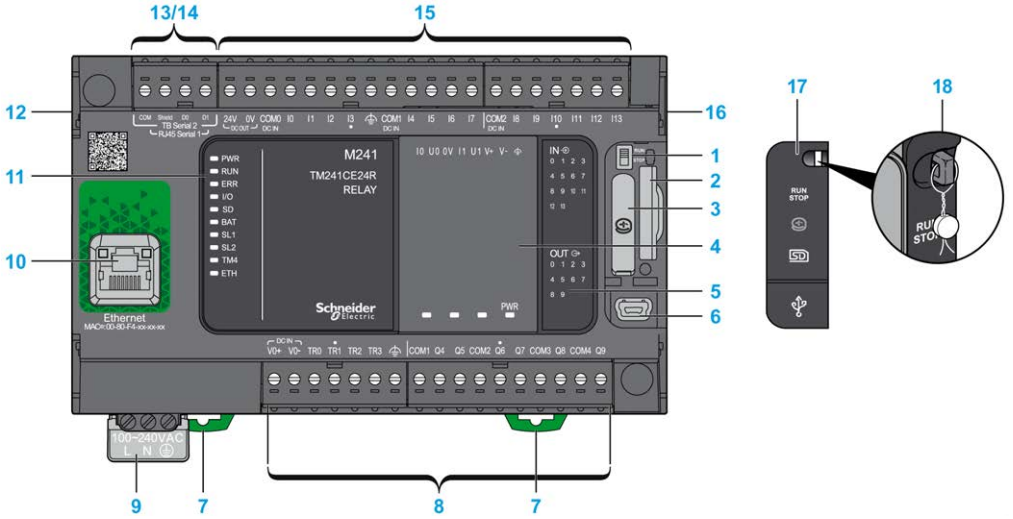
Overview

TM241CE24R logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 relay outputs (2 A)
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 USB mini-B programming port

Description

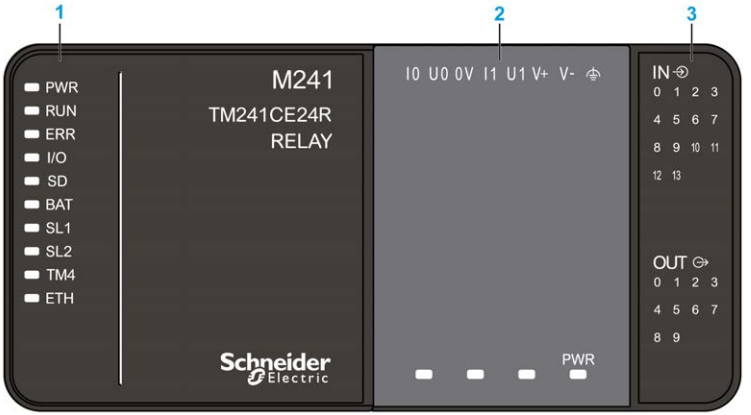
The following figure shows the different components of the TM241CE24R logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Relay Outputs Status LEDs *(see page 207)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded relay outputs	Relay Outputs *(see page 206)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	100...240 Vac 50/60 Hz power supply	AC Power Supply Characteristics and Wiring *(see page 98)*
10	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
11	Status LEDs	–
12	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
13	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
14	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
15	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
16	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
17	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
18	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

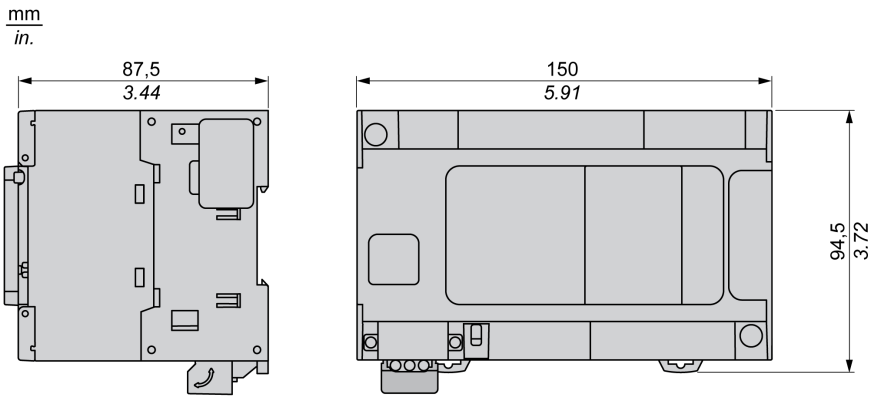
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, or Ethernet port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 6
TM241CEC24R

TM241CEC24R Presentation

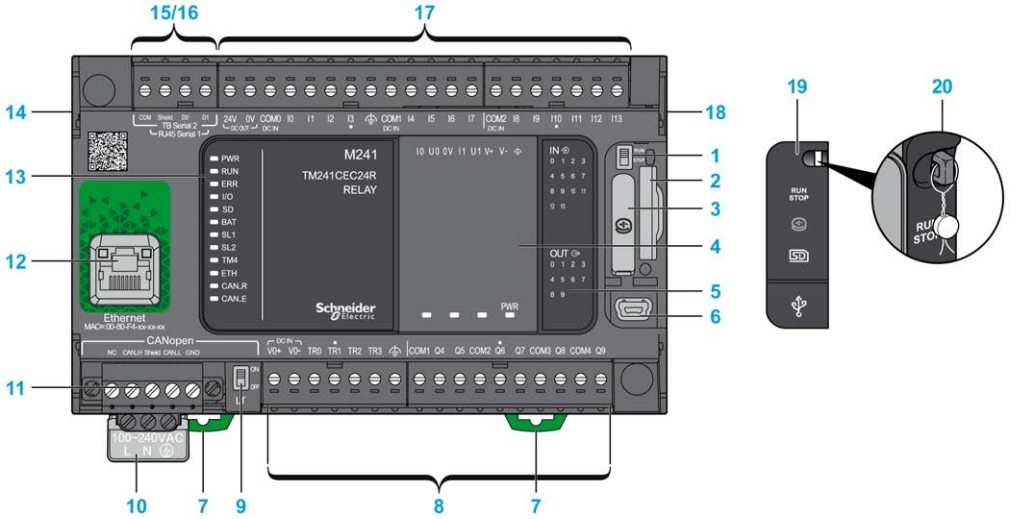
Overview

TM241CEC24R logic controller has:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 relay outputs (2 A)
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 CANopen port
  - 1 USB mini-B programming port

Description

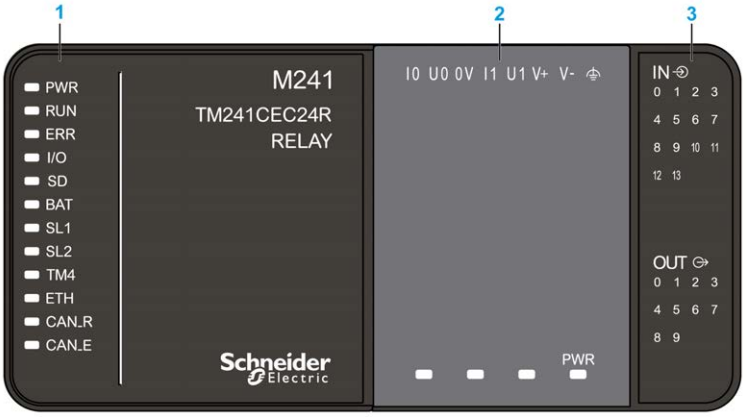
The following figure shows the different components of the TM241CEC24R logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Relay Outputs Status LEDs *(see page 207)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded relay outputs	Relay Outputs *(see page 206)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	CANopen Line termination switch	CANopen Port *(see page 228)*
10	100...240 Vac 50/60 Hz power supply	AC Power Supply Characteristics and Wiring *(see page 98)*
11	CANopen port / Screw terminal block type	–
12	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
13	Status LEDs	–
14	TM4 bus connector	–
15	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
16	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
17	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
18	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
19	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
20	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

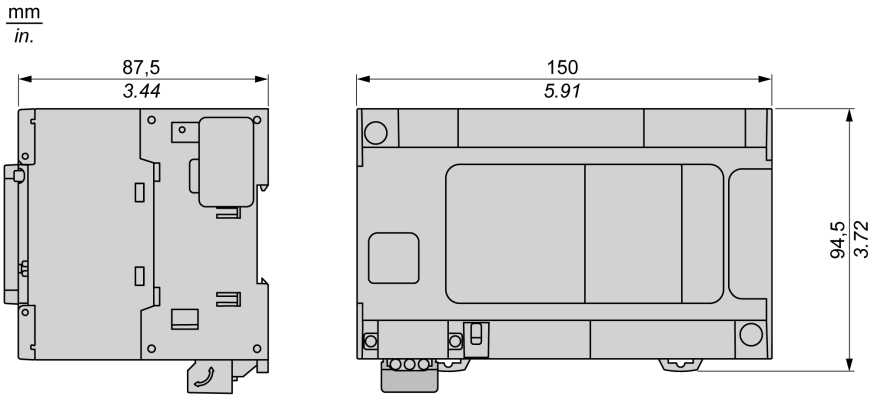
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, Ethernet port or CANopen port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
CAN R	CANopen running status	Green	On	Indicates that the CANopen bus is operational.		
			Off	Indicates that the CANopen master is configured.		
			Flashing	Indicates that the CANopen bus is being initialized.		
			1 flash per second	Indicates that the CANopen bus is stopped.		
CAN E	CANopen error	Red	On	Indicates that the CANopen bus is stopped (BUS OFF).		
			Off	Indicates no CANopen detected error.		
			Flashing	Indicates that the CANopen bus is not valid.		
			1 flash per second	Indicates that the controller has detected that the maximum number of error frames has been reached or exceeded.		
			2 flashes per second	Indicates that the controller has detected either a Node Guarding or a Heartbeat event.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 7
TM241C24T

TM241C24T Presentation

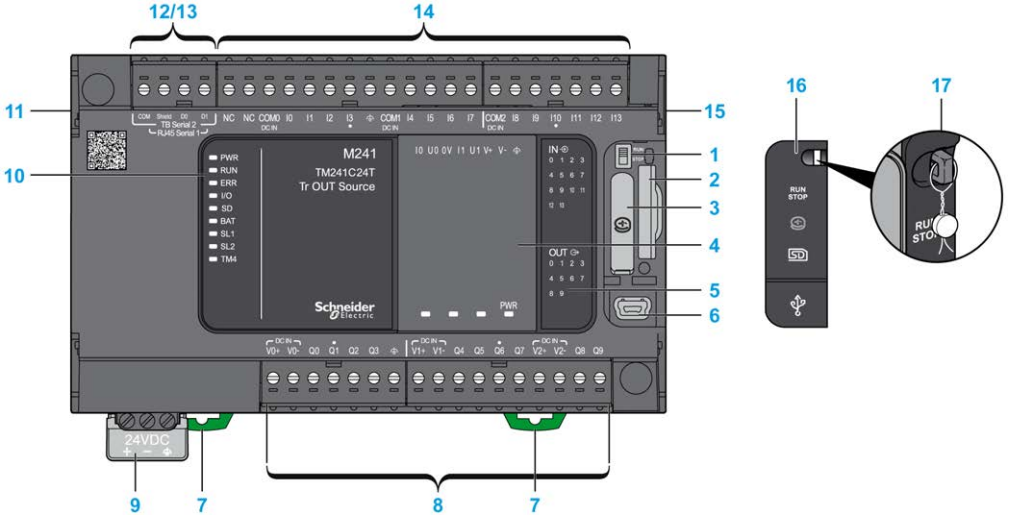
Overview

TM241C24T logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 regular outputs
- Communication port
  - 2 serial line ports
  - 1 USB mini-B programming port

Description

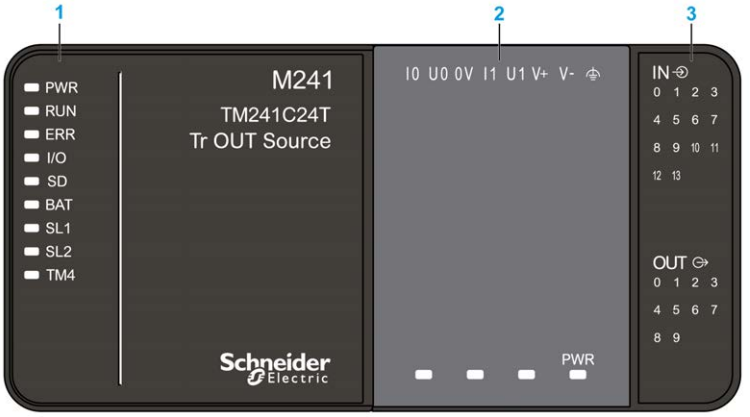
The following figure shows the different components of the TM241C24T logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Status LEDs	–
11	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
12	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
13	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
14	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
15	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
16	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
17	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

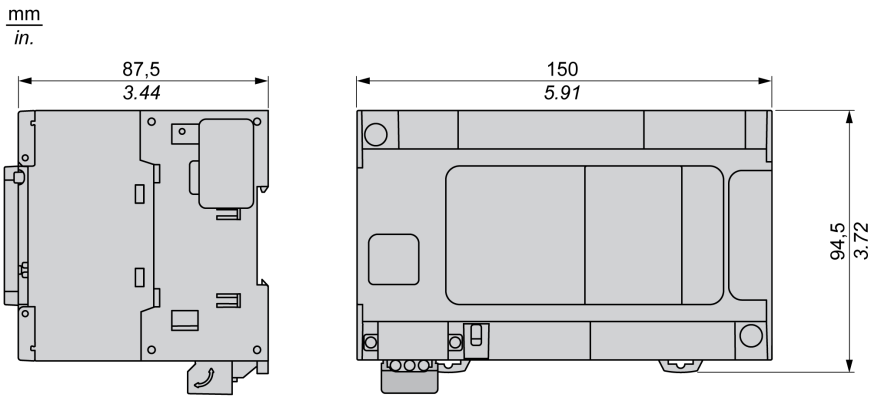
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	Indicates that an operating system error has been detected.	Restricted	No
			Fast flashing	Indicates that the controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 8
TM241CE24T

TM241CE24T Presentation

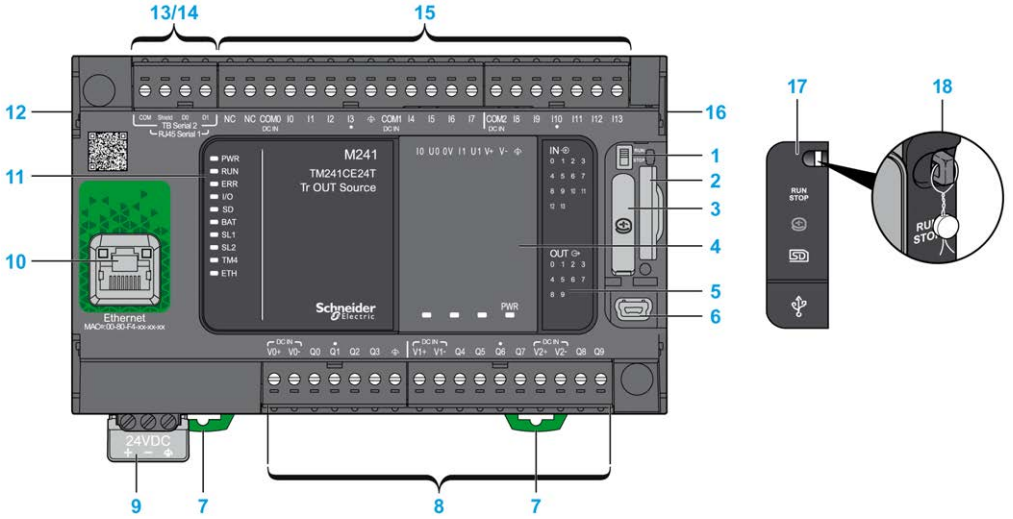
Overview

TM241CE24T logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 regular outputs
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 USB mini-B programming port

Description

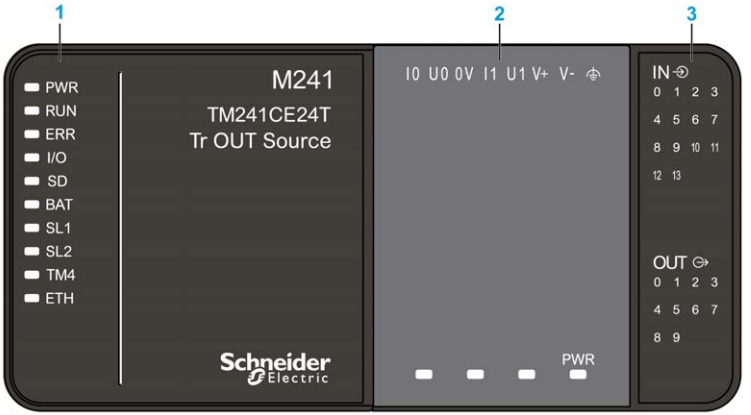
The following figure shows the different components of the TM241CE24T logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
11	Status LEDs	–
12	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
13	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
14	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
15	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
16	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
17	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
18	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

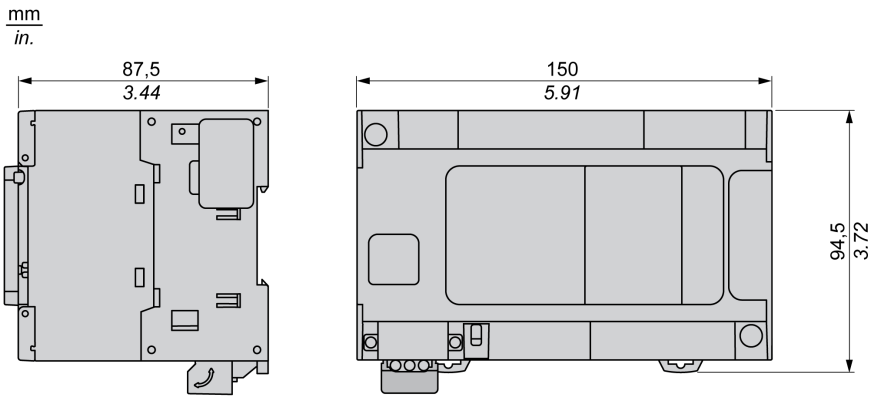
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, or Ethernet port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 9
TM241CEC24T

TM241CEC24T Presentation

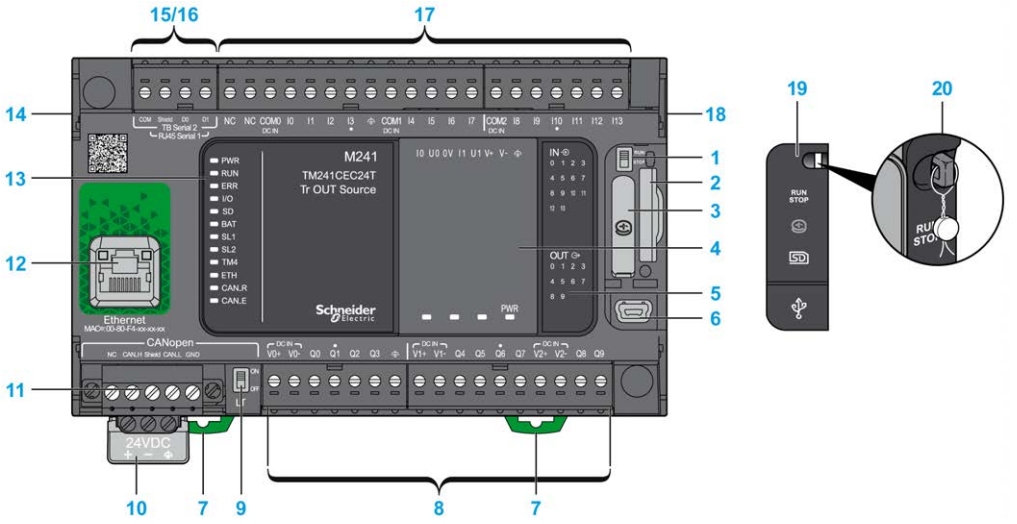
Overview

TM241CEC24T logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 regular outputs
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 CANopen port
  - 1 USB mini-B programming port

Description

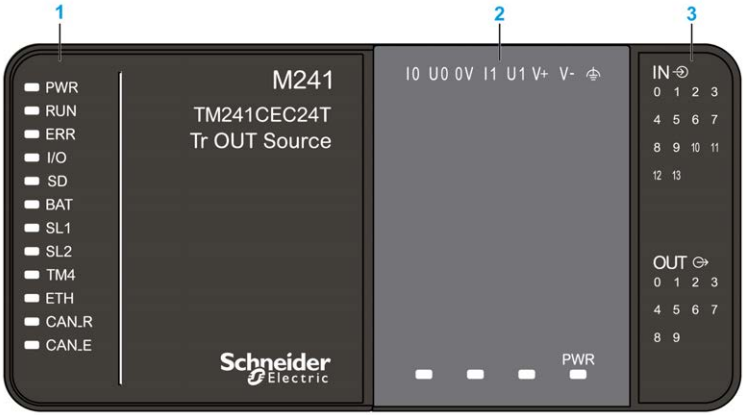
The following figure shows the different components of the TM241CEC24T logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	CANopen Line termination switch	CANopen Port *(see page 228)*
10	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
11	CANopen port / Screw terminal block type	–
12	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
13	Status LEDs	–
14	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
15	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
16	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
17	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
18	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
19	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
20	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

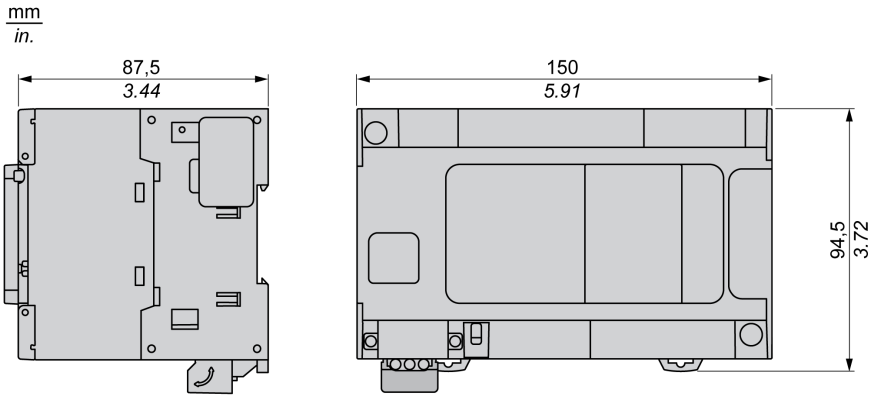
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, Ethernet port or CANopen port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
CAN R	CANopen running status	Green	On	Indicates that the CANopen bus is operational.		
			Off	Indicates that the CANopen master is configured.		
			Flashing	Indicates that the CANopen bus is being initialized.		
			1 flash per second	Indicates that the CANopen bus is stopped.		
CAN E	CANopen error	Red	On	Indicates that the CANopen bus is stopped (BUS OFF).		
			Off	Indicates no CANopen detected error.		
			Flashing	Indicates that the CANopen bus is not valid.		
			1 flash per second	Indicates that the controller has detected that the maximum number of error frames has been reached or exceeded.		
			2 flashes per second	Indicates that the controller has detected either a Node Guarding or a Heartbeat event.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 10
TM241C24U

TM241C24U Presentation

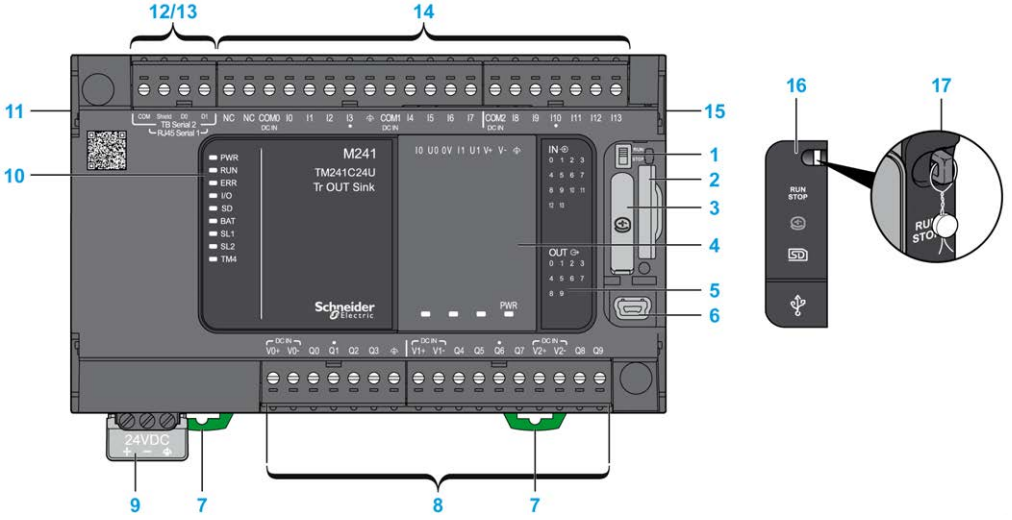
Overview

TM241C24U logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 regular outputs
- Communication port
  - 2 serial line ports
  - 1 USB mini-B programming port

Description

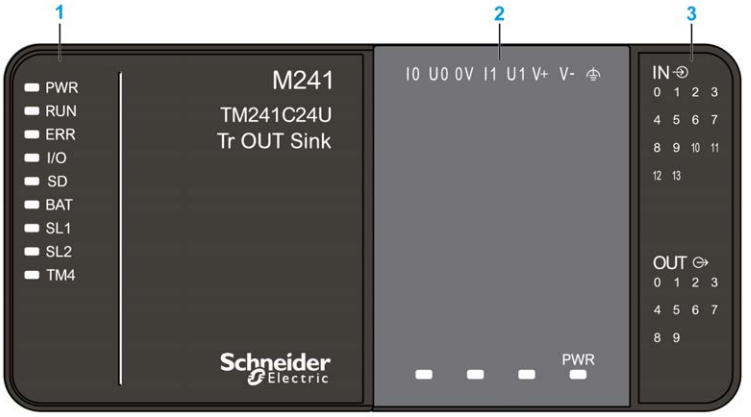
The following figure shows the different components of the TM241C24U logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Status LEDs	–
11	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
12	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
13	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
14	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
15	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
16	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
17	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

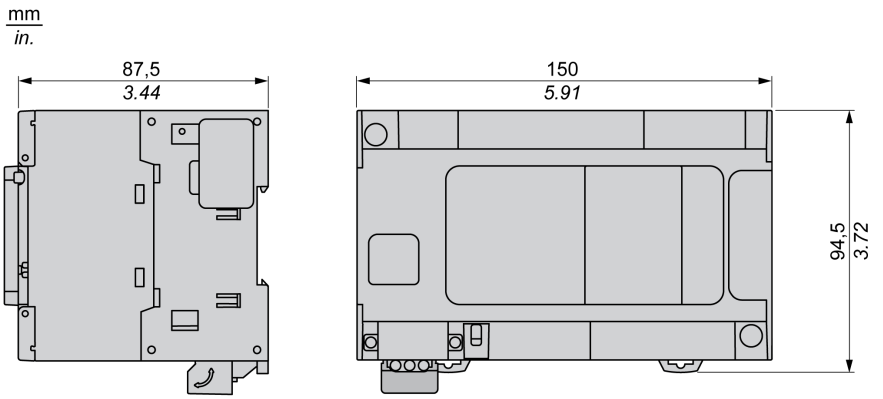
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	Indicates that an operating system error has been detected.	Restricted	No
			Fast flashing	Indicates that the controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 11
TM241CE24U

TM241CE24U Presentation

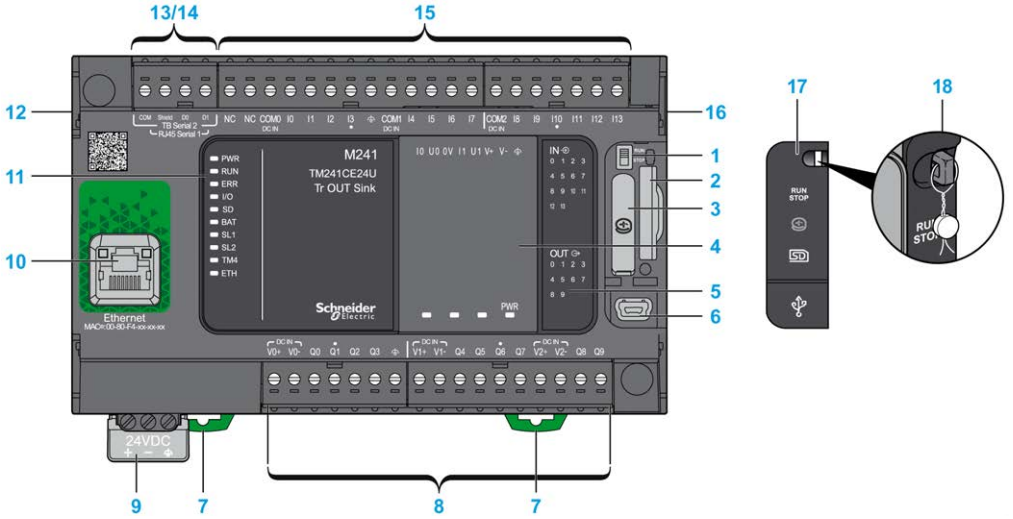
Overview

TM241CE24U logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 regular outputs
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 USB mini-B programming port

Description

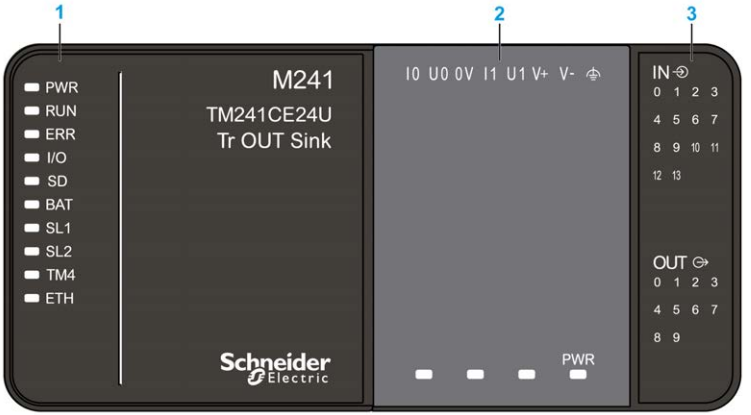
The following figure shows the different components of the TM241CE24U logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
11	Status LEDs	–
12	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
13	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
14	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
15	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
16	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
17	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
18	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

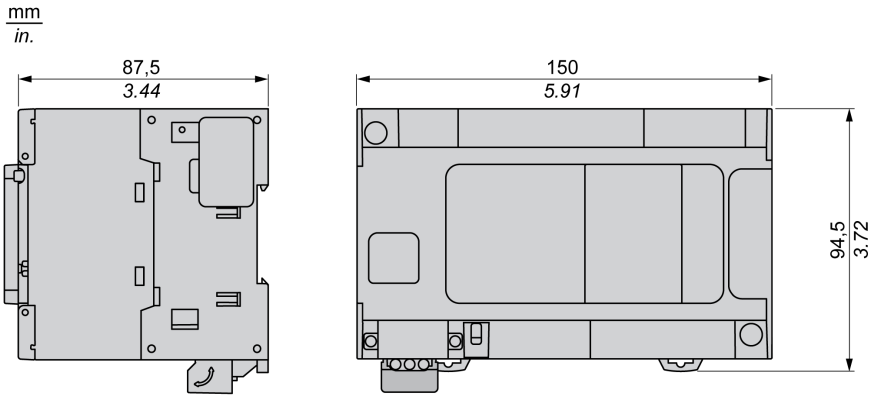
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, Ethernet port or CANopen port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
CAN R	CANopen running status	Green	On	Indicates that the CANopen bus is operational.		
			Off	Indicates that the CANopen master is configured.		
			Flashing	Indicates that the CANopen bus is being initialized.		
			1 flash per second	Indicates that the CANopen bus is stopped.		
CAN E	CANopen error	Red	On	Indicates that the CANopen bus is stopped (BUS OFF).		
			Off	Indicates no CANopen detected error.		
			Flashing	Indicates that the CANopen bus is not valid.		
			1 flash per second	Indicates that the controller has detected that the maximum number of error frames has been reached or exceeded.		
			2 flashes per second	Indicates that the controller has detected either a Node Guarding or a Heartbeat event.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 12
TM241CEC24U

TM241CEC24U Presentation

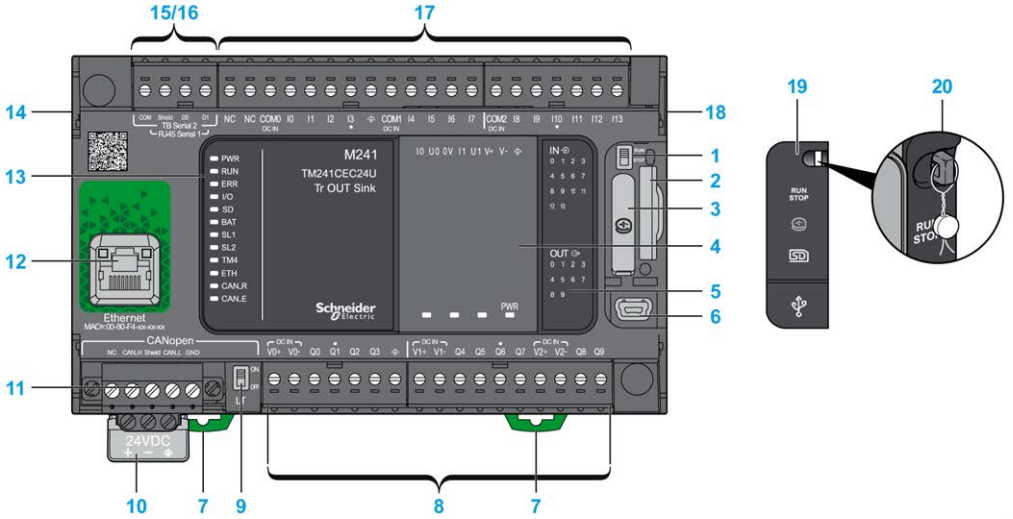
Overview

TM241CEC24U logic controller:

- 14 digital inputs
  - 8 fast inputs
  - 6 regular inputs
- 10 digital outputs
  - 4 fast outputs
  - 6 regular outputs
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 CANopen port
  - 1 USB mini-B programming port

Description

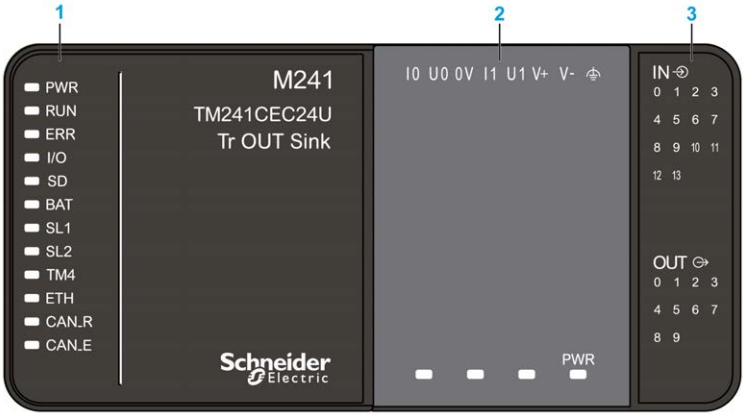
The following figure shows the different components of the TM241CEC24U logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	CANopen Line termination switch	CANopen Port *(see page 228)*
10	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
11	CANopen port / Screw terminal block type	–
12	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
13	Status LEDs	–
14	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
15	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
16	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
17	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
18	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
19	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
20	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

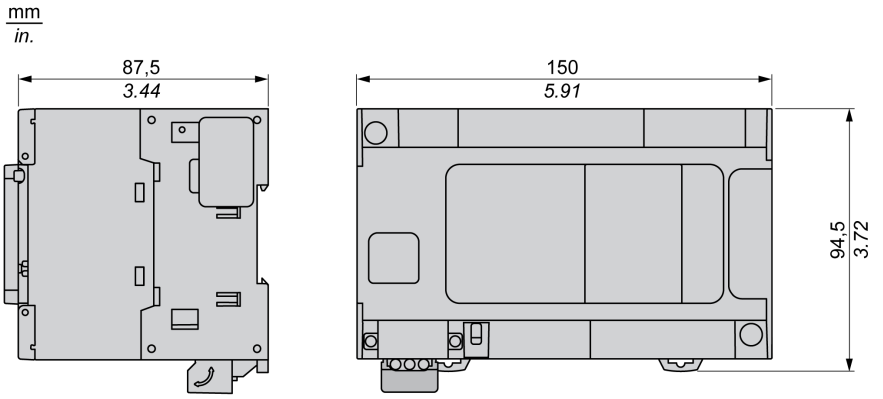
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, Ethernet port or CANopen port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
CAN R	CANopen running status	Green	On	Indicates that the CANopen bus is operational.		
			Off	Indicates that the CANopen master is configured.		
			Flashing	Indicates that the CANopen bus is being initialized.		
			1 flash per second	Indicates that the CANopen bus is stopped.		
CAN E	CANopen error	Red	On	Indicates that the CANopen bus is stopped (BUS OFF).		
			Off	Indicates no CANopen detected error.		
			Flashing	Indicates that the CANopen bus is not valid.		
			1 flash per second	Indicates that the controller has detected that the maximum number of error frames has been reached or exceeded.		
			2 flashes per second	Indicates that the controller has detected either a Node Guarding or a Heartbeat event.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 13
TM241C40R

TM241C40R Presentation

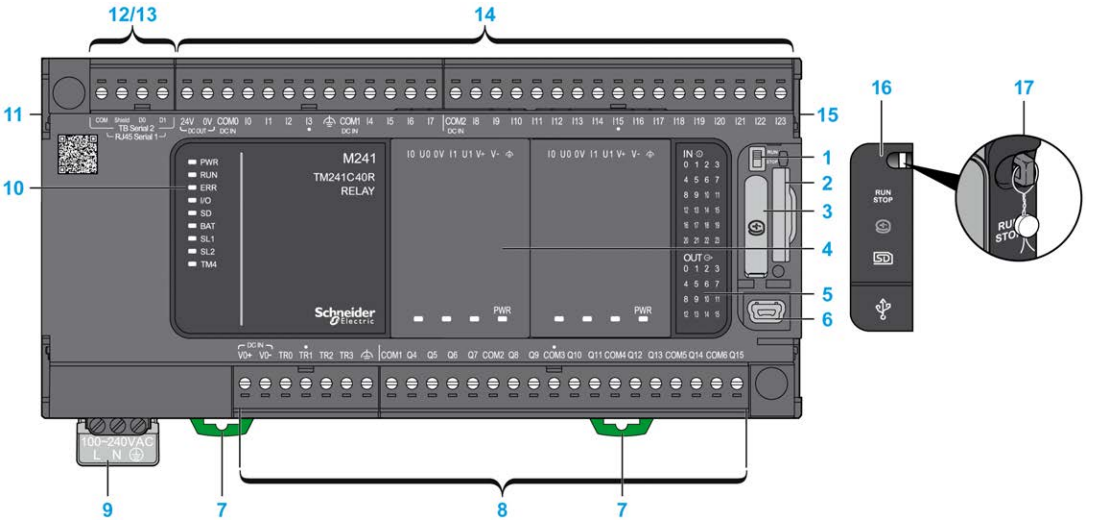
Overview

TM241C40R logic controller:

- 24 digital inputs
  - 8 fast inputs
  - 16 regular inputs
- 16 digital outputs
  - 4 fast outputs
  - 12 relay outputs (2 A)
- Communication port
  - 2 serial line ports
  - 1 USB mini-B programming port

Description

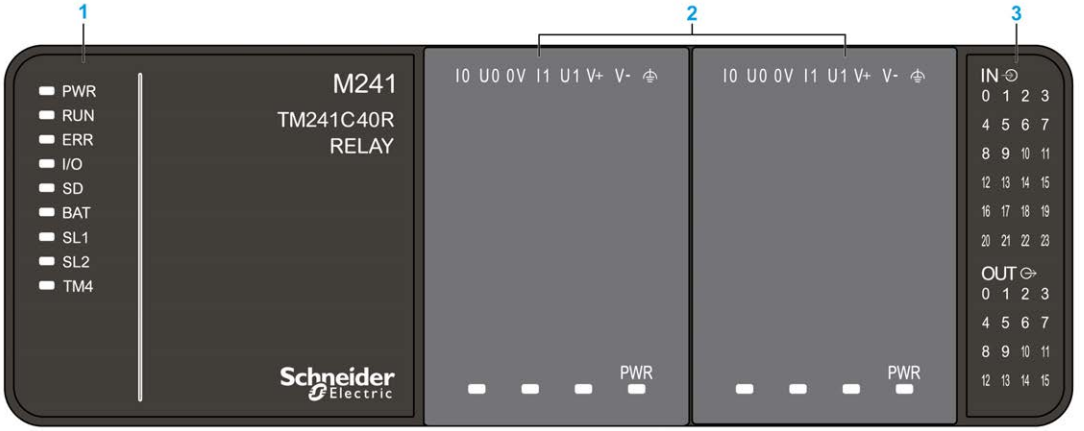
The following figure shows the different components of the TM241C40R logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Relay Outputs Status LEDs *(see page 207)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded relay outputs	Relay Outputs *(see page 206)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	100...240 Vac 50/60 Hz power supply	AC Power Supply Characteristics and Wiring *(see page 98)*
10	Status LEDs	–
11	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
12	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
13	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
14	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
15	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
16	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
17	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

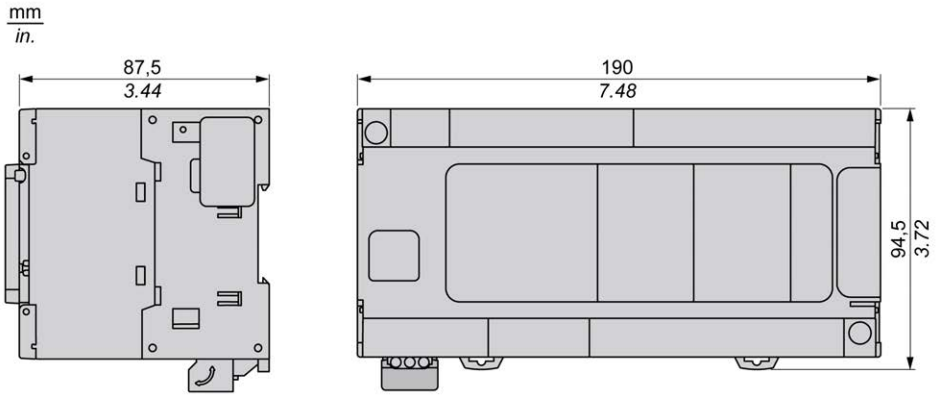
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	Indicates that an operating system error has been detected.	Restricted	No
			Fast flashing	Indicates that the controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 14
TM241CE40R

TM241CE40R Presentation

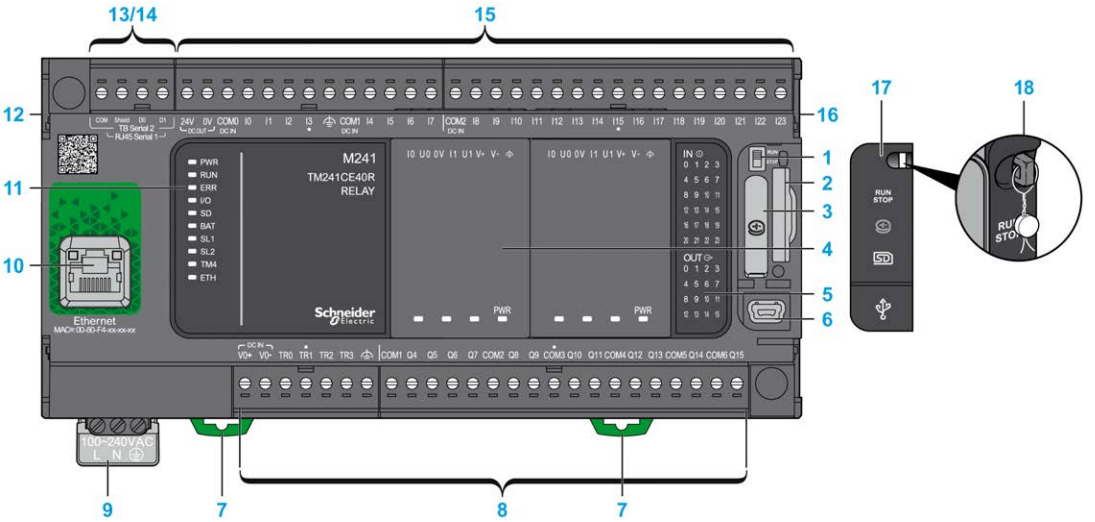
Overview

TM241CE40R logic controller:

- 24 digital inputs
  - 8 fast inputs
  - 16 regular inputs
- 16 digital outputs
  - 4 fast outputs
  - 12 relay outputs (2 A)
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 USB mini-B programming port

Description

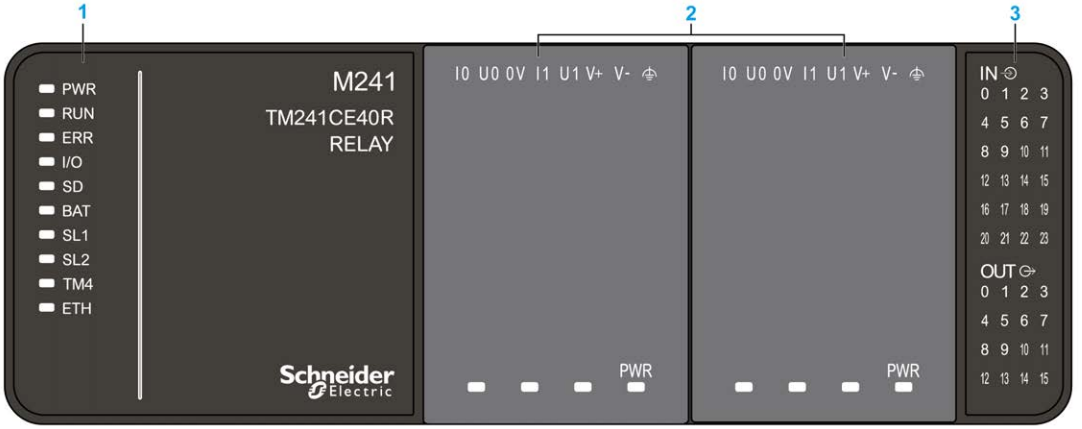
The following figure shows the different components of the TM241CE40R logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Relay Outputs Status LEDs *(see page 207)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded relay outputs	Relay Outputs *(see page 206)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	100...240 Vac 50/60 Hz power supply	AC Power Supply Characteristics and Wiring *(see page 98)*
10	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
11	Status LEDs	–
12	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
13	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
14	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
15	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
16	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
17	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
18	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

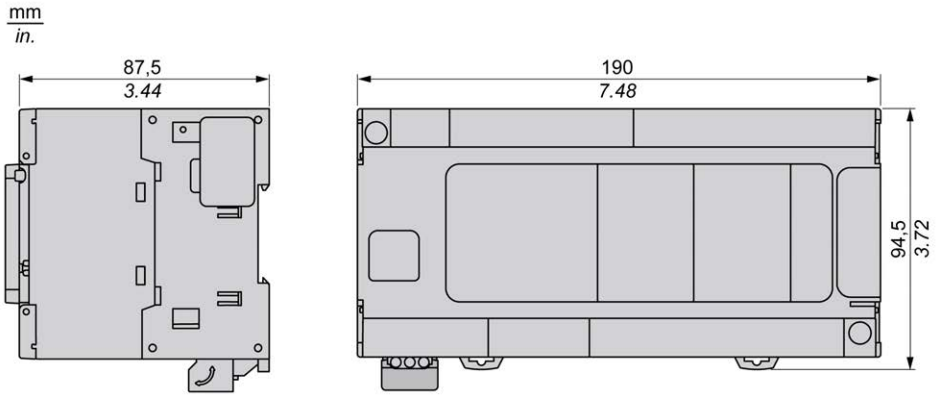
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, or Ethernet port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 15
TM241C40T

TM241C40T Presentation

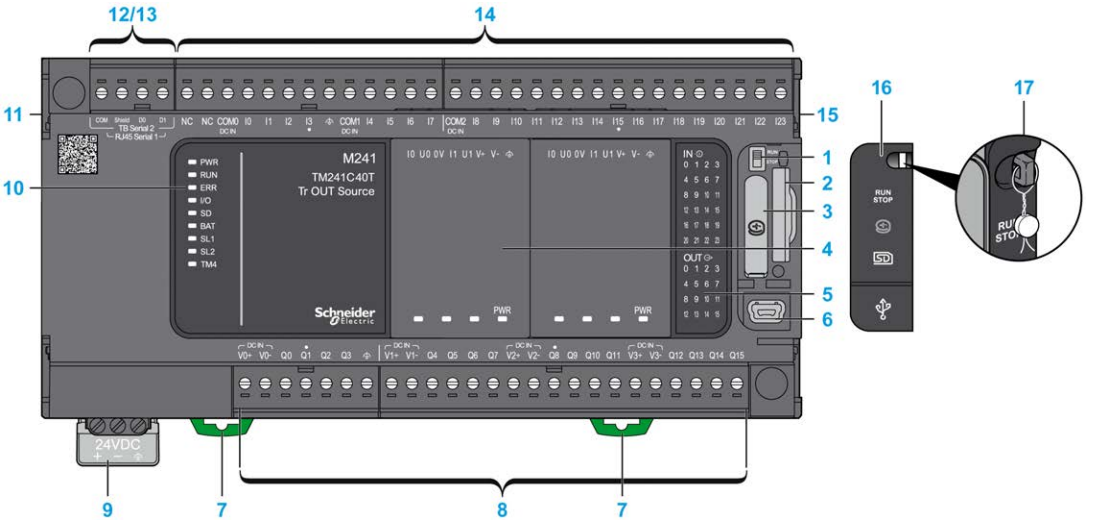
Overview

TM241C40T logic controller:

- 24 digital inputs
  - 8 fast inputs
  - 16 regular inputs
- 16 digital outputs
  - 4 fast outputs
  - 12 regular outputs
- Communication port
  - 2 serial line ports
  - 1 USB mini-B programming port

Description

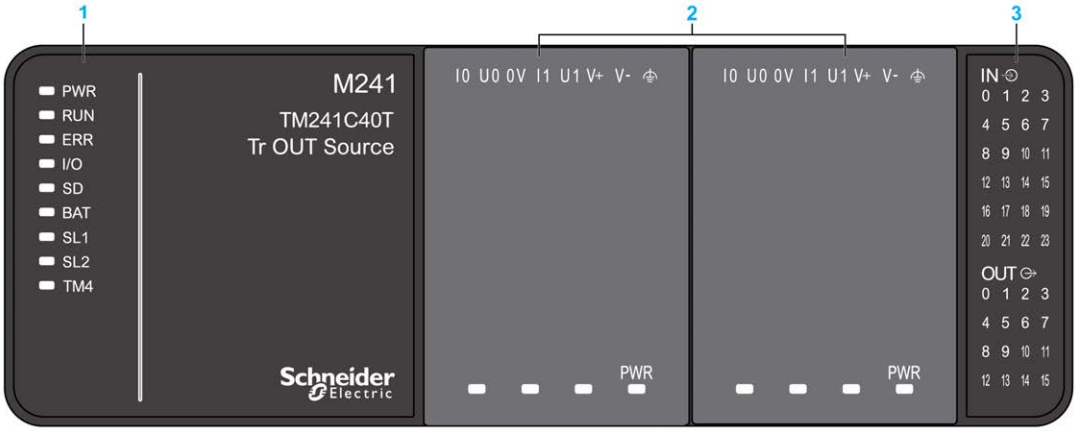
The following figure shows the different components of the TM241C40T logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Status LEDs	–
11	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
12	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
13	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
14	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
15	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
16	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
17	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

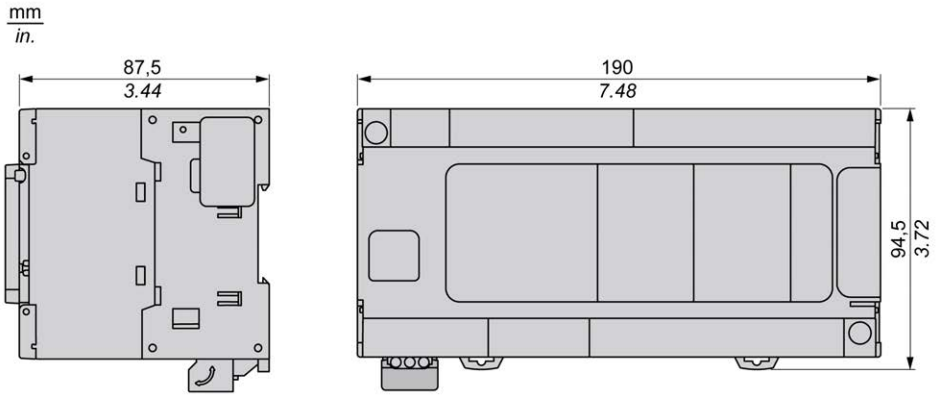
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	Indicates that an operating system error has been detected.	Restricted	No
			Fast flashing	Indicates that the controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 16
TM241CE40T

TM241CE40T Presentation

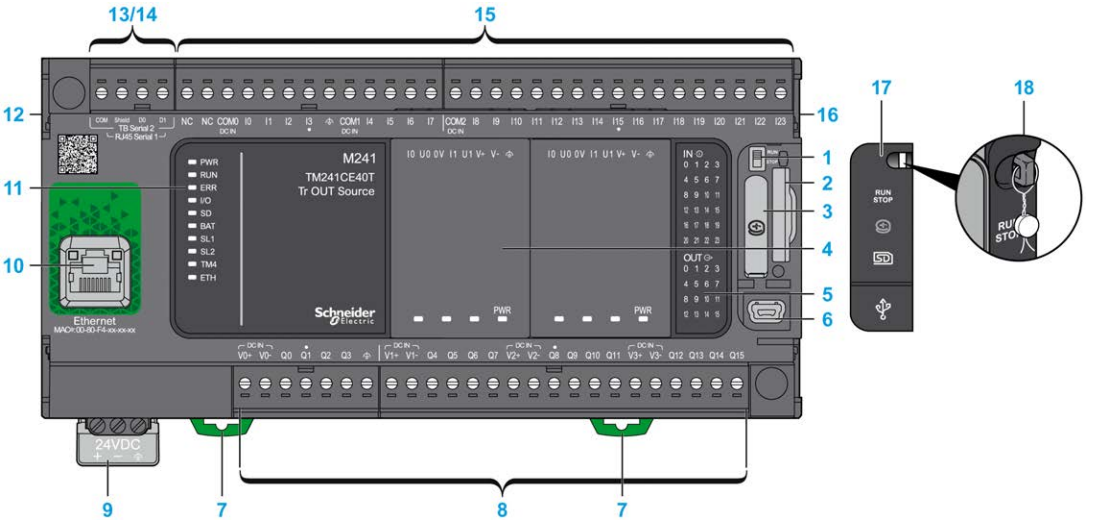
Overview

TM241CE40T logic controller:

- 24 digital inputs
  - 8 fast inputs
  - 16 regular inputs
- 16 digital outputs
  - 4 fast outputs
  - 12 regular outputs
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 USB mini-B programming port

Description

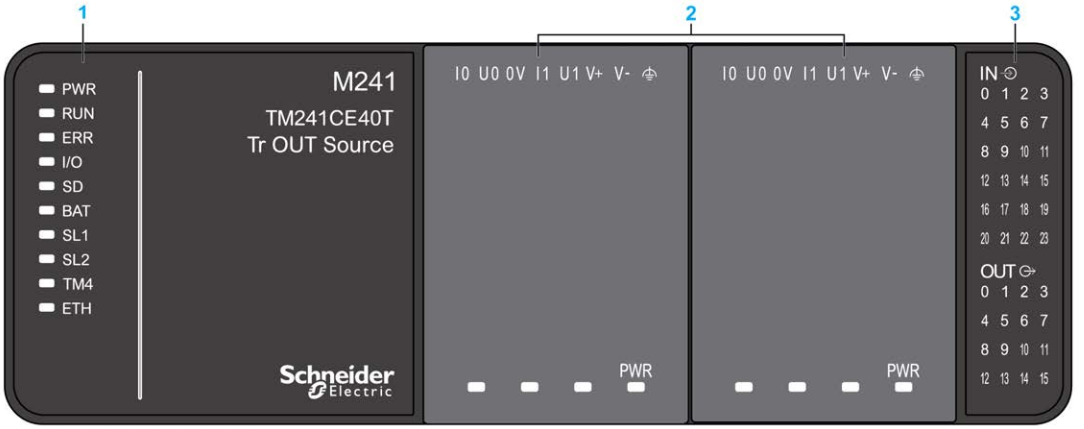
The following figure shows the different components of the TM241CE40T logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
11	Status LEDs	–
12	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
13	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
14	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
15	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
16	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
17	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
18	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

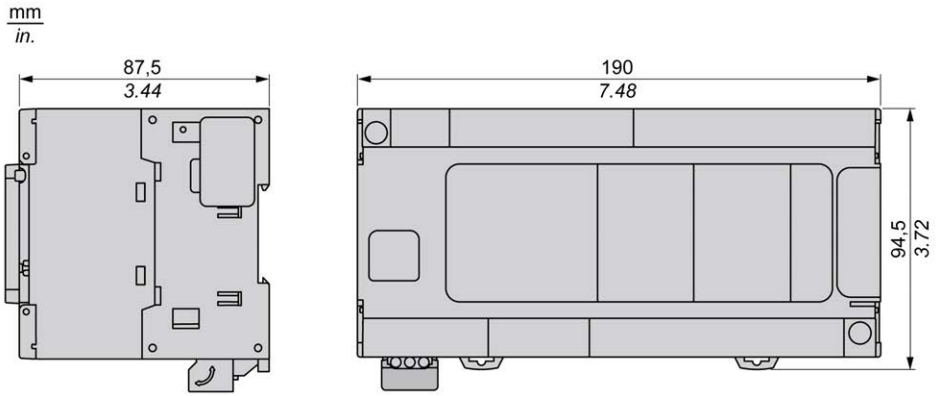
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, or Ethernet port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 17
TM241C40U

TM241C40U Presentation

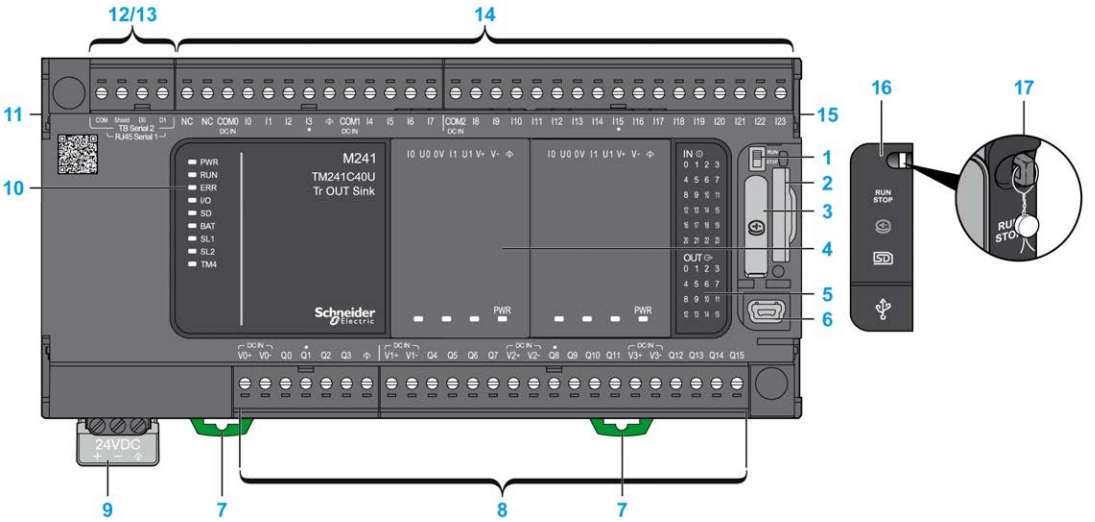
Overview

TM241C24U logic controller:

- 24 digital inputs
  - 8 fast inputs
  - 16 regular inputs
- 16 digital outputs
  - 4 fast outputs
  - 12 regular outputs
- Communication port
  - 2 serial line ports
  - 1 USB mini-B programming port

Description

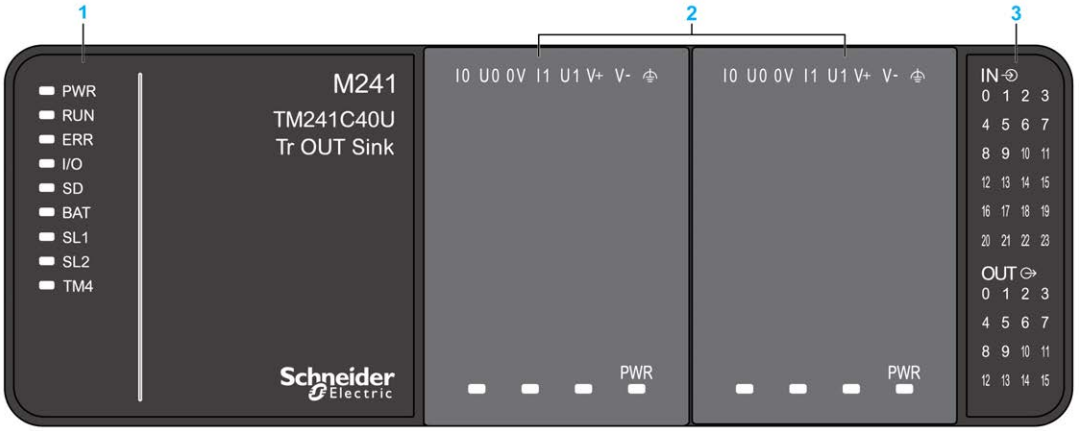
The following figure shows the different components of the TM241C40U logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Status LEDs	–
11	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
12	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
13	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
14	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
15	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
16	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
17	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

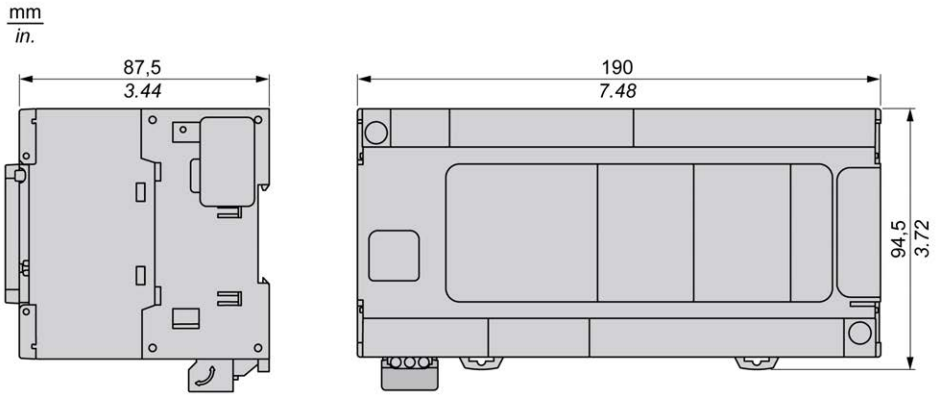
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	Indicates that an operating system error has been detected.	Restricted	No
			Fast flashing	Indicates that the controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 18
TM241CE40U

TM241CE40U Presentation

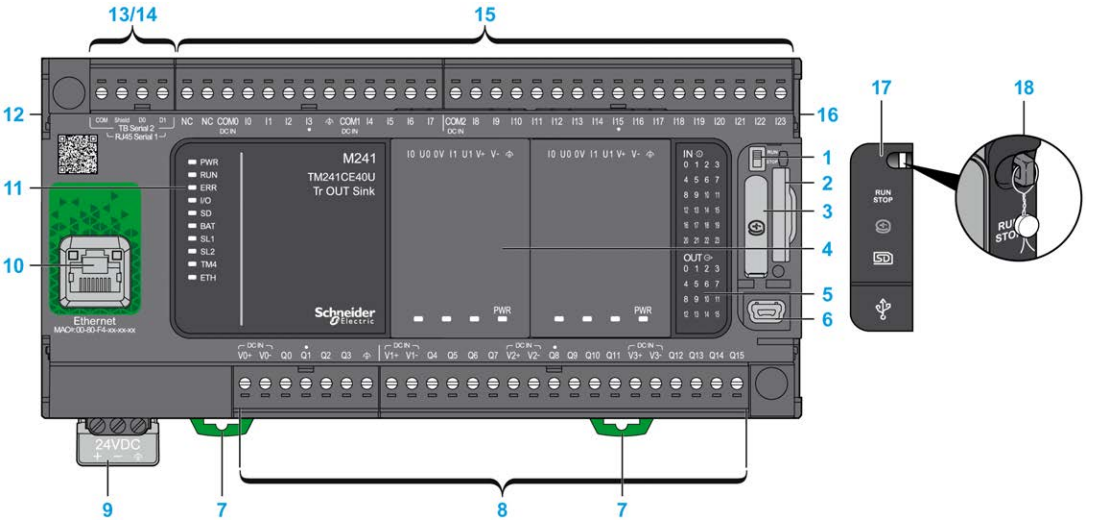
Overview

TM241CE40U logic controllers:

- 24 digital inputs
  - 8 fast inputs
  - 16 regular inputs
- 16 digital outputs
  - 4 fast outputs
  - 12 regular outputs
- Communication port
  - 2 serial line ports
  - 1 Ethernet port
  - 1 USB mini-B programming port

Description

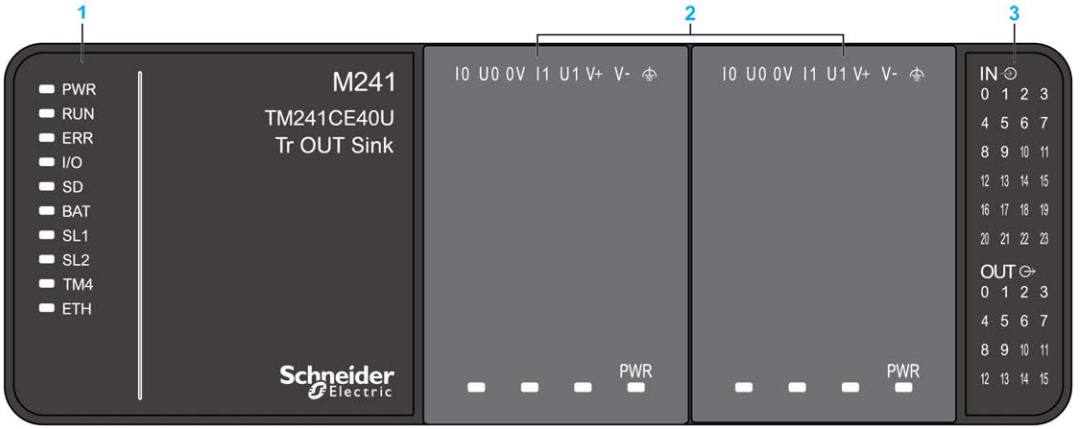
The following figure shows the different components of the TM241CE40U logic controller:



N°	Description	Refer to
1	Run/Stop switch	Run/Stop *(see page 61)*
2	SD card slot	SD Card *(see page 63)*
3	Battery holder	Real Time Clock (RTC) *(see page 48)*
4	Cartridge slot	–
5	LEDs for indicating I/O states	Digital Inputs Status LEDs *(see page 199)*
		Transistor Outputs Status LEDs *(see page 213)* Fast Outputs Status LEDs *(see page 219)*
6	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 235)*
7	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 80)*
8	Embedded regular transistor outputs	Regular Transistor Outputs *(see page 212)*
	Embedded fast transistor outputs	Fast Transistor Outputs *(see page 218)*
	Output removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
9	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 94)*
10	Ethernet port / Type RJ45 (RS-232 or RS-485)	Ethernet Port *(see page 232)*
11	Status LEDs	–
12	TM4 bus connector	TM4 Expansion Modules *(see page 41)*
13	Serial line port 1 / Type RJ45 (RS-232 or RS-485)	Serial Line 1 *(see page 237)*
14	Serial line port 2 / Screw terminal block type (RS-485)	Serial Line 2 *(see page 240)*
15	Embedded digital inputs	Embedded Digital Inputs *(see page 198)*
	Input removable terminal block	Rules for Removable Screw Terminal Block *(see page 90)*
16	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 31)*
17	Protective cover (SD card slot, Run/Stop switch, and USB mini-B programming port)	–
18	Locking hook (Hook not included)	–

Status LEDs

The following figure shows the status LEDs:



1 System status LEDs
2 Cartridge status LEDs (optional)
3 I/Os status LEDs

The following table describes the system status LEDs:

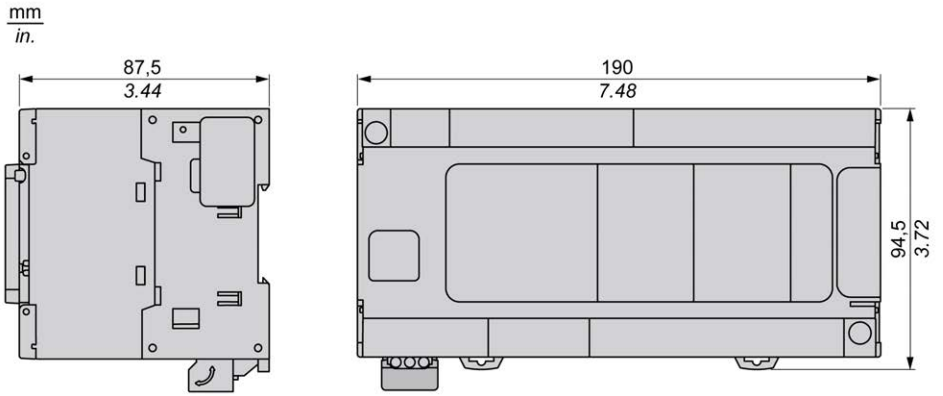
Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
PWR	Power	Green	On	Indicates that power is applied.		
			Off	Indicates that power is removed.		
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.		
			Flashing	Indicates that the controller has a valid application that is stopped.		
			1 flash	Indicates that the controller has paused at BREAKPOINT.		
			Off	Indicates that the controller is not programmed	-	-

Label	Function Type	Color	Status	Description		
				Controller States[1]	Prg Port Communication	Application Execution
ERR	Error	Red	On	An operating system error has been detected.	Restricted	No
			Fast flashing	The controller has detected an internal error.	Restricted	No
			Slow flashing	Indicates either that a minor error has been detected, if the RUN LED is illuminated, or that no application has been detected.	Yes	No
I/O	I/O error	Red	On	Indicates device errors on the embedded I/Os, serial line 1 or 2, SD card, cartridge, TM4 bus, TM3 bus, or Ethernet port.		
SD	SD card access	Green	On	Indicates that the SD card is being accessed		
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.		
			Flashing	Indicates that the battery charge is low.		
SL1	Serial line 1	Green	On	Indicates the status of serial line 1 *(see page 239)*		
			Off	Indicates no serial communication		
SL2	Serial line 2	Green	On	Indicates the status of serial line 2 *(see page 241)*		
			Off	Indicates no serial communication		
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus		
			Off	Indicates that no error has been detected on the TM4 bus		
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.		
			3 flashes	Indicates that the ethernet port is not connected.		
			4 flashes	Indicates that the IP address is already in used.		
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.		
			6 flashes	Indicates that the configured IP address is not valid.		

[1] For more information about the controller state description, refer to the M241 Logic Controller - Programming Guide.

Dimensions

The following figure shows the external dimensions of the logic controller:



Chapter 19
Embedded I/O Channels

Overview

This chapter describes the embedded I/O channels.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Digital Inputs	198
Relay Outputs	206
Regular Transistor Outputs	212
Fast Transistor Outputs	218

Digital Inputs

Overview

The Modicon M241 Logic Controller has digital inputs embedded:

Reference	Total number of digital inputs	Fast inputs which can be used as 200 kHz HSC inputs	Total number of regular inputs	Regular inputs which can be used as 1 kHz HSC inputs
TM241C••24R TM241C••24T TM241C••24U	14	8	6	6
TM241C•40R TM241C•40T TM241C•40U	24	8	16	8

For more information, refer to Input Management *(see page 52)*.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

Digital Input Status LEDs

The following figure shows the status LEDs for the TM241C••24• controller (the TM241C•40• controllers are similar with 40 LEDs):



LED	Color	Status	Description
0...13	Green	On	The input channel is activated
		Off	The input channel is deactivated

Regular Input Characteristics

The table below describes the characteristics of the M241 Logic Controller with regular inputs:

Characteristic		Values	
		TM241C••24•	TM241C•40•
Number of regular inputs		6 inputs (I8...I13)	16 inputs (I8...I23)
Number of channel groups		1 common line for I8...I13	1 common line for I8...I23
Input type		Type 1 (IEC 61131-2 Edition 3)	
Logic type		Sink/Source	
Input voltage range		24 Vdc	
Rated input voltage		0...28.8 Vdc	
Rated input current		5 mA	7 mA
Input impedance		4.7 kΩ	
Input limit values	Voltage at state 1	> 15 Vdc (15...28.8 Vdc)	
	Voltage at state 0	< 5 Vdc (0...5 Vdc)	
	Current at state 1	> 2.5 mA	
	Current at state 0	< 1.0 mA	
Derating		No derating	
[1] For more information, refer to Integrator Filter Principle *(see page 53)*			

Characteristic		Values	
		TM241C••24•	TM241C•40•
Turn on time		50 µs + filter value[1]	
Turn off time		50 µs + filter value[1]	
Isolation	Between input and internal logic	500 Vac	
	Between input terminals	Not Isolated	
Connection type		Removable screw terminal block	
Connector insertion/removal durability		Over 100 times	
Cable	Type	Unshielded	
	Length	Maximum 50 m (164 ft)	
[1] For more information, refer to Integrator Filter Principle *(see page 53)*			

Fast Input Characteristics

The table below describes the characteristics of the M241 Logic Controller with fast inputs:

Characteristic		Value
Number of fast transistor inputs		8 inputs (I0...I7)
Number of channel groups		1 common line for I0...I3 1 common line for I4...I7
Input type		Type 1 (IEC 61131-2 Edition 3)
Logic type		Sink/Source
Rated input voltage		24 Vdc
Input voltage range		0...28.8 Vdc
Rated input current		10.7 mA
Input impedance		2.81 kΩ
Input limit values	Voltage at state 1	> 15 Vdc (15...28.8 Vdc)
	Voltage at state 0	< 5 Vdc (0...5 Vdc)
	Current at state 1	> 5 mA
	Current at state 0	< 1.5 mA
Derating		No derating
Turn on time		2 µs + filter value[1]
Turn off time		2 µs + filter value[1]
HSC maximum frequency	A/B phase	100 kHz
	Pulse/Direction	200 kHz
	Single phase	200 kHz
[1] For more information, refer to Integrator Filter Principle *(see page 53)*		

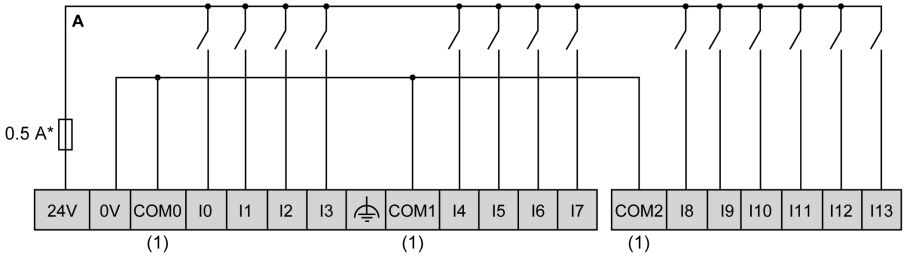
Characteristic		Value
HSC supported operation mode		• A/B phase counter • Pulse/Direction counter • Single/Dual phase counter
Isolation	Between input and internal logic	500 Vac
	Between input terminals	Not isolated
Connection type		Removable screw terminal block
Connector insertion/removal durability		Over 100 times
Cable	Type	Shielded, including the 24 Vdc power supply
	Length	Maximum 10 m (32.8 ft)
[1] For more information, refer to Integrator Filter Principle *(see page 53)*		

Removing Terminal Block

Refer to Removing Terminal Block *(see page 91)*.

TM241C••24R Wiring Diagrams

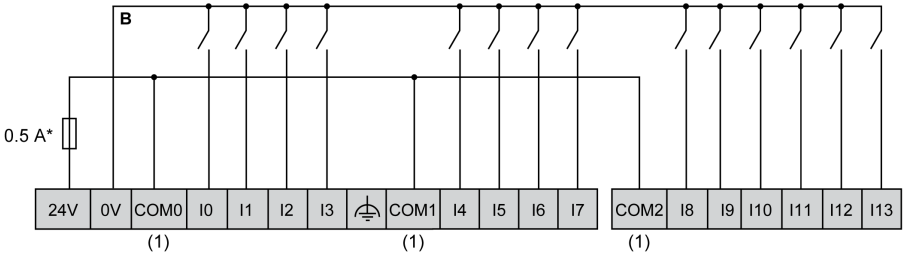
The following figure shows the sink wiring (positive logic) of the controller digital inputs:



* Type T fuse

(1) The COM0, COM1 and COM2 terminals are **not** connected internally.

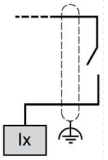
The following figure shows the source wiring (negative logic) of the controller digital inputs:



* Type T fuse

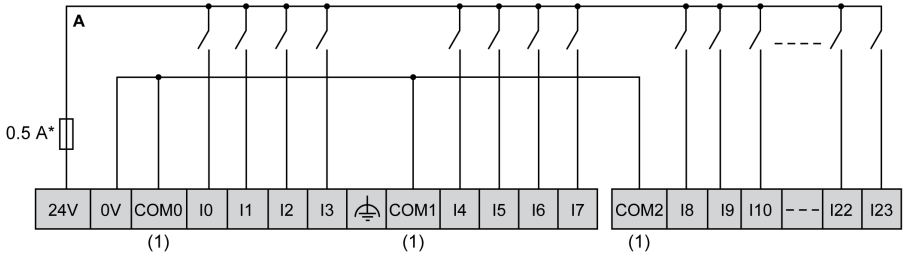
(1) The COM0, COM1 and COM2 terminals are **not** connected internally.

Fast input wiring for I0... I7:



TM241C•40R Wiring Diagrams

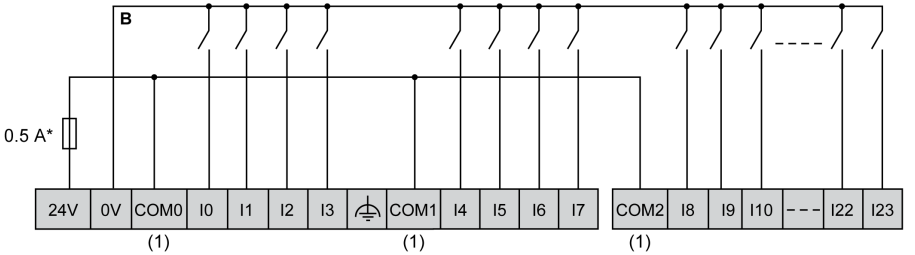
The following figure shows the sink wiring (positive logic) of the controller digital inputs:



* Type T fuse

(1) The COM0, COM1 and COM2 terminals are **not** connected internally.

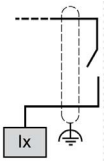
The following figure shows the source wiring (negative logic) of the controller digital inputs:



* Type T fuse

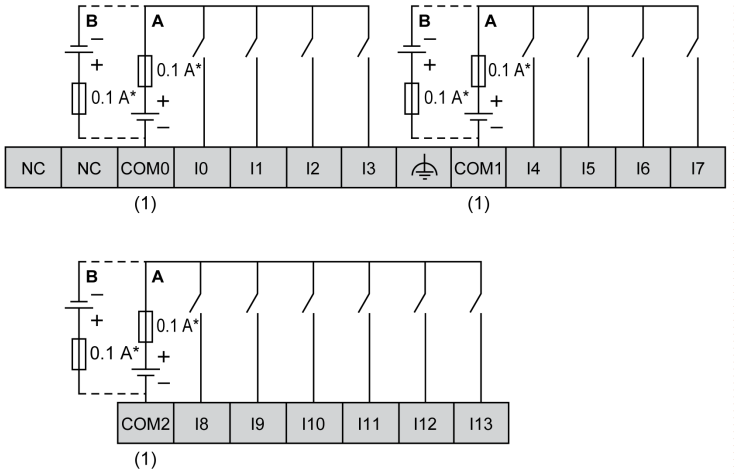
(1) The COM0, COM1 and COM2 terminals are **not** connected internally.

Fast input wiring for I0... I7:



TM241C••24T / TM241C••24U Wiring Diagrams

The following figure shows the connection of the controller digital inputs:



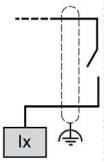
* Type T fuse

(1) The COM0, COM1 and COM2 terminals are **not** connected internally.

A Sink wiring (positive logic).

B Source wiring (negative logic).

Fast input wiring for I0... I7:



⚠ WARNING

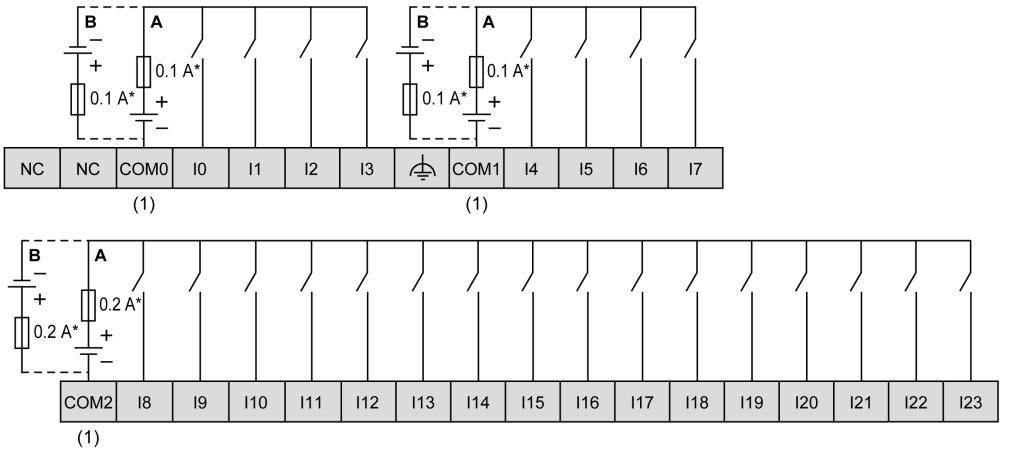
UNINTENDED EQUIPMENT OPERATION

Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”.

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

TM241C•40T / TM241C•40U Wiring Diagrams

The following figure shows the connection of the controller digital inputs:



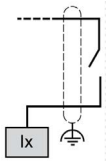
***** Type T fuse

(1) The COM0, COM1 and COM2 terminals are **not** connected internally.

A Sink wiring (positive logic).

B Source wiring (negative logic).

Fast input wiring for I0... I7:



⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”.

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

Relay Outputs

Overview

The Modicon M241 Logic Controller has digital outputs embedded:

Reference	Total number of digital outputs	Fast transistor outputs *(see page 219)* (1)	Relay outputs *(see page 207)*	Regular transistor outputs *(see page 213)*
TM241C••24R	10	4	6	0
TM241C••24T TM241C••24U	10	4	0	6
TM241C•40R	16	4	12	0
TM241C•40T TM241C•40U	16	4	0	12
(1) Fast transistor outputs which can be used as 100 kHz PTO outputs				

For more information, refer to Output Management *(see page 56)*.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

Relay Outputs Status LEDs

The following figure shows the status LEDs for the TM241C••24• controller (the TM241C•40• controllers are similar with 40 LEDs):



LED	Color	Status	Description
0...9	Green	On	The output channel is activated
		Off	The output channel is deactivated

Relay Outputs Characteristics

The following table describes the characteristics of the M241 Logic Controller relay outputs:

Characteristic	Value	
	TM241C••24R	TM241C•40R
Number of relay output channels	6 outputs (Q4...Q9)	12 outputs (Q4...Q15)
Number of channel groups	1 common line for Q4, Q5 1 common line for Q6, Q7 1 line for Q8 1 line for Q9	1 common line for Q4...Q7 1 common line for Q8, Q9 1 common line for Q10, Q11 1 common line for Q12, Q13 1 line for Q14 1 line for Q15
Output type	Relay	
Contact type	NO (Normally Open)	
Rated output voltage	24 Vdc, 240 Vac	
Maximum voltage	30 Vdc, 264 Vac	

1 Refer to Protecting Outputs from Inductive Load Damage *(see page 92)* for additional information concerning output protection.
2 When Q4, Q5, Q6 and Q7 are on the same common line (max output current 4 A), those 4 outputs used simultaneously have a derating of 50%.
3 Outputs Q4 and Q6 used as PTO/PWM/FG can result in a premature wear of those relay outputs.

Characteristic		Value	
		TM241C••24R	TM241C•40R
Minimum switching load		5 Vdc at 10 mA	
Derating		No derating	Derating on Q4...Q7, refer to the note 2.
Rated output current		2 A	
Maximum output current		2 A per output	
		4 A per common	
Maximum output frequency with maximum load		20 operations per minute	
Turn on time		Max. 10 ms	
Turn off time		Max. 10 ms	
Contact resistance		30 mΩ max	
Mechanical life		20 million operations	
Electrical life	Under resistive load	See power limitation	
	Under inductive load		
Protection against short circuit		No	
Isolation	Between output and internal logic	500 Vac	
	Between channel groups	1500 Vac	
Connection type		Removable screw terminal blocks	
Connector insertion/removal durability		Over 100 times	
Cable	Type	Unshielded	
	Length	Max. 30 m (98 ft)	

1 Refer to Protecting Outputs from Inductive Load Damage *(see page 92)* for additional information concerning output protection.
2 When Q4, Q5, Q6 and Q7 are on the same common line (max output current 4 A), those 4 outputs used simultaneously have a derating of 50%.
3 Outputs Q4 and Q6 used as PTO/PWM/FG can result in a premature wear of those relay outputs.

Power Limitation

The following table describes the power limitation of the relay outputs depending on the voltage, the type of load, and the number of operations required.

These controllers do not support capacitive loads.

⚠ WARNING

RELAY OUTPUTS WELDED CLOSED

- Always protect relay outputs from inductive alternating current load damage using an appropriate external protective circuit or device.
- Do not connect relay outputs to capacitive loads.

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

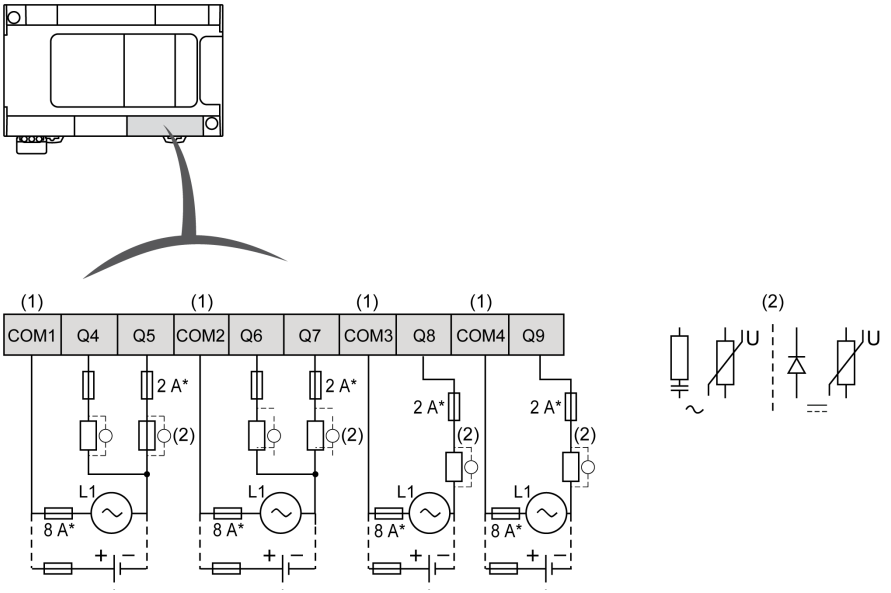
Power Limitations				
Voltage	**24 Vdc**	**120 Vac**	**240 Vac**	**Number of operations**
Power of resistive loads AC-12	–	240 VA 80 VA	480 VA 160 VA	100,000 300,000
Power of inductive loads AC-15 (cos ϕ = 0.35)	–	60 VA 18 VA	120 VA 36 VA	100,000 300,000
Power of inductive loads AC-14 (cos ϕ = 0.7)	–	120 VA 36 VA	240 VA 72 VA	100,000 300,000
Power of resistive loads DC-12	48 W 16 W	–	–	100,000 300,000
Power of inductive loads DC-13 L/R = 7 ms	24 W 7.2 W	–	–	100,000 300,000

Removing Terminal Block

Refer to Removing Terminal Block *(see page 91)*.

TM241C••24R Relay Outputs Wiring Diagrams

The following figure shows the wiring of the outputs:



* Type T fuse

(1) The terminals COM1 to COM4 are **not** connected internally.

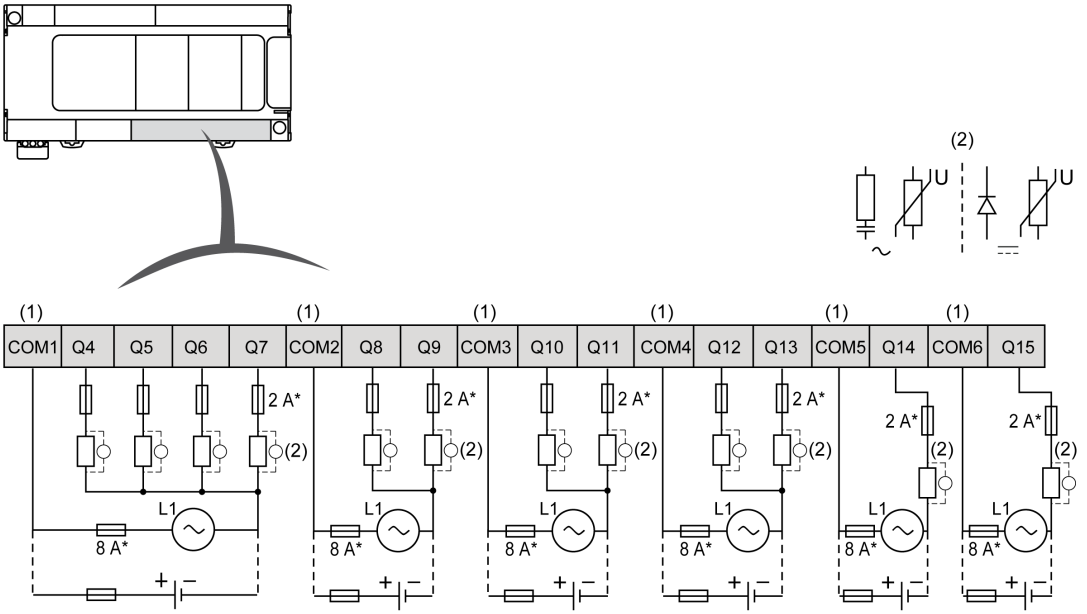
(2) To improve the life time of the contacts, and to protect from potential inductive load damage, you must connect a free wheeling diode in parallel to each inductive DC load or an RC snubber in parallel of each inductive AC load

Refer to Protecting Outputs from Inductive Load Damage *(see page 92)* for additional information concerning output protection.

NOTE: The assigned fuse values have been specified for the maximum current characteristics of the controller I/O and associated commons. You may have other considerations that are applicable based on the unique types of input and output devices you connect, and you should size your fuses accordingly.

TM241C•40R Relay Outputs Wiring Diagrams

The following figure shows the wiring of the outputs:



* Type T fuse

(1) The terminals COM1 to COM6 are **not** connected internally.

(2) To improve the life time of the contacts, and to protect from potential inductive load damage, you must connect a free wheeling diode in parallel to each inductive DC load or an RC snubber in parallel of each inductive AC load

Refer to Protecting Outputs from Inductive Load Damage *(see page 92)* for additional information concerning output protection.

NOTE: The assigned fuse values have been specified for the maximum current characteristics of the controller I/O and associated commons. You may have other considerations that are applicable based on the unique types of input and output devices you connect, and you should size your fuses accordingly.

Regular Transistor Outputs

Overview

The Modicon M241 Logic Controller has digital outputs embedded:

Reference	Total number of digital outputs	Fast transistor outputs *(see page 219)* (1)	Relay outputs *(see page 207)*	Regular transistor outputs *(see page 213)*
TM241C••24R	10	4	6	0
TM241C••24T TM241C••24U	10	4	0	6
TM241C•40R	16	4	12	0
TM241C•40T TM241C•40U	16	4	0	12
(1) Fast transistor outputs which can be used as 100 kHz PTO outputs				

For more information, refer to Output Management *(see page 56)*.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

Regular Transistor Outputs Status LEDs

The following figure shows the status LEDs for the TM241C••24• controller (the TM241C•40• controllers are similar with 40 LEDs):



LED	Color	Status	Description
0...9	Green	On	The output channel is activated
		Off	The output channel is deactivated

Regular Transistor Outputs Characteristics

The following table describes the characteristics of the M241 Logic Controller regular transistor outputs:

Characteristic	TM241C••24T	TM241C••24U	TM241C•40T	TM241C•40U
Number of regular transistor outputs	6 outputs (Q4...Q9)		12 outputs (Q4...Q15)	
Number of channel groups	1 common line for Q4...Q7 common line for Q8, Q9		1 common line for Q4...Q7 common line for Q8...Q11 common line for Q12...Q15	
Output type	Transistor			
Logic type	Source	Sink	Source	Sink
Rated output voltage	24 Vdc			
Output voltage range	19.2...28.8 Vdc			
Rated output current	0.5 A			
Total output current per group	0.5A x number of outputs of the group			
Voltage drop	1 Vdc max			
Leakage current when switched off	< 5 µA			
Maximum power of filament lamp	2.4 W max			
Derating	No derating			

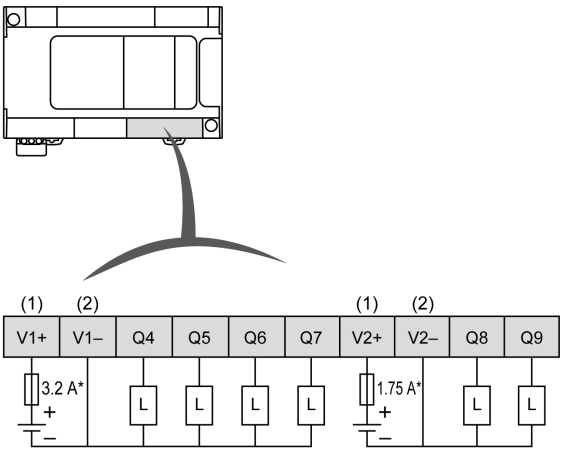
Characteristic		TM241C••24T	TM241C••24U	TM241C•40T	TM241C•40U
Turn on time		Max. 34 µs			
Turn off time		Max. 250 µs			
Protection against short circuit		Yes			
Short circuit output peak current		1.3 A			
Automatic rearming after short circuit or overload		Yes, every 10 ms			
Clamping voltage		Max. 39 Vdc +/- 1 Vdc			
Maximum output frequency		1 kHz			
Isolation	Between output and internal logic	500 Vac			
	Between output terminals	Not isolated			
Connection type		Removable screw terminal block			
Connector insertion/removal durability		Over 100 times			
Cable	Type	Unshielded			
	Length	Max 50 m (164 ft)			

Removing Terminal Block

Refer to Removing Terminal Block *(see page 91)*.

TM241C••24T Regular Transistor Outputs Source Wiring Diagram

The following figure shows the source wiring (positive logic) of the outputs:



* Type T fuse

(1) The V1+ and V2+ terminals are **not** connected internally.

(2) The V1– and V2– terminals are **not** connected internally.

⚠ WARNING

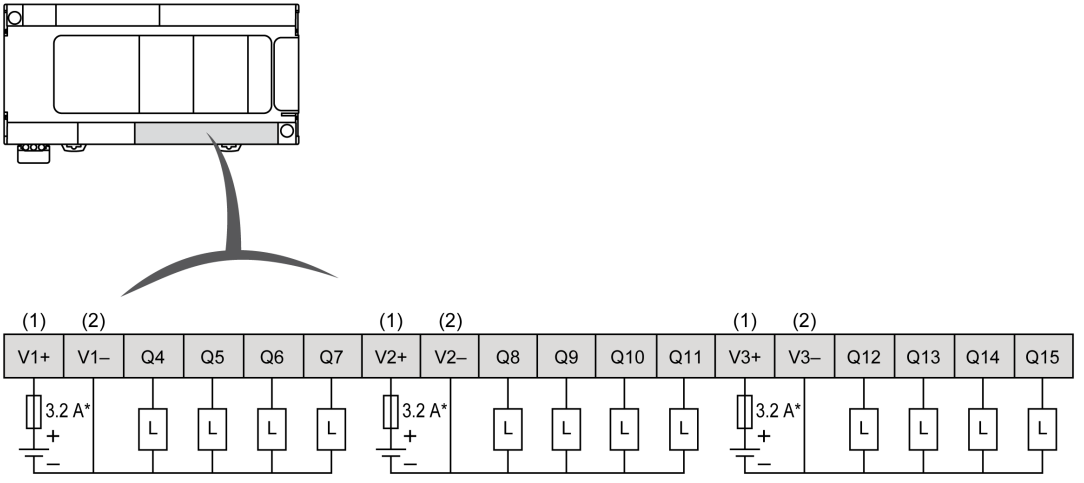
UNINTENDED EQUIPMENT OPERATION

Ensure that the physical wiring respects the connections indicated in the wiring diagram, and, in particular, that both V•+ and V•- are connected, and that only 24Vdc is connected to the V•+ terminal(s) and only 0Vdc is connected to the V•- terminal(s).

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

TM241C•40T Regular Transistor Outputs Source Wiring Diagram

The following figure shows the source wiring (positive logic) of the outputs:



***** Type T fuse

(1) The V1+, V2+ and V3+ terminals are **not** connected internally.

(2) The V1–, V2– and V3– terminals are **not** connected internally.

⚠ WARNING

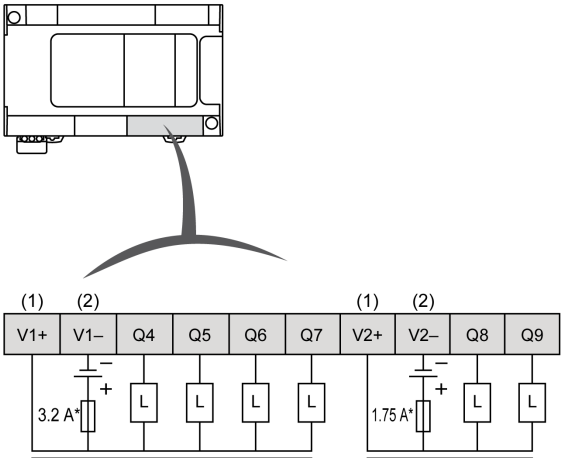
UNINTENDED EQUIPMENT OPERATION

Ensure that the physical wiring respects the connections indicated in the wiring diagram, and, in particular, that both V•+ and V•- are connected, and that only 24Vdc is connected to the V•+ terminal(s) and only 0Vdc is connected to the V•- terminal(s).

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

TM241C••24U Regular Transistor Outputs Sink Wiring Diagrams

The following figure shows the sink wiring (negative logic) of the outputs:



* Type T fuse

(1) The V1+ and V2+ terminals are **not** connected internally.

(2) The V1– and V2– terminals are **not** connected internally.

⚠ WARNING

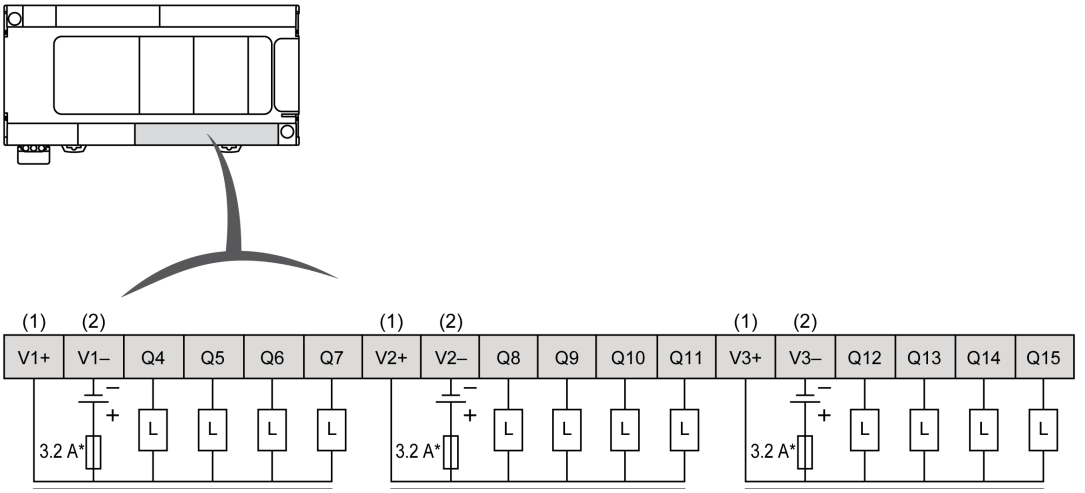
UNINTENDED EQUIPMENT OPERATION

Ensure that the physical wiring respects the connections indicated in the wiring diagram, and, in particular, that both V•+ and V•- are connected, and that only 24Vdc is connected to the V•+ terminal(s) and only 0Vdc is connected to the V•- terminal(s).

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

TM241C•40U Regular Transistor Outputs Sink Wiring Diagrams

The following figure shows the sink wiring (negative logic) of the outputs:



* Type T fuse

(1) The V1+, V2+ and V3+ terminals are **not** connected internally.

(2) The V1–, V2– and V3– terminals are **not** connected internally.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that the physical wiring respects the connections indicated in the wiring diagram, and, in particular, that both V•+ and V•- are connected, and that only 24Vdc is connected to the V•+ terminal(s) and only 0Vdc is connected to the V•- terminal(s).

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

Fast Transistor Outputs

Overview

The Modicon M241 Logic Controller has digital outputs embedded:

Reference	Total number of digital outputs	Fast transistor outputs *(see page 219)* (1)	Relay outputs *(see page 207)*	Regular transistor outputs *(see page 213)*
TM241C••24R	10	4	6	0
TM241C••24T TM241C••24U	10	4	0	6
TM241C•40R	16	4	12	0
TM241C•40T TM241C•40U	16	4	0	12
(1) Fast transistor outputs which can be used as 100 kHz PTO outputs				

For more information, refer to Output Management *(see page 56)*.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

Fast Transistor Outputs Status LEDs

The following figure shows the status LEDs for the TM241C••24• controller (the TM241C•40• controllers are similar with 40 LEDs):



LED	Color	Status	Description
0...9	Green	On	The output channel is activated
		Off	The output channel is deactivated

Fast Transistor Outputs Characteristics

The following table describes the characteristics of the M241 Logic Controller fast transistor outputs:

Characteristic		Value		
		TM241C•••R	TM241C•••T	TM241C•••U
Number of fast transistor outputs		4 outputs (TR0...TR3)	4 outputs (Q0...Q3)	
Number of channel groups		1 common line for TR0...TR3	1 common line for Q0...Q3	
Output type		Transistor		
Logic type		Source	Source	Sink
Rated output voltage		24 Vdc		
Output voltage range		19.2...28.8 Vdc		
Rated output current		0.1 A when configured for a fast function		
		0.5 A when used as a regular output		
Leakage current	Source	≤ 0.3 mA		
	Sink	≤ 2 mA		
Total output current per group		2 A		

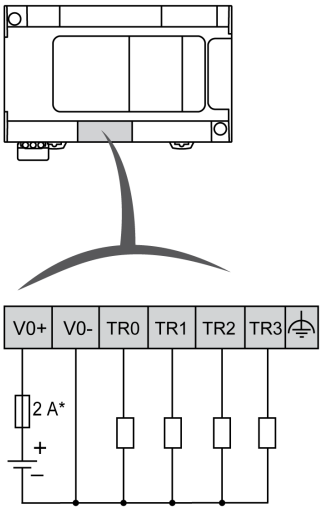
Characteristic		Value		
		TM241C•••R	TM241C•••T	TM241C•••U
Maximum power of filament lamp		2.4 W max		
Derating		No Derating		
Turn on time		Max. 2 µs		
Turn off time		Max. 2 µs		
Protection against short circuit		Yes		
Short circuit output peak current		1.3 A max.		
Automatic rearming after short circuit or overload		Yes, 12 s		
Protection against reverse polarity		Yes		
Clamping voltage		Typically 39 Vdc +/- 1 Vdc		
Maximum output frequency	PTO	100 kHz		
	PWM	20 kHz		
PWM mode duty rate step		0.1% at 20...1 kHz		
Duty rate range		1...99 %		
Isolation	Between output and internal logic	500 Vac		
	Between channel groups	500 Vac		
Connection type		Removable screw terminal block		
Connector insertion/removal durability		Over 100 times		
Cable	Type	Shielded, including 24 Vdc power supply		
	Length	Maximum 3 m (9.84 ft)		

Removing Terminal Block

Refer to Removing Terminal Block *(see page 91)*.

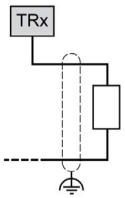
TM241C••24R / TM241C•40R Fast Transistor Outputs Wiring Diagrams

The following figure shows the connection of the fast transistor outputs:



* 2 A fast-blow fuse

Fast output wiring for TR0... TR3:



⚠ WARNING

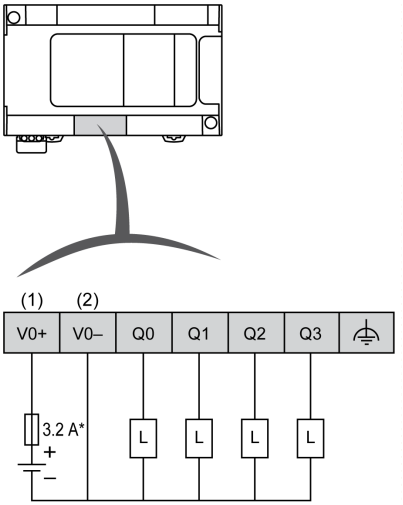
UNINTENDED EQUIPMENT OPERATION

Ensure that the physical wiring respects the connections indicated in the wiring diagram, and, in particular, that both V•+ and V•- are connected, and that only 24Vdc is connected to the V•+ terminal(s) and only 0Vdc is connected to the V•- terminal(s).

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

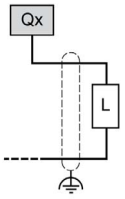
TM241C••••T Fast Transistor Outputs Wiring Diagrams

The following figure shows the connection of the fast transistor outputs:



* Type T fuse
(1) The V0+, V1+, V2+ and V3+ terminals are **not** connected internally.
(2) The V0-, V1-, V2- and V3- terminals are **not** connected internally.

Fast output wiring for Q0... Q3:



⚠ WARNING

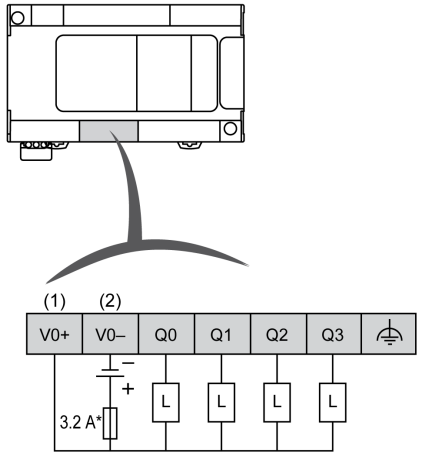
UNINTENDED EQUIPMENT OPERATION

Ensure that the physical wiring respects the connections indicated in the wiring diagram, and, in particular, that both V•+ and V•- are connected, and that only 24Vdc is connected to the V•+ terminal(s) and only 0Vdc is connected to the V•- terminal(s).

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

TM241C••••U Fast Transistor Outputs Wiring Diagrams

The following figure shows the connection of the fast transistor outputs:

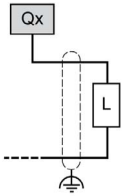


* Type T fuse

(1) The V0+, V1+, V2+ and V3+ terminals are **not** connected internally.

(2) The V0-, V1-, V2- and V3- terminals are **not** connected internally.

Fast output wiring for Q0... Q3:



⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that the physical wiring respects the connections indicated in the wiring diagram, and, in particular, that both V•+ and V•- are connected, and that only 24Vdc is connected to the V•+ terminal(s) and only 0Vdc is connected to the V•- terminal(s).

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

Part III
Modicon M241 Logic Controller Communication

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
20	Integrated Communication Ports	227
21	Connecting the M241 Logic Controller to a PC	243

Chapter 20
Integrated Communication Ports

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
CAN Port	228
Ethernet Port	232
USB Mini-B Programming Port	235
Serial Line 1	237
Serial Line 2	240

CAN Port

CANopen Capabilities

The Modicon M241 Logic Controller CANopen master has the following features:

Feature	Description
Maximum number of slaves on the bus	63 CANopen slave devices
Maximum length of CANopen fieldbus cables	According to the CAN specification (see Transmission Speed and Cable Length *(see page 231)*).
Maximum number of PDOs managed by the master	252 TPDOs + 252 RPDOs

For each additional CANopen slave:

- the application size increases by an average of 10 kbytes, which conceivably could result in exceeding memory limits.
- the configuration initialization time at the startup increases, which conceivably could result in watchdog timeout.

Although EcoStruxure Machine Expert does not restrict you from doing so, do not exceed more than 63 CANopen slave modules (and/or 252 TPDOs and 252 RPDOs) in order to have a sufficient performance tolerance and avoid any performance degradation.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not connect more than 63 CANopen slave devices to the controller to avoid system overload watchdog condition.

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

NOTICE

DEGRADATION OF PERFORMANCE

Do not exceed more than 252 TPDOs and 252 RPDOs for the Modicon M241 Logic Controller.

Failure to follow these instructions can result in equipment damage.

J1939 Capabilities

The Modicon M241 Logic Controller J1939 master has the following features:

Feature	Description
Maximum number of ECUs (slaves) on the bus	Limited only by the address range of 0...253 for Electronic Control Units (ECUs).
Maximum length of J1939 fieldbus cables	According to the CAN specification (see Transmission Speed and Cable Length *(see page 231)*). For J1939, the CAN bus must be configured to run at 250 Kbps.
Maximum number of PGNs managed by the master	Given implicitly by the maximum number of input bits (%I) and output bits (%Q) available on the Modicon M241 Logic Controller: 4096 input bits and 4096 output bits. This results in a maximum of 512 single-packet PGNs (most PGNs are single-packet, containing 8 bytes of data).

For each additional ECU with approximately 10 configured (single frame) Parameter Group Numbers (PGNs):

- the application size increases by an average of 15 Kbytes. This figure includes the memory consumed by implicitly-generated variables for configured Suspected Parameter Numbers (SPNs). This application size increase could result in exceeding memory limits.
- the number of input bits (%I) used on the logic controller increases in proportion to the number and size of PGNs configured as "TX Signals" in a non-local ECU or "RX Signals" in a local ECU.
- the number of output bits (%Q) used on the logic controller increases in proportion to the number and size of PGNs configured as "TX Signals" in a local ECU.

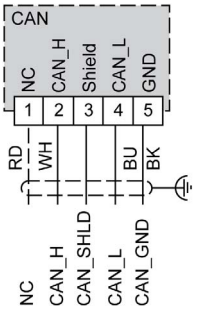
NOTE: Thoroughly test your application regarding the number of configured J1939 ECUs connected to the controller, and the number of PGNs configured on each ECU, to avoid a system overload watchdog condition or performance degradation.

For more information, refer to J1939 Interface Configuration *(see Modicon M241 Logic Controller, Programming Guide)*.

Removing Terminal Block

Refer to Removing Terminal Block *(see page 91)*.

CAN Wiring Diagram



Pin	Signal	Description	Marking	Color of Cable
1	Not used	Reserved	NC	RD: red
2	CAN_H	CAN_L bus line (dominant low)	CAN_H	WH: white
3	CAN_SHLD	Optional CAN shield	Shield	-
4	CAN_L	CAN_L bus line (dominant low)	CAN_L	BU: blue
5	CAN_GND	CAN Ground	GND	BK: black

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”.

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

Transmission Speed and Cable Length

Transmission speed is limited by the bus length and the type of cable used.

The following table describes the relationship between the maximum transmission speed and the bus length (on a single CAN segment without a repeater):

Maximum transmission baud rate	Bus length
1000 Kbps	20 m (65 ft)
800 Kbps	40 m (131 ft)
500 Kbps	100 m (328 ft)
250 Kbps	250 m (820 ft)
125 Kbps	500 m (1,640 ft)
50 Kbps	1000 m (3280 ft)
20 Kbps	2500 m (16,400 ft)

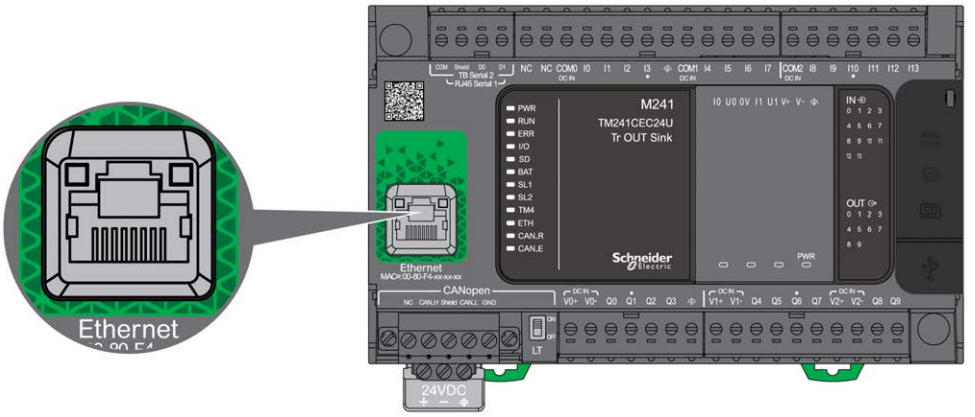
NOTE: The CAN cable must be shielded.

Ethernet Port

Overview

The TM241CE••• are equipped with an Ethernet communications port.

The following figure shows the location of the Ethernet port on the controller:



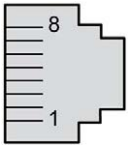
Characteristics

The following table describes the Ethernet characteristics:

Characteristic	Description
Function	Modbus TCP/IP
Connector type	RJ45
Auto negotiation	from 10 M half duplex to 100 M full duplex
Cable type	Shielded
Automatic cross-over detection	Yes

Pin Assignment

The following figure shows the RJ45 Ethernet connector pin assignment:



The following table describes the RJ45 Ethernet connector pins:

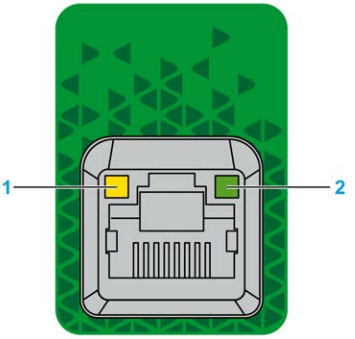
Pin N°	Signal
1	TD+
2	TD-
3	RD+
4	-
5	-
6	RD-
7	-
8	-

NOTE: The controller supports the MDI/MDIX auto-crossover cable function. It is not necessary to use special Ethernet crossover cables to connect devices directly to this port (connections without an Ethernet hub or switch).

NOTE: Ethernet cable disconnection is detected every second. In case of disconnection of a short duration (< 1 second), the network status may not indicate the disconnection.

Status LED

The following figure shows the RJ45 connector status LED:



The following table describes the Ethernet status LED:

Label	Description	LED		
		Color	Status	Description
1	Ethernet Link	Green/Yellow	Off	No link
			Solid yellow	Link at 10 Mbit/s
			Solid green	Activity at 100 Mbit/s
2	Ethernet Activity	Green	Off	No activity
			On	Transmitting or receiving data

USB Mini-B Programming Port

Overview

The USB Mini-B Port is the programming port you can use to connect a PC with a USB host port using EcoStruxure Machine Expert software. Using a typical USB cable, this connection is suitable for quick updates of the program or short duration connections to perform maintenance and inspect data values. It is not suitable for long-term connections such as commissioning or monitoring without the use of specially adapted cables to help minimize electromagnetic interference.

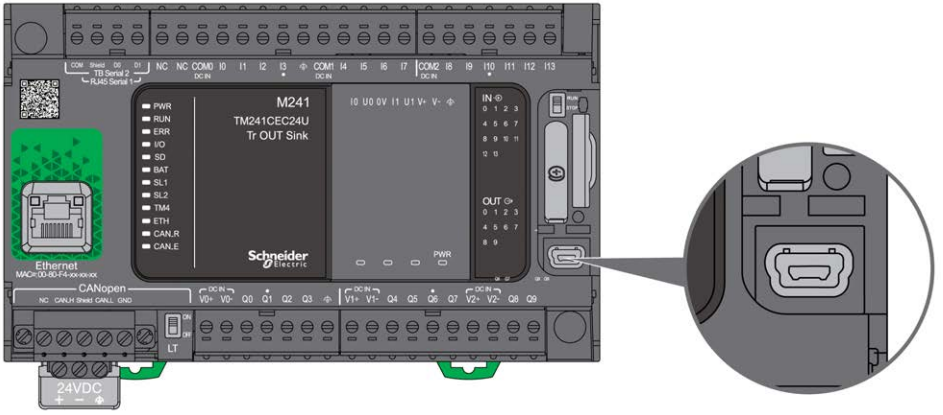
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- You must use a shielded USB cable such as a BMX XCAUSBH0•• secured to the functional ground (FE) of the system for any long-term connection.
- Do not connect more than one controller or bus coupler at a time using USB connections.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

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

The following figure shows the location of the USB Mini-B programming port:



Characteristics

This table describes the characteristics of the USB Mini-B programming port:

Parameter	USB Programming Port
Function	Compatible with USB 2.0
Connector type	Mini-B
Isolation	None
Cable type	Shielded

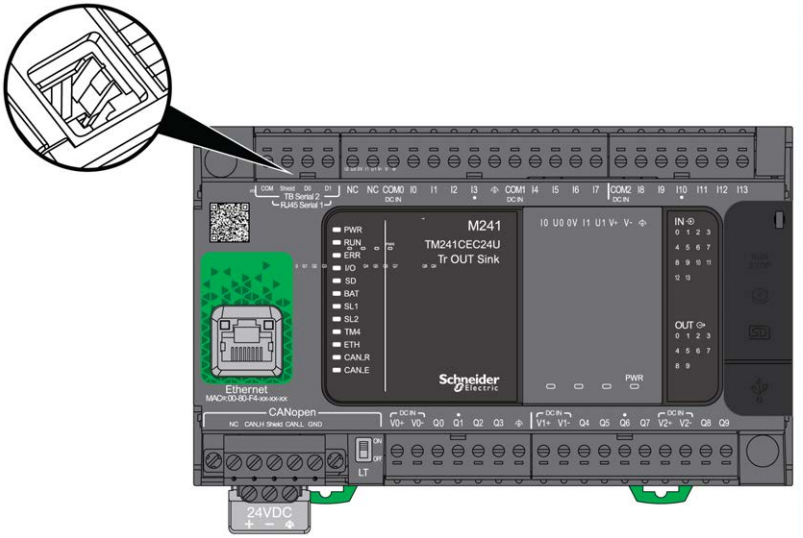
Serial Line 1

Overview

The serial line 1:

- can be used to communicate with devices supporting the Modbus protocol as either master or slave, ASCII protocol (printer, modem...) and Machine Expert Protocol (HMI,...).
- provides a 5 Vdc power distribution.

The following figure shows the location of the serial line 1 port:



Characteristics

Characteristic		Description
Function		RS485 or RS232 software configured
Connector type		RJ45
Isolation		Non-isolated
Maximum baud rate		1200 up to 115 200 bps
Cable	Type	Shielded
	Maximum length (between the controller and an isolated junction box)	15 m (49 ft) for RS485 3 m (9.84 ft) for RS232

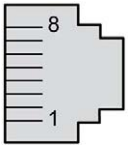
Characteristic	Description
Polarization	Software configuration is used to connect when the node is configured as a Master. 560 Ω resistors are optional.
5 Vdc power supply for RS485	Yes

NOTE: Some devices provide voltage on RS485 serial connections. Do not connect these voltage lines to your controller as they may damage the controller serial port electronics and render the serial port inoperable.

NOTICE
INOPERABLE EQUIPMENT Use only the VW3A8306R•• serial cable to connect RS485 devices to your controller. **Failure to follow these instructions can result in equipment damage.**

Pin Assignment

The following figure shows the pins of the RJ45 connector:



The table below describes the pin assignment of the RJ45 connector:

Pin	RS232	RS485
1	RxD	N.C.
2	TxD	N.C.
3	N.C.	N.C.
4	N.C.	D1
5	N.C.	D0
6	N.C.	N.C.
7	N.C.*	5 Vdc
8	Common	Common
* 5 Vdc delivered by the controller. Do not connect.		

N.C.: No Connection

RxD: Received Data

TxD: Transmitted Data

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Status LED

The following figure shows the status LED of the serial line 1:



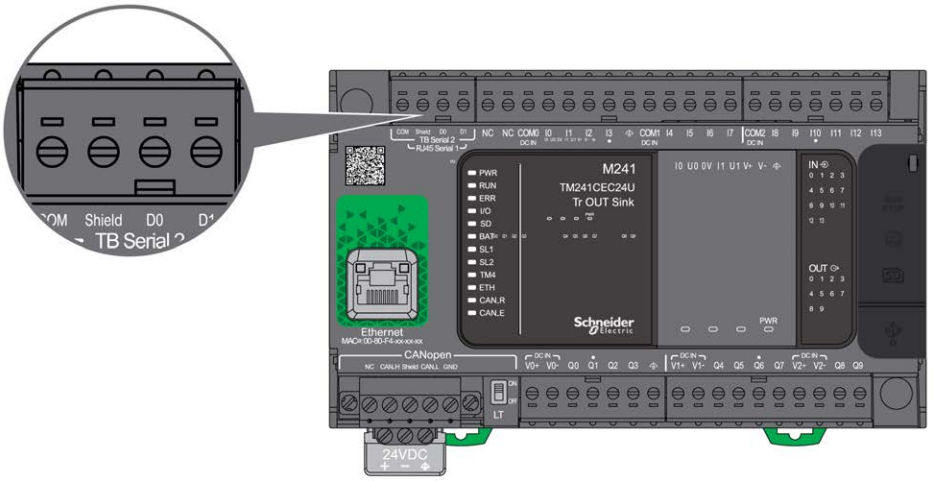
The table below describes the status LED of the serial line 1:

Label	Description	LED		
		Color	Status	Description
SL1	Serial Line 1	Green	On	Indicates the activity of the serial line 1
			Off	Indicates no serial communication

Serial Line 2

Overview

The serial line 2 is used to communicate with devices supporting the Modbus protocol as either a master or slave and ASCII Protocol (printer, modem...) and supports RS485 only.

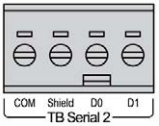


Characteristics

Characteristic		Description
Function		RS485
Connector type		Removable screw terminal block
Isolation		Non-isolated
Maximum baud rate		1200 up to 115 200 bps
Cable	Type	Shielded
	Maximum length	15 m (49 ft) for RS485
Polarization		Software configuration is used to connect when the node is configured as a Master. 560 Ω resistors are optional.
5 Vdc power supply for RS485		No

Pin Assignment

The following figure shows the pins of the removable terminal block:



Pin	RS485
COM	0 V com.
Shield	Shield
D0	D0 (B-)
D1	D1 (A+)

Refer to Removing Terminal Block *(see page 91)*.

Status LED

The following graphic show the status LED:



The table below describes the serial line 2 status LED:

Label	Description	LED		
		Color	Status	Description
SL2	Serial Line 2	Green	On	Indicates the activity of the serial line 2.
			Off	Indicates no serial communication.

Chapter 21
Connecting the M241 Logic Controller to a PC

Connecting the Controller to a PC

Overview

To transfer, run, and monitor the applications, connect the controller to a computer, that has EcoStruxure Machine Expert installed, using either a USB cable or an Ethernet connection (for those references that support an Ethernet port).

NOTICE
INOPERABLE EQUIPMENT Always connect the communication cable to the PC before connecting it to the controller. **Failure to follow these instructions can result in equipment damage.**

USB Powered Download

In order to execute limited operations, the M241 Logic Controller has the capability to be powered through the USB Mini-B port. A diode mechanism avoids having the logic controller both powered by USB and by the normal power supply, or to supply voltage on the USB port.

When powered only by USB, the logic controller executes the firmware and the boot project (if any) and the I/O board is not powered during boot (same duration as a normal boot). USB powered download initializes the internal flash memory with some firmware or some application and parameters when the controller is powered by USB. The preferred tool to connect to the controller is the **Controller Assistant**. Refer to the *EcoStruxure Machine Expert Controller Assistant User Guide*.

The controller packaging allows easy access to USB Mini-B port with minimum opening of the packaging. You can connect the controller to the PC with a USB cable. Long cables are not suitable for the USB powered download.

⚠ WARNING
INSUFFICENT POWER FOR USB DOWNLOAD Do not use a USB cable longer than 3m (9.8 ft) for USB powered download. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: It is not intended that you use the USB Powered Download on an installed controller. Depending on the number of I/O expansion modules in the physical configuration of the installed controller, there may be insufficient power from your PC USB port to accomplish the download.

USB Mini-B Port Connection

TCSXCNAMUM3P: This USB cable is suitable for short duration connections such as quick updates or retrieving data values.

BMXXCAUSBH018: Grounded and shielded, this USB cable is suitable for long duration connections.

NOTE: You can only connect 1 controller or any other device associated with EcoStruxure Machine Expert and its component to the PC at any one time.

The USB Mini-B Port is the programming port you can use to connect a PC with a USB host port using EcoStruxure Machine Expert software. Using a typical USB cable, this connection is suitable for quick updates of the program or short duration connections to perform maintenance and inspect data values. It is not suitable for long-term connections such as commissioning or monitoring without the use of specially adapted cables to help minimize electromagnetic interference.

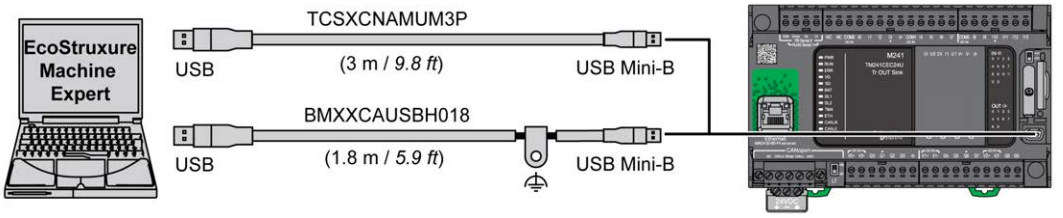
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- You must use a shielded USB cable such as a BMX XCAUSBH0•• secured to the functional ground (FE) of the system for any long-term connection.
- Do not connect more than one controller or bus coupler at a time using USB connections.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

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

The communication cable should be connected to the PC first to minimize the possibility of electrostatic discharge affecting the controller.

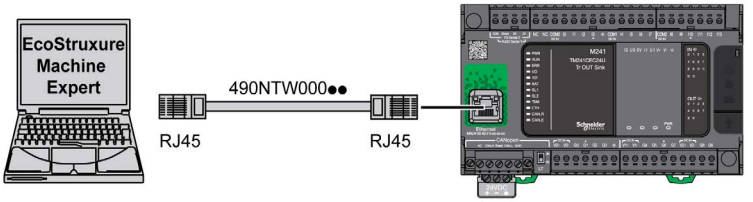


To connect the USB cable to your controller, follow the steps below:

Step	Action
1	**1a** If making a long-term connection using the cable BMXXCAUSBH018, or other cable with a ground shield connection, be sure to securely connect the shield connector to the functional ground (FE) or protective ground (PE) of your system before connecting the cable to your controller and your PC. **1b** If making a short-term connection using the cable TCSXCNAMUM3P or other non-grounded USB cable, proceed to step 2.
2	Connect your USB cable to the computer.
3	Open the hinged access cover.
4	Connect the Mini connector of your USB cable to the controller USB connector.

Ethernet Port Connection

You can also connect the controller to a PC using an Ethernet cable.



To connect the controller to the PC, do the following:

Step	Action
1	Connect the Ethernet cable to the PC.
2	Connect the Ethernet cable to the Ethernet port on the controller.

Glossary

A

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

bps

(*bit per second*) A definition of transmission rate, also given in conjunction with multiplicator kilo (kbps) and mega (mbps).

C

CANopen

An open industry-standard communication protocol and device profile specification (EN 50325-4).

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

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.

continuous function chart language

A graphical programming language (an extension of the IEC61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to inputs of other blocks to create complex expressions.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

D

DIN

(*Deutsches Institut für Normung*) A German institution that sets engineering and dimensional standards.

E

EIA rack

(*electronic industries alliance rack*) A standardized (EIA 310-D, IEC 60297, and DIN 41494 SC48D) system for mounting various electronic modules in a stack or rack that is 19 inches (482.6 mm) wide.

EN

EN identifies one of many European standards maintained by CEN (*European Committee for Standardization*), CENELEC (*European Committee for Electrotechnical Standardization*), or ETSI (*European Telecommunications Standards Institute*).

F

FBD

(*function block diagram*) One of 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks, where each network contains a graphical structure of boxes and connection lines, which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

FE

(functional Earth) A common grounding connection to enhance or otherwise allow normal operation of electrically sensitive equipment (also referred to as functional ground in North America).

In contrast to a protective Earth (protective ground), a functional earth connection serves a purpose other than shock protection, and may normally carry current. Examples of devices that use functional earth connections include surge suppressors and electromagnetic interference filters, certain antennas, and measurement instruments.

FreqGen

(*frequency generator*) A function that generates a square wave signal with programmable frequency.

H

HE10

Rectangular connector for electrical signals with frequencies below 3 MHz, complying with IEC 60807-2.

HSC

(*high-speed counter*) A function that counts pulses on the controller or on expansion module inputs.

I

I/O

(*input/output*)

IEC

(*international electrotechnical commission*) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

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).

IP 20

(*ingress protection*) The protection classification according to IEC 60529 offered by an enclosure, shown by the letter IP and 2 digits. The first digit indicates 2 factors: helping protect persons and for equipment. The second digit indicates helping protect against water. IP 20 devices help protect against electric contact of objects larger than 12.5 mm, but not against water.

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).

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).

M

master/slave

The single direction of control in a network that implements the master/slave mode.

Modbus

The protocol that allows communications between many devices connected to the same network.

N

NEMA

(*national electrical manufacturers association*) The standard for the performance of various classes of electrical enclosures. The NEMA standards cover corrosion resistance, ability to help protect from rain, submersion, and so on. For IEC member countries, the IEC 60529 standard classifies the ingress protection rating for enclosures.

P

PDO

(*process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PE

(*Protective Earth*) A common grounding connection to help avoid the hazard of electric shock by keeping any exposed conductive surface of a device at earth potential. To avoid possible voltage drop, no current is allowed to flow in this conductor (also referred to as *protective ground* in North America or as an equipment grounding conductor in the US national electrical code).

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

PTO

(*pulse train outputs*) A fast output that oscillates between off and on in a fixed 50-50 duty cycle, producing a square wave form. PTO is especially well suited for applications such as stepper motors, frequency converters, and servo motor control, among others.

PWM

(*pulse width modulation*) A fast output that oscillates between off and on in an adjustable duty cycle, producing a rectangular wave form (though you can adjust it to produce a square wave).

R

RJ45

A standard type of 8-pin connector for network cables defined for Ethernet.

RPDO

(*receive process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

RS-232

A standard type of serial communication bus, based on 3 wires (also known as EIA RS-232C or V.24).

RS-485

A standard type of serial communication bus, based on 2 wires (also known as EIA RS-485).

RxD

The line that receives data from one source to another.

S

SFC

(*sequential function chart*) A language that is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

T

terminal block

(*terminal block*) The component that mounts in an electronic module and provides electrical connections between the controller and the field devices.

TPDO

(*transmit process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

TxD

The line that sends data from one source to another.

Index

Symbols

Short-circuit or Over-current on Sink Transistor Outputs, *59*

A

accessories, *45*
analog input modules
    specifications, *35*
analog mixed I/O modules
    specifications, *37*
analog output modules
    specifications, *36*

B

bus coupler
    specifications, *40*

C

CANopen communication, *228*
certifications and standards, *72*
communication
    CANopen, *228*
Communication Ports, *227*
    Ethernet Port, *232*
    Serial Line 1, *237*
    Serial Line 2, *240*
    USB Programming Port, *235*
connections
    to CANopen slaves, *228*
    to J1939 ECUs, *229*

D

digital I/O modules
    specifications, *27*, *28*, *31*, *32*, *34*
Digital I/O modules
    Specifications, *28*

E

ECUs, max. number of J1939, *229*
Electrical Requirements
    Installation, *87*
Electromagnetic Susceptibility, *71*
Environmental Characteristics, *69*

F

fallback
    configuring modes, *57*
features
    key features, *18*
fieldbus interface
    specifications, *42*
Filter
    Bounce Filter, *53*

G

Grounding, *100*

I

inductive load, output protection
    output protection, inductive load, *92*
Input Management, *52*
Installation, *67*
    Electrical Requirements, *87*
installation
    logic/motion controller installation, *73*
intended use, *8*

J

J1939
    capabilities, *229*

L

Latching, *54*
logic/motion controller installation, *73*

M

M241
TM241C24R, *107*
TM241C24T, *125*
TM241C24U, *143*
TM241C40R, *161*
TM241C40T, *173*
TM241C40U, *185*
TM241CE24R, *113*
TM241CE24T, *131*
TM241CE24U, *149*
TM241CE40R, *167*
TM241CE40T, *179*
TM241CE40U, *191*
TM241CEC24R, *119*
TM241CEC24T, *137*
TM241CEC24U, *155*
mounting positions, *77*

N

notice
loss of application data, *63*

O

output management, *56*

P

PGNs, max. number of J1939, *229*
Power Supply, *94*, *98*
presentation
TM241C24R, *107*
TM241C24T, *125*
TM241C24U, *143*
TM241C40R, *161*
TM241C40T, *173*
TM241C40U, *185*
TM241CE24R, *113*
TM241CE24T, *131*
TM241CE24U, *149*
TM241CE40R, *167*
TM241CE40T, *179*
TM241CE40U, *191*
TM241CEC24R, *119*
TM241CEC24T, *137*
TM241CEC24U, *155*
programming languages
IL, LD, Grafcet, *18*

Q

qualification of personnel, *8*

R

real time clock, *48*
regular inputs, *31*, *32*, *34*
regular transistor outputs, *31*, *32*, *34*
relay outputs, *31*, *32*, *34*
Run/Stop, *61*

S

SD Card, *63*
Serial Line 1
Communication Ports, *237*
Serial Line 2
Communication Ports, *240*
short-circuit or over-current on relay outputs, *60*
short-circuit or over-current on transistor outputs, *57*

specifications
analog input modules, *35*
analog mixed I/O modules, *37*
analog output modules, *36*
digital I/O modules, *27*, *31*, *32*, *34*
Specifications
Digital I/O modules, *28*, *28*
specifications
modules, *38*
transmitter and receiver modules, *39*

T

Tesys modules
specifications, *38*
TM241CEC24R
M241, *119*
presentation, *119*
TM241CEC24T
M241, *137*
presentation, *137*
TMC4, *26*
transmitter and receiver modules
specifications, *39*

U

USB Programming Port
Communication Ports, *235*

W

wiring, *88*

Modicon TMC4
Cartridges
Programming Guide

05/2019

EIO0000003107.00

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.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

© 2019 Schneider Electric. All rights reserved.

Table of Contents

	Safety Information	5
	About the Book	7
Chapter 1	Cartridge Configuration General Information	11
	I/O Configuration General Practices	12
	General Description	13
	Adding Cartridges to a Configuration	14
	Configuring Cartridges	15
	Updating Cartridges Firmware	18
Chapter 2	TMC4 Standard Cartridges	19
	TMC4AI2	20
	TMC4TI2	23
	TMC4AQ2	26
Chapter 3	TMC4 Application Cartridges	29
	TMC4HOIS01	30
	TMC4PACK01	33
Glossary		35
Index		37

Safety Information

Important Information

NOTICE

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



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



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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This document describes the software configuration of the TMC4 cartridges for EcoStruxure Machine Expert. For further information, refer to the separate documents provided in the EcoStruxure Machine Expert online help.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.1.

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert Programming Guide	*EIO0000002854 (ENG)* *EIO0000002855 (FRE)* *EIO0000002856 (GER)* *EIO0000002858 (SPA)* *EIO0000002857 (ITA)* *EIO0000002859 (CHS)*
Modicon M241 Logic Controller - Programming Guide	*EIO0000003059 (ENG)* *EIO0000003060 (FRE)* *EIO0000003061 (GER)* *EIO0000003062 (SPA)* *EIO0000003063 (ITA)* *EIO0000003064 (CHS)*
Modicon TMC4 Cartridges - Hardware Guide	*EIO0000003113 (ENG)* *EIO0000003114 (FRE)* *EIO0000003115 (GER)* *EIO0000003116 (SPA)* *EIO0000003117 (ITA)* *EIO0000003118 (CHS)*

Title of Documentation	Reference Number
Modicon M241 Logic Controller - Hardware Guide	*EIO0000003083 (ENG)* *EIO0000003084 (FRE)* *EIO0000003085 (GER)* *EIO0000003086 (SPA)* *EIO0000003087 (ITA)* *EIO0000003088 (CHS)*

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

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Chapter 1
Cartridge Configuration General Information

Introduction

This chapter provides general information to help you configure TMC4 cartridges for EcoStruxure Machine Expert.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
I/O Configuration General Practices	12
General Description	13
Adding Cartridges to a Configuration	14
Configuring Cartridges	15
Updating Cartridges Firmware	18

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 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 or, depending on the controller reference, to or from the controller (in the form of cartridges), then you must 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 no longer function while the embedded I/O that may be present in your controller continues 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.

General Description

Introduction

The TMC4 cartridges connect to Modicon M241 Logic Controllers to increase the number of I/Os available on the controller.

Cartridge Features

The following table describes the TMC4 cartridge features:

Reference	Description
TMC4AI2	TMC4 cartridge with 2 analog voltage or current inputs (0...10 V, 0...20 mA, 4...20 mA), 12 bits
TMC4TI2	TMC4 cartridge with 2 analog temperature inputs (thermocouple, RTD), 14 bits
TMC4AQ2	TMC4 cartridge with 2 analog voltage or current outputs (0...10 V, 4...20 mA), 16 bits
TMC4HOIS01	TMC4 application cartridge with 2 analog voltage or current inputs for hoisting load cells
TMC4PACK01	TMC4 application cartridge with 2 analog voltage or current inputs for packaging

Logic Controller Compatibility

NOTE: For more information on cartridge compatibility with specific controllers, refer to your controller-specific hardware guide.

The following table describes the number of TMC4 cartridges that can be installed in a Modicon M241 Logic Controller:

Reference	Cartridge Slots
TM241C24R	1
TM241CE24R	1
TM241CEC24R	1
TM241C24T	1
TM241CE24T	1
TM241CEC24T	1
TM241C24U	1
TM241CE24U	1
TM241CEC24U	1
TM241C40R	2
TM241CE40R	2
TM241C40T	2
TM241CE40T	2
TM241C40U	2
TM241CE40U	2

Adding Cartridges to a Configuration

Adding a Cartridge

TMC4 cartridges can be connected to Modicon M241 Logic Controllers with 1 or 2 available cartridge slots.

To add a cartridge to your configuration, select the cartridge in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Configuring Cartridges

I/O Configuration

The configuration of a cartridge is carried out through the **I/O Mapping** and **I/O Configuration** tabs of the cartridge module.

To display the configuration tabs:

Step	Action
1	In the **Devices tree**, double-click the cartridge. The **I/O Mapping** tab appears.
2	Edit the parameters of the **I/O Mapping** tab to configure the addresses used by the cartridge module and diagnostic information.
3	Click the **I/O Configuration** tab to configure the cartridge. For details on the **I/O Configuration** tab, refer to the description of individual modules.

I/O Mapping Tab Description

The **I/O Mapping** tab allows you to:

- Map input and output channels onto variables.
- View diagnostic information relating to the current status of the cartridge.

This figure shows an example of the **I/O Mapping** tab:

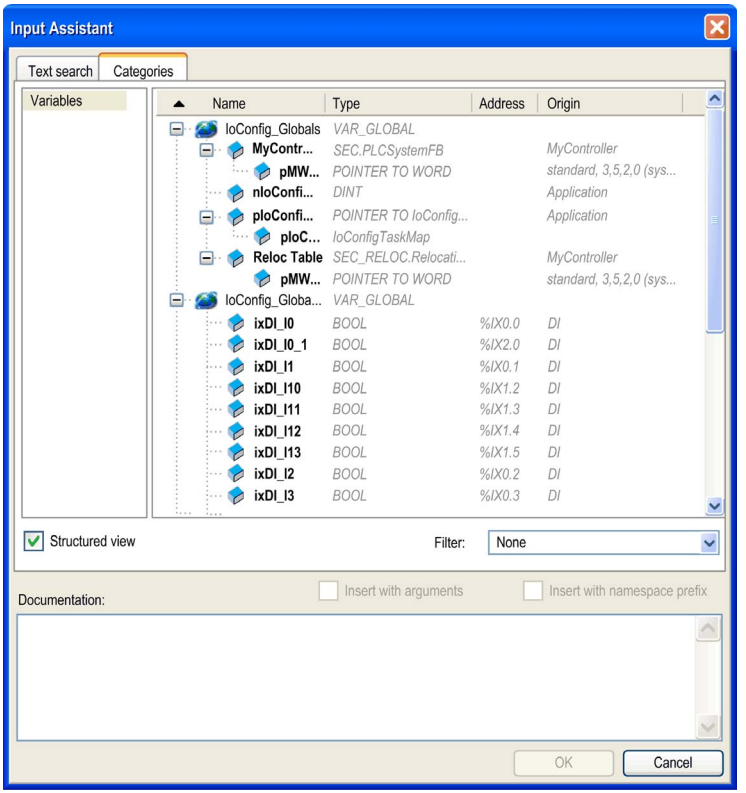
I/O Mapping | I/O Configuration | Information

Channels

Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
Inputs							
		IW0	%IW2	INT			
		IW1	%IW3	INT			
Diagnostic							
		IW2	%IB8	BYTE			
		Reserved	%IX8.0	BOOL			Reserved
		Reserved	%IX8.1	BOOL			Reserved
ixModule_1_2...		24VFault	%IX8.2	BOOL			+24 V Power disable
		Reserved	%IX8.3	BOOL			Reserved
		Reserved	%IX8.4	BOOL			Reserved
ixModule_1_O...		OutOfRan...	%IX8.5	BOOL			Input out of range (CH0)
ixModule_1_O...		OutOfRan...	%IX8.6	BOOL			Input out of range (CH1)
		Reserved	%IX8.7	BOOL			Reserved

I/O Mapping for Inputs/Outputs

This table describes each parameter of the **I/O Mapping** tab for inputs and outputs:

Parameter	Description
Variable	Allows you to map the channel on a variable. **NOTE:** Expand the list of variables from the category **Inputs** or **Outputs**. You can map a channel by either creating a new variable or mapping to an existing variable. **Create new variable**: Double-click the variable to enter the new variable name. A new variable is created if the variable does not already exist. **Map to existing variable**: Double-click the variable and click **[...]** to open the **Input Assistant** window. Select the variable from the list and press **OK**. This figure shows the **Input Assistant** window: 
Mapping	Indicates whether the channel is mapped on a new variable or an existing variable.

Parameter	Description
Channel	Displays the channel name of the device.
Address	Displays the address of the channel. **NOTE:** If the channel is mapped to an existing variable, corresponding address appears as strikethrough text in the table.
Type	Displays the data type of the channel.
Default Value	Indicates the value taken by the output when the controller is in a **STOPPED** or **HALT** state. Double-click the cell to change the default value.
Unit	Displays the unit of the channel value.
Description	Allows you to enter a short description of the channel.

Updating Cartridges Firmware

Introduction

The TMC4 cartridges have a firmware that you can update. The firmware update can only be done when the cartridge is mounted on the controller.

The firmware version of the cartridge can be seen in the `i_uifirmwareVersion` variable of the CART_R_STRUCT *(see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* in the M241 PLCSystem Library Guide.

The cartridge firmware is delivered in `.bin` files.

Description

When the controller starts, it checks if there is a file named `cart1.bin` or `cart2.bin` in the `/sys/0S` directory of the internal file system. If such a file is found, and if a cartridge is installed in the controller and configured, the firmware update of the cartridge starts.

NOTE: The firmware is only updated if the firmware file is different from the current firmware of the cartridge. The firmware file is not automatically deleted from the `/sys/0S` directory.

The firmware update operation lasts approximately 10 seconds per cartridge.

Procedure

Follow this procedure to update the cartridge firmware:

Step	Action
1	Copy the `.bin` file onto the SD card *(see Modicon M241 Logic Controller, Programming Guide)*.
2	Generate a script using the SD Card Mass Storage *(see Modicon M241 Logic Controller, Programming Guide)* editor and the **Download** command to store the `cart1.bin` file into the `/sys/0S` directory of the controller.
3	Insert the SD card into the controller.
4	Restart the controller. **NOTE:** The **PWR** LED of the cartridge is OFF to indicate that the firmware update is in progress.
5	Wait until the **PWR** LED of the cartridge is ON or flashing, indicating that the firmware update is complete.

Chapter 2
TMC4 Standard Cartridges

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TMC4AI2	20
TMC4TI2	23
TMC4AQ2	26

TMC4AI2

Introduction

The TMC4AI2 cartridge features 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 TMC4AI2 *(see Modicon TMC4, Cartridges, Hardware Guide)*.

If you have physically wired the analog channel for a voltage signal and you configure the channel for a current signal in EcoStruxure Machine Expert, 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.**

I/O Mapping Tab

Refer to Configuring Cartridges *(see page 15)* for a description of how to configure the inputs and outputs of the module.

Variables can be defined and named in the **I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

This table describes the **I/O Mapping** tab:

Variable	Channel	Type	Description
Inputs	iiTMC4AI2_IWO	INT	Current value of the input 0
	iiTMC4AI2_IW1	INT	Current value of the input 1
Diagnostic	ibTMC4AI2_IW2	BYTE	Status of the cartridge
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	24VFault	BOOL	+24 V power supply disabled
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	OutOfRange0	BOOL	Input out of range (channel 0)
	OutOfRange1	BOOL	Input out of range (channel 1)
	Reserved	BOOL	Reserved

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 15)*.

I/O Configuration Tab

For each input, you can define:

Parameter		Value	Default Value	Description
Type		**Not used** **0 - 10 V** **0 - 20 mA** **4 - 20 mA**	**Not used**	Choose the mode of the channel.
Min.	**0 - 10 V**	-32768...32767	0	Specifies the lower measurement limit.
	0 - 20 mA		0	
	4 - 20 mA		4000	
Max.	**0 - 10 V**	-32768...32767	10000	Specifies the upper measurement limit.
	0 - 20 mA		20000	
	4 - 20 mA		20000	
Filter Level		**No Filter** **Filter1 (Shortest)** **...** **Filter6 (Longest)**	**No Filter**	Specifies the digital filtering level to apply on this channel.

TMC4TI2

Introduction

The TMC4TI2 cartridge features 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
- PT100
- PT1000
- NI100
- NI1000

For further hardware information, refer to TMC4TI2 *(see Modicon TMC4, Cartridges, Hardware Guide)*.

I/O Mapping Tab

Refer to Configuring Cartridges *(see page 15)* for a description of how to configure the inputs and outputs of the module.

Variables can be defined and named in the **I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

This table describes the **I/O Mapping** tab:

Variable	Channel	Type	Description
Inputs	iiTMC4TI2_IWO	INT	Current value of the input 0
	iiTMC4TI2_IW1	INT	Current value of the input 1
	iiTMC4TI2_IW2	INT	Cold-junction (channel 0)
	iiTMC4TI2_IW3	INT	Cold-junction (channel 1)

Variable	Channel	Type	Description
Diagnostic	ibTMC4TI2_IW4	BYTE	Status of the cartridge
	BrokenWire0	BOOL	Input broken wire warning (channel 0)
	BrokenWire1	BOOL	Input broken wire warning (channel 1)
	24VFault	BOOL	+24 V power supply disabled
	ADCreinitialization	BOOL	0: input values are valid. 1: input values are not valid.
	Reserved	BOOL	Reserved
	OutOfRange0	BOOL	Input out of range (channel 0)
	OutOfRange1	BOOL	Input out of range (channel 1)
	Reserved	BOOL	Reserved

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 15)*.

I/O Configuration Tab

For each input, you can define:

Parameter	Value	Default Value	Description
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	**Customized** **Celsius (0.1°C)** **Fahrenheit (0.1°F)**	**Celsius (0.1°C)**	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.
WireBrakeDetection	**No** **Yes**	**No**	Whether to activate broken wire detection on this channel.

Parameter	Value	Default Value	Description
ColdJunctionEnable	**No** **Yes**	**Yes**	For thermocouple inputs, whether to activate internal cold junction compensation on this channel. Cold junction compensation automatically corrects for temperature variations at the thermocouple reference junction.
RTD Wire Mode	**2-wire** **3-wire** **4-wire**	**3-wire**	For PT100, PT100, NI100, and NI1000 input types, choose the resistor temperature detector (RTD) wiring mode to use.

Type	Celsius (0.1 °C)		Customized		Fahrenheit (0.1 F)	
	Minimum	Maximum	Minimum	Maximum	Minimum	Maximum
K Thermocouple	-2000	13000	-32768	32767	-3280	23720
J Thermocouple	-2000	10000	-32768	32767	-3280	18320
R Thermocouple	0	17600	-32768	32767	320	32000
S Thermocouple	0	17600	-32768	32767	320	32000
T Thermocouple	-2000	4000	-32768	32767	-3280	7520
B Thermocouple	0	18200	-32768	32767	7520	32720
E Thermocouple	-2000	8000	-32768	32767	-3280	14720
N Thermocouple	-2000	13000	-32768	32767	-3280	23720
PT100	-2000	8500	-32768	32767	-3280	15620
PT1000	-2000	8500	-32768	32767	-3280	15620
NI100	-600	1800	-32768	32767	-760	3560
NI1000	-600	1800	-32768	32767	-760	3560

TMC4AQ2

Introduction

The TMC4AQ2 cartridge features 2 voltage or current analog output channels with 16-bit resolution.

The channel output types are:

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

For further hardware information, refer to TMC4AQ2 *(see Modicon TMC4, Cartridges, Hardware Guide)*.

If you have physically wired the analog channel for a voltage signal and you configure the channel for a current signal in EcoStruxure Machine Expert, 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.**

I/O Mapping Tab

Refer to Configuring Cartridges *(see page 15)* for a description of how to configure the inputs and outputs of the module.

Variables can be defined and named in the **I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

This table describes the **I/O Mapping** tab:

Variable	Channel	Type	Description
Outputs	qiTMC4AQ2_QWO	INT	Current value of the output 0
	qiTMC4AQ2_QW1	INT	Current value of the output 1
Diagnostic	ibTMC4AQ2_IWO	BYTE	Status of the cartridge
	BrokenWire0	BOOL	Current output broken wire warning (channel 0)
	BrokenWire1	BOOL	Current output broken wire warning (channel 1)
	24VFault	BOOL	+24 V power supply disabled
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 15)*.

I/O Configuration Tab

For each output, you can define:

Parameter		Value	Default Value	Description
Type		**Not Used** **0 - 10 V** **4 - 20 mA**	**Not Used**	The mode of the channel.
Min.	**0 - 10 V** **4 - 20 mA**	-32768...32767 -32768...32767	0 4000	Specifies the lower measurement limit.
Max.	**0 - 10 V** **4 - 20 mA**	-32768...32767 -32768...32767	10000 20000	Specifies the upper measurement limit.

Chapter 3
TMC4 Application Cartridges

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TMC4HOIS01	30
TMC4PACK01	33

TMC4HOIS01

Introduction

The TMC4HOIS01 cartridge features 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 TMC4HOIS01 *(see Modicon TMC4, Cartridges, Hardware Guide)*.

If you have physically wired the analog channel for a voltage signal and you configure the channel for a current signal in EcoStruxure Machine Expert, 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.**

I/O Mapping Tab

Refer to Configuring Cartridges *(see page 15)* for a description of how to configure the inputs and outputs of the module.

Variables can be defined and named in the **I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

This table describes the **I/O Mapping** tab:

Variable	Channel	Type	Description
Inputs	iiTMC4HOIS01_IW0	INT	Current value of the input 0
	iiTMC4HOIS01_IW1	INT	Current value of the input 1
Diagnostic	ibTMC4HOIS01_IW2	BYTE	Status of the cartridge
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	24VFault	BOOL	+24 V power supply disabled
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	OutOfRange0	BOOL	Input out of range (channel 0)
	OutOfRange1	BOOL	Input out of range (channel 1)
	Reserved	BOOL	Reserved

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 15)*.

I/O Configuration Tab

For each input, you can define:

Parameter		Value	Default Value	Description
Type		**Not used** **0 - 10 V** **0 - 20 mA** **4 - 20 mA**	**Not used**	Choose the mode of the channel.
Min.	**0 - 10 V**	-32768...32767	0	Specifies the lower measurement limit.
	0 - 20 mA		0	
	4 - 20 mA		4000	
Max.	**0 - 10 V**	-32768...32767	10000	Specifies the upper measurement limit.
	0 - 20 mA		20000	
	4 - 20 mA		20000	
Filter Level		**No Filter** **Filter1 (Shortest)** **...** **Filter6 (Longest)**	**No Filter**	Specifies the digital filtering level to apply on this channel.

TMC4PACK01

Introduction

The TMC4PACK01 cartridge module features 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 TMC4PACK01 *(see Modicon TMC4, Cartridges, Hardware Guide)*.

If you have physically wired the analog channel for a voltage signal and you configure the channel for a current signal in EcoStruxure Machine Expert, 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.**

I/O Mapping Tab

Refer to Configuring Cartridges *(see page 15)* for a description of how to configure the inputs and outputs of the module.

Variables can be defined and named in the **I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

This table describes the **I/O Mapping** tab:

Variable	Channel	Type	Description
Inputs	iiTMC4PACK01_IW0	INT	Current value of the input 0
	iiTMC4PACK01_IW1	INT	Current value of the input 1
Diagnostic	ibTMC4PACK01_IW2	BYTE	Status of the cartridge
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	24VFault	BOOL	+24 V power supply disabled
	Reserved	BOOL	Reserved
	Reserved	BOOL	Reserved
	OutOfRange0	BOOL	Input out of range (channel 0)
	OutOfRange1	BOOL	Input out of range (channel 1)
	Reserved	BOOL	Reserved

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 15)*.

I/O Configuration Tab

For each input, you can define:

Parameter		Value	Default Value	Description
Type		**Not used** **0 - 10 V** **0 - 20 mA** **4 - 20 mA**	**Not used**	Choose the mode of the channel.
Min.	**0 - 10 V**	-32768...32767	0	Specifies the lower measurement limit.
	0 - 20 mA		0	
	4 - 20 mA		4000	
Max.	**0 - 10 V**	-32768...32767	10000	Specifies the upper measurement limit.
	0 - 20 mA		20000	
	4 - 20 mA		20000	
Filter Level		**No Filter** **Filter1 (Shortest)** **...** **Filter6 (Longest)**	**No Filter**	Specifies the digital filtering level to apply on this channel.

Glossary

A

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.

Index

C

cartridge
- compatibility, *13*
- description, *13*
- features, *13*

cartridges
- adding, *14*
- configuration, *15*
- configuring, *15*
- properties, *15*

compatibility
- cartridge, *13*

D

description
- cartridge, *13*

F

features
- cartridge, *13*

I

I/O configuration general information
- general practices, *12*

T

TMC4 analog I/O modules
- TMC4AI2, *20*
- TMC4AQ2, *26*
- TMC4HOIS01, *30*
- TMC4PACK01, *33*
- TMC4TI2 , *23*

Modicon TMC4
Cartridges
Hardware Guide

05/2019

EIO0000003113.00

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.

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

© 2019 Schneider Electric. All rights reserved.

Table of Contents

	Safety Information	5
	About the Book	7
Part I	**TMC4 General Overview**	13
Chapter 1	**TMC4 Description**	15
	General Description	15
Chapter 2	**TMC4 Installation**	17
2.1	TMC4 General Rules for Implementing	18
	Environmental Characteristics	19
	Certifications and Standards	20
2.2	TMC4 Installation	21
	Installation and Maintenance Requirements	22
	TMC4 Installation	24
2.3	TMC4 Electrical Requirements	31
	Wiring Best Practices	32
	Grounding the M241 System	35
Part II	**TMC4 Standard Cartridges**	39
Chapter 3	**TMC4AI2 Analog Voltage/Current Inputs**	41
	TMC4AI2 Presentation	42
	TMC4AI2 Characteristics	44
	TMC4AI2 Wiring Diagram	46
Chapter 4	**TMC4TI2 Analog Temperature Inputs**	47
	TMC4TI2 Presentation	48
	TMC4TI2 Characteristics	50
	TMC4TI2 Wiring Diagram	53
Chapter 5	**TMC4AQ2 Analog Voltage/Current Outputs**	55
	TMC4AQ2 Presentation	56
	TMC4AQ2 Characteristics	58
	TMC4AQ2 Wiring Diagram	60
Part III	**TMC4 Application Cartridges**	61
Chapter 6	**TMC4HOIS01 Hoisting**	63
	TMC4HOIS01 Presentation	64
	TMC4HOIS01 Characteristics	66
	TMC4HOIS01 Wiring Diagram	68

Chapter 7 TMC4PACK01 Packaging 69
TMC4PACK01 Presentation 70
TMC4PACK01 Characteristics 72
TMC4PACK01 Wiring Diagram 74
Glossary 75
Index 77

Safety Information

Important Information

NOTICE

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



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



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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

QUALIFICATION OF PERSONNEL

Only appropriately trained persons who are familiar with and understand the contents of this manual and all other pertinent product documentation are authorized to work on and with this product.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

INTENDED USE

The products described or affected by this document, together with software, accessories, and options, are cartridges, intended for industrial use according to the instructions, directions, examples, and safety information contained in the present document and other supporting documentation.

The product may only be used in compliance with all applicable safety regulations and directives, the specified requirements, and the technical data.

Prior to using the product, you must perform a risk assessment in view of the planned application. Based on the results, the appropriate safety-related measures must be implemented.

Since the product is used as a component in an overall machine or process, you must ensure the safety of persons by means of the design of this overall system.

Operate the product only with the specified cables and accessories. Use only genuine accessories and spare parts.

Any use other than the use explicitly permitted is prohibited and can result in unanticipated hazards.

About the Book

At a Glance

Document Scope

This guide describes the hardware implementation of TMC4. It provides the parts description, characteristics, wiring diagrams, and installation details for TMC4.

Validity Note

The information in this manual is applicable **only** for TMC4 products.

This document has been updated for the release of EcoStruxure™ Machine Expert V1.1.

For product compliance and environmental information (RoHS, REACH, PEP, EOLI, etc.), go to *www.schneider-electric.com/green-premium*.

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

Step	Action
1	Go to the Schneider Electric home page *www.schneider-electric.com*.
2	In the **Search** box type the reference of a product or the name of a product range. ● Do not include blank spaces in the reference or product range. ● To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the **Product Datasheets** search results and click on the reference that interests you. If you entered the name of a product range, go to the **Product Ranges** search results and click on the product range that interests you.
4	If more than one reference appears in the **Products** search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the datasheet.
6	To save or print a datasheet as a .pdf file, click **Download XXX product datasheet**.

The characteristics that are presented in the present document 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 document and online information, use the online information as your reference.

Related Documents

Title of Documentation	Reference Number
Modicon TMC4 Cartridges - Programming Guide	*EIO0000003107 (ENG)* *EIO0000003108 (FRE)* *EIO0000003109 (GER)* *EIO0000003110 (SPA)* *EIO0000003111 (ITA)* *EIO0000003112 (CHS)*
Modicon M241 Logic Controller - Hardware Guide	*EIO0000003083 (ENG)* *EIO0000003084 (FRE)* *EIO0000003085 (GER)* *EIO0000003086 (SPA)* *EIO0000003087 (ITA)* *EIO0000003088 (CHS)*

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

Product Related Information

⚠⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

⚠ DANGER

POTENTIAL FOR EXPLOSION

- Only use this equipment in non-hazardous locations, or in locations that comply with Class I, Division 2, Groups A, B, C and D.
- Do not substitute components which would impair compliance to Class I, Division 2.
- Do not connect or disconnect equipment unless power has been removed or the location is known to be non-hazardous.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Part I
TMC4 General Overview

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	TMC4 Description	15
2	TMC4 Installation	17

Chapter 1
TMC4 Description

General Description

Introduction

The cartridges are designed to be connected to the Modicon M241 Logic Controller range.

Cartridges Features

The following table describes the TMC4 cartridges features:

Reference	Description
TMC4AI2 *(see page 41)*	TMC4 cartridge with 2 analog voltage or current inputs (0...10 V, 0...20 mA, 4...20 mA), 12 bits
TMC4TI2 *(see page 47)*	TMC4 cartridge with 2 analog temperature inputs (thermocouple, RTD), 14 bits
TMC4AQ2 *(see page 55)*	TMC4 cartridge with 2 analog voltage or current outputs (0...10 V, 4...20 mA), 16 bits
TMC4HOIS01 *(see page 63)*	TMC4 application cartridge with 2 analog voltage or current inputs for hoisting load cells
TMC4PACK01 *(see page 69)*	TMC4 application cartridge with 2 analog voltage or current inputs for packaging

Logic Controller Compatibility

NOTE: For more information on cartridge compatibility with specific controllers, refer to your controller-specific hardware guide.

The following table describes the number of TMC4 cartridges that can be installed in a Modicon M241 Logic Controller:

Reference	Cartridge Slots
TM241C24R	1
TM241CE24R	1
TM241CEC24R	1
TM241C24T	1
TM241CE24T	1
TM241CEC24T	1
TM241C24U	1
TM241CE24U	1
TM241CEC24U	1
TM241C40R	2
TM241CE40R	2
TM241C40T	2
TM241CE40T	2
TM241C40U	2
TM241CE40U	2

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.

Chapter 2
TMC4 Installation

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	TMC4 General Rules for Implementing	18
2.2	TMC4 Installation	21
2.3	TMC4 Electrical Requirements	31

Section 2.1
TMC4 General Rules for Implementing

What Is in This Section?

This section contains the following topics:

Topic	Page
Environmental Characteristics	19
Certifications and Standards	20

Environmental Characteristics

TMC4

TMC4 cartridge environmental characteristics are the same as the Modicon M241 Logic Controller *(see Modicon M241 Logic Controller, Hardware Guide)*.

Certifications and Standards

Introduction

The M241 Logic Controllers are designed to conform to the main national and international standards concerning electronic industrial control devices:

- IEC/EN 61131-2
- UL 508

The M241 Logic Controllers have obtained the following conformity marks:

- CE
- cULus
- CSA

For product compliance and environmental information (RoHS, REACH, PEP, EOLI, etc.), go to *www.schneider-electric.com/green-premium*.

Section 2.2
TMC4 Installation

What Is in This Section?

This section contains the following topics:

Topic	Page
Installation and Maintenance Requirements	22
TMC4 Installation	24

Installation and Maintenance Requirements

Before Starting

Read and understand this chapter before beginning the installation of your system.

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

Pay particular attention in conforming to any safety information, different electrical requirements, and normative standards that would apply to your machine or process in the use of this equipment.

Disconnecting Power

All options and modules should be assembled and installed before installing the control system on a mounting rail, onto a mounting plate or in a panel. Remove the control system from its mounting rail, mounting plate or panel before disassembling the equipment.

⚠ ⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

Programming Considerations

⚠ 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.

Operating Environment

In addition to the **Environmental Characteristics**, refer to **Product Related Information** in the beginning of the present document for important information regarding installation in hazardous locations for this specific equipment.

NOTE: For important safety information and the environment characteristics of the TMC4 cartridges, see the M241 Logic Controller Hardware Guide.

Installation Considerations

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use appropriate safety interlocks where personnel and/or equipment hazards exist.
- Install and operate this equipment in an enclosure appropriately rated for its intended environment and secured by a keyed or tooled locking mechanism.
- Use the sensor and actuator power supplies only for supplying power to the sensors or actuators connected to the module.
- Power line and output circuits must be wired and fused in compliance with local and national regulatory requirements for the rated current and voltage of the particular equipment.
- Do not use this equipment in safety-critical machine functions unless the equipment is otherwise designated as functional safety equipment and conforming to applicable regulations and standards.
- Do not disassemble, repair, or modify this equipment.
- Do not connect any wiring to reserved, unused connections, or to connections designated as No Connection (N.C.).

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

NOTE: JDYX2 or JDYX8 fuse types are UL-recognized and CSA approved.

TMC4 Installation

Installation Considerations

The TMC4 cartridge is designed to operate within the same temperature range as the controllers, including the controller derating for extended temperature operation, and temperature restrictions associated with the mounting positions. Refer to the controller mounting position and clearance *(see Modicon M241 Logic Controller, Hardware Guide)* for more information.

Installation

⚠ ⚠ DANGER

ELECTRIC SHOCK OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Use protective gloves when installing or removing the cartridges.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

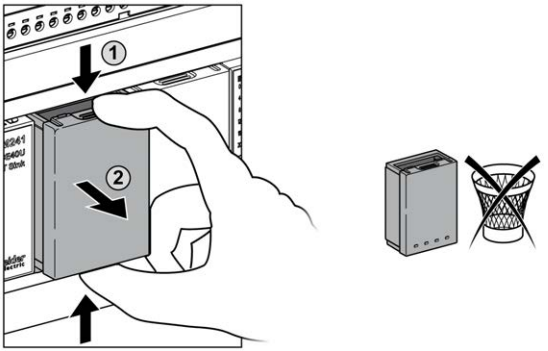
NOTICE

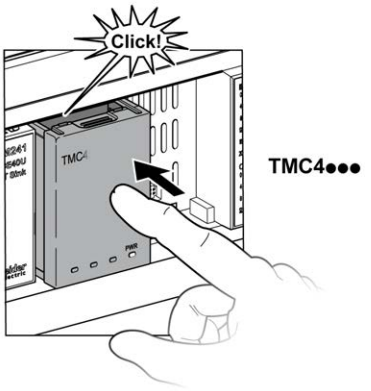
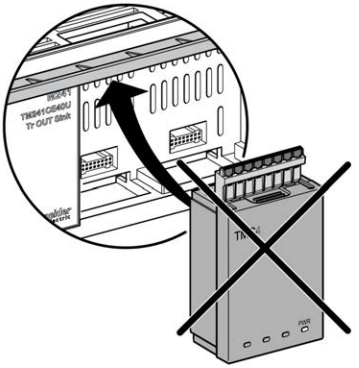
ELECTROSTATIC DISCHARGE

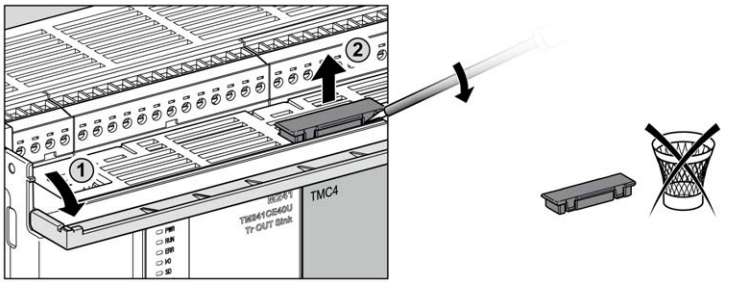
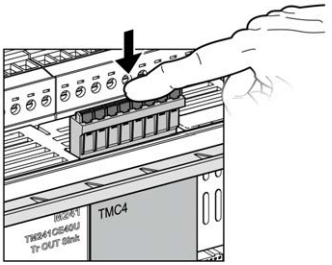
- Verify that empty cartridge slots have their covers in place before applying power to the controller.
- Do not touch the contacts of the cartridge.
- Only handle the cartridge on the housing.
- Take the necessary protective measures against electrostatic discharges.

Failure to follow these instructions can result in equipment damage.

The following table describes the different steps to install a TMC4 cartridge on the controller:

Step	Action
1	Disconnect all power from all equipment prior to removing any covers or installing a cartridge.
2	Remove the cartridge from the packaging.
3	Press the locking clips on the top and bottom of the cover with your fingers and pull up the cartridge slot cover gently. Remove by hand the cartridge slot cover from the controller. **NOTE:** Keep the cover to reuse it for the de-installation. 

Step	Action
4	Place the cartridge in the slot on the controller. Push the cartridge into the slot until it clicks.  **NOTE:** Do not insert the cartridge with its removable spring terminal block connected. 

Step	Action
5	Rotate the controller top connections cover to have more clearance to insert the cartridge removable spring terminal block. Press the locking clip on the side of the terminal block cover with an insulated screwdriver and pull up the cover gently. Remove the slot cover from the controller. **NOTE:** Keep the cover to reuse it for the de-installation. 
6	Insert the removable spring terminal block in the cartridge until it clicks. 

De-installation

⚠⚠ DANGER

ELECTRIC SHOCK OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Use protective gloves when installing or removing the cartridges.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

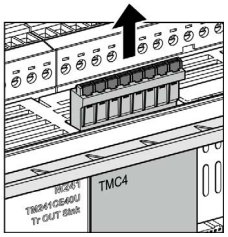
NOTICE

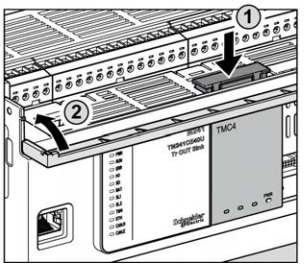
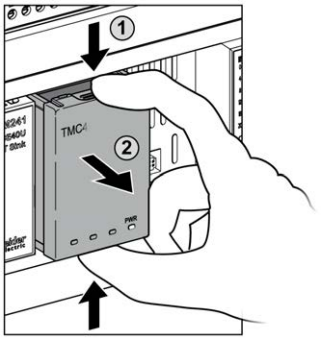
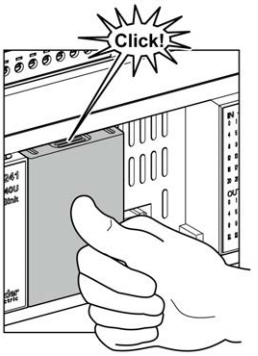
ELECTROSTATIC DISCHARGE

- 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.

The following table describes the different steps to de-install a TMC4 cartridge from the controller.

Step	Action
1	Disconnect all power from all equipment, including connected devices, prior to removing a cartridge.
2	Pull out by hand the removable spring terminal block from the cartridge. 

Step	Action
3	Place the terminal block slot cover in the slot on the top of the controller. Push the slot cover into the slot until it clicks. 
4	Press the locking clips on the top and bottom of the cartridge with your fingers and pull up the cartridge gently. Remove by hand the cartridge from the controller. 
5	Place the cartridge slot cover in the slot on the controller. Push the cartridge slot cover into the slot until it clicks. 

Section 2.3
TMC4 Electrical Requirements

What Is in This Section?

This section contains the following topics:

Topic	Page
Wiring Best Practices	32
Grounding the M241 System	35

Wiring Best Practices

Overview

This section describes the wiring guidelines and associated best practices to be respected when using the M241 Logic Controller system.

⚠⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

⚠ 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.

Wiring Guidelines

The following rules must be applied when wiring a M241 Logic Controller system:

- I/O and communication wiring must be kept separate from the power wiring. Route these 2 types of wiring in separate cable ducting.
- Verify that the operating conditions and environment are within the specification values.
- Use proper wire sizes to meet voltage and current requirements.
- Use copper conductors (required).
- Use twisted pair, shielded cables for analog, and/or fast I/O.
- Use twisted pair, shielded cables for networks, and fieldbus.

Use shielded, properly grounded cables for all analog and high-speed inputs or outputs and communication connections. If you do not use shielded cable for these connections, electromagnetic interference can cause signal degradation. Degraded signals can cause the controller or attached modules and equipment to perform in an unintended manner.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use shielded cables for all fast I/O, analog I/O and communication signals.
- Ground cable shields for all analog I/O, fast I/O and communication signals at a single point[1].
- Route communication and I/O cables separately from power cables.

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

[1]Multipoint grounding is permissible if connections are made to an equipotential ground plane dimensioned to help avoid cable shield damage in the event of power system short-circuit currents.

For more details, refer to Grounding Shielded Cables *(see page 35)*.

NOTE: Surface temperatures may exceed 60 °C (140 °F). To conform to IEC 61010 standards, route primary wiring (wires connected to power mains) separately and apart from secondary wiring (extra low voltage wiring coming from intervening power sources). If that is not possible, double insulation is required such as conduit or cable gains.

Rules for Removable Spring Terminal Block

The following table shows the cable types and wire sizes for a **3.81 mm (0.15 in.) pitch** removable spring terminal block:

mm / in. 9 / 0.35				
mm^2	0.2…1.5	0.2…1.5	0.25…1.5	0.25…0.75
AWG	*24…16*	*24…16*	*23...16*	*23...19*

The use of copper conductors is required.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

The spring clamp connectors of the terminal block are designed for only one wire or one cable end. Two wires to the same connector must be installed with a double wire cable end to help prevent loosening.

⚠ ⚠ DANGER

LOOSE WIRING CAUSES ELECTRIC SHOCK

Do not insert more than one wire per connector of the spring terminal blocks unless using a double wire cable end (ferrule).

Failure to follow these instructions will result in death or serious injury.

Grounding the M241 System

Overview

To help minimize the effects of electromagnetic interference, cables carrying the fast I/O, analog I/O, and field bus communication signals must be shielded.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use shielded cables for all fast I/O, analog I/O, and communication signals.
- Ground cable shields for all fast I/O, analog I/O, and communication signals at a single point[1].
- Route communications and I/O cables separately from power cables.

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

[1]Multipoint grounding is permissible if connections are made to an equipotential ground plane dimensioned to help avoid cable shield damage in the event of power system short-circuit currents.

The use of shielded cables requires compliance with the following wiring rules:

- For protective ground connections (PE), metal conduit or ducting can be used for part of the shielding length, provided there is no break in the continuity of the ground connections. For functional ground (FE), the shielding is intended to attenuate electromagnetic interference and the shielding must be continuous for the length of the cable. If the purpose is both functional and protective, as is often the case for communication cables, the cable must have continuous shielding.
- Wherever possible, keep cables carrying one type of signal separate from the cables carrying other types of signals or power.

Protective Ground (PE) on the Backplane

The protective ground (PE) should be connected to the conductive backplane by a heavy-duty wire, usually a braided copper cable with the maximum allowable cable section.

Shielded Cables Connections

Cables carrying the fast I/O, analog I/O, and field bus communication signals must be shielded. The shielding must be securely connected to ground. The fast I/O and analog I/O shields may be connected either to the functional ground (FE) or to the protective ground (PE) of your M241 Logic Controller. The field bus communication cable shields must be connected to the protective ground (PE) with a connecting clamp secured to the conductive backplane of your installation.

⚠ WARNING

ACCIDENTAL DISCONNECTION FROM PROTECTIVE GROUND (PE)

- Do not use the TM2XMTGB Grounding Plate to provide a protective ground (PE).
- Use the TM2XMTGB Grounding Plate only to provide a functional ground (FE).

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

The shielding of the Modbus cable must be connected to the protective ground (PE).

⚠⚠ DANGER

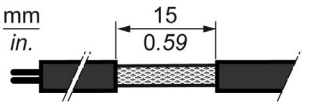
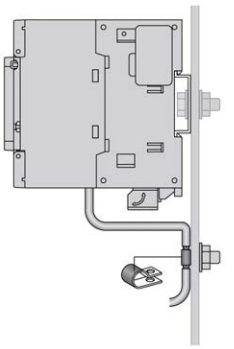
HAZARD OF ELECTRIC SHOCK

- The grounding terminal connection (PE) must be used to provide a protective ground at all times.
- Make sure that an appropriate, braided ground cable is attached to the PE/PG ground terminal before connecting or disconnecting the network cable to the equipment.

Failure to follow these instructions will result in death or serious injury.

Protective Ground (PE) Cable Shielding

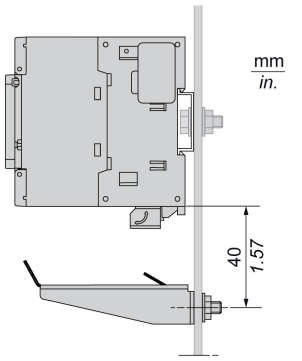
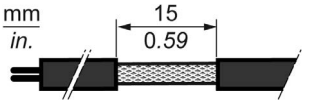
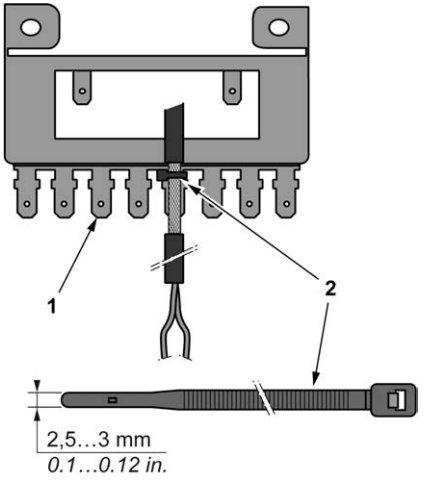
To ground the shield of a cable through a grounding clamp:

Step	Description	
1	Strip the shielding for a length of 15 mm (0.59 in.)	
2	Attach the cable to the conductive backplane plate by attaching the grounding clamp to the stripped part of the shielding as close as possible to the M241 Logic Controller system base.	

NOTE: The shielding must be clamped securely to the conductive backplane to ensure a good contact.

Functional Ground (FE) Cable Shielding

To connect the shield of a cable through the Grounding Bar:

Step	Description	
1	Install the Grounding Bar directly on the conductive backplane below the M241 Logic Controller system as illustrated.	
2	Strip the shielding for a length of 15 mm (0.59 in.	
3	Tightly clamp on the blade connector **(1)** using nylon fastener **(2)**(width 2.5...3 mm (0.1...0.12 in.)) and appropriate tool.	

NOTE: Use the TM2XMTGB Grounding Bar for Functional Ground (FE) connections.

Part II
TMC4 Standard Cartridges

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	TMC4AI2 Analog Voltage/Current Inputs	41
4	TMC4TI2 Analog Temperature Inputs	47
5	TMC4AQ2 Analog Voltage/Current Outputs	55

Chapter 3
TMC4AI2 Analog Voltage/Current Inputs

Overview

This chapter describes the TMC4AI2 cartridge, its characteristics, and its connections.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TMC4AI2 Presentation	42
TMC4AI2 Characteristics	44
TMC4AI2 Wiring Diagram	46

TMC4AI2 Presentation

Overview

The following features are integrated into the TMC4AI2 cartridge:

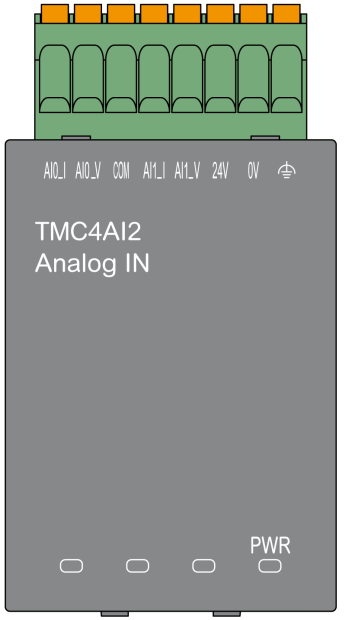
- 2 analog inputs (voltage or current)
- removable spring terminal block, 3.81 mm (0.15 in.) pitch

Main Characteristics

Characteristic		Value	
	Signal type	Voltage	Current
Number of input channels		2	
Input range		0...10 Vdc	0...20 mA 4...20 mA
Resolution		12 bits (4096 steps)	
Connection type		3.81 mm (0.15 in.) pitch, removable spring terminal block	
Weight		55 g (1.94 oz)	

Power LED

The following diagram shows a TMC4AI2 cartridge with its power LED labeled **PWR**:



LED	Color	Status	Description
PWR	Green	On	The cartridge is powered by the logic controller and the external power supply (24 Vdc) is applied.
		Flashing	The cartridge is powered by the logic controller but the external power supply (24 Vdc) is not applied.
		Off	The cartridge is not powered by the logic controller.

TMC4AI2 Characteristics

Introduction

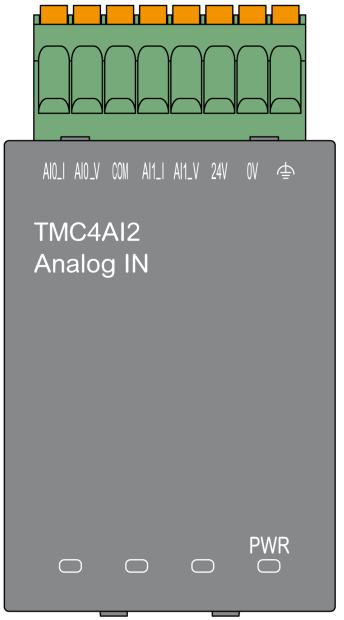
This section provides a general description of the TMC4AI2 cartridge characteristics.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not exceed any of the rated values specified in the environmental and electrical characteristics tables. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: For important safety information and the environment characteristics of the TMC4 cartridges, see the M241 Logic Controller Hardware Guide.

Connectors

The following diagram shows a TMC4AI2 cartridge marking and connectors:



Input Characteristics

The following table describes the cartridge input characteristics:

Characteristics		Value	
	Signal Type	Voltage	Current
Rated input range		0...10 Vdc	0...20 mA 4...20 mA
Input impedance		> 1 MΩ	< 250 Ω
Sample duration time		1 ms per enabled channel	
Input type		single-ended	
Operating mode		self-scan	
Conversion mode		SAR type	
Maximum accuracy at ambient temperature: 25 °C (77 °F)		± 0.2 % of full scale	
Maximum accuracy on full operating temperature range		± 0.5 % of full scale	
Temperature drift		± 0.006 % of full scale per 1 °C (1.8 °F)	
Repeatability after stabilization time		± 0.2 % of full scale	
Non-linearity		± 0.05 % of full scale	
Digital resolution		12 bits (4096 steps)	
Input value of LSB		2.44 mV	4.88 µV
Data type in application program		scalable from –32768 to 32767	
Input data out of detection range		yes	
Noise resistance	maximum temporary deviation during perturbations	± 2.0 % of full scale	
	cable type and maximum length	shielded	
		< 30 m (98.4 ft)	
	crosstalk (minimum)	80 dB	
	common-mode rejection ratio (minimum)	65 dB	
Isolation	isolation between inputs and internal logic	500 Vdc	
	isolation between inputs	not isolated	
Maximum continuous overload allowed (without damage)		30 Vdc	40 mA dc
Input filter		software filter: 6 levels	
External power supply	supply voltage	24 Vdc ± 15 %	
	power consumption	2 W	

TMC4AI2 Wiring Diagram

Introduction

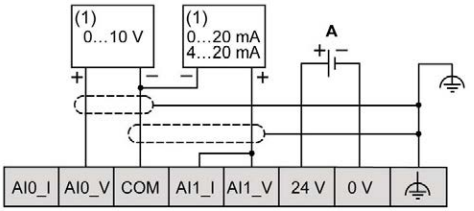
This cartridge has a removable spring terminal block for the connection of the inputs.

Wiring

See Wiring Best Practices *(see page 32)*.

Wiring Diagram

The following figure shows an example of the voltage and current input connection:



(1): Current/Voltage analog output device
A: External power supply

NOTE: Each input can be connected to either a voltage or current input.

Chapter 4
TMC4TI2 Analog Temperature Inputs

Overview

This chapter describes the TMC4TI2 cartridge, its characteristics, and its connections.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TMC4TI2 Presentation	48
TMC4TI2 Characteristics	50
TMC4TI2 Wiring Diagram	53

TMC4TI2 Presentation

Overview

The following features are integrated into the TMC4TI2 cartridge:

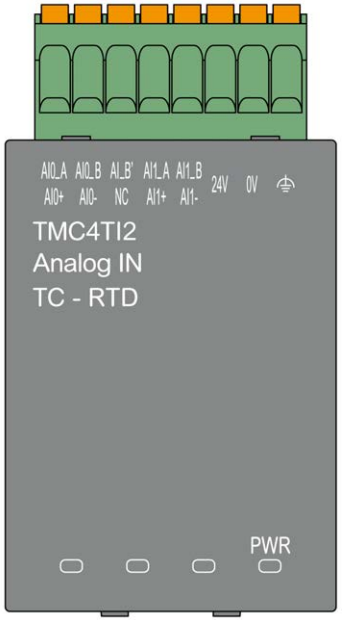
- 2 analog temperature inputs (thermocouple or RTD)
- removable spring terminal block, 3.81 mm (0.15 in.) pitch

Main Characteristics

Characteristic		Value	
	Signal type	Thermocouple	RTD
Number of input channels		2	
Input range		type: K, J, R, S, B, E, T, N	type: Pt100, Pt1000, Ni100, Ni1000
Resolution		14 bits (16384 steps)	
Connection type		3.81 mm (0.15 in.) pitch, removable spring terminal block	
Weight		55 g (1.94 oz)	

Power LED

The following diagram shows a TMC4TI2 cartridge with its power LED labeled **PWR**:



LED	Color	Status	Description
PWR	Green	On	The cartridge is powered by the logic controller and the external power supply (24 Vdc) is applied.
		Flashing	The cartridge is powered by the logic controller but the external power supply (24 Vdc) is not applied.
		Off	The cartridge is not powered by the logic controller.

TMC4TI2 Characteristics

Introduction

This section provides a general description of the TMC4TI2 cartridge characteristics.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not exceed any of the rated values specified in the environmental and electrical characteristics tables. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: For important safety information and the environment characteristics of the TMC4 cartridges, see the M241 Logic Controller Hardware Guide.

Connectors

The following diagram shows a TMC4TI2 cartridge marking and connectors:



Input Characteristics

The following table describes the cartridge input characteristics:

Characteristics		Value	
	Signal Type	**Thermocouple**	**RTD (2, 3, or 4 wires)**
Rated input range		thermocouple type: **K:** –200...+1300 °C (–328...+2372 °F) **J:** –200...+1000 °C (–328...+1832 °F) **R:** 0...+1760 °C (+32...+3200 °F) **S:** 0...+1760 °C (+32...+3200 °F) **B:** +250...+1820 °C (+482...+3308 °F) **E:** –200...+800 °C (–328...+1472 °F) **T:** –200...+400 °C (–328...+752 °F) **N:** –200...+1300 °C (–328...+2372 °F)	RTD type: **Pt100:** –200...+850 °C (–328...+1562 °F) **Pt1000:** –200...+850 °C (–328...+1562 °F) **Ni100:** –60...+180 °C (–76...+356 °F) **Ni1000:** –60...+180 °C (–76...+356 °F)
Cold junction compensation		internal compensation	–
Input impedance		> 1 MΩ	
Sample duration time		100 ms per enabled channel + 1 scan time	
Input type		single-ended	
Operating mode		self-scan	
Conversion mode		SAR type	
Maximum accuracy at ambient temperature: 25 °C (77 °F)		**K, J, R, S, E, T, N:** ± 0.2 % of full scale + junction compensation accuracy (± 4 °C (± 7.2 °F)) **B:** ± 0.2 % of full scale for measured temperature range: 250...400 °C (482...752 °F) ± 0.1 % of full scale for measured temperature range: 400...1280 °C (752...2336 °F)	± 0.5 °C (± 0.9 °F)
Temperature drift		± 0.008 % of full scale per 1 °C (1.8 °F)	
Repeatability after stabilization time		± 0.1 % of full scale	
Non-linearity		± 0.05 % of full scale	
Digital resolution		14 bits (16384 steps)	

Characteristics		Value	
	Signal Type	Thermocouple	RTD (2, 3, or 4 wires)
Input value of LSB		0.1 °C (0.18 °F)	
Data type in application program		scalable from –32768 to 32767	
Input data out of detection range		yes	
Noise resistance	maximum temporary deviation during perturbations	± 2 % of full scale	
	total cable type, length, and resistance	twisted-pair shielded	
		< 100 m (328.1 ft)	
		< 100 Ω	< 30 Ω
	external crosstalk (minimum)	80 dB	
	50/60 Hz common-mode rejection ratio (minimum)	90 dB	
	50/60 Hz differential-mode rejection ratio (minimum)	60 dB	
Isolation	isolation between inputs and internal logic	500 Vdc	
	isolation between inputs	not isolated	
Maximum continuous overload allowed (without damage)		6 Vdc	
Behavior when the temperature sensor is disconnected or broken		detected	
External power supply	supply voltage	24 Vdc ± 15 %	
	power consumption	2 W	

TMC4TI2 Wiring Diagram

Introduction

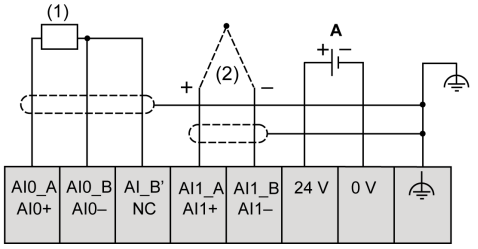
This cartridge has a removable spring terminal block for the connection of the inputs.

Wiring

See Wiring Best Practices *(see page 32)*.

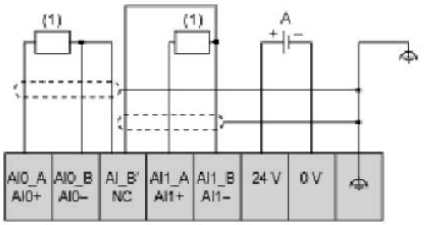
Wiring Diagram

The following figure shows an example of 3-wire RTD and thermocouple probe connections:



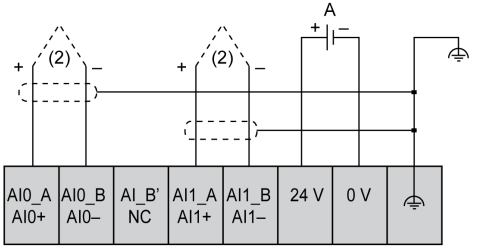
(1): RTD
(2): Thermocouple
A: External power supply

The following figure shows an example of a pair of 3-wire RTD connections:



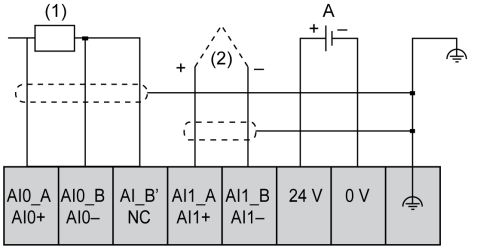
(1): RTD
A: External power supply

The following figure shows an example of a pair of thermocouple connections:



(2): Thermocouple
A: External power supply

The following figure shows an example of 4-wire RTD and thermocouple connections:



(1): RTD
(2): Thermocouple
A: External power supply

NOTE: Each input can be connected to either an RTD or thermocouple probe.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION
Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Chapter 5
TMC4AQ2 Analog Voltage/Current Outputs

Overview

This chapter describes the TMC4AQ2 cartridge, its characteristics, and its connections.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TMC4AQ2 Presentation	56
TMC4AQ2 Characteristics	58
TMC4AQ2 Wiring Diagram	60

TMC4AQ2 Presentation

Overview

The following features are integrated into the TMC4AQ2 cartridge:

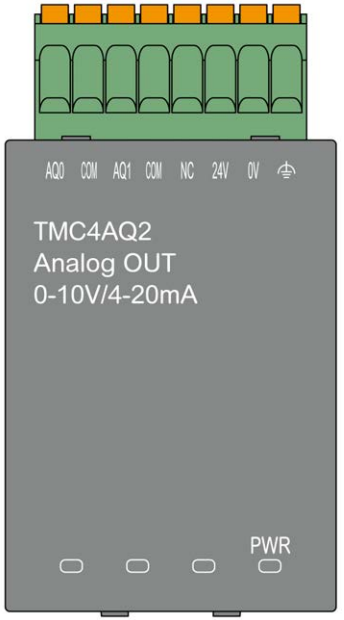
- 2 analog outputs (voltage or current)
- removable spring terminal block, 3.81 mm (0.15 in.) pitch

Main Characteristics

Characteristic		Value	
	Signal type	Voltage	Current
Number of output channels		2	
Output range		0...10 Vdc	4...20 mA (dc)
Resolution		16 bits (65536 steps)	
Connection type		3.81 mm (0.15 in.) pitch, removable spring terminal block	
Weight		55 g (1.94 oz)	

Power LED

The following diagram shows a TMC4AQ2 cartridge with its power LED labeled **PWR**:



LED	Color	Status	Description
PWR	Green	On	The cartridge is powered by the logic controller and the external power supply (24 Vdc) is applied.
		Flashing	The cartridge is powered by the logic controller but the external power supply (24 Vdc) is not applied.
		Off	The cartridge is not powered by the logic controller.

TMC4AQ2 Characteristics

Introduction

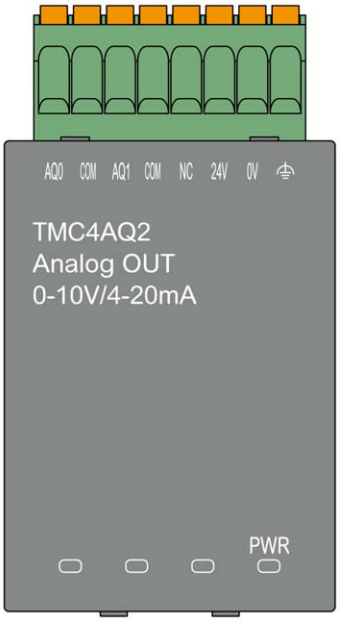
This section provides a general description of the TMC4AQ2 cartridge characteristics.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not exceed any of the rated values specified in the environmental and electrical characteristics tables. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: For important safety information and the environment characteristics of the TMC4 cartridges, see the M241 Logic Controller Hardware Guide.

Connectors

The following diagram shows a TMC4AQ2 cartridge marking and connectors:



IOutput Characteristics

The following table describes the cartridge output characteristics:

Characteristics		Value	
	Signal Type	Voltage	Current
Rated output range		0...10 Vdc	4...20 mA (dc)
Load impedance		> 2 KΩ	< 500 Ω
Application load type		resistive load	
Settling time		10 ms	
Total output system transfer time		10 ms + 1 scan time	
Maximum accuracy at ambient temperature without EMC disturbance: 25 °C (77 °F)		± 0.2 % of full scale	
Temperature drift		± 0.006 % of full scale per 1 °C (1.8 °F)	
Repeatability after stabilization time		± 0.5 % of full scale	
Non-linearity		± 0.05 % of full scale	
Output ripple		± 20 mV	
Output voltage drop		1 %	
Overshoot		0 %	
Maximum output deviation		± 0.5 % of full scale	
Digital resolution		16 bits (65536 steps)	
Output value of LSB		0.153 mV	0.305 µA
Data type in application program		0...4095	
Noise resistance	maximum temporary deviation during perturbations	± 2 % of full scale	
	cable type and maximum length	shielded	
		< 30 m (98.4 ft)	
	external crosstalk (minimum)	80 dB	
	50/60 Hz common-mode rejection ratio (minimum)	90 dB	
Isolation	isolation between outputs and internal logic	500 Vdc	
	isolation between outputs	not isolated	
Output protection		short circuit protection	open circuit protection
Behavior when internal power supply level is lower than threshold		outputs are set to 0	
Behavior when external power is not applied		**PWR** LED flashing	
External power supply	supply voltage	24 Vdc ± 15 %	
	power consumption	2 W	

TMC4AQ2 Wiring Diagram

Introduction

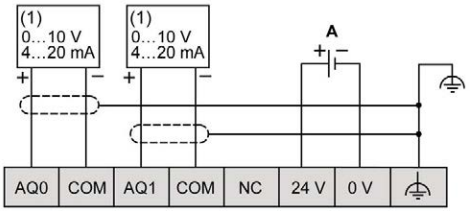
This cartridge has a removable spring terminal block for the connection of the outputs.

Wiring

See Wiring Best Practices *(see page 32)*.

Wiring Diagram

The following figure shows an example of the voltage and current outputs connection:



(1): Current/Voltage analog input device
A: External power supply

NOTE: Each output can be connected either as a voltage or current output.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”.

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

Part III
TMC4 Application Cartridges

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
6	TMC4HOIS01 Hoisting	63
7	TMC4PACK01 Packaging	69

Chapter 6
TMC4HOIS01 Hoisting

Overview

This chapter describes the TMC4HOIS01 cartridge, its characteristics, and its connections.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TMC4HOIS01 Presentation	64
TMC4HOIS01 Characteristics	66
TMC4HOIS01 Wiring Diagram	68

TMC4HOIS01 Presentation

Overview

The following features are integrated into the TMC4HOIS01 cartridge:

- 2 analog inputs (voltage or current) for hoisting load cell
- removable spring terminal block, 3.81 mm (0.15 in.) pitch

Main Characteristics

Characteristic		Value	
	Signal type	Voltage	Current
Number of input channels		2	
Input range		0...10 Vdc	0...20 mA 4...20 mA
Resolution		12 bits (4096 steps)	
Connection type		3.81 mm (0.15 in.) pitch, removable spring terminal block	
Weight		55 g (1.94 oz)	

Power LED

The following diagram shows a TMC4HOIS01 cartridge with its power LED labeled **PWR**:



LED	Color	Status	Description
PWR	Green	On	The cartridge is powered by the logic controller and the external power supply (24 Vdc) is applied.
		Flashing	The cartridge is powered by the logic controller but the external power supply (24 Vdc) is not applied.
		Off	The cartridge is not powered by the logic controller.

TMC4HOIS01 Characteristics

Introduction

This section provides a general description of the TMC4HOIS01 cartridge characteristics.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not exceed any of the rated values specified in the environmental and electrical characteristics tables. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: For important safety information and the environment characteristics of the TMC4 cartridges, see the M241 Logic Controller Hardware Guide.

Connectors

The following diagram shows a TMC4HOIS01 cartridge marking and connectors:



Input Characteristics

The following table describes the cartridge input characteristics:

Characteristics		Value	
	Signal Type	Voltage	Current
Rated input range		0...10 Vdc	0...20 mA 4...20 mA
Input impedance		> 1 MΩ	< 250 Ω
Sample duration time		1 ms per enabled channel	
Input type		single-ended	
Operating mode		self-scan	
Conversion mode		SAR type	
Maximum accuracy at ambient temperature: 25 °C (77 °F)		± 0.2 % of full scale	
Maximum accuracy on full operating temperature range		± 0.5 % of full scale	
Temperature drift		± 0.006 % of full scale per 1 °C (1.8 °F)	
Repeatability after stabilization time		± 0.2 % of full scale	
Non-linearity		± 0.05 % of full scale	
Digital resolution		12 bits (4096 steps)	
Input value of LSB		2.44 mV	4.88 µV
Data type in application program		scalable from –32768 to 32767	
Input data out of detection range		yes	
Noise resistance	maximum temporary deviation during perturbations	± 2.0 % of full scale	
	cable type and maximum length	shielded	
		< 30 m (98.4 ft)	
	crosstalk (minimum)	80 dB	
	common-mode rejection ratio (minimum)	65 dB	
Isolation	isolation between inputs and internal logic	500 Vdc	
	isolation between inputs	not isolated	
Maximum continuous overload allowed (without damage)		30 Vdc	40 mA dc
Input filter		software filter: 6 levels	
External power supply	supply voltage	24 Vdc ± 15 %	
	power consumption	2 W	

TMC4HOIS01 Wiring Diagram

Introduction

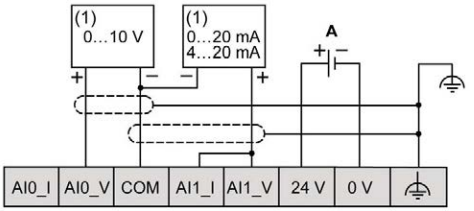
This cartridge has a removable spring terminal block for the connection of the inputs.

Wiring

See Wiring Best Practices *(see page 32)*.

Wiring Diagram

The following figure shows an example of the voltage and current input connection:



(1): Current/Voltage analog output device
A: External power supply

NOTE: Each input can be connected to either a voltage or current input.

Chapter 7
TMC4PACK01 Packaging

Overview

This chapter describes the TMC4PACK01 cartridge, its characteristics, and its connections.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TMC4PACK01 Presentation	70
TMC4PACK01 Characteristics	72
TMC4PACK01 Wiring Diagram	74

TMC4PACK01 Presentation

Overview

The following features are integrated into the TMC4PACK01 cartridge:

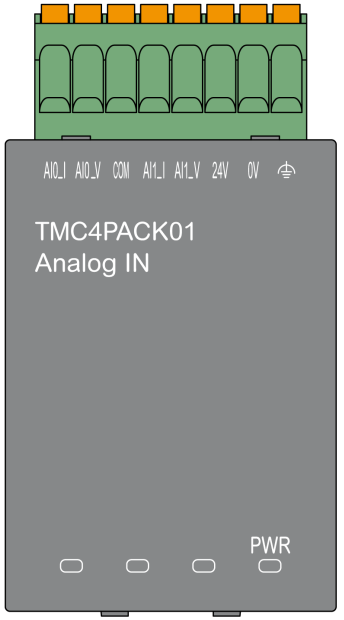
- 2 analog inputs (voltage or current) for packaging
- removable spring terminal block, 3.81 mm (0.15 in.) pitch

Main Characteristics

Characteristic		Value	
	Signal type	Voltage	Current
Number of input channels		2	
Input range		0...10 Vdc	0...20 mA 4...20 mA
Resolution		12 bits (4096 steps)	
Connection type		3.81 mm (0.15 in.) pitch, removable spring terminal block	
Weight		55 g (1.94 oz)	

Power LED

The following diagram shows a TMC4PACK01 cartridge with its power LED labeled **PWR**:



LED	Color	Status	Description
PWR	Green	On	The cartridge is powered by the logic controller and the external power supply (24 Vdc) is applied.
		Flashing	The cartridge is powered by the logic controller but the external power supply (24 Vdc) is not applied.
		Off	The cartridge is not powered by the logic controller.

TMC4PACK01 Characteristics

Introduction

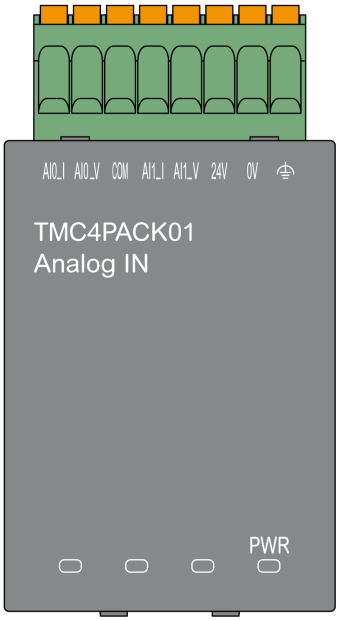
This section provides a general description of the TMC4PACK01 cartridge characteristics.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not exceed any of the rated values specified in the environmental and electrical characteristics tables. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: For important safety information and the environment characteristics of the TMC4 cartridges, see the M241 Logic Controller Hardware Guide.

Connectors

The following diagram shows a TMC4PACK01 cartridge marking and connectors:



Input Characteristics

The following table describes the cartridge input characteristics:

Characteristics		Value	
	Signal Type	Voltage	Current
Rated input range		0...10 Vdc	0...20 mA 4...20 mA
Input impedance		> 1 MΩ	< 250 Ω
Sample duration time		1 ms per enabled channel	
Input type		single-ended	
Operating mode		self-scan	
Conversion mode		SAR type	
Maximum accuracy at ambient temperature: 25 °C (77 °F)		± 0.2 % of full scale	
Maximum accuracy on full operating temperature range		± 0.5 % of full scale	
Temperature drift		± 0.006 % of full scale per 1 °C (1.8 °F)	
Repeatability after stabilization time		± 0.2 % of full scale	
Non-linearity		± 0.05 % of full scale	
Digital resolution		12 bits (4096 steps)	
Input value of LSB		2.44 mV	4.88 µV
Data type in application program		scalable from –32768 to 32767	
Input data out of detection range		yes	
Noise resistance	maximum temporary deviation during perturbations	± 2.0 % of full scale	
	cable type and maximum length	shielded	
		< 30 m (98.4 ft)	
	crosstalk (minimum)	80 dB	
	common-mode rejection ratio (minimum)	65 dB	
Isolation	isolation between inputs and internal logic	500 Vdc	
	isolation between inputs	not isolated	
Maximum continuous overload allowed (without damage)		30 Vdc	40 mA dc
Input filter		software filter: 6 levels	
External power supply	supply voltage	24 Vdc ± 15 %	
	power consumption	2 W	

TMC4PACK01 Wiring Diagram

Introduction

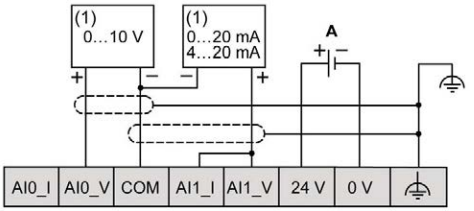
This cartridge has a removable spring terminal block for the connection of the inputs.

Wiring

See Wiring Best Practices *(see page 32)*.

Wiring Diagram

The following figure shows an example of the voltage and current input connection:



(1): Current/Voltage analog output device
A: External power supply

NOTE: Each input can be connected to either a voltage or current input.

Glossary

M

Modbus

The protocol that allows communications between many devices connected to the same network.

P

PE

(*Protective Earth*) A common grounding connection to help avoid the hazard of electric shock by keeping any exposed conductive surface of a device at earth potential. To avoid possible voltage drop, no current is allowed to flow in this conductor (also referred to as *protective ground* in North America or as an equipment grounding conductor in the US national electrical code).

Index

C

cartridge
- compatibility, *16*
- description, *15*
- features, *15*
- TMC4, *39*, *61*
- TMC4AI2, *41*
- TMC4AQ2, *55*
- TMC4HOIS01, *63*
- TMC4PACK01, *69*
- TMC4TI2, *47*

certifications and standards, *20*
compatibility
- cartridge, *16*

D

description
- cartridge, *15*

E

environment, *19*

F

features
- cartridge, *15*

G

Grounding, *35*

I

intended use, *6*

Q

qualification of personnel, *6*

T

TMC4
- cartridge, *39*, *61*

TMC4AI2
- cartridge, *41*

TMC4AQ2
- cartridge, *55*

TMC4HOIS01
- cartridge, *63*

TMC4PACK01
- cartridge, *69*

TMC4TI2
- cartridge, *47*

W

wiring, *32*