Modicon M241 Logic Controller

Programming Guide

Original instructions

EIO0000003059.08
12/2024

Legal Information

The information provided in this document contains general descriptions, technical characteristics and/or recommendations related to products/solutions.

This document is not intended as a substitute for a detailed study or operational and site-specific development or schematic plan. It is not to be used for determining suitability or reliability of the products/solutions for specific user applications. It is the duty of any such user to perform or have any professional expert of its choice (integrator, specifier or the like) perform the appropriate and comprehensive risk analysis, evaluation and testing of the products/solutions with respect to the relevant specific application or use thereof.

The Schneider Electric brand and any trademarks of Schneider Electric SE and its subsidiaries referred to in this document are the property of Schneider Electric SE or its subsidiaries. All other brands may be trademarks of their respective owner.

This document and its content are protected under applicable copyright laws and provided for informative use only. No part of this document may be reproduced or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), for any purpose, without the prior written permission of Schneider Electric.

Schneider Electric does not grant any right or license for commercial use of the document or its content, except for a non-exclusive and personal license to consult it on an "as is" basis.

Schneider Electric reserves the right to make changes or updates with respect to or in the content of this document or the format thereof, at any time without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this document, as well as any non-intended use or misuse of the content thereof.

Table of Contents

Safety Information 7
About the Book....8
About the Modicon M241 Logic Controller 13
M241 Logic Controller Description....13
How to Configure the Controller 18
How to Configure the Controller 18
Libraries 20
Libraries....20
Supported Standard Data Types 21
Supported Standard Data Types 21
Memory Mapping....22
Controller Memory Organization....22
RAM Memory Organization....23
Non-Volatile Memory Organization 25
Relocation Table 28
Tasks....31
Maximum Number of Tasks 31
Task Configuration Screen....31
Task Types 33
System and Task Watchdogs 35
Task Priorities 36
Default Task Configuration....38
Controller States and Behaviors 39
Controller State Diagram 39
Controller States Description 42
State Transitions and System Events 45
Controller States and Output Behavior 46
Commanding State Transitions 48
Error Detection, Types, and Management 55
Remanent Variables 56
Controller Device Editor 58
Controller Parameters 58
Communication Settings....60
PLC Settings 61
Services....62
Ethernet Services 63
Users Rights....64
Embedded Inputs and Outputs Configuration....74
Embedded I/Os Configuration....74
Expert Functions Configuration....79
Expert Functions Overview 79
Counting Function....81
Pulse Generators Embedded Function 83
Cartridge Configuration 85
TMC4 Cartridge Configuration 85
Expansion Modules Configuration....86
TM4/TM3/TM2 Expansion Modules Configuration 86
TM3 I/O Configuration General Description....87

TM3 I/O Bus Configuration 91
Optional I/O Expansion Modules 92
Ethernet Configuration 95
Ethernet Features, Functions and Services 95
Presentation 95
IP Address Configuration 97
Modbus TCP Client/Server 101
FTP Server 102
FTP Client 104
SNMP 105
Controller as a Target Device on EtherNet/IP 105
Controller as a Slave Device on Modbus TCP 122
Changing the Modbus TCP Port 126
Web Server 128
Monitoring Menu 132
Diagnostic Menu 136
Maintenance Menu 139
Firewall Configuration 143
Introduction 143
Dynamic Changes Procedure 144
Firewall Behavior 145
Firewall Script Commands 146
Used Ports 149
Industrial Ethernet Manager 150
Industrial Ethernet 150
DHCP Server 154
Fast Device Replacement 154
Serial Line Configuration 155
Serial Line Configuration 155
Network Manager 156
Modbus Manager 157
ASCII Manager 160
Modbus Serial IOScanner 162
Adding a Device on the Modbus Serial IOScanner 163
ControlChannel: Enables or Disables a Communication Channel 169
Adding a Modem to a Manager 170
CANopen Configuration 171
CANopen Interface Configuration 171
J1939 Configuration 174
J1939 Interface Configuration 174
OPC UA Server Configuration 178
OPC UA Server Overview 178
OPC UA Server Configuration 179
OPC UA Server Symbols Configuration 183
OPC UA Server Performance 185
Post Configuration 188
Post Configuration Presentation 188
Post Configuration File Management 189
Post Configuration Example 191
Connecting a Modicon M241 Logic Controller to a PC 193

Connecting the Controller to a PC 193
SD Card 196
Script Files 196
SD Card Commands 196
Firmware Management 202
Updating Modicon M241 Logic Controller Firmware 202
Updating TM3 Expansion Modules Firmware 204
Compatibility 207
Software and Firmware Compatibilities 207
Appendices 209
How to Change the IP Address of the Controller 210
changeIPAddress: Change the IP Address of the Controller 210
Functions to Get/Set Serial Line Configuration in User Program 212
GetSerialConf: Get the Serial Line Configuration 212
SetSerialConf: Change the Serial Line Configuration 213
LinkNumber: Communication Port Number 214
SERIAL_CONF: Structure of the Serial Line Configuration Data Type 215
Controller Performance 216
Processing Performance 216
Glossary 217
Index 227

Safety Information

Important Information

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

Document Scope

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

NOTE: Read and understand this document and all related documents, page 8 before installing, operating, or maintaining your Modicon M241 Logic Controller.

Validity Note

For more information on the validity of the present document, see the online help for the product.

Available Languages of this Document

This document is available in these languages:

- English (EIO0000003059)
- French (EIO0000003060)
- German (EIO0000003061)
- Spanish (EIO0000003062)
- Italian (EIO0000003063)
- Chinese (EIO0000003064)

Related Documents

Title of Documentation	Reference Number
EcoStruxure Automation Expert - Motion, 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
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert - Industrial Ethernet Overview - User Guide	EIO0000003053 (ENG) EIO0000003054 (FRE) EIO0000003055 (GER) EIO0000003056 (SPA) EIO0000003057 (ITA) EIO0000003058 (CHS)
Modicon TM2 Expansion Modules Configuration - Programming Guide	EIO0000003432 (ENG) EIO0000003433 (FRE) EIO0000003434 (GER) EIO0000003435 (SPA) EIO0000003436 (ITA) EIO0000003437 (CHS)
Modicon TM3 Expansion Modules - Programming Guide	EIO0000003119 (ENG) EIO0000003120 (FRE) EIO0000003121 (GER) EIO0000003122 (SPA) EIO0000003123 (ITA) EIO0000003124 (CHS)
Modicon TM3 Bus Coupler - Programming Guide	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)

Title of Documentation	Reference Number
Modicon M241 Logic Controller - HSC Library Guide	EIO0000003071 (ENG) EIO0000003072 (FRE) EIO0000003073 (GER) EIO0000003074 (SPA) EIO0000003075 (ITA) EIO0000003076 (CHS)
Modicon TM3 Expert I/O Modules - HSC Library Guide	EIO0000003683 (ENG) EIO0000003684 (FRE) EIO0000003685 (GER) EIO0000003686 (SPA) EIO0000003687 (ITA) EIO0000003688 (CHS)
Modicon M241 Logic Controller PTO/PWM - Library Guide	EIO0000003077 (ENG) EIO0000003078 (FRE) EIO0000003079 (GER) EIO0000003080 (SPA) EIO0000003081 (ITA) EIO0000003082 (CHS)
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert - FtpRemoteFileHandling Library Guide	EIO0000002779 (ENG) EIO0000002780 (FRE) EIO0000002781 (GER) EIO0000002783 (SPA) EIO0000002782 (ITA) EIO0000002784 (CHS)
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert - SnmpManager Library Guide	EIO0000002797 (ENG) EIO0000002798 (FRE) EIO0000002799 (GER) EIO0000002801 (SPA) EIO0000002800 (ITA) EIO0000002802 (CHS)
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert - Manage a Cyclic Task Interval - Toolbox_Advance Library Guide	EIO0000000946 (ENG) EIO0000000947 (FRE) EIO0000000948 (GER) EIO0000000950 (SPA) EIO0000000949 (ITA) EIO0000000951 (CHS)
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert - Modem Functions - Modem Library Guide	EIO0000000552 (ENG)

To find documents online, visit the Schneider Electric download center (www.se.com/ww/en/download/).

Product Related Information

⚠ WARNING

LOSS OF CONTROL

- Perform a Failure Mode and Effects Analysis (FMEA), or equivalent risk analysis, of your application, and apply preventive and detective controls before implementation.
- Provide a fallback state for undesired control events or sequences.
- Provide separate or redundant control paths wherever required.
- Supply appropriate parameters, particularly for limits.
- Review the implications of transmission delays and take actions to mitigate them.
- Review the implications of communication link interruptions and take actions to mitigate them.
- Provide independent paths for control functions (for example, emergency stop, over-limit conditions, and error conditions) according to your risk assessment, and applicable codes and regulations.
- Apply local accident prevention and safety regulations and guidelines.[1]
- Test each implementation of a system for proper operation before placing it 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.

Information on Non-Inclusive or Insensitive Terminology

As a responsible, inclusive company, Schneider Electric is constantly updating its communications and products that contain non-inclusive or insensitive terminology. However, despite these efforts, our content may still contain terms that are deemed inappropriate by some customers.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in the information contained herein, 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 13849-1:2023	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2020	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 62061:2021	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 61784-3:2021	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.

About the Modicon M241 Logic Controller

Introduction

This chapter provides information about the Modicon M241 Logic Controller and devices that EcoStruxure Automation Expert - Motion and EcoStruxure Machine Expert can configure and program.

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 Automation Expert - Motion and EcoStruxure Machine Expert software described in detail in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide and in the M241 Logic Controller Programming Guide, page 8.

Programming Languages

The M241 Logic Controller is configured and programmed with the EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert software. They support 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 Automation Expert - Motion and EcoStruxure Machine Expert software can also be used to program the M241 Logic Controller 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, page 74.
- A software command.
- The system variable PLC_W in a Relocation Table
- The Web server

Memory

This table describes the different types of memory:

Memory Type	Size	Used
RAM	64 Mbytes	To execute the application.
Non-volatile	128 Mbytes	To 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 Controller includes an embedded SD card slot.

The main uses of the SD card are:

- Initializing the controller with a new application
- Updating the controller and expansion module firmware, page 202
- Applying post configuration files to the controller, page 188
- Storing recipes files
- Receiving data logging files
- Backup Data Logging File, page 28

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 Compatibility and Migration User Guide.

M241 Logic Controller References

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

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
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

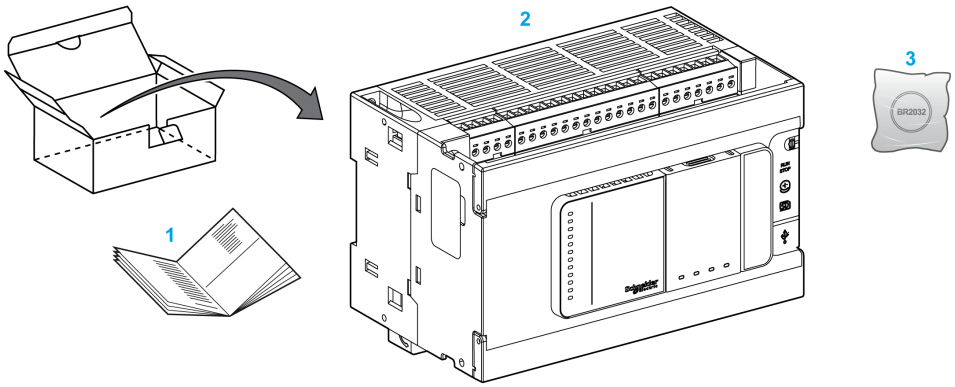
(1) The regular inputs have a maximum frequency of 1 kHz.

(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.

How to Configure the Controller

Introduction

This chapter shows the default configuration of a project.

How to Configure the Controller

Introduction

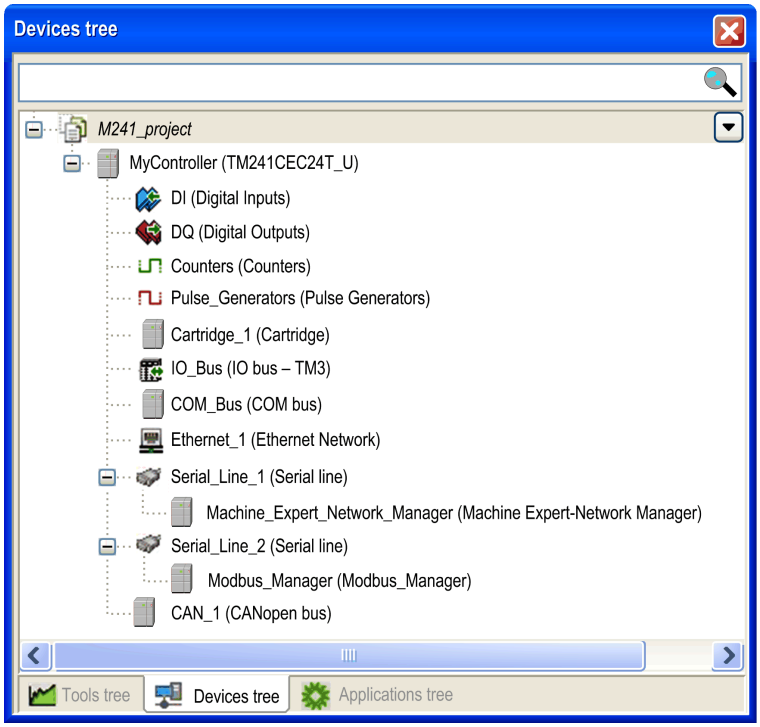
First, create a new project or open an existing project in the EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert software.

Refer to the EcoStruxure Automation Expert - Motion / 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 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 inserted 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.

Libraries

Introduction

This chapter describes the default libraries of the Modicon M241 Logic Controller.

Libraries

Overview

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

The **Library Manager** 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. For more information about the libraries compatible with your controller, refer to the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Libraries Overview.

Modicon M241 Logic Controller

When you select a Modicon M241 Logic Controller for your application, EcoStruxure Automation Expert - Motion and EcoStruxure Machine Expert automatically load 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	**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. See Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide.
M241 PLCSystem	Contains functions and variables to get information and send commands to the controller system. See Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide.
M241 HSC	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. See Modicon M241 Logic Controller, High Speed Counting, HSC Library Guide.
M241 PTOPWM	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. See Modicon M241 Logic Controller, PTOPWM, Library Guide.
Relocation Table	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. See Relocation Table, page 28.

Supported Standard Data Types

Introduction

This chapter provides the different IEC data types supported by the controller.

Supported Standard Data Types

Overview

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	–	1 character = 1 byte
WSTRING	1 character	–	1 character = 1 word
TIME	0	4294967295	32 Bit

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

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.

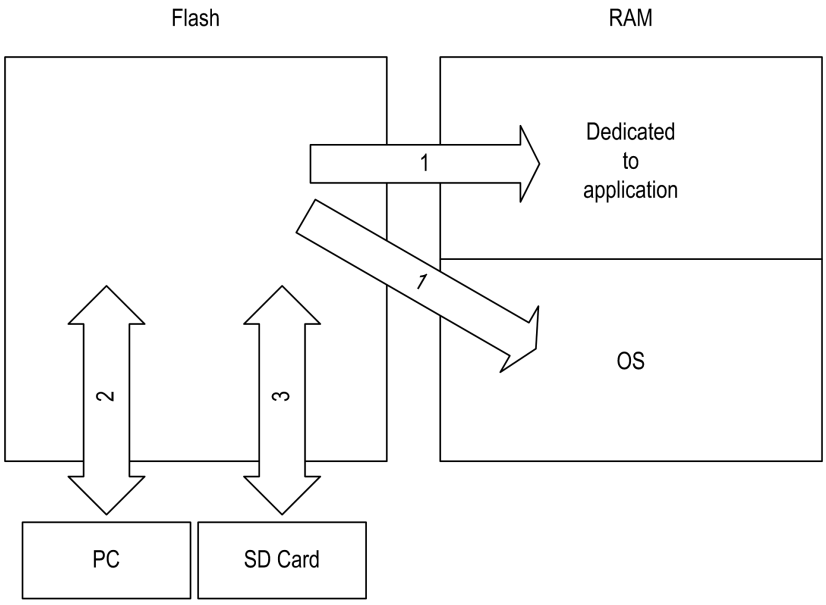
Controller Memory Organization

Introduction

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

- The non-volatile memory, page 25 contains files (application, configuration files, and so on).
- The Random Access Memory (RAM), page 23 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 non-volatile 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, page 128 • FTP server, page 102 • **Controller Assistant** • The software (see the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide)
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 non-volatile memory does not affect a running application. Any changes to files in non-volatile 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
System area	System Area Mappable Addresses %MW0...%MW59999
	System and diagnostic variables, page 24 (%MW60000...%MW60199) This memory is accessible through Modbus requests only. These must be read-only requests.
	Dynamic Memory Area: Read Relocation Table, page 28 (%MW60200...%MW61999) This memory is accessible through Modbus requests only. These must be read-only requests.
	System and diagnostic variables, page 24 (%MW62000...%MW62199) This memory is accessible through Modbus requests only. These can be read or write requests.
	Dynamic Memory Area: Write Relocation Table, page 28 (%MW62200...%MW63999) This memory is accessible through Modbus requests only. These can be read or write requests.
	%MW64000…%MW65535 Reserved
	Retain and Persistent data, page 26
User area	Symbols
	Variables
	Application
	Libraries

To display the memory mapping in EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert, right-click on your controller in the **Devices tree** window and select **Device Memory Info**.

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 Modicon M241 Logic Controller, System Functions and Variables, 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 %MB1, %MW1 contains %MB2 and %MB3.

NOTE: The Modbus communication is asynchronous with the application.

Non-Volatile Memory Organization

Introduction

The non-volatile 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 non-volatile 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 non-volatile 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 non-volatile 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 Automation Expert - Motion and EcoStruxure Machine Expert, according to application properties. Default option is to create the boot application on download. When you download an application from the software to the controller, you are transferring only the binary executable application directly to RAM.

(2): EcoStruxure Automation Expert - Motion and EcoStruxure Machine Expert do 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 non-volatile memory.

File Organization

This table shows the file organization of the non-volatile 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, page 188	Configuration
		CodesysLateConf.cfg (2)	• Name of application to launch • Routing table (main/sub net)	Configuration
/usr	Log	*UserDefinedLogName_1*.log	All *.log files created using the data logging functions (see 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 Automation Expert - Motion and EcoStruxure Machine Expert by viewing the **Log** tab of the **Controller Device Editor**, page 58.	–
		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, page 154
/data	–	–	Retained and persistent data	–
/sd0	–	–	SD card. Removable	–
	–	User files	–	–

(1): v_XX.YY represents the version

(2): if any

(3): if OPC UA, page 179 is configured

(4): the Fdr/FDRS directory is hidden

NOTE: For more information on libraries and available function blocks, refer to Libraries, page 20.

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 non-volatile memory.

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

File extensions	Non-volatile 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 *Modicon M241 Logic Controller PLCSystem – Library Guide*.

Adding a Relocation Table

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

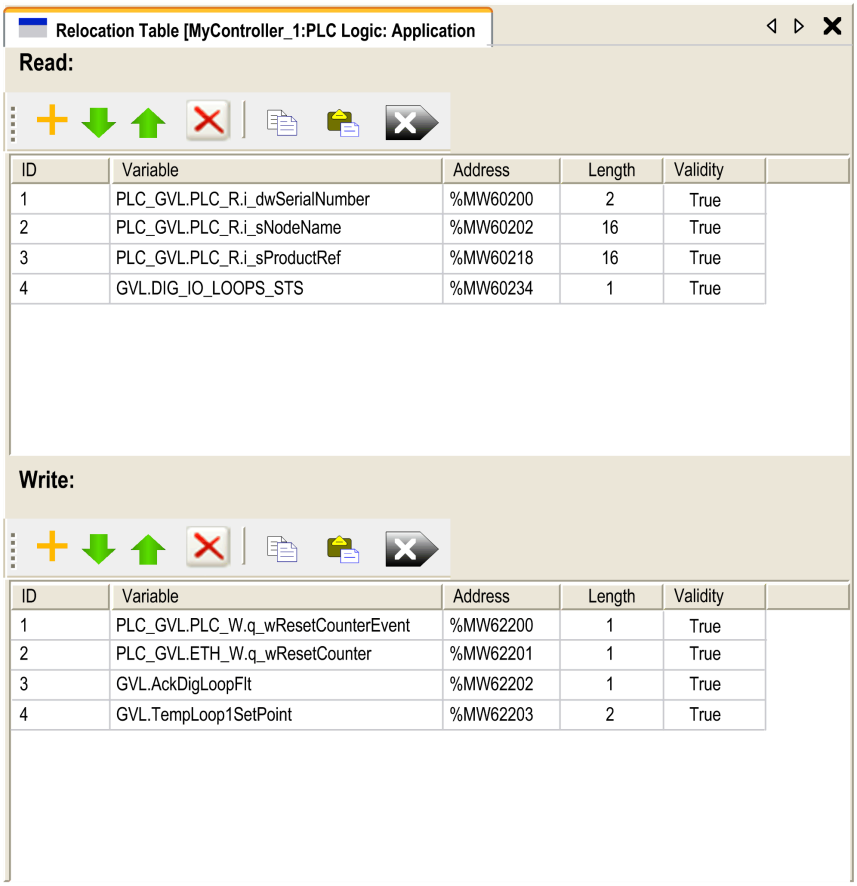
Step	Action
1	In the **Applications tree** tab, select the **Application** node.
2	Click the right mouse button.
3	Click **Objects** > **Relocation Table...**. **Result:** The **Add Relocation Table** window is displayed.
4	Click **Add**. **Result:** The new relocation table is created and initialized. **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:



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.

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.

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 = 16

Special Considerations for Freewheeling

A Freewheeling task, page 34 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, page 35.

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.

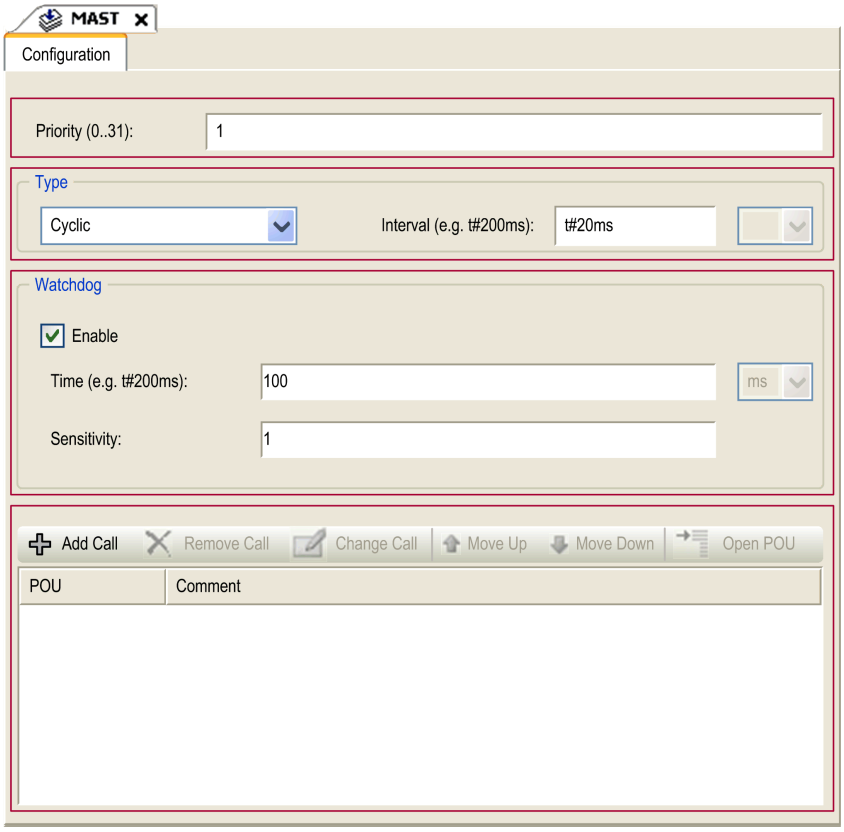
Task Configuration Screen

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 runs: a higher priority task pre-empts a lower priority task. **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 unpredictable. For important information, refer to Task Priorities, page 36.
Type	These task types are available: • Cyclic, page 33 • Event, page 34 • External , page 35 • Freewheeling, page 34
Watchdog	To configure the watchdog, page 36, 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 (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 selected POU of the list by another one, use the command **Change Call**. • POUs are executed in the order shown in the list. To move the POUs 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.

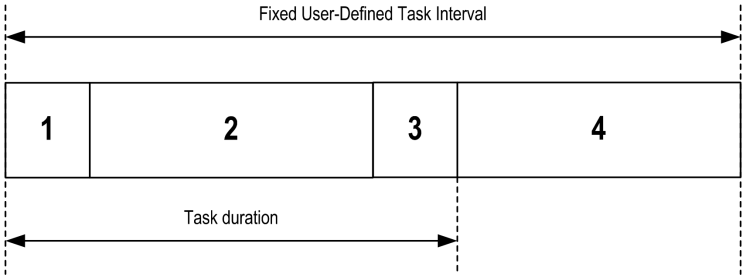
Task Types

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 the 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 the 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 Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide and **PLC Settings**, page 61. For more information on I/O behavior, refer to Controller States Detailed Description, page 42.
4.	**Remaining Interval time:** The controller firmware carries out system processing and other lower priority tasks.

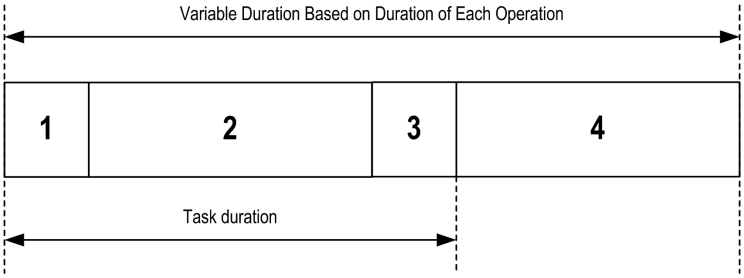
NOTE: If you define too short a period for a cyclic task, it will repeat immediately after the write of the outputs and without executing other lower priority tasks or any system processing. This will affect the execution of all tasks and cause the controller to exceed the system watchdog limits, generating 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. For further details, refer to Functions Descriptions (see Toolbox_Advance Library Guide).

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 the 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 Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide and **PLC Settings**, page 61. For more information on I/O behavior, refer to Controller States Detailed Description, page 42.
4.	**System Processing:** The controller firmware carries out system processing and 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, page 33.

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 an excessive 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 managed by the controller firmware. You cannot configure them.
- **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 Automation Expert - Motion and 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 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: You cannot configure the system watchdogs.

Task Watchdogs

EcoStruxure Automation Expert - Motion and EcoStruxure Machine Expert allow 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 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 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 watchdog settings found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.

Task Priorities

Task Priority Configuration

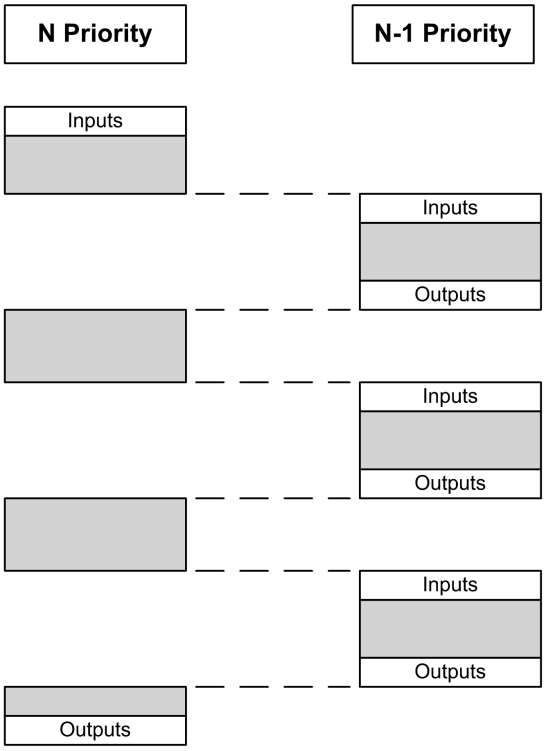
You can configure the priority of each task between 0 and 31 (0 is the highest priority, 31 is the lowest). Each task must have a unique priority. Assigning the same priority to more than one task leads to a build error.

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, page 91.

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

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, page 36 for more information on priority settings. Refer to Task Watchdogs, page 35 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, the software detects an error when you attempt to build the application, and you are not able to download it to the controller.

Controller States and Behaviors

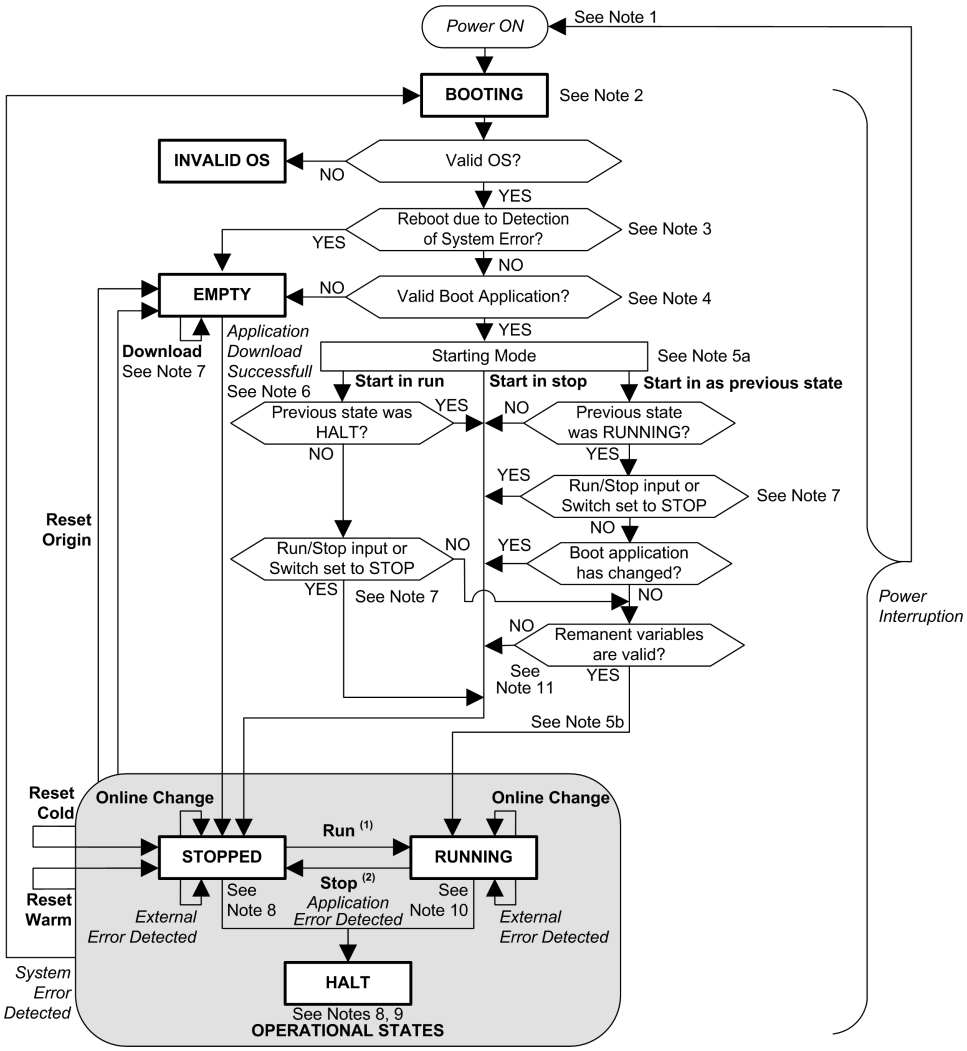
Introduction

This chapter provides 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 task programming options on the behavior of your system.

Controller State Diagram

Controller State Diagram

This diagram describes the controller operating mode:



ALL-CAPS BOLD: Controller states

Bold: User and application commands

***Italics*:** System events

Normal text: Decisions, decision results, and general information

(1) For details on STOPPED to RUNNING state transition, refer to Run Command, page 48.

(2) For details on RUNNING to STOPPED state transition, refer to Stop Command, page 48.

Note 1

The Power Cycle (Power Interruption followed by a Power ON) deletes all output forcing settings. Refer to Controller State and Output Behavior, page 46 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 non-volatile memory. However, the Boot application is not deleted from the non-volatile 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, page 188 file settings (if any) are applied.

During the load of the boot application, a Check context test occurs to verify that the remanent variables are valid. If the Check context test is invalid, the boot application will load but the controller will transitions to the STOPPED state, page 52.

Note 5a

The **Starting Mode** is set in the **PLC settings** tab of the **Controller Device Editor**, page 61.

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 non-volatile 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 two 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 Automation Expert - Motion and EcoStruxure Machine Expert have 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 single download, EcoStruxure Automation Expert - Motion and EcoStruxure Machine Expert do 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 Automation Expert - Motion and EcoStruxure Machine Expert software platform allow 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, page 42 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 power cycle is mandatory.

Note 10

The RUNNING state has 2 two 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, page 87.
- RUNNING with Breakpoint: this exception condition is indicated by the RUN LED, which displays a single green flash. Refer to Controller States Description, page 42 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.

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.

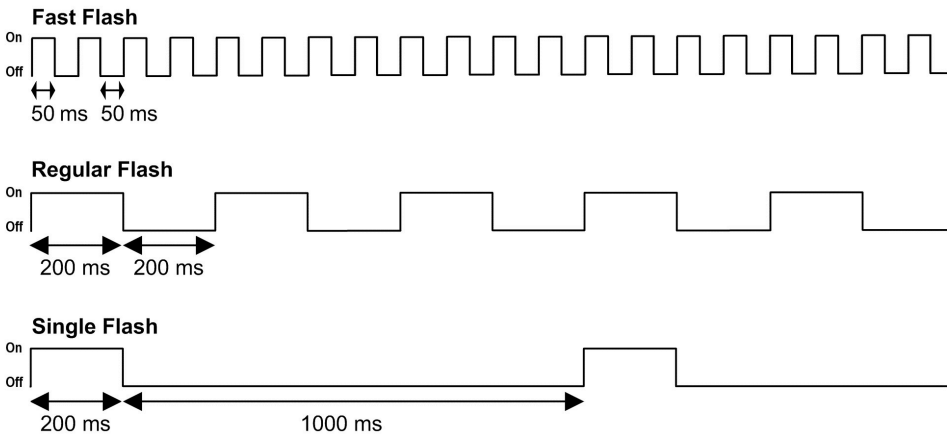
(1) The controller states can be read in the PLC_R.i_wStatus system variable of the M241 PLCSystem library (see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide).

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 non-volatile memory. The controller does not execute the application. Refer to Firmware Management, page 202 to restore a correct state.	OFF	Regular flash	OFF
EMPTY	The controller has no application.	OFF	Single flash	OFF
EMPTY after a system error detected	This state is the same as the other EMPTY state. However the application is present, and is intentionally not loaded. A reboot (power cycle), or a new application download, restores a 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 the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.	Single flash	OFF	OFF
RUNNING with external error detected	The controller is executing a valid application and a configuration, TM3, SD card, or other I/O error is 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, page 44 for an explanation of the behavior of outputs and field buses in this state.	Regular flash	OFF	OFF
STOPPED with external error detected	The controller is executing a valid application and a configuration, TM3, SD card, or other I/O error is 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 non-volatile memory. At next power cycle, the application will be changed by the one from non-volatile memory.	ON or regular flash	Single flash	OFF

This timing diagram shows the difference between the fast flash, 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. 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: PWM, FreqGen (frequency generator) and HSC reflex outputs are set to the configured default values, and PTO outputs are set to 0.
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 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 values. • If the **Behaviour for outputs in Stop** field is **Set all outputs to default** the last 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. 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: PWM, FreqGen (frequency generator) and HSC reflex outputs are set to the configured default values, and PTO outputs are set to 0.
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.

CANopen Behavior in HALT State

The following table describes the CANopen behavior in HALT state:

HALT state is caused by ProcessorLoadWatchdog	• The CANopen bus does not remain operational. • The CANopen Master ceases communications. Devices on the CANopen 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.
HALT state is caused by Task Watchdog **The task defined for the Bus cycle task of the CANopen_ Performance is in Exception**	
HALT state is caused by Task Watchdog **The task defined for the Bus cycle task of the CANopen_ Performance is NOT in Exception**	• The CANopen bus remains operational. Devices on the CANopen bus continue to detect the presence of a functional CANopen 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 values. • If the **Behaviour for outputs in Stop** field is set to **Set all outputs to default**, the last values are updated to the default values and subsequent TPDOs are issued with these default values.

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.

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.

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, page 87.

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 **Devices 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, page 45, except if the task defined for the Bus cycle task of the CANopen_Performance is in Exception, or during ProcessorLoadWatchdog.

Outputs maintain their 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 fieldbuses. Refer to Controller States Description, page 42 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 **Devices tree** and select **Edit Object**.

This output behavior applies:

- when the controller is going from RUNNING state to STOPPED state.
- if the controller is going from RUNNING 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 maintain their 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 fieldbuses. Refer to Controller States Description, page 42 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 Automation Expert - Motion or EcoStruxure Machine Expert.

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

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

When you logout of EcoStruxure Automation Expert - Motion or 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 controller is in STOPPED state.

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, page 76 for more information.
- Software **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 (see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide).
- **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, or EMPTY state.
- The controller is restarted into the RUNNING state automatically under certain conditions.

Refer to Controller State Diagram, page 39 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, page 76 for more information.
- Software **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 (see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide).
- **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, 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, page 52 for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram, page 39 for further details.

Reset Warm

Effect: Resets the 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:

- Software **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 (see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide).

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. The 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. The fieldbus communications are stopped and then restarted after the reset is complete.
10. The inputs are reset to their initialization values. The 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, page 188.

For details on variables, refer to Remanent Variables, page 56.

Reset Cold

Effect: Resets the 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:

- Software **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 (see Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library Guide).

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. The 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. The fieldbus communications are stopped and then restarted after the reset is complete.
10. The inputs are reset to their initialization values. The 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, page 188.

For details on variables, refer to Remanent Variables, page 56.

Reset Origin

Effect: Resets all variables, including the remanent variables, to their initialization values. Erases all user files on the controller, including user rights and certificates. Reboots and places the controller into the EMPTY state.

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Origin Command:

- Software **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. The user files (Boot application, data logging, Post Configuration, user rights and certificates) 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. The 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. The fieldbus communications are stopped.
12. Embedded Expert I/O are reset to their previous user-configured default values.
13. The other inputs are reset to their initialization values.

    The other outputs are reset to their hardware initialization values.
14. The controller reboots.

For details on variables, refer to Remanent Variables, page 56.

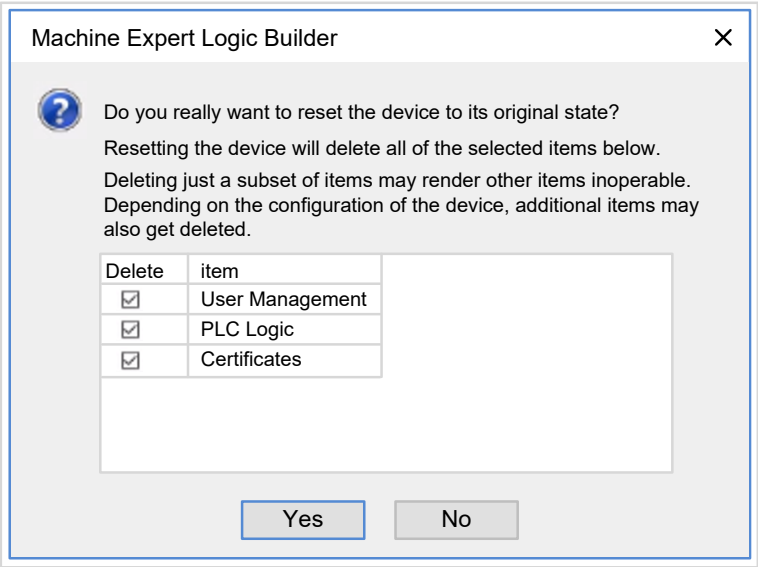
Reset Origin Device

Effect: Resets all variables, including the remanent variables, to their initialization values. Places the controller into the EMPTY state if **PLC Logic** is selected.

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Origin Device Command:

- In the software: Right-click **My controller > Reset Origin Device** command. **Result:** a dialog box allows you to select the items to remove:
  - **User Management**
  - **PLC Logic**
  - **Certificates**



When **User Management** is selected, user and groups are reset to default value.

NOTE: If the controller **User rights** are disabled before this command is used, you can connect to the controller without login prompt afterwards. Use the dedicated command in Online menu: **Security > Reset user rights management to default** to enforce again the use of user management.

When **PLC Logic** is selected:

1. The application stops.
2. Forcing is erased.
3. The WebVisualisation files are erased.
4. Diagnostic indications for errors are reset.
5. The values of the retain variables are reset.
6. The values of the retain-persistent variables are reset.
7. The non-located and non-remanent variables are reset.
8. The fieldbus communications are stopped.
9. Embedded Expert I/O are reset to their previous user-configured default values.
10. The other inputs are reset to their initialization values.

    The other outputs are reset to their hardware initialization values.
11. System Logs are maintained.

When **Certificates** is selected, certificates used for Web server and FTP server are reset.

For details on variables, refer to Remanent Variables, page 56.

Reboot

Effect: Commands a reboot of the controller.

Starting Conditions: Any state.

Methods for Issuing the Reboot Command:

- Power cycle
- REBOOT by Script, page 196

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 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 and the remanent variables are valid.

      The Reboot was provoked by a script and:

      - the **Starting Mode** is set to **Start in run**, and if the Run/Stop input is configured and set to RUN, or the switch is 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 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 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 is configured and set to RUN, or the switch is 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 is not configured, and if the controller was not in HALT, or the switch is set to run 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. The 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. The fieldbus communications are stopped and restarted after the boot application is loaded successfully.
10. The inputs are reset to their initialization values. The 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, page 188.
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, page 56.

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 non-volatile memory.

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

Methods for Issuing the Download Application Command:

- In the software: 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 non-volatile 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. The updated file is applied at the next reboot. Refer to File Transfer with SD Card, page 201 for further details.

Effects of the 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. The 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. The fieldbus communications are stopped and then the 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. The inputs are reset to their initialization values. The 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, page 188.

For details on variables, refer to Remanent Variables, page 56.

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, page 52.

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 non-volatile 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, page 87.		

NOTE: Refer to the Modicon M241 Logic Controler 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	–	–	–
Reset origin device	–	–	–
Download of application program using EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert (3)	–	–	X
Download of application program using an SD card (3)	–	–	–

(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, page 49.

(3) 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**) variables 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.

Controller Device Editor

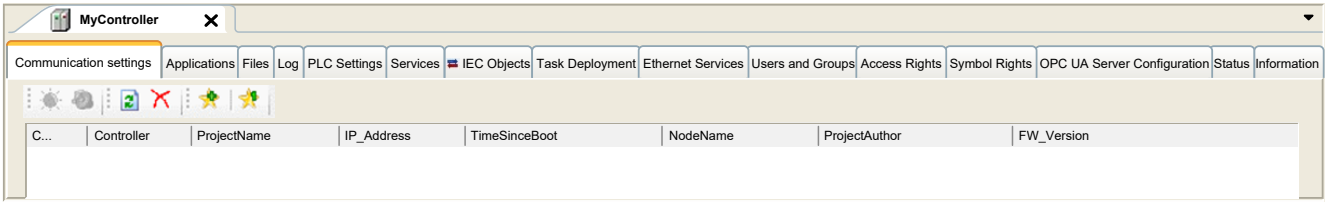
Introduction

This chapter describes how to configure the controller.

Controller Parameters

Controller Parameters

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



Tabs Description

Tab	Description	Restriction
Communication Settings, page 60	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 Communication Settings in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.	Online mode only
Applications	Presents the application running on the controller and allows removing the application from the controller.	Online mode only
Files, page 25	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 non-volatile memory of the controller.	Online mode only
Log	View the controller log file.	Online mode only
PLC settings, page 61	Configuration of: • application name • I/O behavior in stop • bus cycle options	–
Services, page 62	Lets you configure the online services of the controller (RTC, device identification).	Online mode only
IEC Objects	Allows you to access to the device from the IEC application through the listed objects. Displays a monitoring view in online mode. For more information, refer to IEC Object in CoDeSys Online Help.	–
Task deployment	Displays a list of I/Os and their assignments to tasks.	After compilation only
Ethernet Services	The **IP Routing** tab allows you to configure the routes and the cross network transparency through **IP Routing** options. **NOTE:** This tab is empty if no Ethernet connection is available in the configuration.	–
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 projects and devices in online mode. For more details, refer to the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.	–
Access Rights	The **Access Rights** tab allows you to define the device access rights of users. For more details, refer to the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.	–
Symbol Rights	Allows the **Administrator** to configure **Users and Groups** access to the symbol sets. For more information, refer to Symbol Configuration in CoDeSys Online Help.	–
OPC UA Server Configuration	Displays the OPC UA Server Configuration, page 179 window.	–
Status	Not used.	–
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 the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.
- **Classic mode**: refer to the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.
- **Controller selection mode**: refer to the EcoStruxure Automation Expert - Motion / 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 contextual 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:

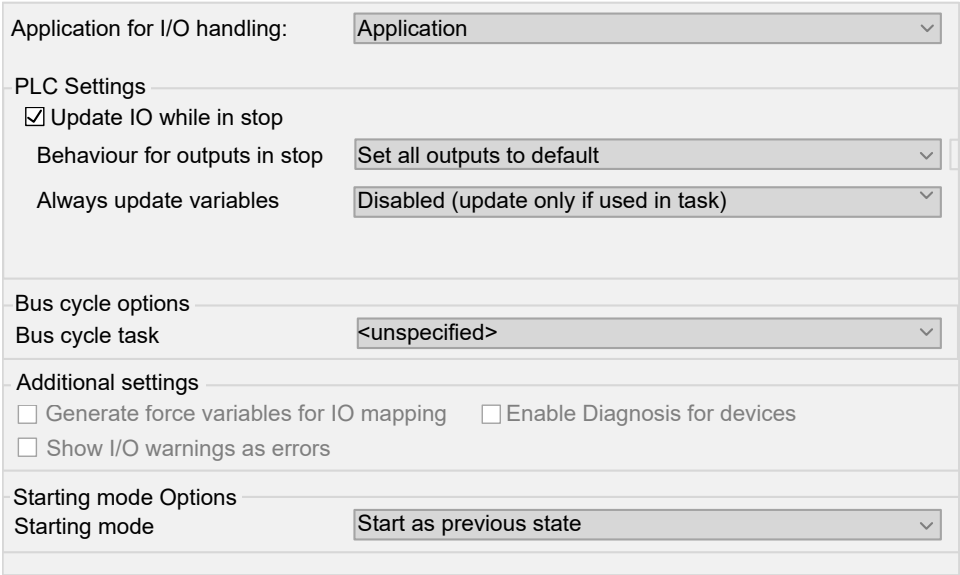
  You can also activate the **Save settings permanently** option before you click **OK**. Once this option is activated, the Ethernet parameters configured here are always taken into account on reset instead of the Ethernet parameters configured into the application.

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

PLC Settings

Overview

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



Element		Description
Application for I/O handling		Select **Application** (as there is only one application in the controller). **NOTE:** If **None** is selected, the application will not be built.
PLC settings	**Update IO while in stop**	If this option is activated (default), the values of the input and output channels are 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	From the selection list, choose one of the following options: • **Disabled (update only if used in task)** • **Enabled 1 (use bus cycle task if not used in any task)** • **Enabled 2 (always in bus cycle task)**
Bus cycle options	**Bus cycle task**	This configuration setting is the parent for all **Bus cycle task** parameters used in the application **Devices 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, page 39. Select with this option one of these starting modes: • **Start as previous state** • **Start in stop** • **Start in run**

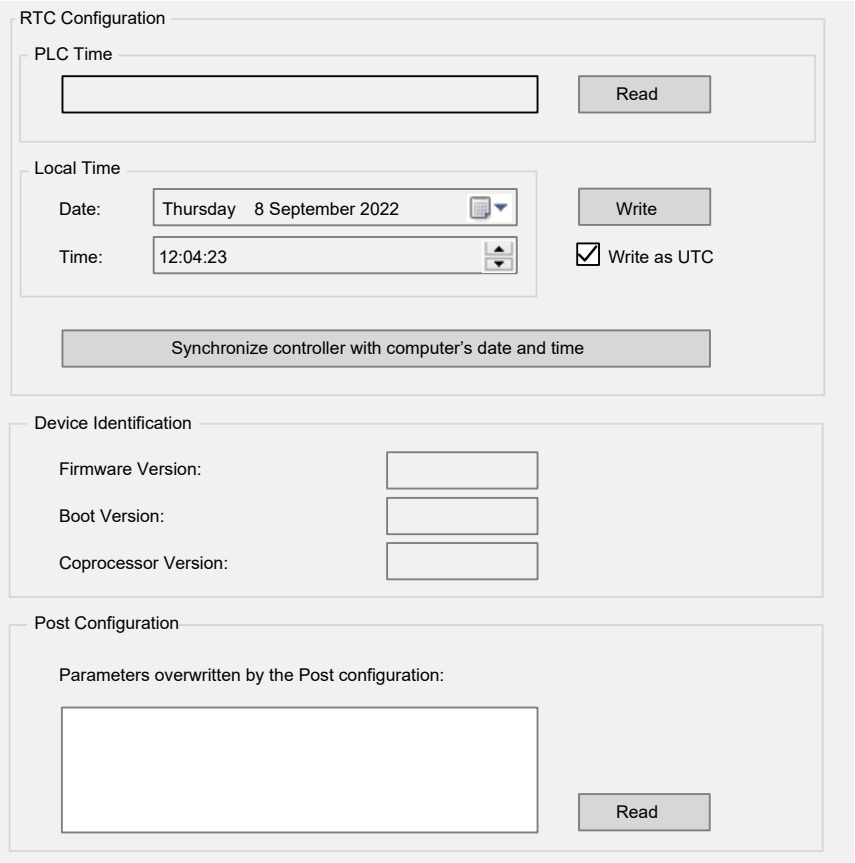
Services

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 you click the **Read** button, with no conversion applied. This read-only field is initially empty.
	Read	Reads the date and time saved on the controller and displays the values in the **PLC Time** field.
	Local Time	Defines a date and time to send to the controller when you click the **Write** button. 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 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 if you want to write the values in UTC format.
	Synchronize controller with computer's date/ time	Sends the PC date and time. A message box informs you of the result of the command. Select **Write as UTC** before running this command if you want to use UTC format.
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, page 188.

Ethernet Services

IP Routing

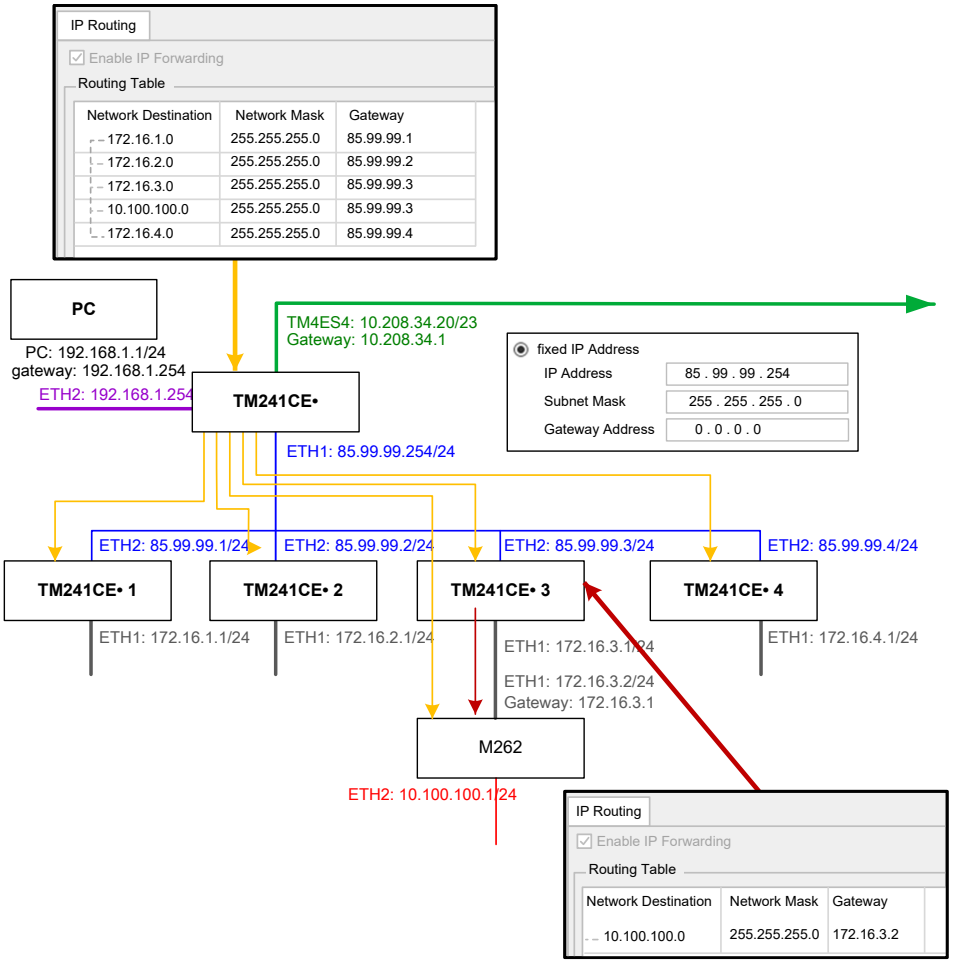
The **IP Routing** subtab allows you to configure the IP routes in the controller.

The parameter **Enable IP forwarding** recalls the options set or not on the configuration page of the TM4ES4 Ethernet module (option not available on the embedded Ethernet port).

When deactivated, the communication is not routed from a network to another one. The devices on the device network are no longer accessible from the control network and related features like Web pages access on device or commissioning of device via DTM, EcoStruxure Machine Expert - Safety and so on are not possible.

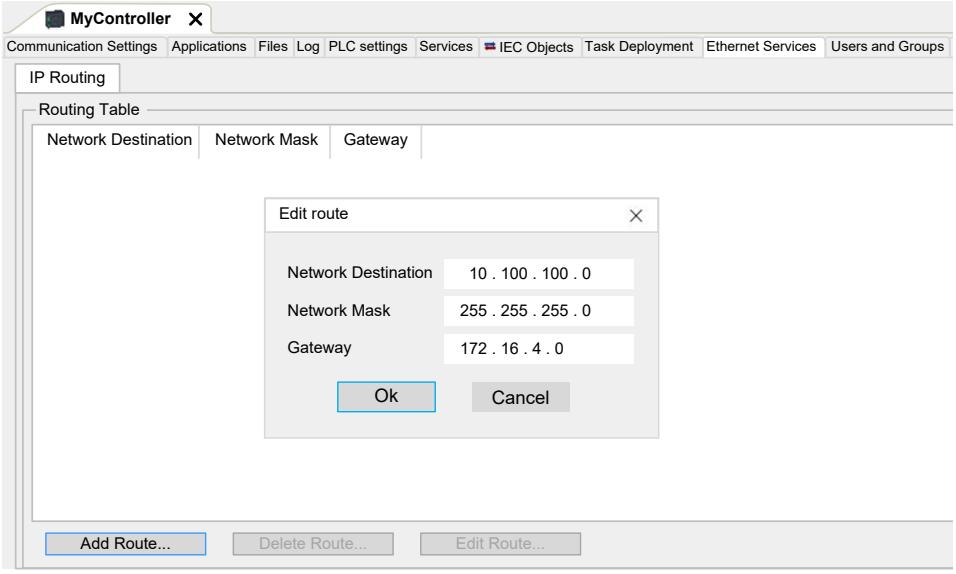
The M241 Logic Controller can have up to two Ethernet interfaces. Using a routing table is necessary to communicate with remote networks connected to different Ethernet interfaces. The gateway is the IP address used to connect to the remote network, which needs to be in local network of the controller.

This graphic depicts an example network, in which the last two rows of devices (gray and red) need to be added in the routing table:



Use the routing tables to manage the IP forwarding.

To add a route, double click **My controller** then click **Ethernet Services > IP Routing > Add Route**.



For reasons of network security, TCP/IP forwarding is disabled by default. Therefore, you must manually enable TCP/IP forwarding if you want to access devices through the controller. However, doing so may expose your network to possible cyberattacks if you do not take additional measures to protect your enterprise. In addition, you may be subject to laws and regulations concerning cybersecurity.

⚠WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT NETWORK INTRUSION

- Observe and respect any an all pertinent national, regional and local cybersecurity and/or personal data laws and regulations when enabling TCP/IP forwarding on an industrial network.
- 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.

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

Users Rights

Introduction

Users rights contain the following elements: **User**, **Group**, **Object**, **Operation**, **User Rights**, **Access rights**. These elements allow you to manage users accounts and users access rights to control the access on the global projects.

- A **User** is a person or a service with specific **User Rights**.
- A **Group** is a **Persona** or a **Function**. It is predefined or added. Each **Group** provides accesses thanks to **Object**.
- An **Object** is composed by predefined accesses thanks to **Operation**.
- An **Operation** is the elementary action possible.
- **User Rights** are the possible **Access rights**: **VIEW**, **MODIFY**, **EXECUTE** and **ADD–REMOVE** for the dedicated operation.

For more informations, refer to the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.

Login and Passwords

Login and password are not set by default. This table describes how to log in:

Server/feature	First connection or connection after reset to default / reset origin / reset origin device	User Rights enabled	Connection after User Rights disabled
EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert	You must first create your login and your password. **NOTE:** The login and the password that you create during the first connection have administrator privileges. **NOTE:** For information on lost login and passwords, see Troubleshooting, page 72.	**Login**: configured login **Password**: configured password	No login or password required.
Web server	No login possible	**Login**: configured login **Password**: configured password	**Login**: Anonymous **Password**: Anonymous or no password.
FTP server	No login possible	**Login**: configured login **Password**: configured password	**Login**: Anonymous **Password**: Anonymous
OPC-UA	No login possible	**Login**: configured login **Password**: configured password	**Login**: Anonymous **Password**: Anonymous
Change Device Name feature	No login possible	**Login**: configured login **Password**: configured password	No login or password required.

⚠ 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 disabling the user rights on the **User accounts management** subpage of the Web server, page 139.

NOTE: The following characters are supported by the controller:

- login: **a...z A...Z 0...9 – = [ ] \ ; ‘ , . / @ # $ % ^ & * ( ) _ + { } | : “ < > ? ` ~**
- password: **a...z A...Z 0...9 – = [ ] \ ; ‘ , . / @ # $ % ^ & * ( ) _ + { } | : “ < > ? ` ~** and **space**

The length is limited to 60 characters.

Default Users and Groups

This table indicates the name and description of the predefined default groups:

Group Name	Group Description
Administrator	• Manages all the user rights. • Is created at first connection.
Persona	
Persona Designer/ Programmer	Group dedicated to the design of the application.
Persona Operator	Group dedicated to the usage of the application.
Persona Web Designer	Group dedicated to the management of the Web server.
Persona Communication	Group dedicated to the management of communication features.
Persona Maintenance	Group dedicated to the maintenance of the application.
Function	
Function External Media	Group to allow the usage of External Command (from SD Card).
Function File Access	Group to allow permissions on files tab.
Function FTP	Group to allow usage of FTP.
Function Symbol Configuration	Group to allow access to **Symbol Configuration**.
Function Web Access	Group to allow command on Web server.
Function Monitor	Group to allow monitoring of IEC variables.
Function OPC UA	Group to allow access to OPC UA server.
Function Variable	Group to allow read/write of IEC variables.

NOTE: Administrator can define a new **Group** if needed.

Object Names

This table indicates the name and description of the predefined objects:

Object name	Object Description
Device	Object related to the connection of the controller through the software.
ExternalCmd	Object related to script command (**Clone** and **CloneCheck**).
FTP	Object related to FTP access (connection, upload and download on FTP server).
Logger	Object related to the message logger.
OPC_UA	Object related to OPC UA server (connection, read and write variables).
PlcLogic	Object related to the application on the controller.
Settings	Object related to the settings of the controller (nodename…).
UserManagement	Object related to User rights Management.
Web	Object related to the access of the Web server.
FileSystem	Object related to the file access (when accessing through the controller Files tab).

Operation Functions

This list indicates the name of the possible predefined operations:

- **SD card command**
  - Script Command: Reboot
  - Script Command: SET_NODE_NAME
  - Script Command: FIREWALL_INSTALL
  - Script Command: Delete
  - Script Command: Download
  - Script Command: Upload
  - Script Command: UpdateBoot
  - Clone operation (clone controller contents to empty SD card)
- **FTP server command**
  - Connection to FTP server
  - List Directory
  - Change Directory
  - Create Folder
  - Rename Folder
  - Suppress Folder
  - Create File
  - Rename File
  - Suppress File
  - Download File
  - Upload File
- **OPC UA server command**
  - Connection to OPC UA server
  - Read Variable
  - Write Variable
- **Web server command**
  - Connection to Web server
  - List Variables
  - Read Variable
  - Write Variable
  - Access to File System
  - Access to logger
- **Software command**
  - Reset Origin Device
  - Login
  - Set Node Name
  - Update Logger
  - Create Application
  - Download application
  - Pass RUN / STOP
  - Reset (Cold / Warm / Origin)
  - Delete Application
  - Create Boot Application
  - Save Retain Variables

- Restore Retain Variables
- Add Group
- Remove Group
- Add User
- Remove User
- Read User Rights
- Import User Rights
- Export User Rights

Access Rights

For each **Group** linked with an **Object**, **User Rights** are predefined with specific **Access Rights**.

This table indicates the **Access Rights**:

Access Rights	Access Rights Description (depends on the Object. See Predefined Access Rights Needed by Object and Associated Operations, page 71).
VIEW	Allow to read only parameters and applications.
MODIFY	Allow to write, modify and download parameters and applications.
ADD_REMOVE	Allow to add and remove files, scripts and folders.
EXECUTE	Allow to execute and start applications and scripts.

Predefined Access Rights for Group Persona

For each **Group**, several **Objects** are predefined with preset **Access Rights**:

Group: Administrator	
Object name	**Access Rights**
Device	**VIEW** / **MODIFY** / **ADD_REMOVE** / **EXECUTE**
FTP	**VIEW** / **MODIFY** / **ADD_REMOVE**
Logger	**VIEW**
OPC_UA	**VIEW** / **MODIFY**
PlcLogic	**VIEW** / **MODIFY** / **ADD_REMOVE** / **EXECUTE**
Settings	**VIEW** / **MODIFY**
UserManagement	**VIEW** / **MODIFY**
Web	**VIEW** / **MODIFY** / **EXECUTE**
FileSystem	**VIEW** / **MODIFY** / **ADD_REMOVE**

Group: Designer / Programmer persona	
Object name	**Access Rights**
Device	**VIEW / ADD_REMOVE**
FTP	**VIEW / MODIFY / ADD_REMOVE**
Logger	**VIEW**
OPC_UA	**VIEW / MODIFY**
PlcLogic	**VIEW / MODIFY / ADD_REMOVE / EXECUTE**
Settings	**VIEW / MODIFY**
UserManagement	**VIEW**
Web	**VIEW / MODIFY / EXECUTE**
FileSystem	**VIEW / MODIFY / ADD_REMOVE**

Group: Operator persona	
Object name	**Access Rights**
Device	**VIEW**
Logger	**VIEW**
PlcLogic	**VIEW / MODIFY / EXECUTE**
Settings	**VIEW**
UserManagement	**VIEW**
Web	**VIEW / MODIFY / EXECUTE**

Group: Designer / Web designer persona	
Object name	**Access Rights**
Device	**VIEW**
FTP	**VIEW / MODIFY / ADD_REMOVE**
Logger	**VIEW**
OPC_UA	**VIEW**
PlcLogic	**VIEW**
Settings	**VIEW**
UserManagement	**VIEW**
Web	**VIEW / MODIFY / EXECUTE**
FileSystem	**VIEW / MODIFY / ADD_REMOVE**

Group: Communication expert persona	
Object name	**Access Rights**
Device	**VIEW**
FTP	**VIEW / MODIFY / ADD_REMOVE**
Logger	**VIEW**
OPC_UA	**VIEW / MODIFY**
PlcLogic	**VIEW / MODIFY / EXECUTE**
Settings	**VIEW**
UserManagement	**VIEW**
Web	**VIEW / MODIFY / EXECUTE**
FileSystem	**VIEW / MODIFY / ADD_REMOVE**

Group: Maintenance persona	
Object name	**Access Rights**
Device	**VIEW**
FTP	**VIEW / MODIFY / ADD_REMOVE**
Logger	**VIEW**
OPC_UA	**VIEW**
PlcLogic	**VIEW / EXECUTE**
Settings	**VIEW**
UserManagement	**VIEW**
Web	**VIEW / MODIFY / EXECUTE**
FileSystem	**VIEW / MODIFY / ADD_REMOVE**

Predefined Access Rights for Group Function

For each **Group**, several **Objects** are predefined with predefined **Access Rights**:

Group: Function External Media (1)	
Object name	**Access Rights**
ExternalCmd	**VIEW / MODIFY / ADD_REMOVE / EXECUTE**
(1) Enabling the objects in the group External Media will allow the access rights regardless of the user. That is to say, that the rights governing SD cards are global and are not confined to defined users.	

Group: Function File Access	
Object name	**Access Rights**
Logger	**VIEW**
FileSystem	**VIEW / MODIFY / ADD_REMOVE**

Group: Function FTP Access	
Object name	**Access Rights**
FTP	**VIEW / MODIFY / ADD_REMOVE**
Logger	**VIEW**

Group: Function Symbol Configuration Access	
Object name	**Access Rights**
Logger	**VIEW**
OPC_UA	**VIEW / MODIFY**
PlcLogic	**VIEW / MODIFY / ADD_REMOVE / EXECUTE**
Web	**VIEW / MODIFY / EXECUTE**

Group: Function Web Access	
Object name	**Access Rights**
Logger	**VIEW**
Web	**VIEW / MODIFY / EXECUTE**

Group: Function Monitor Access	
Object name	**Access Rights**
Logger	**VIEW**
OPC_UA	**VIEW**
PlcLogic	**VIEW**
Web	**VIEW**

Group: Function OPC UA Access	
Object name	**Access Rights**
Logger	**VIEW**
OPC_UA	**VIEW / MODIFY**

Group: Function Variable Access	
Object name	**Access Rights**
Logger	**VIEW**
OPC_UA	**VIEW**
PlcLogic	**VIEW / MODIFY / ADD_REMOVE / EXECUTE**
Web	**VIEW**

Predefined Access Rights Needed by Object and Associated Operations

	Access Rights			
Object Name	**ADD_REMOVE**	**MODIFY**	**VIEW**	**EXECUTE**
Device	Reset origin device	Set node name	Login	–
ExternalCmd	–	Download	Upload Clone	Delete Reboot Set Node Name Firewall install Clone Check
FTP	Connection to FTP Server Create file Create folder Upload file Upload folder Download file Download folder Delete file Delete folder	Connection to FTP Server Download file Download folder Rename File Rename Folder	Connection to FTP Server List directory Change directory Download file Download folder	–
Logger	–	–	Update logger	–
OPC_UA	–	Connection OPC_UA Read Variable Write Variable	Connection OPC_UA Read Variable	–

Object Name	Access Rights			
	ADD_REMOVE	MODIFY	VIEW	EXECUTE
PlcLogic	Create application Download application Delete application Create Boot application	Write Variable	Read Variable Save retain variables	Pass Run / Stop Reset Restore Retains Var
Settings	–	Reject / Trust Certificate Set Node Name	–	–
UserManagement	–	Add Group Remove Group Add User Remove User Edit User Rights Import User Rights Reset Origin Device	Read User Rights Export User Rights	–
Web	–	Set Variables	Connection to Web Server Monitor Variables Access Files System	Execute Command
FileSystem	–	–	–	–

Import / Export User Rights

Import User Rights and Export User Rights commands allow you to import/export Device user management (*.dum2) files found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.

NOTE: The importation of user rights into an M241 Logic Controller is only possible if these user rights have been exported from an M241 Logic Controller or M251 Logic Controller.

Symbol Rights

The Symbol Rights tab (see Tabs Description, page 59) allows you to configure user group access to the symbol sets. It consists in a customizable set of symbols allowing to separate functions and associate them with a user right. If supported by the target device, you can combine different symbol sets from the symbols of the application in the symbol configuration editor. The information about the symbol sets is downloaded to the controller. Then you can define the user group that has access to each symbol set.

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 Reset the User Rights to Default, page 200). This

effectively removes the existing application from the controller memory, but restores the ability to access the controller.

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

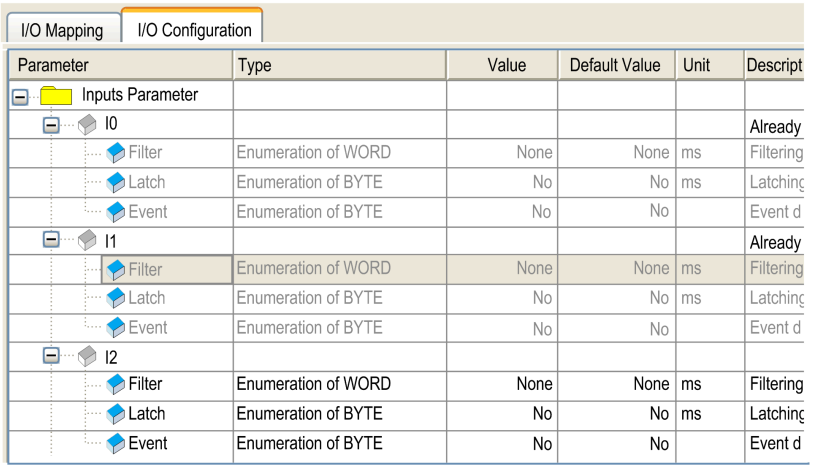
Accessing the I/O Configuration Window

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

Step	Action
1	Double-click **DI** (digital inputs) or **DQ** (digital outputs) in the **Devices tree**. Refer to Devices tree, page 18.
2	Select the **I/O Configuration** tab.

Configuration of Digital Inputs

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



NOTE: For more information on the **I/O Mapping** tab, refer to the EcoStruxure Automation Expert - Motion / 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** are 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** are disabled. When **Both edges** is selected, and the input state is TRUE before the controller is powered on, the first falling edge is ignored.
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: • The software (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), page 61.

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, page 39.

⚠WARNING

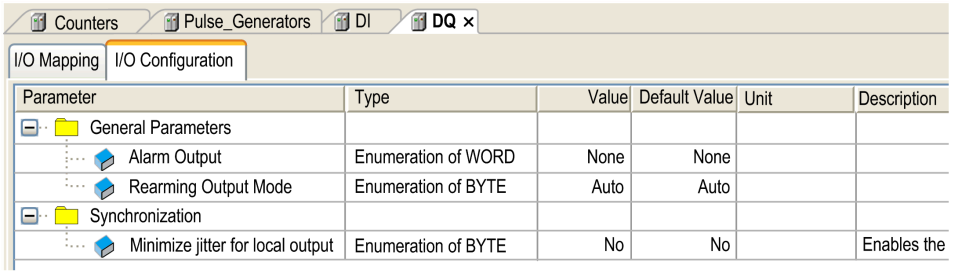
UNINTENDED MACHINE OR PROCESS START-UP

- 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 Automation Expert - Motion / 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, page 77.
Rearming Output Mode	Select the rearming output mode, page 77.
Synchronization	
Minimize jitter for local Output	Select this option to minimize jitter on local outputs, page 78.

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 short-circuit 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 the default value 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 the default value 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:

Expert Functions Configuration

Overview

This chapter describes the expert functions of the M241.

Expert Functions Overview

Introduction

The inputs and outputs available on the Modicon 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 PTO/PWM - Library Guide)	The PTO function provides 4 pulse train output channels to control 4 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 PTO/PWM - 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 PTO/PWM - 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 Automation Expert - Motion and 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, page 79.

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:

<table>
<tr><th colspan="2">Expert Function Type</th><th>24 I/O References (TM241•24•)</th><th>40 I/O References (TM241•40•)</th></tr>
<tr><td colspan="2">Total number of HSC functions</td><td>14</td><td>16</td></tr>
<tr><td rowspan="5">HSC</td><td>Simple</td><td>14</td><td>16</td></tr>
<tr><td>Main Single Phase</td><td colspan="2" rowspan="7">4</td></tr>
<tr><td>Main Dual Phase</td></tr>
<tr><td>Frequency Meter (1)</td></tr>
<tr><td>Period Meter</td></tr>
<tr><td colspan="2">PTO</td></tr>
<tr><td colspan="2">PWM</td></tr>
<tr><td colspan="2">FreqGen</td></tr>
<tr><td colspan="4">(1) When the maximum number is configured, only 12 additional HSC Simple functions can be added.</td></tr>
</table>

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

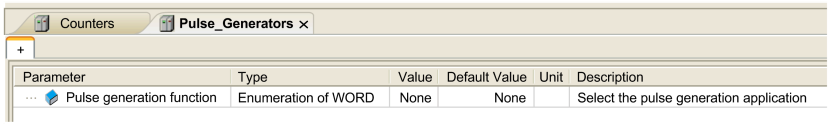
  (2) With no optional I/O configured

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	Action
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.

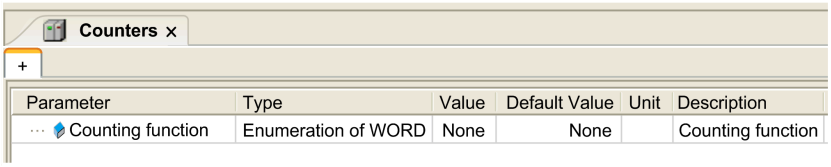
Based on the embedded counting functions, there are 5 types of counters that you can configure in EcoStruxure Automation Expert - Motion and EcoStruxure Machine Expert:

- **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.

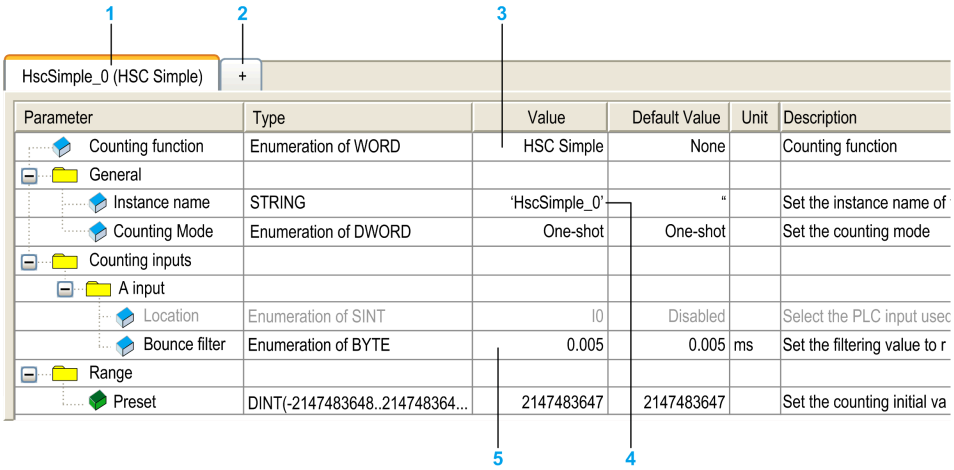
Accessing the Counting Function Configuration Window

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

Step	Action
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	Description
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 the software. 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

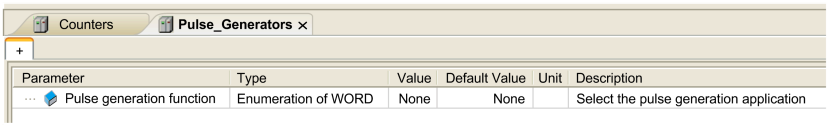
Overview

The pulse generated embedded functions available with the M241 Logic Controller 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%).

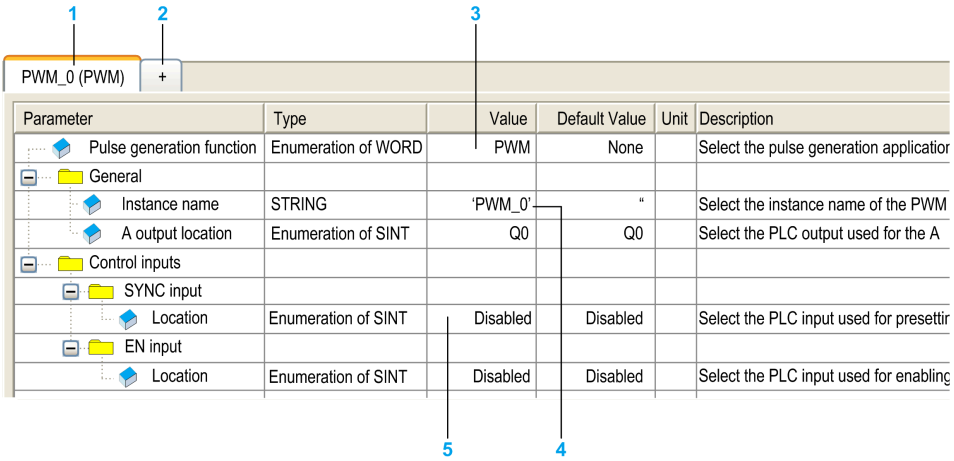
Accessing the Pulse Generators Configuration Window

Follow these steps to access the **Pulse Generators** configuration window:

Step	Action
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	Description
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 the software. 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 selecting the parameter value from the list to access its settings. The parameters available depend on the type of parameter selected.

For detailed information on configuration parameters, refer to the M241 Logic Controller, PTO/PWM Library Guide.

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.

⚠ 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 Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

Expansion Modules Configuration

Overview

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

TM4/TM3/TM2 Expansion Modules Configuration

Introduction

The Modicon M241 Logic Controller supports the following expansion modules:

- TM4 communication 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 TM4,TM3 and TM2 expansion modules configuration, refer to the TM4 Expansion Modules Programming Guide, TM3 Expansion Modules 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 Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

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.

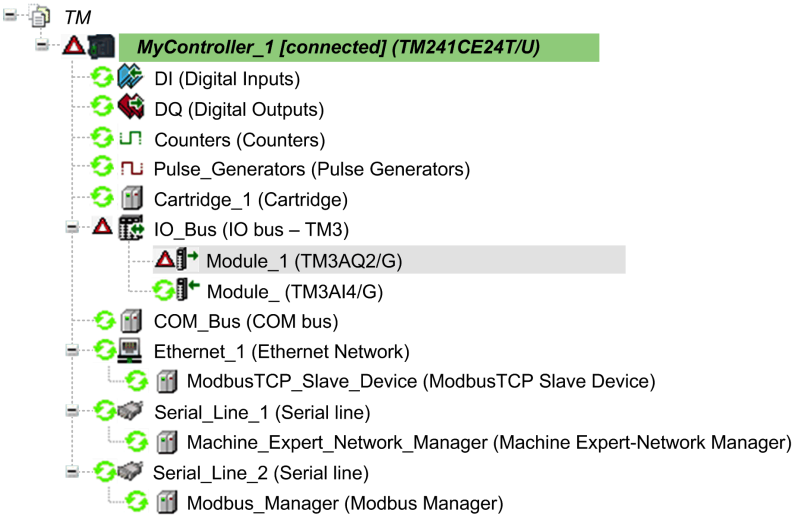
I/O Expansion Bus Errors

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, page 92), the logic controller detects 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 Automation Expert - Motion and EcoStruxure Machine Expert are 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*. The 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 only if no expansion modules are in error (`TM3_MODULE_R[i].i_wModuleState` = `TM3_BUS_ERROR`). Refer to Restarting the I/O Expansion Bus, page 89.
- Issuing a **Reset Warm** or **Reset Cold** command with the software, page 48.

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, the 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 the software, page 48.

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, page 46”) 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 the software, page 48.

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, page 42.

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, page 92.

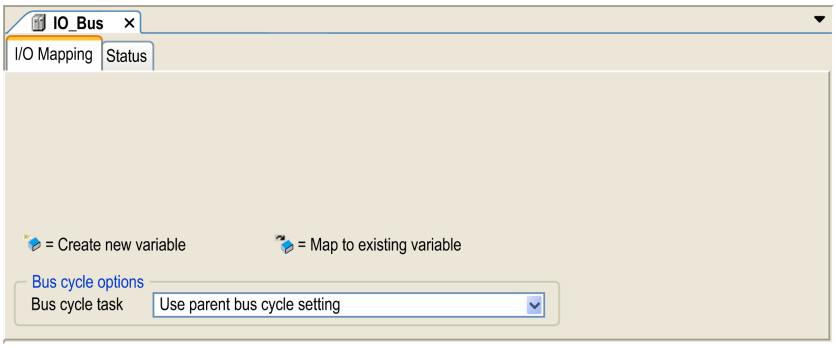
TM3 I/O Bus Configuration

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**, page 61 bus cycle task.

Configuring the I/O Bus

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

Step	Action
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**.

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

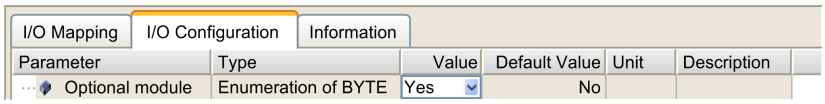
UNINTENDED EQUIPMENT OPERATION

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: 

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 shows the internal ID codes of expansion modules:

Modules sharing the same internal ID code	ID code
TM2DDI16DT, TM2DDI16DK	0
TM2DRA16RT, TM2DDO16UK, TM2DDO16TK	1
TM2DDI8DT, TM2DAI8DT	4
TM2DRA8RT, TM2DDO8UT, TM2DDO8TT	5
TM2DDO32TK, TM2DDO32UK	3
TM2DMM24DRF, TM2DDI32DK	2
TM2DMM8DRT	6
TM2ALM3LT, TM2AMI2HT, TM2AMI2LT, TM2AMI4LT, TM2AMI8HT, TM2AMM3HT, TM2AMM6HT, TM2AMO1HT, TM2ARI8HT, TM2ARI8LRJ, TM2ARI8LT, TM2AVO2HT	96
TM3DI16K, TM3DI16, TM3DI16G	128
TM3DI8, TM3DI8G, TM3DI8A	132
TM3DQ16R, TM3DQ16RG, TM3DQ16T, TM3DQ16TG, TM3DQ16TK, TM3DQ16U, TM3DQ16UG, TM3DQ16UK	129
TM3DQ32TK, TM3DQ32UK	131
TM3DQ8R, TM3DQ8RG, TM3DQ8T, TM3DQ8TG, TM3DQ8U, TM3DQ8UG	133
TM3DM8R, TM3DM8RG	134
TM3DM16R	141
TM3DM24R, TM3DM24RG	135
TM3DM32R	143
TM3SAK6R, TM3SAK6RG	144
TM3SAF5R, TM3SAF5RG	145
TM3SAC5R, TM3SAC5RG	146
TM3SAFL5R, TM3SAFL5RG	147
TM3AI2H, TM3AI2HG	192
TM3AI4, TM3AI4G	193
TM3AI8, TM3AI8G	194
TM3AQ2, TM3AQ2G	195
TM3AQ4, TM3AQ4G	196
TM3AM6, TM3AM6G	197
TM3TM3, TM3TM3G	198
TM3TI4, TM3TI4G	199
TM3TI4D, TM3TI4DG	203
TM3TI8T, TM3TI8TG	200
TM3DI32K	130
TM3XTYS4	136
TM3XHSC202, TM3XHSC202G	217

Optional Modules Diagnostic

The following diagnostic information is available: **TM3_MODULE_R[i].i_wModuleState** system variable, where **[i]** identifies the absent TM3 optional expansion module, is set to **TM3_MISSING_OPT_MOD**.

Ethernet Configuration

Introduction

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

Ethernet Features, Functions and Services

Presentation

Ethernet Features, Functions and Services

The controller supports the following services:

- Modbus TCP Server, page 101
- Modbus TCP Client, page 101
- Web Server, page 128
- FTP Server, page 102
- SNMP, page 105
- Controller as Target Device On EtherNet/IP, page 105
- Controller as Slave Device On Modbus TCP, page 122
- IEC VAR ACCESS, page 96
- Web visualization
- OPC UA Server, page 178

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
CoDeSys Protocol (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. The server connections stay open as long as the controller stays in operational states (*RUNNING*, *STOPPED*, *HALT*).
4. The server connections are closed when leaving 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 Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.

TM241CE••• Specific Considerations

A TM4ES4 expansion module connected to a TM241CE••• controller provides an additional Ethernet network:

- The controller Ethernet 1 port is dedicated to device network connections.
- The TM4ES4 ports are dedicated to communication between machines or with the control network.

For example, you can:

- Connect your PC to a TM4ES4 port.
- Use a Modbus TCP I/O scanner with the Ethernet 1.

The Network Variables List (NVL) communication works on the:

- Ethernet 1 port.
- TM4ES4 ports:
  - if the Ethernet 1 port has a valid IP address and is connected to a device, or
  - if the library behavior is modified to also test the TM4ES4 ports.

Follow these steps to enable testing of TM4ES4 ports:

Step	Action
1	In the **Applications tree**, right-click the **Application** node and select **Properties**.
2	In the **Properties - Application** window, select the **Build** tab.
3	Enter ETH2_NVL_Communication in the **Compiler defines** field and click **OK**.

NOTE: For more information about **Compiler defines**, refer to the Menu Commands Online Help.

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, page 188. 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 tab, page 60
- **changeIPAddress** function block, page 210

NOTE: If the attempted addressing method is unsuccessful, the link uses a default IP address, page 100 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

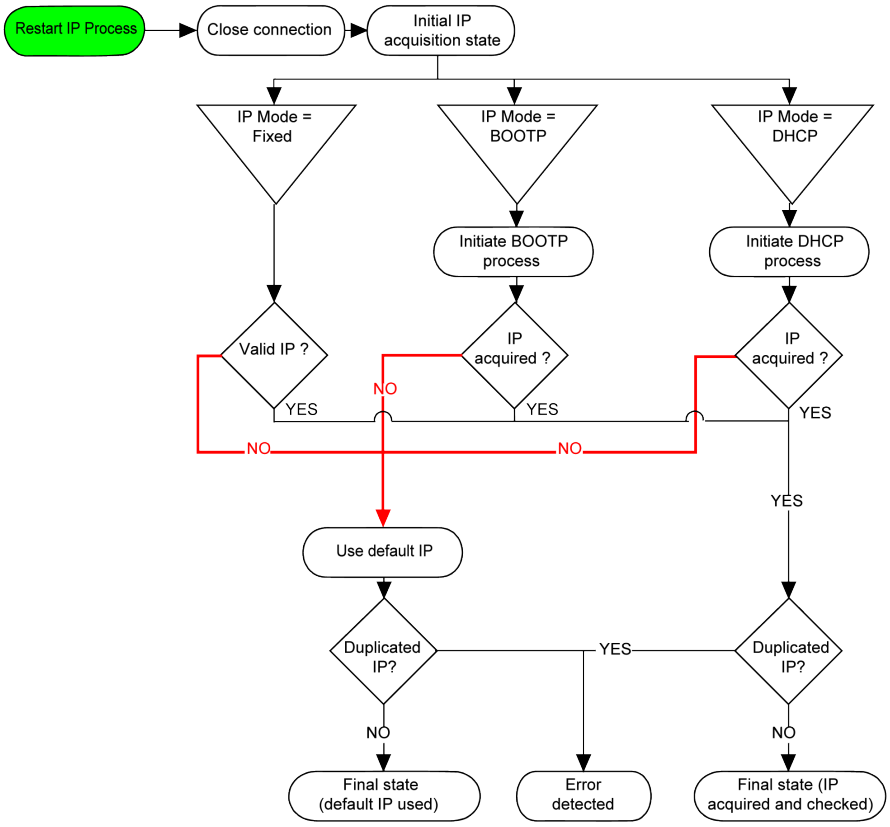
- Verify that there is only one master controller configured on the network or remote link.
- Verify that all devices have unique addresses.
- Obtain your IP address from your system administrator.
- Confirm that the IP address of the device is unique before placing the system into service.
- Do not 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:



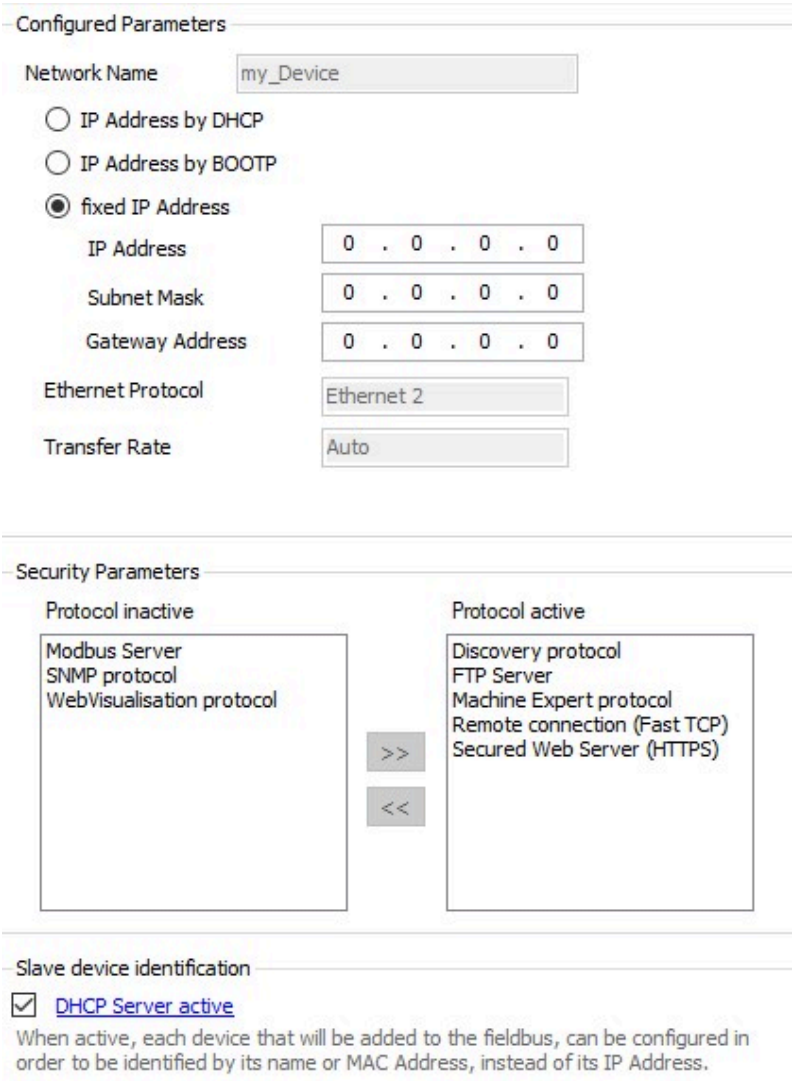
NOTE: If a device programmed to use the DHCP or BOOTP addressing methods is unable to contact its respective server, the controller uses the default IP address. It repeats its request constantly.

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 offline mode, you see the **Configured Parameters** window (displayed above). You can edit the parameters.
- If you are in online mode, you see the **Configured Parameters** and **Current Settings** windows (not shown). You cannot edit the parameters.

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.
Ethernet Protocol	Protocol type used (Ethernet 2).
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 255.255.0.0.

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:

<table>
<tr><th>Address Class</th><th colspan="5">Byte1</th><th>Byte 2</th><th>Byte 3</th><th>Byte 4</th></tr>
<tr><td>Class A</td><td>0</td><td colspan="4">Network ID</td><td colspan="3">Host ID</td></tr>
<tr><td>Class B</td><td>1</td><td>0</td><td colspan="3">Network ID</td><td colspan="3">Host ID</td></tr>
<tr><td>Class C</td><td>1</td><td>1</td><td>0</td><td colspan="4">Network ID</td><td>Host ID</td></tr>
<tr><td>Class D</td><td>1</td><td>1</td><td>1</td><td>0</td><td colspan="4">Multicast Address</td></tr>
<tr><td>Class E</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td colspan="4">Address reserved for subsequent use</td></tr>
</table>

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 same network.

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

The gateway address can be defined on Ethernet_1 interface or on TM4ES4 Ethernet interface. The traffic to unknown networks is sent through this gateway address, or to address configured on IP routing table, page 63.

Security Parameters

This table describes the different security parameters:

Security Parameters	Description	Default settings
Discovery protocol	This parameter activates/deactivates **Discovery protocol**. When deactivated, Discovery requests are ignored.	Active
FTP Server	This parameter activates/deactivates the **FTP Server** of the controller. When deactivated, FTP requests are ignored.	Active
CoDeSys protocol	This parameter activates/deactivates the **CoDeSys protocol** on Ethernet interfaces. When deactivated, any programming software request is rejected, including those from the UDP or TCP connection. Therefore, no connection is possible on Ethernet from a programming PC, 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 activates/deactivates the **Modbus Server** of the controller. When deactivated, any Modbus request to the controller is ignored.	Inactive
Remote connection (Fast TCP)	This parameter activates/deactivates the remote connection. When deactivated, Fast TCP requests are ignored.	Active
Secured Web Server (HTTPS)	This parameter activates/deactivates the Web server of the controller. When deactivated, HTTPS requests to the controller Web server are ignored.	Active
SNMP protocol	This parameter activates/deactivates the SNMP server of the controller. When deactivated, SNMP requests are ignored.	Inactive
WebVisualisation protocol	This parameter activates/deactivates the WebVisualisation pages of the controller. When deactivated, HTTP requests to the logic controller WebVisualisation protocol are ignored.	Inactive

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, page 154.

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 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 a **Modbus TCP Slave Device**, page 122 so that the contents of the *%IW* and *%QW* buffers are time-consistent in the associated IEC task (MAST by default).

FTP Server

Introduction

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

NOTE: When transferring text files to the controller, switch your FTP client into binary transfer type to avoid conversions of Carriage Return and LineFeed character pairs into a single LineFeed.

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 application environments 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, such as an Intrusion Prevention System or Intrusion Detection System.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Install certificates that are issued by publicly known Trusted Certificate Authorities.
- Keep your systems up-to-date and rely only on legitimate sources.
- 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 more information on organizational measures and rules covering access to infrastructures, refer to ISO/IEC 27000 series, Common Criteria for Information Technology Security Evaluation, ISO/IEC 15408, IEC 62351, ISA/IEC 62443, NIST Cybersecurity Framework, Information Security Forum - Standard of Good Practice for Information Security and refer to Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment.

NOTE: Make use of the security-related commands (see 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.

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, page 59.

To access the FTP server you must first connect to the controller with EcoStruxure Automation Expert - Motion, EcoStruxure Machine Expert or Controller Assistant and activate the user rights or create the user for the first login.

NOTE: FTPS (explicit over TLS FTP) is set by default. Simple FTP (non secure) access is not possible at first connection. Set the parameter 1106 to 0 in the post configuration and reboot the controller to allow Simple FTP connection.

Files Access

See File Organization, page 25.

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 application environments 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, such as an Intrusion Prevention System or Intrusion Detection System.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Install certificates that are issued by publicly known Trusted Certificate Authorities.
- Keep your systems up-to-date and rely only on legitimate sources.
- 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 more information on organizational measures and rules covering access to infrastructures, refer to ISO/IEC 27000 series, Common Criteria for Information Technology Security Evaluation, ISO/IEC 15408, IEC 62351, ISA/IEC 62443, NIST Cybersecurity Framework, Information Security Forum - Standard of Good Practice for Information Security and refer to Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment.

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.se.com.

SNMP Client

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

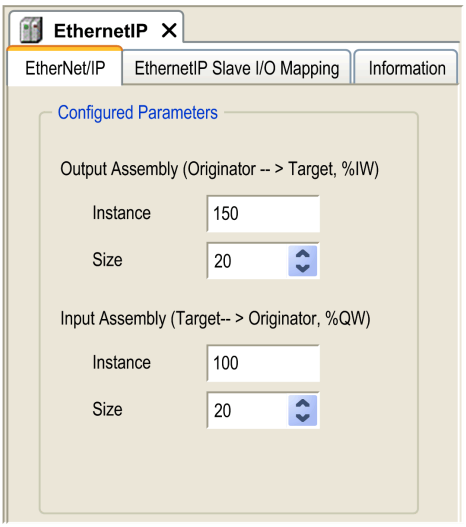
To configure your M241 Logic Controller as an EtherNet/IP target device, you must:

Step	Action
1	Select **EthernetIP** in the **Hardware Catalog**.
2	Drag and drop it to the **Devices tree** on one of the highlighted nodes. **NOTE:** If the chosen node is **COM_Bus**, a TM4ES4 expansion module is automatically added to your configuration. For more information on adding a device to your project, refer to: • Using the Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide • Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

EtherNet/IP Parameters Configuration

To configure the EtherNet/IP parameters, double-click **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	Default Value
Output Assembly	**Instance**	150...189	150
	Size	2...250	20
Input Assembly	**Instance**	100...149	100
	Size	2...250	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 contextual 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 Electric 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, page 105.

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, page 109	01	1	1	Supports the reset service
Message Router Object, page 111	02	1	1	Explicit message connection
Assembly Object, page 112	04	2	2	Defines I/O data format
Connection Manager Object, page 113	06	–	1	–
TCP/IP Interface Object, page 114	F5	1	1	TCP/IP configuration
Ethernet Link Object, page 115	F6	1	1	Counter and status information
Interface Diagnostic Object, page 116	350	1	1	–
IOScanner Diagnostic Object, page 119	351	1	1	–
Connection Diagnostic Object, page 119	352	1	1	–
Explicit Connection Diagnostic Object , page 121	353	1	1	–
Explicit Connections Diagnostic List Object, page 121	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

NOTE: The reset command is rejected by the controller if an active EtherNet/IP connection exists.

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 Electric ID
2	Get	Device type	UINT	0E	Controller
3	Get	Product code	UINT	1001	Controller product code
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.
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
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
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.
2	Get	Configuration Capability	DWORD	Bit level	• 0: BOOTP Client • 1: DNS Client • 2: DHCP Client • 5: Configured in the software 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.
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
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
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.
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, page 123.

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.modbus.org website.

Adding a Modbus TCP Slave Device

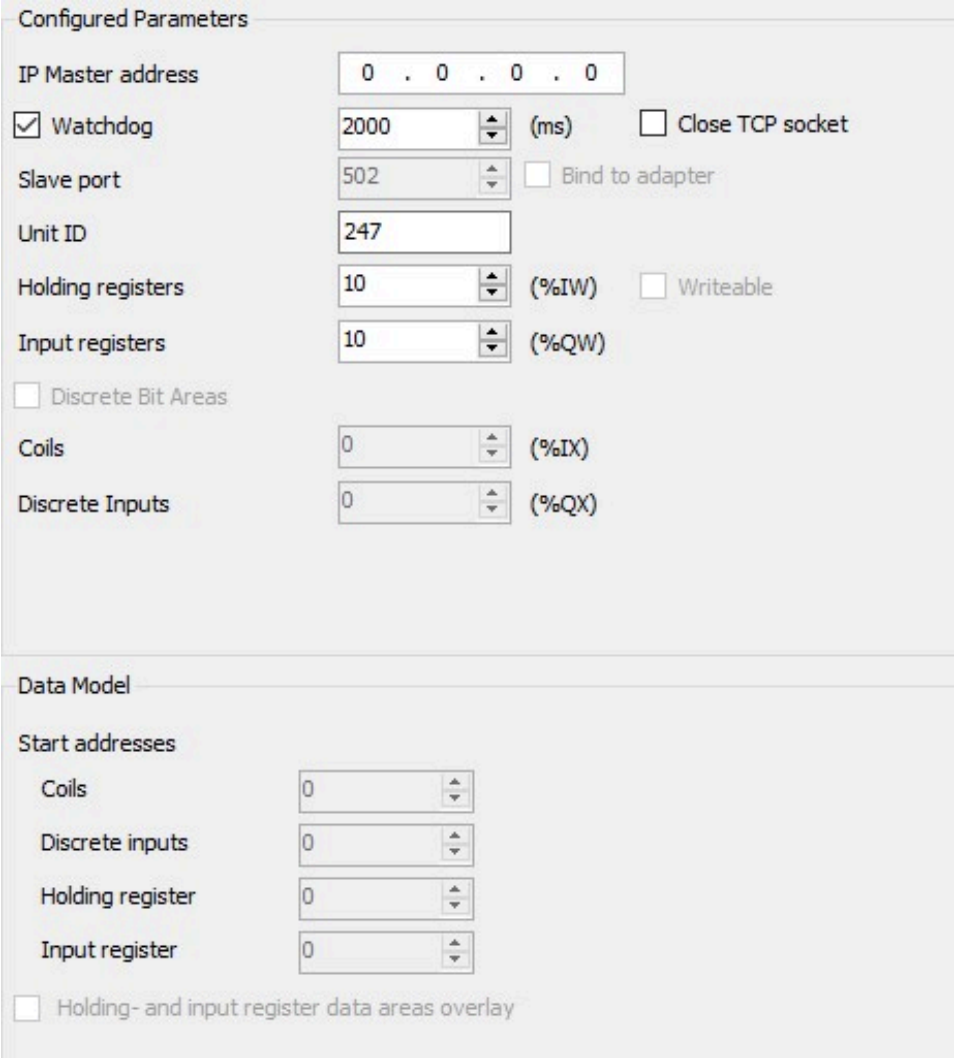
To configure your M241 Logic Controller as a Modbus TCP slave device, you must:

Step	Action
1	Select **Modbus TCP Slave Device** in the **Hardware Catalog**.
2	Drag and drop it to the **Devices tree** on one of the highlighted nodes. **NOTE:** If the chosen node is **COM_Bus**, a TM4ES4 expansion module is automatically added to your configuration. For more information on adding a device to your project, refer to: • Using the Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide • Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

Modbus TCP Configuration

To configure the Modbus TCP slave device, double-click **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.
Close TCP socket	When **Close TCP socket** is selected, the TCP socket is closed if the **Watchdog** is enabled and the set time is exceeded.
Slave Port	Modbus communication port (502) **NOTE:** The port number can be modified using the changeModbusPort script command, page 126.
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, page 101 Slave, it facilitates a read operation by the external I/O scanner.

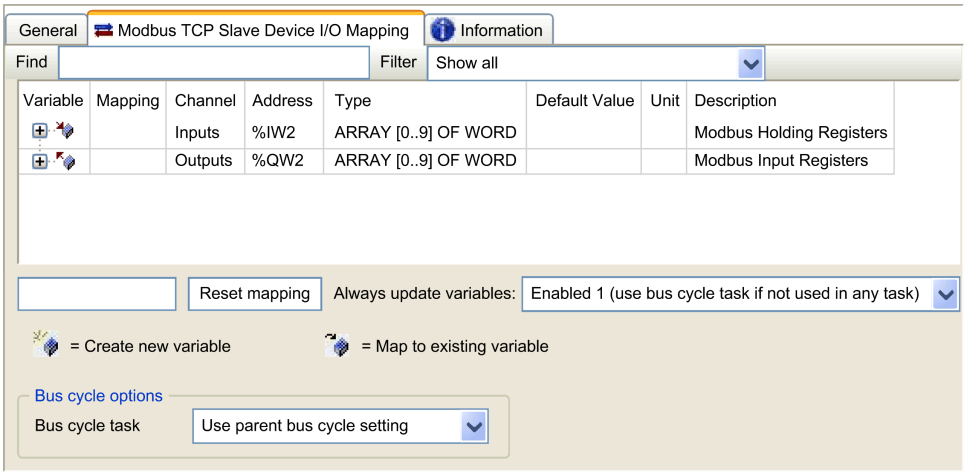
Once a **Modbus TCP Slave Device** has been configured, Modbus commands sent to its Unit ID (Modbus address) access the *%IW* and *%QW* objects of the controller instead of the regular Modbus words (accessed when the Unit ID is 255). This facilitates read/write operations by a Modbus TCP IOScanner application.

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**.

For the parameter **Always update variables**, choose one of the following options:

- **Use parent device setting**
- **Enabled 1 (use bus cycle task if not used in any task)** (default setting)
- **Enabled 2 (always in bus cycle task)**

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** (default setting)
- **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 Automation Expert - Motion / 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, page 123.

The default Modbus port number is 502.

Command	Description
`changeModbusPort` *"portnum"*	*portnum* is the new Modbus port number to use and is passed as a string of characters. Before running the command, refer to Used Ports, page 149 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, protocol active selection for the Modbus Server in the **Security Parameters** group on the Ethernet Configuration window, page 99 is no longer valid.

Running the Command from an SD Card Script

Step	Action

1	Create a script file, page 196, 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 Function Block

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);
```

Web Server

Introduction

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 on Windows PCs or mobile devices. No configuration or programming is required.

The Web server can be accessed by the web browsers listed below:

- Google Chrome (version 87 or greater)
- Mozilla Firefox (version 62 or greater)
- Microsoft Edge (version 42 or greater)

The Web server can be accessed by the mobile device web browsers listed below:

- iOS Safari
- Android Chrome

HTTP (non secured connections) requests are redirected to HTTPS (secure connections).

The Web server is limited to 10 TCP connections, page 95.

The Web server has access to your application for reading and writing data and controlling the state of the controller. By enabling the Web server, you enable these functions. You can disable the Web server on an interface by deselecting the Web server active parameter in the Ethernet Configuration tab, page 99.

If there are cybersecurity concerns over these functions, you must, at a minimum, assign a secure password to the Web server or disable the Web server to help prevent unauthorized access to the application.

The Web server allows you to remotely monitor a controller and its application, and to perform various maintenance activities including modifications to data and configuration parameters, and change the state of the controller. Ensure that the

immediate physical environment of the machine and process is in a state that does not present 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.

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**, page 59.

To access the Web server you must first connect to the controller with EcoStruxure Automation Expert - Motion, 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.

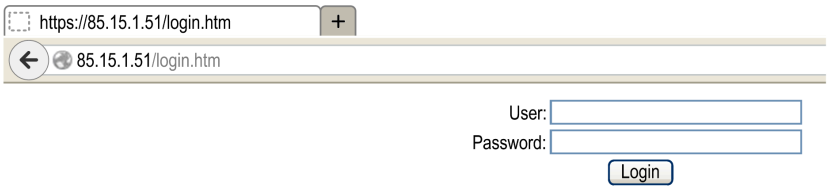
In order to change the password, go to **Users and Groups** tab of the device editor. For more information, refer to the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.

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 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 Reset the User Rights to Default, page 200). 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 application environments 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, such as an Intrusion Prevention System or Intrusion Detection System.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Install certificates that are issued by publicly known Trusted Certificate Authorities.
- Keep your systems up-to-date and rely only on legitimate sources.
- 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 more information on organizational measures and rules covering access to infrastructures, refer to ISO/IEC 27000 series, Common Criteria for Information Technology Security Evaluation, ISO/IEC 15408, IEC 62351, ISA/IEC 62443, NIST Cybersecurity Framework, Information Security Forum - Standard of Good Practice for Information Security and refer to Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment.

Home Page Access Menu

The Home Page Access menu bar lets you access the main Web server pages.

The Web server contains the following pages:

Menu	Page	Description
Home	Home, page 130	Home page of the controller Web server page. Provides access to the tabs: • Monitoring, page 132 • Diagnostics, page 136 • Maintenance, page 139

Monitoring Menu

Monitoring: Data Parameters

Monitoring Web Server Variables

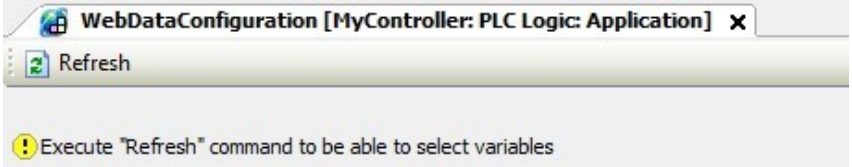
To monitor Web server variables, you must add a **Web Data Configuration** object to your project. Within this object, you can select all variables you want to monitor.

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

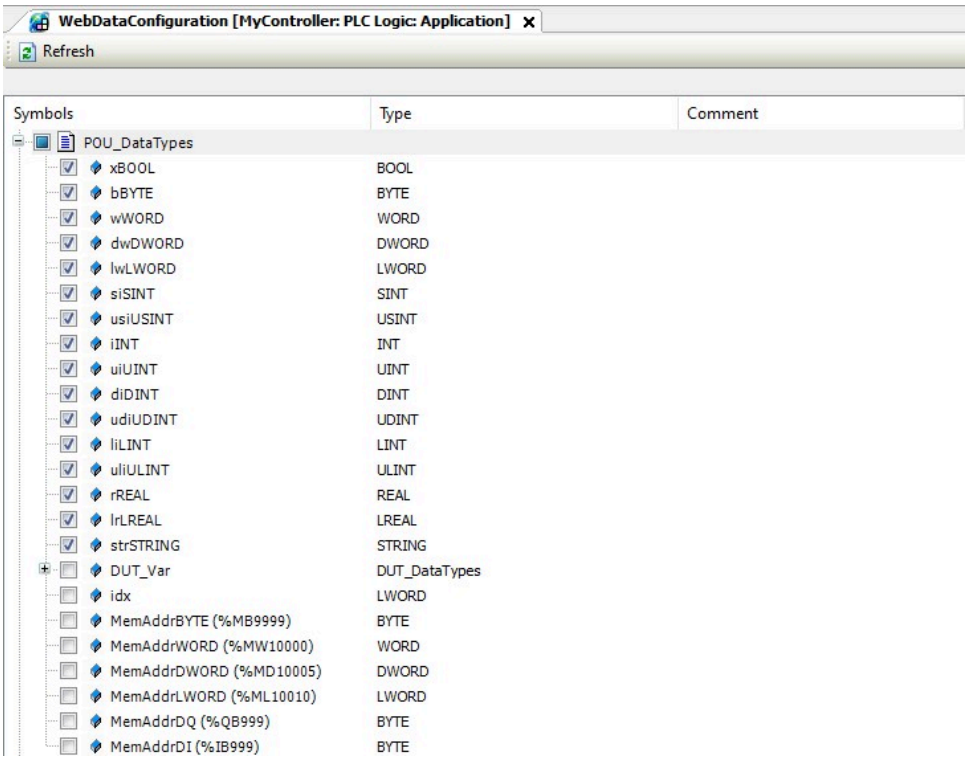
Step	Action
1	Right click the **Application** node in the **Applications tree** tab.
2	Click **Add Object** > **Web Data Configuration...**. **Result:** The **Add Web Data Configuration** window is displayed.
3	Click **Add**. **Result:** The **Web Data Configuration** object is created and the **Web 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.

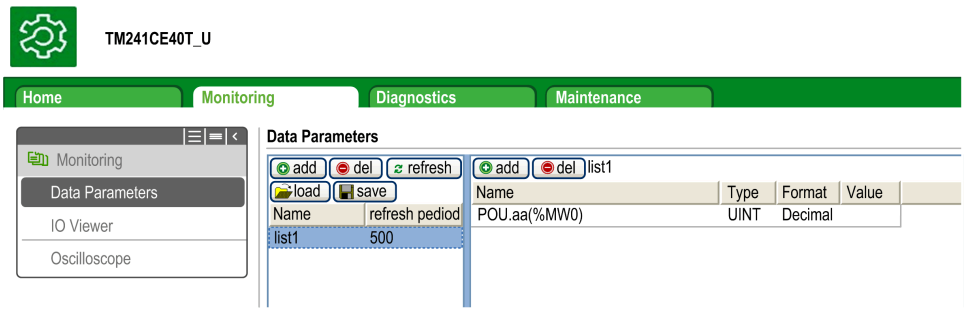
NOTE: The data types supported by the controller are listed in the **IEC variable Base Types** column of this table, page 183.

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 non-volatile 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:



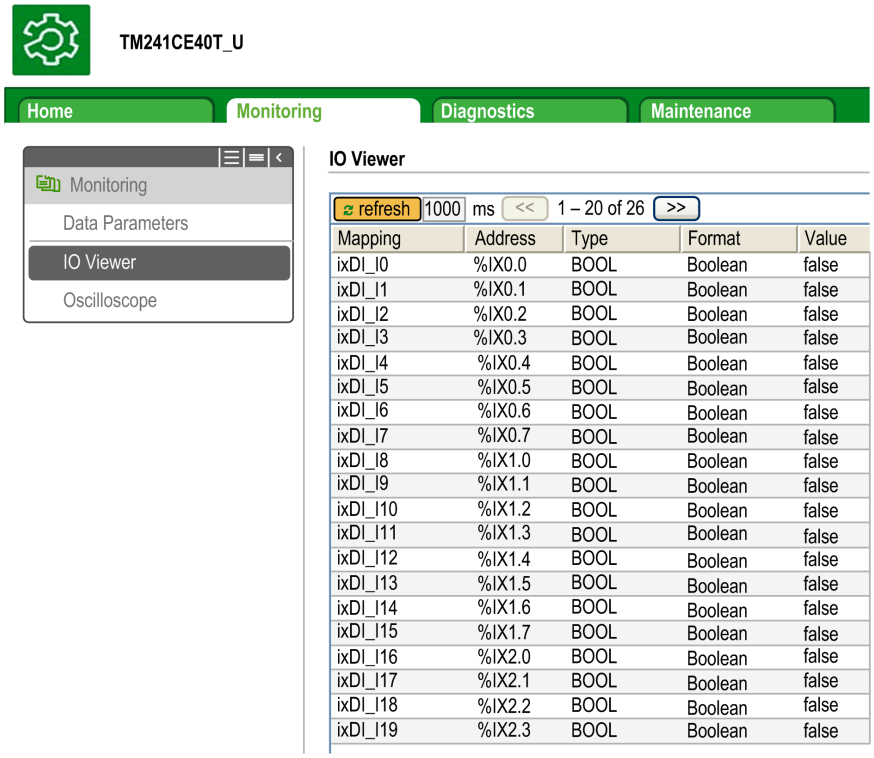
Element	Description
Add	Adds a list description or a variable
Del	Deletes a list description or a variable
Refresh period	Refreshing period of the variables contained in the list description (in ms)
Refresh	Enables I/O refreshing: • Gray button: refreshing disabled • Orange button: refreshing enabled
Load	Loads saved lists from the controller non-volatile memory 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, page 28).

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

Monitoring: IO Viewer Submenu

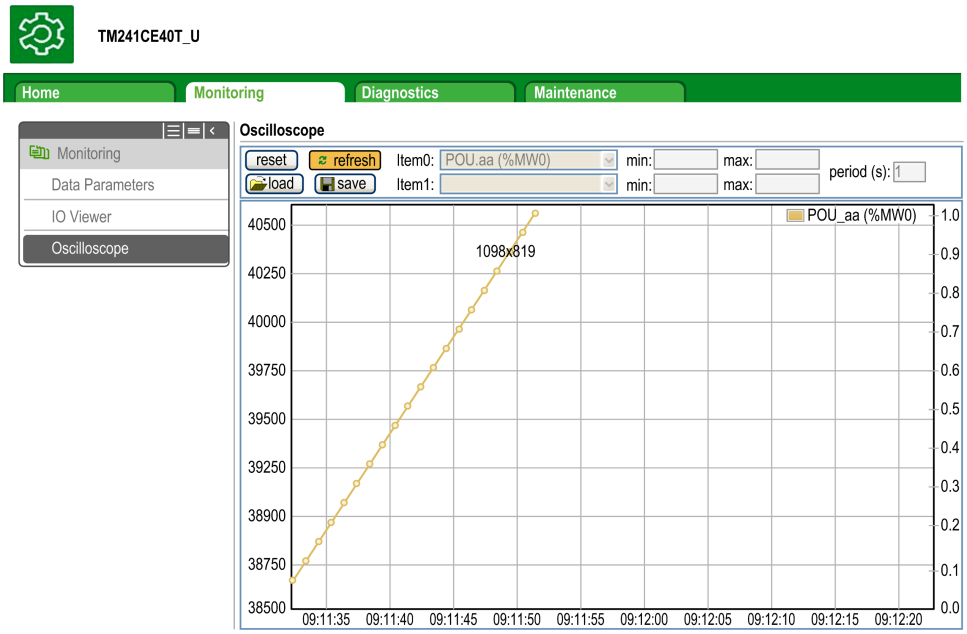
The **IO Viewer** submenu allows you to display and modify the I/O values:



Element	Description
Refresh	Enables I/O refreshing: • Gray button: refreshing disabled • Orange button: refreshing enabled
Period (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:

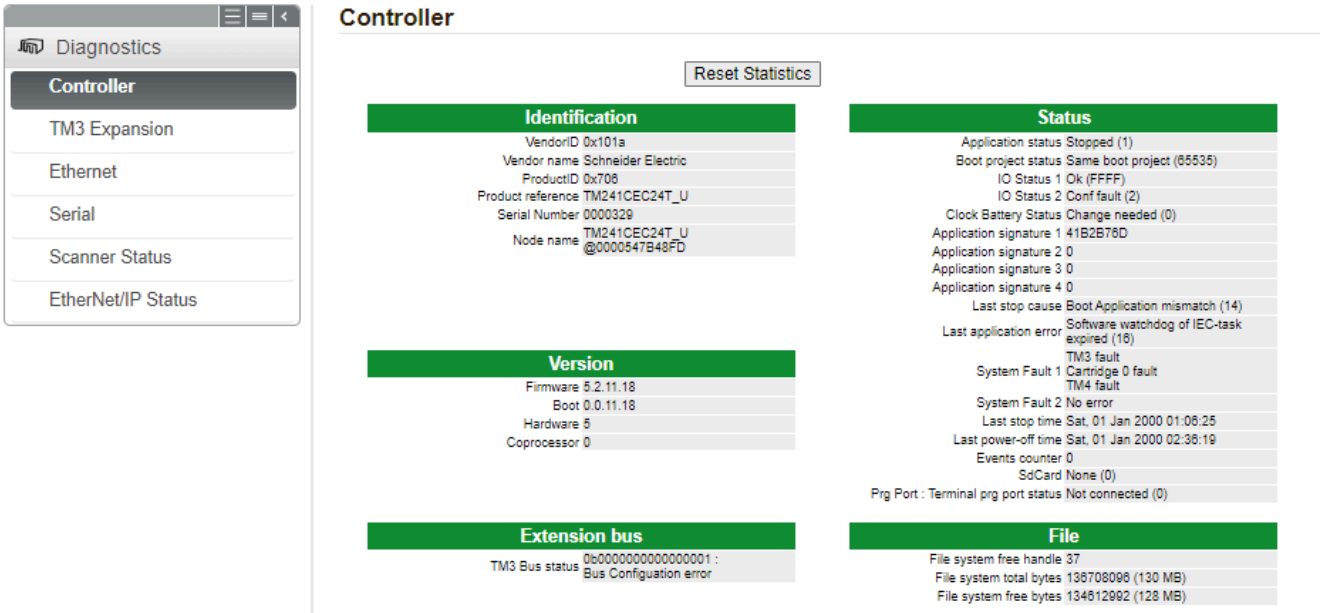


Element	Description
Reset	Erases the memorization
Refresh	Starts/stops refreshing
Load	Loads parameter configuration of Item0 and Item1
Save	Saves parameter configuration of Item0 and Item1 in the controller
Item0	Variable to be displayed
Item1	Variable to be displayed
Min	Minimum value of the variable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

Diagnostic Menu

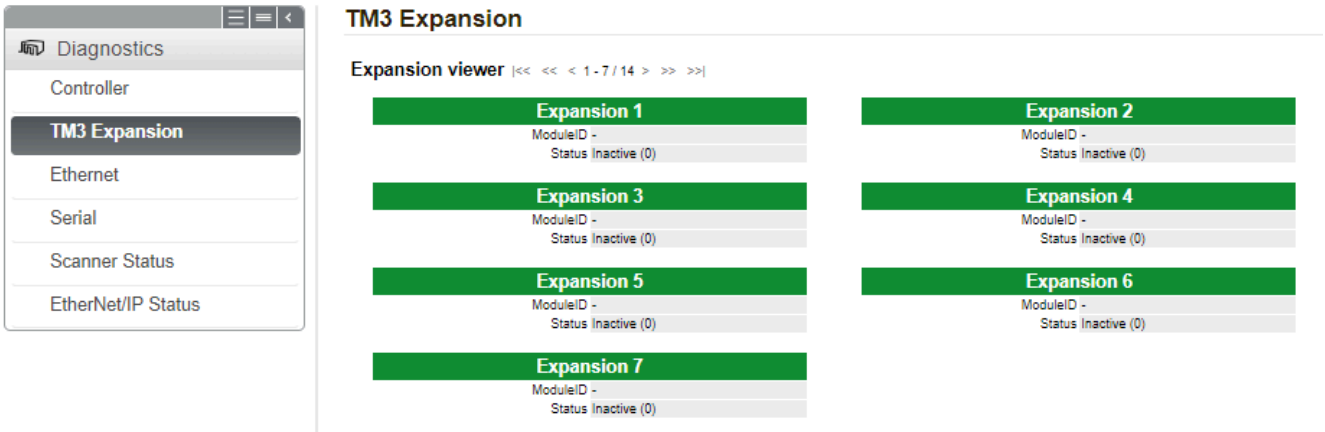
Diagnostics: Controller Submenu

The **Controller** submenu displays information about the controller:



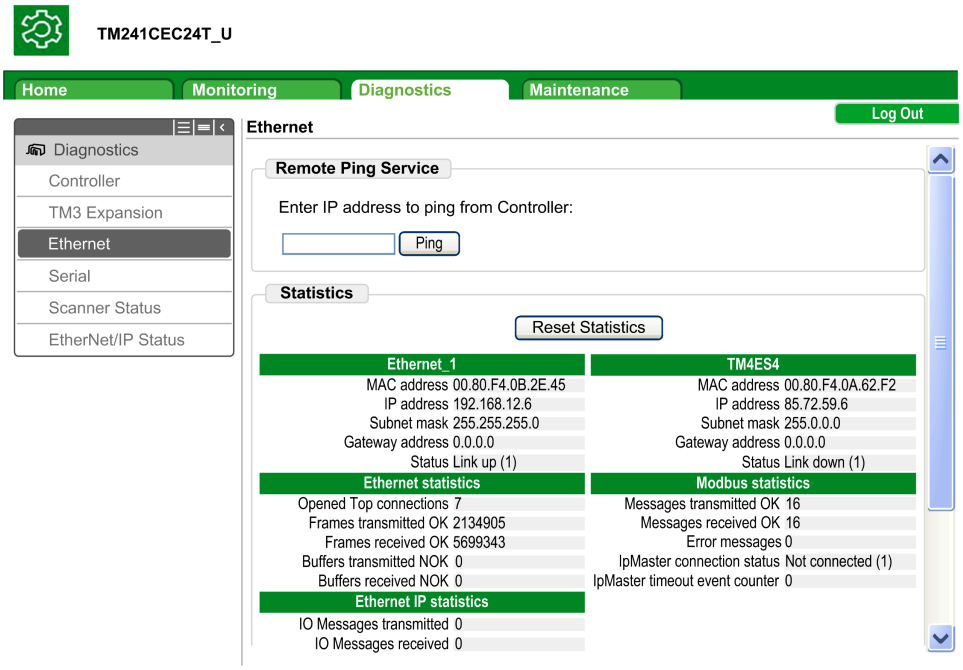
Diagnostics: TM3 Expansion Submenu

The **TM3 Expansion** submenu shows the TM3 expansion modules status:



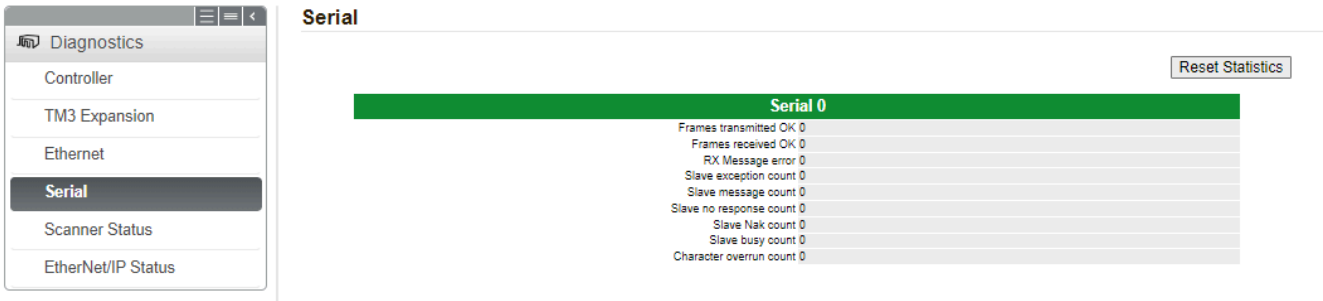
Diagnostics: Ethernet Submenu

The **Ethernet** submenu displays the Ethernet ports status and access to the remote ping service:



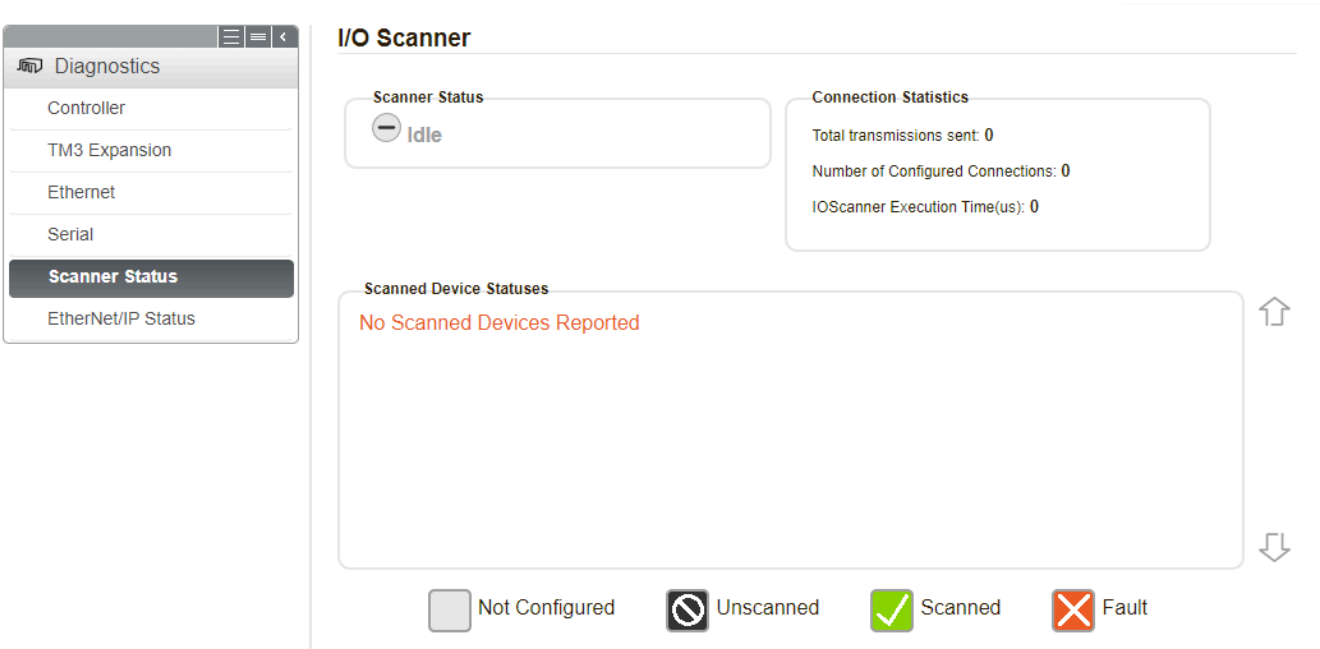
Diagnostics: Serial Submenu

The **Serial** submenu displays the status of Serial Line connection:



Diagnostics: Scanner Status Submenu

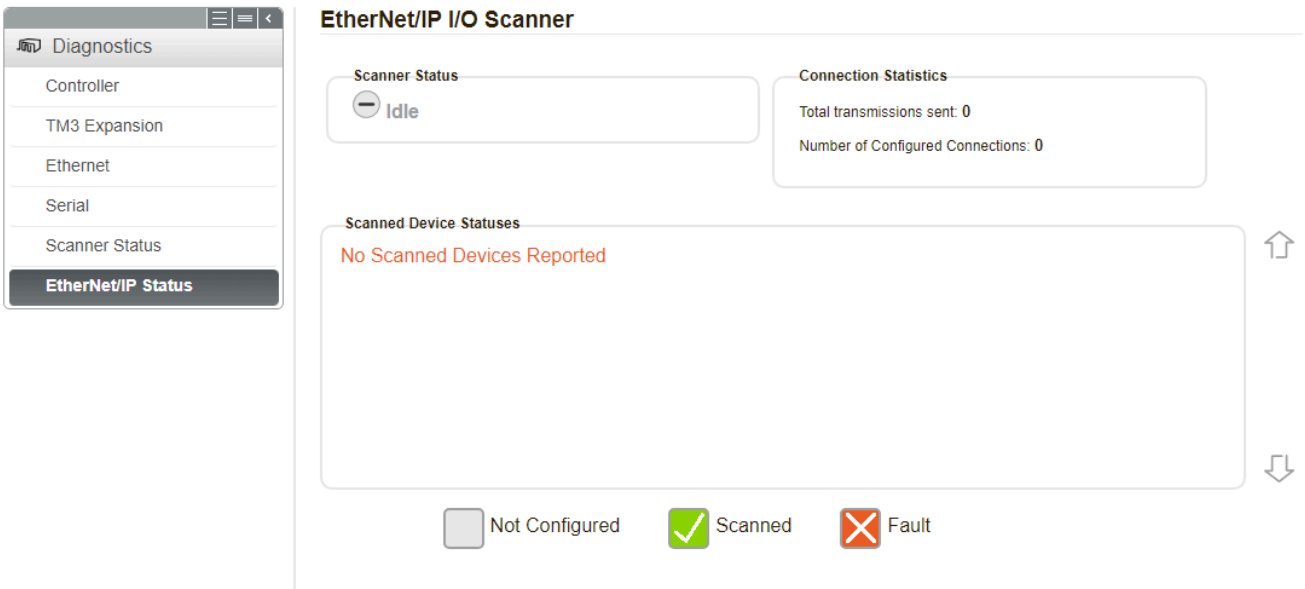
The **Scanner Status** submenu displays status of the Modbus TCP I/O Scanner (IDLE, STOPPED, OPERATIONAL) and the health bit of up to 64 Modbus slave devices:



For more information, refer to 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:

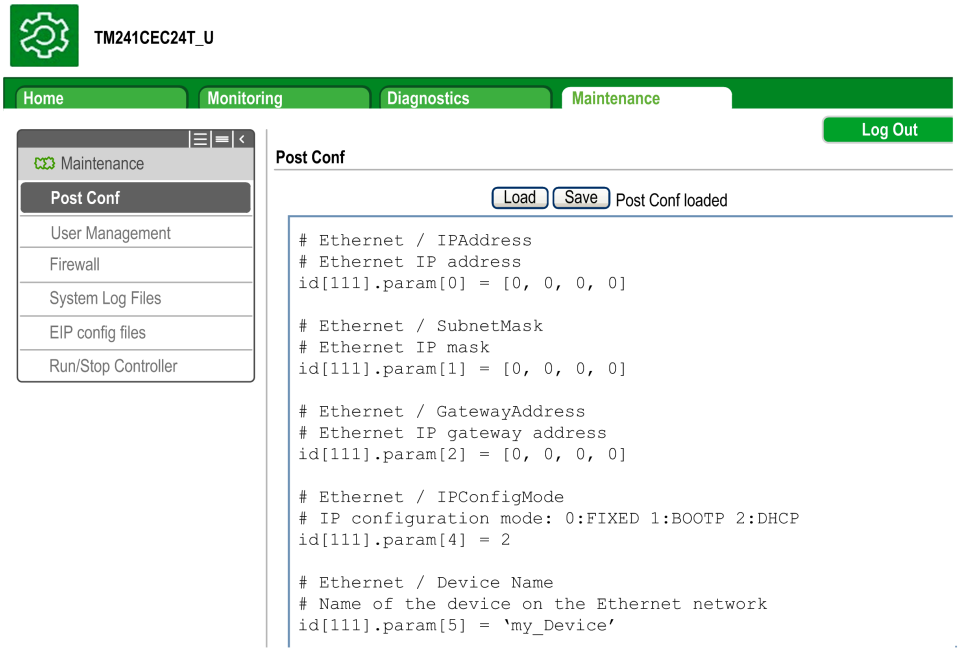


For more information, refer to EtherNet/IP User Guide.

Maintenance Menu

Maintenance: Post Conf Submenu

The **Post Conf** submenu allows you to update the post configuration file, page 188 saved on the controller:



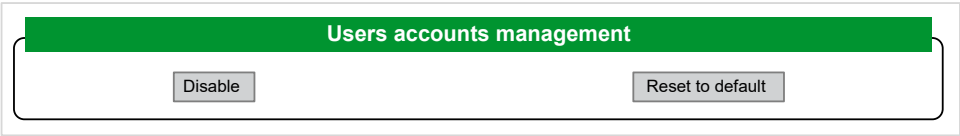
Step	Action
1	Click **Load**.
2	Modify the parameters, page 190.
3	Click **Save**. **NOTE:** The new parameters are considered at the next Post Configuration file reading, page 188.

Maintenance: User Management Submenu

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

- **User accounts management**:

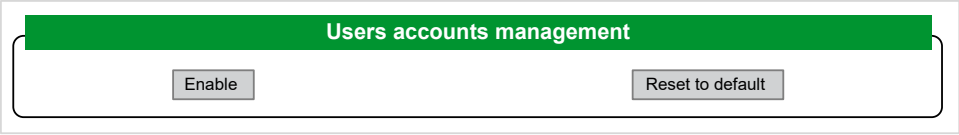
Allows you to manage user accounts management, removing passwords and returning user accounts on the controller to default settings.



Click **Disable** to deactivate all user rights on the controller. (Passwords are saved and are restored if you click **Enable**). Then, 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. See Login and passwords table, page 65.

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



Click **Enable** to restore the previous user rights saved on the controller. Then, click **OK** on the window that appears to confirm. As a result, users have to enter the password previously set to connect to the controller. See Login and passwords table, page 65.

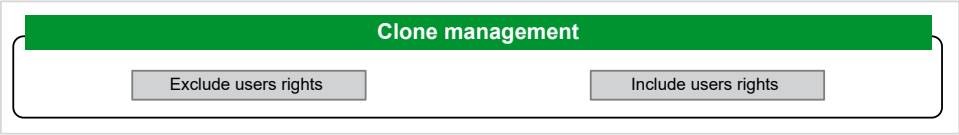
NOTE: The **Enable** button only appears if the user rights are disabled and the user rights backup file is available on the controller.

Click **Reset to default** to return all user accounts on the controller to their default setting state. Then, 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 with an SD Card, page 197.



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

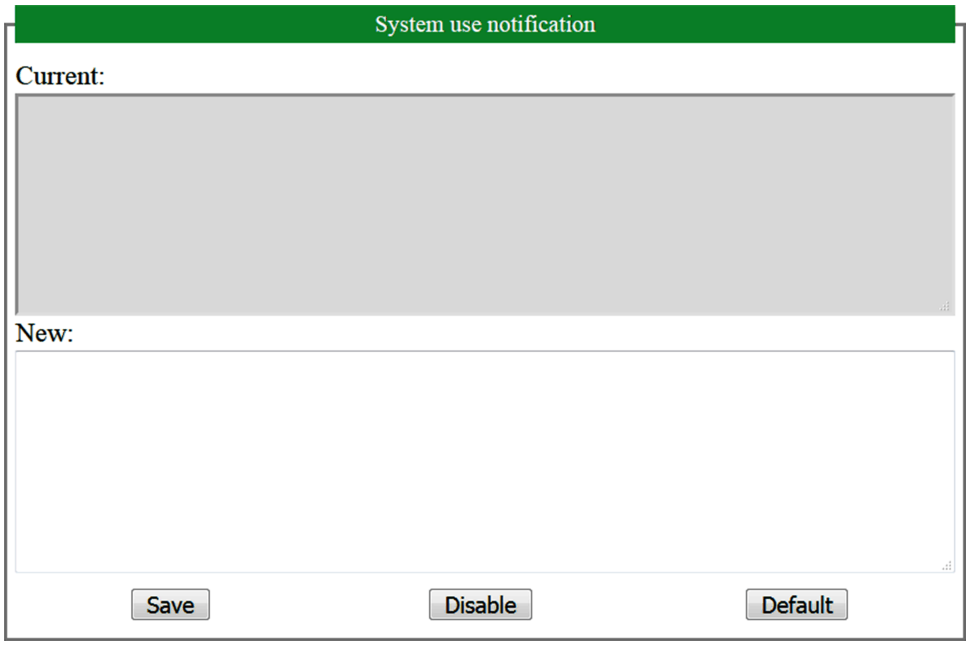
NOTE: By default, the users rights are excluded.

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 user is connected to the controller using a secure protocol.

- **System use notification**:

Allows you to customize a message which is displayed at login.

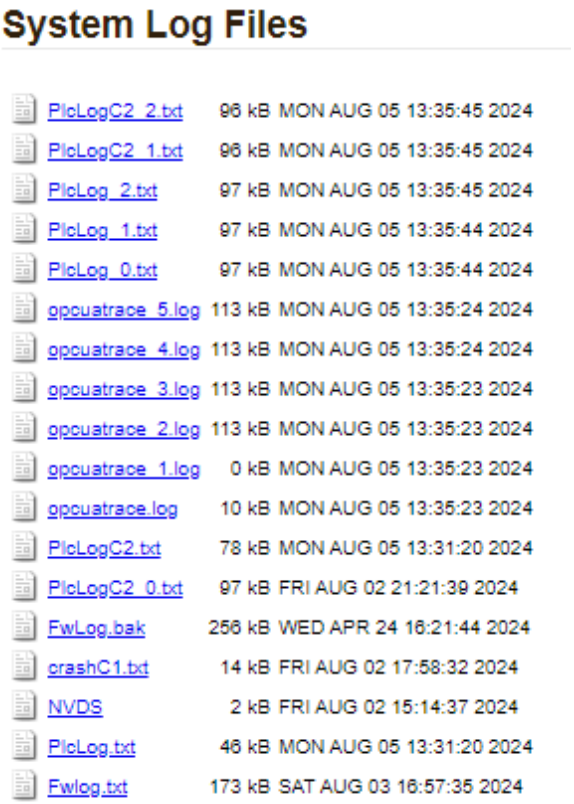


Maintenance: Firewall Submenu

The **Firewall** submenu allows you to modify the default firewall configuration file, page 143:

Maintenance: System Log Files Submenu

The **System Log Files** submenu provides access to log files generated by the controller:



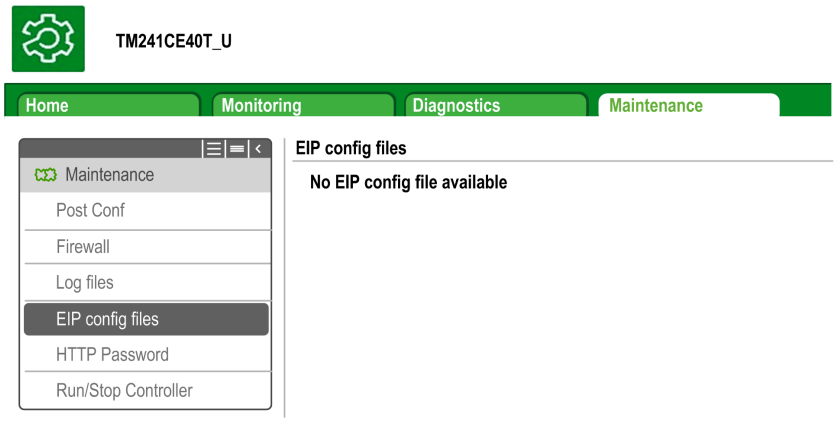
NOTE: A maximum of 4 log files can be stored for PlcLog and PlcLogC2 in the Syslog folder including the 3 backup files. A maximum of 2 and 6 log files can be stored for firmware and OPC UA trace logs respectively.

When the maximum file size is reached, previous logs are saved to backup files. When the maximum file number is reached for PlcLog, firmware and OPC UA trace logs, previous logs must be deleted in order to continue saving new diagnostic information.

Maintenance: EIP Config Files Submenu

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

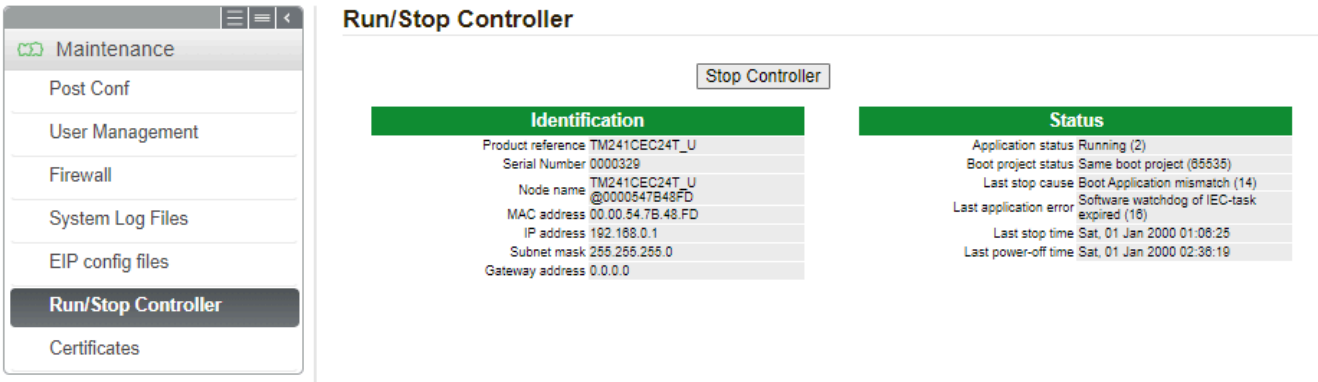
Index of `/usr`:



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

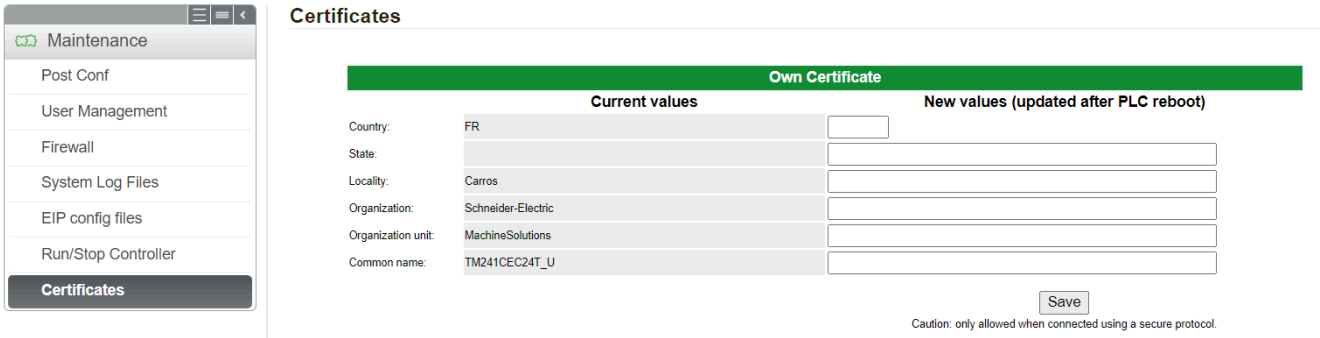
Maintenance: Run/Stop Controller Submenu

The **Run/Stop Controller** submenu allows you to manually stop and restart the controller:



Maintenance: Certificates Submenu

The **Certificates** submenu allows you to modify the certificates owned by the controller:



Firewall Configuration

Introduction

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

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 application environments 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, such as an Intrusion Prevention System or Intrusion Detection System.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Install certificates that are issued by publicly known Trusted Certificate Authorities.
- Keep your systems up-to-date and rely only on legitimate sources.
- 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 more information on organizational measures and rules covering access to infrastructures, refer to ISO/IEC 27000 series, Common Criteria for Information Technology Security Evaluation, ISO/IEC 15408, IEC 62351, ISA/IEC 62443, NIST Cybersecurity Framework, Information Security Forum - Standard of Good

Practice for Information Security and refer to Cybersecurity Guidelines for EcoStruxure Machine Expert, Modicon and PacDrive Controllers and Associated Equipment.

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, page 144.
- A function block, page 145 in the application.

Application Settings

See Ethernet Configuration, page 99.

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, page 146. 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, page 146. 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 inserted 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, page 196.

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)	Frames with the specified destination port number are allowed.
`Firewall Eth1 Reject port_type port Y`	Y = (destination port numbers)	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)	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)	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)	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)	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)	Frames from the specified IP address and with a destination port number in the specified range are allowed.
`Firewall Eth1 Reject IP •.•.•.• on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers)	Frames from the specified IP address and with a destination port number in the specified range are rejected.

Command	Range	Description
`Firewall Eth1 Allow IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type port Y`	• = 0...255 Y = (destination port numbers)	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)	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)	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)	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 Fast TCP on interface ETH1. This allow to connect to the
controller using TCP

Firewall Eth1 Allow TCP port 11740;

; 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.

Used Ports

This table lists the port numbers used by the M241 Logic Controller services:

Protocol	Destination Port Numbers
Programming software	UDP 1740, 1741, 1742, 1743 TCP 1105, 11740 (Fast TCP)
FTP	TCP 21
HTTP / HTTPS	TCP 80, 443 (Web server) TCP 8080 (Web visualization)
Modbus	TCP 502 (1)
OPC UA	TCP 4840
DHCP	UDP 67 (server), 68 (client)
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 changeModbusPort command, page 126.	

Industrial Ethernet Manager

Introduction

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

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.

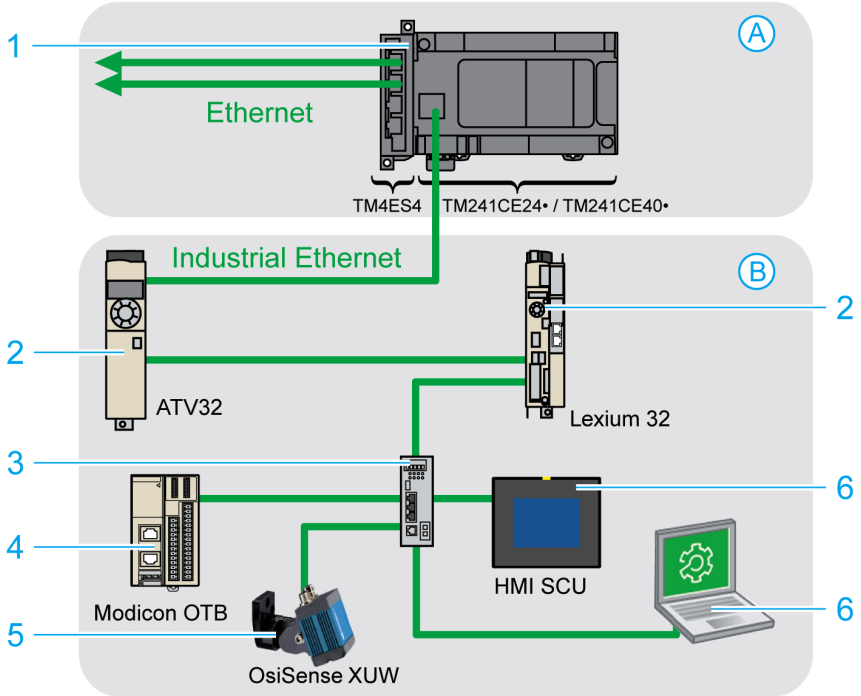
On an Industrial Ethernet network, you can connect:

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

For more information, refer to Industrial Ethernet Overview, User Guide.

Industrial Ethernet Architecture

This figure presents a typical Industrial Ethernet architecture:



A	Control network
B	Device network
1	Logic controller (see 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 Automation Expert - Motion and 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 server devices managed by the logic controller, monitored within a timeslot of 35 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
	Client/Server
Other services	FDT/DTM/EDS management
	FDR (Fast Device Replacement)
	DHCP server
	Security management (refer to Security Parameters, page 101 and Firewall Configuration, page 143)
	Modbus TCP server
	Modbus TCP client
	Web server, page 128
	FTP Server (FTP and TFTP protocols), page 102
	OPC UA, page 178
	SNMP, page 105
	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 **Devices 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 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 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 Modbus TCP, User Guide).

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.

Serial Line Configuration

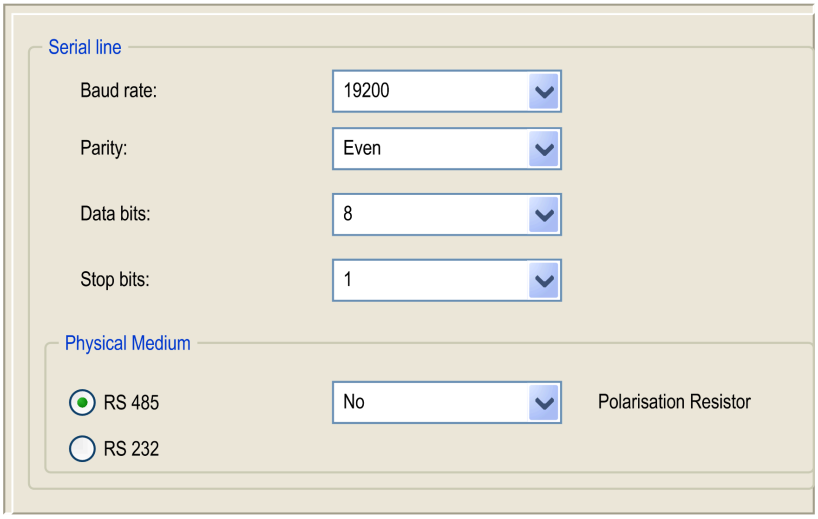
Introduction

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 CoDeSys protocol by default when new or when you update the controller firmware. The CoDeSys 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)
Network Manager	115200
Modbus Manager	
ASCII Manager	
Modbus IOScanner	

Network Manager

Introduction

Use the Network Manager to exchange variables with a XBTGT/XBTGK Advanced Panel with CoDeSys protocol, or when the Serial Line is used for EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert programming.

Adding the Manager

To add a 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 Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

Configuring the Manager

There is no configuration for the Network Manager.

Adding a Modem

To add a modem to the Network Manager, refer to Adding a Modem to a Manager, page 170.

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:

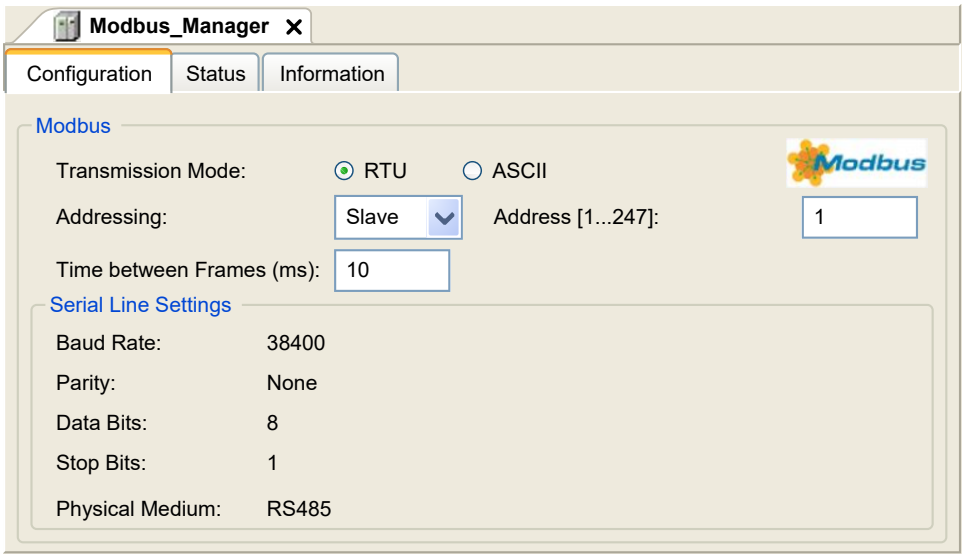
• Using the Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / 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 Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide).

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

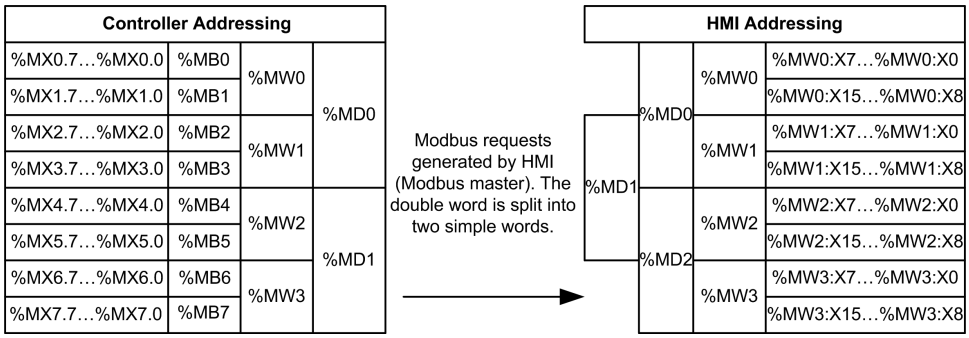
UNINTENDED EQUIPMENT OPERATION

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, page 170.

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:

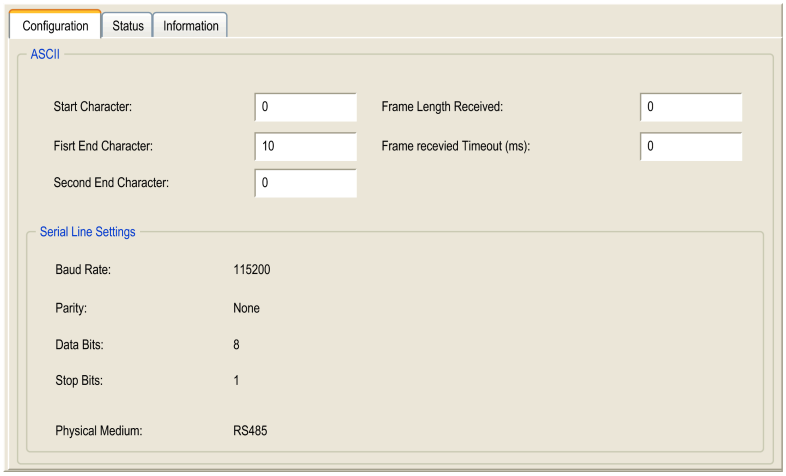
• Using the Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / 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, page 155.

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, page 170.

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:

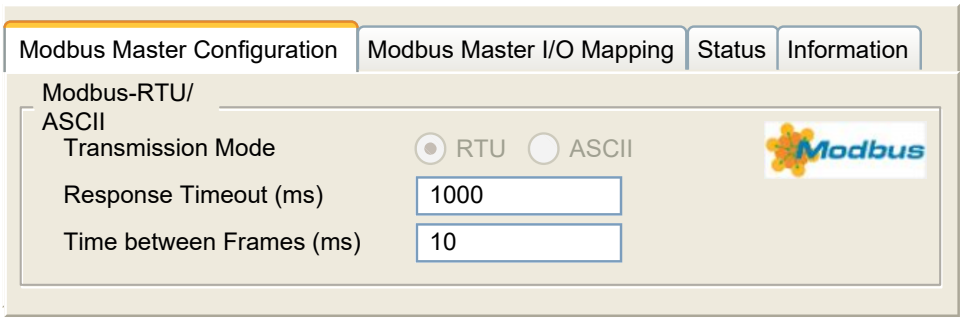
• Using the Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / 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:

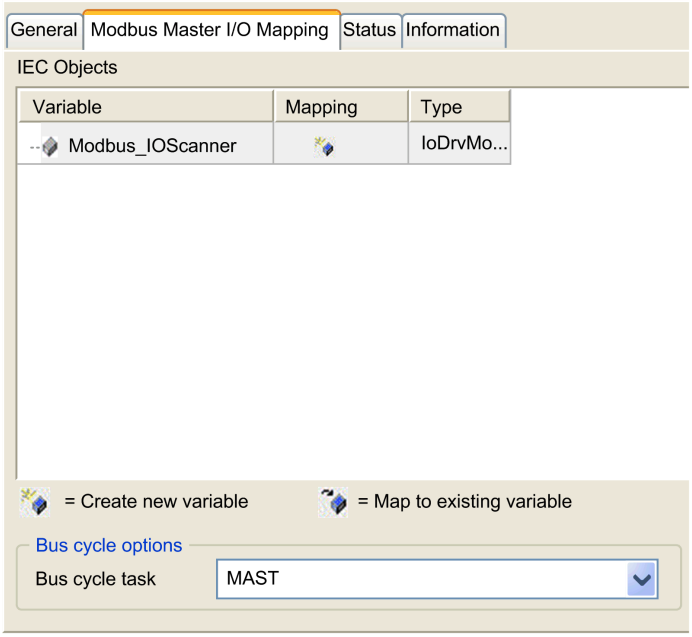
Element	Description
Transmission Mode	Specifies the transmission mode to use: • RTU: uses binary coding and CRC error-checking (8 data bits) • ASCII: messages are in ASCII format, LRC error-checking (7 data bits) Set this parameter identical for each Modbus device on the network.
Response Timeout (ms)	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 Automation Expert - Motion / 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:

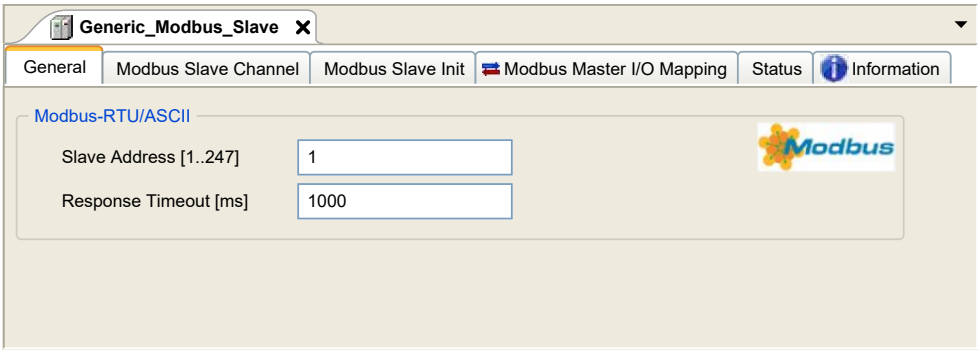
• Using the Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

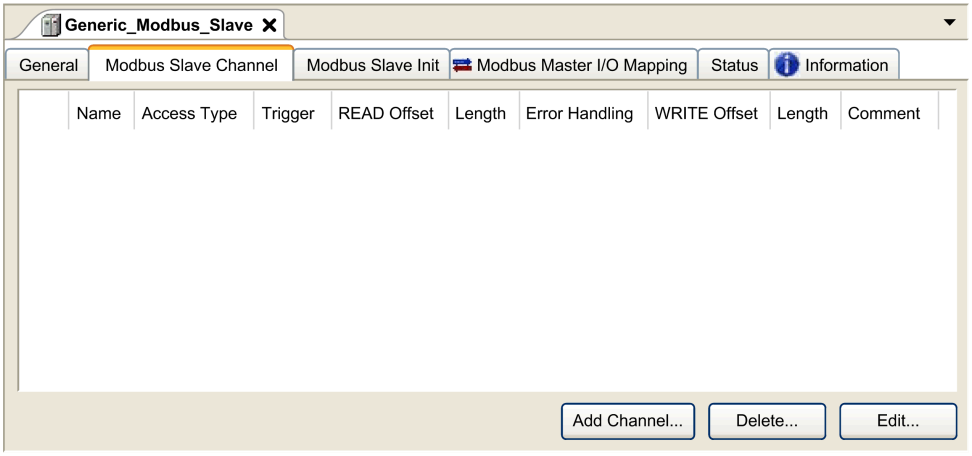
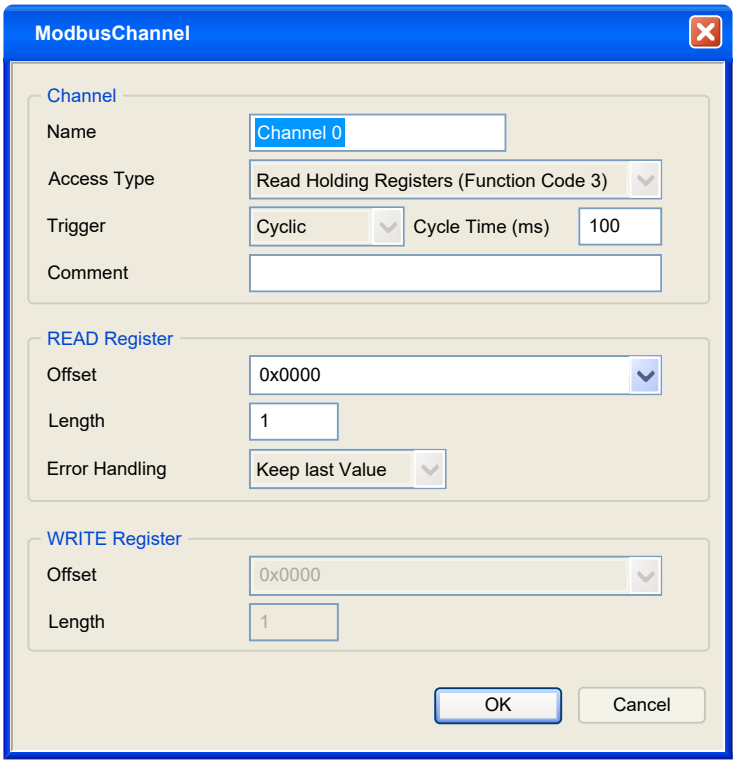
NOTE: The variable for the exchange is automatically created in the `%IWx` and `%QWx` of the **Modbus Serial Master I/O Mapping** tab.

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).

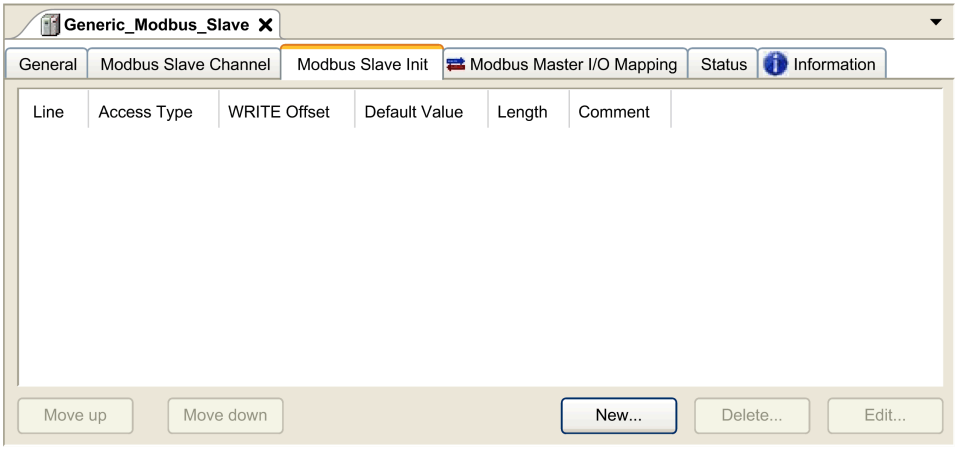
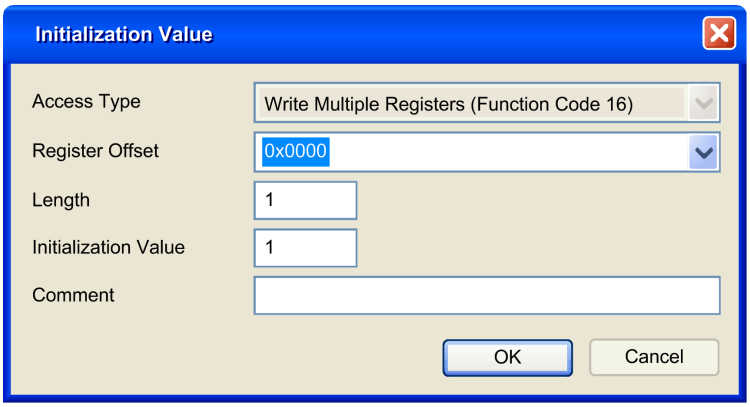
To configure the **Modbus Channels**, proceed as follows:

Step	Action
1	Click the **Modbus Slave Channels** tab: 
2	Click the **Add Channel** button: 

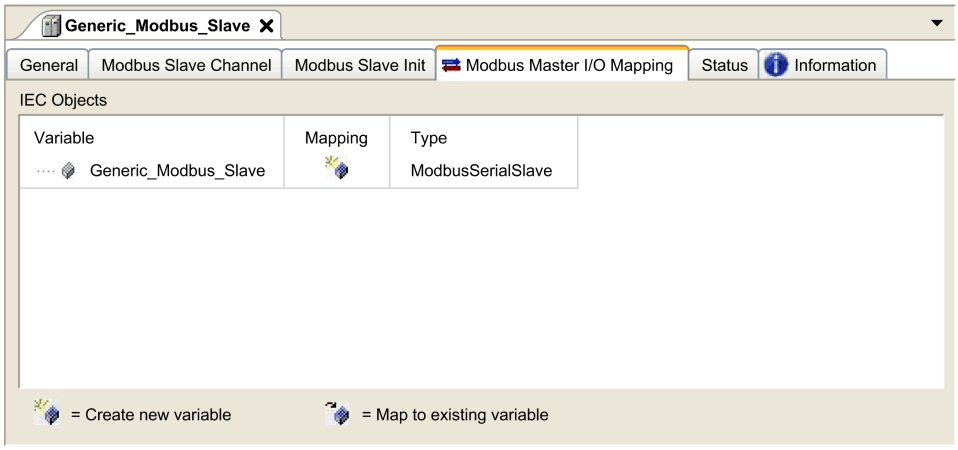
Step	Action
3	Configure an exchange: In the area **Channel**, you can add the following values: • **Name**: Enter a name for your channel. • **Access Type**: Choose the exchange type: Read or Write or Read/Write multiple requests. See Access Types, page 168. • **Trigger**: Choose the trigger of the exchange. 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 area **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 area **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. **Result:** The configured channels are displayed: Library Manager / SR_Main / TM3BC_ModbusTCP / TM3BC_ModbusSL / **Generic_Modbus_Slave** / Modbus_IOScanner General / Modbus Slave Channel / Modbus Slave Init / Modbus Master I/O Mapping / Modbus Master IEC Objects / Status / Information

	Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset	Length	Comment
0	Channel 0	Read Holding Registers (Function Code 03)	Cyclic, t#100ms	16#0000	1	Keep last value			
1	Channel 1	Write Multiple Registers (Function Code 16)	Cyclic, t#100ms				16#0000	1	

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 , page 168. • **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.
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**. For more information on I/O mapping, refer to EcoStruxure Automation Expert - Motion / 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**

ControlChannel: Enables or Disables a Communication Channel

Function Description

This function allows you to enable or disable a communication channel.

A channel managed by this function is reinitialized to its default value after a reset (cold/warm).

After a stop or after a start, the channel remains disabled if it was disabled before.

On the contrary, after a reset, the channel is enabled even if it was disabled before.

In the case of the TM3BCSL Modbus Serial Line bus coupler, there are multiple, separate and independent communication channels.

⚠ WARNING

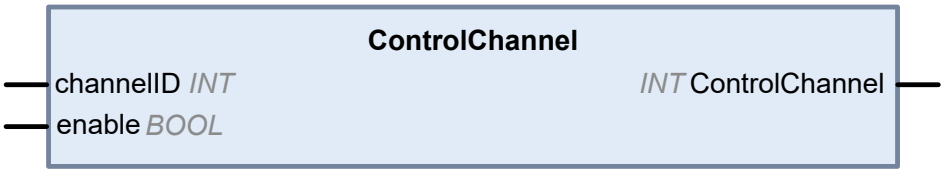
UNINTENDED EQUIPMENT OPERATION

Ensure that the Modbus serial line communication channels of the TM3BCSL bus coupler are set to the same state, either enabled or disabled.

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

NOTE: Use *ChannelID* value -1 to apply the *ControlChannel* on all channels configured on the TM3BCSL Modbus Serial Line bus coupler.

Graphical Representation



I/O Variable Description

This table describes the input variables:

Input	Type	Comment
ControlChannel	INT	Return 0 on success, a negative value on error.
ChannelID	INT	The channel number (visible in the first column of the configuration page). Or -1 to apply the command on all channels of this device.

This table describes the output variable:

Output	Type	Comment
Enable	BOOL	Enable or disable command.

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 Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

• Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide

For further information, refer to Modem Library Guide (see Modem Functions, Modem Library Guide).

CANopen Configuration

Introduction

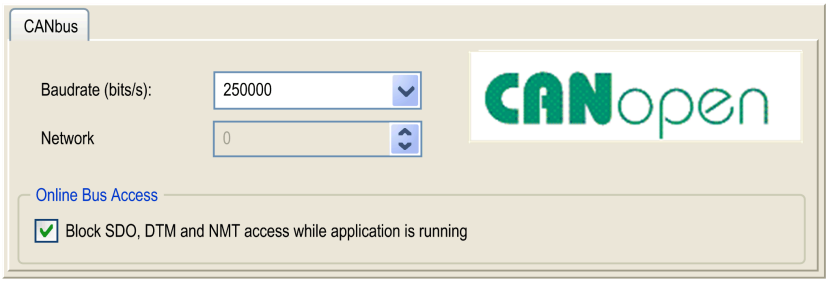
This chapter describes how to configure the CAN interface offered within the controller.

In order to use the CANopen interface, the M241 Logic Controller has 1 CAN connection (CAN0) that supports a CANopen manager.

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

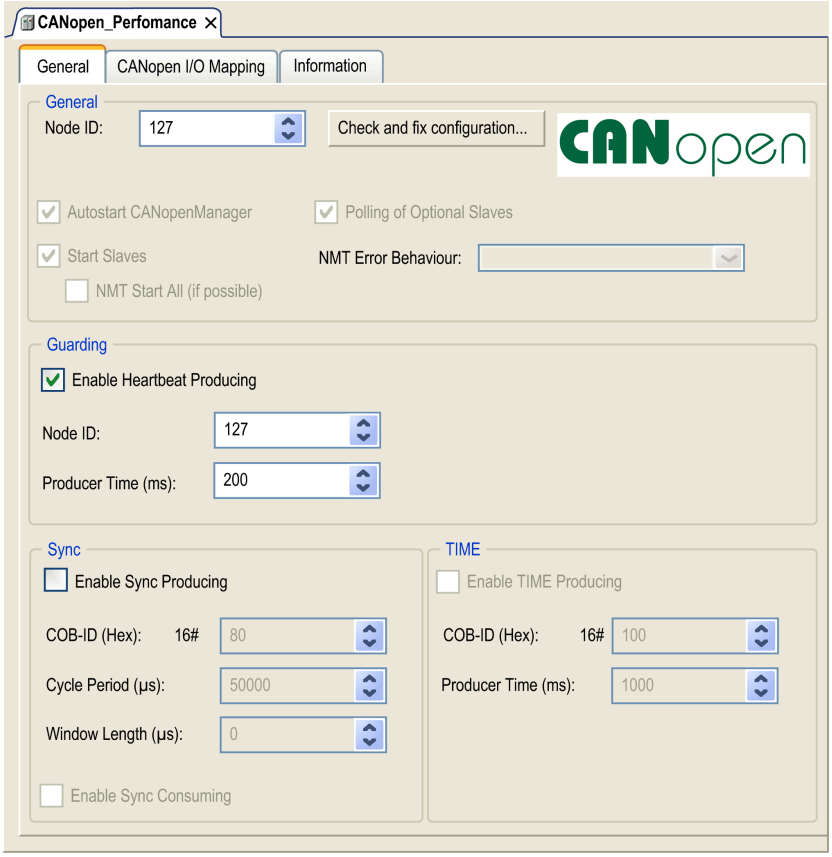
UNINTENDED EQUIPMENT OPERATION

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 Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide • Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / 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, the software 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 Automation Expert - Motion / 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 Receive PDO (RPDO): 252
- Maximum number of Transmit 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.

J1939 Configuration

J1939 Interface Configuration

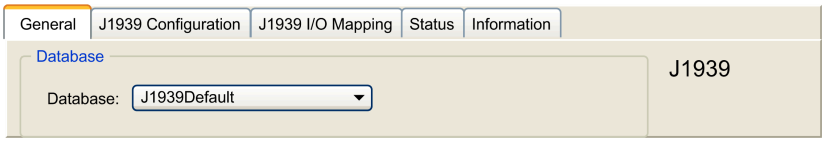
CAN Bus Configuration

To configure the **CAN** bus of your controller, refer to CAN Bus Configuration, page 171.

The CAN bus format is CAN2.0B for J1939.

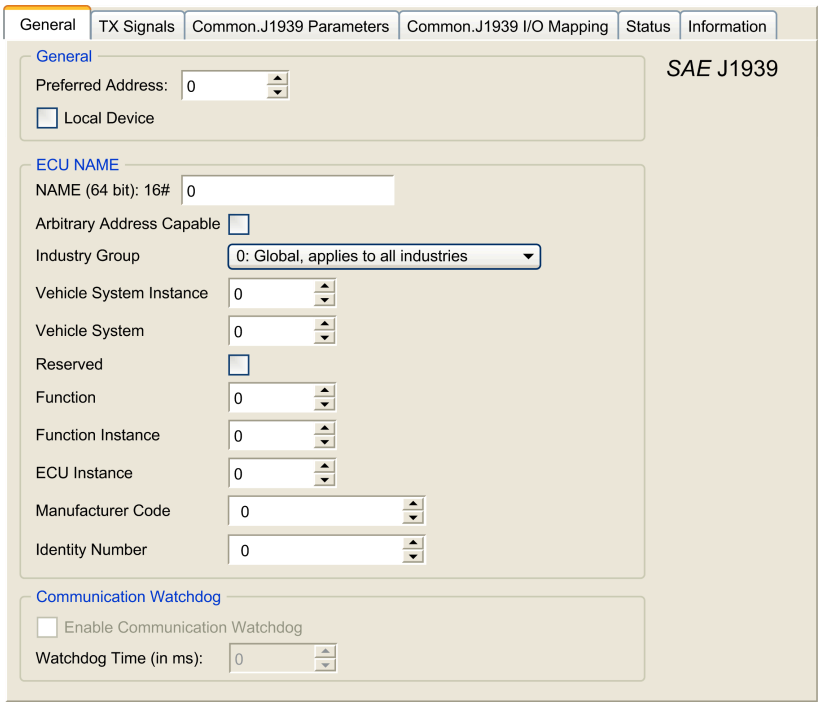
J1939 Manager Creation and Configuration

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 Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide • Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / 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 the *CoDeSys CANbus* document found in the online help (**Software > Communication > Device Editors > CoDeSys CANbus**).

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 Hardware Catalog found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide • Using the Contextual Menu or Plus Button found in the EcoStruxure Automation Expert - Motion / 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, page 176.

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]*256 +
    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 the *CoDeSys CANbus* document found in the online help (**Software > Communication > Device Editors > CoDeSys CANbus**).

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, page 175.
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.

OPC UA Server Configuration

Introduction

This chapter describes how to configure the OPC UA server of the M241 Logic Controller.

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 Rights, page 64.

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. By default the OPC UA server is using encrypted communication with maximum security settings.

Optionally, you can customize the OPC UA server name via the post configuration. Refer to Parameters, page 188.

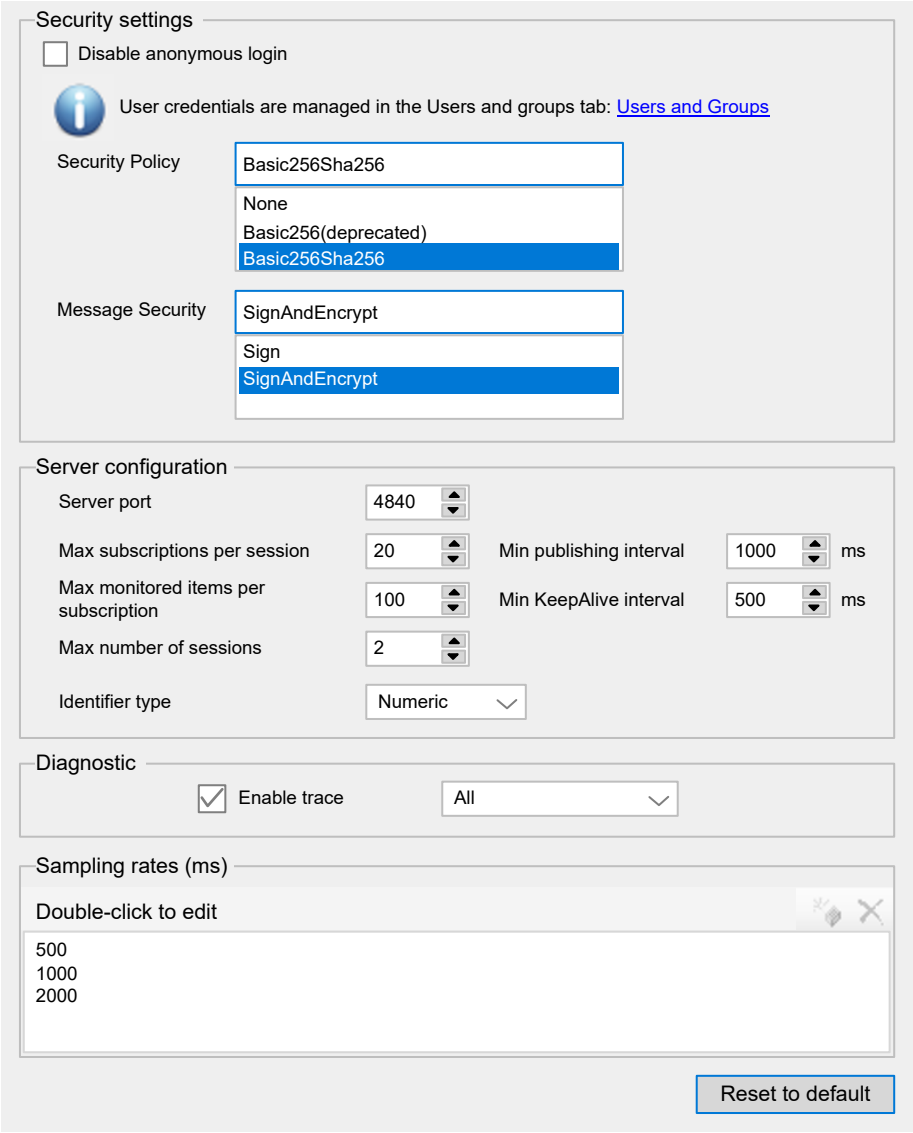
Accessing the OPC UA Server Configuration Tab

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:

Security Settings

Parameter	Value	Default value	Description
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.
Security Policy	None Basic256 (deprecated) (1) Basic256-Sha256	Basic256-Sha256	This drop-down menu allows you to sign and encrypt the data you send and receive.
Message Security	None Sign SignAndEncrypt	SignAndEn-crypt	The messages are related to the selected **Security Policy**.
(1) Security policies marked as deprecated are policies which no longer afford an acceptable level of security.			

Server Configuration

Parameter	Value	Default value	Description
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

Diagnostic

Parameter	Value	Default value	Description
Enable trace	Enabled/ disabled	Enabled	Select this checkbox to include OPC UA diagnostic messages in the controller log file found in the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide. Traces are available from the **Log** tab or from the System Log File, page 141 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)

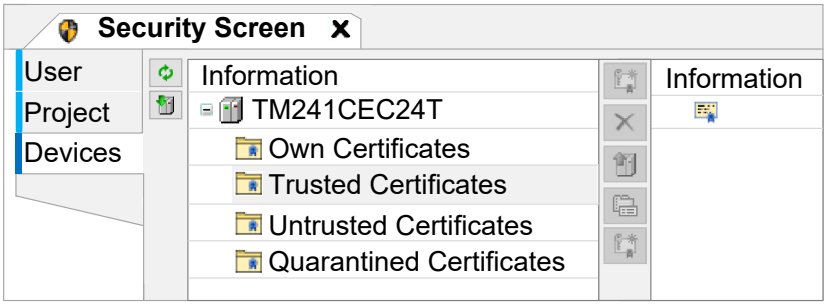
Parameter	Value	Default value	Description
Sampling rates (ms)	200...5000	500 1000 2000	The sampling rate indicates a time interval, in milliseconds (ms). When this interval has elapsed, the server sends the notification package to the client. The sampling rate can be shorter than the publishing interval, 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 sampling 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.

Client Certificates Management Actions

The Security Screen allows you to determine which OPC UA client certificates are trusted by the OPC UA server.

To access the Security Screen, use the **View > Security Screen** command.



The first attempt of the client connection is unsuccessful as the certificate is quarantined. To allow the OPC UA server to accept a client certificate, proceed as follows:

NOTE: Start from step 6 if you already have the trusted certificate.

Step	Action
1	In the **Devices** tab of the **Security Screen**, click the **Refresh** button to update the list of available devices and their certificate store.
2	Select the device entry (device name) on the left side.
3	Open the **Quarantined Certificates**. Quarantined certificates are listed in the table with the symbol.
4	Click the **Properties** button to show details for the selected certificate. Inspect the certificate details. If it is trusted, proceed to the next step.
5	Upload the selected certificate from the device and save it to your PC by clicking the **Upload** button .
6	Open the **Trusted Certificates**. Trusted certificates are listed in the table with the symbol. (By default, no certificate is available).
7	Click the **Download** button and select your trusted certificate. **Result:** The downloaded certificate is stored and listed in the **Trusted Certificates** table. The OPC UA server is then able to accept the client connection with the correct Security Settings.

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.

This table describes IEC variable Base Types versus OPC UA Data Types:

IEC variable Base Types	OPC UA Data Types
BOOL, BIT	Boolean
BYTE, USINT	Byte
INT	Int16
WORD, UINT	Uint16
DINT	Int32
DWORD, UDINT	Uint32
LINT	Int64
LWORD, ULINT	Uint64
REAL	Float
LREAL	Double
STRING	String
SINT	SByte

Bit memory variables (%MX) cannot be selected. In addition to IEC base data types, the OPC UA server can also expose OPC UA variables from IEC symbols that are composed of the following complex types:

- Arrays and Multi-Dimensional Arrays. These are limited to 3 dimensions.
- Structured data types, and nested structured data types. As long as they are not composed of a UNION field.

Displaying the List of Variables

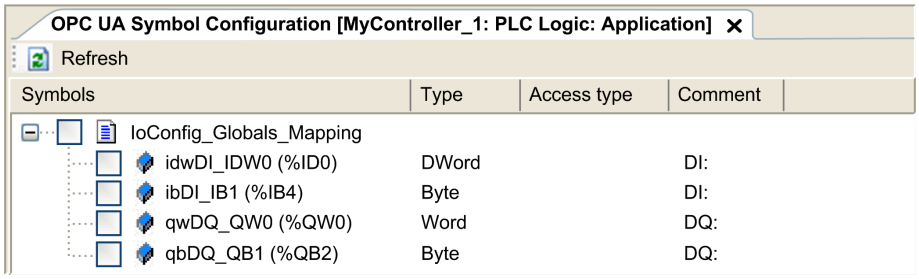
To display the list of variables:

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, page 28).

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 toggle through the access rights of the symbol: • read-only ( ) (default) • write-only ( ) • 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

As an example, the following provides capacity and performance information for the OPC UA server of the M241 Logic Controller. Design considerations are also provided to help you consider the optimal conditions for the performance of the OPC UA server. Of course, the performance realized by your application depend on many variables and conditions, and may differ from this example.

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						
Configuration	**Number of Symbols**					
	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			
Configuration	**Number of Symbols**		
	100	**400**	**1000**
Small	214 ms	227 ms	254 ms
Medium	224 ms	250 ms	292 ms
Large	324 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			
Configuration	**Number of Symbols**		
	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			
Configuration	**Number of Symbols**		
	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, page 78.

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, page 33 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.

Post Configuration

Introduction

This chapter describes how to generate and configure the post configuration file of the Modicon M241 Logic Controller.

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.

OPC UA parameters:

- Server name

  NOTE: The following characters are supported by the server name parameter: **a...z A...Z 0...9 - _**

  The length is limited to 30 characters.

Ethernet parameters:

- IP Address
- Subnet Mask
- Gateway Address
- Transfer Rate
- IP Config Mode
- Device Name
- IP Master Address, page 123

Serial Line parameters, for each serial line in the application (embedded port or PCI module):

- Baud rate
- Parity
- Data bits
- Stop bit

FTP:

- FTP encryption setting parameter

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, page 49
- A Reset Cold command, page 50
- A reboot, page 52
- An application download, page 54

Refer to Controller States and Behaviors, page 39 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
```

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.
- 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 Automation Expert - Motion and 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 Automation Expert - Motion or 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, page 196 (with the proper script)
- Download through the FTP server, page 102
- Download with EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert controller device editor, page 58

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, page 128.

To modify the Post Configuration file in the controller with EcoStruxure Automation Expert - Motion or 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, page 102
- Online with EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert controller device editor, page 58, **Files** tab

For more information on **Files** tab of the Device Editor, refer to the EcoStruxure Automation Expert - Motion / 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, page 49
- A Reset Cold command, page 50
- A reboot, page 52
- An application download, page 54

Post Configuration Example

Post Configuration File Example

```
# TM241CE40T/U / FTP Encryption
# 1=encryption enforced, 0 otherwise
.param[1106] = 1

# TM241CE40T/U / OPCUA server name
# Only ASCII letters, digits, '-' and '_', 30 char max
.param[1204] = 'M241_server'

# TM241CE40T/U / Ethernet_1 / IPAddress
# Ethernet IP address
id[45000].pos[8].id[111].param[0] = [85, 100, 108, 241]

# TM241CE40T/U / Ethernet_1 / SubnetMask
# Ethernet IP mask
id[45000].pos[8].id[111].param[1] = [255, 255, 0, 0]

# TM241CE40T/U / Ethernet_1 / GatewayAddress
# Ethernet IP gateway address
id[45000].pos[8].id[111].param[2] = [0, 0, 0, 0]

# TM241CE40T/U / Ethernet_1 / IPConfigMode
# IP configuration mode: 0:FIXED 1:BOOTP 2:DHCP
id[45000].pos[8].id[111].param[4] = 0

# TM241CE40T/U / Ethernet_1 / DeviceName
# Name of the device on the Ethernet network
id[45000].pos[8].id[111].param[5] = 'my_Device'

# TM241CE40T/U / Serial_Line_1 / Serial Line Configuration /
Baudrate
```

```
# Serial Line Baud Rate in bit/s

id[45000].pos[8].id[40101].param[10000].Bauds = 115200


# TM241CE40T/U / Serial_Line_1 / Serial Line Configuration /
Parity

# Serial Line Parity (0=None, 1=Odd, 2=Even)

id[45000].pos[9].id[40101].param[10000].Parity = 0


# TM241CE40T/U / Serial_Line_1 / Serial Line Configuration /
DataBits

# Serial Line Data bits (7 or 8)

id[45000].pos[9].id[40101].param[10000].DataFormat = 8


# TM241CE40T/U / Serial_Line_1 / Serial Line Configuration /
StopBits

# Serial Line Stop bits (1 or 2)

id[45000].pos[9].id[40101].param[10000].StopBit = 1


# TM241CE40T/U / Serial_Line_2 / Serial Line Configuration /
Baudrate

# Serial Line Baud Rate in bit/s

id[45000].pos[10].id[40102].param[10000].Bauds = 19200


# TM241CE40T/U / Serial_Line_2 / Serial Line Configuration /
Parity

# Serial Line Parity (0=None, 1=Odd, 2=Even)

id[45000].pos[10].id[40102].param[10000].Parity = 2


# TM241CE40T/U / Serial_Line_2 / Serial Line Configuration /
DataBits

# Serial Line Data bits (7 or 8)

id[45000].pos[10].id[40102].param[10000].DataFormat = 8


# TM241CE40T/U / Serial_Line_2 / Serial Line Configuration /
StopBits

# Serial Line Stop bits (1 or 2)

id[45000].pos[10].id[40102].param[10000].StopBit = 1
```

Connecting a Modicon M241 Logic Controller to a PC

Introduction

This chapter shows how to connect a Modicon M241 Logic Controller to a PC.

Connecting the Controller to a PC

Overview

To transfer, run, and monitor the applications, you can use either a USB cable or an Ethernet connection (for those references that support an Ethernet port) to connect the controller to a computer with EcoStruxure Automation Expert - Motion or EcoStruxure Machine Expert installed.

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 non-volatile 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 *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

Cable Reference	Details
BMXXCAUSBH018	Grounded and shielded, this USB cable is suitable for long duration connections.
TCSXCNAMUM3P	This USB cable is suitable for short duration connections such as quick updates or retrieving data values.

NOTE: You can only connect 1 controller or any other device associated with EcoStruxure Automation Expert - Motion or 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.

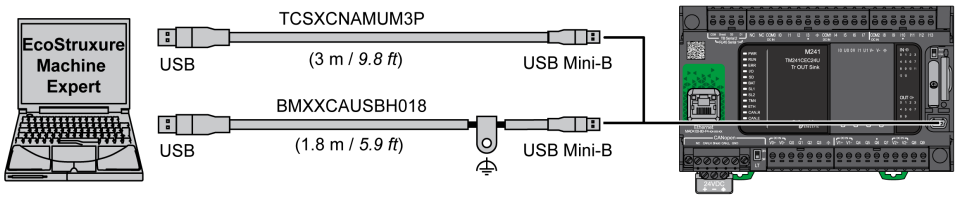
⚠ WARNING

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.

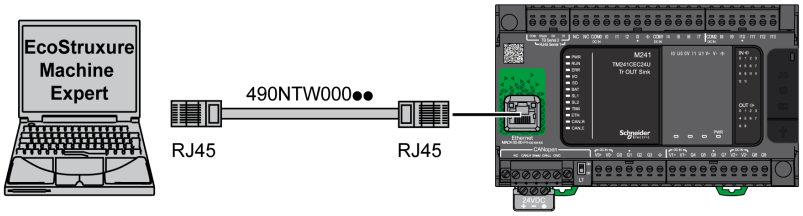


To connect the USB cable to your controller, follow the steps below:

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 protective cover for the USB Mini-B slot on the controller.
4	Connect the Mini-B connector of your USB cable to the controller.

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.

SD Card

Introduction

This chapter describes how to transfer firmware, application, using an SD card to the Modicon M241 Logic Controller.

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, page 146.
- 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, page 126 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, page 197 (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, page 22.

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, page 139.

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 `ExternalCmd` object on **ExternalMedia** group. Refer to Default Users and Groups, page 66. As a result, cloning will be not allowed without using **FB_ ControlClone**.

For more details about this function block, refer to the Modicon M241 Logic Controller, System Functions and Variables, PLCSystem Library 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, page 139) of the source controller before doing the clone operation.

For more details about Access Rights, refer to the EcoStruxure Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.

This procedure describes how to upload the application stored in the source 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), optionally followed by up to 6 unaccented alphanumeric characters (a...z, A...Z, 0...9).
2	Select if you want to clone the **Users Rights**. Refer to the **Clone Management** subpage of the Web server, page 139.
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 target controller:

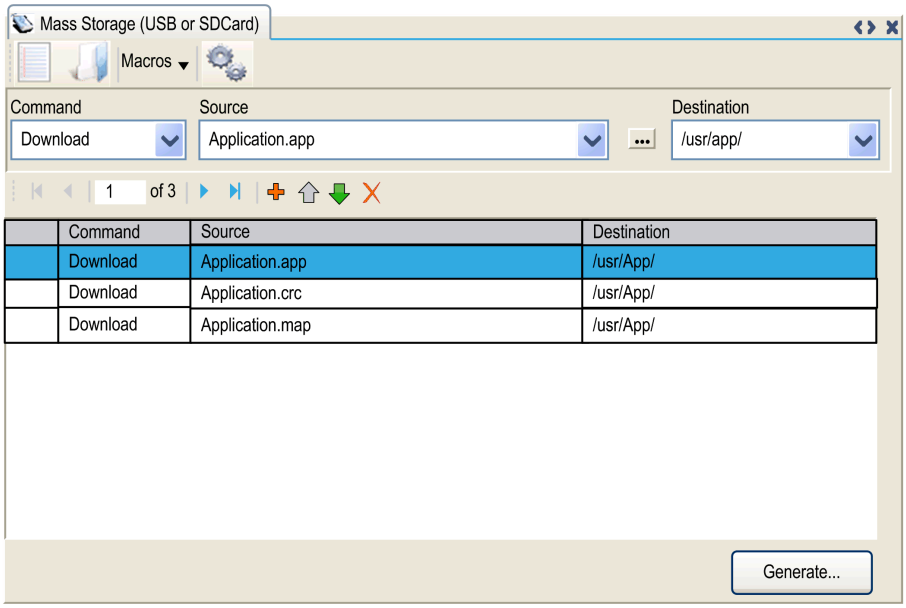
Step	Action
1	Remove power from the controller.
2	Insert the SD card into the controller.
3	Restore power to the controller. **Result:** The download procedure starts and the SD LED is flashing during this procedure.
4	Wait until the end of the download: • If the **SD** LED (green) is ON, and the **ERR** 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 Automation Expert - Motion / 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 User Rights 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.

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.	
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, page 179 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:

```
Format "/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 operation starts automatically. During the operation, the **PWR** and **I/O** LEDs are ON and the **SD** LED flashes regularly.
4	Wait until the operation is completed. **Result:** • The SD LED is ON if the operation is successful. • The ERR LED is ON and the controller does not start if an error is detected.
5	Remove the SD card from the controller. **NOTE:** The controller reboots with the default user rights.

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.

Firmware Management

Overview

The firmware update for the controller and the expansion modules are available on the Schneider Electric website (in .zip or .seco format).

Updating Modicon M241 Logic Controller Firmware

Introduction

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 application program in the device, including the configuration files, the user management, the user rights, the certificates and the Boot Application in non-volatile 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 CoDeSys protocol by default when new or when you update the controller firmware. The CoDeSys 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 box, click the **Read from....** controller button. **Result**: The **Controller selection** dialog box 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 box.
3	Click the button **New / Process...** and then **Update firmware...**. **Result**: The dialog box 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 box.
5	On the **Home** dialog box, click the **Write on....** controller button. **Result**: The **Controller selection** dialog box opens.
6	Select the required connection type and controller and click the **Write** button. **Result**: The **Write Device User Rights Management** dialog box opens.
7	On the **Write Device User Rights Management** dialog box, select an option to handle the user rights management on the controller: 7a: **Keep existing user rights management on the controller** option. 7b: **Overwriting existing user rights management on the controller by the one on the current image** option. 7c: **Reset user rights management on the controller to default (factory settings)** option.
8	Click **OK**. **Result**: The image is transmitted from your computer to the controller. After the transmission, you are automatically returned to the **Home** dialog box automatic reboot.

For more information about the firmware update and creating a new flash disk with firmware, refer to Project Settings - Firmware Update and Non-volatile Memory Organization, page 25.

Updating TM3 Expansion Modules Firmware

Downloading Firmware to TM3 Expansion Modules

The firmware can be updated in:

- TM3XHSC202 and TM3XHSC202G
- TM3D• and TM3XTYS4 with firmware version ≥ 28 (SV ≥ 2.0), except TM3DM16R and TM3DM32R (which are not updatable)
- TM3A• and TM3T• with firmware version ≥ 26 (SV ≥ 1.4)

NOTE: The software 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 Automation Expert - Motion or 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.
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 approximately 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 a targeted module was not updated successfully, or there are no message logger messages for all the targeted modules, see the Recovery Procedure, page 206 below. 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 Automation Expert - Motion or EcoStruxure Machine Expert, or by creating and executing a script containing the following command: `Delete "usr/TM3fwupdate/*"`
13	After the update(s), remove power from the controller (and TM3XREC1 receiver module, if any).
14	Restore power to the controller (and TM3XREC1 receiver module, if any). **Result**: The module(s) is (are) updated.

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 approximately 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.
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 approximately 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.

Compatibility

Software and Firmware Compatibilities

Introduction

Software and Firmware compatibilities are described in the Compatibility and Migration User Guide.

Appendices

What's in This Part

How to Change the IP Address of the Controller.....210
Functions to Get/Set Serial Line Configuration in User Program.....212
Controller Performance.....216

Overview

This appendix lists the documents necessary for technical understanding of the Modicon M241 Logic Controller Programming Guide.

How to Change the IP Address of the Controller

What's in This Chapter

changeIPAddress: Change the IP Address of the Controller 210

changeIPAddress: Change the IP Address of the Controller

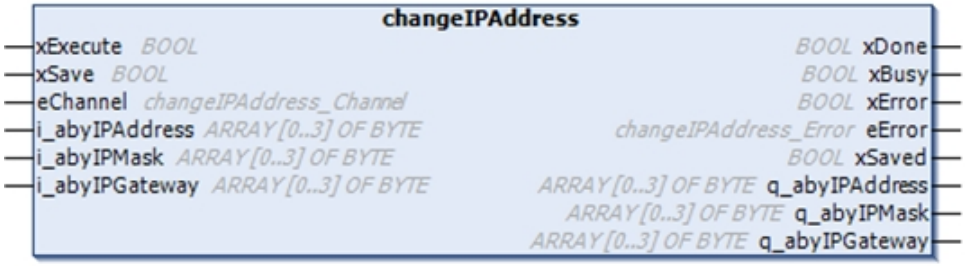
Function Block Description

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, page 97.

NOTE: For more information on the function block, use the **Documentation** tab of the Library Manager Editor. For the use of this editor, refer to 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`	`changeIPAddress_Channel`	The input `eChannel` is the Ethernet port to be configured. Depending on the number of the ports available on the controller in `changeIPAddress_Channel` (0 or 1). See `changeIPAddress_Channel`: Ethernet port to be configured, page 211.
`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, page 100 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	changeIPAddress_Error	Error code of the detected error, page 211.
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.

Functions to Get/Set Serial Line Configuration in User Program

What's in This Chapter

GetSerialConf: Get the Serial Line Configuration 212
SetSerialConf: Change the Serial Line Configuration 213
LinkNumber: Communication Port Number 214
SERIAL_CONF: Structure of the Serial Line Configuration Data Type 215

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 Automation Expert - Motion / EcoStruxure Machine Expert Programming Guide.

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`, page 214	`Link` is the communication port number.
`PointerToSerialConf`	`PointerToSerial-Conf`, page 215	`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. Refer to the `SetSerialConf, page 214` example.

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

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`, page 214	`LinkNumber` is the communication port number.
`PointerToSerialConf`	`PointerToSerial-Conf`, page 215	`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/CoDeSys
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));
```

LinkNumber: Communication Port Number

Enumerated Type Description

The *LinkNumber* enumerated data type is a list of the available communication ports. It contains these values:

Enumerator	Value (hex)	Description
USBConsole	00	USB port not available for communication exchanges
COM1	01	Serial COM 1 (embedded serial link)
COM2	02	Serial COM 2
EthEmbed and TM4ES4	03	Embedded Ethernet link and TM4ES4 expansion modules
CANEmbed	04	Embedded CANopen link
COM3	05	Serial COM 3

If one serial PCI module is installed, then the serial PCI module link is COM2, regardless of the physical PCI slots used.

If two serial PCI modules are installed, then the serial PCI module inserted in the left PCI slot is COM2 and the serial PCI module inserted in the right PCI slot is COM3.

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 CoDeSys protocol (see `CodesysCompliant` below)
		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` (polarization 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 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.
`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: CoDeSys protocol (when `Protocol` = 0)
`CodesysNetType`	`BYTE`	Not used

Controller Performance

What's in This Chapter

Processing Performance.....216

This chapter provides information about the Modicon M241 Logic Controller processing 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 input:

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

analog output:

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

application 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.

application:

A program including configuration data, symbols, and documentation.

ARP:

(*address resolution protocol*) An IP network layer protocol for Ethernet that maps an IP address to a MAC (hardware) address.

ASIC:

(*application specific integrated circuit*) A silicon processor (chip) custom designed especially for an application.

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.

cyclic task:

The cyclic scan time has a fixed duration (interval) specified by the user. If the current scan time is shorter than the cyclic scan time, the controller waits until the cyclic scan time has elapsed before starting a new scan.

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.

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.

G

GRAFCET:

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

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

H

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

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).

I/O:

(*input/output*)

ICMP:

(*Internet control message protocol*) Reports errors detected and provides information related to datagram processing.

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.

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.

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.

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).

LED:

(*light emitting diode*) An indicator that illuminates under a low-level electrical charge.

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.

LREAL:

(*long real*) A floating-point number encoded in a 64-bit format.

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.

NOTE:

MDT:

(*master data telegram*) On Sercos bus, an MDT telegram is sent by the master once during each transmission cycle to transmit data (command values) to the servo drives (slaves).

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).

Modbus:

The protocol that allows communications between many devices connected to the same network.

monitored items:

In OPC UA, the items of data (samples) made available by the OPC UA server that clients subscribe to.

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.

ms:

(*millisecond*)

%MW:

According to the IEC standard, %MW represents a memory word register (for example, a language object of type memory word).

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.

NVM:

(*Non-volatile memory*) A non-volatile memory that can be overwritten. It is stored on a special EEPROM that can be erased and reprogrammed.

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*.

OS:

(*operating system*) A collection of software that manages computer hardware resources and provides common services for computer programs.

P

PCI:

(*peripheral component interconnect*) An industry-standard bus for attaching peripherals.

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.

RSTP:

(*rapid spanning tree protocol*) A high-speed network protocol that builds a loop-free logical topology for Ethernet networks.

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.

RTP:

(*real-time process*) The real-time process is the most important system task. It is responsible for executing all real-time tasks at the correct time. Real-time processing is triggered by the Sercos real-time bus cycle.

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.

STOP:

A command that causes the controller to stop running an application program.

string:

A variable that is a series of ASCII characters.

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.

symbol:

A string of a maximum of 32 alphanumeric characters, of which the first character is alphabetic. It allows you to personalize a controller object to facilitate the maintainability of the application.

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.

NOTE:

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 161

C

changeIPAddress 210
  changing the controller IP address 210
changeModbusPort
  command syntax 126
  script example 127
ControlChannel 169
  Enables or disables a communication channel ... 169
controller configuration
  communication settings 60
Controller Configuration
  PLC Settings 61
  Services 62
cyclic data exchanges, generating EDS file for 106

D

Data Types
  LinkNumber 214
DHCP server 154
Download application 54

E

ECU, creating for J1939 175
EDS file, generating 106
embedded functions configuration
  embedded HSC configuration 81
  embedded I/O configuration 74
  embedded pulse generators configuration 83
Enables or disables a communication channel
  ControlChannel 169
Ethernet
  changeIPAddress function block 210
  FTP Server 102
  Modbus TCP Client/Server 101
  Modbus TCP slave device 122
  Services 95
  SNMP 105
  Web server 128
EtherNet
  EtherNet/IP device 105
EtherNet/IP Adapter 105
ExecuteScript example 127
External Event 35

F

Fast Device Replacement 154
features
  key features 13
file transfer with SD card 196
firewall
  configuration 145
  default script file 145
  script commands 146
firmware
  downloading to TM3 expansion modules 204
FTP client 104
FTP Server
  Ethernet 102
FTPRemoteFileHandling library 104

G

GetSerialConf
  getting the serial line configuration 212

H

Hardware Initialization Values 47

I

I/O bus configuration 91
I/O configuration general information
  general practices 87
Industrial Ethernet
  overview 150
IP address
  changeIPAddress 210

J

J1939
  creating ECU for 175
  interface configuration 174

K

KeepAlive (OPC UA) 178
KeepAlive interval (OPC UA) 181

L

libraries 20
Libraries
  FTPRemoteFileHandling 104
LinkNumber 214
  Data Types 214

M

M2•• communication
  GetSerialConf 212
  LinkNumber 214
  SERIAL_CONF 215
  SetSerialConf 213
Memory Mapping 22
Modbus
  Protocols 101
Modbus Ioscanner 162
Modbus Manager 157
Modbus TCP Client/Server
  Ethernet 101
Modbus TCP port, changing 126
monitored items (OPC UA) 178

O

OPC UA server
  configuration 179
  KeepAlive interval 181
  overview 178

publishing interval 181
sampling interval 181
selecting symbols 184
symbols configuration 183
Output Behavior 47
Output Forcing 47

P

Post Configuration 188
baud rate 188
data bits 188
device name 188
Example 191
file management 189
FTP 188
gateway address 188
IP address 188
IP configuration mode 188
IP master name 188
parity 188
presentation 188
station address 188
stop bit 188
subnet mask 188
transfer rate 188
programming languages
IL, LD, Grafcet 13
protocols
SNMP 105
Protocols 95
IP 97
Modbus 101
publishing interval (OPC UA) 178, 181

R

Reboot 52
Remanent variables 56
Reset cold 50
Reset origin 50
Reset origin device 51
Reset warm 49
Run command 48

S

sampling interval (OPC UA) 178, 181
script commands
firewall 146
script file
syntax rules 196
SD card
commands 196
serial line
ASCII Manager 161
GetSerialConf 212
Modbus Manager 157
SERIAL_CONF 215
SetSerialConf 213
SERIAL_CONF 215
SetSerialConf 213
setting the serial line configuration 213
SNMP
Ethernet 105
protocols 105
Software Initialization Values 47
State diagram 39
Stop command 48
symbols (OPC UA) 183

T

Task
Cyclic task 33
Event task 34
External Event Task 35
Freewheeling task 34
Types 33
Watchdogs 35
TM3 analog I/O modules
downloading firmware to 204

U

updating the firmware of TM3 expansion modules 204

W

Web server
Ethernet 128

Schneider Electric
35 rue Joseph Monier
92500 Rueil Malmaison
France

+ 33 (0) 1 41 29 70 00

www.se.com

As standards, specifications, and design change from time to time, please ask for confirmation of the information given in this publication.

© 2024 Schneider Electric. All rights reserved.

EIO0000003059.08