Modicon M251
Logic Controller

User Guide

08/2020

EIO0000004273.00

www.schneider-electric.com

Schneider Electric

Table of Contents

Modicon M251

Logic Controller
Programming Guide

12/2019

EIO0000003089.02

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

The purpose of this document is to help you to program and operate your Modicon M251 Logic Controller with the EcoStruxure Machine Expert software.

NOTE: Read and understand this document and all related documents *(see page 9)* before installing, operating, or maintaining your Modicon M251 Logic Controller.

The Modicon M251 Logic Controller users should read through the entire document to understand all features.

Validity Note

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

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

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

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert Programming Guide	*EIO0000002854 (ENG)* *EIO0000002855 (FRE)* *EIO0000002856 (GER)* *EIO0000002858 (SPA)* *EIO0000002857 (ITA)* *EIO0000002859 (CHS)*
Modicon M251 Logic Controller Hardware Guide	*EIO0000003101 (ENG)* *EIO0000003102 (FRE)* *EIO0000003103 (GER)* *EIO0000003104 (SPA)* *EIO0000003105 (ITA)* *EIO0000003106 (CHS)*

Title of Documentation	Reference Number
EcoStruxure Machine Expert Industrial Ethernet User Guide	*EIO0000003053 (ENG)* *EIO0000003054 (FRE)* *EIO0000003055 (GER)* *EIO0000003056 (SPA)* *EIO0000003057 (ITA)* *EIO0000003058 (CHS)*
Modicon TM4 Expansion Modules Programming Guide	*EIO0000003149 (ENG)* *EIO0000003150 (FRE)* *EIO0000003151 (GER)* *EIO0000003152 (SPA)* *EIO0000003153 (ITA)* *EIO0000003154 (CHS)*
Modicon TM3 Modules Configuration Programming Guide	*EIO0000003119 (ENG)* *EIO0000003120 (FRE)* *EIO0000003121 (GER)* *EIO0000003122 (SPA)* *EIO0000003123 (ITA)* *EIO0000003124 (CHS)*
Modicon TM3 Bus Coupler - Programming Guide (EcoStruxure Machine Expert)	*EIO0000003635 (ENG)* *EIO0000003636 (FRA)* *EIO0000003637 (GER)* *EIO0000003638 (SPA)* *EIO0000003639 (ITA)* *EIO0000003640 (CHS)*
Modicon TM2 Modules Configuration Programming Guide	*EIO0000003432 (ENG)* *EIO0000003433 (FRE)* *EIO0000003434 (GER)* *EIO0000003435 (SPA)* *EIO0000003436 (ITA)* *EIO0000003437 (CHS)*
Modicon M251 Logic Controller System Functions and Variables PLCSystem Library Guide	*EIO0000003095 (ENG)* *EIO0000003096 (FRE)* *EIO0000003097 (GER)* *EIO0000003098 (SPA)* *EIO0000003099 (ITA)* *EIO0000003100 (CHS)*

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

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

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

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

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

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

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

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

Chapter 1
About the Modicon M251 Logic Controller

M251 Logic Controller Description

Overview

The M251 Logic Controller has various powerful features and can service a wide range of applications.

Software configuration, programming, and commissioning are achieved with the EcoStruxure Machine Expert software described in the EcoStruxure Machine Expert Programming Guide and in the M251 Logic Controller Programming Guide.

Programming Languages

The M251 Logic Controller is configured and programmed with the EcoStruxure Machine Expert software, which supports the following IEC 61131-3 programming languages:

- IL: Instruction list
- ST: Structured text
- FBD: Function block diagram
- SFC: Sequential function chart
- LD: Ladder diagram

EcoStruxure Machine Expert software can also be used to program this controller using CFC (continuous function chart) language.

Power Supply

The power supply of the M251 Logic Controller is 24 Vdc.

Real Time Clock

The M251 Logic Controller includes a Real Time Clock (RTC) system *(see Modicon M251 Logic Controller, Hardware Guide)*.

Run/Stop

The M251 Logic Controller can be operated externally by the following:

- a hardware Run/Stop switch
- an EcoStruxure Machine Expert software command

Memory

This table describes the different types of memory:

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

Removable Storage

M251 Logic Controllers include an embedded SD card slot *(see Modicon M251 Logic Controller, Hardware Guide)*.

The main uses of the SD card are:

- Initializing the controller with a new application
- Updating the controller firmware
- Applying post configuration files to the controller
- Applying recipes
- Receiving data logging files

Embedded Communication Features

The M251 Logic Controller native communication ports include (depending on the controller reference):

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

Expansion Module and Bus Coupler Compatibility

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

M251 Logic Controllers

Reference	Digital Inputs	Digital Outputs	Communication Ports
TM251MESC	0	0	1 serial line port 1 USB mini-B programming port 1 dual port Ethernet switch 1 CANopen port
TM251MESE *(see Modicon M251 Logic Controller, Hardware Guide)*	0	0	1 serial line port 1 USB mini-B programming port 1 dual port Ethernet switch 1 Ethernet port for fieldbus

Chapter 2
How to Configure the Controller

How to Configure the Controller

Introduction

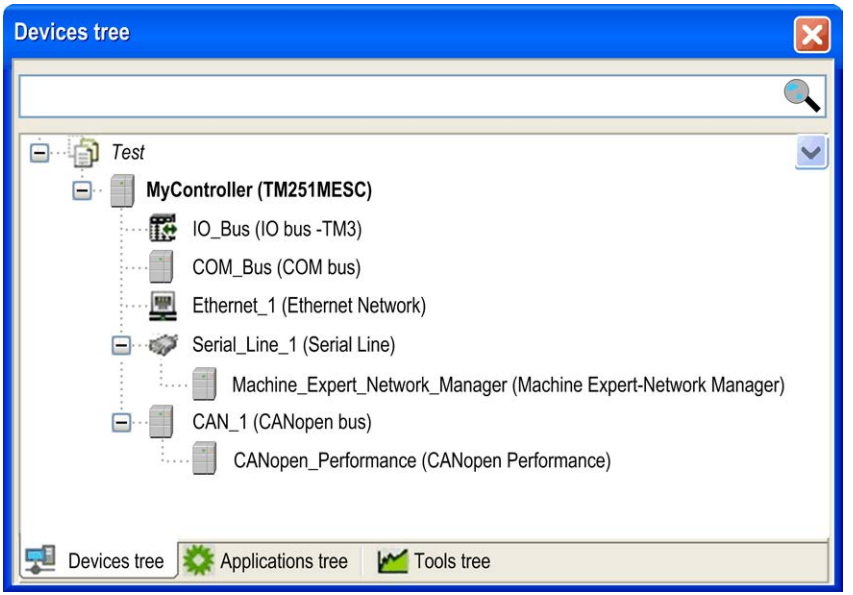
First, create a new project or open an existing project in the EcoStruxure Machine Expert software.

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

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

Devices Tree

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



Item	Use to Configure...
IO_Bus	Expansion modules connected to the logic controller
COM_Bus	Communications bus of the logic controller
Ethernet_x	Embedded Ethernet, serial line, or CANopen communications interfaces **NOTE:** Ethernet and CANopen are only available on some references.
Serial_Line_x	
CAN_x	

Applications Tree

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

Tools Tree

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

Chapter 3
Libraries

Libraries

Introduction

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

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

Modicon M251 Logic Controller

When you select a Modicon M251 Logic Controller for your application, EcoStruxure Machine Expert automatically loads these 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 that are required matching IEC61131-3 as standard POUs for an IEC programming system. The standard POUs must be tied to the project (standard.library).
Util	Analog Monitors, BCD Conversions, Bit/Byte Functions, Controller Datatypes, Function Manipulators, Mathematical Functions, Signals.
M251 PLCSystem *(see Modicon M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*	Contains functions and variables to get information and send commands to the controller system.
PLCCommunication *(see EcoStruxure Machine Expert, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	**SysMem, Standard**. These functions facilitate communications between specific devices. Most of them are dedicated to Modbus exchange. Communication functions are asynchronously processed regarding the application task that called the function.
Relocation Table *(see page 34)*	The relocation table allows you to organize data to optimize exchanges between the Modbus client and the controller, by regrouping non-contiguous data into a contiguous table of registers.
ModbusTCPIOScanner *(see EcoStruxure Machine Expert Modbus TCP, User Guide)*	TM251MESE only. Provides Modbus TCP IOScanner function blocks.

Library Name	Description
EtherNetIP Scanner *(see EcoStruxure Machine Expert EtherNet/IP, User Guide)*	TM251MESE only. Infrastructure function blocks to establish and close CIP connections and to build Explicit Messaging request over EtherNet/IP.
EtherNetIP Explicit Messaging *(see EcoStruxure Machine Expert EtherNet/IP, User Guide)*	TM251MESE only. Explicit Messaging over EtherNet/IP, to communicate with generic devices (e.g. cameras) for which EcoStruxure Machine Expert does not offer a device integration.
Additional libraries: ● 3S CANopenStack ● FDT_CANOpenDriver ● CAA CiA 405	The CAA CiA 405 library offers a set of function blocks to meet the requirements of the CiA405 for the access to the CANopen network from the application (IEC61131-3 program) of the controller (CANopen master).

Chapter 4
Supported Standard Data Types

Supported Standard Data Types

Supported Standard Data Types

The controller supports the following IEC data types:

Data Type	Lower Limit	Upper Limit	Information Content
BOOL	FALSE	TRUE	1 Bit
BYTE	0	255	8 Bit
WORD	0	65,535	16 Bit
DWORD	0	4,294,967,295	32 Bit
LWORD	0	$2^{64}-1$	64 Bit
SINT	-128	127	8 Bit
USINT	0	255	8 Bit
INT	-32,768	32,767	16 Bit
UINT	0	65,535	16 Bit
DINT	-2,147,483,648	2,147,483,647	32 Bit
UDINT	0	4,294,967,295	32 Bit
LINT	-2^{63}	$2^{63}-1$	64 Bit
ULINT	0	$2^{64}-1$	64 Bit
REAL	1.175494351e-38	3.402823466e+38	32 Bit
LREAL	2.2250738585072014e-308	1.7976931348623158e+308	64 Bit
STRING	1 character	255 characters	1 character = 1 byte
WSTRING	1 character	255 characters	1 character = 1 word
TIME	-	-	32 Bit

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

Chapter 5
Memory Mapping

Introduction

This chapter describes the memory maps and sizes of the different memory areas in the Modicon M251 Logic Controller. These memory areas are used to store user program logic, data and the programming libraries.

What Is in This Chapter?

This chapter contains the following topics:

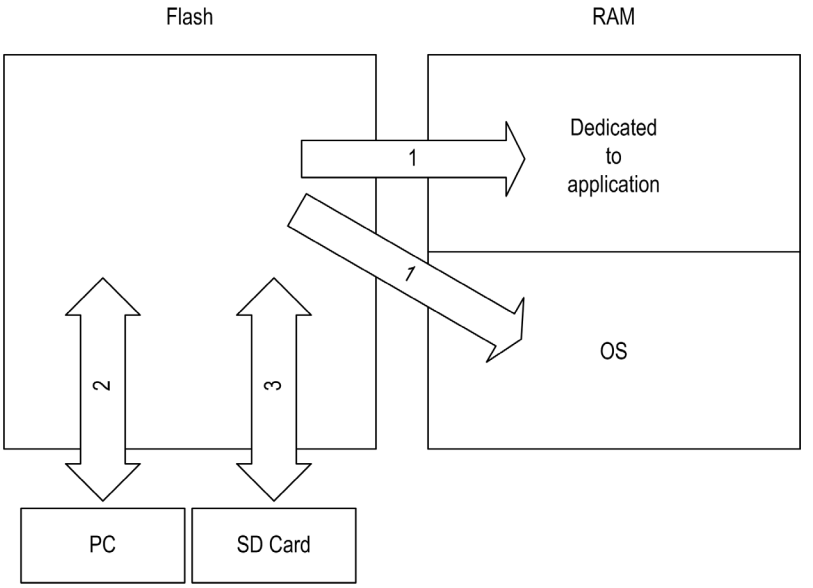
Controller Memory Organization

Introduction

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

- The Flash memory *(see page 30)* contains files (application, configuration files, and so on).
- The Random Access Memory (RAM) *(see page 28)* is used for application execution.

Files Transfers in Memory



Item	Controller State	File Transfer Events	Connection	Description
1	–	Initiated automatically at Power ON and Reboot	Internal	Files transfer from Flash memory to RAM. The content of the RAM is overwritten.
2	All states except INVALID_OS (1)	Initiated by user	Ethernet or USB programming port	Files can be transferred via: ● Web server *(see page 105)* ● FTP server *(see page 121)* ● **Controller Assistant** ● EcoStruxure Machine Expert *(see 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 Flash memory does not affect a running application. Any changes to files in Flash memory are taken into account at the next reboot.

RAM Memory Organization

Introduction

This section describes the RAM (Random Access Memory) size for different areas of the Modicon M251 Logic Controller.

Memory Mapping

The RAM size is 64 Mbytes.

The RAM is composed of 2 areas:

- dedicated application memory
- OS memory

This table describes the dedicated application memory:

Area	Element	Size
System area 192 Kbytes	System Area Mappable Addresses %MW0...%MW59999	128 Kbytes
	System and diagnostic variables (%MW60000...%MW60199) This memory is accessible through Modbus requests only. These must be read-only requests.	
	Dynamic Memory Area: Read Relocation Table *(see page 34)* (%MW60200...%MW61999) This memory is accessible through Modbus requests only. These must be read-only requests.	
	System and diagnostic variables (%MW62000...%MW62199) This memory is accessible through Modbus requests only. These can be read or write requests.	
	Dynamic Memory Area: Write Relocation Table *(see page 34)* (%MW62200...%MW63999) This memory is accessible through Modbus requests only. These can be read or write requests.	
	%MW64000…%MW65535 Reserved	
	Retain and Persistent data *(see page 30)*	64 Kbytes
User area 8 Mbytes	Symbols	Dynamic allocation
	Variables	
	Application	
	Libraries	

System and Diagnostic Variables

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

For more information on system and diagnostic variables, refer to *M251 PLCSystem Library Guide*.

Memory Addressing

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

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

Example of overlap of memory ranges:

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

NOTE: The Modbus communication is asynchronous with the application.

Flash Memory Organization

Introduction

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

File Type

The Modicon M251 Logic Controller manages the following file types:

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

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

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

File Organization

This table shows the file organization of the flash memory:

Disk	Directory	File	Content	Up/Downloaded Data Type
/sys	OS	M241M251FW1v_XX.YY (1)	Firmware of core 1	Firmware
		M241M251FW2v_XX.YY (1)	Firmware of core 2	
		Version.ini	Control file for firmware version	
	Web	Index.htm	HTML pages served by the web server for the website embedded in the controller.	Website
		Conf.htm		–
		...		–
/usr	App	Application.app	Boot application	Application
		Application.crc		–
		Application.map		–
		Archive.prj (2)	Application source	–
		settings.conf (3)	OPC UA configuration	Configuration
		OpcUASymbolConf.map (3)	OPC UA symbols configuration	Configuration
	Cfg	Machine.cfg (2)	Post configuration file *(see page 215)*	Configuration
		CodesysLateConf.cfg (2)	• Name of application to launch • Routing table (main/sub net)	Configuration

(1): v_XX.YY represents the version
(2): if any
(3): if OPC UA *(see page 207)* is configured
(4): the Fdr/FDRS directory is hidden

Disk	Directory	File	Content	Up/Downloaded Data Type
/usr	Log	*UserDefinedLogName_1*.log	All *.log files created using the data logging functions *(see SoMachine, Data Logging Functions, DataLogging Library Guide)*. You must specify the total number of files created and the names and contents of each log file.	log file
		...	–	–
		UserDefinedLogName_n.log	–	–
	Rcp		Main directory for Recipe	–
	Syslog	crashC1.txt(2) crashC2.txt(2) crashBoot.txt(2)	This file contains a record of detected system errors. For use by Schneider Electric Technical Support.	Log file
		PlcLog.txt (2)	This file contains system event data that is also visible online in EcoStruxure Machine Expert by viewing the **Log** tab of the **Controller Device Editor** *(see page 72)*.	–
		FwLog.txt	This file contains a record of firmware system events. For use by Schneider Electric Technical Support.	–
/usr	Fdr/FDRS (4) only for TM251MESE	Device1.prm	Parameter files stored by the FDR client device1	FDR *(see page 175)*
		Device2.prm	Parameter files stored by the FDR client device2	
		...	–	
/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 *(see page 207)* is configured
(4): the Fdr/FDRS directory is hidden

NOTE: For more information on libraries and available function blocks, refer to Libraries *(see page 21)*.

Files Redirection

When system, program or certain user activity creates specific file types, the M251 Logic Controller examines the file extension and automatically moves the file to a corresponding folder in flash memory.

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

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

Backup Data Logging File

Data logging files can become large to the point of exceeding the space available in the file system. Therefore, you should develop a method to archive the log data periodically on an SD card. You could split the log data into several files, for example `LogMonth1, LogMonth2`, and use the **ExecuteScript** *(see Modicon M251 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 as 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 *M251 PLCSystem Library Guide*.

Adding a Relocation Table

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

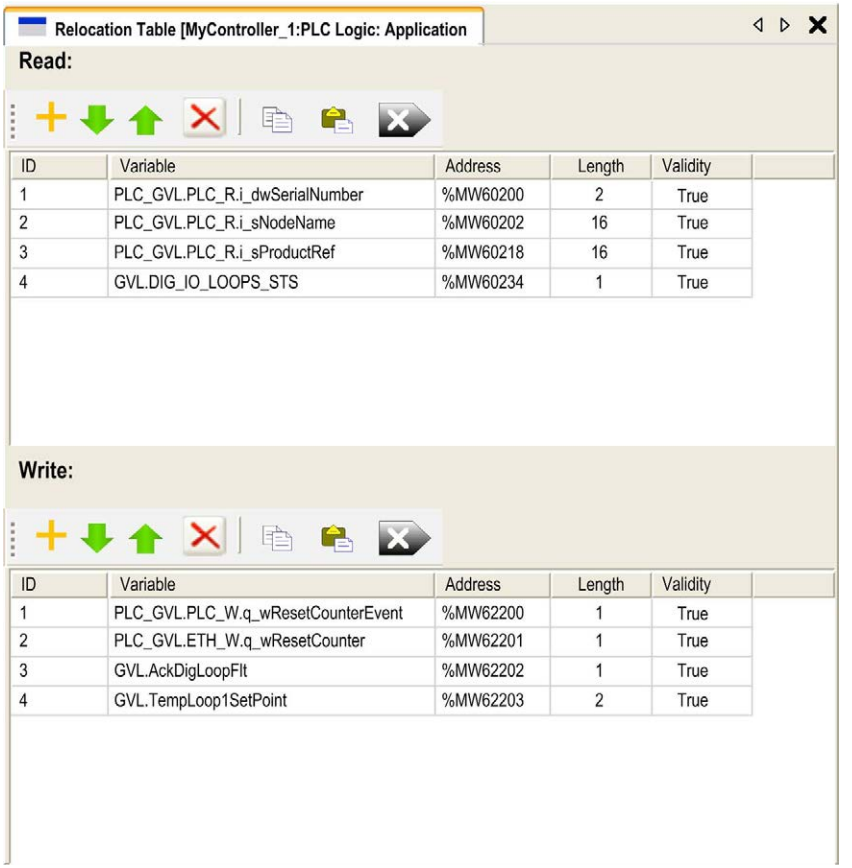
Step	Action
1	Select the **Application** node in the **Applications tree** tab.
2	Click  .
3	Click **Add other objects → Relocation Table...**. **Result:** The **Add Relocation Table** window is displayed.
4	Click **Add**. **Result:** The new relocation table is created and initialized. **NOTE:** As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

Relocation Table Editor

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

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

This picture describes the relocation table editor:



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

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

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

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

Chapter 6
Tasks

Introduction

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

The task types available are:

- Cyclic
- Freewheeling
- Event
- External event

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

What Is in This Chapter?

This chapter contains the following topics:

Maximum Number of Tasks

Maximum Number of Tasks

The maximum number of tasks you can define for the Modicon M251 Logic Controller is:

- Total number of tasks = 19
- Cyclic tasks = 5
- Freewheeling tasks = 1
- Event tasks = 8
- External event task = 1 (TM251MESC only)

Special Considerations for Freewheeling

A Freewheeling task *(see page 42)* does not have a fixed duration. In Freewheeling mode, each task scan starts when the previous scan has been completed and after a period of system processing (30% of the total duration of the Freewheeling task). If the system processing period is reduced to less than 15% for more than 3 seconds due to interruptions by other tasks, a system error is detected. For more information, refer to the System Watchdog *(see page 44)*.

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

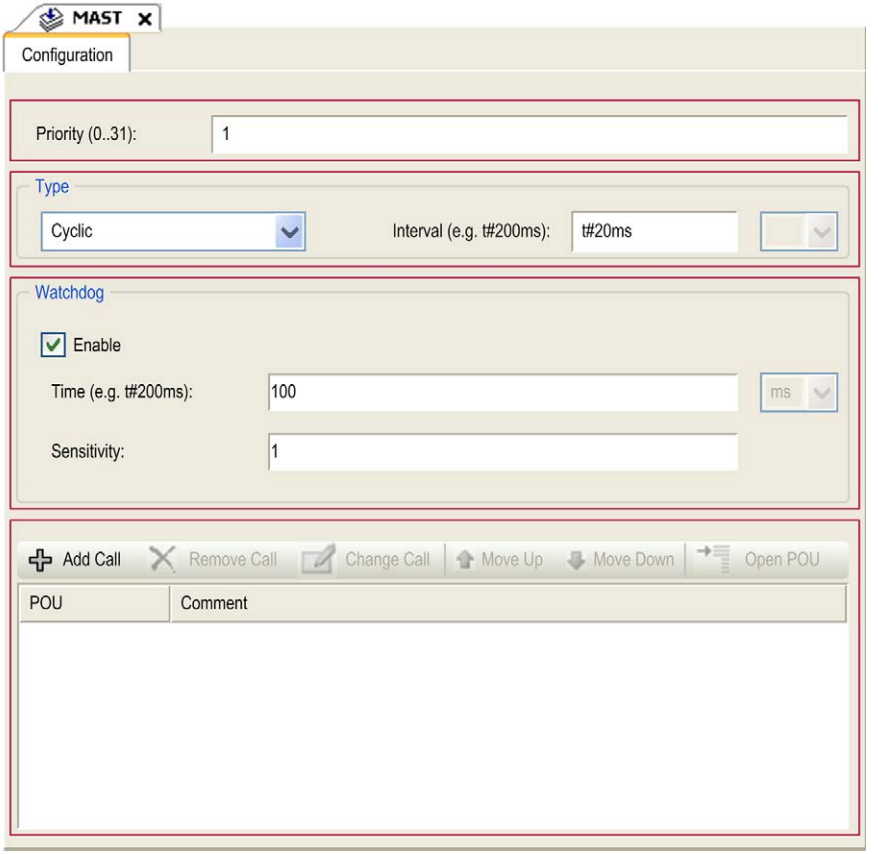
Task Configuration Screen

Screen Description

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

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

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



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

Field Name	Definition
Priority	Configure the priority of each task with a number from 0 to 31 (0 is the highest priority, 31 is the lowest). Only one task at a time can be running. The priority determines when the task will run: ● a higher priority task will pre-empt a lower priority task ● tasks with same priority will run in turn (2 ms time-slice) **NOTE:** Do not assign tasks with the same priority. If there are yet other tasks that attempt to pre-empt tasks with the same priority, the result could be indeterminate and unpredicable. For important safety information, refer to Task Priorities *(see page 45)*.
Type	These task types are available: ● Cyclic *(see page 41)* ● Event *(see page 43)* ● External *(see page 43)* ● Freewheeling *(see page 42)*
Watchdog	To configure the watchdog *(see page 44)*, 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 currently selected POU of the list by another one, use the command **Change Call**. ● **POU**s are executed in the order shown in the list. To move the **POU**s in the list, select a **POU** and use the command **Move Up** or **Move Down**. **NOTE:** You can create as many POUs as you want. An application with several small POUs, as opposed to one large POU, can improve the refresh time of the variables in online mode.

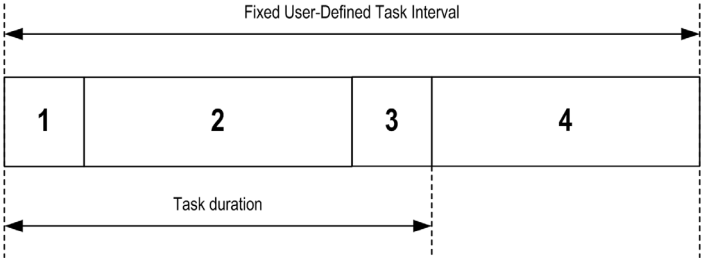
Task Types

Introduction

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

Cyclic Task

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



1. **Read Inputs:** The physical input states are written to the %I input memory variables and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variables are updated according to your application program instructions but not yet written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variables are modified with any output forcing that has been defined; however, the writing of the physical outputs depends upon the type of output and instructions used.
   For more information on defining the bus cycle task, refer to the EcoStruxure Machine Expert Programming Guide and Modicon M251 Logic Controller Settings *(see page 75)*.
4. **Remaining Interval time:** The controller firmware carries out system processing and any other lower priority tasks.

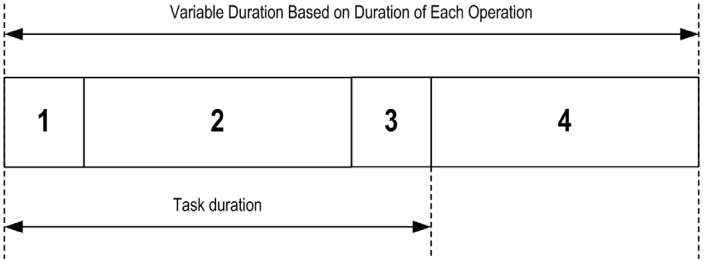
NOTE: If you define too short a period for a cyclic task, it will repeat immediately after the write of the outputs and without executing other lower priority tasks or any system processing. This will affect the execution of all tasks and cause the controller to exceed the system watchdog limits, generating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

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

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

Freewheeling Task

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



1. **Read Inputs:** The physical input states are written to the %I input memory variables and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variables are updated according to your application program instructions but not yet written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variables are modified with any output forcing that has been defined; however, the writing of the physical outputs depends upon the type of output and instructions used.
   For more information on defining the bus cycle task, refer to the EcoStruxure Machine Expert Programming Guide and Modicon M251 Logic Controller Settings *(see page 75)*.
4. **System Processing:** The controller firmware carries out system processing and any other lower priority tasks (for example: HTTP management, Ethernet management, parameters management).

NOTE: If you want to define the task interval, refer to Cyclic Task *(see page 41)*.

Event Task

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

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

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

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

External Event Task

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

The external event task is associated with the CAN Sync event. To associate the **CAN_1_SYNC** event to an external event task, select it from the **External event** dropdown list in the **Configuration** tab.

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

System and Task Watchdogs

Introduction

Two types of watchdog functionality are implemented for the Modicon M251 Logic Controller:

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

System Watchdogs

Three system watchdogs are defined for the Modicon M251 Logic Controller. They are managed by the controller firmware and are therefore sometimes referred to as hardware watchdogs in the EcoStruxure Machine Expert online help. When one of the system watchdogs exceeds its threshold conditions, an error is detected.

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

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

NOTE: System watchdogs are not configurable by the user.

Task Watchdogs

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

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

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

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

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

Task Priorities

Task Priority Configuration

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

Task Priority Suggestions

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

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

You can select the task that drives TM3 and CANopen physical exchanges. In the **PLC settings**, select **Bus cycle task** to define the task for the exchange. By default, the task is set to **MAST**. This definition at the controller level can be overridden by the I/O bus configuration *(see page 87)*. During the read and write phases, all physical I/Os are refreshed at the same time. TM3/TM2 and CANopen data is copied into a virtual I/O image during a physical exchanges phase, as shown in this figure:

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

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

Default Task Configuration

Default Task Configuration

Default Task Configuration

The MAST task can be configured in Freewheeling or Cyclic mode. The MAST task is automatically created by default in Cyclic mode. Its preset priority is medium (15), its preset interval is 20 ms, and its task watchdog service is activated with a time of 100 ms and a sensitivity of 1. Refer to Task Priorities *(see page 45)* for more information on priority settings. Refer to Task Watchdogs *(see page 44)* 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 is added to those tasks. For more information about this function, see the optional SysTask library of the system / SysLibs category of libraries.

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

Chapter 7
Controller States and Behaviors

Introduction

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

What Is in This Chapter?

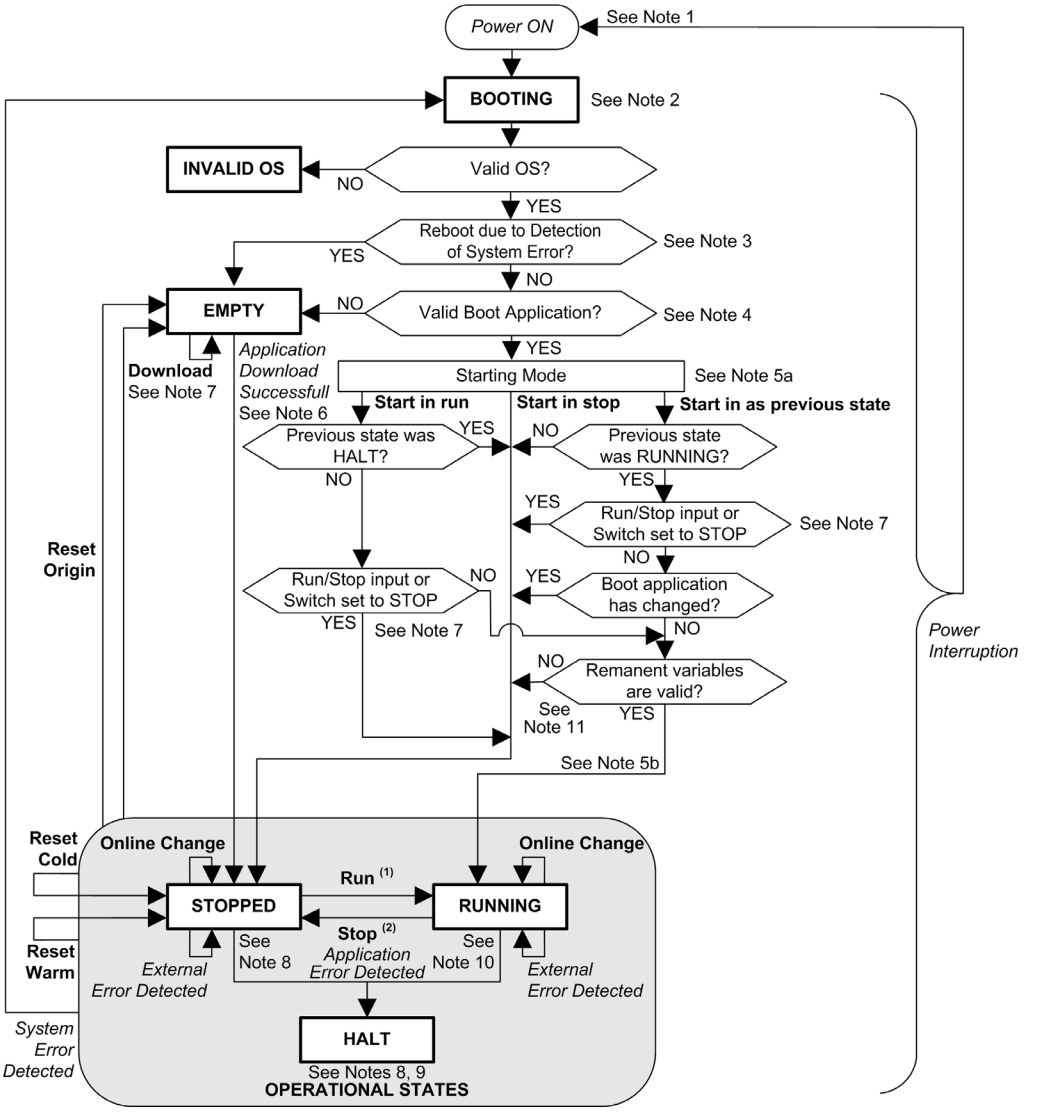
This chapter contains the following sections:

Section 7.1
Controller State Diagram

Controller State Diagram

Controller State Diagram

This diagram describes the controller operating mode:



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

(1) For details on STOPPED to RUNNING state transition, refer to Run Command *(see page 61)*.

(2) For details on RUNNING to STOPPED state transition, refer to Stop Command *(see page 61)*.

Note 1

The Power Cycle (Power Interruption followed by a Power ON) deletes all output forcing settings. Refer to Controller State and Output Behavior *(see page 58)* for further details.

Note 2

The outputs will assume their hardware initialization values.

Note 3

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

Note 4

After verification of a valid Boot application the following events occur:
- The application is loaded into RAM.
- The Post Configuration *(see page 215)* file settings (if any) are applied.

During the load of the boot application, a Check context test occurs to assure that the Remanent variables are valid. If the Check context test is invalid, the boot application will load but the controller will assume STOPPED state *(see page 64)*.

Note 5a

The **Starting Mode** is set in the **PLC settings** tab of the **Controller Device Editor** *(see page 75)*.

Note 5b

Not applicable

Note 6

During a successful application download the following events occur:
- The application is loaded directly into RAM.
- By default, the Boot application is created and saved into the Flash memory.

Note 7

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

However, there are 2 considerations in this regard:

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

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Always verify that online changes to a RUNNING application program operate as expected before downloading them to controllers. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

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

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

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

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

Note 8

The EcoStruxure Machine Expert software platform allows many powerful options for managing task execution and output conditions while the controller is in the STOPPED or HALT states. Refer to Controller States Description *(see page 53)* for further details.

Note 9

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

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

Note 10

The RUNNING state has 2 exception conditions:

- RUNNING with External Error: this exception condition is indicated by the I/O LED, which displays solid Red. You may exit this state by clearing the external error (probably changing the application configuration). No controller commands are required, but may however include the need of a power cycle of the controller. For more information, refer to I/O Configuration General Description *(see page 82)*.
- RUNNING with Breakpoint: this exception condition is indicated by the RUN LED, which displays a single flash. Refer to Controller States Description *(see page 53)* for further details.

Note 11

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

Section 7.2
Controller States Description

Controller States Description

Introduction

This section provides a detailed description of the controller states.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Never assume that your controller is in a certain controller state before commanding a change of state, configuring your controller options, uploading a program, or modifying the physical configuration of the controller and its connected equipment.
- Before performing any of these operations, consider the effect on all connected equipment.
- Before acting on a controller, always positively confirm the controller state by viewing its LEDs, 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 M251 PLCSystem library *(see Modicon M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*

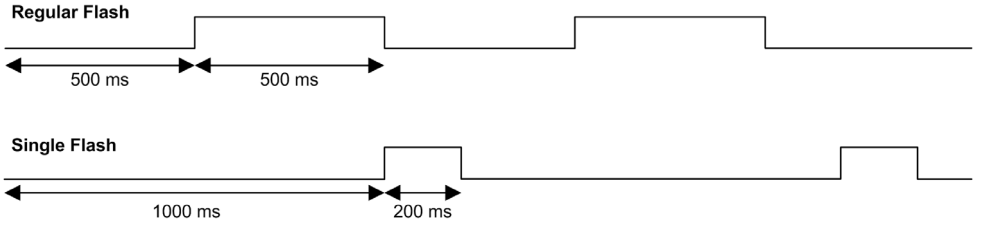
Controller States Table

The following table describes the controller states:

Controller State	Description	LED		
		RUN (Green)	ERR (Red)	I/O (Red)
BOOTING	The controller executes the boot firmware and its own internal self-tests. It then checks the checksum of the firmware and user applications.	OFF	OFF	ON
		OFF	ON	ON
		OFF	ON	OFF
INVALID_OS	There is not a valid firmware file present in the flash memory. The controller does not execute the application. Refer to the Firmware Upgrade section to restore a correct state.	OFF	Regular flash	OFF
EMPTY	The controller has no application.	OFF	Single flash	OFF
EMPTY after a system error detected	This state is the same as the normal EMPTY state. But the application is present, and is intentionally not loaded. A next reboot (power cycle), or a new application download, will restore correct state.	OFF	Fast flash	OFF
RUNNING	The controller is executing a valid application.	ON	OFF	OFF
RUNNING with breakpoint	This state is same as the RUNNING state with the following exceptions: ● The task-processing portion of the program does not resume until the breakpoint is cleared. ● The LED indications are different. ● For more information on breakpoint management, refer to EcoStruxure Machine Expert Programming Guide.	Single flash	OFF	OFF
RUNNING with external error detected	Configuration, TM3, SD card, or other I/O error detected. When I/O LED is ON, the details about the detected error can be found in `PLC_R.i_lwSystemFault_1` and `PLC_R.i_lwSystemFault_2`. Any of the detected error conditions reported by these variables cause the I/O LED to be ON.	ON	OFF	ON
STOPPED	The controller has a valid application that is stopped. See details of the STOPPED state *(see page 55)* for an explanation of the behavior of outputs and field buses in this state.	Regular flash	OFF	OFF
STOPPED with external error detected	Configuration, TM3, SD card, or other I/O error detected.	Regular flash	OFF	ON

Controller State	Description	LED		
		RUN (Green)	ERR (Red)	I/O (Red)
HALT	The controller stops executing the application because it has detected an application error	Regular flash	ON	–
Boot Application not saved	The controller has an application in memory that differs from the application in Flash memory. At next power cycle, the application will be changed by the one from Flash memory.	ON or regular flash	Single flash	OFF

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



Details of the STOPPED State

The following statements are true for the STOPPED state:

- 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. 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.

CAN Behavior When Update IO While In Stop Is Selected

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

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

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

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

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

CAN Behavior When Update IO While In Stop Is Not Selected

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

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

Section 7.3
State Transitions and System Events

Overview

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

What Is in This Section?

This section contains the following topics:

Controller States and Output Behavior

Introduction

The Modicon M251 Logic Controller defines output behavior in response to commands and system events in a way that allows for greater flexibility. An understanding of this behavior is necessary before discussing the commands and events that affect controller states. For example, typical controllers define only two options for output behavior in stop: fallback to default value or keep current value.

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

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

Managed by Application Program

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

NOTE: An exception to this is if the RUNNING with External Error Detected state has been provoked by a I/O expansion bus error. For more information, refer to I/O Configuration General Description *(see page 82)*.

Keep Current Values

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

This output behavior applies in the STOPPED controller state. It also applies to CAN bus in the HALT controller state. Outputs are set to and maintained in their current state, although the details of the output behavior vary greatly depending on the setting of the **Update I/O while in stop** option and the actions commanded via configured fieldbusses. Refer to Controller States Description *(see page 53)* for more details on these variations.

Set All Outputs to Default

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

This output behavior applies:

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

It also applies to CAN bus in the HALT controller state. Outputs are set to and maintained in their current state, although the details of the output behavior vary greatly depending on the setting of the **Update I/O while in stop** option and the actions commanded via configured fieldbusses.Refer to Controller States Description *(see page 53)* for more details on these variations.

Hardware Initialization Values

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

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

- For an analog output: Z (high impedance)
- For a fast transistor output: Z (high impedance)
- For a regular transistor output: 0 Vdc
- For a relay output: Open

Software Initialization Values

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

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

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

Output Forcing

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

You are only able to force the value of an output while your controller is connected to EcoStruxure Machine Expert.

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

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

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

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

Output Forcing Considerations

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

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

⚠ 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 switch goes from stop to run.
- EcoStruxure Machine Expert Online Menu: Select the **Start** command.
- RUN command from Web Server
- By an external call via Modbus request using the PLC_W.q_wPLCControl and PLC_W.q_uiOpenPLCControl system variables of the M251 PLCSystem library *(see Modicon M251 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, HALT, or EMPTY state.
- The controller is restarted into the RUNNING state automatically under certain conditions.

Refer to Controller State Diagram *(see page 49)* 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 switch goes from run to stop.
- EcoStruxure Machine Expert Online Menu: Select the **Stop** command.
- STOP command from WebServer
- By an internal call by the application or an external call via Modbus request using the PLC_W.q_wPLCControl and PLC_W.q_uiOpenPLCControl system variables of the M251 PLCSystem library *(see Modicon M251 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, HALT, or EMPTY state.

- REBOOT by Script: The file transfer script on an SD card can issue a REBOOT as its final command. The controller is rebooted into the STOPPED state provided the other conditions of the boot sequence allow this to occur. Refer to Reboot *(see page 64)* for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram *(see page 49)* for further details.

Reset Warm

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

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Warm Command:

- EcoStruxure Machine Expert Online Menu: Select the **Reset warm** command.
- By an internal call by the application or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M251 PLCSystem library *(see Modicon M251 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. All non-located and non-remanent variables are reset to their initialization values.
7. The values of the first 1000 `%MW` registers are maintained.
8. The values of `%MW1000` to `%MW59999` registers are reset to 0.
9. All fieldbus communications are stopped and then restarted after the reset is complete.
10. All inputs are reset to their initialization values. All outputs are reset to their software initialization values or their default values if no software initialization values are defined.
11. The Post Configuration file is read *(see page 215)*.

For details on variables, refer to Remanent Variables *(see page 68)*.

Reset Cold

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

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Cold Command:

- EcoStruxure Machine Expert Online Menu: Select the **Reset cold** command.
- By an internal call by the application or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M251 PLCSystem library *(see Modicon M251 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. All non-located and non-remanent variables are reset to their initialization values.
7. The values of the first 1000 %MW registers are maintained.
8. The values of %MW1000 to %MW59999 registers are reset to 0.
9. All fieldbus communications are stopped and then restarted after the reset is complete.
10. All inputs are reset to their initialization values. All outputs are reset to their software initialization values or their default values if no software initialization values are defined.
11. The Post Configuration file is read *(see page 215)*.

For details on variables, refer to Remanent Variables *(see page 68)*.

Reset Origin

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

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Origin Command:

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

Effects of the Reset Origin Command:

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

For details on variables, refer to Remanent Variables *(see page 68)*.

Reboot

Effect: Commands a reboot of the controller.

Starting Conditions: Any state.

Methods for Issuing the Reboot Command:

- Power cycle
- REBOOT by Script *(see page 229)*

Effects of the Reboot:

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

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

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

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

2. Forcing is maintained if the boot application is loaded successfully. If not, forcing is erased.
3. Diagnostic indications for errors are reset.
4. The values of the retain variables are restored if saved context is valid.
5. The values of the retain-persistent variables are restored if saved context is valid.
6. All non-located and non-remanent variables are reset to their initialization values.
7. The values of the first 1000 `%MW` registers are restored if saved context is valid.
8. The values of `%MW1000` to `%MW59999` registers are reset to 0.
9. All fieldbus communications are stopped and restarted after the boot application is loaded successfully.
10. All inputs are reset to their initialization values. All outputs are reset to their hardware initialization values and then to their software initialization values or their default values if no software initialization values are defined.
11. The Post Configuration file is read *(see page 215)*.
12. The controller file system is initialized and its resources (sockets, file handles, and so on) are deallocated.
    The file system employed by the controller needs to be periodically re-established by a power cycle of the controller. If you do not perform regular maintenance of your machine, or if you are using an Uninterruptible Power Supply (UPS), you must force a power cycle (removal and reapplication of power) to the controller at least once a year.

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

For details on variables, refer to Remanent Variables *(see page 68)*.

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

Download Application

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

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

Methods for Issuing the Download Application Command:

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

  For important information on the application download commands, refer to Controller State Diagram.
- FTP: Load Boot application file to the Flash memory using FTP. The updated file is applied at the next reboot.
- SD card: Load Boot application file using an SD card in the controller SD card slot. The updated file is applied at the next reboot. Refer to File Transfer with SD Card *(see page 235)* for further details.

Effects of the EcoStruxure Machine Expert Download Command:

1. The existing application stops and then is erased.
2. If valid, the new application is loaded and the controller assumes a STOPPED state.
3. Forcing is erased.
4. Diagnostic indications for errors are reset.
5. The values of the retain variables are reset to their initialization values.
6. The values of any existing retain-persistent variables are maintained.
7. All non-located and non-remanent variables are reset to their initialization values.
8. The values of the first 1000 `%MW` registers are maintained.
9. The values of `%MW1000` to `%MW59999` registers are reset to 0.
10. All fieldbus communications are stopped and then any configured fieldbus of the new application is started after the download is complete.
11. All inputs are reset to their initialization values. All outputs are reset to their hardware initialization values and then to their software initialization values or their default values if no software initialization values are defined, after the download is complete.
12. The Post Configuration file is read *(see page 215)*.

For details on variables, refer to Remanent Variables *(see page 68)*.

Effects of the FTP or SD Card Download Command:

There are no effects until the next reboot. At the next reboot, the effects are the same as a reboot with an invalid context. Refer to Reboot *(see page 64)*.

Error Detection, Types, and Management

Error Management

The controller detects and manages three types of errors:

- External errors
- Application errors
- System errors

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

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

(1) Expansion modules may appear to be absent for any number of reasons, even if the absent I/O module is physically present on the bus. For more information, refer to I/O Configuration General Description *(see page 82)*.

NOTE: Refer to the M251 PLCSystem library Guide for more detailed information on diagnostics.

Remanent Variables

Overview

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

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

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

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

X The value is maintained.
– The value is reinitialized.
(1) Retain variable values are maintained if an online change modifies only the code part of the boot application (for example, `a:=a+1; => a:=a+2;`). In all other cases, retain variables are reinitialized.
(2) For more details on VAR RETAIN, refer to Effects of the Reset warm Command *(see page 62)*.
(3) If the application is downloaded using an SD card, any existing retain-persistent variables used by the application are reinitialized. If the application is downloaded using EcoStruxure Machine Expert, however, existing retain-persistent variables maintain their values. In both cases, if the downloaded application contains the same retain-persistent variables as the existing application, the existing retain variables maintain their values.

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

For example, if you have in your program:

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

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

Adding Retain-Persistent Variables

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

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

Retain and Persistent Variables: Performance Impact

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

Chapter 8
Controller Device Editor

Introduction

This chapter describes how to configure the controller.

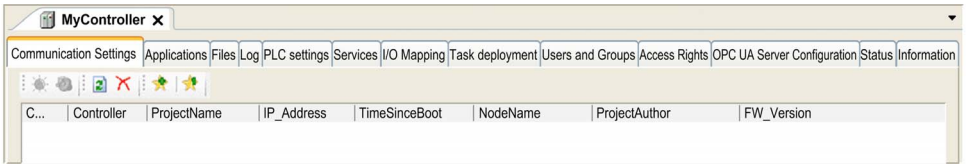
What Is in This Chapter?

This chapter contains the following topics:

Controller Parameters

Controller Parameters

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



Tabs Description

Tab	Description	Restriction
Communication Settings *(see page 74)*	Manages the connection between the PC and the controller: ● helping you find a controller in a network, ● presenting the list of available controllers, so you can connect to the selected controller and manage the application in the controller, ● helping you physically identify the controller from the device editor, ● helping you change the communication settings of the controller. The controller list is detected through NetManage or through the Active Path based on the communication settings. To access the **Communication settings**, click **Project → Project Settings...** in the menu bar. For more information, refer to the EcoStruxure Machine Expert Programming Guide (*Communication Settings*).	Online mode only
Applications	Presents the application running on the controller and allows removing the application from the controller.	Online mode only
Files *(see page 30)*	File management between the PC and the controller. Only one logic controller disk at a time can be seen through this tab. When an SD card is inserted, this file displays the content of the SD card. Otherwise, this tab displays the content of the */usr* directory of the internal flash memory of the controller.	Online mode only
Log	View the controller log file.	Online mode only
PLC settings *(see page 75)*	Configuration of: ● application name ● I/O behavior in stop ● bus cycle options	–
Services *(see page 77)*	Lets you configure the online services of the controller (RTC, device identification).	Online mode only

Tab	Description	Restriction
I/O Mapping	Mapping of the input and output channels of an I/O device on project (application) variables.	–
Task deployment	Displays a list of I/Os and their assignments to tasks.	After compilation only
Users and Groups	The **Users and Groups** tab is provided for devices supporting online user management. It allows setting up users and access-rights groups and assigning them access rights to control the access on EcoStruxure Machine Expert projects and devices in online mode. For more details, refer to the EcoStruxure Machine Expert Programming Guide.	–
OPC UA Server Configuration	Displays the OPC UA Server Configuration *(see page 207)* window.	–
Access Rights	The **Access Rights** tab is provided for devices supporting online user management. It serves to grant or deny the currently defined user groups certain permissions, thus defining the access rights for users on files or objects (for example, an application) on the controller during runtime. For more details, refer to the EcoStruxure Machine Expert Programming Guide.	–
Status	No information delivered.	–
Information	Displays general information about the device (name, description, provider, version, image).	–

Communication Settings

Introduction

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

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

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

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

Edit Communication Settings

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

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

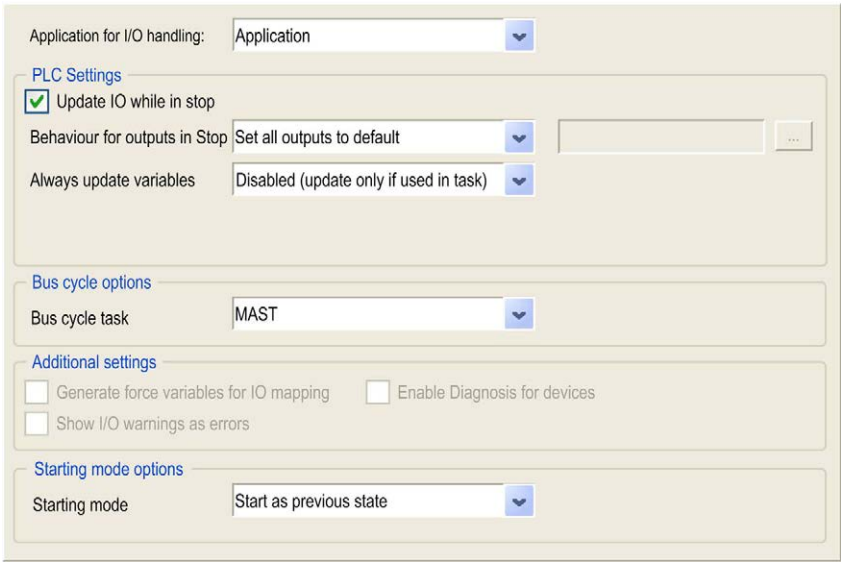
- Without the **Save settings permanently** option:
  Configure the communication parameters and click **OK**. These settings are immediately taken into account and are not kept if the controller is reset. For the next resets, the communication parameters configured into the application are taken into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

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

PLC Settings

Overview

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



Element		Description
Application for I/O handling		By default, set to Application because there is only one application in the controller.
PLC settings	**Update IO while in stop**	If this option is activated (default), the values of the input and output channels get also updated when the controller is stopped.
	Behavior for outputs in Stop	From the selection list, choose one of the following options to configure how the values at the output channels should be handled in case of controller stop: • **Keep current values** • **Set all outputs to default**
	Always update variables	By default, set to **Enabled 1 (use bus cycle task if not used in task)** and not editable.

Element		Description
Bus cycle options	**Bus cycle task**	This configuration setting is the parent for all **Bus cycle task** parameters used in the application device tree. Some devices with cyclic calls, such as a **CANopen manager**, can be attached to a specific task. In the device, when this setting is set to **Use parent bus cycle setting**, the setting set for the controller is used. The selection list offers all tasks currently defined in the active application. The default setting is the MAST task. **NOTE: <unspecified>** means that the task is in "slowest cyclic task" mode.
Additional settings	**Generate force variables for IO mapping**	Not used.
	Enable Diagnosis for devices	Not used.
	Show I/O warnings as errors	Not used.
Starting mode Options	**Starting mode**	This option defines the starting mode on a power-on. For further information, refer to State behavior diagram *(see page 49)*. Select with this option one of these starting modes: • **Start as previous state** • **Start in stop** • **Start in run**

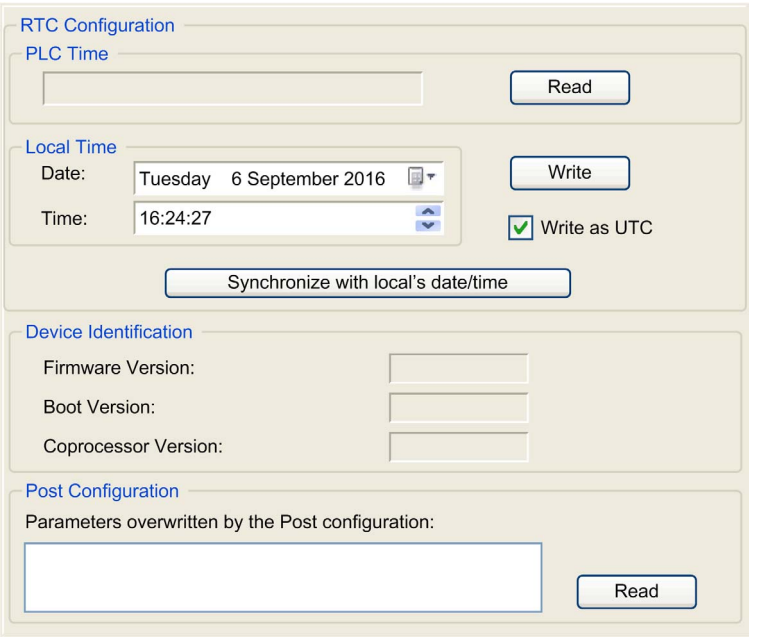
Services

Services Tab

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

- RTC Configuration
- Device Identification
- Post Configuration

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



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

Element		Description
RTC Configuration	**PLC Time**	Displays the date and time read from the controller when the **Read** button is clicked, with no conversion applied. This read-only field is initially empty. If **Write as UTC** is selected, **PLC Time** is in UTC.
	Read	Reads the date and time saved on the controller and displays the values in the **PLC Time** field.
	Local Time	Lets you define a date and a time that are sent to the controller when the **Write** button is clicked. If necessary, modify the default values before clicking the **Write** button. A message box informs you about the result of the command. The date and time fields are initially filled with the current PC settings.
	Write	Writes the date and time defined in the **Local time** field to the logic controller. A message box informs you of the result of the command. Select the **Write as UTC** checkbox before running this command to write the values in UTC format.
	Synchronize with local date/time	Lets you directly send the PC settings. A message box informs you of the result of the command. Select **Write as UTC** before running this command to use UTC format. Use UTC time when using secure communication.
Device Identification		Displays the **Firmware Version**, the **Boot Version**, and the **Coprocessor Version** of the selected controller, if connected.
Post Configuration		Displays the application parameters overwritten by the Post configuration *(see page 215)*.

Users Rights

Introduction

Users and Groups and **Access Right** tabs *(see page 72)* allows to manage user accounts, user access rights groups and the associated access rights, to control the access on projects. For more informations, refer to the EcoStruxure Machine Expert Programming Guide.

Login and passwords

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

This table describes how to log in:

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

⚠ WARNING

UNAUTHORIZED DATA AND/OR APPLICATION ACCESS

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

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

NOTE: Anonymous login can be restored by removing the user rights in the **User Management** page *(see page 118)* of the web server.

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

Default users and groups

One user and two groups are set by default:

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

Access Rights

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

You can allow the following operations through the access rights:

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

Troubleshooting

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

Chapter 9
Expansion Modules Configuration

Overview

This chapter describes how to configure the TM4, TM3, and TM2 expansion modules for the Modicon M251 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

TM3 I/O Configuration General Description

Introduction

In your project, you can add I/O expansion modules to your M251 Logic Controller to increase the number of digital and analog inputs and outputs to the controller.

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 M251 Logic Controller, Hardware Guide)*).

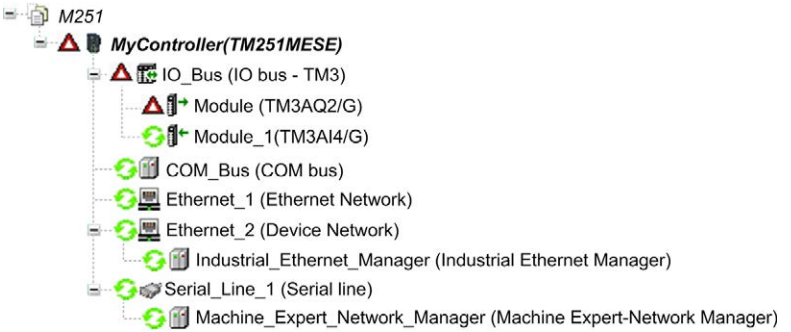
The I/O expansion bus of the M251 Logic Controller is created when you assemble the I/O expansion modules to the logic controller. I/O expansion modules are considered as external devices in the logic controller architecture and, as such, are treated differently than the embedded I/Os of the logic controller.

I/O Expansion Bus Errors

If the logic controller cannot communicate with one or more I/O expansion modules contained in the program configuration, and those modules are not configured as optional modules (refer to Optional I/O Expansion Modules *(see page 90)*), the logic controller considers it as an I/O expansion bus error. The unsuccessful communication may be detected during the startup of the logic controller or during runtime, and there may be any number of causes. Causes of communication exceptions on the I/O expansion bus include, among other things, disconnection of or physically missing I/O modules, electromagnetic radiation beyond published environmental specifications, or otherwise inoperative modules.

If an I/O expansion bus error is detected:

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



The following diagnostic information is also available:

- Bit 0 and bit 1 of the `PLC_R.i_lwSystemFault_1` system variable are set to 0.
- The `PLC_R.i_wIOStatus1` and `PLC_R.i_wIOStatus2` system variables are set to `PLC_R_IO_BUS_ERROR`.
- The `TM3_MODULE_R[i].i_wModuleState` system variable, where `[i]` identifies the TM3 expansion module in error, is set to `TM3_BUS_ERROR`.
- The `TM3_GetModuleBusStatus` function block returns the `TM3_ERR_BUS` error code.

Refer to PLC_R *(see Modicon M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* and TM3_MODULE_R *(see Modicon M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* structures for details on system variables.

Active I/O Expansion Bus Error Handling

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

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

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

- Power cycle
- New application download
- Restarting the I/O Bus by setting the `TM3_BUS_W.q_wIOBusRestart` system variable to 1. The bus is restarted if at least one expansion module is in error (`TM3_MODULE_R[i].i_wModuleState` = `TM3_BUS_ERROR`). Refer to Restarting the I/O Expansion Bus *(see page 85)*.
- Issuing a **Reset Warm** or **Reset Cold** command with EcoStruxure Machine Expert *(see page 61)*.

Passive I/O Expansion Bus Handling

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

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

- For TM3 I/O expansion modules, the value of the I/O channels is maintained (**Keep current values**) for approximately 10 seconds while the logic controller attempts to re-establish communication. If the logic controller cannot re-establish communications within that time, all affected TM3 I/O expansion outputs are set to 0.
- For TM2 I/O expansion modules that may be part of the configuration, the value of the I/O channels is maintained indefinitely. That is to say, the outputs of the TM2 I/O expansion modules are set to “Keep current values” until either power is cycled on the logic controller system, or you issue a **Reset Warm** or **Reset Cold** command with EcoStruxure Machine Expert *(see page 61)*.

In either case, the logic controller continues to solve logic and, if your controller is so equipped, the embedded I/O continues to be managed by the application (“managed by application program *(see page 58)*”) while it attempts to re-establish communication with the incommunicative I/O expansion modules. If the communication is successful, the I/O expansion modules resume to be managed by the application. If communication with the I/O expansion modules is unsuccessful, you must resolve the reason for the unsuccessful communication, and then cycle power on the logic controller system, or issue a **Reset Warm** or **Reset Cold** command with EcoStruxure Machine Expert *(see page 61)*.

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

For more information on the actions taken upon startup of the logic controller when an I/O expansion bus error is detected, refer to Controller States Description *(see page 53)*.

Restarting the I/O Expansion Bus

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

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

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

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

Match Software and Hardware Configuration

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Presentation of the Optional Feature for I/O Expansion Modules

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

NOTE: For more details about this feature, refer to Optional I/O Expansion Modules *(see page 90)*.

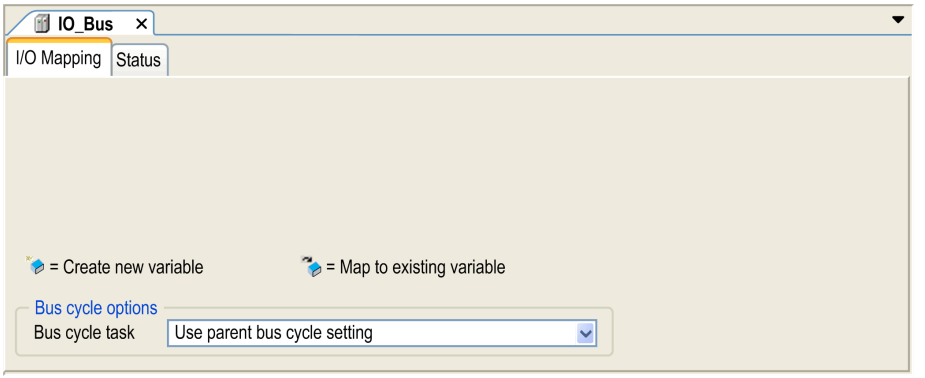
TM3 I/O Bus Configuration

Overview

TM3 I/O bus configuration enables you to select the task that drives TM3 physical exchanges. It can also override the configuration defined in the **PLC settings** *(see page 75)* bus cycle task.

Configuring the I/O Bus

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

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

TM4 Expansion Module Configuration

Introduction

The Modicon M251 Logic Controller supports the TM4 communication expansion modules.

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Adding an Expansion Module

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

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

- Using the Drag-and-drop Method
- Using the Contextual Menu or Plus Button

TM3/TM2 Expansion Module Configuration

Introduction

The Modicon M251 Logic Controller supports the following expansion modules:

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Adding an Expansion Module

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

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

- Using the Drag-and-Drop Method
- Using the Contextual Menu or Plus Button

Optional I/O Expansion Modules

Presentation

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

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

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

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

The following module types can be marked as optional:

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

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

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

⚠ WARNING

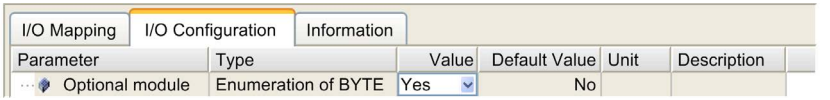
UNINTENDED EQUIPMENT OPERATION

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

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

Marking an I/O Expansion Module as Optional

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

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

Shared Internal ID Codes

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

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

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

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

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

Chapter 10
Ethernet Configuration

Introduction

This chapter describes how to configure the Ethernet network interface of the Modicon M251 Logic Controller.

What Is in This Chapter?

This chapter contains the following sections:

Section 10.1
Ethernet Services

What Is in This Section?

This section contains the following topics:

Presentation

Ethernet Services

The controller supports the following services:

- Modbus TCP Server *(see page 103)*
- Modbus TCP Client *(see page 103)*
- Web Server *(see page 105)*
- FTP Server *(see page 121)*
- SNMP *(see page 123)*
- Controller as Target Device On EtherNet/IP *(see page 124)*
- Controller as Slave Device On Modbus TCP *(see page 150)*
- IEC VAR ACCESS *(see page 96)*

TM251MESE Specific Considerations

The TM251MESE has two different Ethernet networks. Each one gets its own and unique IP and MAC address.

The two Ethernet networks are called Ethernet 1 and Ethernet 2:

- Ethernet 1 is a dual port Ethernet switch dedicated to communication between machines or with the control network.
- Ethernet 2 is a separate Ethernet port dedicated to device network connections.

For example, you can:

- Connect your PC to the Ethernet 1
- Use a Modbus TCP I/O scanner with the Ethernet 2.

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

- Ethernet 1 port.
- Ethernet 2 port only if the Ethernet 1 port has a valid IP address and is connected to a device.

Ethernet Protocols

The controller supports the following protocols:

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

Connections

This table shows the maximum number of connections:

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

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

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

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

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

Services Available

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

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

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

IP Address Configuration

Introduction

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

- Address assignment by DHCP server
- Address assignment by BOOTP server
- Fixed IP address
- Post configuration file *(see page 215)*. If a post configuration file exists, this assignment method has priority over the others.

The IP address can also be changed dynamically through the:

- Communication Settings *(see EcoStruxure Machine Expert, Programming Guide)* tab in EcoStruxure Machine Expert
- **changeIPAddress** function block *(see page 245)*

NOTE: If the attempted addressing method is unsuccessful, the link uses a default IP address *(see 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

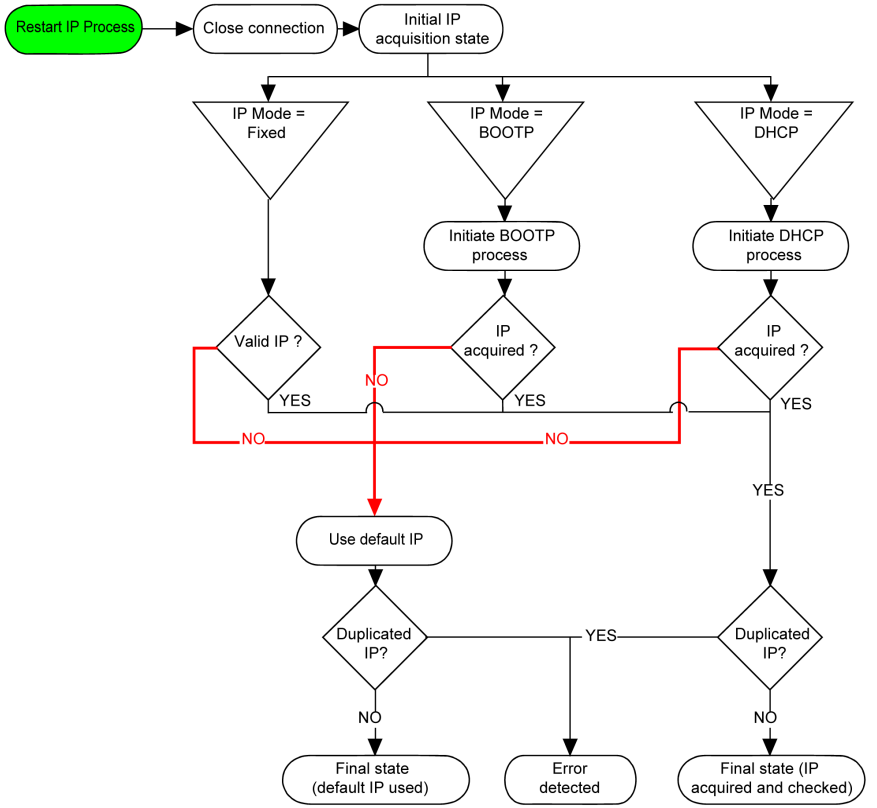
- Verify that there is only one master controller configured on the network or remote link.
- Verify that all devices have unique addresses.
- Obtain your IP address from your system administrator.
- Confirm that the IP address of the device is unique before placing the system into service.
- Do not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



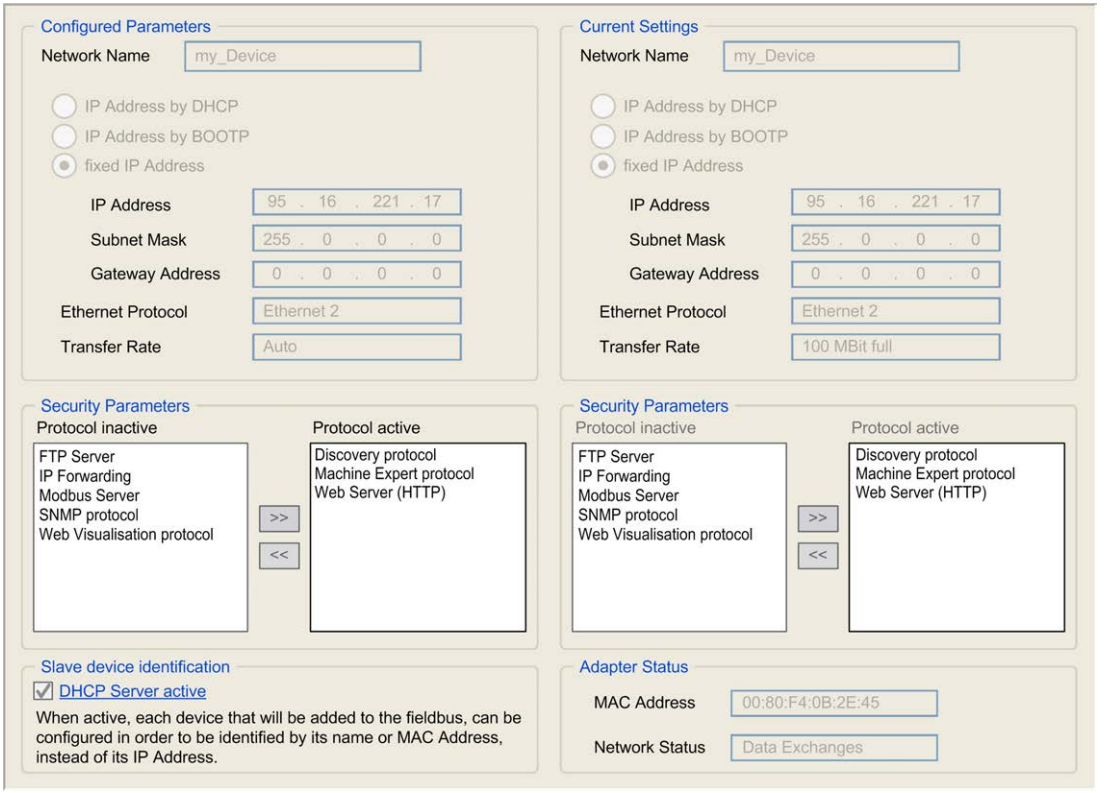
NOTE: If a device programmed to use the DHCP or BOOTP addressing methods is unable to contact its respective server, the controller uses the default IP address. It repeats its request constantly.

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

This table describes the configured parameters:

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

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

Default IP Address

The default IP addresses are:

- 10.11.x.x. for Ethernet_1
- 10.10.x.x. for Ethernet_2 (only available on TM251MESE)

NOTE: The two IP addresses must not be in the same IP network.

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

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

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

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

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

Address Classes

The IP address is linked:

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

An IP address is always coded using 4 bytes.

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

The different IP address classes are defined in this table:

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

Subnet Mask

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

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

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

Example of a subnet address:

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

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

Gateway Address

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

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

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

Security Parameters

This table describes the different security parameters:

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

Security Parameters	Description	Default settings
IP Forwarding	This parameter deactivates the IP forwarding service of the controller. When deactivated, devices on the device network are no longer accessible from the control network (Web pages, DTM, and so on). **NOTE:** This parameter is only available on the Ethernet_1 network.	Inactive
SNMP protocol	This parameter deactivates the SNMP server of the controller. When deactivated, SNMP requests are ignored.	Inactive
Web Server (HTTP)	This parameter deactivates the Web Server of the controller. When deactivated, HTTP requests to the controller Web Server are ignored.	Active
WebVisualisation protocol	This parameter deactivates the Web visualization pages of the controller. When deactivated, HTTP requests to the logic controller WebVisualisation protocol are ignored.	Inactive

Slave Device Identification

When **DHCP Server active** is selected, devices added to the fieldbus can be configured to be identified by their name or MAC address, instead of their IP address. Refer to DHCP Server *(see page 174)*.

NOTE: This parameter is only available on the Ethernet_2 network.

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 M251 Logic Controller implements both client and server services so that it can initiate communications to other controllers and I/O devices, and to respond to requests from other controllers, SCADA, HMIs, and other devices. By default, Modbus Server functionality is not active.

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

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

Modbus TCP Client

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

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

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

Modbus TCP Server

The Modbus server supports the Modbus requests:

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

NOTE: The embedded Modbus server only ensures time-consistency for a single word (2 bytes). If your application requires time-consistency for more than 1 word, add and configure *(see page 150)* a **Modbus TCP Slave Device** so that the contents of the `%IW` and `%QW` buffers are time-consistent in the associated IEC task (MAST by default).

Web Server

Introduction

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

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

- Google Chrome (version 30.0 or greater)
- Mozilla Firefox (version 1.5 or greater)

The Web server is limited to 10 TCP connections *(see page 96)*.

NOTE: The Web server can be disabled by unchecking the **Web Server active** parameter in the Ethernet Configuration tab *(see page 99)*.

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

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

Web Server Access

Access to the Web server is controlled by User Rights when they are enabled in the controller. For more information, refer to **Users and Groups** Tab Description *(see page 72)*.

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

⚠ WARNING

UNAUTHORIZED DATA ACCESS

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

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

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

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

Home Page Access

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

This figure shows the Web Server site login page:

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

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

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

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

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

Monitoring: Data Parameters

Monitoring Web Server Variables

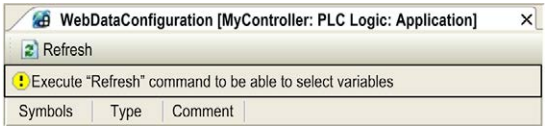
To monitor Web server variables, you must add a **Web Data Configuration** object to your project. Within this object, you can select all variables you want to monitor.

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

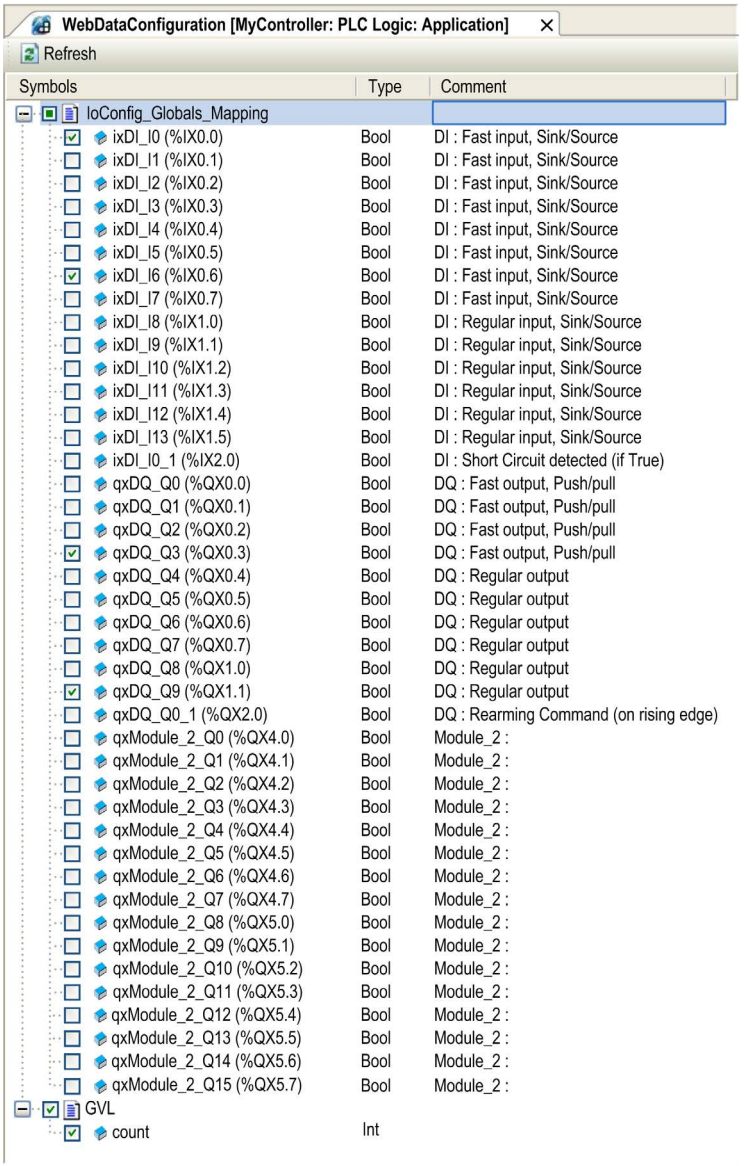
Step	Action
1	Right click the **Application** node in the **Applications tree** tab.
2	Click **Add Object → Web Data Configuration...**. **Result:** The **Add Web Data Configuration** window is displayed.
3	Click **Add**. **Result:** The **Web Data Configuration** object is created and the **Web Data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

Web Data Configuration Editor

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



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



Symbols	Type	Comment
IoConfig_Globals_Mapping		
ixDI_I0 (%IX0.0)	Bool	DI : Fast input, Sink/Source
ixDI_I1 (%IX0.1)	Bool	DI : Fast input, Sink/Source
ixDI_I2 (%IX0.2)	Bool	DI : Fast input, Sink/Source
ixDI_I3 (%IX0.3)	Bool	DI : Fast input, Sink/Source
ixDI_I4 (%IX0.4)	Bool	DI : Fast input, Sink/Source
ixDI_I5 (%IX0.5)	Bool	DI : Fast input, Sink/Source
ixDI_I6 (%IX0.6)	Bool	DI : Fast input, Sink/Source
ixDI_I7 (%IX0.7)	Bool	DI : Fast input, Sink/Source
ixDI_I8 (%IX1.0)	Bool	DI : Regular input, Sink/Source
ixDI_I9 (%IX1.1)	Bool	DI : Regular input, Sink/Source
ixDI_I10 (%IX1.2)	Bool	DI : Regular input, Sink/Source
ixDI_I11 (%IX1.3)	Bool	DI : Regular input, Sink/Source
ixDI_I12 (%IX1.4)	Bool	DI : Regular input, Sink/Source
ixDI_I13 (%IX1.5)	Bool	DI : Regular input, Sink/Source
ixDI_I0_1 (%IX2.0)	Bool	DI : Short Circuit detected (if True)
qxDQ_Q0 (%QX0.0)	Bool	DQ : Fast output, Push/pull
qxDQ_Q1 (%QX0.1)	Bool	DQ : Fast output, Push/pull
qxDQ_Q2 (%QX0.2)	Bool	DQ : Fast output, Push/pull
qxDQ_Q3 (%QX0.3)	Bool	DQ : Fast output, Push/pull
qxDQ_Q4 (%QX0.4)	Bool	DQ : Regular output
qxDQ_Q5 (%QX0.5)	Bool	DQ : Regular output
qxDQ_Q6 (%QX0.6)	Bool	DQ : Regular output
qxDQ_Q7 (%QX0.7)	Bool	DQ : Regular output
qxDQ_Q8 (%QX1.0)	Bool	DQ : Regular output
qxDQ_Q9 (%QX1.1)	Bool	DQ : Regular output
qxDQ_Q0_1 (%QX2.0)	Bool	DQ : Rearming Command (on rising edge)
qxModule_2_Q0 (%QX4.0)	Bool	Module_2 :
qxModule_2_Q1 (%QX4.1)	Bool	Module_2 :
qxModule_2_Q2 (%QX4.2)	Bool	Module_2 :
qxModule_2_Q3 (%QX4.3)	Bool	Module_2 :
qxModule_2_Q4 (%QX4.4)	Bool	Module_2 :
qxModule_2_Q5 (%QX4.5)	Bool	Module_2 :
qxModule_2_Q6 (%QX4.6)	Bool	Module_2 :
qxModule_2_Q7 (%QX4.7)	Bool	Module_2 :
qxModule_2_Q8 (%QX5.0)	Bool	Module_2 :
qxModule_2_Q9 (%QX5.1)	Bool	Module_2 :
qxModule_2_Q10 (%QX5.2)	Bool	Module_2 :
qxModule_2_Q11 (%QX5.3)	Bool	Module_2 :
qxModule_2_Q12 (%QX5.4)	Bool	Module_2 :
qxModule_2_Q13 (%QX5.5)	Bool	Module_2 :
qxModule_2_Q14 (%QX5.6)	Bool	Module_2 :
qxModule_2_Q15 (%QX5.7)	Bool	Module_2 :
GVL		
count	Int	

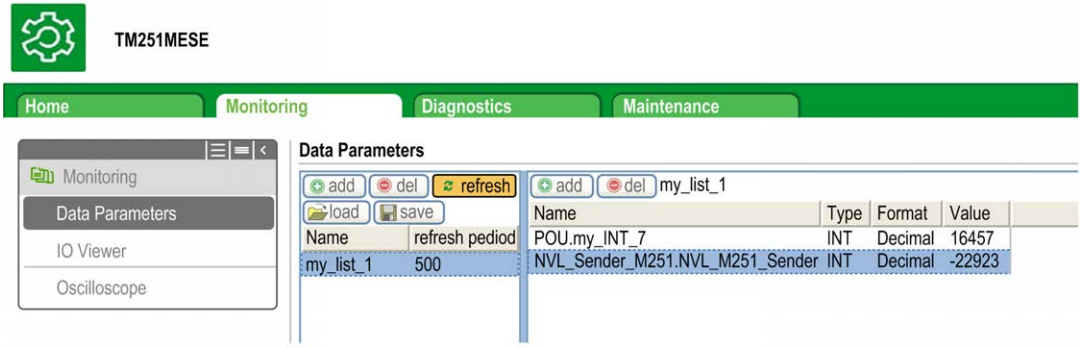
NOTE: The variable selection is possible only in offline mode.

Monitoring: Data Parameters Submenu

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

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

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



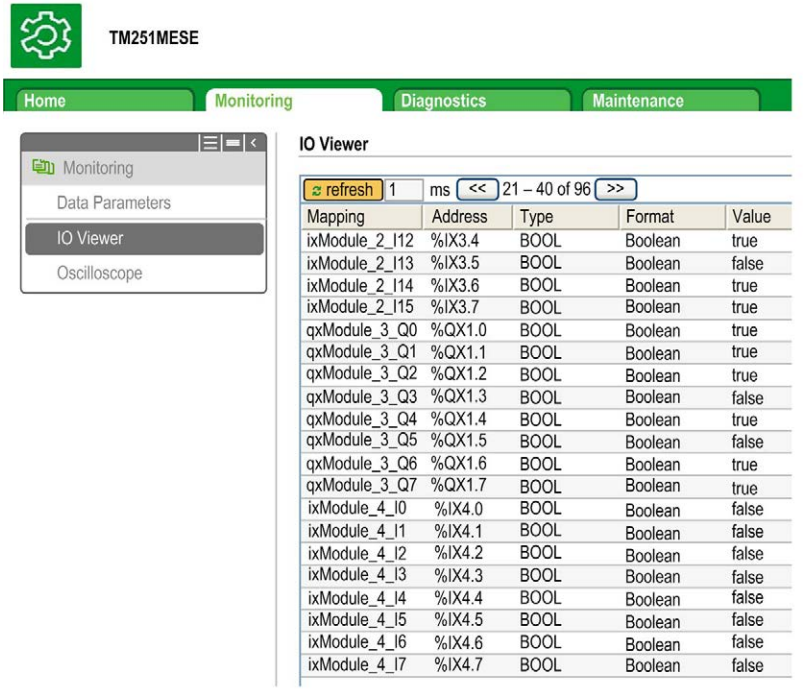
Element	Description
Add	Adds a list description or a variable
Del	Deletes a list description or a variable
Refresh period	Refreshing period of the variables contained in the list description (in ms)
Refresh	Enables I/O refreshing: ● Gray button: refreshing disabled ● Orange button: refreshing enabled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

NOTE: The IEC objects (`%IX`, `%QX`) are not directly accessible. To access IEC objects you must first group their contents in located registers (refer to Relocation Table *(see page 34)*).

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

Monitoring: IO Viewer Submenu

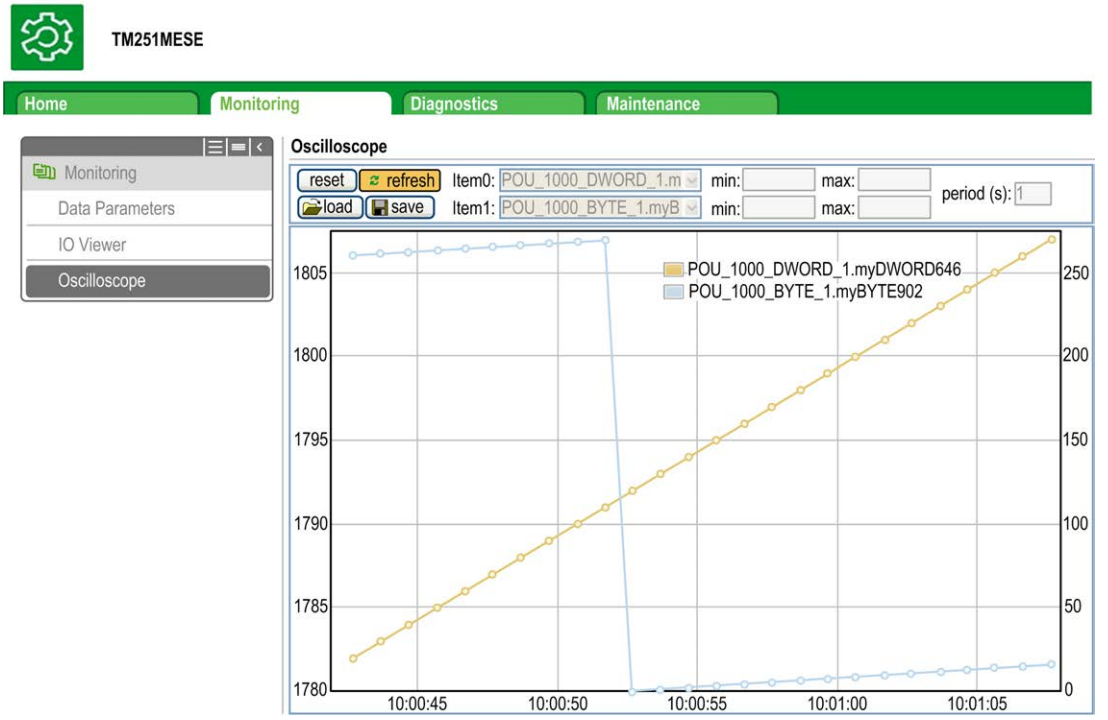
The **IO Viewer** submenu allows you to display and modify the current I/O values:



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

Monitoring: Oscilloscope Submenu

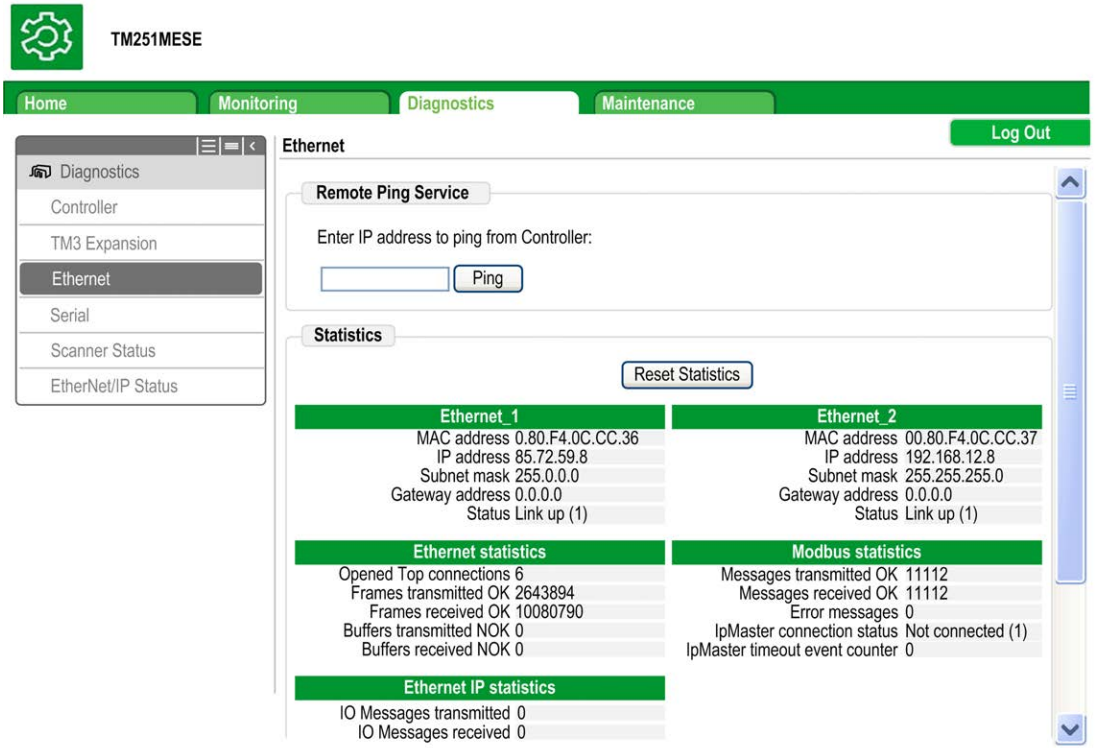
The **Oscilloscope** submenu can display up to 2 variables in the form of a recorder time chart:



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

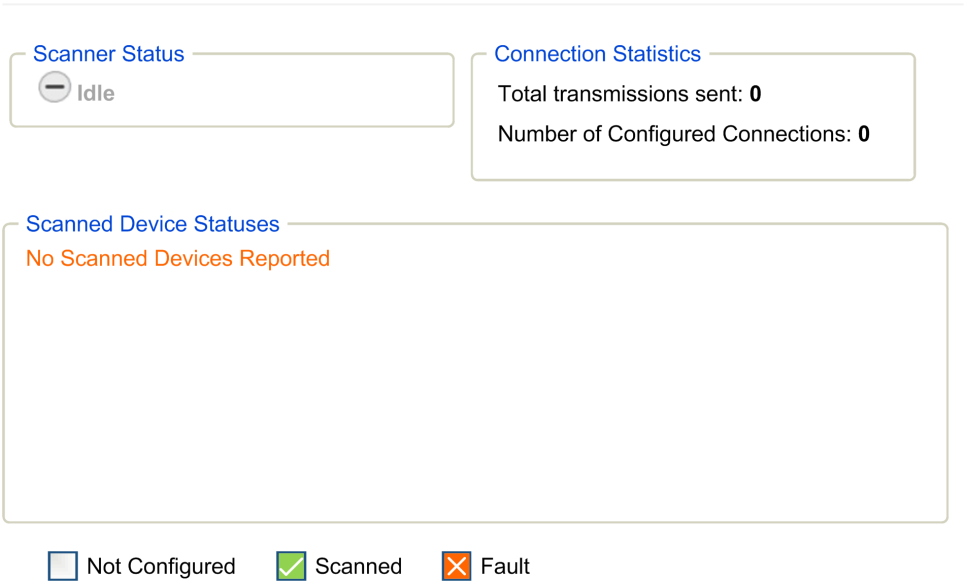
Diagnostics: Ethernet Submenu

This figure shows the remote ping service:



Diagnostics: Scanner Status Submenu

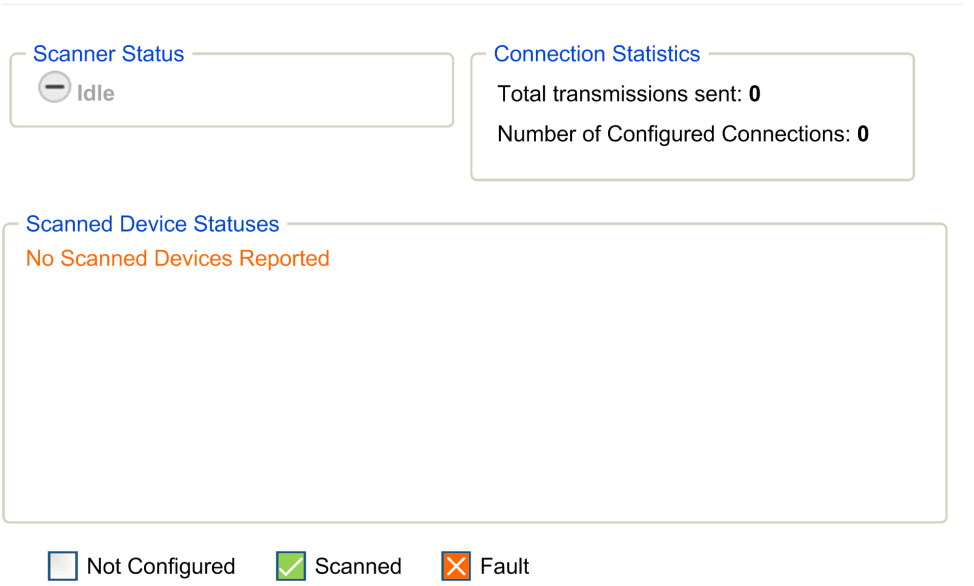
The **Scanner Status** submenu displays status of the Modbus TCP I/O Scanner (IDLE, STOPPED, OPERATIONAL) and the health bit of up to 64 Modbus slave devices:



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

Diagnostics: EtherNet/IP Status Submenu

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



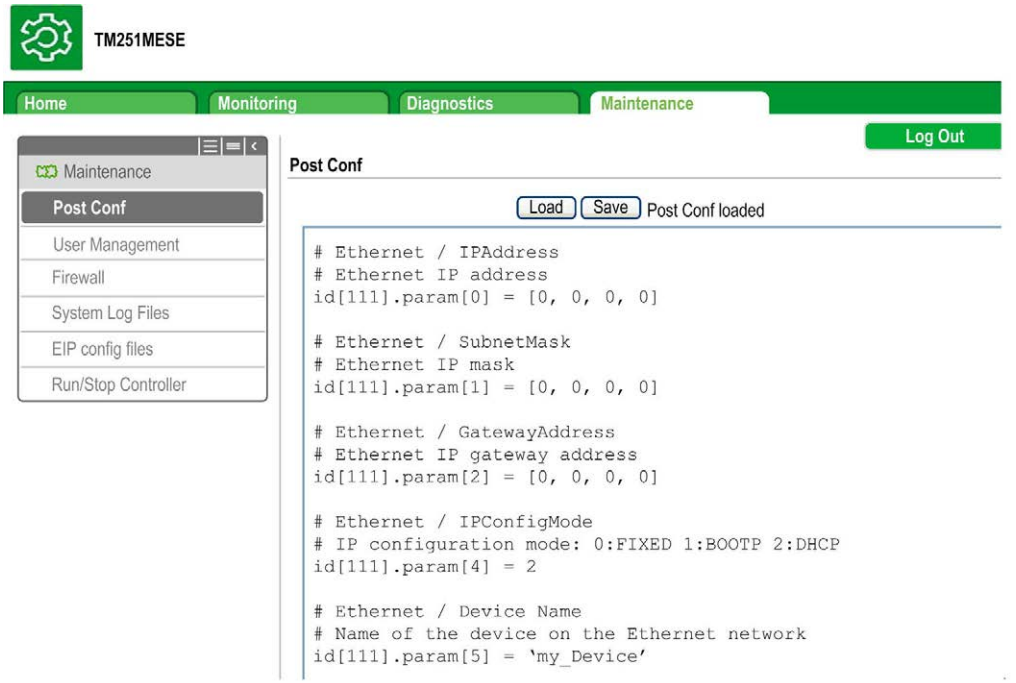
For more information, refer to EcoStruxure Machine Expert EtherNet/IP User guide.

Maintenance Page

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

Maintenance: Post Conf Submenu

The **Post Conf** submenu allows you to update the post configuration file *(see page 215)* saved on the controller:

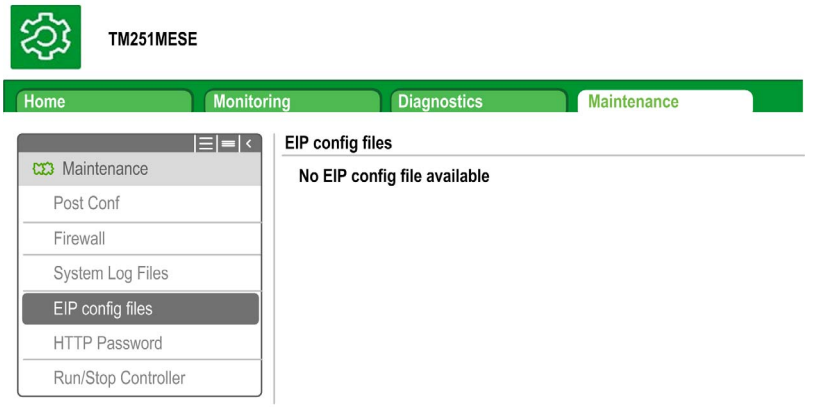


Step	Action
1	Click **Load**.
2	Modify the parameters *(see page 219)*.
3	Click **Save**. **NOTE:** The new parameters will be considered at next Post Configuration file reading *(see page 216)*.

Maintenance: EIP Config Files Submenu

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

Index of `/usr`:



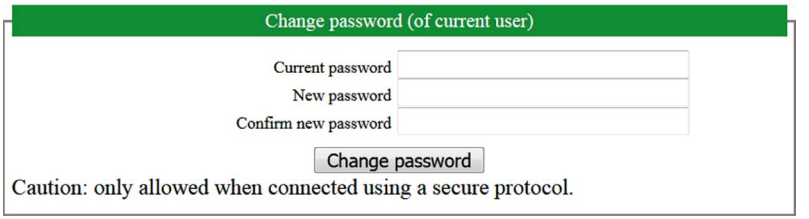
File	Description
My Machine Controller.gz	GZIP file
My Machine Controller.ico	Icon file
My Machine Controller.eds	Electronic Data Sheet file

Maintenance: User Management Submenu

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

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

allows you to change your password.



- **User accounts management**:

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

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

Click **OK** on the window that appears to confirm. As a result:
- Users no longer have to set and enter a password to connect to the controller.
- FTP, HTTP, and OPC UA Server connections accept anonymous user connections.
- Cloning the controller no longer requires authorization by using the FB_ControlClone function block.

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

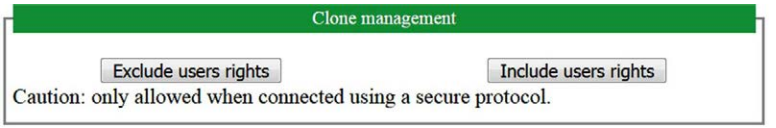
Click **Reset to default** to return all user accounts on the controller to their default setting state.

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

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

- **Clone management**:

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



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

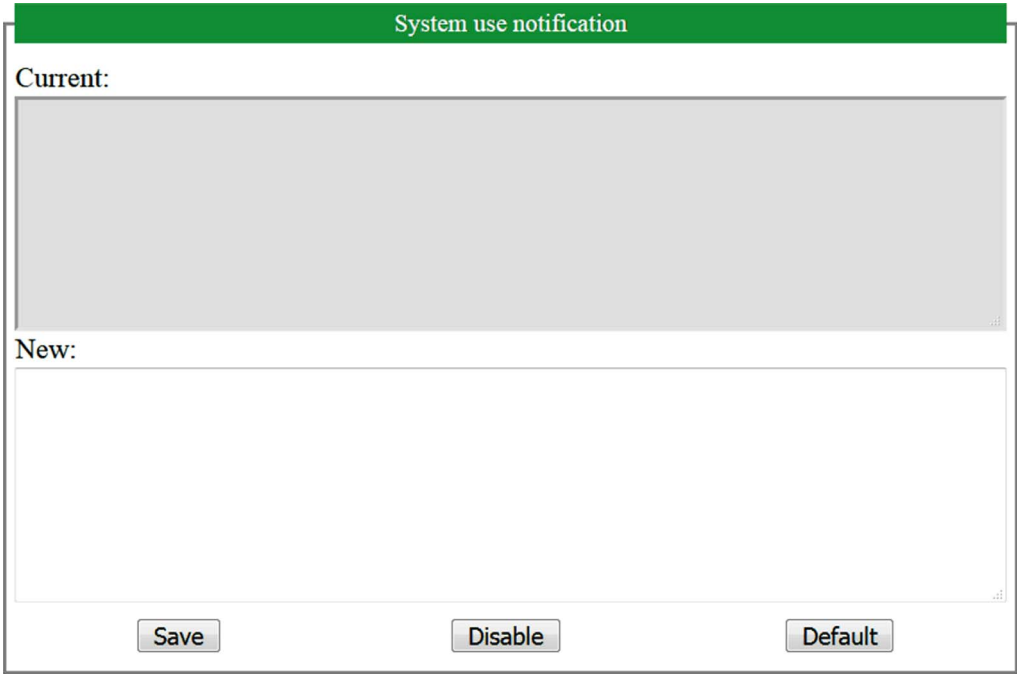
NOTE: By default, the users rights are excluded.

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

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

- **System use notification**:

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



FTP Server

Introduction

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

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

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

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

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

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

The FTP server is deactivated by default.

Files Access

See File Organization *(see page 30)*.

FTP Client

Introduction

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

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

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

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

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

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

For further information, refer to FtpRemoteFileHandling Library Guide.

SNMP

Introduction

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

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

SNMP Server

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

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

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

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

SNMP Client

The M251 Logic Controller supports an SNMP client library to allow you to query SNMP servers. For details, refer to the *SNMP Library Guide*.

Controller as a Target Device on EtherNet/IP

Introduction

This section describes the configuration of the M251 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

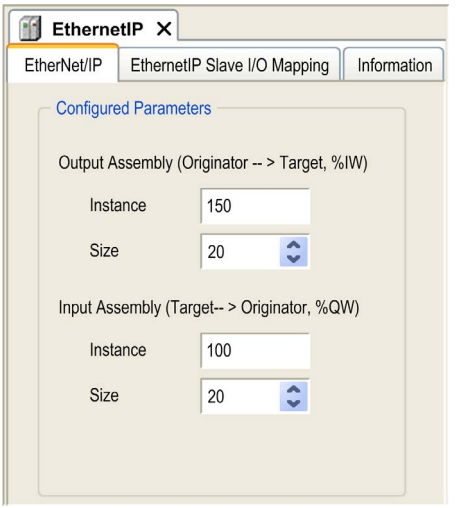
To configure your M251 Logic Controller as an EtherNet/IP target device, you must:

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

EtherNet/IP Parameters Configuration

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

This dialog box is displayed:



The EtherNet/IP configuration parameters are defined as:

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

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

EDS File Generation

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

To generate the EDS file:

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

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

A generic EDS file for the M251 Logic Controller is also available on the Schneider website. You must adapt this file to your application by editing it and defining the required Assembly instances and sizes.

EthernetIP Slave I/O Mapping Tab

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

EthernetIP | EthernetIP Slave I/O Mapping | Information

Channels

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

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

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

The number of words depends on the size parameter configured in EtherNet/IP Target Configuration *(see page 124)*.

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 M251 Logic Controller supports cyclic connections only. If an Originator opens a connection using a change of state as a trigger, packets are sent at the RPI rate.

Profile

The controller supports the following objects:

Object class	Class ID (hex)	Cat.	Number of Instances	Effect on Interface Behavior
Identity Object *(see page 128)*	01	1	1	Supports the reset service
Message Router Object *(see page 131)*	02	1	1	Explicit message connection
Assembly Object *(see page 133)*	04	2	2	Defines I/O data format
Connection Manager Object *(see page 134)*	06		1	–
TCP/IP Interface Object *(see page 137)*	F5	1	1	TCP/IP configuration
Ethernet Link Object *(see page 139)*	F6	1	1	Counter and status information
Interface Diagnostic Object *(see page 140)*	350	1	1	–
IOScanner Diagnostic Object *(see page 144)*	351	1	1	–
Connection Diagnostic Object *(see page 145)*	352	1	1	–
Explicit Connection Diagnostic Object *(see page 147)*	353	1	1	–
Explicit Connections Diagnostic List Object *(see page 148)*	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.

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	07	The largest instance attributes value

The following table describes the Class Services:

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

The following table describes the Instance Services:

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

(1) Reset Service description:

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

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

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

The following table describes the Instance attributes:

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
1	Get	Vendor ID	UINT	F3	Schneider Automation ID
2	Get	Device type	UINT	0E	Controller
3	Get	Product code	UINT	1002	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

Bit	Name	Description
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 M251 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

Attribute ID (hex)	Access	Name	Data Type	Value (hex)	Details
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.

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

Ethernet Link Object (Class ID = F6 hex)

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

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

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

The following table describes the class services:

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

Instance Codes

Only instance 1 is supported.

The following table describes the instance services:

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

The following table describes the instance attributes:

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

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

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

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

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

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

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

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

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

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

The following table describes the class services:

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

IOScanner Diagnostic Object (Class ID = 351 hex)

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

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

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

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

The following table describes the class services:

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

IO Connection Diagnostic Object (Class ID = 352 hex)

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

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

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

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

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

Instance Attributes

The following table describes the class services:

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

Explicit Connection Diagnostic Object (Class ID = 353 hex)

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

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

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

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

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

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

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

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

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

The following table describes the class services:

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

Controller as a Slave Device on Modbus TCP

Overview

This section describes the configuration of the M251 Logic Controller as a **Modbus TCP Slave Device**.

The **Modbus TCP Slave Device** adds another Modbus server function to the controller. This server is addressed by the Modbus client application by specifying a configured Unit ID (Modbus address) in the range 1...247. The embedded Modbus server of the slave controller needs no configuration, and is addressed by specifying a Unit ID equal to 255. Refer to Modbus TCP Configuration *(see page 151)*.

To configure your M251 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.

Only one **Modbus TCP Slave Device** at a time can be configured on one of the Ethernet ports of the M251 Logic Controller (**Ethernet_1** or **Ethernet_2**). Once configured, however, the Modbus TCP slave device can be addressed through both Ethernet ports.

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 M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*.

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

Adding a Modbus TCP Slave Device

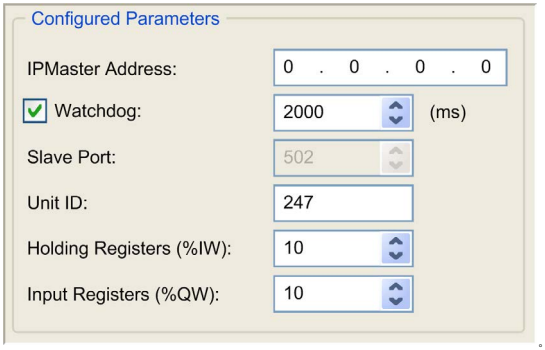
To configure your M251 Logic Controller as a Modbus TCP slave device, you must:

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. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method • Using the Contextual Menu or Plus Button

Modbus TCP Configuration

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

This dialog box appears:



Element	Description
IP Master Address	IP address of the Modbus master The connections are not closed on this address.
Watchdog	Watchdog in 500 ms increments **NOTE:** The watchdog applies to the IP master Address unless the address is 0.0.0.0.
Slave Port	Modbus communication port (502) **NOTE:** The port number can be modified using the changeModbusPort script command *(see page 163)*.
Unit ID	Sends the requests to the Modbus TCP slave device (1...247), instead of to the embedded Modbus server (255).
Holding Registers (%IW)	Number of %IW registers to be used in the exchange (2...120) (each register is 2 bytes)
Input Registers (%QW)	Number of %QW registers to be used in the exchange (2...120) (each register is 2 bytes)

Modbus TCP Slave Device I/O Mapping Tab

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

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

Once a **Modbus TCP Slave Device** has been configured, Modbus commands sent to its Unit ID (Modbus address) are handled differently than the same command would be when addressed to any other Modbus device on the network. For example, when the Modbus command 3 (3 hex) is sent to a standard Modbus device, it reads and returns the value of one or more registers. When this same command is sent to the Modbus TCP *(see page 103)* Slave, it facilitates a read operation by the external I/O scanner.

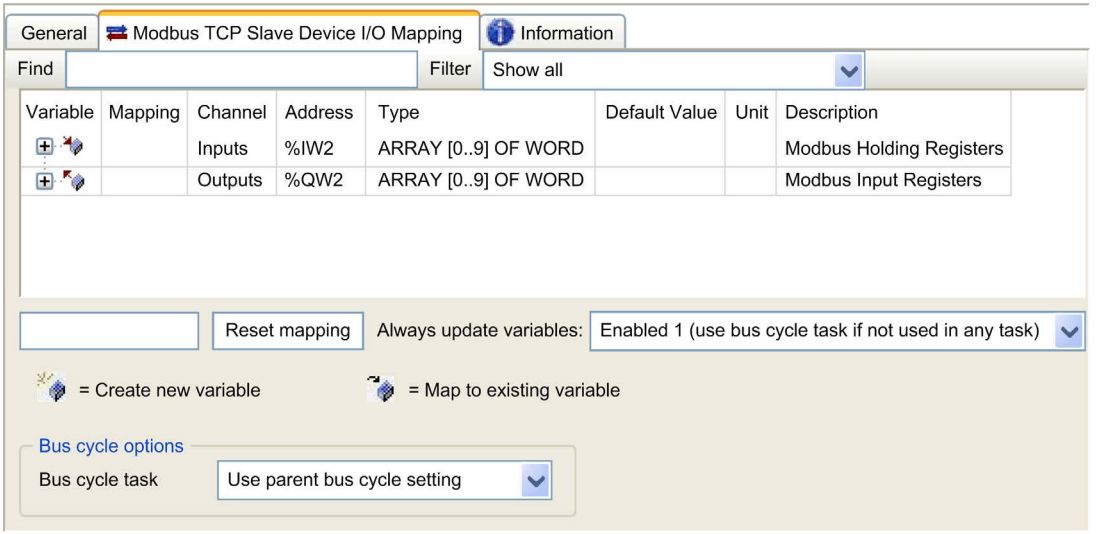
Once a **Modbus TCP Slave Device** has been configured, Modbus commands sent to its Unit ID (Modbus address) access the `%IW` and `%QW` objects of the controller instead of the regular Modbus words (accessed when the Unit ID is 255). This facilitates read/write operations by a Modbus TCP IOScanner application.

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

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

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

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



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

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

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

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

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

Bus Cycle Options

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

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

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

Changing the Modbus TCP Port

changeModbusPort Command

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

The current Modbus **Slave Port** is displayed on the Modbus TCP configuration window *(see page 151)*.

The default Modbus port number is 502.

Command	Description
`changeModbusPort` *"portnum"*	*portnum* is the new Modbus port number to use is passed as a string of characters. Before running the command, refer to Used Ports *(see page 167)* to ensure that *portnum* is not being used by any other TCP/UDP protocols or processes. An error is logged in the */usr/Syslog/FWLog.txt* file if the specified port number is already in use.

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

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

NOTE: After changing the port number, the **Modbus Server Active** checkbox on the Ethernet Configuration window *(see page 99)* is no longer taken into account, as the Modbus server always uses port 502.

Running the Command from an SD Card Script

Step	Action

1	Create a script file *(see page 228)*, for example:
	`; Change Modbus slave port`
	`changeModbusPort "1502";`

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

Running the Command Using ExecuteScript

The `changeModbusPort` command can be run from within an application using the ExecuteScript function block.

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

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

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

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

Section 10.2
Firewall Configuration

Introduction

This section describes how to configure the firewall of the Modicon M251 Logic Controller.

What Is in This Section?

This section contains the following topics:

Introduction

Firewall Presentation

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

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

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

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

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

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

Firewall Configuration

There are three ways to manage the controller firewall configuration:

- Static configuration
- Dynamic changes
- Application settings

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

Static Configuration

The static configuration is loaded at the controller boot.

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

Dynamic Changes

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

There are two ways to load these dynamic changes using:

- A physical SD card *(see page 160)*.
- A function block *(see page 160)* in the application.

Dynamic Changes Procedure

Using an SD Card

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

Step	Action
1	Create a valid script file *(see page 163)*. For example, name the script file *FirewallMaintenance.cmd*.
2	Load the script file on the SD card. For example, load the script file in the *usr/Cfg* folder.

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

4	Insert the SD card on the controller.

Using a Function Block in the Application

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

Step	Action
1	Create a valid script file *(see page 163)*. 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 M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide)* function block. For example, the **[SCmd]** input is `'Firewall_install "/usr/Syslog/FirewallMaintenance.cmd"'`

Firewall Behavior

Introduction

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

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

No Default Script File

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

Default Script File Present

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

Incorrect Default Script File Present

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

Application Settings with No Default Script File

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

Execute Dynamic Script File Already Executed

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

Firewall Script Commands

Overview

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

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

Script File Syntax

The syntax of script files is described in Script Syntax Guidelines *(see page 228)*.

General Firewall Commands

The following commands are available to manage the Ethernet firewall of the M251 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 = For TM251MESC: ● Eth1: Ethernet_1 For TM251MESE: ● Eth1: Ethernet_1 ● Eth2: Ethernet_2	

Specific Firewall Commands

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

Command	Range	Description
`Firewall Eth1 Allow IP •.•.•.•`	• = 0...255	Frames from the specified IP address are allowed on all port numbers and port types.
`Firewall Eth1 Reject IP •.•.•.•`	• = 0...255	Frames from the specified IP address are rejected on all port numbers and port types.
`Firewall Eth1 Allow IPs •.•.•.• to •.•.•.•`	• = 0...255	Frames from the IP addresses in the specified range are allowed for all port numbers and port types.
`Firewall Eth1 Reject IPs •.•.•.• to •.•.•.•`	• = 0...255	Frames from the IP addresses in the specified range are rejected for all port numbers and port types.
`Firewall Eth1 Allow port_type port Y`	Y = (destination port numbers *(see page 167)*)	Frames with the specified destination port number are allowed.
`Firewall Eth1 Reject port_type port Y`	Y = (destination port numbers *(see page 167)*)	Frames with the specified destination port number are rejected. **NOTE:** When IP forwarding is activated, rules with reject port only filter frames with current controller as destination. They are not applied for the frames routed by the current controller.
`Firewall Eth1 Allow port_type ports Y1 to Y2`	Y = (destination port numbers *(see page 167)*)	Frames with a destination port number in the specified range are allowed.
`Firewall Eth1 Reject port_type ports Y1 to Y2`	Y = (destination port numbers *(see page 167)*)	Frames with a destination port number in the specified range are rejected.
`Firewall Eth1 Allow IP •.•.•.• on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 167)*)	Frames from the specified IP address and with the specified destination port number are allowed.
`Firewall Eth1 Reject IP •.•.•.• on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 167)*)	Frames from the specified IP address and with the specified destination port number are rejected.
`Firewall Eth1 Allow IP •.•.•.• on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers *(see page 167)*)	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 *(see page 167)*)	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 *(see page 167)*)	Frames from an IP address in the specified range and with the specified destination port number are allowed.
`Firewall Eth1 Reject IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 167)*)	Frames from an IP address in the specified range and with the specified destination port number are rejected.
`Firewall Eth1 Allow IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers *(see page 167)*)	Frames from an IP address in the specified range and with a destination port number in the specified range are allowed.
`Firewall Eth1 Reject IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type ports Y1 to Y2`	• = 0...255 Y = (destination port numbers *(see page 167)*)	Frames from an IP address in the specified range and with a destination port number in the specified range are rejected.
`Firewall Eth1 Allow MAC ••:••:••:••:••:••`	• = 0...F	Frames from the specified MAC address ••:••:••:••:•• are allowed. **NOTE:** When the rules to allow the MAC address are applied, only the listed MAC addresses can communicate with the controller, even if other rules are allowed.
`Firewall Eth1 Reject MAC ••:••:••:••:••:••`	• = 0...F	Frames with the specified MAC address ••:••:••:••:•• are rejected.

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

Script Example

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

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

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

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

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

Ports Used

Protocol	Destination Port Numbers
Machine Expert	UDP 1740, 1741, 1742, 1743 TCP 1105
FTP	TCP 21, 20
HTTP	TCP 80
Modbus	TCP 502 (1)
Machine Expert Discovery	UDP 27126, 27127
SNMP	UDP 161, 162
NVL	UDP Default value: 1202
EtherNet/IP	UDP 2222 TCP 44818
TFTP	UDP 69 (used for FDR server only)
(1) The default value can be changed using the change ModbusPort command *(see page 155)*.	

Chapter 11
Industrial Ethernet Manager

Introduction

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

What Is in This Chapter?

This chapter contains the following topics:

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.

NOTE: The following information only applies to the TM251MESE controller.

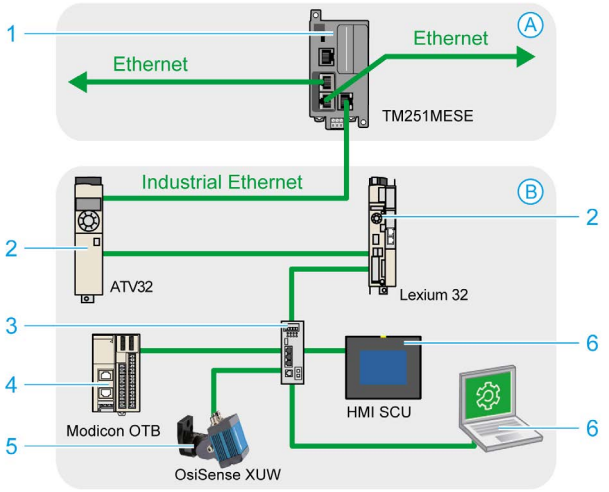
On an Industrial Ethernet network, you can connect:

- industrial devices.(industrial protocols)
- non-industrial devices (other Ethernet protocols)

For more information, refer to Industrial Ethernet User Guide *(see EcoStruxure Machine Expert 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 EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)*
2 Daisy-chained devices
3 Ethernet switch
4 I/O island (Modbus TCP)
5 Vision sensor (EtherNet/IP)
6 PC and HMI (TCP/UDP)
2, 4, and 5 Industrial Ethernet slave devices (EtherNet/IP / Modbus TCP)

This architecture is configurable with EcoStruxure Machine Expert.

Industrial Ethernet Description

TM251MESE logic controller	
Features	**Description**
Topology	Daisy chain and Star via switches
Bandwidth	10/100 Mbit/s
EtherNet/IP Scanner	
Performance	Up to 16 EtherNet/IP target devices managed by the logic controller, monitored within a timeslot of 10 ms
Number of connections	0...16
Number of input words	0...1024
Number of output words	0...1024
I/O communications	EtherNet/IP Scanner service Function block for configuration and data transfer
	Originator/Target
Modbus TCP IOScanner	
Performance	Up to 64 Modbus TCP slave devices managed by the logic controller, monitored within a timeslot of 64 ms.
Number of connections	0...64
Number of input words	0...2048
Number of output words	0...2048
I/O communications	Modbus TCP IOScanner service Function block for data transfer
	Master/Slave

TM251MESE logic controller	
Features	**Description**
Other services	FDT/DTM/EDS management
	FDR (Fast Device Replacement)
	DHCP server
	Security management (refer to Security Parameters *(see page 101)* and Firewall Configuration *(see page 157)*)
	Modbus TCP server
	Modbus TCP client
	EtherNet/IP adapter (controller as a target on EtherNet/IP)
	EtherNet/IP Originator
	Modbus TCP server (controller as a slave on Modbus TCP)
	Web server
	FTP Server (FTP and TFTP protocols)
	SNMP
	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.

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.

The 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_2 (Device Network)** node of the device tree to activate these functions and services:

- EtherNet/IP Scanner
- Modbus TCP IOScanner

The **Industrial_Ethernet_manager** is available by default under the **Ethernet_2 (Device Network)** node. It is automatically added when a slave device is added on the **Ethernet_2 (Device Network)** node.

To manually add the **Industrial_Ethernet_manager** to the **Ethernet_2 (Device Network)** :

Step	Action
1	In the **Devices Tree**, select **Ethernet_2 (Device Network)** and click the green plus button of the node or right-click **Ethernet_2 (Device Network)** and execute the **Add Device...** command from the context menu. **Result**: The **Add Device** dialog box displays.
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 *(see EcoStruxure Machine Expert EtherNet/IP, User Guide)*, EtherNet/IP Target Settings *(see EcoStruxure Machine Expert EtherNet/IP, User Guide)* and Modbus TCP Settings *(see EcoStruxure Machine Expert Modbus TCP, User Guide)*.

DHCP Server

Overview

It is possible to configure a DHCP server on the Ethernet 2 network of the TM251MESE.

The DHCP server offers addresses to the devices connected on the Ethernet 2 network. The DHCP server only delivers static addresses. A unique identified slave gets a unique address. DHCP slave devices are identified either by their MAC address or their DHCP device name. The DHCP server configuration table defines the relation between addresses and identified slave devices.

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

For more information, refer to IP Addressing Methods *(see EcoStruxure Machine Expert Modbus TCP, User Guide)*.

Fast Device Replacement

Overview

The Fast Device Replacement (FDR) helps facilitate replacing and reconfiguring a network device. This function is available on the Ethernet 2 port of the TM251MESE.

For more information, refer to Slave Device Replacement with FDR *(see EcoStruxure Machine Expert Modbus TCP, User Guide)*.

Chapter 12
Serial Line Configuration

Introduction

This chapter describes how to configure the serial line communication of the Modicon M251 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

Serial Line Configuration

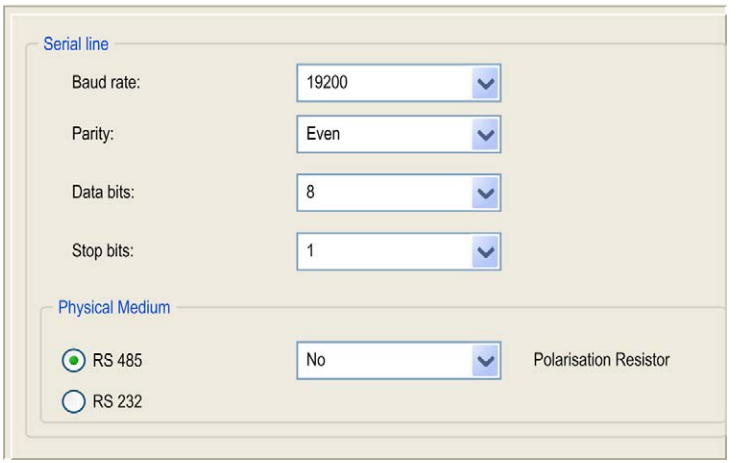
Introduction

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
Polarization Resistor	Polarization resistors are integrated in the controller. They are switched on or off by this parameter.

The serial line ports of your controller are configured for the Machine Expert protocol by default when new or when you update the controller firmware. The Machine Expert protocol is incompatible with that of other protocols such as Modbus Serial Line. Connecting a new controller to, or updating the firmware of a controller connected to, an active Modbus configured serial line can cause the other devices on the serial line to stop communicating. Make sure that the controller is not connected to an active Modbus serial line network before first downloading a valid application having the concerned port or ports properly configured for the intended protocol.

NOTICE
INTERRUPTION OF SERIAL LINE COMMUNICATIONS Be sure that your application has the serial line ports properly configured for Modbus before physically connecting the controller to an operational Modbus Serial Line network. **Failure to follow these instructions can result in equipment damage.**

This table indicates the maximum baud rate value of the managers:

Manager	Maximum Baud Rate (Bits/S)
Machine Expert Network Manager	115200
Modbus Manager	
ASCII Manager	
Modbus IOScanner	

Machine Expert Network Manager

Introduction

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

Adding the Manager

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

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

- Using the Drag-and-drop Method
- Using the Contextual Menu or Plus Button

Configuring the Manager

There is no configuration for Machine Expert Network Manager.

Adding a Modem

To add a modem to the Machine Expert Network Manager, refer to Adding a Modem to a Manager *(see page 196)*.

Modbus Manager

Introduction

The Modbus Manager is used for Modbus RTU or ASCII protocol in master or slave mode.

Adding the Manager

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

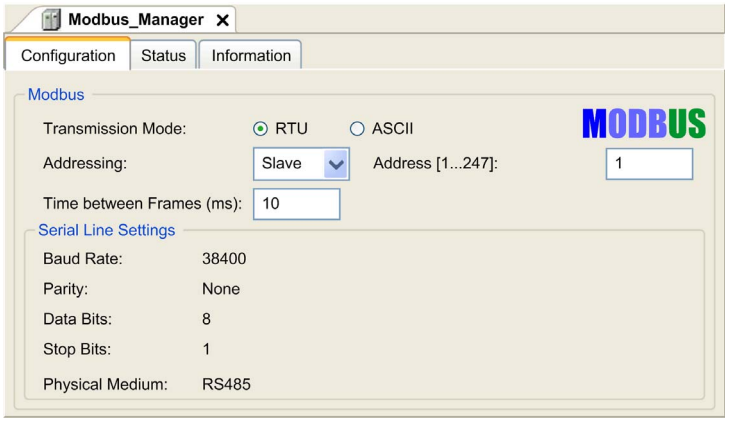
For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method
- Using the Contextual Menu or Plus Button

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

Element	Description
Address	Modbus address of the device, when slave is selected.
Time between Frames (ms)	Time to avoid bus-collision. Set this parameter identical for each Modbus device on the link.
Serial Line Settings	Parameters specified in the Serial Line configuration window.

Modbus Master

When the controller is configured as a Modbus Master, the following function blocks are supported from the PLCCommunication Library:

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

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

Modbus Slave

When the controller is configured as Modbus Slave, the following Modbus requests are supported:

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

This table contains the sub-function codes supported by the diagnostic Modbus request 08:

Sub-Function Code		Function
Dec	Hex	
10	0A	Clears Counters and Diagnostic Register
11	0B	Returns Bus Message Count
12	0C	Returns Bus Communication Error Count
13	0D	Returns Bus Exception Error Count
14	0E	Returns Slave Message Count
15	0F	Returns Slave No Response Count
16	10	Returns Slave NAK Count
17	11	Returns Slave Busy Count
18	12	Returns Bus Character Overrun Count

This table lists the objects that can be read with a read device identification request (basic identification level):

Object ID	Object Name	Type	Value
00 hex	Vendor code	ASCII String	Schneider Electric
01 hex	Product code	ASCII String	Controller reference for example: TM251MESE
02 hex	Major / Minor revision	ASCII String	aa.bb.cc.dd (same as device descriptor)

The following section describes the differences between the Modbus memory mapping of the controller and HMI Modbus mapping. If you do not program your application to recognize these differences in mapping, your controller and HMI will not communicate correctly. Thus it will be possible for incorrect values to be written to memory areas responsible for output operations.

⚠ WARNING

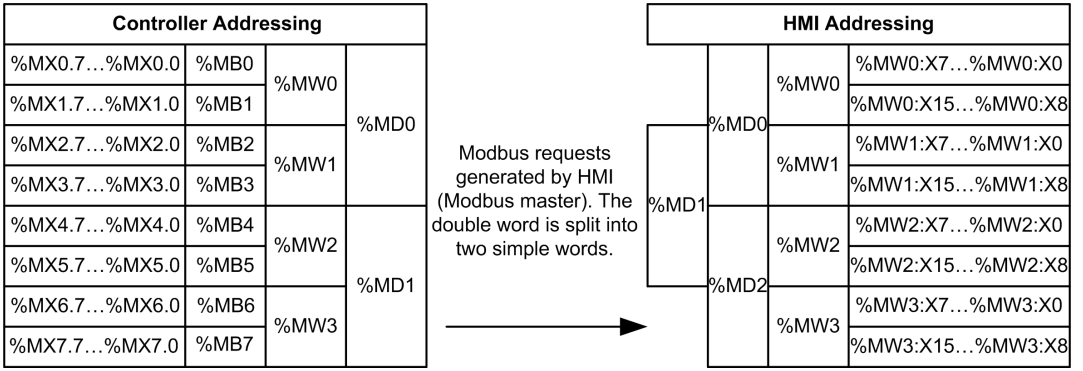
UNINTENDED EQUIPMENT OPERATION

Program your application to translate between the Modbus memory mapping used by the controller and that used by any attached HMI devices.

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

When the controller and the Magelis HMI are connected via Modbus (HMI is master of Modbus requests), the data exchange uses simple word requests.

There is an overlap on simple words of the HMI memory while using double words but not for the controller memory (see following diagram). In order to have a match between the HMI memory area and the controller memory area, the ratio between double words of HMI memory and the double words of controller memory has to be 2.



The following gives examples of memory match for the double words:

- %MD2 memory area of the HMI corresponds to %MD1 memory area of the controller because the same simple words are used by the Modbus request.
- %MD20 memory area of the HMI corresponds to %MD10 memory area of the controller because the same simple words are used by the Modbus request.

The following gives examples of memory match for the bits:

- %MW0:X9 memory area of the HMI corresponds to %MX1.1 memory area of the controller because the simple words are split in 2 distinct bytes in the controller memory.

Adding a Modem

To add a Modem to the Modbus Manager, refer to Adding a Modem to a Manager *(see page 196)*.

ASCII Manager

Introduction

The ASCII manager is used on a Serial Line, to transmit and/or receive data with a simple device.

Adding the Manager

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

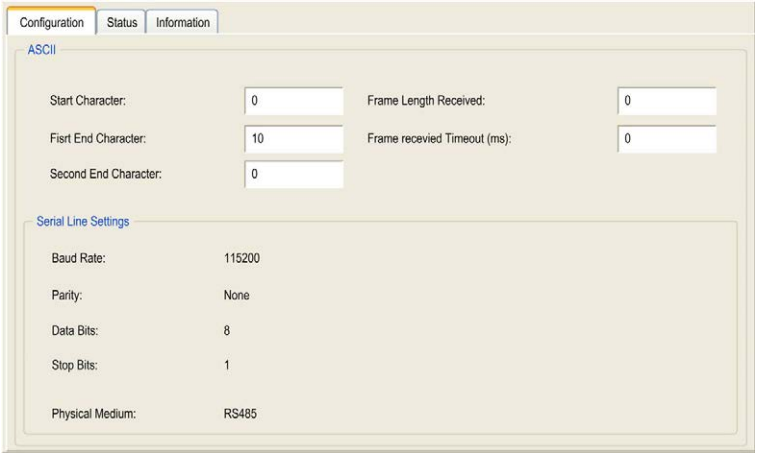
For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method
- Using the Contextual Menu or Plus Button

ASCII Manager Configuration

To configure the ASCII manager of your controller, double-click **ASCII Manager** in the **Devices tree**.

The ASCII Manager configuration window is displayed as below:



Set the parameters as described in this table:

Parameter	Description
Start Character	If 0, no start character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the beginning of a frame. In **Sending Mode**, this character is added at the beginning of the frame.
First End Character	If 0, no first end character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Second End Character	If 0, no second end character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Frame Length Received	If 0, this parameter is not used. This parameter allows the system to conclude an end of frame at reception when the controller received the specified number of characters. **Note:** This parameter cannot be used simultaneously with **Frame Received Timeout (ms)**.
Frame Received Timeout (ms)	If 0, this parameter is not used. This parameter allows the system to conclude the end of frame at reception after a silence of the specified number of ms.
Serial Line Settings	Parameters specified in the Serial Line configuration window *(see page 178)*.

NOTE: In the case of using several frame termination conditions, the first condition to be TRUE terminates the exchange.

Adding a Modem

To add a Modem to the ASCII manager, refer to Adding a Modem to a Manager *(see page 196)*.

Modbus Serial IOScanner

Introduction

The Modbus IOScanner is used to simplify exchanges with Modbus slave devices.

Add a Modbus IOScanner

To add a Modbus IOScanner on a Serial Line, select the **Modbus_IOScanner** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

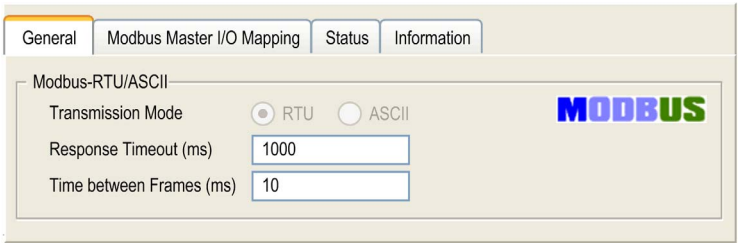
For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method
- Using the Contextual Menu or Plus Button

Modbus IOScanner Configuration

To configure a Modbus IOScanner on a Serial Line, double-click **Modbus IOScanner** in the **Devices tree**.

The configuration window is displayed as below:



Set the parameters as described in this table:

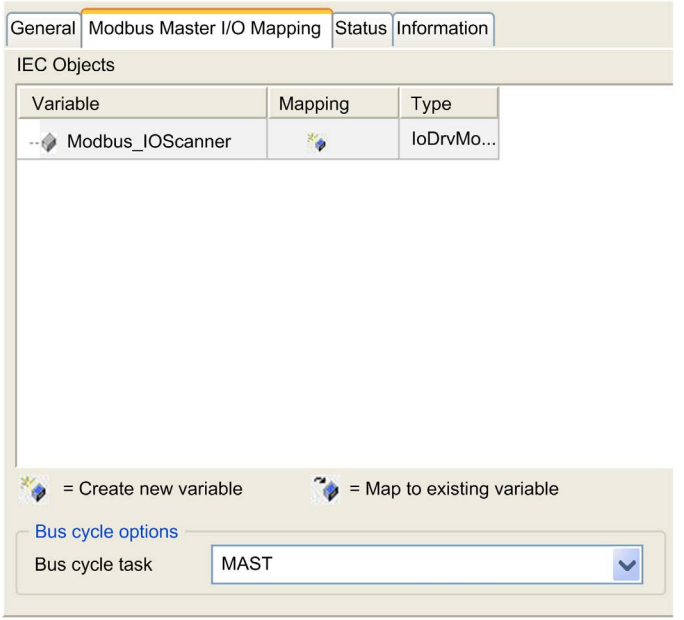
Element	Description
Transmission Mode	Specifies the transmission mode to use: • RTU: uses binary coding and CRC error-checking (8 data bits) • ASCII: messages are in ASCII format, LRC error-checking (7 data bits) Set this parameter identical for each Modbus device on the network.
Response Timeout (ms)	Timeout used in the exchanges.
Time between Frames (ms)	Delay to reduce data collision on the bus. Set this parameter identical for each Modbus device on the network.

NOTE: Do not use function blocks of the PLCCommunication library on a serial line with a Modbus IOScanner configured. This disrupts the Modbus IOScanner exchange.

Bus Cycle Task Selection

The Modbus IOScanner and the devices exchange data at each cycle of the chosen application task.

To select this task, select the **Modbus Master IO Mapping** tab. The configuration window is displayed as below:



The **Bus cycle task** parameter allows you to select the application task that manages the scanner:

- **Use parent bus cycle setting**: associate the scanner with the application task that manages the controller.
- **MAST**: associate the scanner with the MAST task.
- Another existing task: you can select an existing task and associate it to the scanner. For more information about the application tasks, refer to the EcoStruxure Machine Expert Programming Guide *(see EcoStruxure Machine Expert, Programming Guide)*.

The scan time of the task associated with the scanner must be less than 500 ms.

Adding a Device on the Modbus Serial IOScanner

Introduction

This section describes how to add a device on the Modbus IOScanner.

Adding a Device on the Modbus IOScanner

To add a device on the Modbus IOScanner, select the **Generic Modbus Slave** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the **Modbus_IOScanner** node of the **Devices tree**.

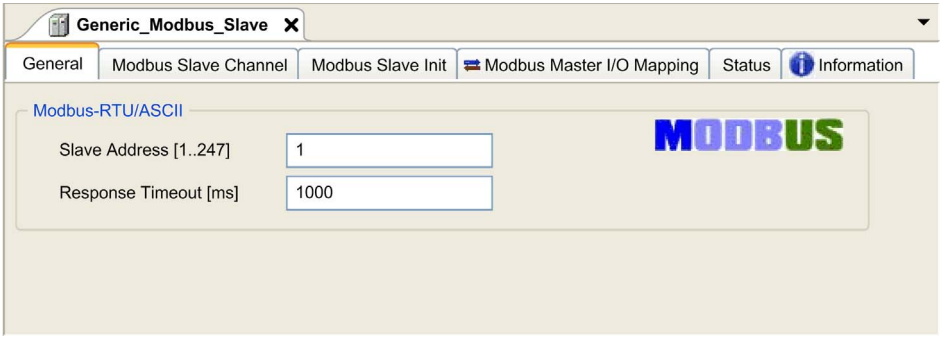
For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method
- Using the Contextual Menu or Plus Button

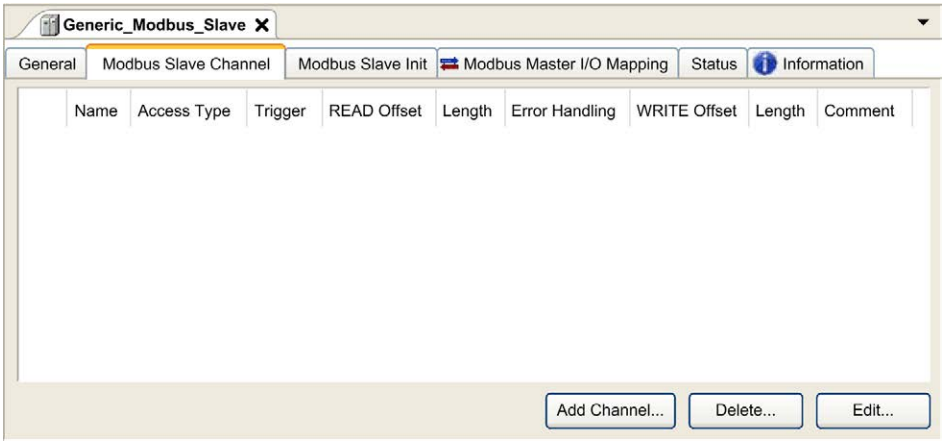
NOTE: The variable for the exchange is automatically created in the `%IWx` and `%QWx` of the **Modbus Serial Master I/O Mapping** tab.

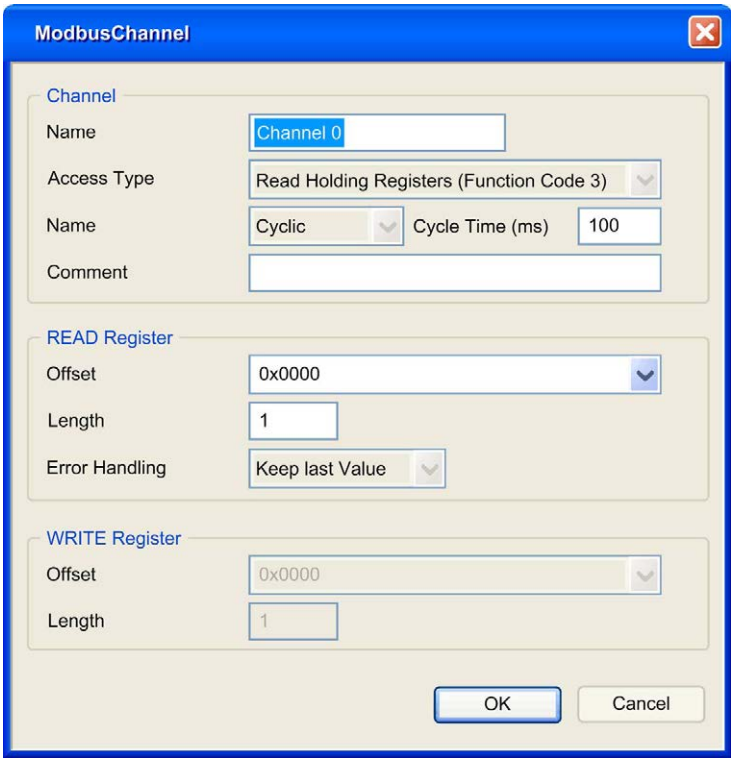
Configuring a Device Added on the Modbus IOScanner

To configure the device added on the Modbus IOScanner, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **Generic Modbus Slave**. **Result:** The configuration window is displayed. 
2	Enter a **Slave Address** value for your device (choose a value from 1 to 247).
3	Choose a value for the **Response Timeout** (in ms).

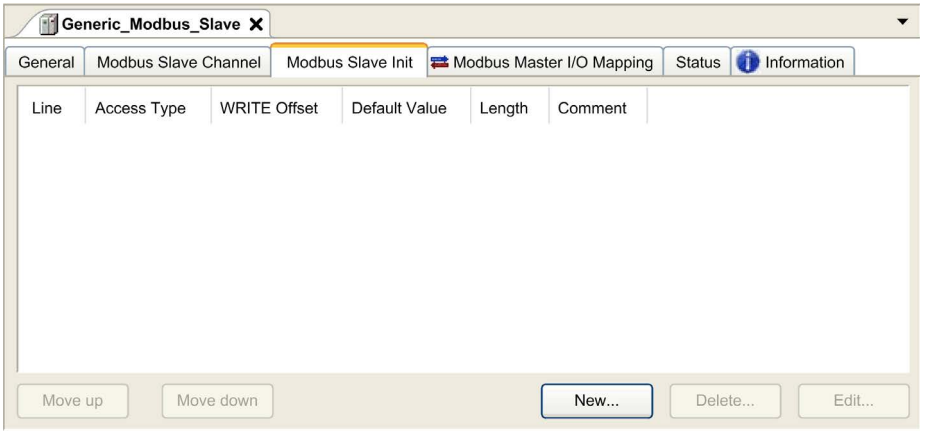
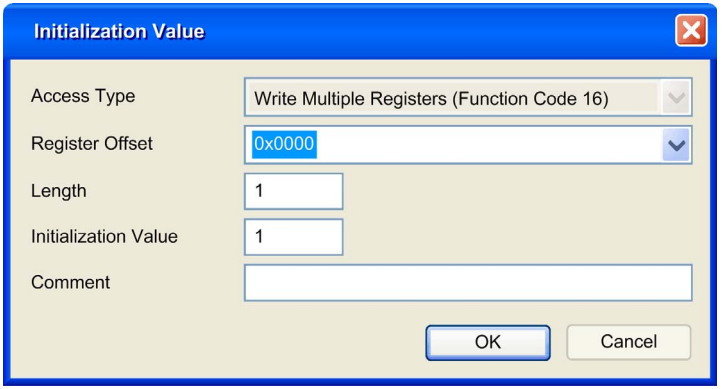
To configure the **Modbus Channels**, proceed as follows:

Step	Action
1	Click the **Modbus Slave Channel** tab: 

Step	Action
2	Click the **Add Channel** button: 

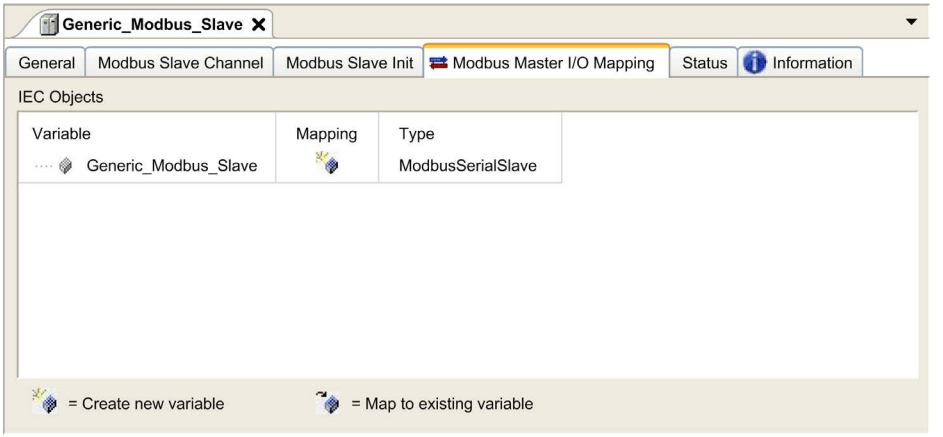
Step	Action
3	Configure an exchange: In the field **Channel**, you can add the following values: ● **Channel**: Enter a name for your channel. ● **Access Type**: Choose the exchange type: Read or Write or Read/Write multiple requests *(see page 195)*. ● **Trigger**: Choose the trigger of the exchange. It can be **CYCLIC** with the period defined in **Cycle Time (ms)** field, started by a **RISING EDGE** on a boolean variable (this boolean variable is then created in the **Modbus Master I/O Mapping** tab), or by the **Application**. ● **Comment**: Add a comment about this channel. In the field **READ Register** (if your channel is Read or Read/Write one), you can configure the %MW to be read on the Modbus slave. Those are mapped on %IW (see Modbus Master I/O Mapping tab): ● **Offset**: Offset of the %MW to read. 0 means that the first object that is read is %MW0. ● **Length**: Number of %MW to be read. For example, if 'Offset' = 2 and 'Length' = 3, the channel reads %MW2, %MW3 and %MW4. ● **Error Handling**: choose the behavior of the related %IW in case of loss of communication. In the field **WRITE Register** (if your channel is Write or Read/Write one), you can configure the %MW to be written to the Modbus slave. Those are mapped on %QW (see Modbus Master I/O Mapping tab): ● **Offset**: Offset of the %MW to write. 0 means that the first object that is written is %MW0. ● **Length**: Number of %MW to be written. For example, if 'Offset' = 2 and 'Length' = 3, the channel writes %MW2, %MW3 and %MW4.
4	Click **OK** to validate the configuration of this channel. **NOTE:** You can also: ● Click the **Delete** button to remove a channel. ● Click the **Edit** button to change the parameters of a channel.

To configure your **Modbus Initialization Value**, proceed as follows:

Step	Action
1	Click the **Modbus Slave Init** tab: 
2	Click **New** to create a new initialization value:  The **Initialization Value** window contains the following parameters: • **Access Type**: Enter the exchange type: Write requests *(see page 195)*. • **Register Offset**: Register number of register to be initialized. • **Length**: Number of %MW to be read. For example, if 'Offset' = 2 and 'Length' = 3, the channel reads %MW2, %MW3 and %MW4. • **Initialization Value**: Value the registers are initialized with. • **Comment**: Add a comment about this channel.

Step	Action
3	Click **OK** to create a new **Initialization Value**. **NOTE:** You can also: ● Click **Move up** or **Move down** to change the position of a value in the list. ● Click **Delete** to remove a value in the list. ● Click **Edit** to change the parameters of a value.

To configure your **Modbus Master I/O Mapping**, proceed as follows:

Step	Action
1	Click the **Modbus Master I/O Mapping** tab: 
2	Double-click in a cell of the **Variable** column to open a text field. Enter the name of a variable or click the browse button **[...]** and chose a variable with the **Input Assistant**.
3	For more information on I/O mapping, refer to EcoStruxure Machine Expert Programming Guide.

Access Types

This table describes the different access types available:

Function	Function Code	Availability
`Read Coils`	1	**ModbusChannel**
`Read Discrete Inputs`	2	**ModbusChannel**
`Read Holding Registers` (default setting for the channel configuration)	3	**ModbusChannel**
`Read Input Registers`	4	**ModbusChannel**
`Write Single Coil`	5	**ModbusChannel** **Initialization Value**
`Write Single Register`	6	**ModbusChannel** **Initialization Value**
`Write Multiple Coils`	15	**ModbusChannel** **Initialization Value**
`Write Multiple Registers` (default setting for the slave initialization)	16	**ModbusChannel** **Initialization Value**
`Read/Write Multiple Registers`	23	**ModbusChannel**

Adding a Modem to a Manager

Introduction

A modem can be added to the following managers:

- ASCII Manager
- Modbus Manager
- Machine Expert Network Manager

NOTE: Use a modem which implements Hayes commands if you need a modem connection with Machine Expert Network Manager.

Adding a Modem to a Manager

To add a modem to your controller, select the modem you want in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the manager node.

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

• Using the Drag-and-drop Method

• Using the Contextual Menu or Plus Button

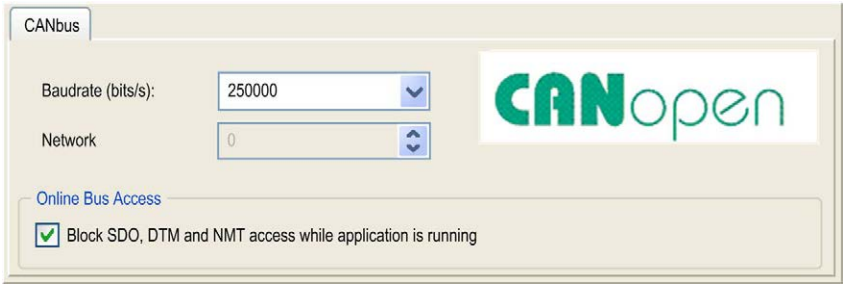
For further information, refer to Modem Library Guide *(see EcoStruxure Machine Expert, Modem Functions, Modem Library Guide)*.

Chapter 13
CANopen Configuration

CANopen Interface Configuration

CAN Bus Configuration

To configure the **CAN** bus of your controller, proceed as follows:

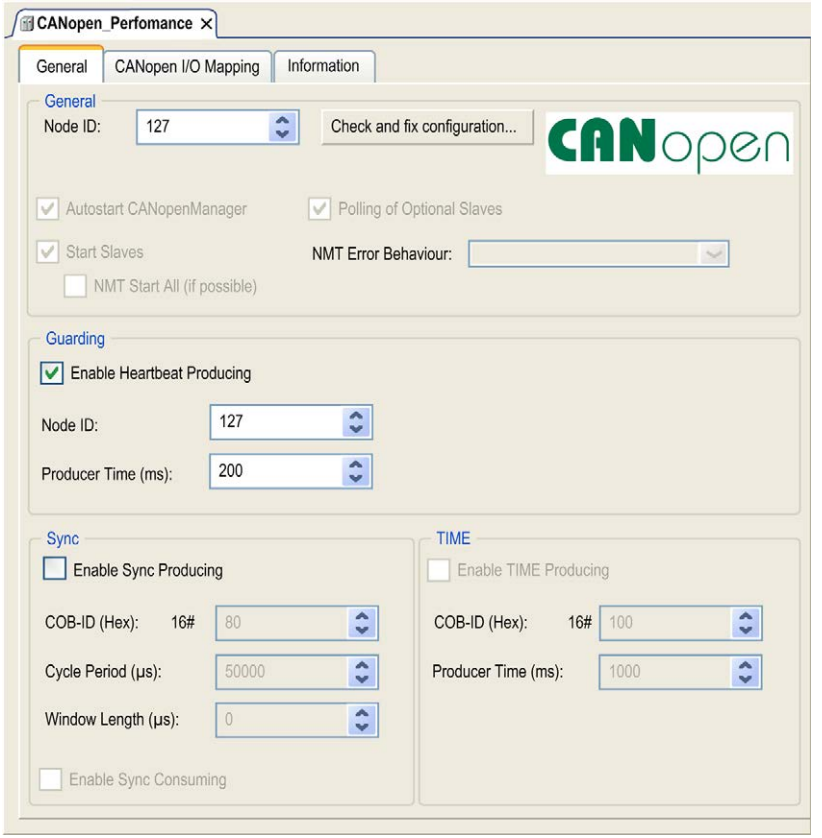
Step	Action
1	In the **Devices tree**, double-click **CAN_1**.
2	Configure the baudrate (by default: 250000 bits/s):  **NOTE:** The **Online Bus Access** option allows you to block SDO, DTM, and NMT sending through the status screen.

When connecting a DTM to a device using the network, the DTM communicates in parallel with the running application. The overall performance of the system is impacted and may overload the network, and therefore have consequences for the coherency of data across devices under control.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Place your machine or process in a state such that DTM communications will not impact its performance. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

CANopen Manager Creation and Configuration

If the **CANopen Manager** is not already present below the **CAN** node, proceed as follows to create and configure it:

Step	Action
1	Click the **Plus Button**  next to the **CAN_1** node in the **Devices Tree**. In the **Add Device** window, select **CANopen Performance** and click the **Add Device** button. For more information on adding a device to your project, refer to: ● Using the Drag-and-Drop Method *(see EcoStruxure Machine Expert, Programming Guide)* ● Using the Contextual Menu or Plus button *(see EcoStruxure Machine Expert, Programming Guide)*
2	Double-click **CANopen_Performance**. **Result:** The **CANopen Manager** configuration window appears: 

NOTE: If **Enable Sync Producing** is checked, the **CAN_x_Sync** task is added to the **Application → Task Configuration** node in the **Applications tree** tab.
Do not delete or change the **Type** or **External event** attributes of **CAN_x_Sync** tasks. If you do so, EcoStruxure Machine Expert will detect an error when you attempt to build the application, and you will not be able to download it to the controller.
If you uncheck the **Enable Sync Producing** option on the **CANopen Manager** subtab of the **CANopen_Performance** tab, the **CAN0_Sync** task is automatically deleted from your program.

Adding a CANopen Device

Refer to the EcoStruxure Machine Expert Programming Guide for more information on Adding Communication Managers and Adding Slave Devices to a Communication Manager.

CANopen Operating Limits

The Modicon M251 Logic Controller CANopen master has the following operating limits:

Maximum number of slave devices	63
Maximum number of Received PDO (RPDO)	252
Maximum number of Transmitted PDO (TPDO)	252

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not connect more than 63 CANopen slave devices to the controller.
- Program your application to use 252 or fewer Transmit PDO (TPDO).
- Program your application to use 252 or fewer Receive PDO (RPDO).

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

CAN Bus Format

The CAN bus format is CAN2.0A for CANopen.

Chapter 14
J1939 Configuration

J1939 Interface Configuration

CAN Bus Configuration

To configure the **CAN** bus of your controller, refer to CAN Bus Configuration *(see page 197)*.

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

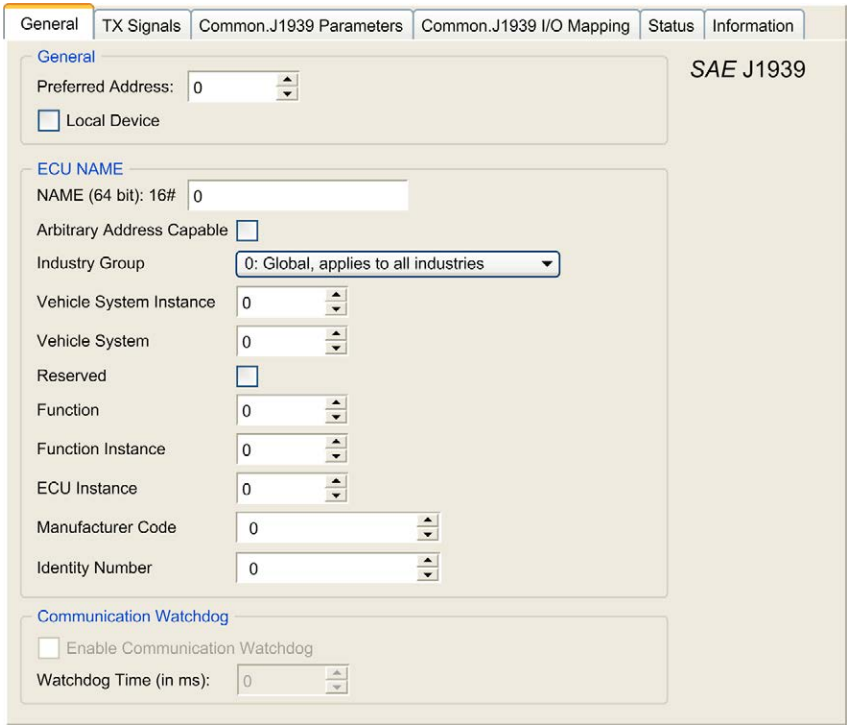
J1939 Manager Creation and Configuration

Proceed as follows to create and configure a J1939 Manager, if not already present, below the **CAN_1** node:

Step	Action
1	Click the **Plus** button  next to the **CAN_1** node in the **Devices** tree.
2	In the **Add Device** window, select **J1939_Manager** and click the **Add Device** button. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method • Using the Contextual Menu or Plus Button
3	Close the **Add Device** window.
4	Double-click **J1939_Manager (J1939_Manager)**. **Result:** The **J1939_Manager** configuration window appears: 
5	To configure the **J1939_Manager**, refer to *Programming with EcoStruxure Machine Expert / Device Editors / J1939 Configuration Editor / J1939 Manager Editor / Manager Editor* found in the EcoStruxure Machine Expert online help.

ECU Creation and Configuration

Proceed as follows to create and configure Electronic Control Units (ECUs):

Step	Action
1	Click the Plus button next to the **J1939_Manager (J1939_Manager)** node in the **Devices** tree.
2	In the **Add Device** window, select **J1939_ECU** and click the **Add Device** button. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method • Using the Contextual Menu or Plus Button
3	Close the **Add Device** window.
4	Double-click **J1939_ECU (J1939_ECU)**. **Result:** The **J1939_ECU** configuration window appears: 
5	To configure the **J1939_ECU**, refer to Configuring J1939 ECUs *(see page 203)*.

Configuring J1939 ECUs

As an overview, the following tasks must be generally accomplished:

- Add one **J1939_ECU** node for each physical J1939 device connected on the CAN bus.
- For each J1939 device, specify a unique **Preferred Address** in the range 1...253.
- For each J1939 device, configure the signals (SPNs) in the **TX Signals** tab. These signals are broadcast by the J1939 device to the other J1939 devices.
  Refer to the device documentation for information on the supported SPNs.
- Associate the SPN signals with variables in the **J1939 I/O Mapping** tab so that they can be processed by the application.
- When signals have been added, verify their settings in the **Conversion** window of the **TX signals** tab, for example, **Scaling**, **Offset**, and **Unit**. The J1939 protocol does not directly support REAL values, which are instead encoded in the protocol and so must be converted in the application. Similarly, in J1939 units are defined according to the International System of Units (SI) and therefore may need to be converted to values of other unit systems.
  Examples:
  - The **Engine Speed** signal of parameter group **EEC1** has a property `Scaling=0.125` that is encoded into a raw variable of type `ARRAY[0..1] OF BYTE`. Use the following ST code to convert this to a REAL variable:
    ```
    rRPM:=(Engine_Speed[1]*256 + Engine_Speed[0])*0.125;
    ```
  - The **Total Vehicle Distance** signal has properties `Scaling=0.125` and `Unit=km`, which are received in a (raw) variable of type `ARRAY[0..3] OF BYTE`. Use the following ST code to convert this to a REAL variable in mile units:
    ```
    rTVD := (Total_Vehicle_Distance[3]*EXPT(256,3) +
    Total_Vehicle_Distance[2]*EXPT(256,2) + Total_Vehicle_Distance[1]*2
    56 +
    Total_Vehicle_Distance[0])*0.125*0.621371;
    ```
  - The **Engine Coolant Temperature** signal of parameter group **ET1** has properties `Offset=-40` and `Unit=C(Celsius)`, which are received in a (raw) variable of type `BYTE`. Use the following ST code to convert it to a REAL variable in Fahrenheit units:
    ```
    rEngineCoolantTemperature := (Engine_Coolant_Temperature -
     40)*1.8 + 32;
    ```

For more details on how to configure the **J1939_ECU**, refer to *Programming with EcoStruxure Machine Expert / Device Editors / J1939 Configuration Editor / J1939 ECU Editor / ECU Editor* found in the EcoStruxure Machine Expert online help.

Configuring the M251 Logic Controller as an ECU Device

The controller can also be configured as a J1939 ECU device:

Step	Action
1	Add a **J1393_ECU** node to the **J1939_Manager**. Refer to ECU Creation and Configuration *(see page 202)*.
2	Select the **Local Device** option in the **General** tab.
3	Configure signals sent from the controller to other J1939 devices in the **TX Signals** tab. Parameter groups are either of type **Broadcast**, that is, sent to all devices, or **P2P** (Peer-to-Peer), that is, sent to one specified device.
4	For **P2P** signals, configure the **Destination Address** of the receiving J1939 ECU device in the parameter group properties window.
5	Add **P2P** signals sent by another J1939 device to the controller in the **RX Signals (P2P)** tab of the J1939 (local) device representing the controller.
6	Configure the **Source Address** of the parameter group by specifying the address of the sending J1939 device.

Chapter 15
OPC UA Server Configuration

Introduction

This chapter describes how to configure the OPC UA server of the M251 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

OPC UA Server Overview

Overview

The OPC Unified Architecture server (OPC UA server) allows the M251 Logic Controller to exchange data with OPC UA clients. Server and client communicate through sessions.

The monitored items of data (also referred to as symbols) to be shared by the OPC UA server are manually selected from a list of the IEC variables used in the application.

OPC UA uses a subscription model; clients subscribe to symbols. The OPC UA server reads the values of symbols from devices at a fixed sampling rate, places the data in a queue, then sends them to clients as notifications at a regular publishing interval. The sampling interval can be shorter than the publishing interval, in which case notifications may be queued until the publishing interval elapses.

Symbols that have not changed value since the previous sample are not re-published. Instead, the OPC UA server sends regular KeepAlive messages to indicate to the client that the connection is still active.

User and Group Access Rights

Access to the OPC UA server is controlled by user rights. Refer to Users and Groups in the EcoStruxure Machine Expert Programming Guide.

OPC UA Services

The following table describes the supported OPC UA services:

OPC UA Service	Description
Address Space Model	Yes
Session services	Yes
Attribute services	Yes
Monitored item services	Yes
Queued items	Yes
Subscription services	Yes
Publishing method	Yes

OPC UA Server Configuration

Introduction

The OPC UA Server Configuration window allows you to configure the OPC UA server.

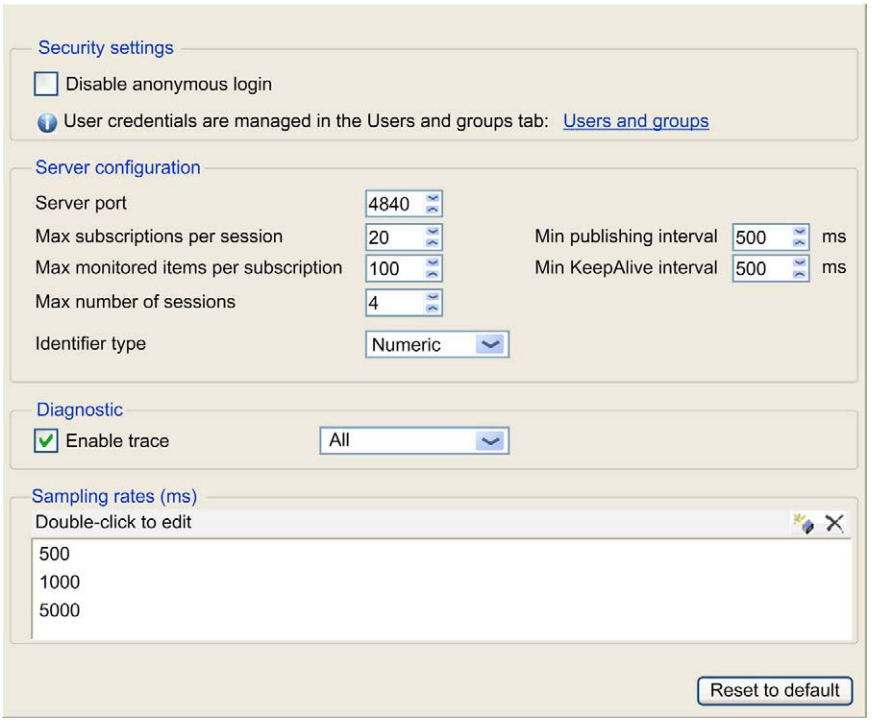
Accessing the OPC UA Server Configuration Tab

To configure the OPC UA Server:

Step	Action
1	In the **Devices tree**, double-click **MyController**.
2	Select the **OPC UA Server Configuration** tab.

OPC UA Server Configuration Tab

The following figure shows the OPC UA Server Configuration window:



OPC UA Server Configuration Description

This table describes the OPC UA Server Configuration parameters:

Parameter	Value	Default value	Description
Security Settings			
Disable anonymous login	Enabled/ Disabled	Disabled	By default, this checkbox is cleared, meaning that OPC UA clients can connect to the server anonymously. Select this checkbox to require that clients provide a valid user name and password to connect to the OPC UA server.
Server Configuration			
Server port	0...65535	4840	The port number of the OPC UA server. OPC UA clients must append this port number to the TCP URL of the controller to connect to the OPC UA server.
Max. subscriptions per session	1...100	20	Specify the maximum number of subscriptions allowed within each session.
Min. publishing interval	200...5000	1000	The publishing interval defines how frequently the OPC UA server sends notification packages to clients. Specify the minimum time that must elapse between notifications, in ms.
Max. monitored items per subscription	1...1000	100	The maximum number of *monitored items* in each subscription that the server assembles into a notification package.
Min. KeepAlive interval	500...5000	500	The OPC UA server only sends notifications when the values of monitored items of data are modified. A *KeepAlive* notification is an empty notification sent by the server to inform the client that although no data has been modified, the subscription is still active. Specify the minimum interval between KeepAlive notifications, in ms.
Max. number of sessions	1...4	2	The maximum number of clients that can connect simultaneously to the OPC UA server.
Identifier type	**Numeric** **String**	**Numeric**	Certain OPC UA clients require a specific format of unique symbol identifier (node ID). Select the format of the identifiers: ● Numeric values ● Text strings

Parameter	Value	Default value	Description
Diagnostic			
Enable trace	Enabled/disabled	Enabled	Select this checkbox to include OPC UA diagnostic messages in the controller log file. Traces are available from the **Log** tab or from the System Log File of the Web Server. You can select the category of events to write to the log file: • **None** • **Error** • **Warning** • **System** • **Information** • **Debug** • **Content** • **All** (default)
Sampling rates (ms)	**200...5000**	**500** **1000** **2000**	The sampling rate indicates a time interval, in milliseconds (ms). When this interval has elapsed, the server sends the notification package to the client. The sampling rate can be shorter than the publishing interval, in which case notifications are queued until the publishing interval has elapsed. Sampling rates must be in the range 200...5000 (ms). Up to 3 different sampling rates can be configured. Double-click on a sampling rate to edit its value. To add a sampling rate to the list, right-click and choose **Add a new rate**. To remove a sample rate from the list, select the value and click 

Click **Reset to default** to return the configuration parameters on this window to their default values.

OPC UA Server Symbols Configuration

Introduction

Symbols are the items of data shared with OPC UA clients. Symbols are selected from a list of all the IEC variables used in the application. The selected symbols are then sent to the logic controller as part of the application download.

Each symbol is assigned a unique identifier. As certain client types may require a specific format, identifiers can be configured to be in either string or numeric format.

The OPC UA server supports the following IEC variable types:

- Boolean
- Byte
- Int16, Int32, Int64
- UInt16, UInt32, UInt64
- Float
- Double
- String (255 bytes)
- Sbyte

Bit memory variables (%MX) cannot be selected.

Displaying the List of Variables

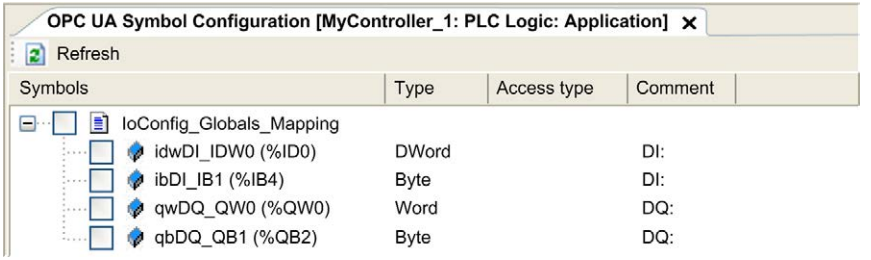
To display the list of variables:

Step	Action
1	On the **Applications tree** tab, right-click **Application** and choose **Add object → OPC UA Symbol Configuration**. **Result:** The OPC UA Symbols window is displayed. The logic controller starts the OPC UA server.
2	Click **Add**.

NOTE: The IEC objects `%MX`, `%IX`, `%QX` are not directly accessible. To access IEC objects you must first group their contents in located registers (refer to Relocation Table *(see page 34)*).

Selecting OPC UA Server Symbols

The **OPC UA Symbols** window displays the variables available for selection as symbols:



Select **IoConfig_Globals_Mapping** to select all the available variables. Otherwise, select individual symbols to share with OPC UA clients. A maximum of 1000 symbols can be selected.

Each symbol has the following properties:

Name	Description
Symbols	The variable name followed by the address of the variable.
Type	The data type of the variable.
Access type	Click repeatedly to specify the access rights of the symbol: read-only ( ) (default), write-only ( ), or read/write ( ). **NOTE:** Click in the **Access type** column of **IoConfig_Globals_Mapping** to set the access rights of all the symbols at once.
Comment	An optional comment.

Click **Refresh** to update the list of available variables.

OPC UA Server Performance

Overview

The following provides capacity and performance information for the OPC UA server of the M251 Logic Controller. Design considerations are also provided to help optimize the performance of the OPC UA server.

System Configurations Used to Evaluate Performance

OPC UA server performance is determined by the system configuration, the number of symbols being published, and the percentage of symbols being refreshed.

The following table presents the number of elements in small, medium, and large sample configurations used for evaluating OPC UA server performance:

Elements	Small	Medium	Large
EtherNet/IP adapters	0	7	0
Expansion modules	0	5	7
CANopen slave devices	0	1	63
PTO functions	0	4	4
HSC functions	0	8	8
Profibus connections	0	0	1
Modbus TCP slave devices	0	6	64

This table presents average read/write request times for each of the sample configurations and for different numbers of symbols:

Average Read/Write Request Times						
	Number of Symbols					
Configuration	50	100	250	400	500	1000
Small	42 ms	70 ms	151 ms	232 ms	284 ms	554 ms
Medium	73 ms	121 ms	265 ms	412 ms	514 ms	1024 ms
Large	520 ms	895 ms	2045 ms	3257 ms	4071 ms	7153 ms

The following tables present the average time required to refresh a monitored set of symbols using a sampling rate of 200 ms and a publishing interval of 200 ms.

This table presents the average time required to refresh 100% of symbols for each of the sample configurations:

Average Time to Refresh 100% of Symbols			
	Number of Symbols		
Configuration	100	400	1000
Small	214 ms	227 ms	254 ms
Medium	224 ms	250 ms	292 ms
Large	234 ms	330 ms	800 ms

This table presents the average time required to refresh 50% of symbols for each of the sample configurations:

Average Time to Refresh 50% of Symbols			
	Number of Symbols		
Configuration	100	400	1000
Small	211 ms	220 ms	234 ms
Medium	219 ms	234 ms	254 ms
Large	284 ms	300 ms	660 ms

This table presents the average time required to refresh 1% of symbols for each of the sample configurations:

Average Time to Refresh 1% of Symbols			
	Number of Symbols		
Configuration	100	400	1000
Small	210 ms	210 ms	212 ms
Medium	215 ms	217 ms	220 ms
Large	270 ms	277 ms	495 ms

Optimizing OPC UA Server Performance

The OPC UA server functionality is dependent on external communication networks, external device performance, and other external parameters. Data transmitted may be delayed or other possible communication errors may arise that impose practical limits on machine control. Do not use the OPC UA server functionality for safety-related data or other time-dependent purposes.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not allow safety-related data in OPC UA server data exchanges.
- Do not use OPC UA server data exchanges for any critical or time-dependent purposes.
- Do not use OPC UA server data exchanges to change equipment states without having done a risk analysis and implementing appropriate safety-related measures.

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

The above tables can be useful in determining whether OPC UA server performance is within acceptable limits. Be aware, however, that other external factors influence overall system performance, such as the volume of Ethernet traffic.

To optimize OPC UA server performance, consider the following:

- Minimize Ethernet traffic by setting the **Min. publishing interval** to the lowest value that yields an acceptable response time.
- The task cycle time *(see page 41)* configured for the M251 Logic Controller must be less than the configured **Min. publishing interval** value.
- Configuring a **Max. number of sessions** (the number of OPC UA clients that can simultaneously connect to the OPC UA server) value of greater than 1 decreases the performance of all sessions.
- The sampling rate determines the frequency at which data is exchanged. Tune the **Sampling rates (ms)** value to product the lowest response time that does not adversely affect the overall performance of the logic controller.

Chapter 16
Post Configuration

Introduction

This chapter describes how to generate and configure the post configuration file of the Modicon M251 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

Post Configuration Presentation

Introduction

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

By default, all parameters are set in the application.The parameters defined in the Post Configuration file are used instead of the corresponding parameters defined in the application. Not all parameters have to be specified in the Post Configuration file (for example: one parameter can change the IP address without changing the Gateway Address).

Parameters

The Post Configuration file allows you to change network parameters.

Ethernet parameters:
- IP Address
- Subnet Mask
- Gateway Address
- Transfer Rate
- IP Config Mode
- Device Name
- IP Master Address *(see page 151)*

Serial Line parameters, for each serial line in the application (embedded port or PCI module):
- Baud rate
- Parity
- Data bits
- Stop bit

Profibus parameters, for each Profibus in the application (TM4 module):
- Station address
- Baud rate

NOTE: Parameter updates with a Post Configuration file that impacts parameters used by other devices via a communication port are not updated in the other devices.

For example, if the IP address used by an HMI is updated in the configuration with a Post Configuration file, the HMI uses the previous address. You must update the address used by the HMI independently.

Operating Mode

The Post Configuration file is read after:

- A Reset Warm command *(see page 62)*
- A Reset Cold command *(see page 62)*
- A reboot *(see page 64)*
- An application download *(see page 66)*

Refer to Controller States and Behaviors *(see page 47)* for further details on controller states and transitions.

Post Configuration File Management

Introduction

The file **Machine.cfg** is located in the directory */usr/cfg*.

Each parameter is specified by a variable type, variable ID, and value. The format is:

```
id[moduleType].pos[param1Id].id[param2Id].param[param3Id].paramField=value
```

where you have to modify only value.

Each parameter is defined on three lines in the Post Configuration file:

- The first line describes the internal 'path' for this parameter.
- The second line is a comment describing the parameter in a comprehensive way.
- The third line is the definition of the parameter (as described above) with its value.

Post Configuration File Generation

The Post Configuration file (**Machine.cfg**) is generated by EcoStruxure Machine Expert.

To generate the file, proceed as follows:

Step	Action
1	In the menu bar, choose **Build → Post Configuration → Generate...** **Result:** An explorer window is displayed.
2	Select the destination folder of the Post Configuration file.
3	Click **OK**.

When you use EcoStruxure Machine Expert to create a Post Configuration file (**Generate**), it reads the value of each parameter assigned in your application program and then writes the values to the **Machine.cfg** Post Configuration file. After generating a Post Configuration file, review the file and remove any parameter assignments that you wish to remain under the control of your application. Keep only those parameter assignments that you wish changed by the Post Configuration function that are necessary to make your application portable and then modify those values appropriately.

Post Configuration File Transfer

After creating and modifying your Post Configuration file, transfer it to the */usr/cfg* directory of the controller. The controller does not read the **Machine.cfg** file unless it is in this directory.

You can transfer the Post Configuration file by the following methods:

- SD card *(see page 228)* (with the proper script)
- Download through the FTP server *(see page 121)*
- Download with EcoStruxure Machine Expert controller device editor *(see page 72)*

Modifying a Post Configuration File

If the Post Configuration file is located in the PC, use a text editor to modify it.

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

To modify the Post Configuration file directly in the controller, use the **Setup** menu of the Web server *(see page 105)*.

To modify the Post Configuration file in the controller with EcoStruxure Machine Expert in online mode:

Step	Action
1	In the **Devices tree**, click the controller name.
2	Click **Build** → **Post Configuration** → **Edit...** **Result:** The Post Configuration file opens in a text editor.
3	Edit the file.
4	If you want to apply the modifications after saving them, select **Reset device after sending**.
5	Click **Save as**.
6	Click **Close**.

NOTE: If the parameters are invalid, they are ignored.

Deleting the Post Configuration File

You can delete the Post Configuration file by the following methods:

- SD card (with the delete script)
- Through the FTP server *(see page 121)*
- Online with EcoStruxure Machine Expert controller device editor *(see page 72)*, **Files** tab

For more information on **Files** tab of the Device Editor, refer to EcoStruxure Machine Expert Programming Guide.

NOTE:
The parameters defined in the application are used instead of the corresponding parameters defined in the Post Configuration file after:

- A Reset Warm command *(see page 62)*
- A Reset Cold command *(see page 62)*
- A reboot *(see page 64)*
- An application download *(see page 66)*

Post Configuration Example

Post Configuration File Example for the TM251MESE

```
# TM251MESE / Ethernet_1 / IPAddress
# Ethernet IP address
id[45000].pos[2].id[45111].param[0] = [192, 168, 2, 24]

# TM251MESE / Ethernet_1 / SubnetMask
# Ethernet IP mask
id[45000].pos[2].id[45111].param[1] = [255, 255, 255, 0]

# TM251MESE / Ethernet_1 / GatewayAddress
# Ethernet IP gateway address
id[45000].pos[2].id[45111].param[2] = [0, 0, 0, 0]

# TM251MESE / Ethernet_1 / IPConfigMode
# IP configuration mode: 0:FIXED 1:BOOTP 2:DHCP
id[45000].pos[2].id[45111].param[4] = 0

# TM251MESE / Ethernet_1 / DeviceName
# Name of the device on the Ethernet network
id[45000].pos[2].id[45111].param[5] = 'my_Device'

# TM251MESE / Ethernet_2 / IPAddress
# Ethernet IP address
id[45000].pos[3].id[111].param[0] = [192, 168, 1, 24]

# TM251MESE / Ethernet_2 / SubnetMask
# Ethernet IP mask
id[45000].pos[3].id[111].param[1] = [255, 255, 255, 0]

# TM251MESE / Ethernet_2 / GatewayAddress
# Ethernet IP gateway address
id[45000].pos[3].id[111].param[2] = [0, 0, 0, 0]
```

```
# TM251MESE / Ethernet_2 / IPConfigMode
# IP configuration mode: 0:FIXED 1:BOOTP 2:DHCP
id[45000].pos[3].id[111].param[4] = 0

# TM251MESE / Ethernet_2 / DeviceName
# Name of the device on the Ethernet network
id[45000].pos[3].id[111].param[5] = 'my_Device'

# TM251MESE / Serial_Line_1 / Serial Line Configuration / Baudrate
# Serial Line Baud Rate in bit/s
id[45000].pos[4].id[40101].param[10000].Bauds = 115200

# TM251MESE / Serial_Line_1 / Serial Line Configuration / Parity
# Serial Line Parity (0=None, 1=Odd, 2=Even)
id[45000].pos[4].id[40101].param[10000].Parity = 0

# TM251MESE / Serial_Line_1 / Serial Line Configuration / DataBits
# Serial Line Data bits (7 or 8)
id[45000].pos[4].id[40101].param[10000].DataFormat = 8

# TM251MESE / Serial_Line_1 / Serial Line Configuration / StopBits
# Serial Line Stop bits (1 or 2)
id[45000].pos[4].id[40101].param[10000].StopBit = 1
```

Post Configuration File Example for the TM251MESC

```
# TM251MESC / Ethernet_1 / IPAddress
# Ethernet IP address
id[45000].pos[2].id[45111].param[0] = [0, 0, 0, 0]

# TM251MESC / Ethernet_1 / SubnetMask
# Ethernet IP mask
id[45000].pos[2].id[45111].param[1] = [0, 0, 0, 0]
```

```
# TM251MESC / Ethernet_1 / GatewayAddress
# Ethernet IP gateway address
id[45000].pos[2].id[45111].param[2] = [0, 0, 0, 0]

# TM251MESC / Ethernet_1 / IPConfigMode
# IP configuration mode: 0:FIXED 1:BOOTP 2:DHCP
id[45000].pos[2].id[45111].param[4] = 0

# TM251MESC / Ethernet_1 / DeviceName
# Name of the device on the Ethernet network
id[45000].pos[2].id[45111].param[5] = 'my_Device'

# TM251MESC / Serial_Line_1 / Serial Line Configuration / Baudrate
# Serial Line Baud Rate in bit/s
id[45000].pos[4].id[40101].param[10000].Bauds = 115200

# TM251MESC / Serial_Line_1 / Serial Line Configuration / Parity
# Serial Line Parity (0=None, 1=Odd, 2=Even)
id[45000].pos[4].id[40101].param[10000].Parity = 0

# TM251MESC / Serial_Line_1 / Serial Line Configuration / DataBits
# Serial Line Data bits (7 or 8)
id[45000].pos[4].id[40101].param[10000].DataFormat = 8

# TM251MESC / Serial_Line_1 / Serial Line Configuration / StopBits
# Serial Line Stop bits (1 or 2)
id[45000].pos[4].id[40101].param[10000].StopBit = 1
```

Chapter 17
Connecting a Modicon M251 Logic Controller to a PC

Connecting the Controller to a PC

Overview

To transfer, run, and monitor the applications, connect the controller to a computer, that has EcoStruxure Machine Expert installed, using either a USB cable or an Ethernet connection (for those references that support an Ethernet port).

NOTICE
INOPERABLE EQUIPMENT Always connect the communication cable to the PC before connecting it to the controller. **Failure to follow these instructions can result in equipment damage.**

USB Powered Download

In order to execute limited operations, the M251 Logic Controller has the capability to be powered through the USB Mini-B port. A diode mechanism avoids having the logic controller both powered by USB and by the normal power supply, or to supply voltage on the USB port.

When powered only by USB, the logic controller executes the firmware and the boot project (if any) and the I/O board is not powered during boot (same duration as a normal boot). USB powered download initializes the internal flash memory with some firmware or some application and parameters when the controller is powered by USB. The preferred tool to connect to the controller is the **Controller Assistant**. Refer to the *EcoStruxure Machine Expert Controller Assistant User Guide*.

The controller packaging allows easy access to USB Mini-B port with minimum opening of the packaging. You can connect the controller to the PC with a USB cable. Long cables are not suitable for the USB powered download.

⚠ WARNING
INSUFFICENT POWER FOR USB DOWNLOAD Do not use a USB cable longer than 3m (9.8 ft) for USB powered download. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: It is not intended that you use the USB Powered Download on an installed controller. Depending on the number of I/O expansion modules in the physical configuration of the installed controller, there may be insufficient power from your PC USB port to accomplish the download.

USB Mini-B Port Connection

TCSXCNAMUM3P: This USB cable is suitable for short duration connections such as quick updates or retrieving data values.

BMXXCAUSBH018: Grounded and shielded, this USB cable is suitable for long duration connections.

NOTE: You can only connect 1 controller or any other device associated with EcoStruxure Machine Expert and its component to the PC at any one time.

The USB Mini-B Port is the programming port you can use to connect a PC with a USB host port using EcoStruxure Machine Expert software. Using a typical USB cable, this connection is suitable for quick updates of the program or short duration connections to perform maintenance and inspect data values. It is not suitable for long-term connections such as commissioning or monitoring without the use of specially adapted cables to help minimize electromagnetic interference.

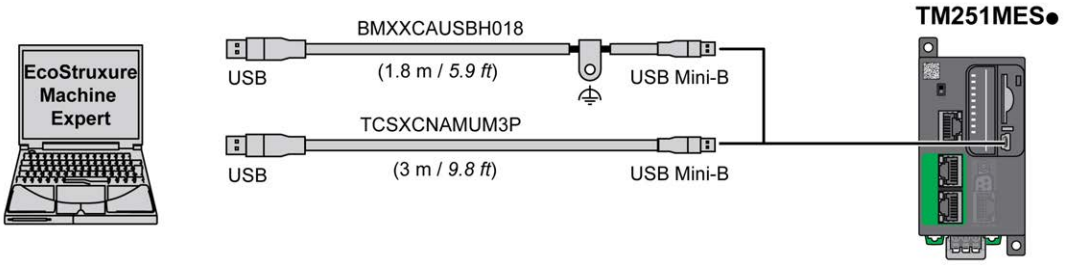
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- You must use a shielded USB cable such as a BMX XCAUSBH0•• secured to the functional ground (FE) of the system for any long-term connection.
- Do not connect more than one controller or bus coupler at a time using USB connections.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

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

The communication cable should be connected to the PC first to minimize the possibility of electrostatic discharge affecting the controller.

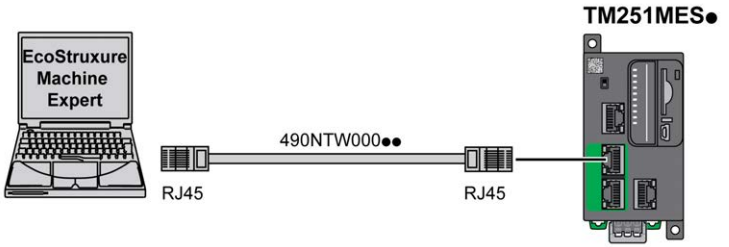


To connect the USB cable to your controller, follow the steps below:

Step	Action
1	**1a** If making a long-term connection using the cable BMXXCAUSBH018, or other cable with a ground shield connection, be sure to securely connect the shield connector to the functional ground (FE) or protective ground (PE) of your system before connecting the cable to your controller and your PC. **1b** If making a short-term connection using the cable TCSXCNAMUM3P or other non-grounded USB cable, proceed to step 2.
2	Connect your USB cable to the computer.
3	Open the hinged access cover.
4	Connect the Mini connector of your USB cable to the controller USB connector.

Ethernet Port Connection

You can also connect the controller to a PC using an Ethernet cable.



To connect the controller to the PC, do the following:

Step	Action
1	Connect the Ethernet cable to the PC.
2	Connect the Ethernet cable to either of the Ethernet 1 ports on the controller.

Chapter 18
SD Card

Introduction

This chapter describes how to transfer firmware, application, using an SD card to the Modicon M251 Logic Controller.

What Is in This Chapter?

This chapter contains the following topics:

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.

Script files can be used to:

- Configure the Ethernet firewall *(see page 163)*.
- Perform file transfer operations. The script files for these commands can be generated automatically and the necessary files copied to the SD card using the **Mass Storage (USB or SD Card)** command.
- Change the Modbus slave port *(see page 155)* 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 M251 Logic Controller allows file transfers with an SD card.

To upload or download files to the controller with an SD card, use one of the following methods:

- The clone function *(see page 230)* (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 M251 Logic Controller accepts only SD cards formatted in FAT or FAT32.

The SD card must have a label. To add a label, insert the SD card in your PC, right-click on the drive in Windows Explorer and choose **Properties**.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

If you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE
INOPERABLE EQUIPMENT • Do not interrupt the transfer of the application program or a firmware change once the transfer has begun. • Re-initiate the transfer if the transfer is interrupted for any reason. • Do not attempt to place the device into service until the file transfer has completed successfully. **Failure to follow these instructions can result in equipment damage.**

Clone Function

The clone function allows you to upload the application from one controller and to download it only to a same controller reference.

This function clones every parameter of the controller (for example applications, firmware, data file, post configuration). Refer to Memory Mapping *(see page 25)*.

NOTE: User access rights can only be copied if the **Include User Rights** button has previously been clicked on the **Clone Management** subpage of the Web server.

By default, clone is allowed without using the function block **FB_ControlClone**. If you want to restrict access to the clone feature, you can remove the access rights of the `USBExecCommand object` on **Everyone** group *(see page 80)*. As a result, cloning will be not allowed without using **FB_ControlClone**. For more details about this function block, refer to the M262 System Library Guide *(see Modicon M251 Logic Controller, System Functions and Variables, PLCSystem Library Guide)*. For more details about Access Rights, refer to the EcoStruxure Machine Expert Programming Guide.

If you wish to control access to the cloned application in the target controller, you must use the **Include users rights** button (on the **Clone Management** subpage of the Web Server *(see page 118)*) of the source controller before doing the clone operation. For more details about Access Rights, refer to the EcoStruxure Machine Expert Programming Guide.

This procedure describes how to upload the application stored in the controller to your SD card:

Step	Action
1	Erase an SD card and set the card label as follows: **CLONExxx** **NOTE:** The label must begin with '**CLONE**' (not case sensitive), followed by any normal character.
2	Select if you want to clone the **Users Rights**. Refer to the **Clone Management** subpage *(see page 118)* of the web server.
3	Remove power from the controller.
4	Insert the prepared SD card in the controller.
5	Restore power to the controller. **Result:** The clone procedure starts automatically. During the clone procedure, the **PWR** and **I/O** LEDs are ON and the **SD** LED flashes regularly. **NOTE:** The clone procedure lasts 2 or 3 minutes. **Result:** At the end of the clone procedure, the SD LED is ON and the controller starts in normal application mode. If an error was detected, the ERR LED is ON and the controller is in STOPPED state.
6	Remove the SD card from the controller.

This procedure describes how to download the application stored in the SD card to your controller:

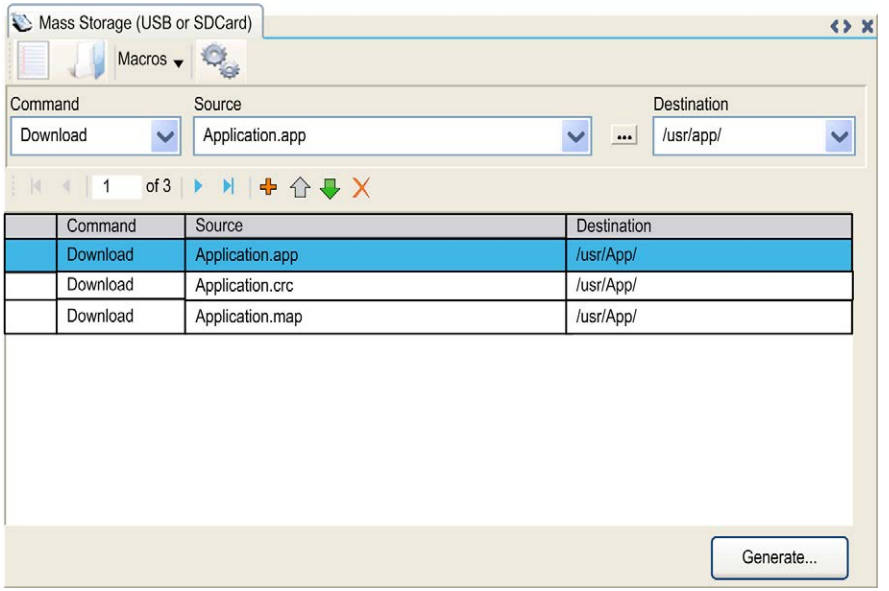
Step	Action
1	Remove power from the controller.
2	Insert the SD card into the controller.
3	Restore power to the controller. **Result:** The download procedure starts and the SD LED is flashing during this procedure.
4	Wait until the end of the download: ● If the **SD** LED (green) is ON, and the **ERR** LED (red) flashes regularly, the download ended successfully. ● If the **SD** LED (green) is OFF, and the **ERR** and **I/O** LEDs (red) flash regularly, an error is detected.
5	Remove the SD card to restart the controller.

NOTE: If you wish to control access to the cloned application in the target controller, you will need to enable and establish user access-rights, and any Web Server/FTP passwords, which are controller-specific. For more details about Access Rights, refer to the EcoStruxure Machine Expert Programming Guide.

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

Script and Files Generation with Mass Storage

Click **Project → Mass Storage (USB or SDCard)** in the main menu:



Element	Description
New	Create a new script.
Open	Open a script.
Macros	Insert a Macro. A macro is a sequence of unitary commands. A macro helps to perform many common operations such as upload application, download application, and so on.
Generate	Generate the script and all necessary files on the SD card.
Command	Basic instructions.
Source	Source file path on the PC or the controller.
Destination	Destination directory on the PC or the controller.
Add New	Add a script command.
Move Up/Down	Change the script commands order.
Delete	Delete a script command.

Commands descriptions:

Command	Description	Source	Destination	Syntax
Download	Downloads a file from the SD card to the controller.	Select the file to download.	Select the controller destination directory.	`'Download "/usr/Cfg/*"'`
SetNodeName	Sets the node name of the controller.	New node name.	Controller node name	`'SetNodeName "Name_PLC"'`
	Resets the node name of the controller.	Default node name.	Controller node name	`'SetNodeName ""'`
Upload	Uploads files contained in a controller directory to the SD card.	Select the directory.	-	`'Upload "/usr/*"'`
Delete	Deletes files contained in a controller directory. **NOTE:** Delete "*" does not delete system files.	Select the directory and enter a specific file name **Important:** by default, all directory files are selected.	-	`'Delete "/usr/SysLog/*"'`
	Removes the UserRights from the controller.	-	-	`'Delete "/usr/*"'`
	Deletes the files contained in the SD card or a folder of the SD card	-	-	`'Delete "/sd0/*"'` or `'Delete "/sd0/folder name"'`
Reboot	Restarts the controller (only available at the end of the script).	-	-	`'Reboot'`

NOTE: When User Rights are activated on a controller and if the user is not allowed to read/write/delete file system, scripts used to **Upload**/**Download**/**Delete** files are disabled. It includes the clone operation. For more details about User Rights, refer to the EcoStruxure Machine Expert Programming Guide.

This table describes the macros:

Macros	Description	Directory/Files

Download App	Download the application from the SD card to the controller.	`/usr/App/*.app`
		`/usr/App/*.crc`
		`/usr/App/*.map`
		`/usr/App/*.conf`(1)

Upload App	Upload the application from the controller to the SD card.	
Download Sources	Download the project archive from the SD card to the controller.	`/usr/App/*.prj`
Upload Sources	Upload the project archive from the controller to the SD card.	
Download Multi-files	Download multiple files from the SD card to a controller directory.	Defined by user
Upload Log	Upload the log files from the controller to the SD card.	`/usr/Log/*.log`
(1) If OPC UA *(see page 207)* is configured.		

Reset the User Rights to Default

You can manually create a script to remove the user rights, along with the application, from the controller. This script must contain this command:

```
delete /usr/*
Reboot
```

NOTE: This command also removes user application and data.

Step	Action
1	Remove power from the controller.
2	Insert the prepared SD card in the source controller.
3	Restore power to the source controller. **Result:** The copy starts automatically. During the copy, the **PWR** and **I/O** LEDs are ON and the **SD** LED flashes regularly.
4	Wait until the copy is completed. **Result:** The SD LED is ON and the controller reboots with default user rights. If an error was detected, the ERR LED is ON and the controller is in STOPPED state.

Transfer Procedure

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Step	Action
1	Create the script with the **Mass Storage (USB or SDCard)** editor.
2	Click **Generate...** and select the SD card root directory. **Result**: The script and files are transferred on the SD card.
3	Insert the SD card into the controller. **Result:** The transfer procedure starts and the **SD** LED is flashing during this procedure.
4	Wait until the end of the download: ● If the **SD** LED (green) is ON, and the **ERR** LED (red) flashes regularly, the download ended successfully. ● If the **SD** LED (green) is OFF, and the **ERR** and **I/O** LEDs (red) flash regularly, an error is detected.
5	Remove the SD card from the controller. **NOTE:** Changes will be applied after next restart.

When the controller has executed the script, the result is logged on the SD card (file `/sys/cmd/Cmd.log`).

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Consult the controller state and behavior diagram in this document to understand the state that will be assumed by the controller after you cycle power.

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

Updating Modicon M251 Logic Controller Firmware

Introduction

The firmware updates for Modicon M251 Logic Controller are available on the http://www.schneider-electric.com website (in .zip format).

Updating the firmware is possible by:

- Using an SD card with a compatible script file
- Using the **Controller Assistant**

Performing a firmware update deletes the current application program in the device, including the Boot Application in Flash memory.

NOTICE
LOSS OF APPLICATION DATA ● Perform a backup of the application program to the hard disk of the PC before attempting a firmware update. ● Restore the application program to the device after a successful firmware update. **Failure to follow these instructions can result in equipment damage.**

If you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE
INOPERABLE EQUIPMENT ● Do not interrupt the transfer of the application program or a firmware change once the transfer has begun. ● Re-initiate the transfer if the transfer is interrupted for any reason. ● Do not attempt to place the device into service until the file transfer has completed successfully. **Failure to follow these instructions can result in equipment damage.**

The serial line ports of your controller are configured for the Machine Expert protocol by default when new or when you update the controller firmware. The Machine Expert protocol is incompatible with that of other protocols such as Modbus Serial Line. Connecting a new controller to, or updating the firmware of a controller connected to, an active Modbus configured serial line can cause the other devices on the serial line to stop communicating. Make sure that the controller is not connected to an active Modbus serial line network before first downloading a valid application having the concerned port or ports properly configured for the intended protocol.

NOTICE
INTERRUPTION OF SERIAL LINE COMMUNICATIONS Be sure that your application has the serial line ports properly configured for Modbus before physically connecting the controller to an operational Modbus Serial Line network. **Failure to follow these instructions can result in equipment damage.**

Updating Firmware by SD Card

Follow these steps to update the firmware by an SD card:

Step	Action
1	Extract the .zip file to the root of the SD card. **NOTE:** The SD card folder \sys\cmd\ contains the download script file.
2	Remove power from the controller.
3	Insert the SD card into the controller.
4	Restore power to the controller. **NOTE:** The SD LED (green) is flashing during the operation.
5	Wait until the end of the download: ● If the **SD** LED (green) is ON, and the **ERR** LED (red) flashes regularly, the download ended successfully. ● If the **SD** LED (green) is OFF, and the **ERR** and **I/O** LEDs (red) flash regularly, an error is detected.
6	Remove the SD card from the controller. **Result**: The controller restarts automatically with new firmware if the download ended successfully.

Updating Firmware by Controller Assistant

To update the firmware, you must open the **Controller Assistant**. Click **Tools** → **External Tools** → **Open Controller Assistant**.

To execute a complete firmware update of a controller without replacing the Boot application and data, proceed as follows:

Step	Action
1	On the **Home** dialog, click the **Read from....** controller button. **Result**: The **Controller selection** dialog opens.
2	Select the required connection type and controller and click the **Reading** button. **Result**: The image is transmitted from the controller to the computer. After this has been accomplished successfully, you are automatically redirected to the **Home** dialog.
3	Click the button **New / Process...** and then **Update firmware...**. **Result**: The dialog for updating the firmware opens.
4	Execute individual steps for updating the firmware in the current image (Changes are only effected in the image on your computer). In the final step, you can decide whether you want to create a backup copy of the image read by the controller. **Result**: Following the update of the firmware, you are automatically returned to the **Home** dialog.
5	On the **Home** dialog, click the **Write on....** controller button. **Result**: The **Controller selection** dialog opens.
6	Select the required connection type and controller and click the **Write** button. **Result**: The image is transmitted from your computer to the controller. After the transmission, you are automatically returned to the **Home** dialog.

For more information about the firmware update and creating a new flash disk with firmware, refer to Project Settings - Firmware Update and Flash Memory Organization *(see page 30)*.

Chapter 19
Firmware Management

Updating TM3 Expansion Modules Firmware

Overview

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

Downloading Firmware to TM3 Expansion Modules

The firmware can be updated in:
- TM3X•HSC•
- TM3DI16 and TM3DI16G with firmware version (SV) ≥ 2.0
- TM3A• and TM3T• with firmware version (SV) ≥ 2.0

NOTE: The firmware version (SV) is found on the packaging and product labels.

Firmware updates are performed if, during a power on, at least one firmware file is present in the `/usr/TM3fwupdate/` directory of controller. You can download the file(s) to the controller using the SD card, an FTP file transfer or through EcoStruxure Machine Expert.

The controller updates the firmware of the TM3 expansion modules on the I/O bus, including those that are:
- Connected remotely, using a TM3 Transmitter/Receiver module.
- In configurations comprising a mix of TM3 and TM2 expansion modules.

The following table describes how to download firmware to one or more TM3 expansion modules using an SD card:

Step	Action
1	Insert an empty SD card into the PC.
2	Create the folder path `/sys/Cmd` and create a file called `Script.cmd`.
3	Edit the file and insert the following command for each firmware file you wish to transfer to the controller: `Download "usr/TM3fwupdate/<filename>"`
4	Create the folder path `/usr/TM3fwupdate/` in the SD card root directory and copy the firmware files to the `TM3fwupdate` folder.
5	Ensure that power is removed from controller.
6	Remove the SD card from the PC and insert it into the SD card slot of the controller.

Step	Action
7	Restore power to the controller. Wait until the end of the operation (until the **SD** LED is green ON). **Result:** The controller begins transferring the firmware file(s) from the SD card to the `/usr/TM3fwupdate` in the controller. During this operation, the **SD** LED on the controller is flashing. A `SCRIPT.log` file is created on the SD card and contains the result of the file transfer. If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `SCRIPT.log` file.
8	Remove power from the controller.
9	Remove SD card from the controller.
10	Restore power to the controller. **Result:** The controller transfers the firmware file(s) to the appropriate TM3 I/O module(s). **NOTE:** The TM3 update process adds approximatively 15 seconds to the controller boot duration.
11	Verify in the message logger of the controller that the firmware is successfully updated: `Your TM3 Module X successfully updated`. X corresponds to the position of the module on the bus. **NOTE:** You can also obtain the logger information in the `PlcLog.txt` file in the `/usr/Syslog/` directory of the controller file system. **NOTE:** If the controller encounters an error during the update, the update terminates with that module.
12	If all targeted modules were successfully updated, delete the firmware file(s) from `/usr/TM3fwupdate/` folder on the controller. You can delete the files directly using EcoStruxure Machine Expert or by creating and executing a script containing the following command: `Delete "usr/TM3fwupdate/*"` **NOTE:** If a targeted module was not updated successfully, or there are no message logger messages for all the targeted modules, see the Recovery Procedure *(see page 241)* below.

Recovery Procedure

If you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE
INOPERABLE EQUIPMENT • Do not interrupt the transfer of the application program or a firmware change once the transfer has begun. • Re-initiate the transfer if the transfer is interrupted for any reason. • Do not attempt to place the device into service until the file transfer has completed successfully. **Failure to follow these instructions can result in equipment damage.**

If, during the reattempted firmware update, the update prematurely terminates with an error, it means that the communication interruption or power outage had damaged the firmware of one of your modules in your configuration, and that module must be reinitialized.

NOTE: Once the firmware update process detects an error with the firmware in the destination module, the update process is terminated. After you have reinitialized the damaged module following the recovery procedure, any modules that followed the damaged module remain unchanged and will need to have their firmware updated.

The following table describes how to reinitialize the firmware on TM3 expansion modules:

Step	Action
1	Ensure that the correct firmware is present in the `/usr/TM3fwupdate/` directory of the controller.
2	Remove power from the controller.
3	Disassemble from the controller all TM3 expansion modules that are functioning normally, up to the first module to recover. Refer to the hardware guides of the modules for disassembly instructions.
4	Apply power to the controller. **NOTE:** The TM3 update process adds approximatively 15 seconds to the controller boot duration.
5	Verify in the message logger of the controller that the firmware is successfully updated: `Your TM3 Module X successfully updated`. X corresponds to the position of the module on the bus.
6	Remove power from the controller.

Step	Action
7	Reassemble the TM3 expansion module configuration to the controller. Refer to the hardware guides of the modules for assembly instructions.
8	Restore power to the controller. **Result:** The controller transfers the firmware file(s) to the appropriate and yet to be updated TM3 I/O module(s). **NOTE:** The TM3 update process adds approximatively 15 seconds to the controller boot duration.
9	Verify in the message logger of the controller that the firmware is successfully updated: `Your TM3 Module X successfully updated`. X corresponds to the position of the module on the bus. **NOTE:** You can also obtain the logger information in the `Sys.log` file in the `/usr/Log` directory of the controller file system.
10	Delete the firmware file(s) from `/usr/TM3fwupdate/` folder on the controller.

Appendices

Overview

This appendix lists the documents necessary for technical understanding of the Modicon M251 Logic Controller Programming Guide.

What Is in This Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	How to Change the IP Address of the Controller	245
B	Functions to Get/Set Serial Line Configuration in User Program	249
C	Controller Performance	255

Appendix A
How to Change the IP Address of the Controller

changeIPAddress: Change the IP address of the controller

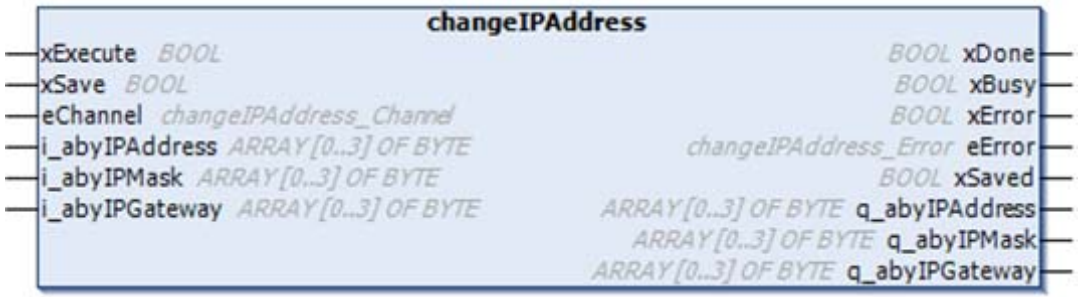
Function Block Description

The `changeIPAddress` function block provides the capability to change dynamically a controller IP address, its subnet mask and its gateway address. The function block can also save the IP address so that it is used in subsequent reboots of the controller.

NOTE: Changing the IP addresses is only possible if the IP mode is configured to **fixed IP address**. For more details, refer to IP Address Configuration *(see page 97)*.

NOTE: For more information on the function block, use the **Documentation** tab of EcoStruxure Machine Expert Library Manager Editor. For the use of this editor, refer EcoStruxure Machine Expert Programming Guide *(see EcoStruxure Machine Expert, Functions and Libraries User Guide)*.

Graphical Representation



Parameter Description

Input	Type	Comment
xExecute	BOOL	• Rising edge: action starts. • Falling edge: resets outputs. If a falling edge occurs before the function block has completed its action, the outputs operate in the usual manner and are only reset if either the action is completed or in the event that an error is detected. In this case, the corresponding output values (xDone, xError, iError) are present at the outputs for exactly one cycle.
xSave	BOOL	TRUE: save configuration for subsequent reboots of the controller.
eChannel	changeIPAd-dress_Channel	The input eChannel is the Ethernet port to be configured. Depending on the number of the ports available on the controller, it is one of 5 values *(see page 247)* in changeIPAddress_Channel (0 or 1).
i_abyIPAddress	ARRAY[0..3] OF BYTE	The new IP Address to be configured. Format: 0.0.0.0. **NOTE:** If this input is set to 0.0.0.0 then the controller default IP addresses *(see page 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	changeIPAd-dress_Error	Error code of the detected error *(see page 247)*.
xSaved	BOOL	Configuration saved for the subsequent reboots of the controller.
q_abyIPAddress	ARRAY[0..3] OF BYTE	Current controller IP address. Format: 0.0.0.0.
q_abyIPMask	ARRAY[0..3] OF BYTE	Current subnet mask. Format: 0.0.0.0.
q_abyIPGateway	ARRAY[0..3] OF BYTE	Current gateway IP address. Format: 0.0.0.0.

changeIPAddress_Channel: Ethernet port to be configured

The changeIPAddress_Channel enumeration data type contains the following values:

Enumerator	Value	Description
CHANNEL_ETHERNET_NETWORK	0	M241, M251MESC, M258, LMC058, LMC078: **Ethernet port** M251MESE: **Ethernet_2 port**
CHANNEL_DEVICE_NETWORK	1	M241: **TM4ES4 Ethernet port** M251MESE: **Ethernet_1 port**

changeIPAddress_Error: Error Codes

The changeIPAddress_Error enumeration data type contains the following values:

Enumerator	Value	Description
ERR_NO_ERROR	00 hex	No error detected.
ERR_UNKNOWN	01 hex	Internal error detected.
ERR_INVALID_MODE	02 hex	IP address is not configured as a fixed IP address.
ERR_INVALID_IP	03 hex	Invalid IP address.
ERR_DUPLICATE_IP	04 hex	The new IP address is already used in the network.
ERR_WRONG_CHANNEL	05 hex	Incorrect Ethernet communication port.
ERR_IP_BEING_SET	06 hex	IP address is already being changed.
ERR_SAVING	07 hex	IP addresses not saved due to a detected error or no non-volatile memory present.
ERR_DHCP_SERVER	08 hex	A DHCP server is configured on this Ethernet communication port.

Appendix B
Functions to Get/Set Serial Line Configuration in User Program

Overview

This section describes the functions to get/set the serial line configuration in your program.

To use these functions, add the **M2xx Communication** library.

For further information on adding a library, refer to the EcoStruxure Machine Expert Programming Guide.

What Is in This Chapter?

This chapter contains the following topics:

GetSerialConf: Get the Serial Line Configuration

Function Description

`GetSerialConf` returns the configuration parameters for a specific serial line communication port.

Graphical Representation

Parameter Description

Input	Type	Comment
`Link`	LinkNumber *(see EcoStruxure Machine Expert, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	`Link` is the communication port number.
`PointerToSerialConf`	POINTER TO `SERIAL_CONF` *(see page 253)*	`PointerToSerialConf` is the address of the configuration structure (variable of `SERIAL_CONF` type) in which the configuration parameters are stored. The `ADR` standard function must be used to define the associated pointer. (See the example below.)

Output	Type	Comment
`GetSerialConf`	`WORD`	This function returns: ● 0: The configuration parameters are returned ● 255: The configuration parameters are not returned because: ❍ the function was not successful ❍ the function is in progress

Example

Refer to the `SetSerialConf` *(see page 252)* example.

SetSerialConf: Change the Serial Line Configuration

Function Description

`SetSerialConf` is used to change the serial line configuration.

Graphical Representation

NOTE: Changing the configuration of the Serial Line(s) port(s) during programming execution can interrupt ongoing communications with other connected devices.

⚠ WARNING

LOSS OF CONTROL DUE TO CONFIGURATION CHANGE

Validate and test all the parameters of the `SetSerialConf` function before putting your program into service.

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

Parameter Description

Input	Type	Comment
`Link`	`LinkNumber` *(see SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	`LinkNumber` is the communication port number.
`PointerToSerialConf`	POINTER TO `SERIAL_CONF` *(see page 253)*	`PointerToSerialConf` is the address of the configuration structure (variable of `SERIAL_CONF` type) in which the new configuration parameters are stored. The `ADR` standard function must be used to define the associated pointer. (See the example below.) If 0, set the application default configuration to the serial line.

Output	Type	Comment
SetSerialConf	WORD	This function returns: ● 0: The new configuration is set ● 255: The new configuration is refused because: ❍ the function is in progress ❍ the input parameters are not valid

Example

```
VAR
  MySerialConf: SERIAL_CONF
  result: WORD;
END_VAR

(*Get current configuration of serial line 1*)
GetSerialConf(1, ADR(MySerialConf));

(*Change to modbus RTU slave address 9*)
MySerialConf.Protocol := 0;         (*Modbus RTU/Machine Expert protocol
(in this case CodesysCompliant selects the protocol)*)
MySerialConf.CodesysCompliant := 0; (*Modbus RTU*)
MySerialConf.address := 9;          (*Set modbus address to 9*)

(*Reconfigure the serial line 1*)
result := SetSerialConf(1, ADR(MySerialConf));
```

SERIAL_CONF: Structure of the Serial Line Configuration Data Type

Structure Description

The SERIAL_CONF structure contains configuration information about the serial line port. It contains these variables:

Variable	Type	Description
Bauds	DWORD	baud rate
InterframeDelay	WORD	minimum time (in ms) between 2 frames in Modbus (RTU, ASCII)
FrameReceivedTimeout	WORD	In the ASCII protocol, FrameReceivedTimeout allows the system to conclude the end of a frame at reception after a silence of the specified number of ms. If 0 this parameter is not used.
FrameLengthReceived	WORD	In the ASCII protocol, FrameLengthReceived allows the system to conclude the end of a frame at reception, when the controller received the specified number of characters. If 0, this parameter is not used.
Protocol	BYTE	0: Modbus RTU or Machine Expert (see CodesysCompliant)
		1: Modbus ASCII
		2: ASCII
Address	BYTE	Modbus address 0 to 255 (0 for Master)
Parity	BYTE	0: none
		1: odd
		2: even
Rs485	BYTE	0: RS232
		1: RS485
ModPol (polarizartion resistor)	BYTE	0: no
		1: yes
DataFormat	BYTE	7 bits or 8 bits
StopBit	BYTE	1: 1 stop bit
		2: 2 stop bits
CharFrameStart	BYTE	In the ASCII protocol, 0 means there is no start character in the frame. Otherwise, the corresponding ASCII character is used to detect the beginning of a frame in receiving mode. In sending mode, this character is added at the beginning of the user frame.
CharFrameEnd1	BYTE	In the ASCII protocol, 0 means there is no second end character in the frame. Otherwise, the corresponding ASCII character is used to detect the end of a frame in receiving mode. In sending mode, this character is added at the end of the user frame.

Variable	Type	Description
CharFrameEnd2	BYTE	In the ASCII protocol, 0 means there is no second end character in the frame. Otherwise, the corresponding ASCII character is used (along with CharFrameEnd1) to detect the end of a frame in receiving mode. In sending mode, this character is added at the end of the user frame.
CodesysCompliant	BYTE	0: Modbus RTU
		1: Machine Expert (when Protocol = 0)
CodesysNetType	BYTE	not used

Appendix C
Controller Performance

Processing Performance

Introduction

This chapter provides information about the M251 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.

Glossary

A

analog output

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

application

A program including configuration data, symbols, and documentation.

application source

The collection of human-readable controller instructions, configuration data, HMI instructions, symbols, and other program documentation. The application source file is saved on the PC and you can download the application source file to most logic controllers. The application source file is used to build the executable program that runs in the logic controller.

ARP

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

B

BCD

(*binary coded decimal*) The format that represents decimal numbers between 0 and 9 with a set of 4 bits (a nybble/nibble, also titled as half byte). In this format, the 4 bits used to encode decimal numbers have an unused range of combinations.

For example, the number 2,450 is encoded as `0010 0100 0101 0000`.

BOOL

(*boolean*) A basic data type in computing. A `BOOL` variable can have one of these values: 0 (FALSE), 1 (TRUE). A bit that is extracted from a word is of type `BOOL`; for example, `%MW10.4` is a fifth bit of memory word number 10.

Boot application

(*boot application*) The binary file that contains the application. Usually, it is stored in the controller and allows the controller to boot on the application that the user has generated.

BOOTP

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

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

continuous function chart language

A graphical programming language (an extension of the IEC61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to inputs of other blocks to create complex expressions.

control network

A network containing logic controllers, SCADA systems, PCs, HMI, switches, ...

Two kinds of topologies are supported:

- flat: all modules and devices in this network belong to same subnet.
- 2 levels: the network is split into an operation network and an inter-controller network.

These two networks can be physically independent, but are generally linked by a routing device.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

CRC

(*cyclical redundancy check*) A method used to determine the validity of a communication transmission. The transmission contains a bit field that constitutes a checksum. The message is used to calculate the checksum by the transmitter according to the content of the message. Receiving nodes, then recalculate the field in the same manner. Any discrepancy in the value of the 2 CRC calculations indicates that the transmitted message and the received message are different.

D

data log

The controller logs events relative to the user application in a *data log*.

device network

A network that contains devices connected to a specific communication port of a logic controller. This controller is seen as a master from the devices point of view.

DHCP

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

DINT

(*double integer type*) Encoded in 32-bit format.

DNS

(*domain name system*) The naming system for computers and devices connected to a LAN or the Internet.

DTM

(*device type manager*) Classified into 2 categories:

- Device DTMs connect to the field device configuration components.
- CommDTMs connect to the software communication components.

The DTM provides a unified structure for accessing device parameters and configuring, operating, and diagnosing the devices. DTMs can range from a simple graphical user interface for setting device parameters to a highly sophisticated application capable of performing complex real-time calculations for diagnosis and maintenance purposes.

DWORD

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

E

EDS

(*electronic data sheet*) A file for fieldbus device description that contains, for example, the properties of a device such as parameters and settings.

equipment
A part of a machine including sub-assemblies such as conveyors, turntables, and so on.

Ethernet
A physical and data link layer technology for LANs, also known as IEEE 802.3.

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

F

FBD
(*function block diagram*) One of 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks, where each network contains a graphical structure of boxes and connection lines, which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

FE
(functional Earth) A common grounding connection to enhance or otherwise allow normal operation of electrically sensitive equipment (also referred to as functional ground in North America).

In contrast to a protective Earth (protective ground), a functional earth connection serves a purpose other than shock protection, and may normally carry current. Examples of devices that use functional earth connections include surge suppressors and electromagnetic interference filters, certain antennas, and measurement instruments.

firmware
Represents the BIOS, data parameters, and programming instructions that constitute the operating system on a controller. The firmware is stored in non-volatile memory within the controller.

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

freewheeling
When a logic controller is in freewheeling scan mode, a new task scan starts as soon as the previous scan has been completed. Contrast with *periodic scan mode*.

FTP
(*file transfer protocol*) A standard network protocol built on a client-server architecture to exchange and manipulate files over TCP/IP based networks regardless of their size.

H

HE10

Rectangular connector for electrical signals with frequencies below 3 MHz, complying with IEC 60807-2.

I

I/O

(*input/output*)

ICMP

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

IEC

(*international electrotechnical commission*) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

IEC 61131-3

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

IL

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

instruction list language

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

INT

(*integer*) A whole number encoded in 16 bits.

IP

(*Internet protocol*) Part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

K

KeepAlive

Messages sent by the OPC UA server to keep a subscription active. This is necessary when none of the monitored items of data have been updated since the previous publication.

L

ladder diagram language

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

LD

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

LINT

(*long integer*) A whole number encoded in a 64-bit format (4 times `INT` or 2 times `DINT`).

LRC

(*longitudinal redundancy checking*) An error-detection method for determining the correctness of transmitted and stored data.

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.

MIB

(*management information base*) An object database that is monitored by a network management system like SNMP. SNMP monitors devices are defined by their MIBs. Schneider Electric has obtained a private MIB, groupeschneider (3833).

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

ms
(*millisecond*)

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

N

network
A system of interconnected devices that share a common data path and protocol for communications.

NMT
(*network management*) CANopen protocols that provide services for network initialization, detected error control, and device status control.

node
An addressable device on a communication network.

notifications
In OPC UA, messages sent by the OPC UA server to inform clients that new items of data are available.

P

PDO
(*process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PE
(*Protective Earth*) A common grounding connection to help avoid the hazard of electric shock by keeping any exposed conductive surface of a device at earth potential. To avoid possible voltage drop, no current is allowed to flow in this conductor (also referred to as *protective ground* in North America or as an equipment grounding conductor in the US national electrical code).

post configuration
(*post configuration*) An option that allows to modify some parameters of the application without changing the application. Post configuration parameters are defined in a file that is stored in the controller. They are overloading the configuration parameters of the application.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

protocol

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

publishing interval

In OPC UA, the frequency at which the OPC_UA server sends notifications to clients informing them that data updates are available.

R

REAL

A data type that is defined as a floating-point number encoded in a 32-bit format.

RJ45

A standard type of 8-pin connector for network cables defined for Ethernet.

RPDO

(*receive process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

RPI

(*requested packet interval)* The time period between cyclic data exchanges requested by the scanner. EtherNet/IP devices publish data at the rate specified by the RPI assigned to them by the scanner, and they receive message requests from the scanner with a period equal to RPI.

RTC

(*real-time clock*) A battery-backed time-of-day and calender clock that operates continuously, even when the controller is not powered for the life of the battery.

S

sampling rate

In OPC UA, the frequency at which the OPC UA server reads items of data from connected devices.

scan

A function that includes:

- reading inputs and placing the values in memory
- executing the application program 1 instruction at a time and storing the results in memory
- using the results to update outputs

SDO

(*service data object*) A message used by the field bus master to access (read/write) the object directories of network nodes in CAN-based networks. SDO types include service SDOs (SSDOs) and client SDOs (CSDOs).

SFC

(*sequential function chart*) A language that is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

SINT

(*signed integer*) A 15-bit value plus sign.

SNMP

(*simple network management protocol*) A protocol that can control a network remotely by polling the devices for their status and viewing information related to data transmission. You can also use it to manage software and databases remotely. The protocol also permits active management tasks, such as modifying and applying a new configuration.

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

string

A variable that is a series of ASCII characters.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

TCP

(*transmission control protocol*) A connection-based transport layer protocol that provides a simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

terminal block

(*terminal block*) The component that mounts in an electronic module and provides electrical connections between the controller and the field devices.

TPDO

(*transmit process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

U

UDINT

(*unsigned double integer*) Encoded in 32 bits.

UDP

(*user datagram protocol*) A connectionless mode protocol (defined by IETF RFC 768) in which messages are delivered in a datagram (data telegram) to a destination computer on an IP network. The UDP protocol is typically bundled with the Internet protocol. UDP/IP messages do not expect a response, and are therefore ideal for applications in which dropped packets do not require retransmission (such as streaming video and networks that demand real-time performance).

UINT

(*unsigned integer*) Encoded in 16 bits.

V

variable

A memory unit that is addressed and modified by a program.

W

watchdog

A watchdog is a special timer used to ensure that programs do not overrun their allocated scan time. The watchdog timer is usually set to a higher value than the scan time and reset to 0 at the end of each scan cycle. If the watchdog timer reaches the preset value, for example, because the program is caught in an endless loop, an error is declared and the program stopped.

WORD

A type encoded in a 16-bit format.

Index

J

K

L

M

O

P

R

S

T

U

W

Modicon M251

Logic Controller System Functions and Variables PLCSystem Library Guide

EIO0000003095.01

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This document acquaints you with the system functions and variables offered within the Modicon M251 Logic Controller. The M251 PLCSystem library contains functions and variables to get information from and send commands to the controller system.

This document describes the data type functions and variables of the M251 PLCSystem library.

The following knowledge is required:

- Basic information on the functionality, structure, and configuration of the M251 Logic Controller
- Programming in the FBD, LD, ST, IL, or CFC language
- System variables (global variables)

Validity Note

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

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert Programming Guide	*EIO0000002854 (ENG)*; *EIO0000002855 (FRE)*; *EIO0000002856 (GER)*; *EIO0000002858 (SPA)*; *EIO0000002857 (ITA)*; *EIO0000002859 (CHS)*
Modicon M251 Logic Controller Hardware Guide	*EIO0000003101 (ENG)*; *EIO0000003102 (FRE)*; *EIO0000003103 (GER)*; *EIO0000003104 (SPA)*; *EIO0000003105 (ITA)*; *EIO0000003106 (CHS)*
Modicon M251 Logic Controller Programming Guide	*EIO0000003089 (ENG)*; *EIO0000003090 (FRE)*; *EIO0000003091 (GER)*; *EIO0000003092 (SPA)*; *EIO0000003093 (ITA)*; *EIO0000003094 (CHS)*

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

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Chapter 1
M251 System Variables

Overview

This chapter:

- gives an introduction to the system variables *(see page 12)*
- describes the system variables *(see page 18)* included with the M251 PLCSystem library

What Is in This Chapter?

This chapter contains the following sections:

Section 1.1
System Variables: Definition and Use

Overview

This section defines system variables and how to implement them in the Modicon M251 Logic Controller.

What Is in This Section?

This section contains the following topics:

Understanding System Variables

Introduction

This section describes how system variables are implemented. System variables:

- allow you to access general system information, perform system diagnostics, and command simple actions.
- are structured variables conforming to IEC 61131-3 definitions and naming conventions. You can access the system variables using the IEC symbolic name PLC_GVL. Some of the PLC_GVL variables are read-only (for example, PLC_R) and some are read/write (for example, PLC_W).
- are automatically declared as global variables. They have system-wide scope and can be accessed by any Program Organization Unit (POU) in any task.

Naming Convention

The system variables are identified by:

- a structure name that represents the category of system variable. For example, PLC_R represents a structure name of read-only variables used for the controller diagnostic.
- a set of component names that identifies the purpose of the variable. For example, i_wVendorID represents the controller vendor ID.

You can access the system variables by typing the structure name of the variables followed by the name of the component.

Here is an example of system variable implementation:

```
VAR
    myCtr_Serial : DWORD;
    myCtr_ID : DWORD;
    myCtr_FramesRx : UDINT;
END_VAR
```

```
myCtr_Serial := PLC_GVL.PLC_R.i_dwSerialNumber;
myCtr_ID := PLC_GVL.PLC.R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK
```

NOTE: The fully-qualified name of the system variable in the example above is PLC_GVL.PLC.R. The PLC_GVL is implicit when declaring a variable using the **Input Assistant**, but it may also be entered in full. Good programming practice often dictates the use of the fully-qualified variable name in declarations.

System Variables Location

2 kinds of system variables are defined for use when programming the controller:

- located variables
- unlocated variables

The located variables:

- have a fixed location in a static %MW area: %MW60000 to %MW60199 for read-only system variables.
- are accessible through Modbus TCP, Modbus serial, and EtherNet/IP requests both in RUNNING and STOPPED states.
- are used in EcoStruxure Machine Expert programs according to the `structure_name.component_name` convention explained previously. %MW addresses from 0 to 59999 can be accessed directly. Addresses greater than this are considered out of range by EcoStruxure Machine Expert and can only be accessed through the `structure_name.component_name` convention.

The unlocated variables:

- are not physically located in the %MW area.
- are not accessible through any fieldbus or network requests unless you locate them in the relocation table, and only then these variables can be accessed in RUNNING and STOPPED states. The relocation table uses the following dynamic %MW areas:
  - %MW60200 to %MW61999 for read-only variables
  - %MW62200 to %MW63999 for read/write variables
- are used in EcoStruxure Machine Expert programs according to the `structure_name.component_name` convention explained previously. %MW addresses from 0 to 59999 can be accessed directly. Addresses greater than this are considered out of range by EcoStruxure Machine Expert and can only be accessed through the `structure_name.component_name` convention.

Using System Variables

Introduction

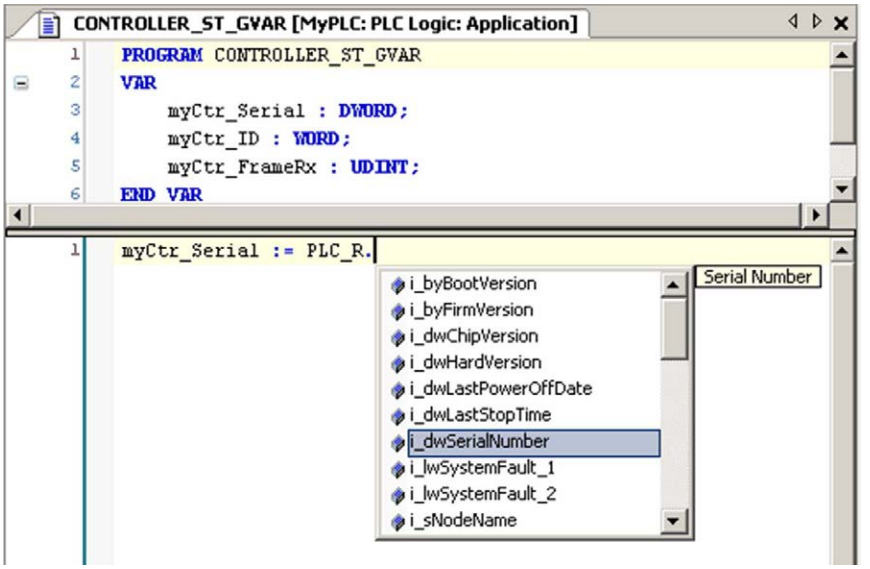
This section describes the steps required to program and to use system variables in EcoStruxure Machine Expert.

System variables are global in scope, and you can use them in all the Program Organization Units (POUs) of the application.

System variables do not need to be declared in the Global Variable List (GVL). They are automatically declared from the controller system library.

Using System Variables in a POU

EcoStruxure Machine Expert has an auto-completion feature. In a **POU**, start by entering the system variable structure name (`PLC_R`, `PLC_W`...) followed by a dot. The system variables appear in the **Input Assistant**. You can select the desired variable or enter the full name manually.



NOTE: In the example above, after you type the structure name `PLC_R.`, EcoStruxure Machine Expert offers a pop-up menu of possible component names/variables.

Example

The following example shows the use of some system variables:

```
VAR
    myCtr_Serial : DWORD;
    myCtr_ID : WORD;
    myCtr_FramesRx : UDINT;
END_VAR

myCtr_Serial := PLC_R.i_dwSerialNumber;
myCtr_ID := PLC_R.i_wVendorID;
myCtr_FramesRx := SERIAL_R[0].i_udiFramesReceivedOK;
```

Section 1.2
PLC_R and PLC_W Structures

Overview

This section lists and describes the different system variables included in the PLC_R and PLC_W structures.

What Is in This Section?

This section contains the following topics:

PLC_R: Controller Read-Only System Variables

Variable Structure

This table describes the parameters of the PLC_R system variable (PLC_R_STRUCT type):

Modbus Address (1)	Var Name	Type	Comment
60000	i_wVendorID	WORD	Controller Vendor ID. 101A hex = Schneider Electric
60001	i_wProductID	WORD	Controller Reference ID. **NOTE:** Vendor ID and Reference ID are the components of the Target ID of the controller displayed in the communication settings view (Target ID = 101A XXXX hex).
60002	i_dwSerialNumber	DWORD	Controller Serial Number
60004	i_byFirmVersion	ARRAY[0..3] OF BYTE	Controller Firmware Version [aa.bb.cc.dd]: • i_byFirmVersion[0] = aa • ... • i_byFirmVersion[3] = dd
60006	i_byBootVersion	ARRAY[0..3] OF BYTE	Controller Boot Version [aa.bb.cc.dd]: • i_byBootVersion[0] = aa • ... • i_byBootVersion[3] = dd
60008	i_dwHardVersion	DWORD	Controller Hardware Version.
60010	i_dwChipVersion	DWORD	Controller Coprocessor Version.
60012	i_wStatus	PLC_R_STATUS *(see page 70)*	State of the controller.
60013	i_wBootProjectStatus	PLC_R_BOOT_PROJECT_STATUS *(see page 67)*	Returns information about the boot application stored in FLASH memory.
60014	i_wLastStopCause	PLC_R_STOP_CAUSE *(see page 71)*	Cause of the last transition from RUN to another state.
60015	i_wLastApplicationError	PLC_R_APPLICATION_ERROR *(see page 65)*	Cause of the last controller exception.

Modbus Address (1)	Var Name	Type	Comment
60016	i_lwSystemFault_1	LWORD	Bit field FFFF FFFF FFFF FFFF hex indicates no error detected. A bit at low level means that an error has been detected: ● bit 0 = Reserved ● bit 1 = TM3 error detected ● bit 2 = Ethernet IF1 error detected ● bit 3 = Ethernet IF2 error detected ● bit 4 = Serial 1 in overcurrent error detected ● bit 5 = Reserved ● bit 6 = CAN 1 error detected ● bit 7 = Reserved ● bit 8 = Reserved ● bit 9 = TM4 error detected ● bit 10 = SD Card error detected ● bit 11 = Firewall error detected ● bit 12 = DHCP server error detected ● bit 13 = OPC UA server error detected
60024	i_wIOStatus1	PLC_R_IO_STATUS *(see page 68)*	Reserved
60025	i_wIOStatus2	PLC_R_IO_STATUS *(see page 68)*	TM3 I/O status.
60026	i_wClockBatterystatus	WORD	Status of the battery of the RTC: ● 0 = Battery change needed ● 100 = Battery fully charged Other values (1...99) represents the percentage of charge. For example, if the value is 75, it represents that the battery charge is 75%.
60028	i_dwAppliSignature1	DWORD	First DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.
60030	i_dwAppliSignature2	DWORD	Second DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.
60032	i_dwAppliSignature3	DWORD	Third DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.

Modbus Address (1)	Var Name	Type	Comment
60034	i_dwAppliSignature4	DWORD	Fourth DWORD of 4 DWORD signature (16 bytes total). The application signature is generated by the software during build.
(1) Not accessible through the application.			

n/a	i_sVendorName	STRING(31)	Name of the vendor: “Schneider Electric”.
n/a	i_sProductRef	STRING(31)	Reference of the controller.
n/a	i_sNodeName	STRING(99)	Node name on EcoStruxure Machine Expert Network.
n/a	i_dwLastStopTime	DWORD	The time of the last detected STOP in seconds beginning with January 1, 1970 at 00:00 UTC.
n/a	i_dwLastPowerOffDate	DWORD	The date and time of the last detected power off in seconds beginning with January 1, 1970 at 00:00 UTC. **NOTE:** Convert this value into date and time by using the function SysTimeRtcConvertUtcToDate. For more information about Time and Date conversion, refer to the Systime Library Guide *(see EcoStruxure Machine Expert, Getting & Setting Real Time Clock, SysTimeRtc and SysTimeCore Library Guide)*.
n/a	i_uiEventsCounter	UINT	Reserved
n/a	i_wTerminalPortStatus	PLC_R_ TERMINAL_ PORT_STATUS *(see page 73)*	Status of the USB Programming Port (USB Mini-B).
n/a	i_wSdCardStatus	PLC_R_SDCARD_ STATUS *(see page 69)*	Status of the SD card.
n/a	i_wUsrFreeFileHdl	WORD	Number of available File Handles. A File Handle is the resource allocated by the system when you open a file.
n/a	i_udiUsrFsTotalBytes	UDINT	User FileSystem total memory size (in bytes). It is the size of the flash memory for the directory "/usr/".
n/a	i_udiUsrFsFreeBytes	UDINT	User FileSystem free memory size (in bytes).

n/a	i_uiTM3BusState	PLC_R_TM3_BUS_STATE *(see page 74)*	TM3 bus state. i_uiTM3BusState can have the following values: ● 1: TM3_CONF_ERROR Configuration mismatch between physical configuration and EcoStruxure Machine Expert configuration. ● 3: TM3_OK Physical configuration matches EcoStruxure Machine Expert configuration. ● 4: TM3_POWER_SUPPLY_ERROR TM3 bus is not powered (for example when the Logic Controller is powered by USB).
n/a	i_ExpertIO_RunStop_Input	BYTE	Reserved
n/a	i_x10msClk	BOOL	TimeBase bit of 10 ms. This variable is toggling On/Off with period = 10 ms. The value toggles when the logic controller is in Stop and in Run state.
n/a	i_x100msClk	BOOL	TimeBase bit of 100 ms. This variable is toggling On/Off with period = 100 ms. The value toggles when the logic controller is in Stop and in Run state.
n/a	i_x1sClk	BOOL	TimeBase bit of 1 s. This variable is toggling On/Off with period = 1 s. The value toggles when the logic controller is in Stop and in Run state.

NOTE: n/a means that there is no pre-defined Modbus address mapping for this system variable.

PLC_W: Controller Read/Write System Variables

Variable Structure

This table describes the parameters of the PLC_W system variable (PLC_W_STRUCT type):

%MW	Var Name	Type	Comment
n/a	q_wResetCounterEvent	WORD	Transition from 0 to 1 resets the events counter (PLC_R.i_uiEventsCounter). To reset the counter again, it is necessary to write 0 to this variable before another transition from 0 to 1 can take place.
n/a	q_uiOpenPLCControl	UINT	When Value passes from 0 to 6699, the command previously written in the following PLC_W.q_wPLCControl is executed.
n/a	q_wPLCControl	PLC_W_COMMAND *(see page 75)*	Controller RUN / STOP command executed when the system variable PLC_W.q_uiOpenPLCControl value passes from 0 to 6699.

NOTE: n/a means that there is no pre-defined %MW mapping for this system variable.

Section 1.3
SERIAL_R and SERIAL_W Structures

Overview

This section lists and describes the different system variables included in the SERIAL_R and SERIAL_W structures.

What Is in This Section?

This section contains the following topics:

SERIAL_R[0...1]: Serial Line Read-Only System Variables

Introduction

SERIAL_R is an array of 2 SERIAL_R_STRUCT type. Each element of the array returns diagnostic system variables for the corresponding serial line.

For the M251 Logic Controller:

- Serial_R[0] refers to serial line
- Serial_R[1] is reserved

Variable Structure

This table describes the parameters of the SERIAL_R[0...1] system variables:

%MW	Var Name	Type	Comment
Serial Line			
n/a	i_udiFramesTransmittedOK	UDINT	Number of frames successfully transmitted.
n/a	i_udiFramesReceivedOK	UDINT	Number of frames received without any errors detected.
n/a	i_udiRX_MessagesError	UDINT	Number of frames received with errors detected (checksum, parity).
Modbus Specific			
n/a	i_uiSlaveExceptionCount	UINT	Number of Modbus exception responses returned by the logic controller.
n/a	i_udiSlaveMsgCount	UINT	Number of messages received from the Master and addressed to the logic controller.
n/a	i_uiSlaveNoRespCount	UINT	Number of Modbus broadcast requests received by the logic controller.
n/a	i_uiSlaveNakCount	UINT	Not used
n/a	i_uiSlaveBusyCount	UINT	Not used
n/a	i_uiCharOverrunCount	UINT	Number of character overruns.

n/a means that there is no predefined %MW mapping for this system variable.
Not used means that the variable is not maintained by the system, and that if the value of the variable is non-zero, it should be considered extraneous.

The SERIAL_R counters are reset on:

- Download.
- Controller reset.
- SERIAL_W[x].q_wResetCounter command.
- Reset command by Modbus request function code number 8.

SERIAL_W[0...1]: Serial Line Read/Write System Variables

Introduction

SERIAL_W is an array of 2 SERIAL_W_STRUCT type. Each element of the array resets the SERIAL_R system variables for the corresponding serial line to be reset.

For the M251 Logic Controller:

- Serial_W[0] refers to serial line
- Serial_W[1] is reserved

Variable Structure

This table describes the parameters of the SERIAL_W[0...1] system variable:

%MW	Var Name	Type	Comment
n/a	q_wResetCounter	WORD	Transition from 0 to 1 resets all SERIAL_R[0...1] counters. To reset the counters again, it is necessary to write 0 to this variable before another transition from 0 to 1 can take place.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Section 1.4
ETH_R and ETH_W Structures

Overview

This section lists and describes the different system variables included in the ETH_R and ETH_W structures.

What Is in This Section?

This section contains the following topics:

ETH_R: Ethernet Port Read-Only System Variables

Variable Structure

This table describes the parameters of the ETH_R system variable (ETH_R_STRUCT type):

%MW	Var Name	Type	Comment
60050	i_byIPAddress	ARRAY[0..3] OF BYTE	IP address of the Ethernet or Ethernet_1 interface [aaa.bbb.ccc.ddd]: • i_byIPAddress[0] = aaa • ... • i_byIPAddress[3] = ddd
60052	i_bySubNetMask	ARRAY[0..3] OF BYTE	Subnet Mask of the Ethernet or Ethernet_1 interface [aaa.bbb.ccc.ddd]: • i_bySub-netMask[0] = aaa • ... • i_bySub-netMask[3] = ddd
60054	i_byGateway	ARRAY[0..3] OF BYTE	Gateway address of the Ethernet or Ethernet_1 interface [aaa.bbb.ccc.ddd]: • i_byGateway[0] = aaa • ... • i_byGateway[3] = ddd
60056	i_byMACAddress	ARRAY[0..5] OF BYTE	MAC address of the Ethernet or Ethernet_1 interface [aa.bb.cc.dd.ee.ff]: • i_byMACAddress[0] = aa • ... • i_byMACAddress[5] = ff
60059	i_sDeviceName	STRING(15)	Name used to get IP address from server.
n/a	i_wIpMode	ETH_R_IP_MODE *(see page 82)*	Method used to obtain an IP address.
n/a means that there is no predefined %MW mapping for this system variable.			

%MW	Var Name	Type	Comment
n/a	i_byFDRServerIPAddress	ARRAY[0..3] OF BYTE	The IP address [aaa.bbb.ccc.ddd] of the DHCP or BootP server: • i_byFDRServerIPAddress[0] = aaa • ... • i_byFDRServerIPAddress[3] = ddd Equals 0.0.0.0 if Stored IP or Default IP used.
n/a	i_udiOpenTcpConnections	UDINT	Number of open TCP connections.
n/a	i_udiFramesTransmittedOK	UDINT	Number of frames successfully transmitted. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiFramedReceivedOK	UDINT	Number of frames successfully received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiTransmitBufferErrors	UDINT	Numbers of frames transmitted with detected errors. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiReceiveBufferErrors	UDINT	Numbers of frames received with detected errors. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_wFrameSendingProtocol	ETH_R_FRAME_PROTOCOL *(see page 81)*	Ethernet protocol configured for frames sending (IEEE 802.3 or Ethernet II).
n/a	i_wPortALinkStatus	ETH_R_PORT_LINK_STATUS *(see page 85)*	Link of the Ethernet Port (0 = No Link, 1 = Link connected to another Ethernet device).
n/a	i_wPortASpeed	ETH_R_PORT_SPEED *(see page 86)*	Ethernet Port network speed (10Mb/s, 100Mb/s).
n/a	i_wPortADuplexStatus	ETH_R_PORT_DUPLEX-_STATUS *(see page 83)*	Ethernet Port duplex status (0 = Half or 1 = Full duplex).
n/a means that there is no predefined %MW mapping for this system variable.			

%MW	Var Name	Type	Comment
n/a	i_udiPortACollisions	UDINT	Number of frames involved in one or more collisions and subsequently transmitted successfully. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_byIPAddress_If2	ARRAY[0..3] OF BYTE	IP address of the Ethernet or Ethernet_2 interface [aaa.bbb.ccc.ddd]: ● i_byIPAddress[0] = aaa ● ... ● i_byIPAddress[3] = ddd
n/a	i_bySubNetMask_If2	ARRAY[0..3] OF BYTE	Subnet Mask of the Ethernet or Ethernet_2 interface [aaa.bbb.ccc.ddd]: ● i_bySub-netMask[0] = aaa ● ... ● i_bySub-netMask[3] = ddd
n/a	i_byGateway_If2	ARRAY[0..3] OF BYTE	Gateway address of the Ethernet or Ethernet_2 interface [aaa.bbb.ccc.ddd]: ● i_byGateway[0] = aaa ● ... ● i_byGateway[3] = ddd
n/a	i_byMACAddress_If2	ARRAY[0..5] OF BYTE	MAC address of the Ethernet or Ethernet_2 interface [aa.bb.cc.dd.ee.ff]: ● i_byMACAddress[0] = aa ● ... ● i_byMACAddress[5] = ff
n/a	i_sDeviceName_If2	STRING(15)	Name used to get IP address from server.
n/a	i_wIpMode_If2	ETH_R_IP_MODE *(see page 82)*	Method used to obtain an IP address.
n/a	i_wPortALinkStatus_If2	ETH_R_PORT_LINK_STATUS *(see page 85)*	Link of the Ethernet Port (0 = No Link, 1 = Link connected to another Ethernet device).
n/a	i_wPortASpeed_If2	ETH_R_PORT_SPEED *(see page 86)*	Ethernet Port network speed (10Mb/s or 100Mb/s).
n/a means that there is no predefined %MW mapping for this system variable.			

%MW	Var Name	Type	Comment
n/a	i_wPortADuplexStatus_If2	ETH_R_PORT_DUPLEX-_STATUS *(see page 83)*	Ethernet Port duplex status: • 0: Half • 1: Full duplex
n/a	i_wPortAIpStatus_If2	ETH_R_PORT_IP_STATUS *(see page 84)*	Ethernet TCP/IP port stack status.
Modbus TCP/IP Specific			
n/a	i_udiModbusMessageTransmitted	UDINT	Number of Modbus messages transmitted. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiModbusMessageReceived	UDINT	Number of Modbus messages received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiModbusErrorMessage	UDINT	Modbus detected error messages transmitted and received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a means that there is no predefined %MW mapping for this system variable.			

%MW	Var Name	Type	Comment
EtherNet/IP Specific			
n/a	i_udiETHIP_IOMessagingTransmitted	UDINT	EtherNet/IP Class 1 frames transmitted. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiETHIP_IOMessagingReceived	UDINT	EtherNet/IP Class 1 frames received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a means that there is no predefined %MW mapping for this system variable. **Not used** means that the variable is not maintained by the system, and that if the value of the variable is non-zero, it should be considered extraneous.			

%MW	Var Name	Type	Comment
n/a	i_udiUCMM_Request	UDINT	EtherNet/IP Unconnected Messages received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiUCMM_Error	UDINT	EtherNet/IP invalid Unconnected Messages received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiClass3_Request	UDINT	EtherNet/IP Class 3 requests received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_udiClass3_Error	UDINT	EtherNet/IP invalid class 3 requests received. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_uiAssemblyInstanceInput	UINT	Input Assembly Instance number. See the appropriate Programming Guide of your controller for more information.
n/a	i_uiAssemblyInstanceInputSize	UINT	Input Assembly Instance size. See the appropriate Programming Guide of your controller for more information.
n/a	i_uiAssemblyInstanceOutput	UINT	Output Assembly Instance number. See the appropriate Programming Guide of your controller for more information.
n/a	i_uiAssemblyInstanceOutputSize	UINT	Output Assembly Instance size. See the appropriate Programming Guide of your controller for more information.

n/a means that there is no predefined %MW mapping for this system variable.
Not used means that the variable is not maintained by the system, and that if the value of the variable is non-zero, it should be considered extraneous.

%MW	Var Name	Type	Comment
n/a	i_uiETHIP_ConnectionTimeouts	UINT	Number of connection timeouts. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_ucEipRunIdle	ETH_R_RUN_IDLE *(see page 87)*	Run (value = 1)/Idle(value = 0) flag for EtherNet/IP class 1 connection.
n/a	i_byMasterIpTimeouts	BYTE	Ethernet Modbus TCP Master timeout events counter. Reset at Power ON or with reset command ETH_W.q_wResetCounter.
n/a	i_byMasterIpLost	BYTE	Ethernet Modbus TCP Master link status: 0 = link OK, 1 = link lost.
n/a	i_wPortAIpStatus	ETH_R_PORT_IP_STATUS *(see page 84)*	Ethernet TCP/IP port stack status.
n/a means that there is no predefined %MW mapping for this system variable. **Not used** means that the variable is not maintained by the system, and that if the value of the variable is non-zero, it should be considered extraneous.			

NOTE: **n/a** means that there is no predefined %MW mapping for this system variable.

ETH_W: Ethernet Port Read/Write System Variables

Variable Structure

This table describes the parameters of the ETH_W system variable (ETH_W_STRUCT type):

%MW	Var Name	Type	Comment
n/a	q_wResetCounter	WORD	Transition from 0 to 1 resets all ETH_R counters. To reset again, it is necessary to write 0 to this variable before another transition from 0 to 1 can take place.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Section 1.5
TM3_MODULE_R Structure

TM3_MODULE_R[0...13]: TM3 Modules Read-Only System Variables

Introduction

The TM3_MODULE_R is an array of 14 TM3_MODULE_R_STRUCT type. Each element of the array returns diagnostic system variables for the corresponding TM3 expansion module.

For the Modicon M251 Logic Controller:

- TM3_MODULE_R[0] refers to the TM3 expansion module 0
- ...
- TM3_MODULE_R[13] refers to the TM3 expansion module 13

Variable Structure

The following table describes the parameters of the TM3_MODULE_R[0...13] system variable:

%MW	Var Name	Type	Comment
n/a	i_wProductID	WORD	TM3 expansion module ID.
n/a	i_wModuleState	TM3_MODULE_STATE *(see page 91)*	Describes the state of the TM3 module.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Section 1.6
TM3_BUS_W Structure

TM3_BUS_W: TM3 Bus System Variables

Variable Structure

This table describes the parameters of the TM3_BUS_W system variable (TM3_BUS_W_STRUCT type):

Var Name	Type	Comment
q_wIOBusErrPassiv	TM3_BUS_W_IOBUSERRMOD	When set to ERR_ACTIVE (the default), bus errors detected on TM3 expansion modules stop all I/O exchanges. When set to ERR_PASSIVE, passive I/O error handing is used: the controller attempts to continue data bus exchanges.
q_wIOBusRestart	TM3_BUS_W_IOBUSINIT	When set to 1, restarts the I/O expansion bus. This is only necessary when q_wIOBusErrPassiv is set to ERR_ACTIVE and at least one bit of TM3_MODULE_R[i].i_wModuleState is set to TM3_BUS_ERROR.

For more information, refer to I/O Configuration General Description *(see Modicon M251 Logic Controller, Programming Guide)*.

Section 1.7
PROFIBUS_R Structure

PROFIBUS_R: PROFIBUS Read-Only System Variables

Variable Structure

This table describes the parameters of the PROFIBUS_R system variable (PROFIBUS_R_STRUCT type):

%MW	Var Name	Type	Comment
n/a	i_wPNOIdentifier	WORD	Slave identification code.
n/a	i_wBusAdr	UINT	PROFIBUS slave address.
n/a	i_CommState	UDINT	Value representing the state of the PROFIBUS module: • 0x00: Unknown • 0x01: Not configured • 0x02: Stop • 0x03: Idle • 0x04: Operate
n/a	i_CommError	UDINT	Communication error code.
n/a	i_ErrorCount	UDINT	Communication error counter.

NOTE: n/a means that there is no predefined %MW mapping for this system variable.

Chapter 2
M251 System Functions

Overview

This chapter describes the system functions included in the M251 PLCSystem library.

What Is in This Chapter?

This chapter contains the following sections:

Section 2.1
M251 Read Functions

Overview

This section describes the read functions included in the M251 PLCSystem library.

What Is in This Section?

This section contains the following topics:

GetRtc: Get Real Time Clock

Function Description

This function returns RTC time in seconds in UNIX format (time expired in seconds since January 1, 1970 at 00:00 UTC).

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

The following table describes the I/O variable:

Output	Type	Comment
GetRtc	DINT	RTC in seconds in UNIX format.

Example

The following example describes how to get the RTC value:

```
VAR
    MyRTC : DINT := 0;
END_VAR

MyRTC := GetRtc();
```

IsFirstMastColdCycle: Indicate if this Cycle is the First MAST Cold Start Cycle

Function Description

This function returns TRUE during the first MAST cycle after a cold start (first cycle after download or reset cold).

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

The table describes the output variable:

Output	Type	Comment
`IsFirstMastColdCycle`	`BOOL`	TRUE during the first MAST task cycle after a cold start.

Example

Refer to the function `IsFirstMastCycle` *(see page 41)*.

IsFirstMastCycle: Indicate if this Cycle is the First MAST Cycle

Function Description

This function returns TRUE during the first MAST cycle after a start.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

Output	Type	Comment
IsFirstMastCycle	BOOL	TRUE during the first MAST task cycle after a start.

Example

This example describes the three functions `IsFirstMastCycle`, `IsFirstMastColdCycle` and `IsFirstMastWarmCycle` used together.

Use this example in MAST task. Otherwise, it may run several times or possibly never (an additional task might be called several times or not called during 1 MAST task cycle):

```
VAR
MyIsFirstMastCycle : BOOL;
MyIsFirstMastWarmCycle : BOOL;
MyIsFirstMastColdCycle : BOOL;
END_VAR

MyIsFirstMastWarmCycle := IsFirstMastWarmCycle();
MyIsFirstMastColdCycle := IsFirstMastColdCycle();
MyIsFirstMastCycle := IsFirstMastCycle();

IF (MyIsFirstMastWarmCycle) THEN

(*This is the first Mast Cycle after a Warm Start: all variables are set
to their initialization values except the Retain variables*)

(*=> initialize the needed variables so that your application runs as
expected in this case*)

END_IF;

IF (MyIsFirstMastColdCycle) THEN

(*This is the first Mast Cycle after a Cold Start: all variables are set
to their initialization values including the Retain Variables*)

(*=> initialize the needed variables so that your application runs as
expected in this case*)

END_IF;

IF (MyIsFirstMastCycle) THEN

(*This is the first Mast Cycle after a Start, i.e. after a Warm or Cold
Start as well as STOP/RUN commands*)

(*=> initialize the needed variables so that your application runs as
expected in this case*)

END_IF;
```

IsFirstMastWarmCycle: Indicate if this Cycle is the First MAST Warm Start Cycle

Function Description

This function returns TRUE during the first MAST cycle after a warm start.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

This table describes the output variable:

Output	Type	Comment
IsFirstMastWarmCycle	BOOL	TRUE during the first MAST task cycle after a warm start.

Example

Refer to the function IsFirstMastCycle *(see page 41)*.

Section 2.2
M251 Write Functions

`SetRTCDrift`: Set Compensation Value to the RTC

Function Description

This function accelerates or slows down the frequency of the RTC to give control to the application for RTC compensation, depending on the operating environment (temperature, …). The compensation value is given in seconds per week. It can be positive (accelerate) or negative (slow down).

NOTE: The `SetRTCDrift` function must be called only once. Each new call replaces the compensation value by the new one. The value is kept in the controller hardware while the RTC is powered by the main supply or by the battery. If both battery and power supply are removed, the RTC compensation value is not available.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variables Description

This table describes the input parameters:

Inputs	Type	Comment
RtcDrift	SINT(-36...+73)	Correction in seconds per week (-36 ... +73).

NOTE: The parameters Day, Hour, and Minute are used only to ensure backwards compatibility.

NOTE: If the value entered for RtcDrift exceeds the limit value, the controller firmware sets the value to its maximum value.

This table describes the output variable:

Output	Type	Comment
SetRTCDrift	RTCSETDRIFT_ERROR *(see page 93)*	Returns RTC_OK (00 hex) if command is correct otherwise returns the ID code of the detected error.

Example

In this example, the function is called only once during the first MAST task cycle. It accelerates the RTC by 4 seconds a week (18 seconds a month).

```
VAR
    MyRTCDrift : SINT (-36...+73) := 0;
    MyDay : DAY_OF_WEEK;
    MyHour : HOUR;
    MyMinute : MINUTE;
END_VAR
```

```
IF IsFirstMastCycle() THEN
    MyRTCDrift := 4;
    MyDay := 0;
    MyHour := 0;
    MyMinute := 0;
    SetRTCDrift(MyRTCDrift, MyDay, MyHour, MyMinute);
END_IF
```

Section 2.3
M251 User Functions

Overview

This section describes the `FB_Control_Clone`, `DataFileCopy` and `ExecuteScript` functions included in the M251 PLCSystem library.

What Is in This Section?

This section contains the following topics:

FB_ControlClone: Clone the Controller

Function Block Description

Cloning is possible by SD card or **Controller Assistant**. When user rights are enabled, the cloning function is not allowed, and the function block enables cloning functionality one time on the next controller power on.

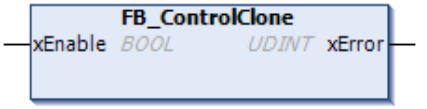
NOTE: You can choose whether user rights are included in the clone on the **Clone Management** page of the Web server *(see Modicon M251 Logic Controller, Programming Guide)*.

This table shows how to set the function block and the user rights:

Function block setting	When user rights enabled	When user rights disabled
xEnable = 1	Cloning is allowed	Cloning is still allowed
xEnable = 0	Cloning is not allowed	Cloning is not allowed

Read from controller with **Controller Assistant** is also affected by FB_ControlClone.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation (see page 97)*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
xEnable	BOOL	If TRUE, enables the cloning functionality one time. If FALSE, disables the cloning functionality.

The following table describes the output variables:

Output	Type	Comment
xError	BOOL	TRUE indicates that an error is detected and the function block aborted the action.

DataFileCopy: Copy File Commands

Function Block Description

This function block copies memory data to a file and vice versa. The file is located either within the internal file system or an external file system (SD card).

The DataFileCopy function block can:

- Read data from a formatted file or
- Copy data from memory to a formatted file. For further information, refer to Flash Memory Organization *(see Modicon M251 Logic Controller, Programming Guide)*.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation (see page 97)*.

I/O Variable Description

This table describes the input variables:

Input	Type	Comment
xExecute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when any ongoing execution terminates. **NOTE:** With the falling edge, the function continues until it concludes its execution and updates its outputs. The outputs are retained for one cycle and reset.
sFileName	STRING	File name without extension (the extension *.DTA* is automatically added). Use only a...z, A...Z, 0...9 alphanumeric characters.
xRead	BOOL	TRUE: copy data from the file identified by sFileName to the internal memory of the controller. FALSE: copy data from the internal memory of the controller to the file identified by sFileName.

Input	Type	Comment
xSecure	BOOL	TRUE: The MAC address is always stored in the file. Only a controller with the same MAC address can read from the file. FALSE: Another controller with the same type of memory can read from the file.
iLocation	INT	0: the file location is */usr/DTA* in internal file system. 1: the file location is */usr/DTA* in external file system (SD card). **NOTE:** If the file does not already exist in the directory, the file is created.

uiSize	UINT	Indicates the size in bytes. Maximum is 65534 bytes.
		Only use addresses of variables conforming to IEC 61131-3 (variables, arrays, structures), for example:
		`Variable : int;`
		`uiSize := SIZEOF (Variable);`

dwAdd	DWORD	Indicates the address in the memory that the function will read from or write to. Only use addresses of variables conforming to IEC 61131-3 (variables, arrays, structures), for example:
		`Variable : int;`
		`dwAdd := ADR (Variable);`

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Verify that the memory location is of the correct size and the file is of the correct type before copying the file to memory.

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

This table describes the output variables:

Output	Type	Comment
xDone	BOOL	TRUE = indicates that the action is successfully completed.
xBusy	BOOL	TRUE = indicates that the function block is running.
xError	BOOL	TRUE = indicates that an error is detected and the function block aborted the action.
eError	DataFileCopyError *(see page 77)*	Indicates the type of the data file copy detected error.

NOTE: If you write to memory variable within the area of the file write, a CRC integrity error results.

Example

This example describes how to copy file commands:

```
VAR
LocalArray : ARRAY [0..29] OF BYTE;
myFileName: STRING := 'exportfile';
EXEC_FLAG: BOOL;
DataFileCopy: DataFileCopy;
END_VAR
DataFileCopy(
xExecute:= EXEC_FLAG,
sFileName:= myFileName,
xRead:= FALSE,
xSecure:= FALSE,
iLocation:= DFCL_INTERNAL,
uiSize:= SIZEOF(LocalArray),
dwAdd:= ADR(LocalArray),
xDone=> ,
xBusy=> ,
xError=> ,
eError=> );
```

ExecuteScript: Run Script Commands

Function Block Description

This function block can run the following SD card script commands:

- `Download`
- `Upload`
- `SetNodeName`
- `Delete`
- `Reboot`
- `ChangeModbusPort`

For information on the required script file format, refer to Script Files for SD Cards.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

This table describes the input variables:

Input	Type	Comment
`xExecute`	`BOOL`	On detection of a rising edge, starts the function block execution. On detection of a falling edge, resets the outputs of the function block when any on-going execution terminates. **NOTE:** With the falling edge, the function continues until it concludes its execution and updates its outputs. The outputs are retained for one cycle and reset.
`sCmd`	`STRING`	SD card script command syntax. Simultaneous command executions are not allowed: if a command is being executed from another function block or from an SD card script then the function block queues the command and does not execute it immediately. **NOTE:** An SD card script executed from an SD card is considered as being executed until the SD card has been removed.

This table describes the output variables:

Output	Type	Comment
xDone	BOOL	TRUE indicates that the action is successfully completed.
xBusy	BOOL	TRUE indicates that the function block is running.
xError	BOOL	TRUE indicates error detection; the function block aborts the action.
eError	ExecuteScriptError *(see page 79)*	Indicates the type of the execute script detected error.

Example

This example describes how to execute an Upload script command:

```
VAR
EXEC_FLAG: BOOL;
ExecuteScript: ExecuteScript;
END_VAR
ExecuteScript(
xExecute:= EXEC_FLAG,
sCmd:= 'Upload "/usr/Syslog/*"',
xDone=> ,
xBusy=> ,
xError=> ,
eError=> );
```

Section 2.4
M251 Disk Space Functions

Overview

This section describes the disk space functions included in the SystemInterface library.

What Is in This Section?

This section contains the following topics:

FC_GetFreeDiskSpace: Gets the Free Memory Space

Function Description

This function retrieves the amount of free memory space of a memory medium (flash disk, RAM disk, SD card) in bytes. The name of the memory medium is transferred:

- Flash disk = "ide0:"
- RAM disk = "ram0:"
- SD card = "sd0:"

The free memory space of a remote device cannot be accessed. If a remote device is specified as parameter, then the function returns "-1".

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

This table describes the input variables:

Input	Data type	Description
`i_sVolumeName`	STRING[80]	Name of the device whose free memory space must be accessed
`iq_uliFreeDiskSpace`	ULINT	Free memory space in bytes

This table describes the output variables:

Output	Data type	Description
`FC_GetFreeDiskSpace`	DINT	0: The amount of free memory space was retrieved successfully -1: Error when attempting to access the amount of free memory. For example, an invalid device or remote device was selected -318: Invalid parameter (`i_sVolumeName`)

FC_GetLabel: Gets the Label of Memory

Function Description

This function retrieves the label of a memory medium. If a device has no label, then an empty string is returned.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

This table describes the input variables:

Input	Data type	Description
i_sVolumeName	STRING[80]	Name of the device whose label must be accessed
iq_sLabel	STRING[11]	Label of the device

This table describes the output variables:

Output	Data type	Description
FC_GetLabel	DINT	0: The label was retrieved successfully -1: Error when accessing the label -318: Invalid parameter

FC_GetTotalDiskSpace: Gets the Size of Memory

Function Description

This function retrieves the size of a memory medium (flash disk, RAM disk, SD card) in bytes.

The name of the memory medium is transferred:

- Flash disk = "ide0:"
- RAM disk = "ram0:"
- SD card = "sd0:"

The size of a remote device cannot be accessed. If a remote device is specified as parameter, then the function returns "-1".

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

This table describes the input variables:

Input	Data type	Description
`i_sVolumeName`	STRING[80]	Name of the device whose memory size must be accessed
`iq_uliTotalDiskSpace`	ULINT	Size of the memory medium in byte

This table describes the output variables:

Output	Data type	Description
`FC_GetTotalDiskSpace`	DINT	0: Size was retrieved successfully -1: Error when reading the size -318: At least one of the parameters is invalid

Section 2.5
TM3 Read Functions

Overview

This section describes the TM3 read functions included in the M251 PLCSystem library.

What Is in This Section?

This section contains the following topics:

TM3_GetModuleBusStatus: Get TM3 Module Bus Status

Function Description

This function returns the bus status of the module. The index of the module is given as an input parameter.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
ModuleIndex	BYTE	Index of the module (0 for the first expansion, 1 for the second, and so on).

The following table describes the output variable:

Output	Type	Comment
TM3_GetModuleBusStatus	TM3_ERR_CODE *(see page 89)*	Returns TM3_OK (00 hex) if command is correct otherwise returns the ID code of the detected error.

TM3_GetModuleFWVersion: Get TM3 Module Firmware Version

Function Description

This function returns the firmware version of a specified TM3 module.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
ModuleIndex	BYTE	Index of the module (0 for the first expansion, 1 for the second, and so on).

The following table describes the output variable:

Output	Type	Comment
TM3_GetModuleFWVersion	UINT	Returns the firmware version of the module, or FFFF hex if the information cannot be read. For example, 001A hex indicates firmware version 26.

TM3_GetModuleInternalStatus: Get TM3 Module Internal Status

Function Description

This function fills the pStatusBuffer with the status table of the module ModuleIndex.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to the chapter *Function and Function Block Representation* *(see page 97)*.

I/O Variable Description

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Ensure that the pStatusBuffer is allocated.

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

The following table describes the input variables:

Input	Type	Comment
ModuleIndex	BYTE	Index of the module (0 for the first expansion, 1 for the second, and so on).
StatusOffset	BYTE	Offset of the first status to be read in the status table.
StatusSize	BYTE	Number of bytes to be read in the status table.
pStatusBuffer	POINTER TO BYTE	Buffer containing the read status table.

The following table describes the output variable:

Output	Type	Comment
TM3_GetModuleInternalStatus	TM3_ERR_CODE *(see page 89)*	Returns TM3_OK (00 hex) if command is correct otherwise returns the ID code of the error.

Example

The following example describes how to get the module internal status:

```
VAR
AMM3HT_Channel1_Input_Status: BYTE;
END_VAR
TM3_GetModuleInternalStatus(0, 1, 1, ADR(AMM3HT_Channel1_Input_Status));
```

Chapter 3
M251 PLCSystem Library Data Types

Overview

This chapter describes the data types of the M251 PLCSystem Library.

There are 2 kinds of data types available:

- System variable data types are used by the system variables *(see page 11)* of the M251 PLCSystem Library (PLC_R, PLC_W,...).
- System function data types are used by the read/write system functions *(see page 37)* of the M251 PLCSystem Library.

What Is in This Chapter?

This chapter contains the following sections:

Section 3.1
PLC_RW System Variables Data Types

Overview

This section lists and describes the system variable data types included in the PLC_R and PLC_W structures.

What Is in This Section?

This section contains the following topics:

PLC_R_APPLICATION_ERROR: Detected Application Error Status Codes

Enumerated Type Description

The PLC_R_APPLICATION_ERROR enumeration data type contains the following values:

Enumerator	Value	Comment	What to do
PLC_R_APP_ERR_UNKNOWN	FFFF hex	Undefined error detected.	Contact your Schneider Electric service representative.
PLC_R_APP_ERR_NOEXCEPTION	0000 hex	No error detected.	–
PLC_R_APP_ERR_WATCHDOG	0010 hex	Task watchdog expired.	Check your application. A reset is needed to enter Run mode.
PLC_R_APP_ERR_ HARDWAREWATCHDOG	0011 hex	System watchdog expired.	If the problem is reproducible, verify that there are no configured but disconnected communication ports. If the problem persists, update the firmware. If the problem still persists, contact your Schneider Electric service representative.
PLC_R_APP_ERR_IO_CONFIG_ ERROR	0012 hex	Incorrect I/O configuration parameters detected.	Your application might be corrupted. To resolve this issue, use one of the methods: 1. **Build → Clean All** 2. Export/Import your application. 3. Upgrade EcoStruxure Machine Expert to the latest version.
PLC_R_APP_ERR_UNRESOLVED_ EXTREFS	0018 hex	Undefined functions detected.	Delete the unresolved functions from the application.

Enumerator	Value	Comment	What to do
PLC_R_APP_ERR_IEC_TASK_ CONFIG_ERROR	0025 hex	Incorrect Task configuration parameters detected.	Your application might be corrupted. To resolve this issue, use one of the methods: 1. **Build → Clean All** 2. Export/Import your application. 3. Upgrade EcoStruxure Machine Expert to the latest version.
PLC_R_APP_ERR_ILLEGAL_ INSTRUCTION	0050 hex	Undefined instruction detected.	Debug your application to resolve the problem.
PLC_R_APP_ERR_ACCESS_ VIOLATION	0051 hex	Attempted access to reserved memory area.	Debug your application to resolve the problem.
PLC_R_APP_ERR_DIVIDE_BY_ ZERO	0102 hex	Integer division by zero detected.	Debug your application to resolve the problem.
PLC_R_APP_ERR_PROCESSORLOAD_ WATCHDOG	0105 hex	Processor overloaded by Application Tasks.	Reduce the application workload by improving the application architecture. Increase the task cycle duration. Reduce event frequency.
PLC_R_APP_ERR_DIVIDE_REAL_ BY_ZERO	0152 hex	Real division by zero detected.	Debug your application to resolve the problem.
PLC_R_APP_ERR_EXPIO_EVENTS_ COUNT_EXCEEDED	4E20 hex	Too many events on expert I/Os are detected.	Reduce the number of event tasks.
PLC_R_APP_ERR_APPLICATION_ VERSION_MISMATCH	4E21 hex	Mismatch in the application version detected.	The application version in the logic controller does not match the version in EcoStruxure Machine Expert. Refer to Applications *(see EcoStruxure Machine Expert, Programming Guide)*.

PLC_R_BOOT_PROJECT_STATUS: Boot Project Status Codes

Enumerated Type Description

The PLC_R_BOOT_PROJECT_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_R_NO_BOOT_PROJECT	0000 hex	Boot project does not exist in Flash memory.
PLC_R_BOOT_PROJECT_CREATION_IN_PROGRESS	0001 hex	Boot project is being created.
PLC_R_DIFFERENT_BOOT_PROJECT	0002 hex	Boot project in Flash is different from the project loaded in RAM.
PLC_R_VALID_BOOT_PROJECT	FFFF hex	Boot project in Flash is the same as the project loaded in RAM.

PLC_R_IO_STATUS: I/O Status Codes

Enumerated Type Description

The PLC_R_IO_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_R_IO_OK	FFFF hex	Inputs/Outputs are operational.
PLC_R_IO_NO_INIT	0001 hex	Inputs/Outputs are not initialized.
PLC_R_IO_CONF_FAULT	0002 hex	Incorrect I/O configuration parameters detected.
PLC_R_IO_SHORTCUT_FAULT	0003 hex	Inputs/Outputs short-circuit detected.
PLC_R_IO_POWER_SUPPLY_FAULT	0004 hex	Inputs/Outputs power supply error detected.

PLC_R_SDCARD_STATUS: SD Card Slot Status Codes

Enumerated Type Description

The PLC_R_SDCARD_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
NO_SDCARD	0000 hex	No SD card detected in the slot or the slot is not connected.
SDCARD_READONLY	0001 hex	SD card is in read-only mode.
SDCARD_READWRITE	0002 hex	SD card is in read/write mode.
SDCARD_ERROR	0003 hex	Error detected in the SD card. More details on the error that occurred are written to the file FwLog.txt.

PLC_R_STATUS: Controller Status Codes

Enumerated Type Description

The PLC_R_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_R_EMPTY	0000 hex	Controller does not contain an application.
PLC_R_STOPPED	0001 hex	Controller is stopped.
PLC_R_RUNNING	0002 hex	Controller is running.
PLC_R_HALT	0004 hex	Controller is in a HALT state (see the controller state diagram in your controller programming guide *(see Modicon M251 Logic Controller, Programming Guide)*).
PLC_R_BREAKPOINT	0008 hex	Controller has paused at a breakpoint.

PLC_R_STOP_CAUSE: From RUN State to Other State Transition Cause Codes

Enumerated Type Description

The PLC_R_STOP_CAUSE enumeration data type contains the following values:

Enumerator	Value	Comment	What to do
PLC_R_STOP_REASON_UNKNOWN	00 hex	Initial value or stop cause is indeterminable.	Contact your local Schneider Electric representative.
PLC_R_STOP_REASON_HW_WATCHDOG	01 hex	Stopped after hardware watchdog timeout.	Contact your local Schneider Electric representative.
PLC_R_STOP_REASON_RESET	02 hex	Stopped after reset.	See reset possibilities in Controller State Diagram.
PLC_R_STOP_REASON_EXCEPTION	03 hex	Stopped after exception.	Verify your application, and correct if necessary. See System and Task Watchdogs. A reset is needed to enter Run mode.
PLC_R_STOP_REASON_USER	04 hex	Stopped after a user request.	Refer to Stop Command in Commanding State Transitions *(see Modicon M251 Logic Controller, Programming Guide)*.
PLC_R_STOP_REASON_IECPROGRAM	05 hex	Stopped after a program command request (for example: control command with parameter PLC_W.q_wPLCControl:=PLC_W_COMMAND.PLC_W_STOP;).	–
PLC_R_STOP_REASON_DELETE	06 hex	Stopped after a remove application command.	See the **Applications** tab of the Controller Device Editor *(see Modicon M251 Logic Controller, Programming Guide)*.
PLC_R_STOP_REASON_DEBUGGING	07 hex	Stopped after entering debug mode.	–
PLC_R_STOP_FROM_NETWORK_REQUEST	0A hex	Stopped after a request from the network, the controller Web server, or PLC_W command.	–
PLC_R_STOP_FROM_INPUT	0B hex	Stop required by a controller input.	–
PLC_R_STOP_FROM_RUN_STOP_SWITCH	0C hex	Stop required by the controller switch.	–

Enumerator	Value	Comment	What to do
PLC_R_STOP_REASON_ RETAIN_MISMATCH	0D hex	Stopped after an unsuccessful check context test during rebooting.	There are retained variables in non-volatile memory that do not exist in the executing application. Verify your application, correct if necessary, then reestablish the boot application.
PLC_R_STOP_REASON_ BOOT_APPLI_MISMATCH	0E hex	Stopped after an unsuccessful compare between the boot application and the application that was in the memory before rebooting.	Create a valid boot application.
PLC_R_STOP_REASON_ POWERFAIL	0F hex	Stopped after a power interruption.	–

For more information on the reasons why the controller has stopped, refer to the Controller State Description *(see Modicon M251 Logic Controller, Programming Guide)*.

PLC_R_TERMINAL_PORT_STATUS: Programming Port Connection Status Codes

Enumerated Type Description

The PLC_R_TERMINAL_PORT_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
TERMINAL_NOT_CONNECTED	00 hex	No PC is connected to the programming port.
TERMINAL_CONNECTION_IN_PROGRESS	01 hex	Connection is in progress.
TERMINAL_CONNECTED	02 hex	PC is connected to the programming port.
TERMINAL_ERROR	0F hex	Error detected during connection.

PLC_R_TM3_BUS_STATE: TM3 Bus Status Codes

Enumerated Type Description

The PLC_R_TM3_BUS_STATE enumeration data type contains the following values:

Enumerator	Value	Comment
TM3_CONF_ERROR	01 hex	Error detected due to mismatch in the physical configuration and the configuration in EcoStruxure Machine Expert.
TM3_OK	03 hex	The physical configuration and the configuration in EcoStruxure Machine Expert match.
TM3_POWER_SUPPLY_ERROR	04 hex	Error detected in power supply.

PLC_W_COMMAND: Control Command Codes

Enumerated Type Description

The PLC_W_COMMAND enumeration data type contains the following values:

Enumerator	Value	Comment
PLC_W_STOP	0001 hex	Command to stop the controller.
PLC_W_RUN	0002 hex	Command to run the controller.
PLC_W_RESET_COLD	0004 hex	Command to initiate a Controller cold reset.
PLC_W_RESET_WARM	0008 hex	Command to initiate a Controller warm reset.

Section 3.2
DataFileCopy System Variables Data Types

Overview

This section lists and describes the system variable data types included in the `DataFileCopy` structures.

What Is in This Section?

This section contains the following topics:

DataFileCopyError: Detected Error Codes

Enumerated Type Description

The `DataFileCopyError` enumeration data type contains the following values:

Enumerator	Value	Description
`ERR_NO_ERR`	00 hex	No error detected.
`ERR_FILE_NOT_FOUND`	01 hex	The file does not exist.
`ERR_FILE_ACCESS_REFUSED`	02 hex	The file cannot be opened.
`ERR_INCORRECT_SIZE`	03 hex	The request size is not the same as size read from file.
`ERR_CRC_ERR`	04 hex	The CRC is not correct and the file is assumed to be corrupted.
`ERR_INCORRECT_MAC`	05 hex	The controller attempting to read from the file does not have the same MAC address as that contained in the file.

DataFileCopyLocation: Location Codes

Enumerated Type Description

The `DataFileCopyLocation` enumeration data type contains the following values:

Enumerator	Value	Description
`DFCL_INTERNAL`	00 hex	Data file with DTA extension is located in */usr/Dta* directory.
`DFCL_EXTERNAL`	01 hex	Data file with DTA extension is located in */sd0/usr/Dta* directory.
`DFCL_TBD`	02 hex	Not used.

Section 3.3
ExecScript System Variables Data Types

ExecuteScriptError: Detected Error Codes

Enumerated Type Description

The `ExecuteScriptError` enumeration data type contains the following values:

Enumerator	Value	Description
`CMD_OK`	00 hex	No error detected.
`ERR_CMD_UNKNOWN`	01 hex	The command is not recognized.
`ERR_SD_CARD_MISSING`	02 hex	SD card is not present.
`ERR_SEE_FWLOG`	03 hex	There was an error detected during command execution, see `FwLog.txt`. For more information, refer to File Type *(see Modicon M251 Logic Controller, Programming Guide)*.
`ERR_ONLY_ONE_COMMAND_ALLOWED`	04 hex	An attempt was made to execute several scripts simultaneously.
`CMD_BEING_EXECUTED`	05 hex	A script is already in progress.

Section 3.4
ETH_RW System Variables Data Types

Overview

This section lists and describes the system variable data types included in the ETH_R and ETH_W structures.

What Is in This Section?

This section contains the following topics:

ETH_R_FRAME_PROTOCOL: Frame Transmission Protocol Codes

Enumerated Type Description

The ETH_R_FRAME_PROTOCOL enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_802_3	00 hex	The protocol used for frame transmission is IEEE 802.3.
ETH_R_ETHERNET_II	01 hex	The protocol used for frame transmission is Ethernet II.

ETH_R_IP_MODE: IP Address Source Codes

Enumerated Type Description

The ETH_R_IP_MODE enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_STORED	00 hex	Stored IP address is used.
ETH_R_BOOTP	01 hex	Bootstrap protocol is used to get an IP address.
ETH_R_DHCP	02 hex	DHCP protocol is used to get an IP address.
ETH_DEFAULT_IP	FF hex	Default IP address is used.

ETH_R_PORT_DUPLEX_STATUS: Transmission Mode Codes

Enumerated Type Description

The ETH_R_PORT_DUPLEX_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_PORT_HALF_DUPLEX	00 hex	Half duplex transmission mode is used.
ETH_R_FULL_DUPLEX	01 hex	Full duplex transmission mode is used.
ETH_R_PORT_NA_DUPLEX	03 hex	No duplex transmission mode is used.

ETH_R_PORT_IP_STATUS: Ethernet TCP/IP Port Status Codes

Enumerated Type Description

The ETH_R_PORT_IP_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
WAIT_FOR_PARAMS	00 hex	Waiting for parameters.
WAIT_FOR_CONF	01 hex	Waiting for configuration.
DATA_EXCHANGE	02 hex	Ready for data exchange.
ETH_ERROR	03 hex	Ethernet TCP/IP port error detected (cable disconnected, invalid configuration, and so on).
DUPLICATE_IP	04 hex	IP address already used by another equipment.

ETH_R_PORT_LINK_STATUS: Communication Link Status Codes

Enumerated Type Description

The ETH_R_PORT_LINK_STATUS enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_LINK_DOWN	00 hex	Communication link not available to another device.
ETH_R_LINK_UP	01 hex	Communication link available to another device.

ETH_R_PORT_SPEED: Communication Speed of the Ethernet Port Codes

Enumerated Type Description

The ETH_R_PORT_SPEED enumeration data type contains the following values:

Enumerator	Value	Comment
ETH_R_SPEED_NA	0 dec	Network speed is not available.
ETH_R_SPEED_10_MB	10 dec	Network speed is 10 megabits per second.
ETH_R_100_MB	100 dec	Network speed is 100 megabits per second.

ETH_R_RUN_IDLE: Ethernet/IP Run and Idle States Codes

Enumerated Type Description

The `ETH_R_RUN_IDLE` enumeration data type contains the following values:

Enumerator	Value	Comment
`IDLE`	00 hex	EtherNet/IP connection is idle.
`RUN`	01 hex	EtherNet/IP connection is running.

Section 3.5
TM3_MODULE_RW System Variables Data Types

Overview

This section lists and describes the system variable data types included in the `TM3_MODULE_R` and `TM3_MODULE_W` structures.

What Is in This Section?

This section contains the following topics:

TM3_ERR_CODE: TM3 Expansion Module Detected Error Codes

Enumerated Type Description

The TM3_ERR_CODE enumeration data type contains the following values:

Enumerator	Value	Comment
TM3_NO_ERR	00 hex	Last bus exchange with the expansion module was successful.
TM3_ERR_FAILED	01 hex	Error detected due to the last bus exchange with the expansion module was unsuccessful.
TM3_ERR_PARAMETER	02 hex	Parameter error detected in the last bus exchange with the module.
TM3_ERR_COK	03 hex	Temporary or permanent hardware error detected on one of the TM3 expansion modules.
TM3_ERR_BUS	04 hex	Bus error detected in the last bus exchange with the expansion module.

TM3_MODULE_R_ARRAY_TYPE: TM3 Expansion Module Read Array Type

Description

The TM3_MODULE_R_ARRAY_TYPE is an array of 0...13 TM3_MODULE_R_STRUCT.

TM3_MODULE_STATE: TM3 Expansion Module State Codes

Enumerated Type Description

The TM3_MODULE_STATE enumeration data type contains the following values:

Enumerator	Value	Comment
TM3_EMPTY	00 hex	No module.
TM3_CONF_ERROR	01 hex	Physical expansion module does not match with the one configured in EcoStruxure Machine Expert.
TM3_BUS_ERROR	02 hex	Bus error detected in the last exchange with the module.
TM3_OK	03 hex	Last bus exchange with this module was successful.
TM3_MISSING_OPT_MOD	05 hex	Optional module is not physically present.

TM3_BUS_W_IOBUSERRMOD: TM3 bus error mode

Enumerated Type Description

The TM3_BUS_W_IOBUSERRMOD enumeration data type contains the following values:

Enumerator	Value	Comment
IOBUS_ERR_ACTIVE	00 hex	Active mode. The logic controller stops all I/O exchanges on the TM3 bus on detection of a permanent error. Refer to I/O Configuration General Description *(see Modicon M251 Logic Controller, Programming Guide)*.
IOBUS_ERR_PASSIVE	01 hex	Passive mode. I/O exchanges continue on the TM3 bus even if an error is detected.

Section 3.6
System Function Data Types

RTCSETDRIFT_ERROR: `SetRTCDrift` Function Detected Error Codes

Enumerated Type Description

The `RTCSETDRIFT_ERROR` enumeration data type contains the following values:

Enumerator	Value	Comment
`RTC_OK`	00 hex	RTC drift correctly configured.
`RTC_BAD_DAY`	01 hex	Not used.
`RTC_BAD_HOUR`	02 hex	Not used.
`RTC_BAD_MINUTE`	03 hex	Not used.
`RTC_BAD_DRIFT`	04 hex	RTC Drift parameter out of range.
`RTC_INTERNAL_ERROR`	05 hex	RTC Drift settings rejected on internal error detected.

Appendices

Appendix A
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

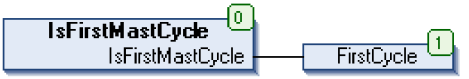
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

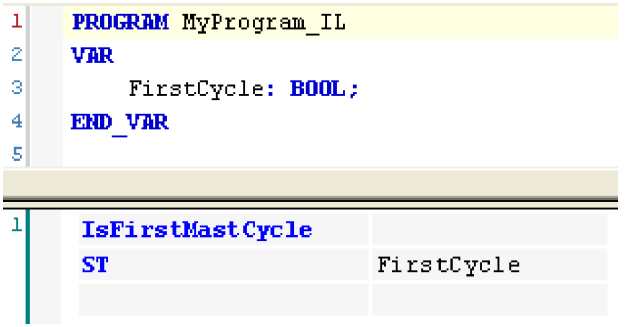
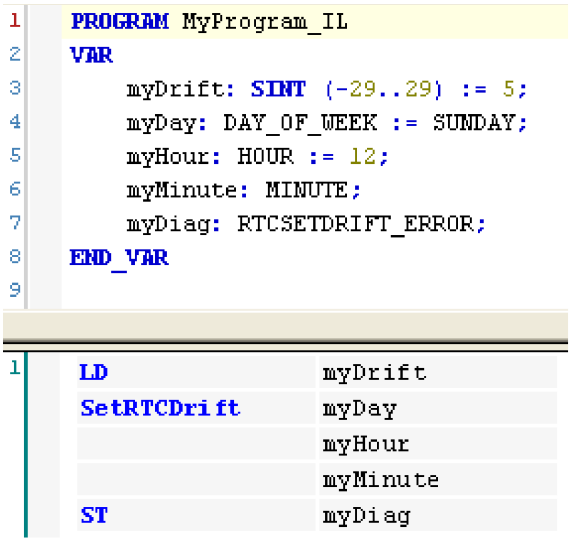
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

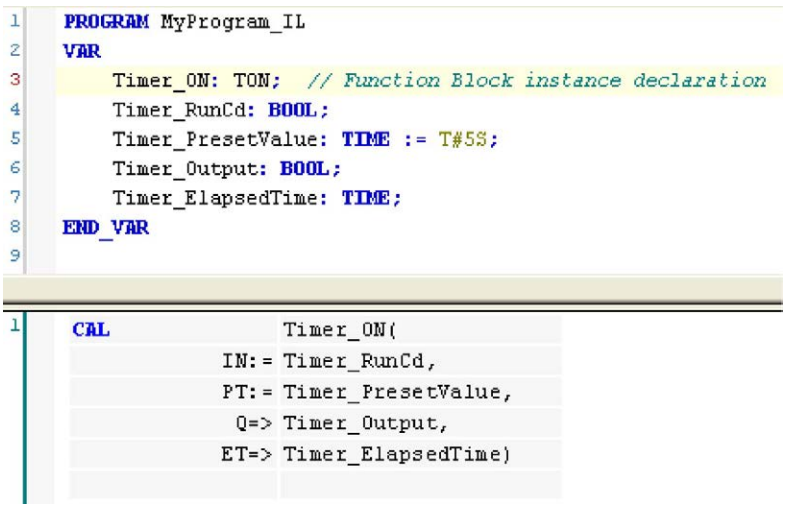
This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a CAL instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the TON Function Block graphically presented below:

Function Block	Graphical Representation
TON	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	

```
PROGRAM MyProgram_IL
VAR
    Timer_ON: TON;  // Function Block instance declaration
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
CAL          Timer_ON(
         IN:= Timer_RunCd,
         PT:= Timer_PresetValue,
          Q=> Timer_Output,
         ET=> Timer_ElapsedTime)
```

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

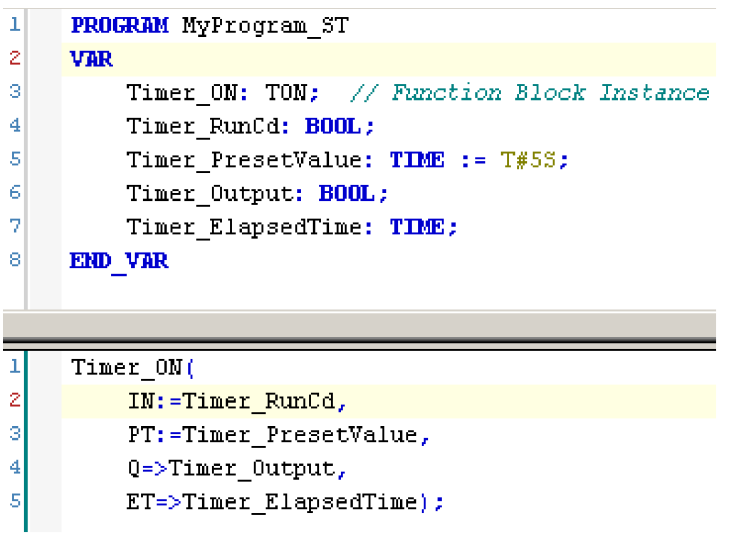
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

!

%MW

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

A

application

A program including configuration data, symbols, and documentation.

ARRAY

The systematic arrangement of data objects of a single type in the form of a table defined in logic controller memory. The syntax is as follows: `ARRAY [<dimension>] OF <Type>`

Example 1: `ARRAY [1..2] OF BOOL` is a 1-dimensional table with 2 elements of type `BOOL`.

Example 2: `ARRAY [1..10, 1..20] OF INT` is a 2-dimensional table with 10 x 20 elements of type `INT`.

B

BOOL

(*boolean*) A basic data type in computing. A `BOOL` variable can have one of these values: 0 (FALSE), 1 (TRUE). A bit that is extracted from a word is of type `BOOL`; for example, `%MW10.4` is a fifth bit of memory word number 10.

Boot application

(*boot application*) The binary file that contains the application. Usually, it is stored in the controller and allows the controller to boot on the application that the user has generated.

BOOTP

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

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

control network

A network containing logic controllers, SCADA systems, PCs, HMI, switches, ...

Two kinds of topologies are supported:

- flat: all modules and devices in this network belong to same subnet.
- 2 levels: the network is split into an operation network and an inter-controller network.

These two networks can be physically independent, but are generally linked by a routing device.

CRC

(*cyclical redundancy check*) A method used to determine the validity of a communication transmission. The transmission contains a bit field that constitutes a checksum. The message is used to calculate the checksum by the transmitter according to the content of the message. Receiving nodes, then recalculate the field in the same manner. Any discrepancy in the value of the 2 CRC calculations indicates that the transmitted message and the received message are different.

D

DHCP

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

DWORD

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

E

element

The short name of the ARRAY element.

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.

EtherNet/IP

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

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

firmware

Represents the BIOS, data parameters, and programming instructions that constitute the operating system on a controller. The firmware is stored in non-volatile memory within the controller.

flash memory

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

function

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

Examples: boolean (AND) operators, calculations, conversions (BYTE_TO_INT)

function block

A programming unit that has 1 or more inputs and returns 1 or more outputs. FBs are called through an instance (function block copy with dedicated name and variables) and each instance has a persistent state (outputs and internal variables) from 1 call to the other.

Examples: timers, counters

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

G

GVL

(*global variable list*) Manages global variables within an EcoStruxure Machine Expert project.

H

hex

(*hexadecimal*)

I

ID

(*identifier/identification*)

IEC

(*international electrotechnical commission*) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

IEC 61131-3

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

IEEE 802.3

A collection of IEEE standards defining the physical layer, and the media access control sublayer of the data link layer, of wired Ethernet.

IL

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

INT

(*integer*) A whole number encoded in 16 bits.

IP

(*Internet protocol* Part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

L

LD

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

LWORD

(*long word*) A data type encoded in a 64-bit format.

M

MAC address

(*media access control address*) A unique 48-bit number associated with a specific piece of hardware. The MAC address is programmed into each network card or device when it is manufactured.

MAST

A processor task that is run through its programming software. The MAST task has 2 sections:

- **IN:** Inputs are copied to the IN section before execution of the MAST task.
- **OUT:** Outputs are copied to the OUT section after execution of the MAST task.

N

network

A system of interconnected devices that share a common data path and protocol for communications.

P

PLC

(*programmable logic controller*) An industrial computer used to automate manufacturing, industrial, and other electromechanical processes. PLCs are different from common computers in that they are designed to have multiple input and output arrays and adhere to more robust specifications for shock, vibration, temperature, and electrical interference among other things.

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

protocol

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

R

run

A command that causes the controller to scan the application program, read the physical inputs, and write to the physical outputs according to solution of the logic of the program.

S

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

STOP

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

string

A variable that is a series of ASCII characters.

system variable

A variable that provides controller data and diagnostic information and allows sending commands to the controller.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

TCP

(*transmission control protocol*) A connection-based transport layer protocol that provides a simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

U

UDINT

(*unsigned double integer*) Encoded in 32 bits.

UINT

(*unsigned integer*) Encoded in 16 bits.

unlocated variable

A variable that does not have an address (refer to *located variable*).

V

variable

A memory unit that is addressed and modified by a program.

W

watchdog

A watchdog is a special timer used to ensure that programs do not overrun their allocated scan time. The watchdog timer is usually set to a higher value than the scan time and reset to 0 at the end of each scan cycle. If the watchdog timer reaches the preset value, for example, because the program is caught in an endless loop, an error is declared and the program stopped.

WORD

A type encoded in a 16-bit format.

Index

Specials

C

D

E

F

G

I

M

P

R

S

T

U

Modicon M251

Logic Controller
Hardware Guide

12/2019

EIO0000003101.01

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

QUALIFICATION OF PERSONNEL

Only appropriately trained persons who are familiar with and understand the contents of this manual and all other pertinent product documentation are authorized to work on and with this product.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

INTENDED USE

The products described or affected by this document, together with software, accessories, and options, are programmable logic controllers (referred to herein as “logic controllers”), intended for industrial use according to the instructions, directions, examples, and safety information contained in the present document and other supporting documentation.

The product may only be used in compliance with all applicable safety regulations and directives, the specified requirements, and the technical data.

Prior to using the product, you must perform a risk assessment in view of the planned application. Based on the results, the appropriate safety-related measures must be implemented.

Since the product is used as a component in an overall machine or process, you must ensure the safety of persons by means of the design of this overall system.

Operate the product only with the specified cables and accessories. Use only genuine accessories and spare parts.

Any use other than the use explicitly permitted is prohibited and can result in unanticipated hazards.

About the Book

At a Glance

Document Scope

Use this document to:

- Install and operate your M251 Logic Controller.
- Connect the M251 Logic Controller to a programming device equipped with EcoStruxure Machine Expert software.
- Interface the M251 Logic Controller with I/O expansion modules, HMI and other devices.
- Familiarize yourself with the M251 Logic Controller features.

NOTE: Read and understand this document and all related documents before installing, operating, or maintaining your controller.

Validity Note

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

For product compliance and environmental information (RoHS, REACH, PEP, EOLI, etc.), go to *www.schneider-electric.com/green-premium*.

The technical characteristics of the devices described in this manual also appear online (*https://www.se.com/*).

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

Related Documents

Title of Documentation	Reference Number
Modicon M251 Logic Controller - Programming Guide	EIO0000003089 (ENG) EIO0000003090 (FRE) EIO0000003091 (GER) EIO0000003092 (SPA) EIO0000003093 (ITA) EIO0000003094 (CHS)
EcoStruxure Machine Expert Industrial Ethernet User Guide	EIO0000003053 (ENG) EIO0000003054 (FRE) EIO0000003055 (GER) EIO0000003056 (SPA) EIO0000003057 (ITA) EIO0000003058 (CHS)
Modicon TM3 Digital I/O Modules - Hardware Guide	EIO0000003125 (ENG) EIO0000003126 (FRE) EIO0000003127 (GER) EIO0000003128 (SPA) EIO0000003129 (ITA) EIO0000003130 (CHS) EIO0000003425 (TUR) EIO0000003424 (POR)
Modicon TM3 Expert I/O Modules - Hardware Guide	EIO0000003137 (ENG) EIO0000003138 (FRE) EIO0000003139 (GER) EIO0000003140 (SPA) EIO0000003141 (ITA) EIO0000003142 (CHS) EIO0000003429 (TUR) EIO0000003428 (POR)
Modicon TM3 Safety Modules - Hardware Guide	EIO0000003353 (ENG) EIO0000003354 (FRE) EIO0000003355 (GER) EIO0000003356 (SPA) EIO0000003357 (ITA) EIO0000003358 (CHS) EIO0000003359 (POR) EIO0000003360 (TUR)

Title of Documentation	Reference Number
Modicon TM3 Transmitter and Receiver Modules - Hardware Guide	*EIO0000003143 (ENG)* *EIO0000003144 (FRE)* *EIO0000003145 (GER)* *EIO0000003146 (SPA)* *EIO0000003147 (ITA)* *EIO0000003148 (CHS)* *EIO0000003431 (TUR)* *EIO0000003430 (POR)*
Modicon TM3 Bus Coupler - Hardware Guide	*EIO0000003635 (ENG)* *EIO0000003636 (FRE)* *EIO0000003637 (GER)* *EIO0000003638 (SPA)* *EIO0000003639 (ITA)* *EIO0000003640 (CHS)* *EIO0000003641 (POR)* *EIO0000003642 (TUR)*
Modicon TM4 Expansion Modules - Hardware Guide	*EIO0000003155 (ENG)* *EIO0000003156 (FRE)* *EIO0000003157 (GER)* *EIO0000003158 (SPA)* *EIO0000003159 (ITA)* *EIO0000003160 (CHS)*
Modicon TM5 Fieldbus Interface - Hardware Guide	*EIO0000003715 (ENG)* *EIO0000003716 (FRE)* *EIO0000003717 (GER)* *EIO0000003718 (SPA)* *EIO0000003719 (ITA)* *EIO0000003720 (CHS)*
M251 Logic Controller - Instruction Sheet	*HRB59604*

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

Product Related Information

⚠ ⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

⚠ DANGER

POTENTIAL FOR EXPLOSION

- Only use this equipment in non-hazardous locations, or in locations that comply with Class I, Division 2, Groups A, B, C and D.
- Do not substitute components which would impair compliance to Class I, Division 2.
- Do not connect or disconnect equipment unless power has been removed or the location is known to be non-hazardous.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

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

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

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

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

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

Part I
Modicon M251 Logic Controller Introduction

What Is in This Part?

This part contains the following chapters:

Chapter 1
M251 General Overview

Overview

This chapter provides general information about the M251 Logic Controller system architecture and its components.

What Is in This Chapter?

This chapter contains the following topics:

M251 Logic Controller Description

Overview

The M251 Logic Controller has various powerful features and can service a wide range of applications.

Software configuration, programming, and commissioning are achieved with the EcoStruxure Machine Expert software described in the EcoStruxure Machine Expert Programming Guide *(see EcoStruxure Machine Expert, Programming Guide)* and in the M251 Logic Controller Programming Guide.

Programming Languages

The M251 Logic Controller is configured and programmed with the EcoStruxure Machine Expert software, which supports the following IEC 61131-3 programming languages:

- IL: Instruction list
- ST: Structured text
- FBD: Function block diagram
- SFC: Sequential function chart
- LD: Ladder diagram

EcoStruxure Machine Expert software can also be used to program this controller using CFC (continuous function chart) language.

Power Supply

The power supply of the M251 Logic Controller is 24 Vdc *(see page 78)*.

Real Time Clock

The M251 Logic Controller includes a Real Time Clock (RTC) system *(see page 44)*.

Run/Stop

The M251 Logic Controller can be operated externally by the following:

- a hardware Run/Stop switch *(see page 48)*
- an EcoStruxure Machine Expert software command

Memory

This table describes the different types of memory:

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

Removable Storage

M251 Logic Controllers include an embedded SD card slot *(see page 49)*.

The main uses of the SD card are:

- Initializing the controller with a new application
- Updating the controller firmware
- Applying post configuration files to the controller
- Applying recipes
- Receiving data logging files

Embedded Communication Features

The M251 Logic Controller native communication ports include (depending on the controller reference):

- CANopen Master
- Ethernet *(see page 101)*
- USB Mini-B *(see page 105)*
- Serial Line *(see page 106)*

Expansion Module and Bus Coupler Compatibility

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

M251 Logic Controllers

Reference	Digital Inputs	Digital Outputs	Communication Ports
TM251MESC *(see page 87)*	0	0	1 serial line port 1 USB mini-B programming port 1 dual port Ethernet switch 1 CANopen port
TM251MESE *(see page 91)*	0	0	1 serial line port 1 USB mini-B programming port 1 dual port Ethernet switch 1 Ethernet port for fieldbus

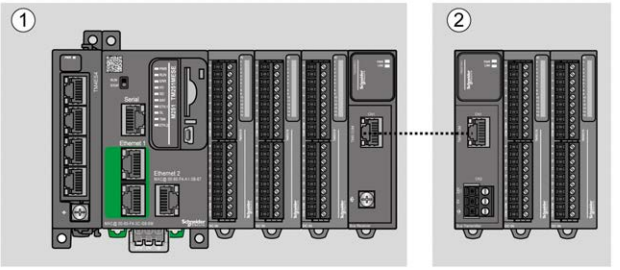
Maximum Hardware Configuration

Introduction

The M251 Logic Controller is a control system that offers a scalable solution with optimized configurations and an expandable architecture.

Local and Remote Configuration Principle

The following figure defines the local and remote configurations:



(1) Local configuration
(2) Remote configuration

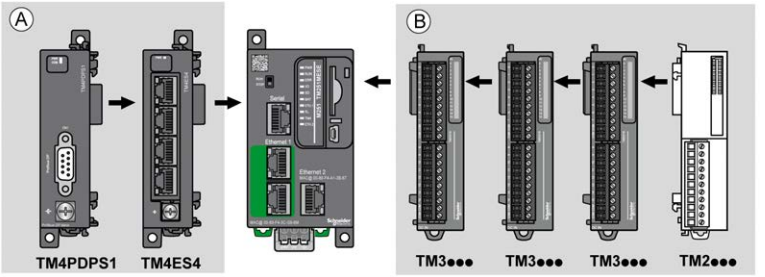
M251 Logic Controller Local Configuration Architecture

Optimized local configuration and flexibility are provided by the association of:

- M251 Logic Controller
- TM4 expansion modules
- TM3 expansion modules
- TM2 expansion modules

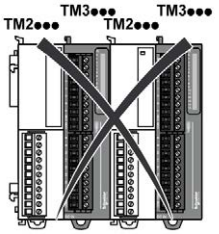
Application requirements determine the architecture of your M251 Logic Controller configuration.

The following figure represents the components of a local configuration:



(A) Expansion modules (3 maximum)
(B) Expansion modules (7 maximum)

NOTE: It is prohibited to mount a TM2 module before any TM3 module as indicated in the following figure:



M251 Logic Controller Remote Configuration Architecture

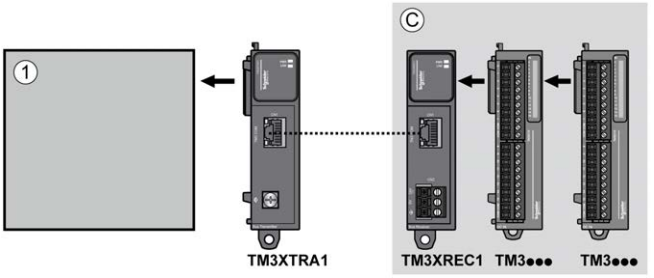
Optimized remote configuration and flexibility are provided by the association of:

- M251 Logic Controller
- TM4 expansion modules
- TM3 expansion modules
- TM3 transmitter and receiver modules

Application requirements determine the architecture of your M251 Logic Controller configuration.

NOTE: You cannot use TM2 modules in configurations that include the TM3 transmitter and receiver modules.

The following figure represents the components of a remote configuration:



(1) Logic controller and modules
(C) TM3 expansion modules (7 maximum)

Maximum Number of Modules

The following table shows the maximum configuration supported:

References	Maximum	Type of Configuration
TM251••••	7 TM3 / TM2 expansion modules	Local
TM251••••	3 TM4 expansion modules	Local
TM3XREC1	7 TM3 expansion modules	Remote
NOTE: TM3 transmitter and receiver modules are not included in a count of the maximum number of expansion modules.		

NOTE: The configuration with its TM4, TM3, and TM2 expansion modules is validated by EcoStruxure Machine Expert software in the **Configuration** window.

NOTE: In some environments, the maximum configuration populated by high consumption modules, coupled with the maximum distance allowable between the TM3 transmitter and receiver modules, may present bus communication issues although the EcoStruxure Machine Expert software allows for the configuration. In such a case you will need to analyze the power consumption of the modules chosen for your configuration, as well as the minimum cable distance required by your application, and possibly seek to optimize your choices.

TM2 Expansion Modules

Overview

You can expand the number of I/Os of your M251 Logic Controller by adding TM2 I/O expansion modules.

The following types of electronic modules are supported:
- TM2 digital I/O expansion modules
- TM2 analog I/O expansion modules

For more information, refer to the following documents:
- TM2 Digital I/O Expansion Modules Hardware Guide
- TM2 Analog I/O Expansion Modules Hardware Guide

NOTE: TM2 modules can only be used in the local configuration, and only if there is no TM3 transmitter and receiver modules present in the configuration.

NOTE: It is prohibited to mount a TM2 module before any TM3 module. The TM2 modules must be mounted and configured at the end of the local configuration.

TM2 Digital Input Expansion Modules

The following table shows the compatible TM2 digital input expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type
TM2DAI8DT	8	Regular inputs	120 Vac 7.5 mA	Removable screw terminal block
TM2DDI8DT	8	Regular inputs	24 Vdc 7 mA	Removable screw terminal block
TM2DDI16DT	16	Regular inputs	24 Vdc 7 mA	Removable screw terminal block
TM2DDI16DK	16	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector
TM2DDI32DK	32	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector

TM2 Digital Output Expansion Modules

The following table shows the compatible TM2 digital output expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal type
TM2DRA8RT	8	Relay outputs	30 Vdc / 240 Vac 2 A max	Removable screw terminal block
TM2DRA16RT	16	Relay outputs	30 Vdc / 240 Vac 2 A max	Removable screw terminal block
TM2DDO8UT	8	Regular transistor outputs (sink)	24 Vdc 0.3 A max per output	Removable screw terminal block
TM2DDO8TT	8	Regular transistor outputs (source)	24 Vdc 0.5 A max per output	Removable screw terminal block
TM2DDO16UK	16	Regular transistor outputs (sink)	24 Vdc 0.1 A max per output	HE10 (MIL 20) connector
TM2DDO16TK	16	Regular transistor outputs (source)	24 Vdc 0.4 A max per output	HE10 (MIL 20) connector
TM2DDO32UK	32	Regular transistor outputs (sink)	24 Vdc 0.1 A max per output	HE10 (MIL 20) connector
TM2DDO32TK	32	Regular transistor outputs (source)	24 Vdc 0.4 A max per output	HE10 (MIL 20) connector

TM2 Digital Mixed Input/Output Expansion Modules

The following table shows the compatible TM2 digital mixed I/O expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal type
TM2DMM8DRT	4	Regular inputs	24 Vdc 7 mA	Removable screw terminal block
	4	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM2DMM24DRF	16	Regular inputs	24 Vdc 7 mA	Non-removable spring terminal block
	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	

TM2 Analog Input Expansion Modules

The following table shows the compatible TM2 analog input expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal Type
TM2AMI2HT	2	High-level inputs	0...10 Vdc 4...20 mA	Removable screw terminal block
TM2AMI2LT	2	Low-level inputs	Thermocouple type J,K,T	Removable screw terminal block
TM2AMI4LT	4	Analog inputs	0...10 Vdc 0...20 mA PT100/1000 Ni100/1000	Removable screw terminal block
TM2AMI8HT	8	Analog inputs	0...20 mA 0...10 Vdc	Removable screw terminal block
TM2ARI8HT	8	Analog inputs	NTC / PTC	Removable screw terminal block
TM2ARI8LRJ	8	Analog inputs	PT100/1000	RJ11 connector
TM2ARI8LT	8	Analog inputs	PT100/1000	Removable screw terminal block

TM2 Analog Output Expansion Modules

The following table shows the compatible TM2 analog output expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal Type
TM2AMO1HT	1	Analog outputs	0...10 Vdc 4...20 mA	Removable screw terminal block
TM2AVO2HT	2	Analog outputs	+/- 10 Vdc	Removable screw terminal block

TM2 Analog Mixed Input/Output Expansion Modules

The following table shows the compatible TM2 analog mixed I/O expansion modules with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel type	Voltage Current	Terminal Type
TM2AMM3HT	2	Analog inputs	0...10 Vdc 4...20 mA	Removable screw terminal block
	1	Analog outputs	0...10 Vdc 4...20 mA	
TM2AMM6HT	4	Analog inputs	0...10 Vdc 4...20 mA	Removable screw terminal block
	2	Analog outputs	0...10 Vdc 4...20 mA	
TM2ALM3LT	2	Low-level inputs	Thermo J,K,T, PT100	Removable screw terminal block
	1	Analog outputs	0...10 Vdc 4...20 mA	

TM3 Expansion Modules

Introduction

The range of TM3 expansion modules includes:

- Digital modules, classified as follows:
  - Input modules *(see page 27)*
  - Output modules *(see page 28)*
  - Mixed input/output modules *(see page 30)*
- Analog modules, classified as follows:
  - Input modules *(see page 31)*
  - Output modules *(see page 32)*
  - Mixed input/output modules *(see page 33)*
- Expert modules *(see page 34)*
- Safety modules *(see page 34)*
- Transmitter and Receiver modules *(see page 35)*

For more information, refer to the following documents:

- TM3 Digital I/O Modules Hardware Guide
- TM3 Analog I/O Modules Hardware Guide
- TM3 Expert I/O Modules Hardware Guide
- TM3 Safety Modules Hardware Guide
- TM3 Transmitter and Receiver Modules Hardware Guide

TM3 Digital Input Modules

The following table shows the TM3 digital input expansion modules, with corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DI8A	8	Regular inputs	120 Vac 7.5 mA	Removable screw terminal block / 5.08 mm
TM3DI8	8	Regular inputs	24 Vdc 7 mA	Removable screw terminal block / 5.08 mm
TM3DI8G	8	Regular inputs	24 Vdc 7 mA	Removable spring terminal block / 5.08 mm
TM3DI16	16	Regular inputs	24 Vdc 7 mA	Removable screw terminal blocks / 3.81 mm

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DI16G	16	Regular inputs	24 Vdc 7 mA	Removable spring terminal blocks / 3.81 mm
TM3DI16K	16	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector
TM3DI32K	32	Regular inputs	24 Vdc 5 mA	HE10 (MIL 20) connector

TM3 Digital Output Modules

The following table shows the TM3 digital output expansion modules, with corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DQ8R	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	Removable screw terminal block / 5.08 mm
TM3DQ8RG	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	Removable spring terminal block / 5.08 mm
TM3DQ8T	8	Regular transistor outputs (source)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable screw terminal block / 5.08 mm
TM3DQ8TG	8	Regular transistor outputs (source)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable spring terminal block / 5.08 mm
TM3DQ8U	8	Regular transistor outputs (sink)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable screw terminal block / 5.08 mm
TM3DQ8UG	8	Regular transistor outputs (sink)	24 Vdc 4 A maximum per common line/0.5 A maximum per output	Removable spring terminal block / 5.08 mm
TM3DQ16R	16	Relay outputs	24 Vdc / 240 Vac 8 A maximum per common line / 2 A maximum per output	Removable screw terminal blocks / 3.81 mm

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DQ16RG	16	Relay outputs	24 Vdc / 240 Vac 8 A maximum per common line / 2 A maximum per output	Removable spring terminal blocks / 3.81 mm
TM3DQ16T	16	Regular transistor outputs (source)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable screw terminal blocks / 3.81 mm
TM3DQ16TG	16	Regular transistor outputs (source)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable spring terminal blocks / 3.81 mm
TM3DQ16U	16	Regular transistor outputs (sink)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable screw terminal blocks / 3.81 mm
TM3DQ16UG	16	Regular transistor outputs (sink)	24 Vdc 8 A maximum per common line / 0.5 A maximum per output	Removable spring terminal blocks / 3.81 mm
TM3DQ16TK	16	Regular transistor outputs (source)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connector
TM3DQ16UK	16	Regular transistor outputs (sink)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connector
TM3DQ32TK	32	Regular transistor outputs (source)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connectors
TM3DQ32UK	32	Regular transistor outputs (sink)	24 Vdc 2 A maximum per common line / 0.1 A maximum per output	HE10 (MIL 20) connectors

TM3 Digital Mixed Input/Output Modules

This following table shows the TM3 mixed I/O modules, with corresponding channel type, nominal voltage/current, and terminal type:

Reference	Channels	Channel Type	Voltage Current	Terminal Type / Pitch
TM3DM8R	4	Regular inputs	24 Vdc 7 mA	Removable screw terminal block / 5.08 mm
	4	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM3DM8RG	4	Regular inputs	24 Vdc 7 mA	Removable spring terminal block /5.08 mm
	4	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM3DM24R	16	Regular inputs	24 Vdc 7 mA	Removable screw terminal blocks / 3.81 mm
	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	
TM3DM24RG	16	Regular inputs	24 Vdc 7 mA	Removable spring terminal blocks / 3.81 mm
	8	Relay outputs	24 Vdc / 240 Vac 7 A maximum per common line / 2 A maximum per output	

TM3 Analog Input Modules

The following table shows the TM3 analog input expansion modules, with corresponding resolution, channel type, nominal voltage/current, and terminal type:

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3AI2H	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 5.08 mm
TM3AI2HG	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 5.08 mm
TM3AI4	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 3.81 mm
TM3AI4G	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal blocks / 3.81 mm
TM3AI8	12 bit, or 11 bit + sign	8	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA 0...20 mA extended 4...20 mA extended	Removable screw terminal block / 3.81 mm
TM3AI8G	12 bit, or 11 bit + sign	8	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA 0...20 mA extended 4...20 mA extended	Removable spring terminal blocks / 3.81 mm
TM3TI4	16 bit, or 15 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable screw terminal block / 3.81 mm

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3TI4G	16 bit, or 15 bit + sign	4	inputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable spring terminal blocks / 3.81 mm
TM3TI4D	16 bit, or 15 bit + sign	4	inputs	Thermocouple	Removable screw terminal block / 3.81 mm
TM3TI4DG	16 bit, or 15 bit + sign	4	inputs	Thermocouple	Removable spring terminal blocks / 3.81 mm
TM3TI8T	16 bit, or 15 bit + sign	8	inputs	Thermocouple NTC/PTC Ohmmeter	Removable screw terminal block / 3.81 mm
TM3TI8TG	16 bit, or 15 bit + sign	8	inputs	Thermocouple NTC/PTC Ohmmeter	Removable spring terminal blocks / 3.81 mm

TM3 Analog Output Modules

The following table shows the TM3 analog output modules, with corresponding resolution, channel type, nominal voltage/current, and terminal type:

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3AQ2	12 bit, or 11 bit + sign	2	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 5.08 mm
TM3AQ2G	12 bit, or 11 bit + sign	2	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 5.08 mm
TM3AQ4	12 bit, or 11 bit + sign	4	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 5.08 mm
TM3AQ4G	12 bit, or 11 bit + sign	4	outputs	0...10 Vdc -10…+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 5.08 mm

TM3 Analog Mixed Input/Output Modules

This following table shows the TM3 analog mixed I/O modules, with corresponding resolution, channel type, nominal voltage/current, and terminal type:

Reference	Resolution	Channels	Channel Type	Mode	Terminal Type / Pitch
TM3AM6	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	Removable screw terminal block / 3.81 mm
		2	outputs		
TM3AM6G	12 bit, or 11 bit + sign	4	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	Removable spring terminal block / 3.81 mm
		2	outputs		
TM3TM3	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable screw terminal block / 5.08 mm
	12 bit, or 11 bit + sign	1	outputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	
TM3TM3G	16 bit, or 15 bit + sign	2	inputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA Thermocouple PT100/1000 NI100/1000	Removable spring terminal block / 5.08 mm
	12 bit, or 11 bit + sign	1	outputs	0...10 Vdc -10...+10 Vdc 0...20 mA 4...20 mA	

TM3 Expert Modules

The following table shows the TM3 expert expansion modules, with corresponding terminal types:

Reference	Description	Terminal Type / Pitch
TM3XTYS4 *(see Modicon TM3, Expert I/O Modules, Hardware Guide)*	TeSys module	4 front connectors RJ-45 1 removable power supply connector / 5.08 mm
TM3XHSC202 *(see Modicon TM3, Expert I/O Modules, Hardware Guide)*	High Speed Counting (HSC) module	Removable screw terminal blocks / 3.81 mm
TM3XHSC202G *(see Modicon TM3, Expert I/O Modules, Hardware Guide)*	High Speed Counting (HSC) module	Removable spring terminal blocks / 3.81 mm

TM3 Safety Modules

This table contains the TM3 safety modules, with the corresponding channel type, nominal voltage/current, and terminal type:

Reference	Function Category	Channels	Channel type	Voltage Current	Terminal type
TM3SAC5R	1 function, up to category 3	1 or 2 (1)	Safety input	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start (2)	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAC5RG	1 function, up to category 3	1 or 2 (1)	Safety input	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start (2)	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAF5R	1 function, up to category 4	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	

(1) Depending on external wiring
(2) Non-monitored start

Reference	Function Category	Channels	Channel type	Voltage Current	Terminal type
TM3SAF5RG	1 function, up to category 4	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAFL5R	2 functions, up to category 3	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAFL5RG	2 functions, up to category 3	2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAK6R	3 functions, up to category 4	1 or 2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable screw terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	
TM3SAK6RG	3 functions, up to category 4	1 or 2 (1)	Safety inputs	24 Vdc 100 mA maximum	3.81 mm (0.15 in.) and 5.08 mm (0.20 in.), removable spring terminal block
		Start	Input		
		3 in parallel	Relay outputs Normally open	24 Vdc / 230 Vac 6 A maximum per output	

(1) Depending on external wiring
(2) Non-monitored start

TM3 Transmitter and Receiver Modules

The following table shows the TM3 transmitter and receiver expansion modules:

Reference	Description	Terminal Type / Pitch
TM3XTRA1	Data transmitter module for remote I/O	1 front connector RJ-45 1 screw for functional ground connection
TM3XREC1	Data receiver module for remote I/O	1 front connector RJ-45 Power supply connector / 5.08 mm

TM3 Bus Couplers

Introduction

The TM3 bus coupler is a device designed to manage fieldbus communication when using TM2 and TM3 expansion modules in a distributed architecture.

For more information, refer to the Modicon TM3 Bus Coupler Hardware Guide *(see Modicon TM3 Bus Coupler, Hardware Guide)*.

Modicon TM3 Bus Couplers

The following table shows the TM3 bus couplers, with ports and terminal types:

Reference	Port	Communication type	Terminal type
TM3BCEIP	2 isolated switched Ethernet ports	EtherNet/IP Modbus TCP	RJ45
	1 USB mini-B port	USB 2.0	USB mini-B
TM3BCSL	2 isolated ports	Serial Line Modbus	RJ45
	1 USB mini-B port	USB 2.0	USB mini-B

TM4 Expansion Modules

Introduction

The range of TM4 expansion modules includes communication modules.

For more information, refer to the TM4 Expansion Modules Hardware Guide.

TM4 Expansion Modules

The following table shows the TM4 expansion module features:

Module reference	Type	Terminal type
TM4ES4	Ethernet communication	4 RJ45 connectors 1 screw for functional ground connection
TM4PDPS1	PROFIBUS DP slave communication	1 SUB-D 9 pins female connector 1 screw for functional ground connection
NOTE: The TM4ES4 module has two applications: expansion or standalone. For more information, refer to TM4 Compatibility.		

TM5 Fieldbus Interfaces

Introduction

The TM5 fieldbus interfaces are devices designed to manage EtherNet/IP communication when using TM5 System and TM7 expansion modules with a controller in a distributed architecture.

For more information, refer to the Modicon TM5 System Interface – Hardware Guide.

TM5 Fieldbus Interfaces

The following table shows the TM5 fieldbus interfaces with ports and terminal type:

Reference	Port	Communication type	Terminal type
TM5NEIP1	2 Ethernet switched ports	EtherNet/IP	RJ45

TM5 CANopen Fieldbus Interfaces

Introduction

The TM5 fieldbus module is a CANopen interface with built-in power distribution and is the first TM5 distributed I/O island.

For more information, refer to the Modicon TM5 CANopen Interface Hardware Guide.

Modicon TM5 CANopen Fieldbus Interfaces

The following table shows the TM5 CANopen fieldbus interfaces:

Reference	Communication type	Terminal type
TM5NCO1	CANopen	1 SUB-D 9, male

TM7 CANopen Fieldbus Interfaces

Introduction

The TM7 fieldbus modules are CANopen interfaces with 24 Vdc digital configurable input or output on 8 or 16 channels.

For more information, refer to the Modicon TM7 CANopen Interface I/O Blocks Hardware Guide.

Modicon TM7 CANopen Fieldbus Interfaces

The following table shows the TM7 CANopen fieldbus interfaces:

Reference	Number of channels	Voltage/Current	Communication type	Terminal type
TM7NCOM08B	8 inputs 8 outputs	24 Vdc / 4 mA 24 Vdc / 500 mA	CANopen	M8 Connector
TM7NCOM16A	16 inputs 16 outputs	24 Vdc / 4 mA 24 Vdc / 500 mA	CANopen	M8 Connector
TM7NCOM16B	16 inputs 16 outputs	24 Vdc / 4 mA 24 Vdc / 500 mA	CANopen	M12 Connector

Accessories

Overview

This section describes the accessories and cables.

Accessories

Reference	Description	Use	Quantity
TMASD1	SD Card *(see page 49)*	Use to update the controller firmware, initialize a controller with a new application or clone a controller, manage user files, etc.,.	1
TMAT2PSET	Set of 5 removable screw terminal block	Connects 24 Vdc power supply.	1
NSYTRAAB35	End brackets	Helps secure the controller or receiver module and their expansion modules on a top hat section rail (DIN rail).	1
TM2XMTGB	Grounding Bar	Connects the cable shield and the module to the functional ground.	1
TM200RSRCEMC	Shielding take-up clip	Mounts and connects the ground to the cable shielding.	25 pack

Cables

Reference	Description	Details	Length
TCSXCNAMUM3P	Terminal port/USB port cordset	From the USB mini-B port on the M251 Logic Controller to USB port on the PC terminal.	3 m (10 ft)
TCSMCN3M4F3C2	RS-232 serial link cordset 1 RJ45 connector and 1 SUB-D 9 connector	For DTE terminal (printer).	3 m (9.84 ft)
TCSMCN3M4M3S2	RS-232 serial link cordset 1 RJ45 connector and 1 SUB-D 9 connector	For DCE terminal (modem, converter).	3 m (9.84 ft)

Reference	Description	Details	Length
490NTW000••	Ethernet shielded cable for DTE connections	Standard cable, equipped with RJ45 connectors at each end for DTE. CE compliant.	2, 5, 12, 40, or 80 m (6.56, 16.4, 39.37, 131.23 or 262.47 ft)
490NTW000••U		Standard cable, equipped with RJ45 connectors at each end for DTE. UL compliant.	2, 5, 12, 40, or 80 m (6.56, 16.4, 39.37, 131.23, or 262.47 ft)
TCSECE3M3M••S4		Cable for harsh environment, equipped with RJ45 connectors at each end. CE compliant.	1, 2, 3, 5, or 10 m (3.28, 6.56, 9.84, 16.4, 32.81 ft)
TCSECU3M3M••S4		Cable for harsh environment, equipped with RJ45 connectors at each end. UL compliant.	1, 2, 3, 5, or 10 m (3.28, 6.56, 9.84, 16.4, 32.81 ft)

Chapter 2
M251 Features

Overview

This chapter describes the Modicon M251 Logic Controller features.

What Is in This Chapter?

This chapter contains the following topics:

Real Time Clock (RTC)

Overview

The M251 Logic Controller includes an RTC to provide system date and time information, and to support related functions requiring a real-time clock. To continue keeping time when power is off, a non-rechargeable battery is required (see reference below). A battery LED on the front panel of the controller indicates if the battery is depleted or absent.

This table shows how RTC drift is managed:

RTC Characteristics	Description
RTC drift	Less than 60 seconds per month without any user calibration at 25 °C (77 °F)

Battery

The controller has one battery.

In the event of a power interruption, the backup battery maintains the RTC for the controller.

This table shows the characteristics of the battery:

Characteristics	Description
Use	In the event of a transient power outage, the battery powers the RTC.
Backup life	At least 2 years at 25 °C maximum (77 °F). At higher temperatures, the time is reduced.
Battery monitoring	Yes
Replaceable	Yes
Controller battery type	Lithium carbon monofluoride, type Panasonic BR2032

Installing and Replacing the Battery

While lithium batteries are preferred due to their slow discharge and long life, they can present hazards to personnel, equipment and the environment and must be handled properly.

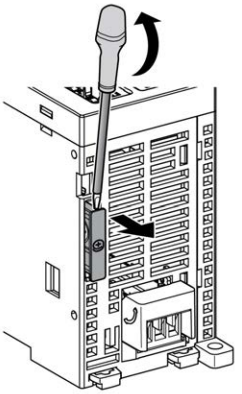
⚠ DANGER

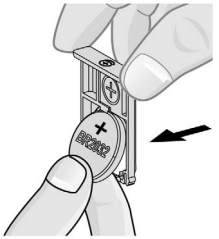
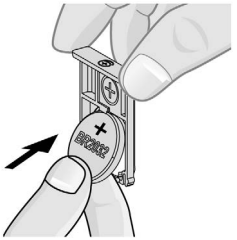
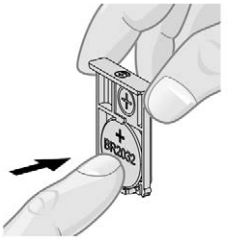
EXPLOSION, FIRE, OR CHEMICAL BURNS

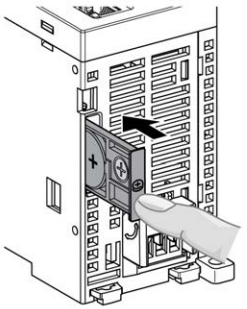
- Replace with identical battery type.
- Follow all the instructions of the battery manufacturer.
- Remove all replaceable batteries before discarding unit.
- Recycle or properly dispose of used batteries.
- Protect battery from any potential short-circuit.
- Do not recharge, disassemble, heat above 100 °C (212 °F), or incinerate.
- Use your hands or insulated tools to remove or replace the battery.
- Maintain proper polarity when inserting and connecting a new battery.

Failure to follow these instructions will result in death or serious injury.

To install or replace the battery, follow these steps:

Step	Action
1	Remove power from your controller.
2	Use an insulated screw-driver to pull out the battery holder. 
3	Side out the battery holder of the controller

Step	Action
4	Remove the battery from the battery holder. 
5	Insert the new battery into the battery holder in accordance with the polarity markings on the battery. 
6	Replace the battery holder on the controller and verify that the latch clicks into place. 

Step	Action
7	Slide in the battery holder of the controller. 
8	Power up your M251 Logic Controller.
9	Set the internal clock. For further details on the internal clock, refer to M251 Logic Controller Programming Guide *(see Modicon M251 Logic Controller, Programming Guide)*.

NOTE: Replacement of the battery in the controllers other than with the type specified in this documentation may present a risk of fire or explosion.

⚠ WARNING
IMPROPER BATTERY CAN PROVOKE FIRE OR EXPLOSION
Replace battery only with identical type: Panasonic Type BR2032.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

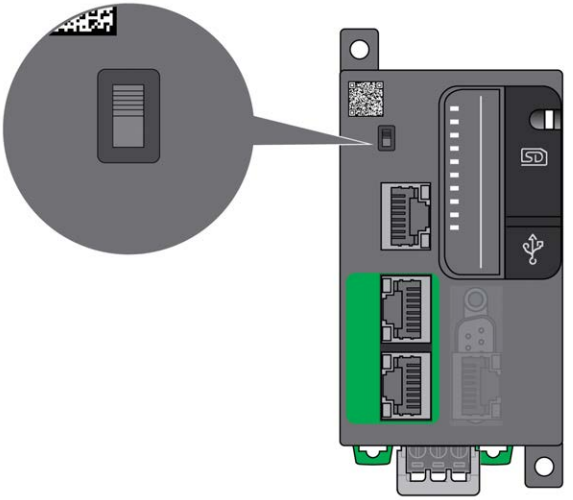
Run/Stop

Run/Stop

The M251 Logic Controller can be operated externally by the following:

- a hardware Run/Stop switch
- an EcoStruxure Machine Expert software command.

The M251 Logic Controller has a Run/Stop hardware switch, which puts the controller in a RUN or STOP state.



SD Card

Overview

When handling the SD card, follow the instructions below to help prevent internal data on the SD card from being corrupted or lost or an SD card malfunction from occurring:

NOTICE
LOSS OF APPLICATION DATA ● Do not store the SD card where there is static electricity or probable electromagnetic fields. ● Do not store the SD card in direct sunlight, near a heater, or other locations where high temperatures can occur. ● Do not bend the SD card. ● Do not drop or strike the SD card against another object. ● Keep the SD card dry. ● Do not touch the SD card connectors. ● Do not disassemble or modify the SD card. ● Use only SD cards formatted using FAT or FAT32. **Failure to follow these instructions can result in equipment damage.**

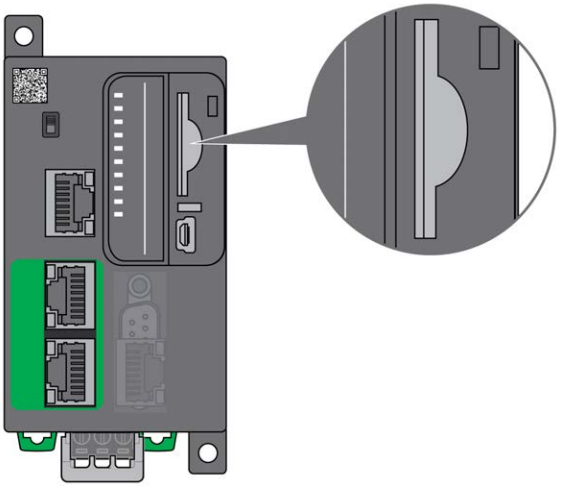
The M251 Logic Controller does not recognize NTFS formatted SD cards. Format the SD card on your computer using FAT or FAT32.

When using the M251 Logic Controller and an SD card, observe the following to avoid losing valuable data:

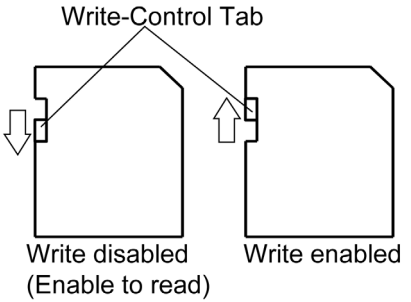
- Accidental data loss can occur at any time. Once data is lost it cannot be recovered.
- If you forcibly extract the SD card, data on the SD card may become corrupted.
- Removing an SD card that is being accessed could damage the SD card, or corrupt its data.
- If the SD card is not positioned correctly when inserted into the controller, the data on the card and the controller could become damaged.

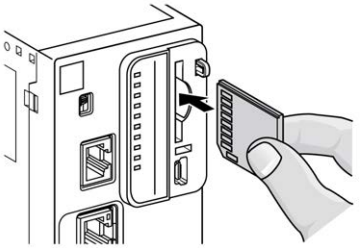
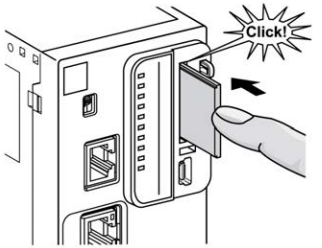
NOTICE
LOSS OF APPLICATION DATA ● Backup SD card data regularly. ● Do not remove power or reset the controller, and do not insert or remove the SD card while it is being accessed. **Failure to follow these instructions can result in equipment damage.**

This figure shows the SD card slot:



It is possible to set the Write-Control Tab to prevent write operations to the SD card. Push the tab up, as shown in the example on the right-hand side, to release the lock and enable writing to the SD card. Before using an SD card, read the manufacturer's instructions.



Step	Action
1	Insert the SD card into the SD card slot: 
2	Push until you hear it “click”: 

SD Card Slot Characteristics

Topic	Characteristics	Description
Supported type	Standard Capacity	SD (SDSC)
	High Capacity	SDHC
Global memory	Size	16 GB max.

TMASD1 Characteristics

Characteristics	Description
Card removal durability	Minimum 1000 times
File retention time	10 years @ 25 °C (77 °F)
Flash type	SLC NAND
Memory size	256 MB
Ambient operation temperature	–10 … +85°C (14...185 °F)
Storage temperature	–25 … +85°C (–13...185 °F)
Relative humidity	95% max. non-condensing
Write/Erase cycles	3,000,000 (approximately)

NOTE: The TMASD1 has been rigorously tested in association with the logic controller. For other commercially available cards, consult your local sales representative.

NOTE: The SD card can be used directly on your PC.

Chapter 3
M251 Installation

Overview

This chapter provides installation safety guidelines, device dimensions, mounting instructions, and environmental specifications.

What Is in This Chapter?

This chapter contains the following sections:

Section 3.1
M251 Logic Controller General Rules for Implementing

What Is in This Section?

This section contains the following topics:

Environmental Characteristics

Enclosure Requirements

M251 Logic Controller system components are designed as Zone B, Class A industrial equipment according to IEC/CISPR Publication 11. If they are used in environments other than those described in the standard, or in environments that do not meet the specifications in this manual, the ability to meet electromagnetic compatibility requirements in the presence of conducted and/or radiated interference may be reduced.

All M251 Logic Controller system components meet European Community (CE) requirements for open equipment as defined by IEC/EN 61131-2. You must install them in an enclosure designed for the specific environmental conditions and to minimize the possibility of unintended contact with hazardous voltages. Use metal enclosures to improve the electromagnetic immunity of your M251 Logic Controller system. Use enclosures with a keyed locking mechanism to minimize unauthorized access.

Environmental Characteristics

All the M251 Logic Controller module components are electrically isolated between the internal electronic circuit and the input/output channels within the limits set forth and described by these environmental characteristics. For more information on electrical isolation, see the technical specifications of your particular controller found later in the current document. This equipment meets CE requirements as indicated in the table below. This equipment is intended for use in a Pollution Degree 2 industrial environment.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

The following table shows the general environmental characteristics:

Characteristic	Minimum Specification	Tested Range	
Standard compliance	IEC/EN 61131-2 IEC/EN 61010-2-201	–	
Ambient operating temperature	–	Horizontal installation	–10...55 °C (14...131 °F)
	–	Vertical installation	–10...35 °C (14...95 °F)
Storage temperature	–	–25...70 °C (- 13...158 °F)	
Relative humidity	–	Transport and storage	10...95 % (non-condensing)
		Operation	10...95 % (non-condensing)
Degree of pollution	IEC/EN 60664-1	2	
Degree of protection	IEC/EN 61131-2	IP20 with protective covers in place	
Corrosion immunity	–	Atmosphere free from corrosive gases	
Operating altitude	–	0...2000 m (0...6560 ft)	
Storage altitude	–	0...3000 m (0...9843 ft)	
Vibration resistance	IEC/EN 61131-2	Panel mounting or mounted on a top hat section rail (DIN rail)	3.5 mm (0.13 in) fixed amplitude from 5...8.4 Hz 9.8 m/s^2 (32.15 ft/s^2) (1 g_n) fixed acceleration from 8.4...150 Hz 10 mm (0.39 in) fixed amplitude from 5...8.7 Hz 29.4 m/s^2 (96.45 ft/s^2) (3 g_n) fixed acceleration from 8.7...150 Hz
Mechanical shock resistance	–	147 m/s^2 or 482.28 ft/s^2 (15 g_n) for a duration of 11 ms	

NOTE: The tested ranges may indicate values beyond that of the IEC Standard. However, our internal standards define what is necessary for industrial environments. In all cases, we uphold the minimum specification if indicated.

Electromagnetic Susceptibility

The M251 Logic Controller system meets electromagnetic susceptibility specifications as indicated in the following table:

Characteristic	Minimum Specification	Tested Range		
Electrostatic discharge	IEC/EN 61000-4-2	8 kV (air discharge) 4 kV (contact discharge)		
Radiated electromagnetic field	IEC/EN 61000-4-3	10 V/m (80...1000 MHz) 3 V/m (1.4...2 GHz) 1 V/m (2...3 GHz)		
Fast transients burst	IEC/EN 61000-4-4	24 Vdc main power lines	2 kV (CM[1] and DM[2])	
		24 Vdc I/Os	2 kV (clamp)	
		Relay output	1 kV (clamp)	
		Digital I/Os	1 kV (clamp)	
		Communication line	1 kV (clamp)	
Surge immunity	IEC/EN 61000-4-5 IEC/EN 61131-2	–	CM[1]	DM[2]
		DC Power lines	0.5 kV	0.5 kV
		Relay Outputs	–	–
		24 Vdc I/Os	–	–
		Shielded cable (between shield and ground)	1 kV	–
Induced electromagnetic field	IEC/EN 61000-4-6	10 Vrms (0.15...80 MHz)		
Conducted emission	IEC 61000-6-4	• 10...150 kHz: 120...69 dBµV/m QP • 150...1500 kHz: 79...63 dBµV/m QP • 1.5...30 MHz: 63 dBµV/m QP		
Radiated emission	IEC 61000-6-4	30...230 MHz: 40 dBµV/m QP 230...1000 MHz: 47 dBµV/m QP		

1 Common Mode
2 Differential Mode

NOTE: The tested ranges may indicate values beyond that of the IEC Standard. However, our internal standards define what is necessary for industrial environments. In all cases, we uphold the minimum specification if indicated.

Certifications and Standards

Introduction

The M251 Logic Controllers are designed to conform to the main national and international standards concerning electronic industrial control devices:

- IEC/EN 61131-2
- UL 508
- CSA 22.2 n° 142
- CSA E61131-2

The M251 Logic Controllers have obtained the following conformity marks:

- CE
- cULus
- CSA

For product compliance and environmental information (RoHS, REACH, PEP, EOLI, etc.), go to *www.schneider-electric.com/green-premium*.

Section 3.2
M251 Logic Controller Installation

What Is in This Section?

This section contains the following topics:

Installation and Maintenance Requirements

Before Starting

Read and understand this chapter before beginning the installation of your system.

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

Pay particular attention in conforming to any safety information, different electrical requirements, and normative standards that would apply to your machine or process in the use of this equipment.

Disconnecting Power

All options and modules should be assembled and installed before installing the control system on a mounting rail, onto a mounting plate or in a panel. Remove the control system from its mounting rail, mounting plate or panel before disassembling the equipment.

⚠ ⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

Programming Considerations

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Operating Environment

In addition to the **Environmental Characteristics**, refer to **Product Related Information** in the beginning of the present document for important information regarding installation in hazardous locations for this specific equipment.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Install and operate this equipment according to the conditions described in the Environmental Characteristics.

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

Installation Considerations

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use appropriate safety interlocks where personnel and/or equipment hazards exist.
- Install and operate this equipment in an enclosure appropriately rated for its intended environment and secured by a keyed or tooled locking mechanism.
- Use the sensor and actuator power supplies only for supplying power to the sensors or actuators connected to the module.
- Power line and output circuits must be wired and fused in compliance with local and national regulatory requirements for the rated current and voltage of the particular equipment.
- Do not use this equipment in safety-critical machine functions unless the equipment is otherwise designated as functional safety equipment and conforming to applicable regulations and standards.
- Do not disassemble, repair, or modify this equipment.
- Do not connect any wiring to reserved, unused connections, or to connections designated as No Connection (N.C.).

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

NOTE: JDYX2 or JDYX8 fuse types are UL-recognized and CSA approved.

M251 Logic Controller Mounting Positions and Clearances

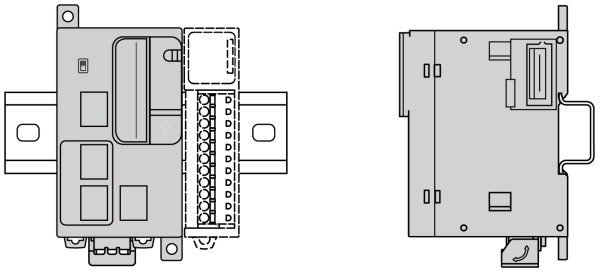
Introduction

This section describes the correct mounting positions for the M251 Logic Controller.

NOTE: Keep adequate spacing for proper ventilation and to maintain the operating temperature specified in the Environmental Characteristics *(see page 55)*.

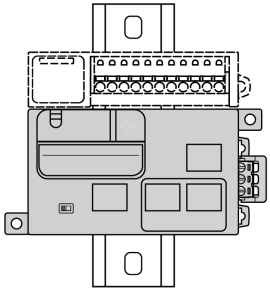
Correct Mounting Position

To obtain optimal operating characteristics, the M251 Logic Controller should be mounted horizontally on a vertical plane as shown in the figure below:



Acceptable Mounting Positions

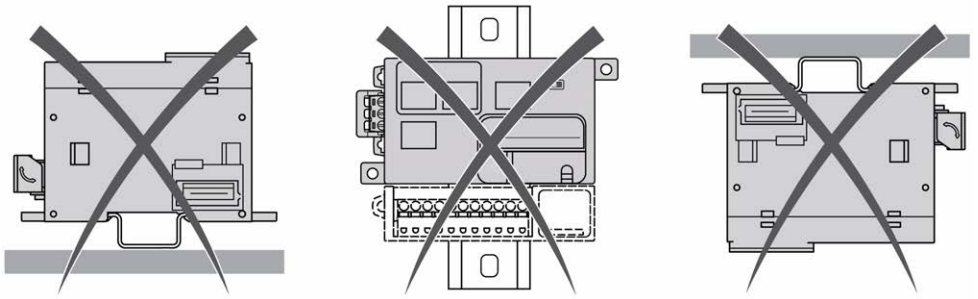
The M251 Logic Controller can also be mounted vertically on a vertical plane as shown below.



NOTE: Expansion modules must be mounted above the controller.

Incorrect Mounting Position

The M251 Logic Controller should only be positioned as shown in the Correct Mounting Position figure. The figures below show the incorrect mounting positions.



Minimum Clearances

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Place devices dissipating the most heat at the top of the cabinet and ensure adequate ventilation.
- Avoid placing this equipment next to or above devices that might cause overheating.
- Install the equipment in a location providing the minimum clearances from all adjacent structures and equipment as directed in this document.
- Install all equipment in accordance with the specifications in the related documentation.

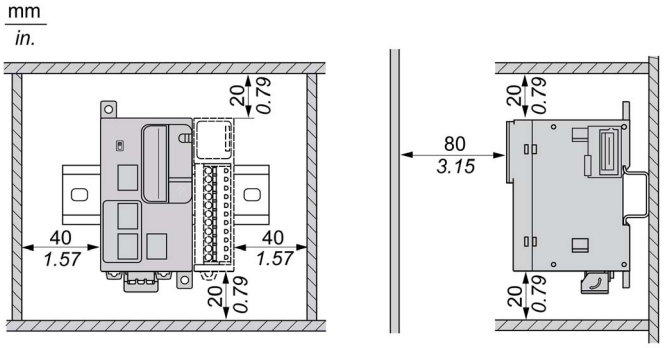
Failure to follow these instructions can result in death, serious injury, or equipment damage.

The M251 Logic Controller has been designed as an IP20 product and must be installed in an enclosure. Clearances must be respected when installing the product.

There are 3 types of clearances to consider:

- The M251 Logic Controller and all sides of the cabinet (including the panel door).
- The M251 Logic Controller terminal blocks and the wiring ducts to help reduce potential electromagnetic interference between the controller and the duct wiring.
- The M251 Logic Controller and other heat generating devices installed in the same cabinet.

The following figure shows the minimum clearances that apply to all M251 Logic Controller references:



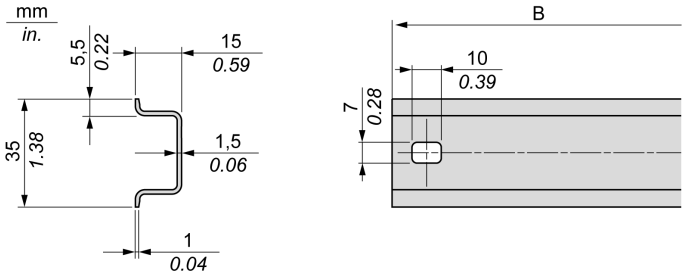
Top Hat Section Rail (DIN rail)

Dimensions of Top Hat Section Rail DIN Rail

You can mount the controller or receiver and their expansions on a 35 mm (1.38 in.) top hat section rail (DIN rail). The DIN rail can be attached to a smooth mounting surface or suspended from a EIA rack or mounted in a NEMA cabinet.

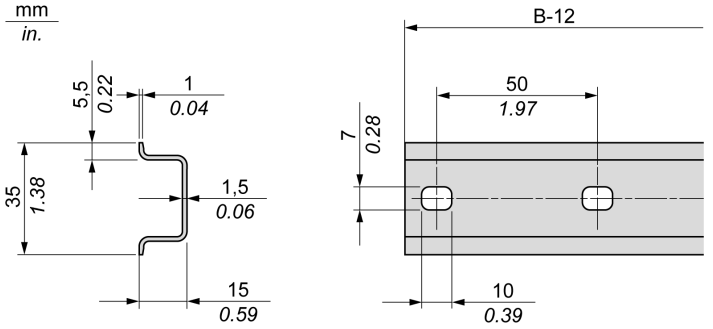
Symmetric Top Hat Section Rails (DIN Rail)

The following illustration and table indicate the references of the top hat section rails (DIN rail) for the wall-mounting range:



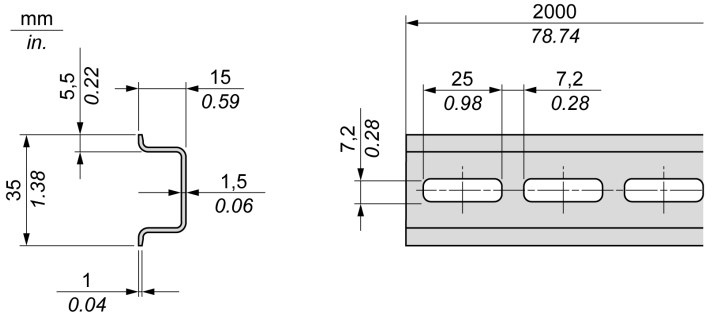
Reference	Type	Rail Length (B)
NSYSDR50A	A	450 mm (17.71 in.)
NSYSDR60A	A	550 mm (21.65 in.)
NSYSDR80A	A	750 mm (29.52 in.)
NSYSDR100A	A	950 mm (37.40 in.)

The following illustration and table indicate the references of the symmetric top hat section rails (DIN rail) for the metal enclosure range:



Reference	Type	Rail Length (B-12 mm)
NSYSDR60	A	588 mm (23.15 in.)
NSYSDR80	A	788 mm (31.02 in.)
NSYSDR100	A	988 mm (38.89 in.)
NSYSDR120	A	1188 mm (46.77 in.)

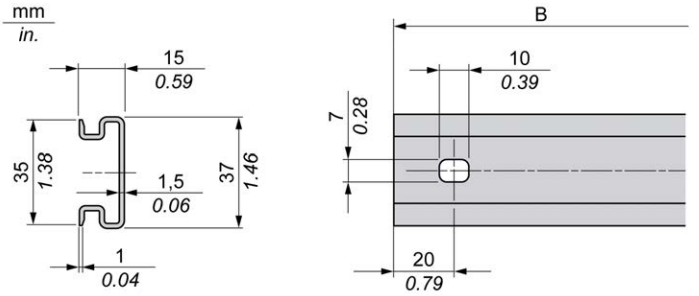
The following illustration and table indicate the references of the symmetric top hat section rails (DIN rail) of 2000 mm (78.74 in.):



Reference	Type	Rail Length
NSYSDR200[1]	A	2000 mm (78.74 in.)
NSYSDR200D[2]	A	
1 Unperforated galvanized steel **2** Perforated galvanized steel		

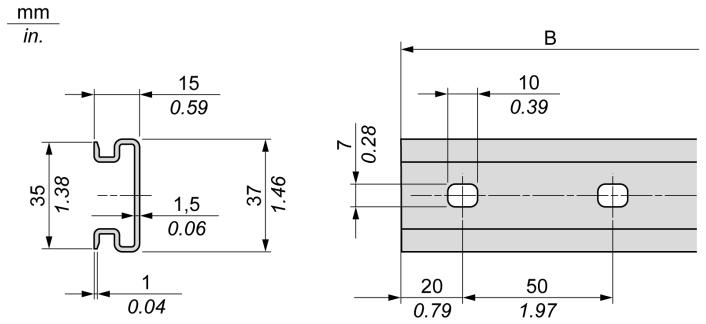
Double-Profile Top Hat Section Rails (DIN rail)

The following illustration and table indicate the references of the double-profile top hat section rails (DIN rails) for the wall-mounting range:



Reference	Type	Rail Length (B)
NSYDPR25	W	250 mm (9.84 in.)
NSYDPR35	W	350 mm (13.77 in.)
NSYDPR45	W	450 mm (17.71 in.)
NSYDPR55	W	550 mm (21.65 in.)
NSYDPR65	W	650 mm (25.60 in.)
NSYDPR75	W	750 mm (29.52 in.)

The following illustration and table indicate the references of the double-profile top hat section rails (DIN rail) for the floor-standing range:



Reference	Type	Rail Length (B)
NSYDPR60	F	588 mm (23.15 in.)
NSYDPR80	F	788 mm (31.02 in.)
NSYDPR100	F	988 mm (38.89 in.)
NSYDPR120	F	1188 mm (46.77 in.)

Installing and Removing the Controller with Expansions

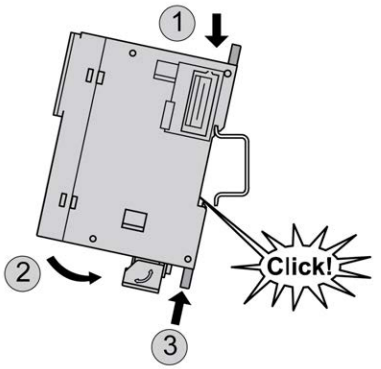
Overview

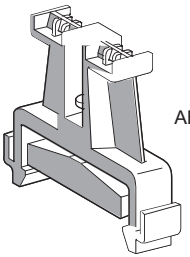
This section describes how to install and remove the controller with its expansion modules from a top hat section rail (DIN rail).

To assemble expansion modules to a controller or receiver module, or to other modules, refer to the respective expansion modules hardware guide(s).

Installing a Controller with its Expansions on a DIN Rail

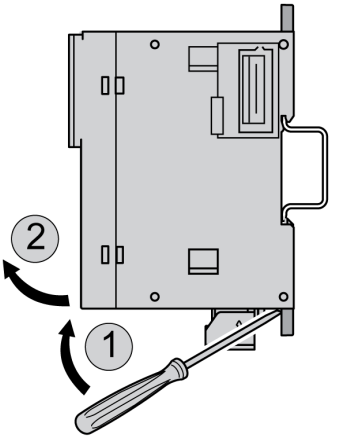
The following procedure describes how to install a controller with its expansion modules on a top hat section rail (DIN rail):

Step	Action
1	Fasten the top hat section rail (DIN rail) to a panel surface using screws.
2	Position the top groove of the controller and its expansion modules on the top edge of the DIN rail and press the assembly against the top hat section rail (DIN rail) until you hear the top hat section rail (DIN rail) clip snap into place. 

Step	Action
3	Place 2 terminal block end clamps on both sides of the controller and expansion module assembly.  AB1AB8P35 **NOTE:** Type ABB8P35 or equivalent terminal block end clamps help minimize sideways movement and improve the shock and vibration characteristics of the controller and expansion module assembly.

Removing a Controller with its Expansions from a Top Hat Section Rail (DIN Rail)

The following procedure describes how to remove a controller with its expansion modules from a top hat section rail (DIN rail):

Step	Action
1	Remove all power from your controller and expansion modules.
2	Insert a flat screwdriver into the slot of the top hat section rail (DIN rail) clip. 
3	Pull down the DIN rail clip.
4	Pull the controller and its expansion modules from the top hat section rail (DIN rail) from the bottom.

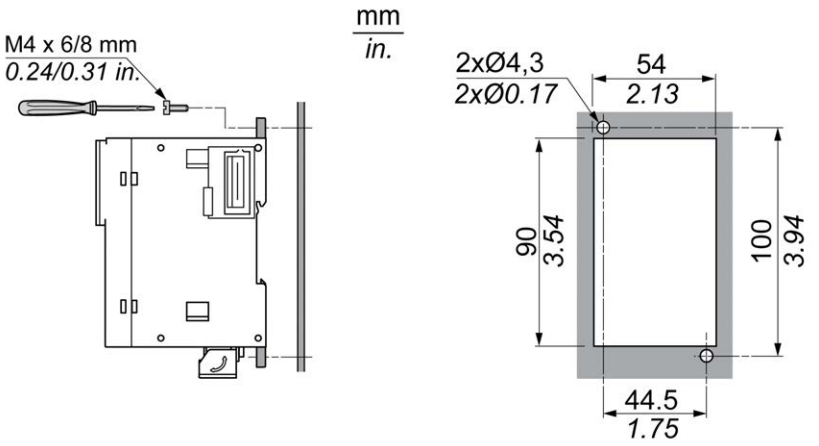
Direct Mounting on a Panel Surface

Overview

This section shows how to install M251 Logic Controller on a panel surface using the mounting holes.

Mounting Hole Layout

This diagram shows the mounting hole layout for M251 Logic Controller:



Section 3.3
M251 Electrical Requirements

What Is in This Section?

This section contains the following topics:

Wiring Best Practices

Overview

This section describes the wiring guidelines and associated best practices to be respected when using the M251 Logic Controller system.

⚠ ⚠ DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

- Disconnect all power from all equipment including connected devices prior to removing any covers or doors, or installing or removing any accessories, hardware, cables, or wires except under the specific conditions specified in the appropriate hardware guide for this equipment.
- Always use a properly rated voltage sensing device to confirm the power is off where and when indicated.
- Replace and secure all covers, accessories, hardware, cables, and wires and confirm that a proper ground connection exists before applying power to the unit.
- Use only the specified voltage when operating this equipment and any associated products.

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

LOSS OF CONTROL

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

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

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

Wiring Guidelines

This rules must be applied when wiring a M251 Logic Controller system:

- Communication wiring must be kept separate from the power wiring. Route these 2 types of wiring in separate cable ducting.
- Verify that the operating conditions and environment are within the specification values.
- Use proper wire sizes to meet voltage and current requirements.
- Use copper conductors (required).
- Use twisted pair, shielded cables for networks, and fieldbus.

Use shielded, properly grounded cables for all communication connections. If you do not use shielded cable for these connections, electromagnetic interference can cause signal degradation. Degraded signals can cause the controller or attached modules and equipment to perform in an unintended manner.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use shielded cables for all communication signals.
- Ground cable shields for all communication signals at a single point[1].
- Route communication separately from power cables.

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

[1]Multipoint grounding is permissible if connections are made to an equipotential ground plane dimensioned to help avoid cable shield damage in the event of power system short-circuit currents.

For more details, refer to Grounding Shielded Cables *(see page 83)*.

NOTE: Surface temperatures may exceed 60 °C (140 °F).
To conform to IEC 61010 standards, route primary wiring (wires connected to power mains) separately and apart from secondary wiring (extra low voltage wiring coming from intervening power sources). If that is not possible, double insulation is required such as conduit or cable gains.

Rules for Removable Screw Terminal Block

The following tables show the cable types and wire sizes for a **5.08 pitch** removable screw terminal block (power supply):

mm / in. 7 / 0.28								
mm^2	0.2…2.5	0.2…2.5	0.25…2.5	0.25…2.5	2 x 0.2…1	2 x 0.2…1.5	2 x 0.25…1	2 x 0.5...1.5
AWG	*24…14*	*24…14*	*23...14*	*23...14*	*2 x 24…17*	*2 x 24…16*	*2 x 23…17*	*2 x 20...16*

Ø 3,5 mm (*0.14 in.*)	C	N•m	0.5...0.6
		lb-in	*4.42...5.31*

The use of copper conductors is required.

DANGER

LOOSE WIRING CAUSES ELECTRIC SHOCK

Tighten connections in conformance with the torque specifications.

Failure to follow these instructions will result in death or serious injury.

DANGER

FIRE HAZARD

Use only the correct wire sizes for the maximum current capacity of the power supplies.

Failure to follow these instructions will result in death or serious injury.

DC Power Supply Characteristics and Wiring

Overview

This section provides the characteristics and the wiring diagrams of the DC power supply.

DC Power Supply Voltage Range

If the specified voltage range is not maintained, outputs may not switch as expected. Use appropriate safety interlocks and voltage monitoring circuits.

⚠ DANGER

FIRE HAZARD

- Use only the correct wire sizes for the maximum current capacity of the I/O channels and power supplies.
- For relay output (2 A) wiring, use conductors of at least 0.5 mm^2 (AWG 20) with a temperature rating of at least 80 °C (176 °F).
- For common conductors of relay output wiring (7 A), or relay output wiring greater than 2 A, use conductors of at least 1.0 mm^2 (AWG 16) with a temperature rating of at least 80 °C (176 °F).

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not exceed any of the rated values specified in the environmental and electrical characteristics tables.

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

DC Power Supply Requirements

The M251 Logic Controllerand associated I/O (TM2, TM3, with a nominal voltage of 24 Vdc. The 24 Vdc power supplies must be rated Safety Extra Low Voltage (SELV) or Protective Extra Low Voltage (PELV) according to IEC 61140. These power supplies are isolated between the electrical input and output circuits of the power supply.

⚠ WARNING

POTENTIAL OF OVERHEATING AND FIRE

- Do not connect the equipment directly to line voltage.
- Use only isolating PELV power supplies and circuits to supply power to the equipment[1].

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

[1] For compliance to UL (Underwriters Laboratories) requirements, the power supply must also conform to the various criteria of NEC Class 2, and be inherently current limited to a maximum power output availability of less than 100 VA (approximately 4 A at nominal voltage), or not inherently limited but with an additional protection device such as a circuit breaker or fuse meeting the requirements of clause 9.4 Limited-energy circuit of UL 61010-1. In all cases, the current limit should never exceed that of the electric characteristics and wiring diagrams for the equipment described in the present documentation. In all cases, the power supply must be grounded, and you must separate Class 2 circuits from other circuits. If the indicated rating of the electrical characteristics or wiring diagrams are greater than the specified current limit, multiple Class 2 power supplies may be used.

Controller DC Characteristics

This table shows the characteristics of the DC power supply required for the controller:

Characteristic		Value
Rated voltage		24 Vdc
Power supply voltage range		19.2...28.8 Vdc
Power interruption time		10 ms at 24 Vdc
Maximum inrush current		50 A
Power consumption		32.6 W, max. 40.4 W(1)
Isolation	between DC power supply and internal logic	Not isolated
	between DC power supply and protective earth ground (PE)	500 Vac
(1) Controller + 7 TM3 expansion modules		

Power Interruption

The duration of power interruptions where the M251 Logic Controller is able to continue normal operation varies depending upon the load to the power supply of the controller, but a minimum of 10 ms is maintained as specified by IEC standards.

When planning the management of the power supplied to the controller, you must consider the power interruption duration due to the fast cycle time of the controller.

There could potentially be many scans of the logic and consequential updates to the I/O image table during the power interruption, while there is no external power supplied to the inputs, the outputs or both depending on the power system architecture and power interruption circumstances.

⚠ WARNING

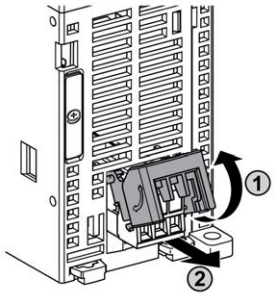
UNINTENDED EQUIPMENT OPERATION

- Individually monitor each source of power used in the controller system including input power supplies, output power supplies and the power supply to the controller to allow appropriate system shutdown during power system interruptions.
- The inputs monitoring each of the power supply sources must be unfiltered inputs.

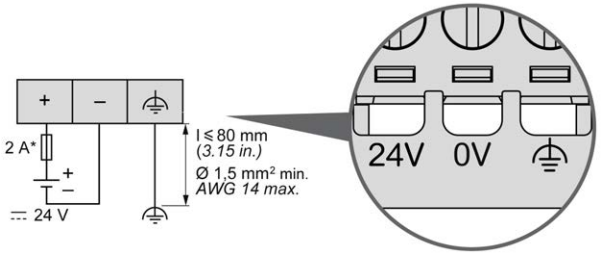
Failure to follow these instructions can result in death, serious injury, or equipment damage.

DC Power Supply Wiring Diagram

This figure shows the power supply terminal block removal procedure:



The following figure shows the wiring of the DC power supply:



* Type T fuse

For more information, refer to the 5.08 pitch Rules for Removable Screw Terminal block *(see page 77)*.

Grounding the M251 System

Overview

To help minimize the effects of electromagnetic interference, cables carrying fieldbus communication signals must be shielded.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Use shielded cables for communication signals.
- Ground cable shields for communication signals at a single point [1].
- Always comply with local wiring requirements regarding grounding of cable shields.

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

[1]Multipoint grounding is permissible if connections are made to an equipotential ground plane dimensioned to help avoid cable shield damage in the event of power system short-circuit currents.

The use of shielded cables requires compliance with the following wiring rules:

- For protective ground connections (PE), metal conduit or ducting can be used for part of the shielding length, provided there is no break in the continuity of the ground connections. For functional ground (FE), the shielding is intended to attenuate electromagnetic interference and the shielding must be continuous for the length of the cable. If the purpose is both functional and protective, as is often the case for communication cables, the cable must have continuous shielding.
- Wherever possible, keep cables carrying one type of signal separate from the cables carrying other types of signals or power.

Protective Ground (PE) on the Backplane

The protective ground (PE) should be connected to the conductive backplane by a heavy-duty wire, usually a braided copper cable with the maximum allowable cable section.

Shielded Cables Connections

Cables carrying fieldbus communication signals must be shielded. The shielding must be securely connected to ground. The fieldbus communication cable shields must be connected to the protective ground (PE) with a connecting clamp secured to the conductive backplane of your installation.

The shielding of the Modbus cable must be connected to the protective ground (PE).

⚠⚠ DANGER

HAZARD OF ELECTRIC SHOCK

- The grounding terminal connection (PE) must be used to provide a protective ground at all times.
- Make sure that an appropriate, braided ground cable is attached to the PE/PG ground terminal before connecting or disconnecting the network cable to the equipment.

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

ACCIDENTAL DISCONNECTION FROM PROTECTIVE GROUND (PE)

- Do not use the TM2XMTGB Grounding Plate to provide a protective ground (PE).
- Use the TM2XMTGB Grounding Plate only to provide a functional ground (FE).

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

Part II
Modicon M251 Logic Controller

What Is in This Part?

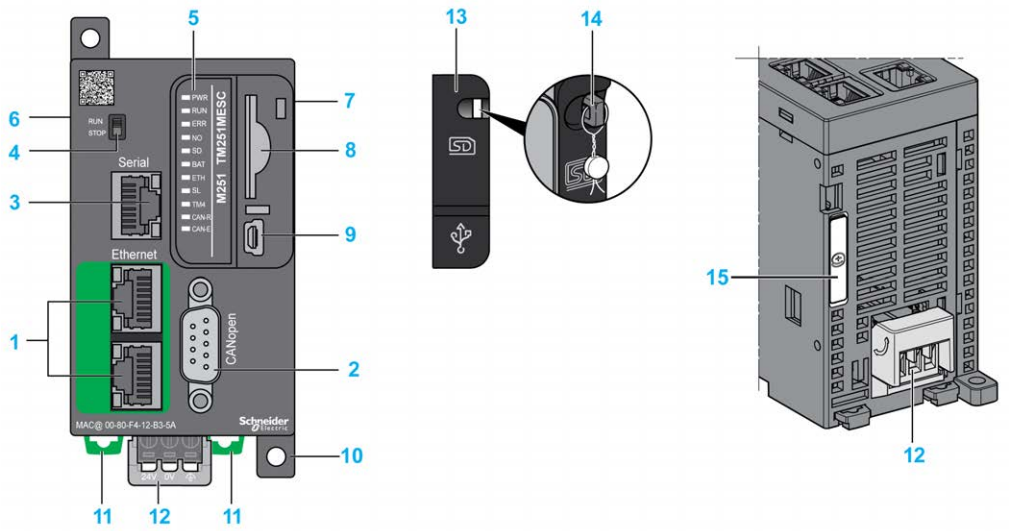
This part contains the following chapters:

Chapter 4
TM251MESC

TM251MESC Presentation

Description

This figure shows the different components of the TM251MESC logic controller:

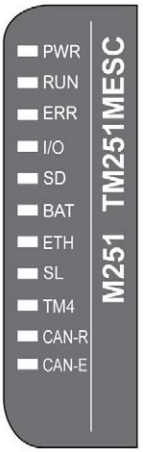


N°	Description	Refer to
1	Dual port Ethernet switch	Ethernet port *(see page 101)*
2	CANopen port	CANopen port
3	Serial line port / Type RJ45 (RS-232 or RS-485)	Serial Line *(see page 106)*
4	Run/Stop switch	Run/Stop *(see page 48)*
5	Status LEDs	–
6	TM4 bus connector	TM4 Expansion Modules *(see page 37)*
7	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 27)*
8	SD card slot	SD Card *(see page 49)*
9	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 105)*

N°	Description	Refer to
10	Surface mounting lugs	–
11	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 66)*
12	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 78)*
13	Protective cover (SD card slot and USB mini-B programming port)	–
14	Locking hook (Hook not included)	–
15	Battery holder	Real Time Clock (RTC) *(see page 44)*

Status LEDs

This figure shows the status LEDs:



The following table describes the system status LEDs:

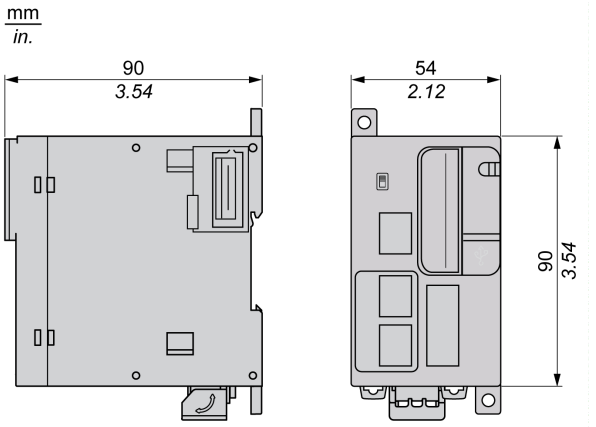
Label	Function Type	Color	Status	Description
PWR	Power	Green	On	Indicates that power is applied.
			Off	Indicates that power is removed.
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.
			Flashing	Indicates that the controller has a valid application that is stopped.
			1 flash	Indicates that the controller has paused at BREAKPOINT.
			Off	Indicates that the controller is not programmed

Label	Function Type	Color	Status	Description
ERR	Internal Error	Red	On	Indicates that an operating system error has been detected
			Fast flashing	Indicates that the controller has detected an internal error
			Slow flashing	Indicates either that a minor error has been detected if **RUN** is ON or that no application has been detected
I/O	I/O error	Red	On	Indicates device errors on the serial line, SD card, TM4 bus, TM3 bus, Ethernet port(s) or CANopen port.
SD	SD card access	Green	On	Indicates that the SD card is being accessed
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.
			Flashing	Indicates that the battery charge is low.
ETH	Ethernet port status	Green	On	Indicates that the ethernet port is connected and the IP address is defined.
			3 flashes	Indicates that the ethernet port is not connected.
			4 flashes	Indicates that the IP address is already in used.
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.
			6 flashes	Indicates that the configured IP address is not valid.
SL	Serial line	Green	On	Indicates the status of serial line *(see page 108)*
			Off	Indicates no serial communication
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus
			Off	Indicates that no error has been detected on the TM4 bus
CAN-R	CANopen running status	Green	On	Indicates that the CANopen bus is operational.
			Off	Indicates that the CANopen master is configured.
			Flashing	Indicates that the CANopen bus is being initialized.
			1 flash per second	Indicates that the CANopen bus is stopped.
CAN-E	CANopen error	Red	On	Indicates that the CANopen bus is stopped (BUS OFF).
			Off	Indicates no CANopen detected error.
			Flashing	Indicates that the CANopen bus is not valid.
			1 flash per second	Indicates that the controller has detected that the maximum number of error frames has been reached or exceeded.
			2 flashes per second	Indicates that the controller has detected either a Node Guarding or a Heartbeat event.

NOTE: All the LEDs flash when the logic controller is being identified. For more details, refer to the EcoStruxure Machine Expert Programming Guide.

Dimensions

This figures shows the external dimensions of the logic controller:

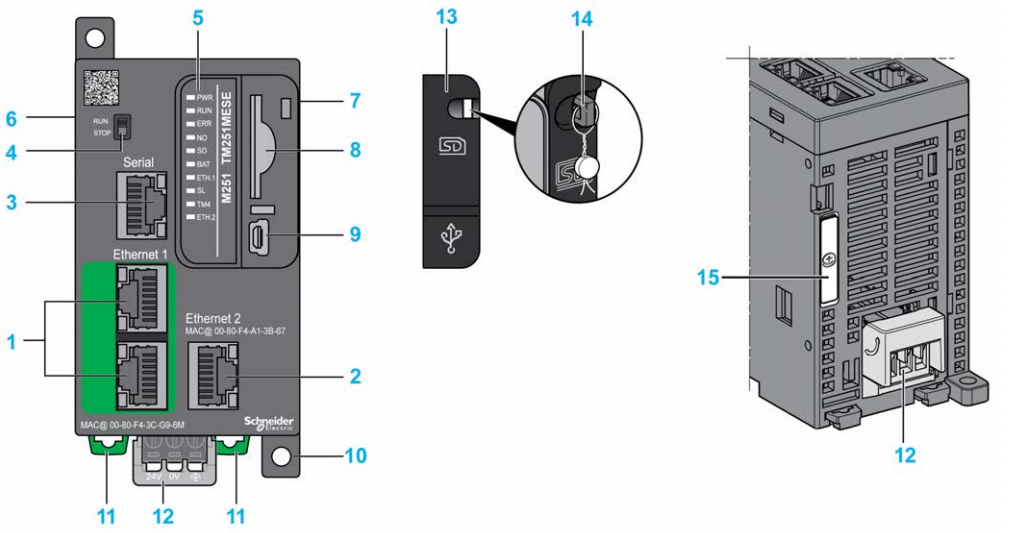


Chapter 5
TM251MESE

TM251MESE Presentation

Description

This figure shows the different components of the TM251MESE logic controller:

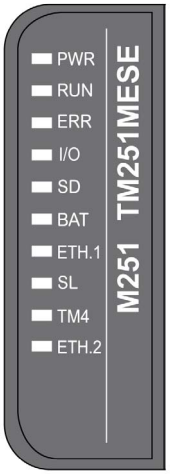


N°	Description	Refer to
1	Dual port Ethernet switch	Ethernet port *(see page 101)*
2	Ethernet port 2	Ethernet ports *(see page 103)*
3	Serial line port / Type RJ45 (RS-232 or RS-485)	Serial Line *(see page 106)*
4	Run/Stop switch	Run/Stop *(see page 48)*
5	Status LEDs	–
6	TM4 bus connector	TM4 Expansion Modules *(see page 37)*
7	TM3/TM2 bus connector	TM3 Expansion Modules *(see page 27)*
8	SD card slot	SD Card *(see page 49)*
9	USB mini-B programming port / For terminal connection to a programming PC (EcoStruxure Machine Expert)	USB Mini-B Programming Port *(see page 105)*

N°	Description	Refer to
10	Surface mounting lugs	–
11	Clip-on lock for 35 mm (1.38 in.) top hat section rail (DIN-rail)	Top Hat Section Rail *(see page 66)*
12	24 Vdc power supply	DC Power supply Characteristics and Wiring *(see page 78)*
13	Protective cover (SD card slot and USB mini-B programming port)	–
14	Locking hook (Hook not included)	–
15	Battery holder	Real Time Clock (RTC) *(see page 44)*

Status LEDs

This figure shows the status LEDs:



The following table describes the system status LEDs:

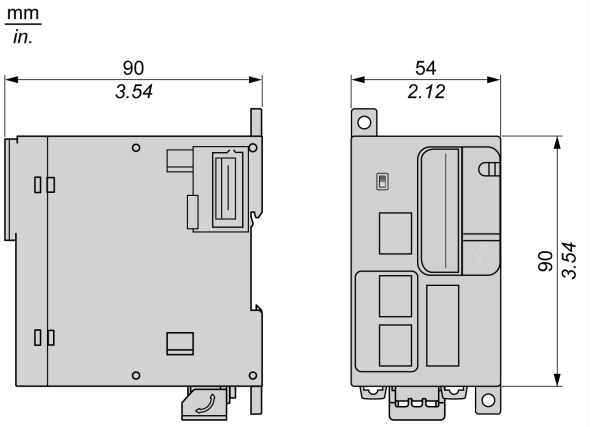
Label	Function Type	Color	Status	Description
PWR	Power	Green	On	Indicates that power is applied.
			Off	Indicates that power is removed.
RUN	Machine status	Green	On	Indicates that the controller is running a valid application.
			Flashing	Indicates that the controller has a valid application that is stopped.
			1 flash	Indicates that the controller has paused at BREAKPOINT.
			Off	Indicates that the controller is not programmed

Label	Function Type	Color	Status	Description
ERR	Internal Error	Red	On	Indicates that an operating system error has been detected
			Fast flashing	Indicates that the controller has detected an internal error
			Slow flashing	Indicates either that a minor error has been detected if **RUN** is ON or that no application has been detected
I/O	I/O error	Red	On	Indicates device errors on the serial line, SD card, TM4 bus, TM3 bus, Ethernet port(s) or CANopen port.
SD	SD card access	Green	On	Indicates that the SD card is being accessed
BAT	Battery	Red	On	Indicates that the battery needs to be replaced.
			Flashing	Indicates that the battery charge is low.
ETH.1 ETH.2	Ethernet port status	Green	On	Indicates that the Ethernet port is connected and the IP address is defined.
			3 flashes	Indicates that the Ethernet port is not connected.
			4 flashes	Indicates that the IP address is already in use.
			5 flashes	Indicates that the module is waiting for BOOTP or DHCP sequence.
			6 flashes	Indicates that the configured IP address is not valid.
SL	Serial line	Green	On	Indicates the status of serial line *(see page 108)*
			Off	Indicates no serial communication
TM4	Error on TM4 bus	Red	On	Indicates that an error has been detected on the TM4 bus
			Off	Indicates that no error has been detected on the TM4 bus

NOTE: All the LEDs flash when the logic controller is being identified. For more details, refer to the EcoStruxure Machine Expert Programming Guide.

Dimensions

This figure shows the external dimensions of the logic controller:



Part III
Modicon M251 Logic Controller Communication

What Is in This Part?

This part contains the following chapters:

Chapter 6
Integrated Communication Ports

What Is in This Chapter?

This chapter contains the following topics:

CAN Port

CANopen Capabilities

The Modicon M251 Logic Controller CANopen master has the following features:

Feature	Description
Maximum number of slaves on the bus	63 CANopen slave devices
Maximum length of CANopen fieldbus cables	According to the CAN specification (see Transmission Speed and Cable Length *(see page 100)*).
Maximum number of PDOs managed by the master	252 TPDOs + 252 RPDOs

For each additional CANopen slave:

- the application size increases by an average of 10 kbytes, which conceivably could result in exceeding memory limits.
- the configuration initialization time at the startup increases, which conceivably could result in watchdog timeout.

Although EcoStruxure Machine Expert does not restrict you from doing so, do not exceed more than 63 CANopen slave modules (and/or 252 TPDOs and 252 RPDOs) in order to have a sufficient performance tolerance and avoid any performance degradation.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not connect more than 63 CANopen slave devices to the controller to avoid system overload watchdog condition.

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

NOTICE

DEGRADATION OF PERFORMANCE

Do not exceed more than 252 TPDOs and 252 RPDOs for the Modicon M251 Logic Controller.

Failure to follow these instructions can result in equipment damage.

J1939 Capabilities

The Modicon M251 Logic Controller J1939 master has the following features:

Feature	Description
Maximum number of ECUs (slaves) on the bus	Limited only by the address range of 0...253 for Electronic Control Units (ECUs).
Maximum length of J1939 fieldbus cables	According to the CAN specification (see Transmission Speed and Cable Length *(see page 100)*). For J1939, the CAN bus must be configured to run at 250 Kbps.
Maximum number of PGNs managed by the master	Given implicitly by the maximum number of input bits (%I) and output bits (%Q) available on the Modicon M251 Logic Controller: 4096 input bits and 4096 output bits. This results in a maximum of 512 single-packet PGNs (most PGNs are single-packet, containing 8 bytes of data).

For each additional ECU with approximately 10 configured (single frame) Parameter Group Numbers (PGNs):

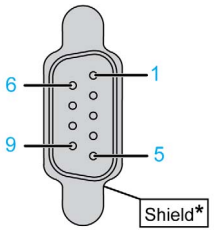
- the application size increases by an average of 15 Kbytes. This figure includes the memory consumed by implicitly-generated variables for configured Suspected Parameter Numbers (SPNs). This application size increase could result in exceeding memory limits.
- the number of input bits (%I) used on the logic controller increases in proportion to the number and size of PGNs configured as "TX Signals" in a non-local ECU or "RX Signals" in a local ECU.
- the number of output bits (%Q) used on the logic controller increases in proportion to the number and size of PGNs configured as "TX Signals" in a local ECU.

NOTE: Thoroughly test your application regarding the number of configured J1939 ECUs connected to the controller, and the number of PGNs configured on each ECU, to avoid a system overload watchdog condition or performance degradation.

For more information, refer to J1939 Interface Configuration *(see Modicon M251 Logic Controller, Programming Guide)*.

CAN Wiring Diagram

The CAN plug is a male sub-D9 terminal block:



* To be connected externally to the protective earth

Pin	Signal	Description
1	–	Reserved
2	CAN_L	CAN_L bus line
3	CAN_GND	CAN ground
4	–	Reserved
5	(CAN_SHLD)	Optional CAN shield
6	GND	Ground
7	CAN_H	CAN_H bus line
8	–	Reserved
9	(CAN_V+)	Optional CAN external positive supply

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”.

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

Transmission Speed and Cable Length

Transmission speed is limited by the bus length and the type of cable used.

The following table describes the relationship between the maximum transmission speed and the bus length (on a single CAN segment without a repeater):

Maximum transmission baud rate	Bus length
1000 Kbps	20 m (65 ft)
800 Kbps	40 m (131 ft)
500 Kbps	100 m (328 ft)
250 Kbps	250 m (820 ft)
125 Kbps	500 m (1,640 ft)
50 Kbps	1000 m (3280 ft)
20 Kbps	2500 m (16,400 ft)

NOTE: The CAN cable must be shielded.

Ethernet Port

Overview

The M251 Logic Controller is equipped with Ethernet communications ports:

Reference	Number of Ports	Port Name
TM251MESC	2 (one dual Ethernet port switch)	Ethernet
TM251MESE	2 (one dual Ethernet port switch)	Ethernet 1
	1	Ethernet 2

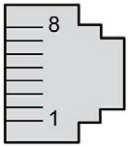
Characteristics

This table describes the different Ethernet characteristics:

Characteristic	Description
Function	Modbus TCP/IP, Machine Expert protocol, EtherNet I/P
Connector type	RJ45
Auto negotiation	from 10 M half duplex to 100 M full duplex
Cable type	Shielded
Automatic cross-over detection	Yes

Pin Assignment

This figure shows the RJ45 Ethernet connector pin assignment:



This table describes the RJ45 Ethernet connector pins:

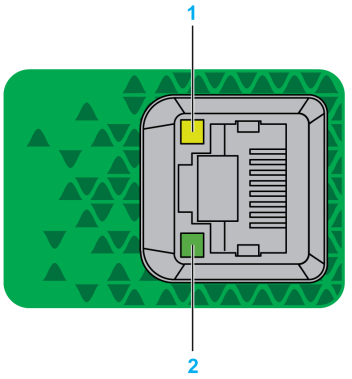
Pin N°	Signal
1	TD+
2	TD-
3	RD+
4	-
5	-
6	RD-
7	-
8	-

NOTE: The controller supports the MDI/MDIX auto-crossover cable function. It is not necessary to use special Ethernet crossover cables to connect devices directly to this port (connections without an Ethernet hub or switch).

NOTE: Ethernet cable disconnection is detected every second. In case of disconnection of a short duration (< 1 second), the network status may not indicate the disconnection.

Status LED

This figure shows RJ45 connectors status LED:



This table describes the Ethernet status LEDs:

Label	Description	LED		
		Color	Status	Description
1	Ethernet link	Green/Yellow	Off	No link
			Solid yellow	Link at 10 Mbit/s
			Solid green	Activity at 100 Mbit/s
2	Ethernet activity	Green	Off	No activity
			On	Transmitting or receiving data

TM251MESE Specific Considerations

Ethernet Ports

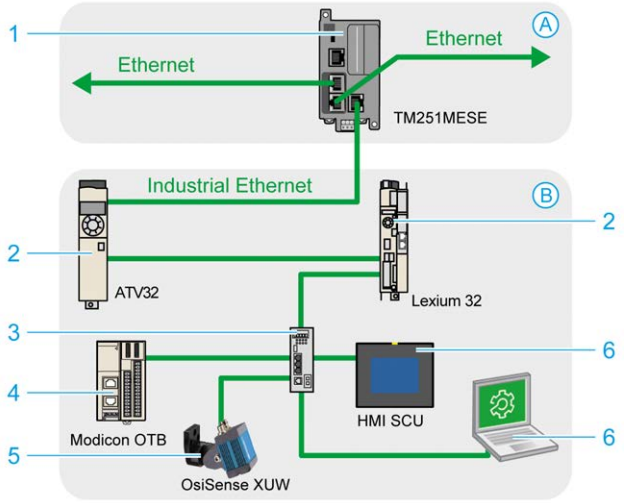
The TM251MESE has two different Ethernet networks. Each has its own unique IP and MAC addresses.

The two Ethernet networks are called Ethernet 1 and Ethernet 2:

- Ethernet 1 is made of two switched Ethernet ports dedicated to communication between machines or with the control network.
- Ethernet 2 is made of one Ethernet port dedicated to the device network and supporting industrial Ethernet connections.

Industrial Ethernet Architecture

This figure presents a typical industrial Ethernet architecture:



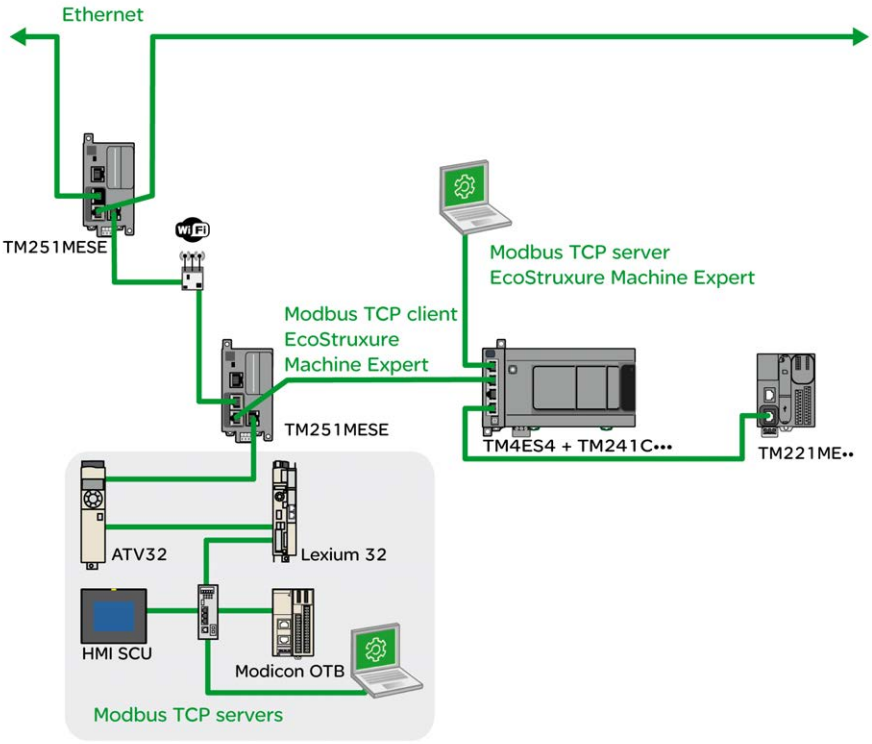
A Control network
B Device network
1 Logic controller *(see EcoStruxure Machine Expert Industrial Ethernet, User Guide)*
2 Daisy-chained slaves
3 Ethernet switch
4 I/O island (Modbus TCP)
5 Vision sensor (EtherNet/IP)
6 PC and HMI (TCP/UDP)
2, 4, and 5 Industrial Ethernet slave devices (EtherNet/IP /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

Industrial Ethernet Connections with Modbus TCP IOScanner Architecture

For example, you can:

- Connect your PC to Ethernet 1.
- Use a Modbus TCP IOScanner or EtherNet/IP Scanner with the Ethernet 2.

This figure is an example of an industrial Ethernet architecture with the TM251MESE.



USB Mini-B Programming Port

Overview

The USB Mini-B Port is the programming port you can use to connect a PC with a USB host port using EcoStruxure Machine Expert software. Using a typical USB cable, this connection is suitable for quick updates of the program or short duration connections to perform maintenance and inspect data values. It is not suitable for long-term connections such as commissioning or monitoring without the use of specially adapted cables to help minimize electromagnetic interference.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- You must use a shielded USB cable such as a BMX XCAUSBH0•• secured to the functional ground (FE) of the system for any long-term connection.
- Do not connect more than one controller or bus coupler at a time using USB connections.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

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

Characteristics

This table describes the characteristics of the USB Mini-B programming port:

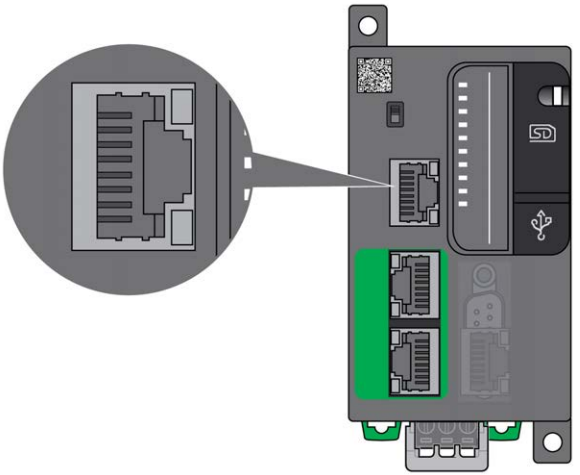
Parameter	USB Programming Port
Function	Compatible with USB 2.0
Connector type	Mini-B
Isolation	None
Cable type	Shielded

Serial Line

Overview

The serial line:

- Can be used to communicate with devices supporting the Modbus protocol as either master or slave, ASCII protocol (printer, modem...) and Machine Expert Protocol (HMI,...).
- provides a 5 Vdc power distribution.



Characteristics

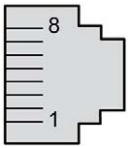
Characteristic		Description
Function		RS485 or RS232 software configured
Connector type		RJ45
Isolation		Non-isolated
Maximum baud rate		1200 up to 115 200 bps
Cable	Type	Shielded
	Maximum length (between the controller and an isolated junction box)	15 m (49 ft) for RS485 3 m (9.84 ft) for RS232
Polarization		Software configuration is used to connect when the node is configured as a master. 560 Ω resistors are optional.
5 Vdc power supply for RS485		Yes

NOTE: Some devices provide voltage on RS485 serial connections. Do not connect these voltage lines to your controller as they may damage the controller serial port electronics and render the serial port inoperable.

NOTICE
INOPERABLE EQUIPMENT Use only the VW3A8306R•• serial cable to connect RS485 devices to your controller. **Failure to follow these instructions can result in equipment damage.**

Pin Assignment

The following figure shows the pins of the RJ45 connector:



This table describes the pin assignment of the RJ45 connector:

Pin	RS232	RS485
1	RxD	N.C.
2	TxD	N.C.
3	N.C.	N.C.
4	N.C.	D1
5	N.C.	D0
6	N.C.	N.C.
7	N.C. *	5 Vdc
8	Common	Common

*: 5 Vdc delivered by the controller, do not connect.

N.C.: No connection

RxD: Received data

TxD: Transmitted data

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not connect wires to unused terminals and/or terminals indicated as “No Connection (N.C.)”. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Status LED

This table describes the serial line status LED:

Label	Description	LED		
		Color	Status	Description
SL	Serial line	Green	On	Indicates the activity of the serial line.
			Off	Indicates no serial communication.

Chapter 7
Connecting the M251 Logic Controller to a PC

Connecting the Controller to a PC

Overview

To transfer, run, and monitor the applications, connect the controller to a computer, that has EcoStruxure Machine Expert installed, using either a USB cable or an Ethernet connection (for those references that support an Ethernet port).

NOTICE
INOPERABLE EQUIPMENT Always connect the communication cable to the PC before connecting it to the controller. **Failure to follow these instructions can result in equipment damage.**

USB Powered Download

In order to execute limited operations, the M251 Logic Controller has the capability to be powered through the USB Mini-B port. A diode mechanism avoids having the logic controller both powered by USB and by the normal power supply, or to supply voltage on the USB port.

When powered only by USB, the logic controller executes the firmware and the boot project (if any) and the I/O board is not powered during boot (same duration as a normal boot). USB powered download initializes the internal flash memory with some firmware or some application and parameters when the controller is powered by USB. The preferred tool to connect to the controller is the **Controller Assistant**. Refer to the *EcoStruxure Machine Expert Controller Assistant User Guide*.

The controller packaging allows easy access to USB Mini-B port with minimum opening of the packaging. You can connect the controller to the PC with a USB cable. Long cables are not suitable for the USB powered download.

⚠ WARNING
INSUFFICENT POWER FOR USB DOWNLOAD Do not use a USB cable longer than 3m (9.8 ft) for USB powered download. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: It is not intended that you use the USB Powered Download on an installed controller. Depending on the number of I/O expansion modules in the physical configuration of the installed controller, there may be insufficient power from your PC USB port to accomplish the download.

USB Mini-B Port Connection

TCSXCNAMUM3P: This USB cable is suitable for short duration connections such as quick updates or retrieving data values.

BMXXCAUSBH018: Grounded and shielded, this USB cable is suitable for long duration connections.

NOTE: You can only connect 1 controller or any other device associated with EcoStruxure Machine Expert and its component to the PC at any one time.

The USB Mini-B Port is the programming port you can use to connect a PC with a USB host port using EcoStruxure Machine Expert software. Using a typical USB cable, this connection is suitable for quick updates of the program or short duration connections to perform maintenance and inspect data values. It is not suitable for long-term connections such as commissioning or monitoring without the use of specially adapted cables to help minimize electromagnetic interference.

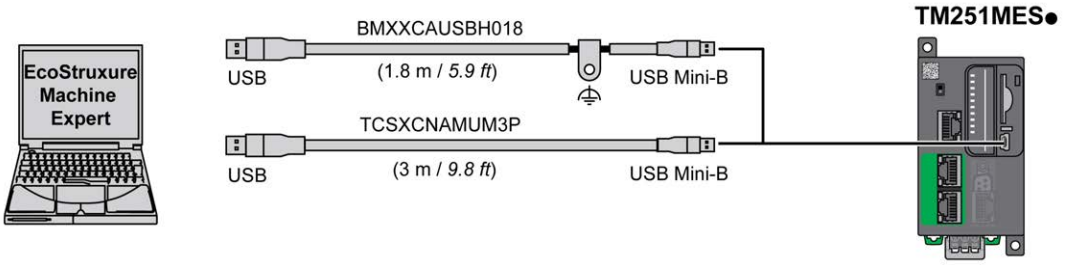
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION OR INOPERABLE EQUIPMENT

- You must use a shielded USB cable such as a BMX XCAUSBH0•• secured to the functional ground (FE) of the system for any long-term connection.
- Do not connect more than one controller or bus coupler at a time using USB connections.
- Do not use the USB port(s), if so equipped, unless the location is known to be non-hazardous.

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

The communication cable should be connected to the PC first to minimize the possibility of electrostatic discharge affecting the controller.

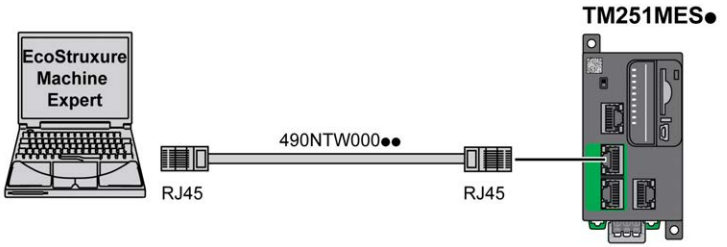


To connect the USB cable to your controller, follow the steps below:

Step	Action
1	**1a** If making a long-term connection using the cable BMXXCAUSBH018, or other cable with a ground shield connection, be sure to securely connect the shield connector to the functional ground (FE) or protective ground (PE) of your system before connecting the cable to your controller and your PC. **1b** If making a short-term connection using the cable TCSXCNAMUM3P or other non-grounded USB cable, proceed to step 2.
2	Connect your USB cable to the computer.
3	Open the hinged access cover.
4	Connect the Mini connector of your USB cable to the controller USB connector.

Ethernet Port Connection

You can also connect the controller to a PC using an Ethernet cable.



To connect the controller to the PC, do the following:

Step	Action
1	Connect the Ethernet cable to the PC.
2	Connect the Ethernet cable to either of the Ethernet 1 ports on the controller.

Glossary

A

application

A program including configuration data, symbols, and documentation.

ASCII

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

B

bps

(*bit per second*) A definition of transmission rate, also given in conjunction with multiplicator kilo (kbps) and mega (mbps).

C

CANopen

An open industry-standard communication protocol and device profile specification (EN 50325-4).

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

continuous function chart language

A graphical programming language (an extension of the IEC61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to inputs of other blocks to create complex expressions.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

D

DIN

(*Deutsches Institut für Normung*) A German institution that sets engineering and dimensional standards.

E

EIA rack

(*electronic industries alliance rack*) A standardized (EIA 310-D, IEC 60297, and DIN 41494 SC48D) system for mounting various electronic modules in a stack or rack that is 19 inches (482.6 mm) wide.

EN

EN identifies one of many European standards maintained by CEN (*European Committee for Standardization*), CENELEC (*European Committee for Electrotechnical Standardization*), or ETSI (*European Telecommunications Standards Institute*).

F

FBD

(*function block diagram*) One of 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks, where each network contains a graphical structure of boxes and connection lines, which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

FE

(functional Earth) A common grounding connection to enhance or otherwise allow normal operation of electrically sensitive equipment (also referred to as functional ground in North America).

In contrast to a protective Earth (protective ground), a functional earth connection serves a purpose other than shock protection, and may normally carry current. Examples of devices that use functional earth connections include surge suppressors and electromagnetic interference filters, certain antennas, and measurement instruments.

H

HE10

Rectangular connector for electrical signals with frequencies below 3 MHz, complying with IEC 60807-2.

I

I/O

(*input/output*)

IEC

(*international electrotechnical commission*) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

IEC 61131-3

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

IL

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

instruction list language

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

IP 20

(*ingress protection*) The protection classification according to IEC 60529 offered by an enclosure, shown by the letter IP and 2 digits. The first digit indicates 2 factors: helping protect persons and for equipment. The second digit indicates helping protect against water. IP 20 devices help protect against electric contact of objects larger than 12.5 mm, but not against water.

L

ladder diagram language

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

LD

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

M

master/slave

The single direction of control in a network that implements the master/slave mode.

Modbus

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

N

NEMA

(*national electrical manufacturers association*) The standard for the performance of various classes of electrical enclosures. The NEMA standards cover corrosion resistance, ability to help protect from rain, submersion, and so on. For IEC member countries, the IEC 60529 standard classifies the ingress protection rating for enclosures.

P

PDO

(*process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PE

(*Protective Earth*) A common grounding connection to help avoid the hazard of electric shock by keeping any exposed conductive surface of a device at earth potential. To avoid possible voltage drop, no current is allowed to flow in this conductor (also referred to as *protective ground* in North America or as an equipment grounding conductor in the US national electrical code).

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

R

RJ45

A standard type of 8-pin connector for network cables defined for Ethernet.

RPDO

(*receive process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

RS-485

A standard type of serial communication bus, based on 2 wires (also known as EIA RS-485).

RxD

The line that receives data from one source to another.

S

SFC

(*sequential function chart*) A language that is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

T

terminal block

(*terminal block*) The component that mounts in an electronic module and provides electrical connections between the controller and the field devices.

TPDO

(*transmit process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

TxD

The line that sends data from one source to another.

Index

N

P

Q

R

S

T

U

W