Modicon LMC058 Motion Controller

Programming Guide

03/2018

EIO0000000408.09

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 7
About the Book 9
Chapter 1 About the Modicon LMC058 Motion Controller 15
About the Modicon LMC058 Motion Controller 15
Chapter 2 How to Configure the Controller 17
How to Configure the Controller 17
Chapter 3 Libraries 21
Libraries 21
Chapter 4 Supported Standard Data Types 23
Supported Standard Data Types 23
Chapter 5 Memory Mapping 25
Controller Memory Organization 26
RAM Memory Organization 28
Flash Memory Organization 30
Relocation Table 34
Chapter 6 Tasks 37
Maximum Number of Tasks 38
Task Configuration Screen 39
Task Types 41
Motion Task 44
System and Task Watchdogs 47
Task Priorities 48
Default Task Configuration 50
Chapter 7 Controller States and Behaviors 51
7.1 Controller State Diagram 52
Controller State Diagram 53
7.2 Controller States Description 58
Controller States Description 58
7.3 State Transitions and System Events 62
Controller States and Output Behavior 63
Commanding State Transitions 66
Error Detection, Types, and Management 72
Remanent Variables 73

Chapter 8 Controller Device Editor 75
Controller Parameters 76
Controller Selection 78
PLC Settings 80
Services 82
Chapter 9 Embedded Expert I/O 85
9.1 Overview 86
Expert I/O Overview 86
9.2 DM72F0 and DM72F1 89
DM72Fx Configuration 90
Add an Expert function 94
Embedded Expert I/O Mapping 97
Event_Latch Function 100
9.3 Hardware Encoder Interface 102
Hardware Encoder Interface 103
Add an Encoder 104
9.4 Controller Power Distribution Module 105
Controller Power Distribution Module 105
Chapter 10 TM5 Modules 107
10.1 TM5 Manager Configuration 108
TM5 Manager Configuration 108
10.2 Embedded Regular I/O Modules Configuration 111
Embedded Regular I/O Configuration 112
DI12DE Embedded Regular I/O Module 114
DO12TE Embedded Regular I/O Module 115
AI4LE Embedded Regular I/O Module 118
10.3 TM5 Expansion Modules Configuration 126
TM5 Expansion Modules General Description 127
TM5 PCI Expansion Modules General Description 132
TM7 Expansion Modules General Description 133
Chapter 11 Ethernet Configuration 137
11.1 Ethernet Services 138
Presentation 139
IP Address Configuration 141
Modbus TCP Client/Server 146

Web Server 148
FTP Server 169
FTP Client 172
SNMP 173
11.2 Firewall Configuration 174
Introduction 175
Dynamic Changes Procedure 177
Firewall Behavior 178
Firewall Script Commands 180
11.3 Ethernet Optional Devices 183
Ethernet Manager 184
EtherNet/IP Device 185
Modbus TCP Slave Device 207
Chapter 12 CANopen Configuration 211
CANmotion Principle 212
CANmotion Interface Configuration 217
CANopen Interface Configuration 222
Chapter 13 Serial Line Configuration 225
Serial Line Configuration 226
ASCII Manager 228
SoMachine Network Manager 230
Modbus Serial IOScanner 231
Adding a Device on the Modbus Serial IOScanner 233
Modbus Manager 240
Adding a Modem to a Manager 244
Chapter 14 Post Configuration 245
Post Configuration Presentation 246
Post Configuration File Management 248
Post Configuration Example 250
Chapter 15 Connecting a Modicon LMC058 Motion Controller to a PC 253
Connecting the Controller to a PC 253
Chapter 16 Transfer by USB memory Key 255
Changing Modicon LMC058 Motion Controller Firmware 256
File Transfer with USB Memory Key 258
Chapter 17 Compatibility 265
Software and Firmware Compatibilities 265
Appendices 267

Appendix A Functions to Get/Set Serial Line Configuration in User Program 269
GetSerialConf: Get the Serial Line Configuration 270
SetSerialConf: Change the Serial Line Configuration 271
SERIAL_CONF: Structure of the Serial Line Configuration Data Type 273
Appendix B How to change the IP address of the controller 275
changeIPAddress: Change the IP address of the controller 275
Appendix C Controller Performance 279
Processing Performance 279
Glossary 281
Index 293

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

The purpose of this document is to help you to program and operate your Modicon LMC058 Motion Controller.

Validity Note

This document has been updated for the release of TM3TI4D Add-on for SoMachine V4.3.

Related Documents

Title of Documentation	Reference Number
SoMachine Programming Guide	*EIO0000000067 (ENG)* *EIO0000000069 (FRE)* *EIO0000000068 (GER)* *EIO0000000071 (SPA)* *EIO0000000070 (ITA)* *EIO0000000072 (CHS)*
Modicon LMC058 Motion Controller Hardware Guide	*EIO0000000438 (ENG)* *EIO0000000439 (FRE)* *EIO0000000440 (GER)* *EIO0000000441 (SPA)* *EIO0000000442 (ITA)* *EIO0000000443 (CHS)*
Modicon TM5 Expansion Modules Configuration Programming Guide	*EIO0000000420 (ENG)* *EIO0000000421 (FRE)* *EIO0000000422 (GER)* *EIO0000000423 (SPA)* *EIO0000000424 (ITA)* *EIO0000000425 (CHS)*
Modicon TM7 Expansion Blocks Configuration Programming Guide	*EIO0000000880 (ENG)* *EIO0000000881 (FRE)* *EIO0000000882 (GER)* *EIO0000000883 (SPA)* *EIO0000000884 (ITA)* *EIO0000000885 (CHS)*

Title of Documentation	Reference Number
Modicon TM5 PCI Modules Configuration Programming Guide	*EIO0000000590 (ENG)* *EIO0000000591 (FRE)* *EIO0000000592 (GER)* *EIO0000000593 (SPA)* *EIO0000000594 (ITA)* *EIO0000000595 (CHS)*
Modicon LMC058 Motion Controller System Functions and Variables LMC058 PLCSystem Library Guide	*EIO0000000566 (ENG)* *EIO0000000567 (FRE)* *EIO0000000568 (GER)* *EIO0000000569 (SPA)* *EIO0000000570 (ITA)* *EIO0000000571 (CHS)*
Modicon LMC058 Motion Controller High Speed Counting LMC058 Expert I/O Library Guide	*EIO0000000554 (ENG)* *EIO0000000555 (FRE)* *EIO0000000556 (GER)* *EIO0000000557 (SPA)* *EIO0000000558 (ITA)* *EIO0000000559 (CHS)*
Modicon LMC058 Motion Controller Pulse Width Modulation LMC058 Expert I/O Library Guide	*EIO0000000560 (ENG)* *EIO0000000561 (FRE)* *EIO0000000562 (GER)* *EIO0000000563 (SPA)* *EIO0000000564 (ITA)* *EIO0000000565 (CHS*
SoMachine Modbus and ASCII Read/Write Functions PLCCommunication Library Guide	*EIO0000000361 (ENG)* *EIO0000000362 (FRE)* *EIO0000000363 (GER)* *EIO0000000364 (SPA)* *EIO0000000365 (ITA)* *EIO0000000366 (CHS)*
SoMachine Modem Functions Modem Library Guide	*EIO0000000552 (ENG)* *EIO0000000491 (FRE)* *EIO0000000492 (GER)* *EIO0000000493 (SPA)* *EIO0000000494 (ITA)* *EIO0000000495 (CHS)*
SoMachine Data Logging Functions DataLogging Library Guide	*EIO0000000551 (ENG)* *EIO0000000486 (FRE)* *EIO0000000487 (GER)* *EIO0000000488 (SPA)* *EIO0000000489 (ITA)* *EIO0000000490 (CHS)*

Title of Documentation	Reference Number
SoMachine Controller Assistant User Guide	*EIO0000001671 (ENG)* *EIO0000001672 (FRE)* *EIO0000001673 (GER)* *EIO0000001675 (SPA)* *EIO0000001674 (ITA)* *EIO0000001678 (CHS)*
SoMachine Compatibility and Migration User Guide	*EIO0000001684 (ENG)* *EIO0000001685 (FRE)* *EIO0000001686 (GER)* *EIO0000001688 (SPA)* *EIO0000001687 (ITA)* *EIO0000001689 (CHS*
FTPRemoteFileHandling Library Guide	*EIO0000002405 (ENG)* *EIO0000002406 (FRE)* *EIO0000002407 (GER)* *EIO0000002409 (SPA)* *EIO0000002408 (ITA)* *EIO0000002410 (CHS)*

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

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

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

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

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

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

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

Chapter 1
About the Modicon LMC058 Motion Controller

About the Modicon LMC058 Motion Controller

Overview

The Schneider Electric Modicon LMC058 Motion Controller is a controller with a variety of features.

This controller is the optimized solution for axis positioning with the SoMachine software platform, which includes embedded automation functions and an ergonomic interface for axis configuration. Combined with Lexium servo drives or Lexium SD3 Stepper drives, this lets you design and commission your applications.

The Software configuration is described in the SoMachine Programming Guide.

Key Features

The SoMachine software supports the following IEC61131-3 programming languages for use with these controllers:

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

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

All controllers support the following fieldbuses and network capabilities:

- CANmotion Master dedicated to motion device synchronization
- CANopen Master
- Ethernet
- Serial Line

All controllers support the following functions and I/O types:

- Encoder Master
- Expert functions (counting, reflex outputs...)
- Embedded I/Os

All controllers support up to 21 application program tasks with the following limits:

- 4 cyclic tasks: one is configured by default (MAST)
- 1 freewheeling task
- 8 software event driven tasks
- 9 hardware event driven tasks: 1 is the motion task synchronized with the CANmotion Master

Controller Range

	PCI	CAN	USB A	USB Pgr	Ethernet	Serial Line	Encoder
LMC058LF42 *(see Modicon LMC058, Motion Controller, Hardware Guide)*	0	2	1	1	1	1	1
LMC058LF424 *(see Modicon LMC058, Motion Controller, Hardware Guide)*	2	2	1	1	1	1	1

	Embedded Expert I/O				Embedded Regular I/O			
		Fast Inputs	Fast Outputs	Regular Inputs		Digital Inputs	Digital Outputs	Analog Inputs
LMC058LF42 *(see Modicon LMC058, Motion Controller, Hardware Guide)*	2x	5	2	2	1x	12	12	0
LMC058LF424 *(see Modicon LMC058, Motion Controller, Hardware Guide)*	2x	5	2	2	1x	12	12	4

Chapter 2
How to Configure the Controller

How to Configure the Controller

Introduction

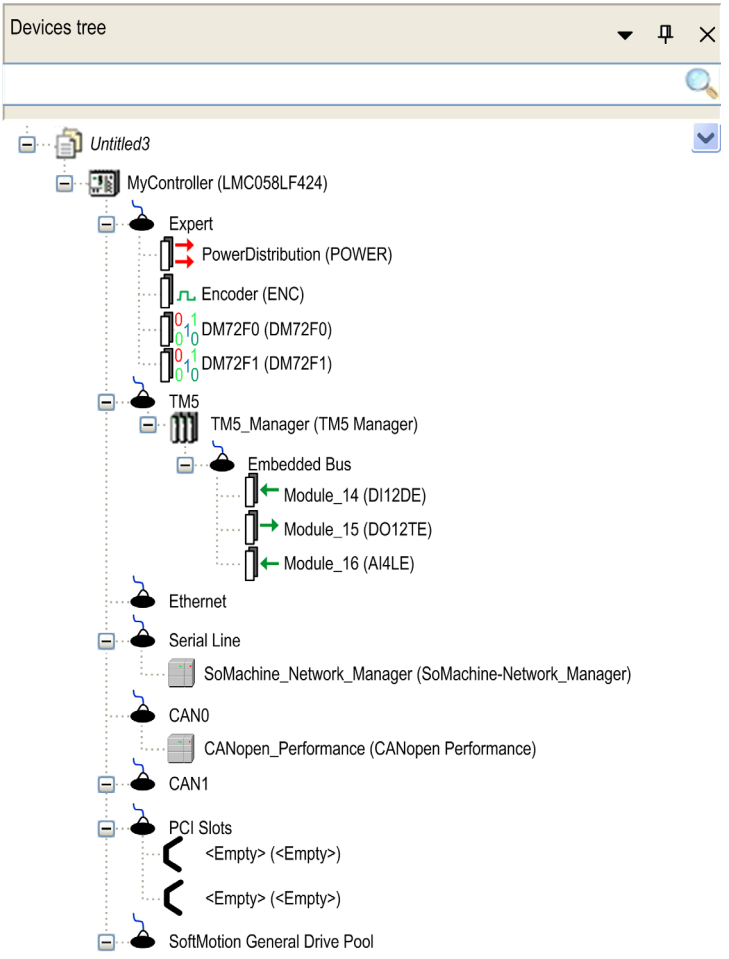
First, create a new project or open an existing project in the SoMachine software.

Refer to the *SoMachine 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	Description
Expert	Presents the Embedded Expert I/O.
TM5	Contains the TM5 bus manager, the embedded regular I/O modules and the expansion modules in the controller.
Ethernet **Serial Line** **CAN0** **CAN1**	Embedded communications interfaces.
PCI slots	Communication interfaces on the bus are presented as slots.
SoftMotion General Drive Pool	SoftMotion devices (Virtual axis configuration).

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

Modicon LMC058 Motion Controller

When you select a Modicon LMC058 Motion Controller for your application, SoMachine automatically loads the following libraries:

Library name	Description
IoStandard	**CmpIoMgr** configuration types, **ConfigAccess**, Parameters and help functions: manages the I/Os in the application.
Standard	Contains functions and function blocks which are required matching IEC61131-3 as standard POUs for an IEC programming system. Link the standard POUs to the project (standard.library).
Util	Analog Monitors, BCD Conversions, Bit/Byte Functions, Controller Datatypes, Function Manipulators, Mathematical Functions, Signals.
PLCCommunication *(see SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	**SysMem, Standard**. These functions facilitate communications between specific devices. Most of them are dedicated to Modbus exchange. Communication functions are processed asynchronously with regard to the application task that called the function.
LMC058 PLCSystem *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 PLCSystem Library Guide)*	Contains functions and variables to get information and send commands to the controller system.
LMC058 Relocation Table *(see page 34)*	Allows organization of data to optimize exchanges between the Modbus client and the controller, by regrouping non-contiguous data into a contiguous table of registers.

Chapter 4
Supported Standard Data Types

Supported Standard Data Types

Supported Standard Data Types

The controller supports the following IEC data types:

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

Chapter 5
Memory Mapping

Introduction

This chapter describes the memory maps and sizes of the different memory areas in the Modicon LMC058 Motion Controller. These memory areas are used to store user program logic, data and the programming libraries.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Controller Memory Organization	26
RAM Memory Organization	28
Flash Memory Organization	30
Relocation Table	34

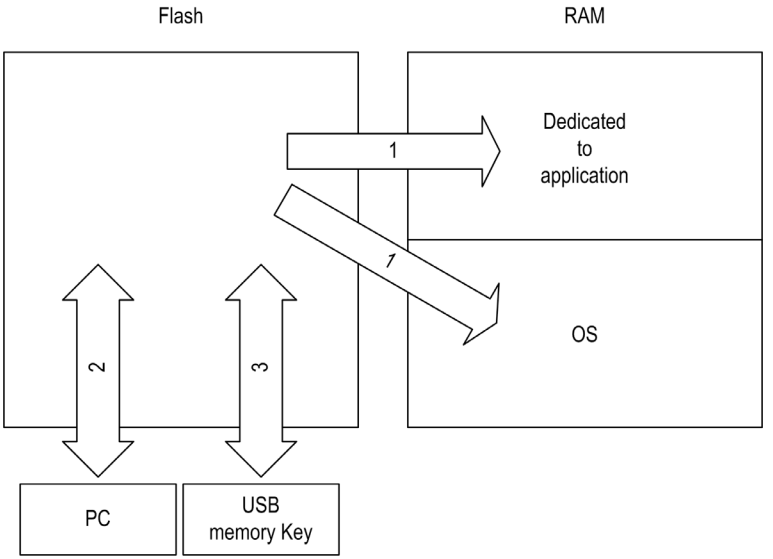
Controller Memory Organization

Introduction

The controller memory is composed of 2 types of physical memory:

- The Flash memory *(see page 30)* contains files (application, configuration files, and so on).
- The RAM (Random Access Memory) *(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 148)* ● FTP server *(see page 169)* ● SoMachine

(1) If the controller is in the INVALID_OS state, the Flash memory is accessible only via the USB host connection and only for firmware upgrades.

Item	Controller State	File Transfer Events	Connection	Description
3	All states	Initiated automatically by script when a USB memory key is connected	USB host connection	Up/Download with USB memory key *(see page 255)*
(1) If the controller is in the INVALID_OS state, the Flash memory is accessible only via the USB host connection and only for firmware upgrades.				

NOTE: All files in Flash memory can be read, written, or erased, no matter the controller state. 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 LMC058 Motion 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 128 Kbytes	System Area Mappable Addresses %MW0...%MW59999	125000 bytes
	System and diagnostic variables *(see page 29)* (%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 can be read or write requests. However, if this memory is declared in the relocation table, these must be read-only requests.	
	System and diagnostic variables *(see page 29)* (%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. However, if this memory is declared in the relocation table, it must write-only requests.	
	Reserved	3000 bytes
	Retains Data *(see page 30)*	32 ko
	Persistent Data *(see page 30)*	64 ko
User area 10 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.
SERIAL_R	Structure of Serial Lines read only system variables.
SERIAL_W	Structure of Serial Lines read / write system variables.
TM5_MODULE_R	Structure of TM5 modules read only system variables.
PROFIBUS_R	Structure of Profibus read system variables.

For more information on System and Diagnostic Variables, refer to *LMC058 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.

The total size of the Flash memory is 128 MB, of which 10 MB is available for the application.

File Type

The Modicon LMC058 Motion Controller manages the following file types:

Type	Description
Executable application	User application. This is the binary code that is executed when the controller is in the RUNNING state.
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 user 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 SoMachine, according to application properties. Default option is to create the boot application on download. When you download an application from SoMachine to the controller, you are transferring only the binary executable application directly to RAM.

(2) SoMachine 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.

There are 2 ways to create the boot application:

- Select the option during the application download process.
- Choose **Online → Create boot application** at any point after download.

If you do not create a boot application, the controller will enter the EMPTY state after the next reboot.

File Organization

This table shows the file organization of the flash memory:

Disk	Directory	File	Content	Up/Downloaded Data Type
/sys	OS	M258FW1v_XX.YY (1)	Firmware of core 1	Firmware
		M258FW2v_XX.YY (1)	Firmware of core 2	
		M258_top_Vxx.bit	Firmware	
		Version.ini	Control file for firmware version	
		NXCIF50-RTE.bin	Profibus firmware file	Firmware
		cifxdps.nxf		
	Web	Index.htm	HTML pages displayed 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	
	App/MFW	DeviceID_X.fw (2)	Expansion modules Firmware	Firmware
	Cfg	Machine.cfg(2)	Post configuration file *(see page 245)*	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

Disk	Directory	File	Content	Up/Downloaded Data Type
/usr	Dta	*UserDefinedDtaName_1*.Dta	All *.Data files created using the DataFileCopy function block *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 PLCSystem Library Guide)*	Data files
		...		
		UserDefinedDtaName_n.Dta		
	Log	*UserDefinedLogName_1*.log	All *.log files created using the data logging functions *(see SoMachine, Data Logging Functions, DataLogging Library Guide)*. Specify the total number of files created and the names and contents of each log file.	log file
		...		
		UserDefinedLogName_n.log		
	Ntx	NXCIF50-RTE.bin	Profibus firmware file	Firmware
		cifxdps.nxf		
	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 in SoMachine online by viewing the **Log** tab of the **Controller Device Editor** *(see page 76)*.	
		FWLog.txt	This file contains a record of firmware system events. For use by Schneider Electric Technical Support.	
	Eip	My_Machine_Controller.eds My_Machine_Controller.gz My_Machine_Controller.ico	These files are necessary to configure and operate your controller as an EtherNet/IP Master.	Configuration and icon files
/bd0	–	–	USB memory	Application Configuration log file Firmware Website
(1) v_XX.YY represents the version **(2)** If any				

NOTE: Use the sysFile, sysDir, and CAAFile libraries to access to /bd0 and /usr. For more information on the function blocks of these libraries, refer to the CoDeSys Libraries topic in the SoMachine online help.

Moving Files to Flash Memory

When user activity creates certain file types, the LMC058 Motion 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 quite large to the point of exceeding the space available in the file system. You should, therefore, develop a method to periodically archive the log data on a USB key. For example, you could split the log data into several files, e.g. `LogMonth1, LogMonth2`, and use the **ExecuteScript** command *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 PLCSystem Library Guide)* to copy the first file to a USB key, and thereafter, 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 DATA

Back up your *.log files to a USB key on a regular schedule that avoids saturating the available free space in your controller file system.

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 *LMC058 PLCSystem Library Guide*.

Adding a Relocation Table

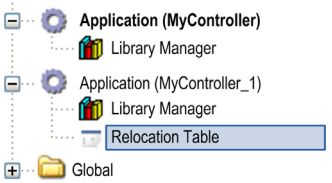
This table describes how to add a **Relocation Table** to your project:

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.

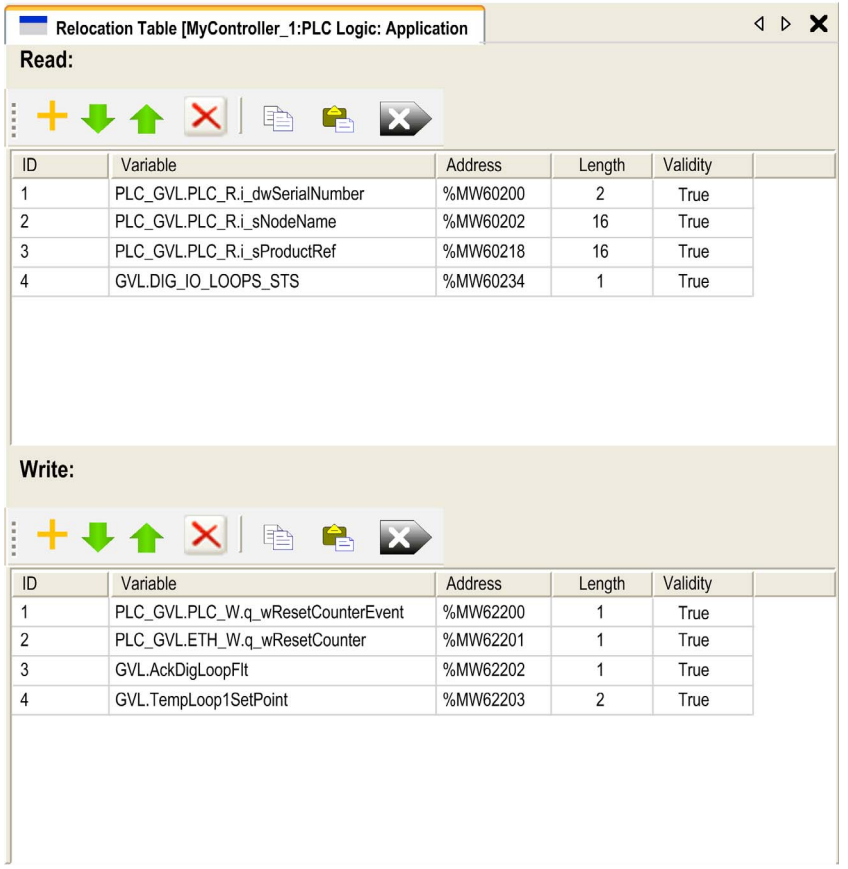
Relocation Table Editor

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:



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
- Motion (The Motion task is an External event task)

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

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Maximum Number of Tasks	38
Task Configuration Screen	39
Task Types	41
Motion Task	44
System and Task Watchdogs	47
Task Priorities	48
Default Task Configuration	50

Maximum Number of Tasks

Maximum Number of Tasks

The maximum number of tasks you can define for the Modicon LMC058 Motion Controller are:

- Total number of tasks = 21
- Cyclic tasks = 4
- Freewheeling tasks = 1
- Event tasks = 8
- External Event tasks = 9

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 47)*.

NOTE: You may wish to avoid using a Freewheeling task in a multi-task application when some high priority and time-consuming tasks (typically the Motion task) 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.

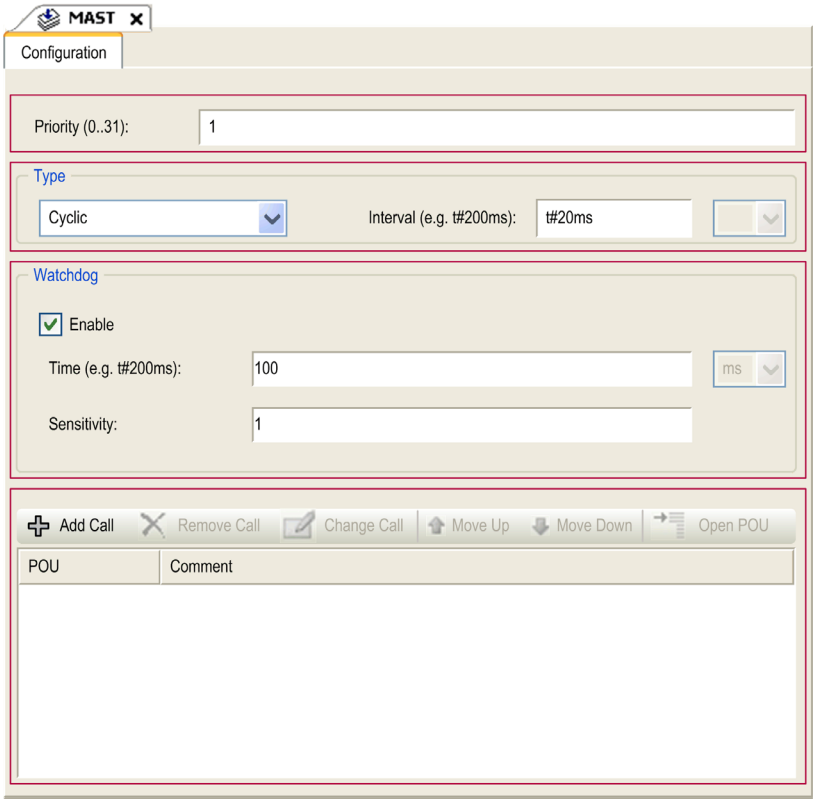
Task Configuration Screen

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 48)*.
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 47)*, 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 *(see page 53)*.
POUs	The list of POUs *(see SoMachine, Programming Guide)* (Programming Organization Units) controlled by the task is defined in the task configuration window: ● To add a POU linked to the task, use the command **Add Call** and select the POU in the **Input Assistant** editor. ● To remove a POU from the list, use the command **Remove Call**. ● To replace the currently selected POU of the list by another one, use the command **Change Call**. ● **POU**s are executed in the order shown in the list. To move the **POU**s in the list, select a **POU** and use the command **Move Up** or **Move Down**. **NOTE:** You can create as many POUs as you want. An application with several small POUs, as opposed to one large POU, can improve the refresh time of the variables in online mode.

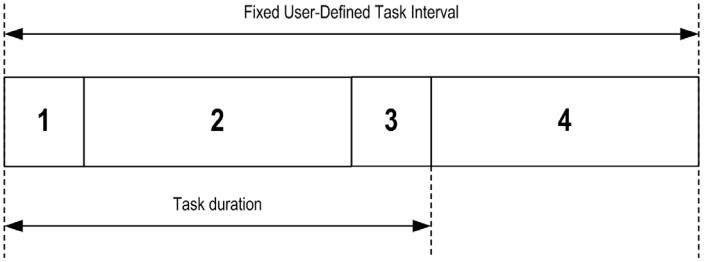
Task Types

Introduction

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

Cyclic Task

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



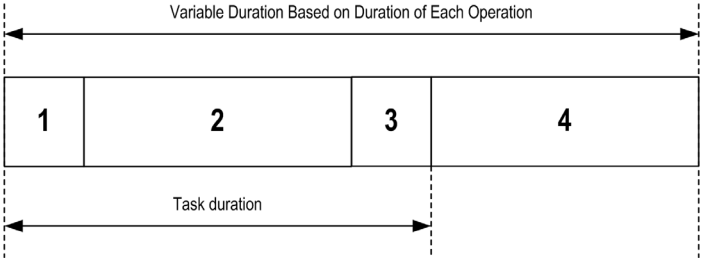
1. **Read Inputs:** The physical input states are written to the %I input memory variables and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variables are updated according to your application program instructions but not yet written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variables are modified with any output forcing that has been defined; however, the writing of the physical outputs depends upon the type of output and instructions used.
   For more information on defining the bus cycle task, refer to the SoMachine Programming Guide and Modicon LMC058 Motion Controller Settings *(see page 80)*.
   For more information on I/O behavior, refer to Controller States Detailed Description *(see page 58)*.
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: 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 SoMachine Programming Guide and Modicon LMC058 Motion Controller Settings *(see page 80)*.
   For more information on I/O behavior, refer to Controller States Detailed Description *(see page 58)*.
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: The maximum frequency admissible for the event triggering an Event task is 100 Hz.

External Event Task

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

For example, an External event task could be associated with an HSC Stop event. To associate the **BLOCK0_HSCSTOP** event to an External event task, select it from the **External** event drop-down list on the **Configuration** tab.

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

- Rising edge on an advanced input (`DI0...DI7`)
- HSC thresholds
- HSC Stop
- CAN Sync

NOTE: CAN Sync is a specific event object, dependent on the **CANopen manager** configuration. When the **Sync Generation** is enabled in the **CANopen Manager**, an associated CANx_Sync task is automatically created in the task configuration.

NOTE: The maximum frequency admissible for an event triggering an Event task is 100 Hz.

Motion Task

Introduction

This section presents the characteristics of the Motion task and provides information on the performance possible when using an optimally configured motion system. The Motion task is created automatically with the **External Event** name of CAN1_Sync when a **CANmotion Manager** is configured. This mechanism allows a synchronization of the Motion task with the devices declared on the CANmotion bus. By default the Motion task is configured with priority 1.

NOTE: You can modify this priority setting but you must ensure that the Motion task will have enough time to execute within the CANmotion **Sync cycle period (µs)**.
An adequately defined cycle time meets both of the following requirements:

- The program processing defined in your Motion task must have enough time to execute in full. Test the execution time of your Motion task under all operating conditions to determine this value.
- The **Sync cycle period (µs)** must be of sufficient length to allow the physical exchange of all PDO and SDO data between the controller and all of the configured devices.

If you do not configure a sufficient **Sync cycle period (µs)** this can result in a system watchdog exception or even a loss of synchronization for the controlled devices and unpredictable operation. For example, an insufficient **Sync cycle period (µs)** may result in the detection of the loss of the CANmotion master for all controlled devices. In this case, devices detecting a loss of the CANmotion master will assume their programmed fallback states. Always confirm that your **Sync cycle period (µs)** is sufficient to allow full execution of the Motion Task and a complete physical exchange of all PDO and SDO data before placing your system into service.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

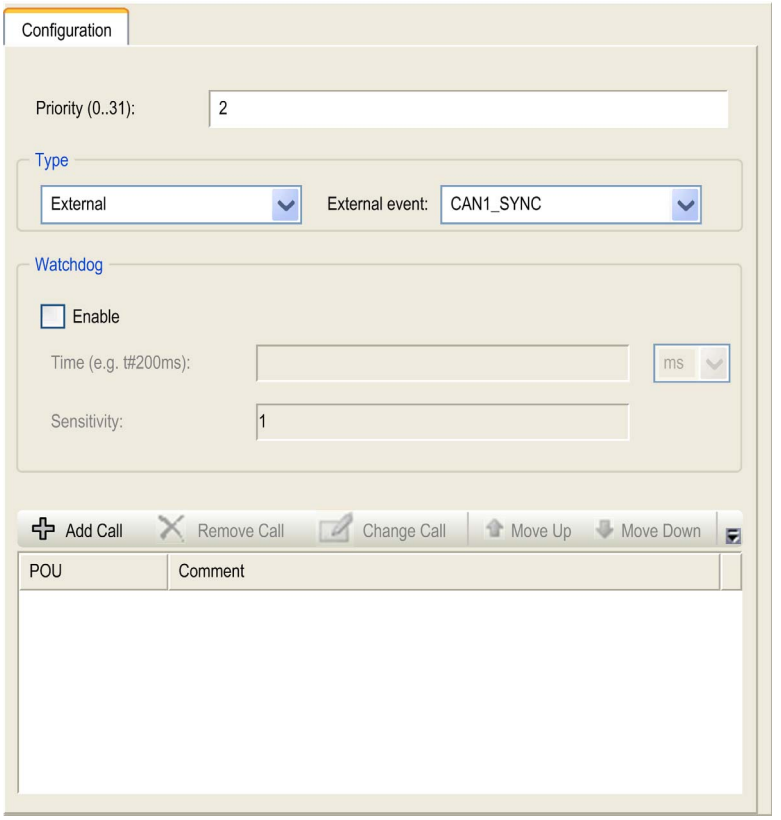
Take the following steps to define an adequate **Sync cycle period (µs)** for your Motion task:

- Calculate the required minimum cycle time for your task processing and physical data exchange (1).
- Define a task (software) watchdog for the Motion task with a watchdog period slightly larger than the **Sync cycle period (µs)** defined for the **CANmotion Manager**.
- Thoroughly test your CANmotion system under normal and non-normal operating conditions before placing your system into service.

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

(1) Refer to CANmotion Cycle Time Configuration *(see page 217)* for instructions on calculating the required minimum cycle time for your Motion task.

This picture displays the settings for the Motion task:



NOTE: Do not delete the Motion task or change its **Name**, **Type**, or **External Event** attributes. If you do so, SoMachine will detect an error when you attempt to build the application, and you will not be able to download it to the controller.

Motion Task Programming Requirements

You must use the Motion task to manage every aspect of programming related to the CANmotion bus and its connected motion devices such as drive controllers.

This includes:

- Local inputs used to acquire motion events
- Encoder inputs used to acquire motion events
- Task processing for all motion functions (CANmotion, Motion task POU, etc.)
- Transmission of all RPDO to motion devices
- Receipt of all TPDO from motion devices
- Transmission and receipt of all SDO and Optional PDO related to motion processing and events
- Encoder outputs configured to respond to motion events
- Local outputs configured to respond to motion events

If you attempt to manage motion-related inputs, outputs, task processing, or CAN communications outside of the motion task, your system may behave in an unexpected manner.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Use the Motion task to manage all motion-related inputs, outputs, task processing, and CAN communications.

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

Motion Task Performance

The Modicon LMC058 Motion Controller is capable of achieving very high performance under certain conditions. For example, if you write a very efficient application using a small subset of the available motion functions, and do not use interpolation, the controller can manage up to:

- 4 axes in 2 ms
- 8 axes in 4 ms

The subset of functions that can be used and still allow you to achieve similar performance (provided you write an efficient application) are:

- Virtual axes
- Relative and absolute positioning
- Speed control
- Cam profiles
- Electronic gearing
- Linear and circular interpolation

NOTE: The interpolation functions are required for certain applications such as CNC milling.

For more information refer to CANmotion Principle *(see page 212)* and CANmotion Interface Configuration *(see page 217)*.

System and Task Watchdogs

Introduction

Two types of watchdog functionality are implemented for the Modicon LMC058 Motion 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 SoMachine.

System Watchdogs

Three system watchdogs are defined for the Modicon LMC058 Motion Controller. They are managed by the controller firmware and are therefore sometimes referred to as hardware watchdogs in the SoMachine 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

SoMachine 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 SoMachine 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 SoMachine 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. If you assign the same priority to more than one task, execution for those tasks is indeterminate and unpredictable, which may lead to unintended consequences.

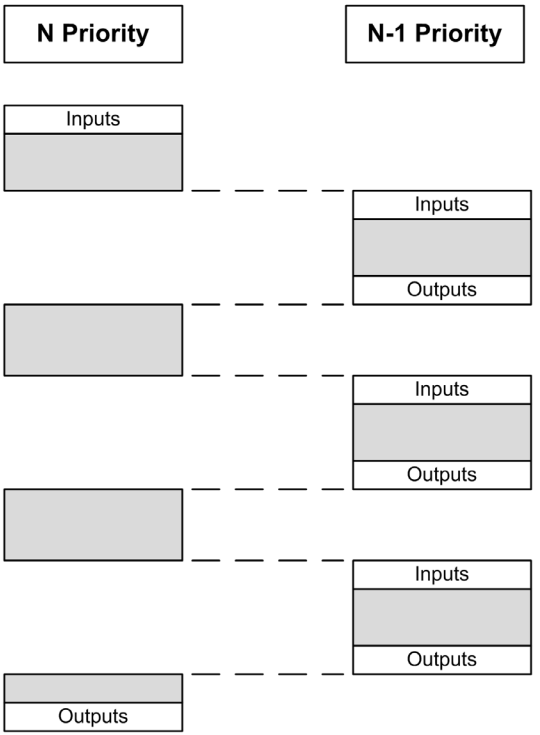
⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Do not assign the same priority to different tasks. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

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 Preemption Due to Task Priorities

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



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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Default Task Configuration

Default Task Configuration

Default Task Configuration

For the Modicon LMC058 Motion Controller:

- 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 48)* for more information on priority settings. Refer to System and Task Watchdogs *(see page 47)* for more information on watchdogs.
- A CANx_Sync task is created automatically when a CANopen Manager is added to the CANx (CAN0 or CAN1) interface and configured with Sync Generation enabled. This task is declared as an External Event task, and reduces the number of External Event tasks you can configure for other operations by one. By default, the CANx_Sync task is assigned a priority of 2 (or 3 if another CANx_Sync task has already been created). This is appropriate for many installations, but it is your responsibility to verify the correct task priority setting for your system. Refer to Task Priorities *(see page 48)* for more information.
- A Motion task is created automatically when a CANmotion Manager is added to the CAN1 interface. This task is declared as an External Event task, and reduces the number of External Event tasks you can configure for other operations by one. By default, the Motion task is assigned a priority of 1. This is appropriate for many installations, but it is your responsibility to verify the correct task priority setting for your motion system. Refer to Task Priorities *(see page 48)* for more information.

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, Motion and CANx_Sync tasks. If you do so, SoMachine detects an error when you attempt to build the application, and you will not be able to download it to the controller.

NOTE: Do not change the Type or External Event attributes of the Motion or CANx_Sync tasks. If you do so, SoMachine will detect 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 SoMachine task programming options on the behavior of your system.

What Is in This Chapter?

This chapter contains the following sections:

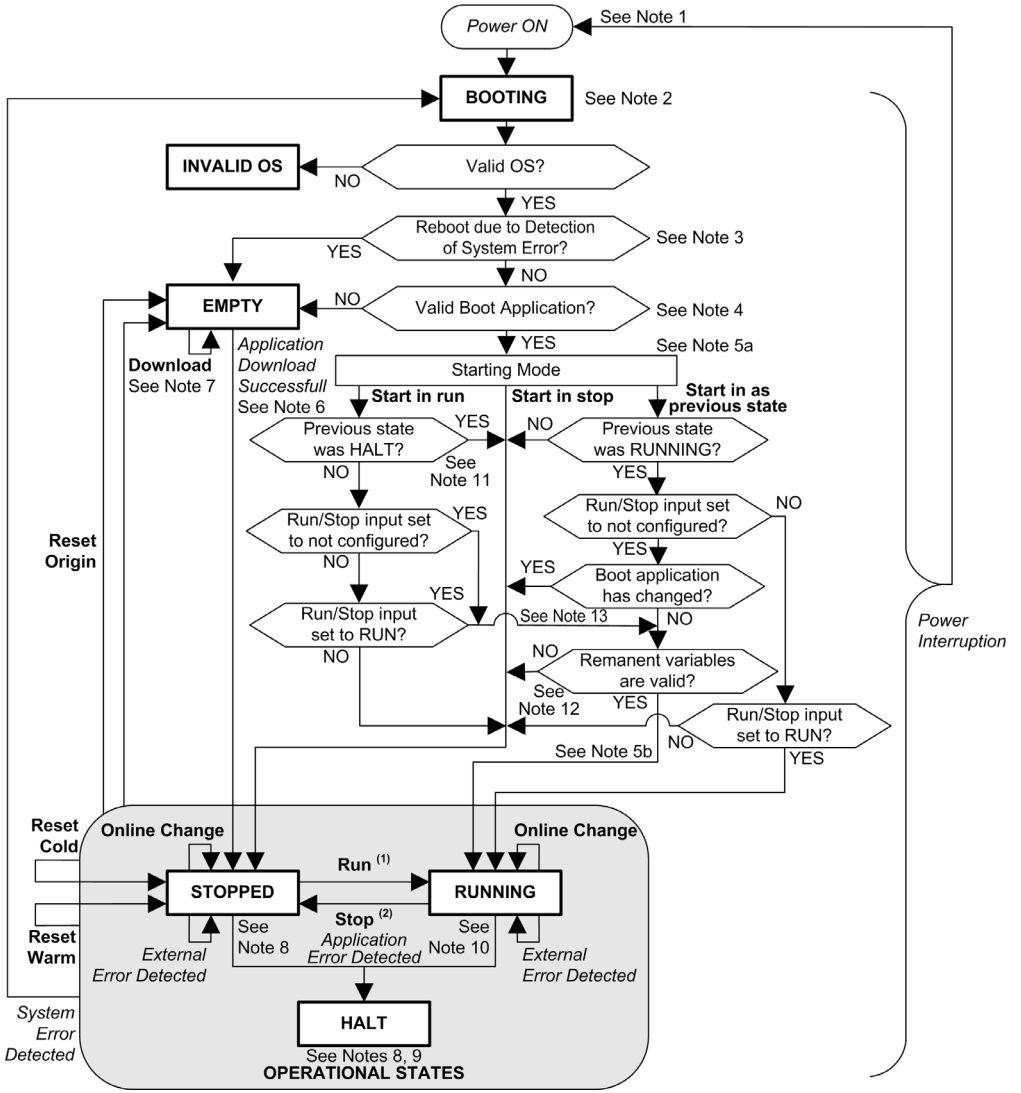
Section	Topic	Page
7.1	Controller State Diagram	52
7.2	Controller States Description	58
7.3	State Transitions and System Events	62

Section 7.1
Controller State Diagram

Controller State Diagram

Controller State Diagram

The following 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 66)*.

(2) For details on RUNNING to STOPPED state transition, refer to Stop Command *(see page 66)*.

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 63)* for further details.

Note 2

There is a 4-5 second delay between entering the BOOTING state and the LED indication of this state. The boot process can take up to 45 seconds under normal conditions. The outputs will assume their initialization states.

Note 3

In some cases, when a system error is detected, it will cause the controller to automatically reboot into the EMPTY state as if no Boot application were present in the Flash memory. However, the Boot application is not actually deleted from the Flash memory.

Note 4

After verification of a valid Boot application the following events occur:
- The application is loaded into RAM.
- The Post Configuration *(see page 245)* 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 69)*.

Note 5a

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

Note 5b

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

Note 6

During a successful application download, the following events occur:

- The application is loaded directly into RAM.
- By default, the Boot application is created and saved into the Flash memory.
- The Post Configuration *(see page 245)* file settings (if any) are applied.

Note 7

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

However, there are two important considerations in this regard:

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

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

NOTE: During a multiple download, unlike a normal download, SoMachine 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 (the controller must be in the STOPPED state for this operation).

Note 8

The SoMachine 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 58)* 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 (system watchdog or internal error), a cycle power is mandatory.

Note 10

The RUNNING state has two exception conditions.

They are:

- RUNNING with External Error Detected: this exception condition is indicated by the MS Status LED, which displays solid green with 1 red flash. You may exit this state by clearing the external detected error. No controller commands are required.
- RUNNING with Breakpoint: this exception condition is indicated by the MS Status LED, which displays 3 green flashes. Refer to Controller States Description *(see page 58)* for further details.

Note 11

When Starting Mode is set to Start in run and if the Run/Stop input is not configured, the controller will reboot in STOPPED state. A second reboot will be necessary to set the controller in RUNNING state.

Note 12

Remanent variables can be invalid if battery is not present for example.

Note 13

The boot application can be different from the application loaded. It can happen when the boot application was downloaded through USB Key, FTP or File Transfer or when an online change was performed without creating the boot application.

Section 7.2
Controller States Description

Controller States Description

Introduction

This section provides a detailed description of the controller states.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Never assume that your controller is in a certain controller state before commanding a change of state, configuring your controller options, uploading a program, or modifying the physical configuration of the controller and its connected equipment.
- Before performing any of these operations, consider the effect on all connected equipment.
- Before acting on a controller, always positively confirm the controller state by viewing its LEDs, confirming the condition of the Run/Stop input, verifying the presence of output forcing, and reviewing the controller status information via SoMachine.(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 LMC058 PLCSystem library *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 PLCSystem Library Guide)*

Controller States Table

The following table describes the controller states:

Controller State	Description	RUN/MS LED
BOOTING	The controller executes the boot firmware and its own internal self-tests. It then verifies the checksum of the firmware and user applications. It does not execute the application nor does it communicate.	Green/red flashing
BOOTING after detection of a *System Error*	This state is the same as the normal BOOTING state except that a flag is set to make it appear as if no Boot application is present and the LED indications are different.	Rapid red flashing

Controller State	Description	RUN/MS LED
INVALID_OS	There is not a valid firmware file present In the Flash memory. The controller does not execute the application. Communication is only possible through the USB host port, and then only for uploading a valid OS. Refer to Upgrading Modicon LMC058 Motion Controller Firmware *(see page 256)*.	Red flashing
EMPTY	There is no application present or an invalid application. PCI expansion modules are inactive.	Single green flash
EMPTY after detection of a *System Error*	This state is the same as the normal EMPTY state except that a flag is set to make it appear as if no Boot Application is present (no Application is loaded) and the LED indications are different.	Rapid red flashing
RUNNING	The controller is executing a valid application.	Green
RUNNING with Breakpoint	This state is the same as the RUNNING state with the following exceptions: ● The task-processing portion of the program does not resume until the breakpoint is cleared. ● The LED indications are different. For more information on breakpoint management, refer to the SoMachine Menu Commands Online Help.	3 green flashes
RUNNING with detection of an *External Error*	This state is the same as the normal RUNNING state except the LED indications are different.	Green / single red flash
STOPPED	The controller has a valid application that is stopped. See Details of the STOPPED State *(see page 60)* for an explanation of the behavior of outputs and field buses in this state.	Green flashing
STOPPED with detection of an *External Error*	This state is the same as the normal STOPPED state except the LED indications are different.	Green flashing / single red flash
HALT	The controller stops executing the application because it has detected an Application Error. This description is the same as for the STOPPED state with the following exceptions: ● Expert I/O and TM5 I/O busses cease communications. Expert and TM5 outputs assume their Initialization values *(see page 63)*. ● CAN bus behaves as if the Update IO while in stop option was not selected when managed by a task responsible for the Application Detected Error. Else, CAN bus behavior follows the actual setting. ● The LED indications are different.	Single red flash

Details of the STOPPED State

The following statements are true for the STOPPED state:

- The input configured as the Run/Stop input remains operational.
- The output configured as the Alarm output remains operational and goes to a value of 0.
- Ethernet, Serial (Modbus, ASCII, and so on), and USB communication services remain operational and commands written by these services can continue to affect the application, the controller state, and the memory variables.
- All outputs initially assume their configured default state (**Keep current values** or **Set all outputs to default**) or the state dictated by output forcing if used. The subsequent state of the outputs depends on the value of the **Update IO while in stop** setting and on commands received from remote devices.

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

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

- The Read Inputs operation continues normally. The physical inputs are read and then written to the `%I` input memory variables.
- The Task Processing operation is not executed.
- The Write Outputs operation continues. The `%Q` output memory variables are updated to reflect either the **Keep current values** configuration or the **Set all outputs to default** configuration, adjusted for any output forcing, and then written to the physical outputs.
  NOTE: Expert functions continue to operate. For example, a counter will continue to count. However, these Expert functions do not affect the state of the outputs. The outputs of Expert I/O conform to the behavior stated here.

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

CAN Behavior When Update IO While In Stop Is Not Selected

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

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

Section 7.3
State Transitions and System Events

Overview

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

What Is in This Section?

This section contains the following topics:

Topic	Page
Controller States and Output Behavior	63
Commanding State Transitions	66
Error Detection, Types, and Management	72
Remanent Variables	73

Controller States and Output Behavior

Introduction

The Modicon LMC058 Motion 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.

Keep Current Values

Select this option by choosing **Keep current values** in the **Behavior for outputs in Stop** drop-down menu of the **PLC settings** subtab of the **Controller Editor**. 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 58)* for more details on these variations.

Set All Outputs to Default

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

This output behavior applies when the application is going from RUN state to STOPPED state or if the application is going from RUN state to HALT 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 58)* 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. It also applies to **Expert I/O** and TM5 I/O busses in the HALT controller state.

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 from 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 SoMachine.

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

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

When you logout of SoMachine 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 will have no effect on the output. However, once the task that had been delayed is executed, the forcing will take 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 SoMachine without removing the forcing.

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

Commanding State Transitions

Run Command

Effect: Commands a transition to the RUNNING controller state.

Starting Conditions: BOOTING or STOPPED state.

Methods for Issuing a Run Command:

- Run/Stop Input: If configured, command a rising edge to the Run/Stop input (assuming the Run/Stop switch is in the RUN position). Set the Run/Stop to 1 for all of the subsequent options to be effective.
  Refer to Run/Stop Input *(see page 87)* for more information.
- SoMachine 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 LMC058 PLCSystem library *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 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 53)* for further details.

Stop Command

Effect: Commands a transition to the STOPPED controller state.

Starting Conditions: BOOTING, EMPTY, or RUNNING state.

Methods for Issuing a Stop Command:

- Run/Stop Input: If configured, command a value of 0 to the Run/Stop input. Refer to Run/Stop Input *(see page 87)* for more information.
- SoMachine 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 LMC058 PLCSystem library *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 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 a USB memory key can issue a REBOOT as its final command. The controller will be rebooted into the STOPPED state provided the other conditions of the boot sequence allow this to occur. Refer to Script and Files Generation with USB Mass Storage *(see page 260)* and Reboot *(see page 69)* for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram *(see page 53)* 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:

- SoMachine Online Menu: Select the **Reset warm** command.
- By an internal call by the application using the PLC_W. q_wPLCControl and PLC_W. q_uiOpen-PLCControl system variables of the LMC058 PLCSystem library *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 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 I/O are reset to their initialization values.
11. The Post Configuration *(see page 245)* file is read.

For details on variables, refer to Remanent Variables *(see page 73)*.

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:

- SoMachine Online Menu: Select the **Reset cold** command.
- By an internal call by the application using the PLC_W. q_wPLCControl and PLC_W. q_uiOpen-PLCControl system variables of the LMC058 PLCSystem library *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 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 I/O are reset to their initialization values.
11. The Post Configuration file is read *(see page 245)*.

For details on variables, refer to Remanent Variables *(see page 73)*.

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:

- SoMachine Online Menu: Select the **Reset origin** command.

Effects of the Reset Origin Command:

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

11. Embedded Expert I/O are reset to their previous user-configured default values.
12. All other I/O are reset to their initialization values.

For details on variables, refer to Remanent Variables *(see page 73)*.

Reboot

Effect: Commands a reboot of the controller.

Starting Conditions: Any state.

Methods for Issuing the Reboot Command:

- Power cycle
- REBOOT by Script

Effects of the Reboot:

1. The state of the controller depends on a number of conditions:
   a. The controller state will be 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 will be STOPPED if:
   The Reboot was provoked by a power cycle or a reboot by script 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.

c. The controller state will be 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 will be 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 I/O are reset to their initialization values and then to their user-configured default values if the controller assumes a STOPPED state after the reboot.
11. The Post Configuration file is read *(see page 245)*.
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 73)*.

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

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

NOTE: 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 will detect a difference in context at the next reboot, the remanent variables will be reset as per a Reset cold command, and the controller will enter 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:

- SoMachine:
  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 *(see page 53)*.
- FTP: Load Boot application file to the Flash memory using FTP. The updated file is applied at the next reboot.
- USB memory key: Load Boot application file using a USB memory key connected to the controller USB host port. The updated file is applied at the next reboot. Refer to File Transfer with USB Memory Key *(see page 258)* for further details.

Effects of the SoMachine Download Command:

1. The existing application stops and then is erased.
2. If valid, the new application is loaded and the controller assumes a STOPPED state.
3. Forcing is erased.
4. Diagnostic indications for errors are reset.
5. The values of the retain variables are reset to their initialization values.
6. The values of any existing retain-persistent variables are maintained.
7. All non-located and non-remanent variables are reset to their initialization values.
8. The values of the first 1000 `%MW` registers are maintained.
9. The values of `%MW1000` to `%MW59999` registers are reset to 0.
10. All fieldbus communications are stopped and then any configured fieldbus of the new application is started after the download is complete.
11. Embedded Expert I/O are reset to their previous user-configured default values and then set to the new user-configured default values after the download is complete.
12. All other I/O are reset to their initialization values and then set to the new user-configured default values after the download is complete.
13. The Post Configuration file is read *(see page 245)*.

For details on variables, refer to Remanent Variables *(see page 73)*.

Effects of the FTP or USB key 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 69)*.

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 the state of an output. ● The controller detects a communication interruption with a device. ● The controller is configured for a module that is not present or not detected. ● 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 time-out occurs. **NOTE:** There are some system errors that can be managed by runtime and are therefore treated like application errors.	BOOTING → EMPTY

NOTE: Refer to the LMC058 PLCSystem library Guide *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 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 have the same behavior as variables declared as retain-persistent.

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

Action	VAR	VAR RETAIN	VAR GLOBAL PERSISTENT RETAIN
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 67)*.
(3) If the application is downloaded via SD card, any existing persistent variables used by the application are reinitialized. If the application is downloaded using SoMachine, however, existing persistent variables maintain their values. In both cases, if the downloaded application contains the same 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` will behave 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	Select the **Application** node in the **Applications tree**.
2	Click  .
3	Choose **Add other objects → Persistent variables**
4	Click **Add**. **Result:** The **PersistentVars** window is displayed.

Chapter 8
Controller Device Editor

Introduction

This chapter describes how to configure the controller.

What Is in This Chapter?

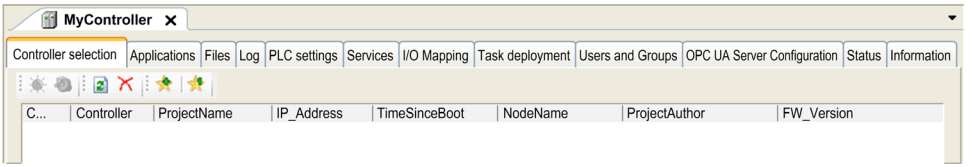
This chapter contains the following topics:

Topic	Page
Controller Parameters	76
Controller Selection	78
PLC Settings	80
Services	82

Controller Parameters

Controller Parameters

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



Tabs Description

Tab	Description	Restriction
Controller selection *(see page 78)*	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.	Online mode only
Applications	Presents the application running on the controller and allows removing the application from the controller.	Online mode only
Files	File management between the PC and the controller.	Online mode only
Log	View the controller log file.	Online mode only
PLC settings *(see page 80)*	Configuration of: ● application name ● I/O behavior in stop ● bus cycle options	–
Services *(see page 82)*	Lets you configure the online services of the controller (RTC, device identification).	Online mode only
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

Tab	Description	Restriction
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 SoMachine projects and devices in online mode. For more details, refer to the SoMachine Programming Guide.	–
Status	No information delivered.	–
Information	Displays general information about the device (name, description, provider, version, image).	–

Controller Selection

Introduction

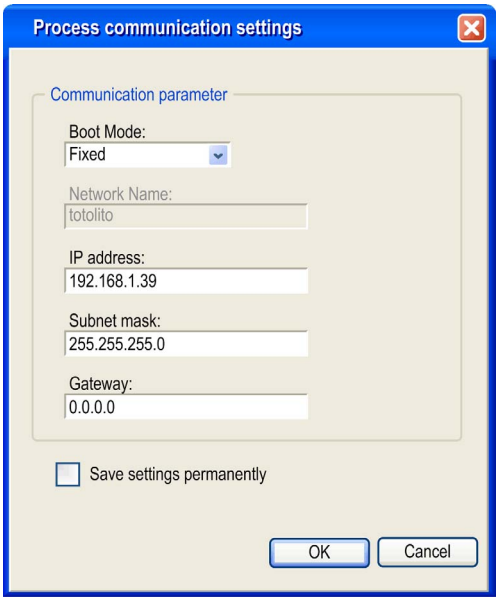
This tab allows you to manage the connection from the PC to the controller:

- 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.

Process Communication Settings

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

The **Process communication settings** window appears as shown below:



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

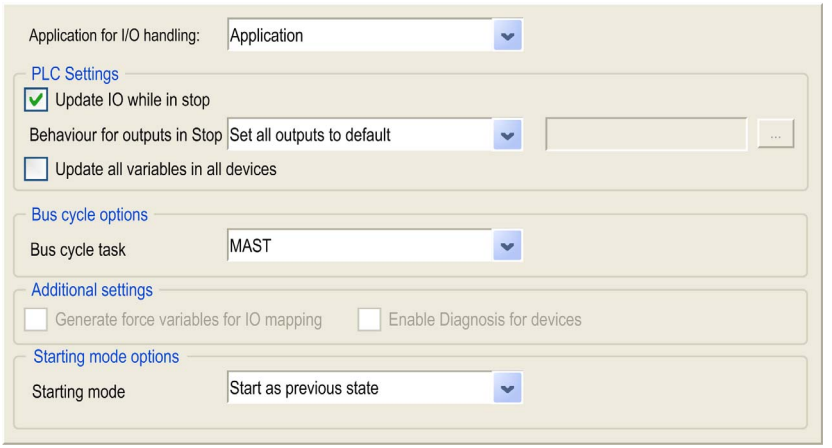
- Without the **Save settings permanently** option:
  Configure the communication parameters and click **OK**. These settings are immediately taken into account and are not kept if the controller is reset. For the next resets, the communication parameters configured into the application are taken into 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 SoMachine application.

For more information on the **Controller selection** view of the device editor, refer to the SoMachine 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**
	Update all variables in all devices	If this option is activated, then for all devices of the current controller configuration all I/O variables will get updated in each cycle of the bus cycle task. This corresponds to the option **Always update variables**, which can be set separately for each device in the **I/O Mapping** dialog.
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.

Element		Description
Additional settings	**Generate force variables for IO mapping**	Not used.
	Enable Diagnosis for devices	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 53)*. Select with this option one of these starting modes: ● Start as previous state ● Start in stop ● Start in run

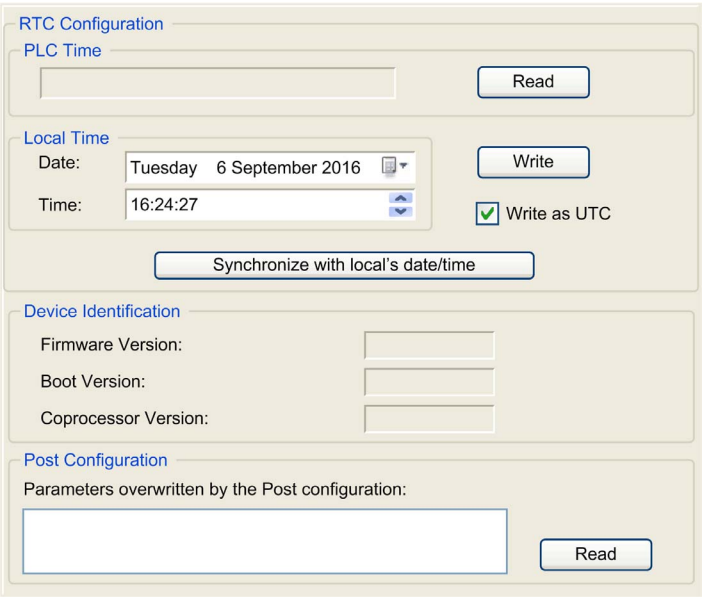
Services

Services Tab

The **Services** tab is divided in 3 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.
	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.
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 245)*.

Chapter 9
Embedded Expert I/O

Introduction

This chapter describes how to configure LMC058 Embedded Expert I/O.

The controller base provides:

- 1 Controller Power Distribution Module (CPDM)
- 1 Hardware Encoder port that can support:
  - Incremental encoder
  - SSI absolute encoder
- 2 embedded expert I/O modules (DM72F0 and DM72F1) with:
  - 5 fast inputs
  - 2 regular inputs
  - 2 fast outputs

Each embedded expert I/O module (DM72F•) can support expert functions *(see page 94)*.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	Overview	86
9.2	DM72F0 and DM72F1	89
9.3	Hardware Encoder Interface	102
9.4	Controller Power Distribution Module	105

Section 9.1
Overview

Expert I/O Overview

Introduction

The controller base provides:

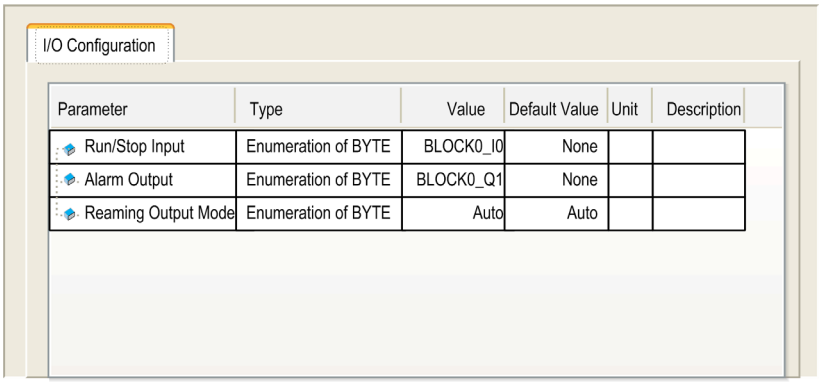
- 2 embedded expert I/O modules (DM72F0 and DM72F1) with:
  - 5 fast inputs
  - 2 regular inputs
  - 2 fast outputs
- 1 Hardware Encoder port that can support:
  - Incremental encoder
  - SSI absolute encoder
- 1 Controller Power Distribution Module (CPDM)

Each embedded expert I/O module (DM72F•) can support expert functions *(see page 94)*.

Embedded Expert I/O Configuration

To configure the expert I/Os, double-click the **Expert** node in the **Devices tree**.

This figure presents the configuration tab screenshot:



This table presents the function of the different parameters:

Parameter	Function
Run/Stop Input	Define one input to be used as Run/Stop input *(see page 87)*.
Alarm Output	Define one output to be used as alarm output *(see page 88)*.
Rearming Output Mode	Define the rearming output mode *(see page 88)*.

Run/Stop Input

This table presents the different states:

Input states	Result
State 0	Stops the controller and ignores external Run commands.
A rising edge	From the STOPPED state, initiates a start-up of an application in RUNNING state.
State 1	The application can be controlled by: ● SoMachine (Run/Stop) ● the application (Controller command) ● a network command

NOTE: Run/Stop input is managed even if the option **Update IO while in stop** is not selected in the PLC settings tab *(see page 76)*.
Inputs assigned to configured expert functions can not be configured as Run/Stop.

For further details about controller states and states transitions, refer to Controller State Diagram *(see page 53)*.

⚠ WARNING

UNINTENDED MACHINE OR PROCESS START-UP

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

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

Alarm Output

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

Outputs assigned to configured expert functions can not be configured as the Alarm output.

NOTE:
The alarm output is set to 0 when:

- a task is stopped at a breakpoint, the alarm output signals that the controller has stopped executing the application.
- an error is detected on the expert I/O (power interruption, short-circuit detection).

Rearming Output Mode

Fast outputs of DM72F• modules are push/pull technology. In the case of an error detected (short-circuit or over temperature), the output is put in tri-state and the condition is signaled by status bit (DM72F• channel IB1.0) and PLC_R.i_wLocalIOStatus *(see Modicon M258 Logic Controller, System Functions and Variables, M258 PLCSystem Library Guide)*.

Two behaviors are possible:

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

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

⚠ WARNING

UNINTENDED MACHINE START-UP

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

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

Section 9.2
DM72F0 and DM72F1

What Is in This Section?

This section contains the following topics:

Topic	Page
DM72Fx Configuration	90
Add an Expert function	94
Embedded Expert I/O Mapping	97
Event_Latch Function	100

DM72Fx Configuration

DM72Fx I/O Configuration

The DM72Fx editor allows you to configure the I/Os when they are not used by an expert function.

Step	Action
1	Double-click **Expert → DM72Fx** in the **Devices tree**.
2	Select the **I/O Configuration** tab.

User can configure the following parameters:

Parameter		Value	Unit	Description	Constraint
Ix	**Filter**	No 1.5 4(default) 12	ms	Filtering value reduces the effect of noise on a controller input.	Enabled if input is not used by expert function.
Synchronization	**Synchronized**	Yes: Enabled No: Disabled (default)	–	Minimizes jitter on outputs by delaying the write to the physical outputs until the read inputs operation of the next **Bus cycle task** starts. (The end time of a task is often less easy to predict than the start time.)	–

NOTE: When inputs are used as regular, they can be filtered by integrator filter *(see Modicon LMC058, Motion Controller, Hardware Guide)*.
When inputs are used by an expert function (Event_Latch, HSC, PWM, and so on), the corresponding lines are disabled and the filter value is over-ridden by the particular expert function.
When an output is used by an expert function, the configuration made at the DM72Fx level is ignored. Output management depends on the expert function configuration.

I/O Management

At the beginning of each task, the used %I memory variables for the inputs are updated from physical information.

At the end of each task, the used %Q memory variables values for the outputs are updated.

If **Synchronized** is disabled, the physical output is updated from the %Q memory variable value at the end of the task configured as the **Bus cycle task**.

If **Synchronized** is enabled, the physical output is updated from the %Q memory variable value at the beginning of the subsequent **Bus cycle task**.

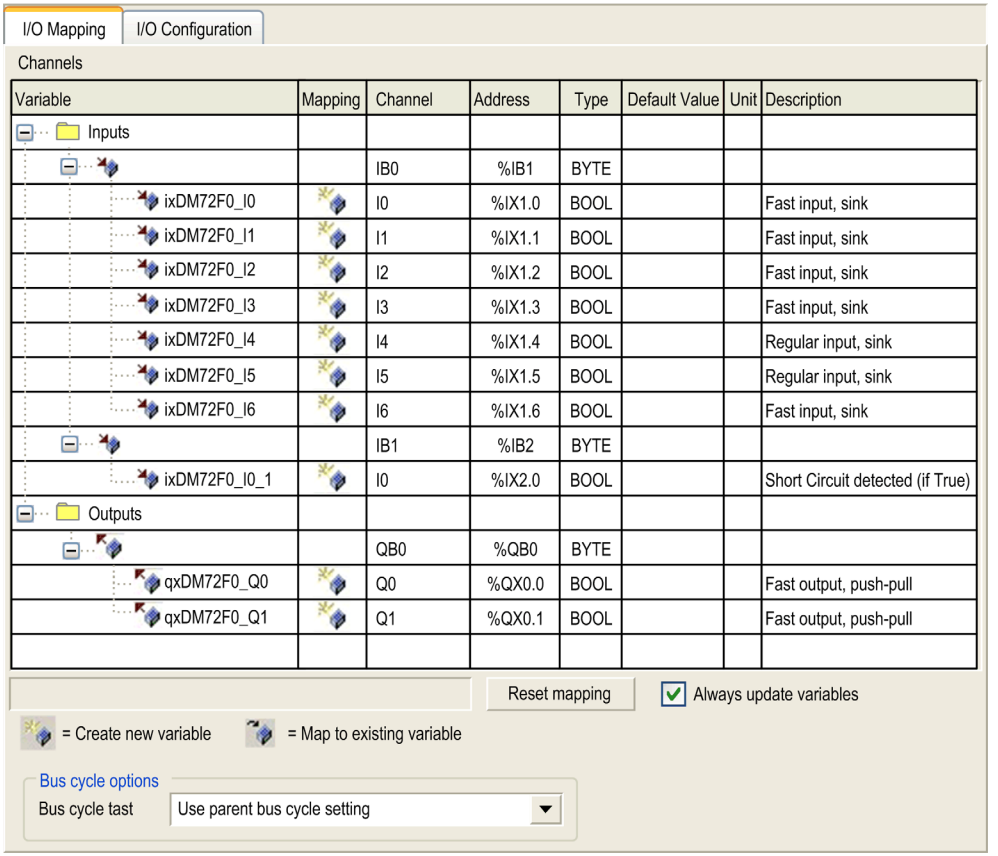
NOTE: The interest is to synchronize the effective activation of output with command or motion control on network.

For more information on **Bus cycle task**, refer to Controller PLC Settings *(see page 80)*.

DM72F• I/O Mapping

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

Step	Action
1	Double-click the **Expert → DM72Fx** in the **Devices tree**.
2	Select the **I/O Mapping** tab.



The table below describes the DM72Fx modules I/O Mapping configuration:

Channel			Type	Default value	Description
Inputs	IB0		BYTE	–	State of all inputs (bit 7 = 0, not used)
		I0	BOOL	–	State of input 0
		...			...
		I6			State of input 6
	IB1		BYTE	–	Status byte of all outputs (bits 1-7 = 0, not used)
		I0	BOOL	–	Status bit of all outputs: 0: OK 1: overload or shortcut outputs detected
Outputs	QB0		BYTE	–	Command byte of all output (bits 2-7 = 0, not used)
		Q0	BOOL	TRUE FALSE	Command bit of output 0
		Q1			Command bit of output 1
	QB1		BYTE	–	Rearming output command byte (bits 1-7 = 0, not used)
		Q0	BOOL	TRUE FALSE	Rearming output command *(see page 86)* bit

User can associate variables with listed inputs and outputs.

For more information, refer to the SoMachine Programming Guide.

Bus Cycle Options

The Bus cycle task parameter allows you to define a specific task to the DM72Fx Expert I/O. If **Use parent bus cycle** setting is selected (default value), the Bus cycle task parameter set in the Controller PLC Settings *(see page 80)* is used.

To attach a specific task, select the desired one from the selection list. The list provides the tasks currently defined.

NOTE: A cycle task is your best option for the expert I/O bus cycle task.

Add an Expert function

Introduction

Each DM72F• expert module can support expert functions. Expert functions are defined as either simple or complex. Only one type can be configured per module:

- simple functions:
  - High Speed Counter Simple
  - Event_Latch I/O
- complex functions:
  - High Speed Counter Main
  - Encoder
  - Frequency Generator (FreqGen)
  - Pulse Width Modulation (PWM)

When an I/O is not used by an expert function, it can be used as a regular I/O.

NOTE:

- When a regular input is used as Run/Stop, it can not be used by an expert function.
- When a regular output is used as Alarm, it can not be used by an expert function.

For more details, refer to Embedded expert I/O Configuration *(see page 86)*.

Adding an Expert Function

To add an Expert function (Event_Latch, HSC, PWM or Frequency Generator) to your controller, select the Expert function you want in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

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

• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

To add an Encoder function, select the **Standard Encoder** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

The following expert functions can be added:

Function	Description	Refer to...
Event_Latch	With the Event_Latch function, the Embedded Expert inputs can be configured as event or latch.	Event_Latch configuration *(see page 100)*
HSC	The HSC functions can execute fast counts of pulses from sensors, encoders, switches, etc. that are connected to dedicated fast inputs.	LMC058 HSC Library *(see Modicon LMC058 Motion Controller, High Speed Counting, LMC058 Expert I/O Library Guide)*
PWM Frequency Generator	The PWM function generates a square wave signal on dedicated output channels with a variable duty cycle. The Frequency Generator function generates a square wave signal on dedicated output channels with a fixed duty cycle (50%).	LMC058 PWM library *(see Modicon LMC058 Motion Controller, Pulse Width Modulation, LMC058 Expert I/O Library Guide)*
Encoder	The goal of this function is to connect an encoder to acquire a position. This function can be implemented on an Embedded Expert I/O interface and an Hardware Encoder interface. The Encoder can be Incremental or absolute SSI on an Hardware Encoder interface. The Embedded Expert I/O interface supports only an Incremental Encoder. You can configure a linear or rotary axis for incremental encoder.	LMC058 HSC Library *(see Modicon LMC058 Motion Controller, High Speed Counting, LMC058 Expert I/O Library Guide)*

Expert Function Assignment

Expert functions assignment according to the interface (columns exclude each other):

I/F Interface	Expert Functions					
	Simple functions: • Fast I/O: Event or latched • HSC Simple	HSC_Main	SM_Encoder	Encoder	PWM	Frequency Generator
DM72F0	Up to 4	1	1	1	1	1
DM72F1	Up to 4	1	1	1	1	1
Encoder	Not allowed	Not allowed	1	1	Not allowed	Not allowed

For more details, refer to Expert I/O Mapping *(see page 97)*.

Expert Function I/O within Regular I/O

Expert Function I/O within Regular I/O:

- Inputs can be read through memory variable standard even if configured in expert function.
- An Input can not be configured in an expert function if it has already been configured as a Run/Stop.
- An Output can not be configured in an expert function if it has already been configured as an Alarm.
- %Q will not have any impact on reflex output.
- Short-Circuit management still applies on all outputs. Status of outputs are available.
- All I/O that are not used by expert functions are available as fast or regular I/O.

When inputs are used in expert functions (Latch, HSC,…), integrator filter is replaced by anti-bounce filter *(see Modicon LMC058, Motion Controller, Hardware Guide)*. Filter value will be configured in expert function screen.

Embedded Expert I/O Mapping

I/O Mapping for Expert Functions on DM72F•

Embedded Expert I/O mapping by expert function:

		I0	I1	I2	I3	I4	I5	Q0	Q1
Event_Latch 0/4	Input	M							
Event_Latch 1/5	Input		M						
Event_Latch 2/6	Input			M					
Event_Latch 3/7	Input				M				
HSC Simple 0/4	Input A	M							
HSC Simple 1/5	Input A		M						
HSC Simple 2/6	Input A			M					
HSC Simple 3/7	Input A				M				
HSC Main 0/1	Input A	M							
	Input B		C						
	SYNC			C					
	CAP				C				
	EN					C			
	REF						C		
	Outputs							C	C
PWM 0/1	Outputs							M	
	SYNC			C					
	EN					C			
Frequency Generator 0/1	Outputs							M	
	SYNC			C					
	EN					C			
Standard Encoder	Input A	M							
	Input B		M						
	SYNC			C					
	CAP				C				
	EN					C			
	REF						C		
	Outputs							C	C

M Mandatory
C Depends on Configuration

		I0	I1	I2	I3	I4	I5	Q0	Q1
Motion Encoder	Input A	M							
	Input B		M						
	Input Z			M					
	CAP				C				
M Mandatory **C** Depends on Configuration									

NOTE: The DM72F• I6 inputs can only be configured by encoder on ENC.

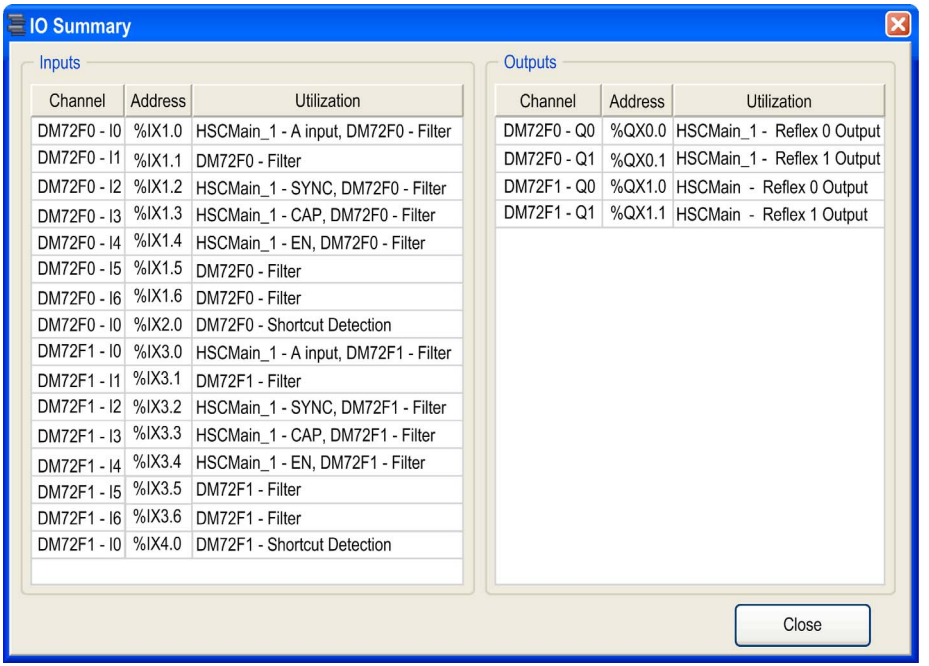
IO Summary

The **IO Summary** window displays the DM72F• I/O and the I/O used by expert functions.

The **IO Summary** window is accessible from **DM72F•** nodes:

Step	Action
1	In the **Devices tree** tab, expand the **Expert** node.
2	Right-click **DM72F•** and select **IO Summary** in context menu.

Example of IO Summary:



Event_Latch Function

Introduction

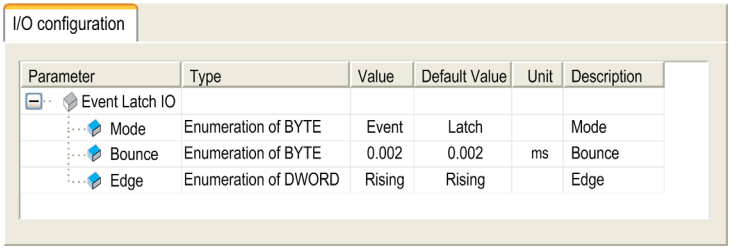
With the Event_Latch function, the Embedded Expert inputs can be configured as event or latch.

Adding a Event_Latch Function

To add an Event_Latch function, proceed as explained in Add an Expert Function *(see page 94)*.

Event_Latch Function Configuration

To configure the Event_Latch function, click the Event_Latch function:



Event_Latch inputs are used to enable event I/O or latched I/O and they are simple function added under **DM72F0** or **DM72F1** for input 0 to 3.

Event_Latch input function parameters are:

Parameter	Value	Description	Constraint
Mode	Latch (default)	Latching allows incoming pulses with duration shorter than the controller scan time to be captured and recorded. When input reaches state 1, this state is maintained until the MAST task reads the input.	Use latch inputs in MAST task only.
	Event	Event detection allows to start an event task on edge. The “External task” can be triggered by the rising edge, the falling edge, or both of the input (I0 to I3).	Maximum lead time between the input transition and the External task to start is 0.5 ms (except if a task with a greater priority is running).
Bounce (in ms)	0.002 (default) 0.004 0.012 0.04 0.12 0.4 1.2 4	Filtering value reduces the effect of bounce on a controller input.	

Parameter	Value	Description	Constraint
Edge	Rising (default) Falling Both	Define the edge detection when the event mode is selected.	In latch mode, this parameter is disabled.

NOTE: Choice of input that supports Run/Stop function is made in Expert I/O Configuration Screen *(see page 86)*.

Section 9.3
Hardware Encoder Interface

What Is in This Section?

This section contains the following topics:

Topic	Page
Hardware Encoder Interface	103
Add an Encoder	104

Hardware Encoder Interface

Introduction

The controller has a specific hardware encoder interface that can support:

- Incremental encoder
- SSI absolute encoder

Encoder function

The goal of this function is to connect an encoder to acquire a position so that it can be used as Master Axis for motion drives on CAN.

This function can be implemented on an Embedded Expert I/O interface and a hardware Encoder interface. The Encoder can be Incremental or absolute SSI on a hardware Encoder interface. The Embedded Expert I/O interface supports only an Incremental Encoder.

You can configure a linear or rotary axis for incremental encoder.

I/O mapping

Input of Embedded Expert I/O modules (DM72F•) used by standard encoder function:

	DM72F0 I6	DM72F1 I6
CAP0	C	
CAP1		C
EN	C	
REF		C

Input of Embedded Expert I/O modules (DM72F•) used by Motion encoder function:

	DM72F0 I6	DM72F1 I6
CAP0	C	
CAP1		C

C = depends on configuration

Add an Encoder

Introduction

In order to use the Encoder interface, the Modicon LMC058 Motion Controller has a specific hardware Encoder interface that can support:

- Incremental encoder
- SSI absolute encoder

Add an Encoder

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

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

• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

Configure an encoder

To configure an Encoder, refer to ENCODER Description *(see Modicon LMC058 Motion Controller, High Speed Counting, LMC058 Expert I/O Library Guide)*.

Section 9.4
Controller Power Distribution Module

Controller Power Distribution Module

Presentation

The controller power distribution module is divided into 3 power supplies:

- Power 24 Vdc expert modules
- Main power 24 Vdc (for controller, fieldbus and slice power supply)
- Power 24 Vdc I/O

There is no configuration necessary for this module.

I/O Mapping Tab

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

This table describes the controller power distribution module I/O mapping configuration:

Channel			Type	Default value	Description
Inputs	IB0		BYTE	–	State of all inputs (bits 3-7 = 0, not used)
		I0	BOOL	–	Power 24 Vdc expert modules False when 24 Vdc is applied.
		I1			Main power 24 Vdc False when 24 Vdc is applied.
		I2			Power 24 Vdc I/O False when 24 Vdc is applied.

NOTE: When all power is present, the IB0 channel = 00 hex.

Chapter 10
TM5 Modules

Introduction

The TM5 Bus contains:

- Embedded I/O modules
- TM5 expansion modules

This chapter describes how to configure the TM5 Bus.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
10.1	TM5 Manager Configuration	108
10.2	Embedded Regular I/O Modules Configuration	111
10.3	TM5 Expansion Modules Configuration	126

Section 10.1
TM5 Manager Configuration

TM5 Manager Configuration

TM5 Manager Configuration

To configure the TM5 Manager, proceed as follows:

Step	Action
1	In the **Devices tree**, expand the **TM5** node.
2	Double-click the **TM5_Manager** node. **Result**: The TM5 Manager configuration window is displayed.
3	Select the **I/O Configuration** tab.

Parameters of the I/O Configuration:

Parameter	Value	Default Value	Unit	Description
Bus Cycle Time	0.5 ms 1 ms 2 ms 3 ms 4 ms 5 ms	1 ms	ms	Expansion Bus Cycle Time
Maximum number of physical slots	Number of Embedded modules...250	250	-	Maximum number of modules on the expansion bus.
Name of FW repository	Not configurable	-	-	This parameter indicates the Flash memory repository for the modules firmware.
Maximum bus length in meters (feet)	1...2500 (3.28...8202)	100 (328)	m	Total cable length used on the expansion bus.

NOTE: For more information about the maximum capacities of your system, refer to the TM5 / TM7 System Planning and Installation Guide *(see Modicon TM5 / TM7 Flexible System, System Planning and Installation Guide)*.

Bus Cycle Time

Bus Cycle Time can be configured from 0.5 to 5 ms. Very fast cycles reduce the idle time for handling monitoring, diagnostics and acyclic commands.

The Bus Cycle Time follows 2 rules:

- Be longer than the greatest **Minimum Cycle Time** of any expansion module or block in the configuration.
- Be long enough to permit the data exchange with all the modules and the blocks.

Minimum Cycle Time

The Minimum Cycle Time of a module or of a block is the time needed by the module or the block to perform I/O management. If the Bus Cycle Time is shorter than this minimum value, the module will not operate properly.

Minimum I/O Update Time

The Minimum I/O Update Time of a module or block is the time needed by the module or block to update I/O on the bus. If the Bus Cycle Time is shorter than this minimum value, the I/O will be updated on the bus at the next Bus Cycle Time.

I/O Management

At the beginning of each task, the %I memory variable for the inputs used in the task is updated with the physical state of the input.

At the end of each task, the used %Q memory variable value for the outputs is updated.

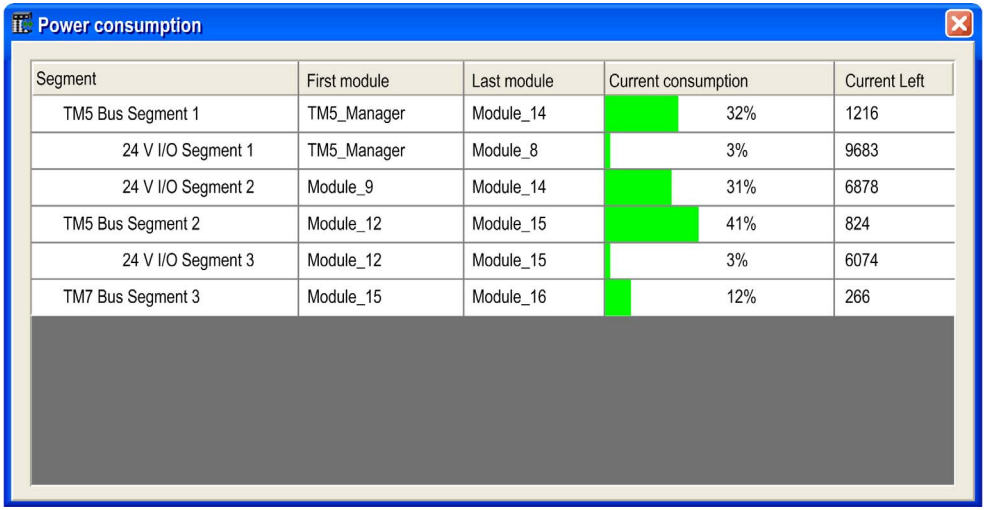
On the next bus cycle after the end of the task configured as the **Bus cycle task**, the physical output is updated from the %Q memory variable value.

For more details on **Bus cycle task**, refer to the controller **PLC settings** tab.

Power Consumption

To display the estimated power consumption of the expansion modules:

Step	Action
1	Right-click the **TM5_Manager** node of the **Device tree**.
2	Select **Power consumption**.



NOTE: The current consumption figures presented by the **Power consumption** function are based on assumed values, and not on actual current measurements. The assumed values for the outputs are based on classic loads but can be adjusted using the 24 Vdc I/O segment external current setting in the I/O Configuration *(see Modicon TM5, Expansion Modules Configuration, Programming Guide)* tab of each module. The assumptions for input signals are based on known internal loads and are therefore not modifiable. While the use of the **Power consumption** function to test the power budget is required, it is no substitute for actual and complete system testing and commissioning. Refer to the TM5 / TM7 System Planning and Installation Guide *(see Modicon TM5 / TM7 Flexible System, System Planning and Installation Guide)*.

Section 10.2
Embedded Regular I/O Modules Configuration

Introduction

The following section describes the configuration of the Embedded Regular I/O Modules.

What Is in This Section?

This section contains the following topics:

Topic	Page
Embedded Regular I/O Configuration	112
DI12DE Embedded Regular I/O Module	114
DO12TE Embedded Regular I/O Module	115
AI4LE Embedded Regular I/O Module	118

Embedded Regular I/O Configuration

Introduction

This table shows the embedded regular I/O modules and their associated controller reference:

Controller	Embedded Regular I/O	Description
LMC058LF42	DI12DE	12 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
	DO12TE	12 Outputs 24 VDC / 0.5 A
LMC058LF424	DI12DE	12 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
	DO12TE	12 Outputs 24 VDC / 0.5 A
	AI4LE	4 Inputs ±10 V / 0... 20 mA / 4...20 mA

Embedded Regular I/O Configuration

To configure embedded regular I/O, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **MyController → TM5 → TM5_Manager → Embedded Bus → Module_x**.
2	Select the **I/O Configuration** tab.

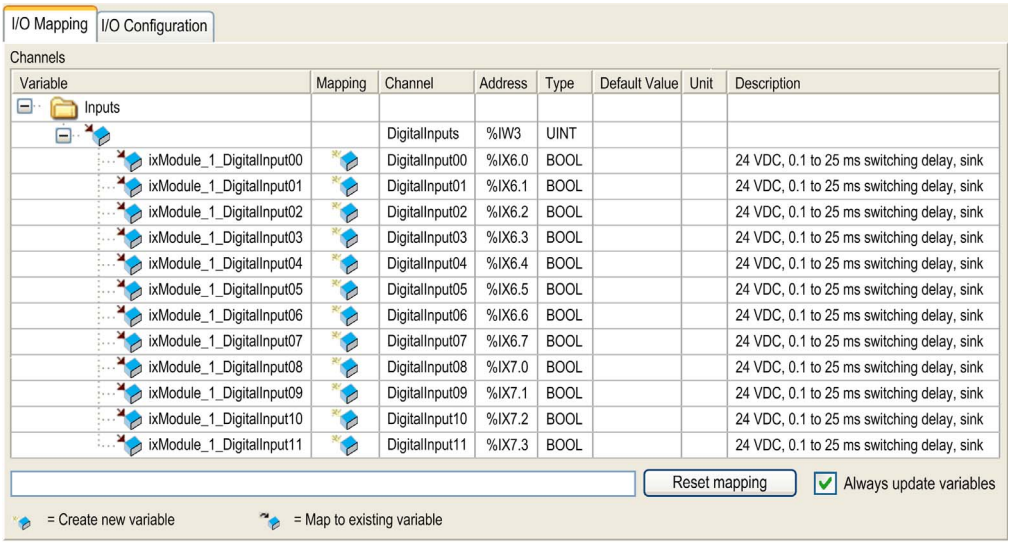
I/O Configuration Tab Description

The **I/O Configuration** tab contains the following columns:

Column	Description	Editable
Parameter	Parameter name	No
Type	Parameter data type	No
Value	Value of the parameter	If the parameter is editable, an edit frame can be opened by double-clicking.
Default Value	Default parameter value	No
Unit	Unit value of the parameter	No
Description	Short description of the parameter	No

I/O Mapping Tab Description

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



The **I/O Mapping** tab contains the following columns:

Column	Description
Variable	Lets you map the channel on a variable. Double-click the icon to enter the variable name. If it is a new variable, the variable is created. New variables are automatically created on each channel according to **Automatic I/O mapping** project option settings. It is also possible to map an existing variable with the variables **Input Assistant** by clicking the **...** button.
Mapping	Indicates if the channel is mapped on a new variable or an existing variable
Channel	Name of the channel of the device
Address	Address of the channel
Type	Data type of the channel
Default Value	Value taken by the Output when the controller is in a STOPPED state *(see page 51)*. Double-click to change the default value.
Unit	Unit of the channel value
Description	Description of the channel

DI12DE Embedded Regular I/O Module

Introduction

The DI12DE embedded regular I/O module is a 24 Vdc Digital Inputs module with 12 inputs.

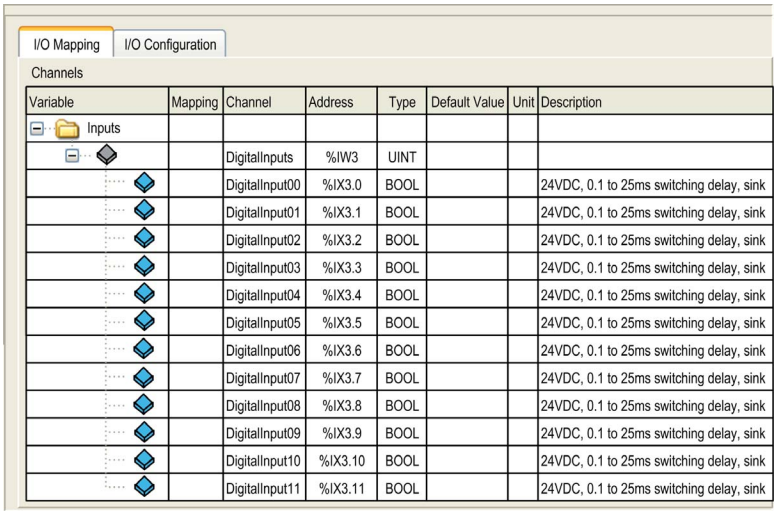
I/O Configuration Tab

The table below describes the module parameters configuration:

Parameter	Value	Default Value	Unit	Description
Input filter	0...250	10 (1 ms)	0.1 ms	Specifies the filter time of digital inputs

I/O Mapping Tab

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



Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
Inputs							
		DigitalInputs	%IW3	UINT			
		DigitalInput00	%IX3.0	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput01	%IX3.1	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput02	%IX3.2	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput03	%IX3.3	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput04	%IX3.4	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput05	%IX3.5	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput06	%IX3.6	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput07	%IX3.7	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput08	%IX3.8	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput09	%IX3.9	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput10	%IX3.10	BOOL			24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput11	%IX3.11	BOOL			24VDC, 0.1 to 25ms switching delay, sink

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 113)*.

The table below describes the DI12DE I/O Mapping configuration:

Variable		Channel	Type	Default value	Description
Inputs	**DigitalInputs**		UINT	-	State of all inputs (bits 13...16 = 0, not used)
		DigitalInput00	BOOL	-	State of input 0
		...			...
		DigitalInput11			State of input 11

DO12TE Embedded Regular I/O Module

Introduction

The DO12TE embedded regular I/O module is a 24 Vdc Digital Outputs module with 12 transistor outputs.

I/O Configuration Tab

This table describes the module parameters configuration:

Parameter	Value	Default Value	Unit	Description
Output status information	On Off	On		Additional output status information. **On**: the **StatusDigitalOutputs** word is added to the **I/O Mapping** tab.
24 V I/O segment external current	0...500	100	mA	24 V I/O segment external current on TM5 power bus supply

I/O Mapping Tab

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

I/O Mapping | I/O Configuration

Channels

Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
Inputs							
		StatusDigitalOut..	%IW4	UINT			
		StatusDigitalOut..	%IX8.0	BOOL			Status digital out...
		StatusDigitalOut..	%IX8.1	BOOL			Status digital out...
		StatusDigitalOut..	%IX8.2	BOOL			Status digital out...
		StatusDigitalOut..	%IX8.3	BOOL			Status digital out...
		StatusDigitalOut..	%IX8.4	BOOL			Status digital out...
		StatusDigitalOut..	%IX8.5	BOOL			Status digital out...
		StatusDigitalOut..	%IX8.6	BOOL			Status digital out...
		StatusDigitalOut..	%IX8.7	BOOL			Status digital out...
		StatusDigitalOut..	%IX9.0	BOOL			Status digital out...
		StatusDigitalOut..	%IX9.1	BOOL			Status digital out...
		StatusDigitalOut..	%IX9.2	BOOL			Status digital out...
		StatusDigitalOut..	%IX9.3	BOOL			Status digital out...
Outputs							
		DigitalOutputs	%QW3	UINT			
		DigitalOutput00	%QX6.0	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput01	%QX6.1	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput02	%QX6.2	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput03	%QX6.3	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput04	%QX6.4	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput05	%QX6.5	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput06	%QX6.6	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput07	%QX6.7	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput08	%QX7.0	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput09	%QX7.1	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput10	%QX7.2	BOOL			24 VDC / 0.5 A, ...
		DigitalOutput11	%QX7.3	BOOL			24 VDC / 0.5 A, ...

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 113)*.

This table describes the I/O Mapping configuration:

Variable	Channel		Type	Default Value	Description
Inputs	**StatusDigitalOutputs**		UINT	-	Status word of all outputs
		StatusDigitalOutput00	BOOL	-	Status bit associated to each output: ● 0: OK ● 1: error detected
		...			
		StatusDigitalOutput11			
Outputs	**DigitalOuputs**		UINT	-	Command word of all outputs
		DigitalOuput00	BOOL	TRUE FALSE	Command bit of output 0
		...			...
		DigitalOuput11			Command bit of output 11

AI4LE Embedded Regular I/O Module

Introduction

The AI4LE embedded regular I/O module is a ±10 Vdc/0...20 mA/4...20 mA analog input module with 4 inputs.

If you have wired your input for a voltage measurement, and you configure SoMachine for a current type of measurement (or vice-versa), you may permanently damage the electronic module.

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

I/O Configuration Tab

The table below describes the modules parameters configuration:

Parameter		Value	Default Value	Description
General	**Lower limit**	-32768...32767	-32767	Specifies the lower measurement limit *(see page 119)*
	Upper limit	-32768...32767	32767	Specifies the upper measurement limit *(see page 119)*
	Input filter	Off level 2 level 4 level 8 level 16 level 32 level 64 level 128	Off	Definition of the filter level *(see page 120)*
	Input limitation	Off 16383 8191 4095 2047 1023 511 255	Off	Specifies the limitation of input ramp *(see page 122)* **NOTE:** Parameter available if an input filter is selected.
Channel 00	**Channel type**	±10 V 0 to 20 mA 4 to 20 mA	±10 V	Specifies the channel type

Parameter		Value	Default Value	Description
Channel 01	**Channel type**	±10 V 0 to 20 mA 4 to 20 mA	±10 V	Specifies the channel type
Channel 02	**Channel type**	±10 V 0 to 20 mA 4 to 20 mA	±10 V	Specifies the channel type
Channel 03	**Channel type**	±10 V 0 to 20 mA 4 to 20 mA	±10 V	Specifies the channel type

Analog Inputs

The input status is registered with a fixed offset with respect to the network cycle and is transferred in the same cycle.

Input Filter

The electronic module is equipped with a configurable Input filter. Filtering is automatically deactivated for shorter cycle times (t < 500 µs).

If the Input filter is active, then all of the input channels are repeatedly scanned with millisecond-level resolution. The time offset between the channels is 200 µs. The conversion of the physical signal at the input to the filtered signal takes place asynchronously to the Bus Cycle Time. Refer to Cycle time and I/O update time *(see Modicon TM5, Expansion Modules Configuration, Programming Guide)*

Limit values

You can define 2 different type of limits:

- **Lower limit**
- **Upper limit**

The **Lower limit** value range is between -32768 to 32767. This value is applied on every channel of the module being configured.

NOTE: the **Lower limit** cannot be greater than the **Upper limit**.

Channel Configuration	Digital Value Behavior	Comments
± 10V	-10 V = -32768 +10 V = +32767	If the **Lower limit** value is configured between -32768 and +32767, the digital value is limited to the **Lower limit** value.
0 to 20 mA	0 mA = 0 20 mA = +32767	If the **Lower limit** value is configured between -32768 and 0, the digital value is limited to 0. If the **Lower limit** value is configured between 0 and 32 767, the digital value is limited to the **Lower limit** value.

Channel Configuration	Digital Value Behavior	Comments
4 to 20 mA	0 mA = -8192 4 mA = 0 20 mA = +32767	If the **Lower limit** is configured between -32768 and -8192, the digital value is limited to -8192. If the **Lower limit** is configured between -8192 and 32767, the digital value is limited to the **Lower limit** value.

The **Upper limit** value range is between -32768 to 32767. This value is applied on every channel of the module being configured.

NOTE: The **Upper limit** value cannot be less than the **Lower limit** value.

Channel Configuration	Digital Value Behavior	Comments
± 10V	-10 V = -32768 +10 V = +32767	If the **Upper limit** value is configured between -32768 and +32767, the digital value is limited to the **Upper limit** value.
0 to 20 mA	0 mA = 0 20 mA = +32767	If the **Upper limit** value is configured between -32768 and 0, the digital value stays at 0; hence, set the **Upper limit** value to a positive value. If the **Upper limit** value is configured between 1 and +32767, the digital value is limited to the **Upper limit** value.
4 to 20 mA	0 mA = -8192 4 mA = 0 20 mA = +32767	If the **Upper limit** value is configured between -32768 and -8192, the digital value is limited to -8192. If the **Upper limit** value is configured between -8192 and 32767, the digital value is limited to the **Upper limit** value.

Filter Level

The input value is evaluated according to the filter level. An input ramp limitation can then be applied using this evaluation.

Formula for the evaluation of the input value:

$$Value_{new} = Value_{old} - \frac{Value_{old}}{Filterlevel} + \frac{Inputvalue}{Filterlevel}$$

Adjustable filter levels:

Filter Level
Filter switched off
Filter level 2
Filter level 4
Filter level 8
Filter level 16

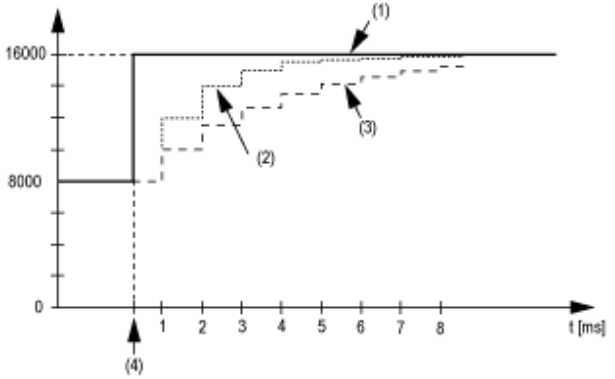
Filter Level
Filter level 32
Filter level 64
Filter level 128

The following examples show the function of the filter level based on an input jump and a disturbance.

Example 1: The input value makes a jump from 8,000 to 16,000. The diagram displays the evaluated value with the following settings:

Input ramp limitation = 0

Filter level = 2 or 4

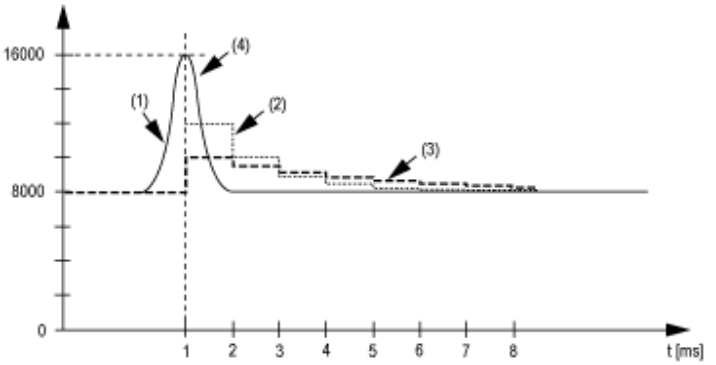


1 Input value.
2 Evaluated value: Filter level 2.
3 Evaluated value: Filter level 4.
4 Input jump.

Example 2: A disturbance is imposed on the input value. The diagram shows the evaluated value with the following settings:

Input ramp limitation = 0

Filter level = 2 or 4



1 Input value.
2 Evaluated value: Filter level 2.
3 Evaluated value: Filter level 4.
4 Disturbance (Spike).

Input Ramp Limitation

Input ramp limitation can only take place when a filter is used. Input ramp limitation is executed before filtering takes place.

The amount of the change in the input value is checked to ensure the specified limits are not exceeded. If the values are exceeded, the adjusted input value is equal to the old value ± the limit value.

This table shows the adjustable limit values:

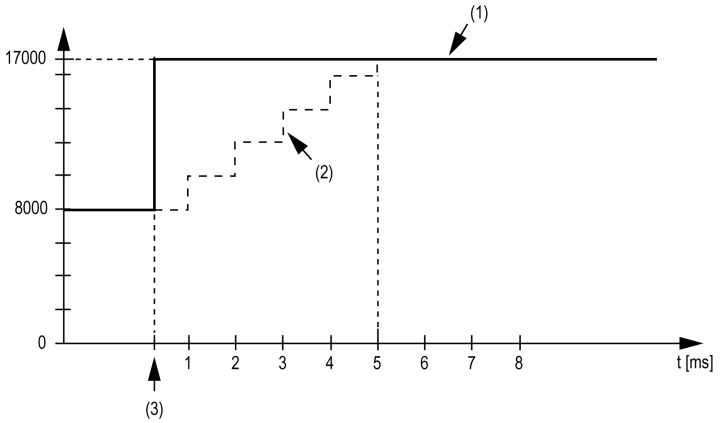
Limit Value
The input value is used without limitation.
3FFF hex = 16383
1FFF hex = 8191
0FFF hex = 4095
07FF hex = 2047
03FF hex = 1023
01FF hex = 511
00FF hex = 255

The input ramp limitation is well suited for suppressing disturbances (spikes). The following examples show the function of the input ramp limitation based on an input jump and a disturbance.

Example 1: The input value makes a jump from 8,000 to 17,000. The diagram displays the adjusted input value for the following settings:

Input ramp limitation = 4 = 07FF hex = 2047

Filter level = 2

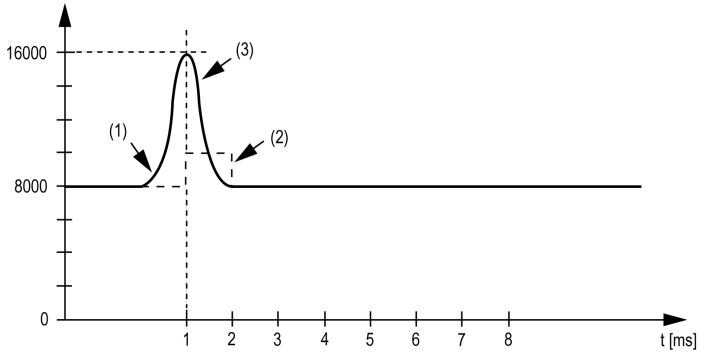


1 Input value.
2 Internal adjusted input value before filter.
3 Input jump.

Example 2: A disturbance is imposed on the input value. The diagram shows the adjusted input value with the following settings:

Input ramp limitation = 4 = 07FF hex = 2047

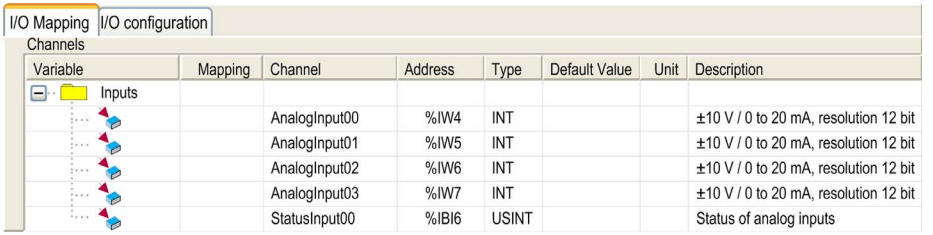
Filter level = 2



1 Input value.
2 Internal adjusted input value before filter.
3 Disturbance (Spike).

I/O Mapping Tab

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



For further generic descriptions, refer to I/O Mapping Tab Description *(see page 113)*.

This table describes the I/O Mapping configuration:

Variable	Channel	Type	Default Value	Description
Inputs	**AnalogInput00**	INT	-	Current value of the input 0
	...			...
	AnalogInput03			Current value of the input 3
	StatusInput00	USINT	-	Status of analog input channels (see description below)

Status Input Register

The **StatusInput** byte describes the status of each input channel:

Bit	Description	Bits value
0-1	Channel 0 status	00: No detected error 01: Below lower limit value[1] 10: Above upper limit value 11: Wire break
2-3	Channel 1 status	
4-5	Channel 2 status	
6-7	Channel 3 status	

1
Default setting: The input value has a lower limit. Underflow monitoring is, therefore, not necessary.
After lower limit value changes: The input value is limited to the set value. The status bit is set when the lower limit value is passed.

Cycle Time and I/O Update Time

This table gives the module characteristics allowing the TM5 Bus Cycle Time configuration:

Characteristic	Value	
	Without Filter	With Filter
Minimum Cycle Time	100 µs	500 µs
Minimum I/O update time	300 µs	1 ms

For further information, refer to TM5 Manager Configuration *(see page 108)*.

Section 10.3
TM5 Expansion Modules Configuration

Introduction

This section describes the configuration of the TM5 Expansion Modules.

What Is in This Section?

This section contains the following topics:

Topic	Page
TM5 Expansion Modules General Description	127
TM5 PCI Expansion Modules General Description	132
TM7 Expansion Modules General Description	133

TM5 Expansion Modules General Description

Introduction

The range of expansion modules includes:

- TM5 Compact I/O modules with integrated electronic modules
- TM5 Digital I/O modules
- TM5 Analog I/O modules
- TM5 Expert I/O modules
- TM5 Transmitter - Receiver modules
- TM5 Power distribution modules
- TM5 Common distribution modules
- TM5 Dummy modules

Compact, digital, and analog input modules convert measured values (voltages, currents) into numerical values that can be processed by the controller.

Compact, digital, and analog output modules convert controller-internal numerical values into voltages or currents.

Expert modules are used for counting. They use either a Synchronous Serial Interface (SSI) encoder, incremental encoder, or event counting.

The transmitter and receiver modules handle the communication between remote modules via expansion bus cables.

Power distribution modules are used to manage the power supply for the various I/O modules.

Common distribution modules provide 0 Vdc and/or 24 Vdc terminal connections for the 24 Vdc I/O power segment(s) integrated into the bus bases, which expand the wiring possibilities for sensors and actuators.

The dummy module is a non-functional module. This module is used to separate modules which have specific thermal or EMC requirements, or as a placeholder for later system expansion.

The communication module is used to connect complex devices of the TM5 System. This communication module can only be used with the TM5NS31 sercos interface module.

Compact I/O Expansion Features

Reference	Number of Channels	Voltage/Current
TM5C24D18T	24 digital inputs	24 Vdc / 3.75 mA
	18 digital outputs	24 Vdc / 0.5 A
TM5C12D8T	12 digital inputs	24 Vdc / 3.75 mA
	8 digital outputs	24 Vdc / 0.5 A
TM5C24D12R	24 inputs	24 Vdc / 3.75 mA
	12 relays NO contact	24 Vdc / 230 Vac 2 A

Reference	Number of Channels	Voltage/Current
TM5CAI8O8VL	8 analog inputs	-10...+10 Vdc
	8 analog outputs	-10...+10 Vdc
TM5CAI8O8CL	8 analog inputs	0...20 mA / 4...20 mA
	8 analog outputs	0...20 mA
TM5CAI8O8CVL	4 analog inputs	-10...+10 Vdc
	4 analog inputs	0...20 mA / 4...20 mA
	4 analog outputs	-10...+10 Vdc
	4 analog outputs	0...20 mA
TM5C12D6T6L	12 digital inputs	24 Vdc / 3.75 mA
	6 digital outputs	24 Vdc / 0.5 A
	4 analog inputs	-10...+10 Vdc 0...20 mA/4...20 mA
	2 analog outputs	-10...+10 Vdc 0...20 mA

Digital I/O Expansion Features

Reference	Number of Channels	Voltage/Current
TM5SDI2D	2 inputs	24 Vdc / 3.75 mA
TM5SDI2DF	2 fast inputs	24 Vdc / 10.5 mA
TM5SDI4D	4 inputs	24 Vdc / 3.75 mA
TM5SDI6D	6 inputs	24 Vdc / 3.75 mA
TM5SDI12D	12 inputs	24 Vdc / 3.75 mA
TM5SDI16D	16 inputs	24 Vdc / 2.68 mA
TM5SDI2A	2 inputs	100...240 Vac
TM5SDI4A	4 inputs	100...240 Vac
TM5SDI6U	6 inputs	100...120 Vac
TM5SDO2T	2 outputs	24 Vdc / 0.5 A
TM5SDO4T	4 outputs	24 Vdc / 0.5 A
TM5SDO6T	6 outputs	24 Vdc / 0.5 A
TM5SDO12T	12 outputs	24 Vdc / 0.5 A
TM5SDO16T	16 outputs	24 Vdc / 0.5 A
TM5SDO4TA	4 outputs	24 Vdc / 2 A
TM5SDO8TA	8 outputs	24 Vdc / 2 A
TM5SDO2R	2 relays C/O contact	30 Vdc / 230 Vac 5 A

Reference	Number of Channels	Voltage/Current
TM5SDO4R	4 relays NO contact	30 Vdc / 230 Vac 5 A
TM5SDO2S	2 outputs	230 Vac / 1 A
TM5SDM12DT	8 inputs	24 Vdc / 7 mA
	4 outputs	24 Vdc / 0.5 A
TM5SMM6D2L	4 digital inputs	24 Vdc / 3.3 mA
	2 digital outputs	24 Vdc / 0.5 A
	1 analog input	-10...+10 Vdc 0...20 mA / 4...20 mA
	1 analog output	-10...+10 Vdc 0...20 mA

Analog I/O Expansion Features

Reference	Number of Channels	Voltage/Current
TM5SAI2L	2 inputs	-10...+10 Vdc 0...20 mA / 4...20 mA
TM5SAI4L	4 inputs	-10...+10 Vdc 0...20 mA / 4...20 mA
TM5SAI2H	2 inputs	-10...+10 Vdc 0...20 mA
TM5SAI4H	4 inputs	-10...+10 Vdc 0...20 mA
TM5SAO2L	2 outputs	-10...+10 Vdc 0...20 mA
TM5SAO2H	2 outputs	-10...+10 Vdc 0...20 mA
TM5SAO4L	4 outputs	-10...+10 Vdc 0...20 mA
TM5SAO4H	4 outputs	-10...+10 Vdc 0...20 mA

Temperature Analog Expansion Features

Reference	Number of Channels	Sensor Type
TM5SAI2PH	2 inputs	PT100/1000
TM5SAI4PH	4 inputs	PT100/1000
TM5SAI2TH	2 inputs	Thermocouple J, K, N, S
TM5SAI6TH	6 inputs	Thermocouple J, K, N, S

Analog Strain Gauge Input Electronic Module Features

Reference	Number of Channels	Sensor Type
TM5SEAISG	1 input	Full-bridge strain gauge

Expert Expansion Features

Reference	Number of Channels	Encoder Inputs
TM5SE1IC02505	1	5 Vdc Symmetrical
TM5SE1IC01024	1	24 Vdc Asymmetrical
TM5SE2IC01024	2	24 Vdc Asymmetrical
TM5SE1SC10005	1	5 Vdc Symmetrical

Transmitter-Receiver Expansion Features

Reference	Modules Description
TM5SBET1	TM5 data transmitter electronic module.
TM5SBET7	TM5 data transmitter electronic module. It also distributes power to the TM7 bus.
TM5SBER2	TM5 data receiver electronic module. It also distributes power to the TM5 bus and to the 24 Vdc I/O power segment.

Power Distribution Expansion Features

Reference	Modules Description
TM5SPS1	24 Vdc I/O power segment supply
TM5SPS1F	24 Vdc I/O power segment supply with integrated fuse
TM5SPS2	24 Vdc I/O power segment supply and TM5 bus supply
TM5SPS2F	24 Vdc I/O power segment supply with integrated fuse and TM5 bus supply

Common Distribution Expansion Features

Reference	Number of Channels	Voltage
TM5SPDG12F	12	24 Vdc
TM5SPDD12F	12	24 Vdc
TM5SPDG5D4F	2 x 5	0 Vdc - 24 Vdc
TM5SPDG6D6F	2 x 6	0 Vdc - 24 Vdc

Dummy Expansion Features

Reference	Number of Channels	Voltage
TM5SD000	–	–

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 will no longer function while the embedded I/O that may be present in your controller will continue to operate.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

To verify if the hardware and the software configuration match, use the GVL **TM5_Module_R** regularly to monitor the expansion bus status.

Adding a TM5 Expansion Module

Refer to the TM5 Expansion Modules Configuration Programming Guide.

TM5 PCI Expansion Modules General Description

Introduction

The controller accepts the following PCI Expansion Modules:

Reference	Description
TM5PCRS2	Serial Line RS232
TM5PCRS4	Serial Line RS485
TM5PCDPS	Profibus DP Slave

Refer to the documentation of your M258 Logic Controller *(see page 15)* or your LMC058 Motion Controller *(see page 15)* to find out whether the controller type you are using is equipped with a PCI slot.

NOTE:
For information on compatibility rules between PCI communication electronic modules and controllers, refer to:

- Modicon M258 Logic Controller Hardware Guide
- Modicon LMC058 Motion Controller Hardware Guide

Adding a PCI Expansion Module

To add a PCI Expansion Module to your configuration, refer to Modicon TM5 PCI Modules Configuration Programming Guide.

TM7 Expansion Modules General Description

Introduction

The range of expansion I/O includes:

- TM7 Digital I/O blocks
- TM7 Analog I/O blocks
- TM7 Power Distribution Blocks

Digital or analog input blocks convert measured values (voltages, currents) into numerical values which can be processed by the controller.

Digital or analog output blocks convert controller internal numerical values into voltages or currents.

Analog temperature blocks convert the temperature measurement values into number values which can be processed by the controller. For temperature measurements, the temperature block returns the measured value using 0.1 °C *(0.18 °F)* steps.

The Power Distribution Blocks PDB are used to manage the power supply for the various I/O blocks. The PDB feeds the TM7 power bus.

NOTE: The TM7 I/O blocks are associated with power cables, TM7 bus cables and I/O cables.

Expansion Block Features

This table lists the digital blocks described in this programming guide:

Reference	Number of Channels	Voltage/Current	Refer To
TM7BDI8B	8 inputs	24 Vdc / 7 mA	TM7BDI8B, TM7BDI16A and TM7BDI16B
TM7BDI16B	16 inputs	24 Vdc / 7 mA	TM7BDI8B, TM7BDI16A and TM7BDI16B
TM7BDI16A	16 inputs	24 Vdc / 7 mA	TM7BDI8B, TM7BDI16A and TM7BDI16B
TM7BDO8TAB	8 outputs	24 Vdc / 2 A	TM7BDO8TAB
TM7BDM8B[1]	8 inputs 8 outputs	24 Vdc / 4.4 mA 24 Vdc / 0.5 A	TM7BDM8B, TM7BDM16A and TM7BDM16B
TM7BDM16A[1]	16 inputs 16 outputs	24 Vdc / 4.4 mA 24 Vdc / 0.5 A	TM7BDM8B, TM7BDM16A and TM7BDM16B
TM7BDM16B[1]	16 inputs 16 outputs	24 Vdc / 4.4 mA 24 Vdc / 0.5 A	TM7BDM8B, TM7BDM16A and TM7BDM16B
[1] I/O is individually configurable as either input or output			

This table lists the analog blocks described in this programming guide:

Reference	Number of Channels	Voltage/Current	Refer To
TM7BAI4VLA	4 inputs	-10...+10 Vdc	TM7BAI4VLA
TM7BAI4CLA	4 inputs	0...20 mA	TM7BAI4CLA
TM7BAO4VLA	4 outputs	-10...+10 Vdc	TM7BAO4VLA
TM7BAO4CLA	4 outputs	0...20 mA	TM7BAO4CLA
TM7BAM4VLA	2 inputs 2 outputs	-10...+10 Vdc -10...+10 Vdc	TM7BAM4VLA
TM7BAM4CLA	2 inputs 2 outputs	0...20 mA 0...20 mA	TM7BAM4CLA

This table lists the analog temperature input blocks described in this programming guide:

Reference	Number of Channels	Sensor Type	Refer To
TM7BAI4TLA	4 inputs	PT100/1000 KTY10-6/84-130	TM7BAI4TLA
TM7BAI4PLA	4 inputs	Thermocouple J,K,S	TM7BAI4PLA

This table lists the power distribution block described in this programming guide:

Reference	Description	Refer To
TM7SPS1A	TM7 Power Distribution Block	TM7SPS1A

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 will no longer function while the embedded I/O that may be present in your controller will continue to operate.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

To verify if the hardware and software configuration match, use the **GVL TM5_Module_R** function to monitor the expansion bus status.

Adding a TM7 Expansion Module

To add a TM7 expansion module, refer to the Modicon TM7 Expansion Blocks Configuration Programming Guide.

Chapter 11
Ethernet Configuration

Introduction

This chapter describes how to configure the Ethernet network interface of the Modicon LMC058 Motion Controller.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
11.1	Ethernet Services	138
11.2	Firewall Configuration	174
11.3	Ethernet Optional Devices	183

Section 11.1
Ethernet Services

What Is in This Section?

This section contains the following topics:

Topic	Page
Presentation	139
IP Address Configuration	141
Modbus TCP Client/Server	146
Web Server	148
FTP Server	169
FTP Client	172
SNMP	173

Presentation

Ethernet Services

The controller supports the following services:

- Modbus TCP Server *(see page 146)*
- Modbus TCP Client *(see page 146)*
- Web Server *(see page 148)*
- FTP Server *(see page 169)*
- SNMP *(see page 173)*
- EtherNet/IP Device *(see page 185)*
- Modbus TCP Slave Device *(see page 207)*

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
SoMachine Protocol (SoMachine software, trace, Web visualization, HMI devices)	8

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).

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 SoMachine Programming Guide.

IP Address Configuration

Introduction

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

- address assignment by DHCP server
- address assignment by BOOTP server
- fixed IP address
- post configuration file *(see page 245)*. If a post configuration file exists, this assignment method has priority over the others.

The IP address can be changed dynamically:

- via the Controller Selection tab in SoMachine.
- via the **changeIPAddress** function block *(see page 275)*.

NOTE: If the attempted addressing method is unsuccessful, the controller will start using a default IP address *(see page 144)* 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

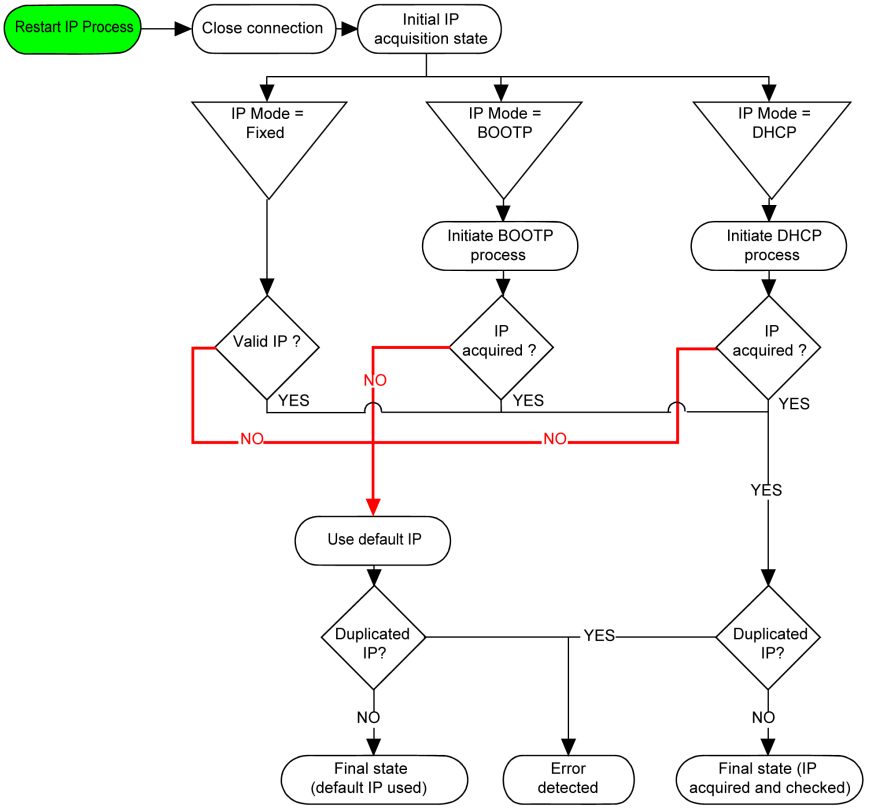
- Verify that there is only one master controller configured on the network or remote link.
- Verify that all devices have unique addresses.
- Obtain your IP address from your system administrator.
- Confirm that the IP address of the device is unique before placing the system into service.
- Do not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

Address Management

The different types of address systems for the controller are shown in this diagram:



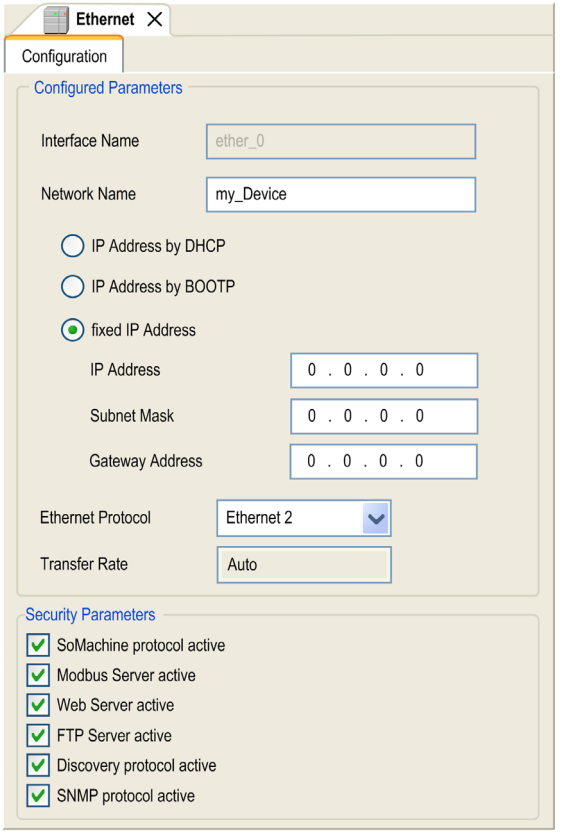
NOTE: If a device programmed to use the DHCP or BOOTP addressing methods is unable to contact its respective server, the controller uses the default IP address. It will, however, constantly repeat its request.

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**:



The configured parameters are explained as below:

Configured Parameters	Description
Interface Name	Name of the network link.
Network Name	Used as device name to retrieve IP address through DHCP, maximum 16 characters.
IP Address by DHCP	IP address is obtained via DHCP.
IP Address by BOOTP	IP address is obtained via BOOTP.
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	Transfer rate and direction on the bus are automatically configured.

Default IP Address

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

The last 2 fields in the default IP address are composed of the decimal equivalent of the last 2 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 always 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.

Security Parameters

Security Parameters	Description
SoMachine protocol active	This parameter allows you to deactivate the SoMachine protocol on Ethernet interfaces. When deactivated, every SoMachine 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 SoMachine, from an HMI target that wants to exchange variables with this controller, from an OPC server, or from Controller Assistant.
Modbus Server active	This parameter allows you to deactivate the Modbus Server of the Logic Controller. When deactivated, every Modbus request to the Logic Controller is ignored.
Web Server active	This parameter allows you to deactivate the Web Server of the Logic Controller. When deactivated, the HTTP requests to the Logic Controller Web Server are ignored.
FTP Server active	This parameter allows you to deactivate the FTP Server of the Logic Controller. When deactivated, FTP requests are ignored.
Discovery protocol active	This parameter allows you to deactivate Discovery protocol. When deactivated, Discovery requests are ignored.
SNMP protocol active	This parameter allows you to deactivate SNMP server of the Logic Controller. When deactivated, SNMP requests are ignored.

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 LMC058 Motion 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.

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 SoMachine, 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)

Function Code Dec (Hex)	Subfunction Dec (Hex)	Function
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 207)* 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).

Diagnostic Request

This table contains the data selection code list:

Data Selection Code (hex)	Description
00	Reserved
01	Basic Network Diagnostics
02	Ethernet Port Diagnostic
03	Modbus TCP/Port 502 Diagnostics
04	Modbus TCP/Port 502 Connection Table
05 - 7E	Reserved for other public codes
7F	Data Structure Offsets

Web Server

Introduction

The controller provides as a standard equipment an embedded Web server with a predefined factory 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:

- Microsoft Internet Explorer (version 6.0 or higher)
- Mozilla Firefox (version 1.5 or higher)

The Web server is limited to 10 TCP connections *(see page 139)*.

NOTE: The Web server can be disabled by unchecking the **Web Server active** parameter in the Ethernet Configuration tab *(see page 143)*.

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

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

Web Server Access

Access to the Web server is controlled by User Rights when they are enabled in the controller. For more information, refer to **Users and Groups** Tab Description *(see page 76)*.

If User Rights are not activated in the controller, you are prompted for a user name and password unique to the FTP/Web server. The default user name is USER and the default password is also USER.

NOTE: For compatibility reasons, the access to the FTP/Web server has been maintained from previous versions of SoMachine. That is to say, if you convert your application to the current version of SoMachine, the unique login to the FTP/Web server continues to function as it had. However, it is preferable to use the newly implemented User Rights to help protect your controller as a whole. If you implement User Rights, this unique login to the FTP/Web server is disabled in favor of the more robust method exercised by the User Rights implementation.

Access to the Web Server site requires a login on first prompt with a **User** and a **Password** (by default USER for both).

For reasons of security for your installation, you must immediately change the default password upon first login if User Rights are not enabled in the controller.

⚠ WARNING

UNAUTHORIZED DATA ACCESS

- Immediately change the default password to a new, secure password.
- Do not distribute the password to unauthorized or otherwise unqualified personnel.
- 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.

NOTE: A secure password is one that has not been shared or distributed to any unauthorized personnel and does not contain any personal or otherwise obvious information. Further, a mix of upper and lower case letters and numbers offer greater security. You should choose a password length of at least 7 characters.

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 SoMachine Programming Guide). This effectively removes the existing application from the controller memory, but restores the ability to access the controller.

If you have not enabled User Rights and if you have lost or forgotten the password, you will need to connect directly to the controller with SoMachine and perform a reset origin to reestablish the default password. After doing so, set up a new, secure password.

NOTE: For users who have a controller with a firmware version less than or equal to 2.0.2.0, the log into the Web server is anonymous and without password.

Home Page Access

To access to the website home page shown in the following illustration, type in your navigator the IP address of the controller, or 90.0.0.1 for a USB connection:

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 (the LMC058 home page is similar):

Item	Description
1	Generic menu bar *(see page 152)*
2	Active page submenu

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.

Generic Menu Bar

The generic menu bar lets you access to the main Web server pages.

The Web Server contains the following pages:

Menu	Page	Description
Home	Home *(see page 150)*	Home page of the controller Web server page. Provides access to the tabs: ● Monitoring ● Diagnostics ● Maintenance ● Setup
Documentation	References	Link to the brand site.

Home page menu descriptions:

Menu	Submenu	Description
Monitoring	Controller Viewer *(see page 155)*	• Serial Number • Version (firmware, boot...) • Configuration status
	Expansion Viewer *(see page 156)*	Shows the status of expansion modules.
	I/O Viewer *(see page 157)*	Shows the module with module I/O values.
	Oscilloscope *(see page 158)*	Displays 2 variables in the form of a recorder-type time chart.
	Data parameters *(see page 159)*	Lets you display and modify controller variables.
Diagnostics	PLC *(see page 162)*	Controller status
	Ethernet *(see page 163)*	Ethernet status
	Serial *(see page 164)*	Serial line status
Maintenance	FTP *(see page 165)*	Link to the file system server (`/Usr`, `/bd0` and `/Sys` folders)
Setup	Post configuration setup *(see page 245)*	Lets you set the Ethernet and serial line parameters.
	EthernetIP configurations files *(see page 167)*	Lets you set the EthernetIP configuration files.
	Security *(see page 167)*	Lets you modify the user login password (Default password is USER).

The following submenu is visible in each tab:

Submenu	Description
Info	Current Controller information • reference • running state • user login name
Control	Allows you to Start or Stop the controller

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

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

Page Access

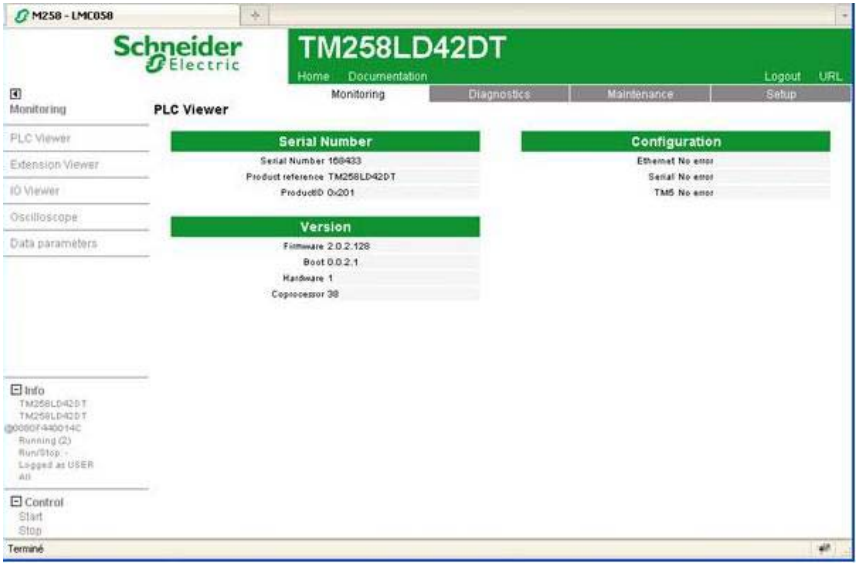
This table lists the necessary status of the controller to access different pages:

Menu	Submenu	Controller State			
		EMPTY	STOPPED	RUNNING	HALT
Home	Home	X	X	X	X
Documentation	References	X	X	X	X

Menu	Submenu	Controller State			
		EMPTY	STOPPED	RUNNING	HALT
Monitoring	PLC Viewer	X	X	X	X
	Expansion Viewer	-	X	X	-
	I/O Viewer	-	X	X	-
	Oscilloscope	-	X	X	-
	Data parameters	-	X	X	-
Diagnostics	PLC diagnostic	X	X	X	X
	Ethernet diagnostic	X	X	X	X
	Serial diagnostic	X	X	X	X
Maintenance	`/Usr` or `/bd0`	X	X	X	X
	`/Sys`	X	X	X	X
Setup	Post configuration setup	X	X	X	X
	EthernetIP configurations files	X	X	X	X

Monitoring: Controller Viewer Submenu

The Controller Viewer page shows the controller status:

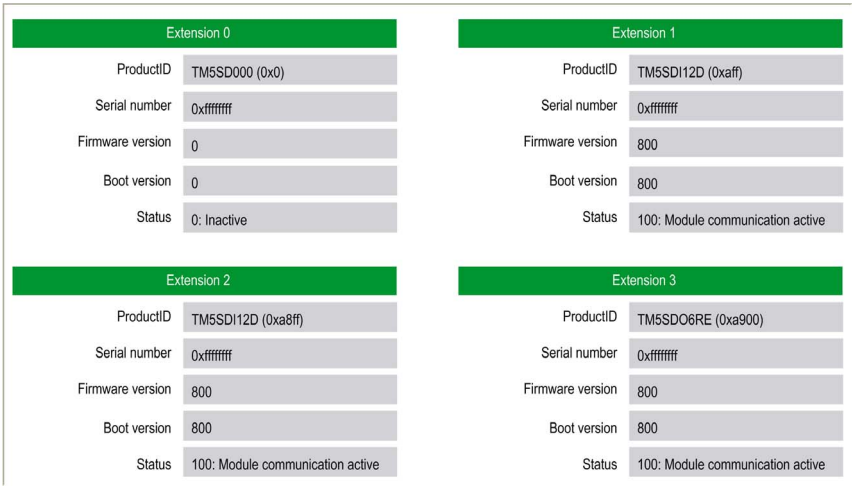


The Configuration status field can change depending on the controller reference visualized (a TM258LD42DT in the previous screenshot) and can have the following status:

Configuration Status	Description
No error	No error detected on the corresponding element.
Error	An error is detected on the corresponding element.

Monitoring: Expansion Viewer Submenu

The Expansion Viewer page shows the expansion module status:

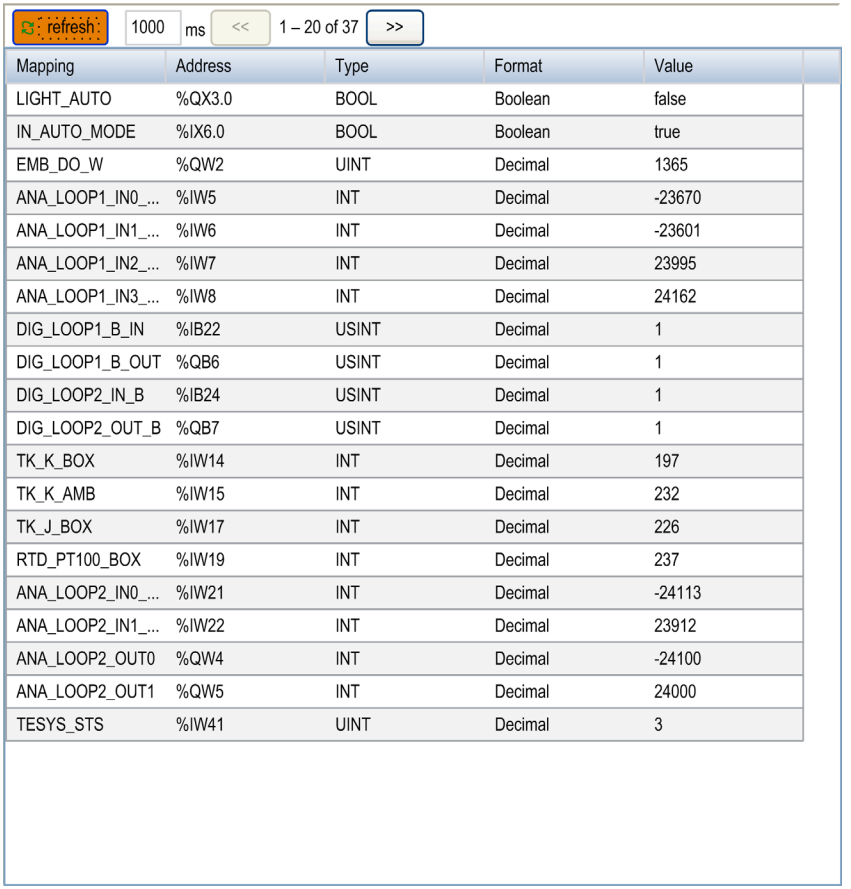


The following table describes each status code:

Status Code	Description
0	INACTIVE: module inactive
10	BOOT: boot state
11	FWDNLD: firmware download in progress
20	PREOP: basic initialization
30	OPERATE: register initialization
100	ACTIVE: module communication active
200	ERROR: an error has been detected
201	UNSUP: unsupported module
202	NOCFG: no configuration available

Monitoring: IO Viewer Submenu

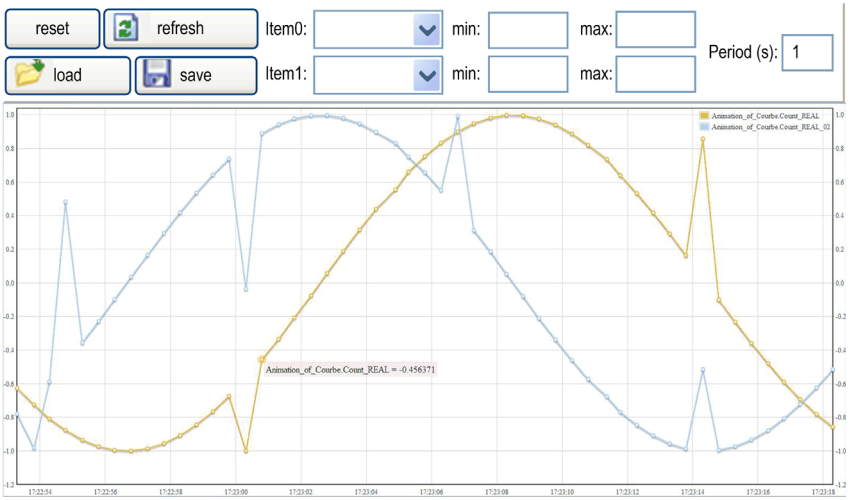
The IO Viewer allows you to display and modify the current I/O values:



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 page 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(s)	Page refresh period in seconds

Monitoring: Data Parameters

Monitoring variables in the Web Server

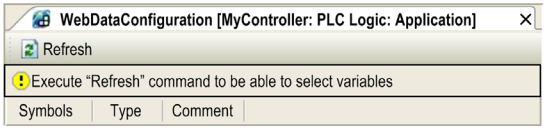
To monitor variables in the web server, you should add a **Web Data Configuration** object to your project. Within this object, you can select all variables you want to monitor.

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

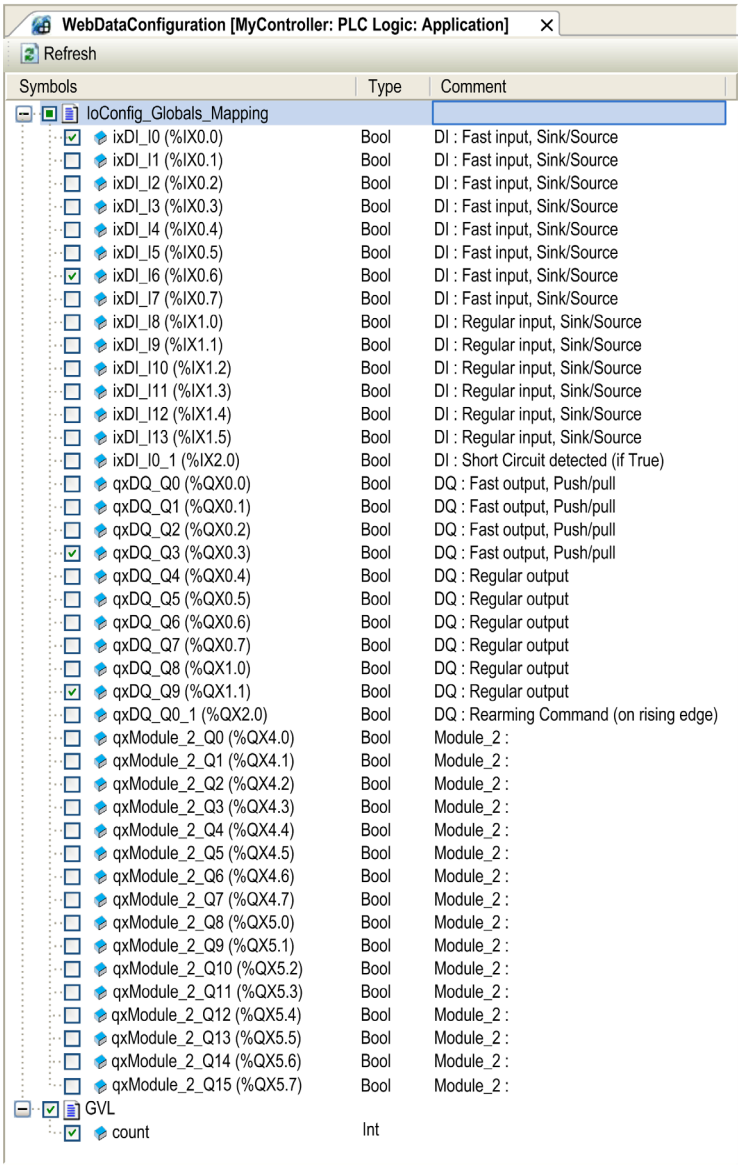
Step	Action
1	Right click the **Application** node in the **Applications tree** tab.
2	Click **Add Object → Web Data Configuration...**. **Result:** The **Add Web Data Configuration** window is displayed.
3	Click **Add**. **Result:** The **Web Data Configuration** object is created and the **Web Data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

Web Data Configuration Editor

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



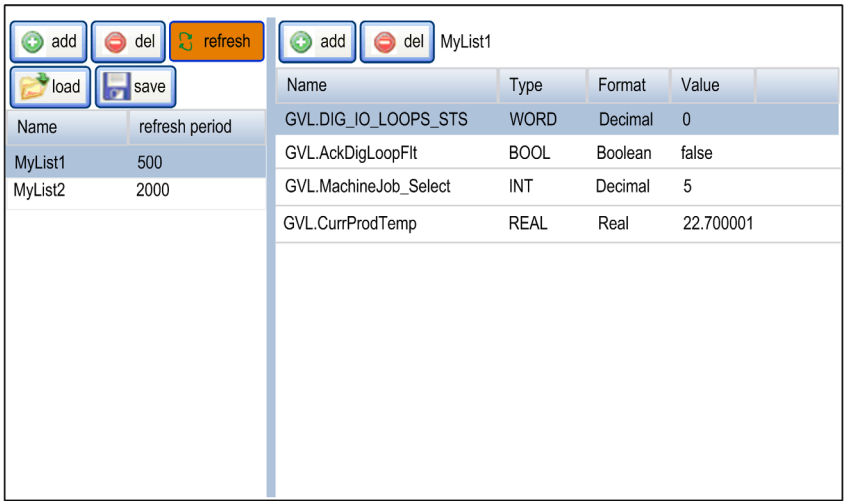
Select the variables you want to monitor in the web server:



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

Monitoring: Data Parameters Submenu

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



Element	Description
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* or */bd0/web* directory)
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

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

Diagnostic: Controller Submenu

The Controller page displays information of the current status of the controller:

Identification	
VendorID	0x101a
Vendor name	Schneider Electric
ProductID	0x218
Product reference	LMC058LF424S0
Serial Number	168433
Node name	My_Machine_Controller

Version	
Firmware	2.0.0.29
Boot	0.0.0.29
Hardware	0x1
Chip	0x12

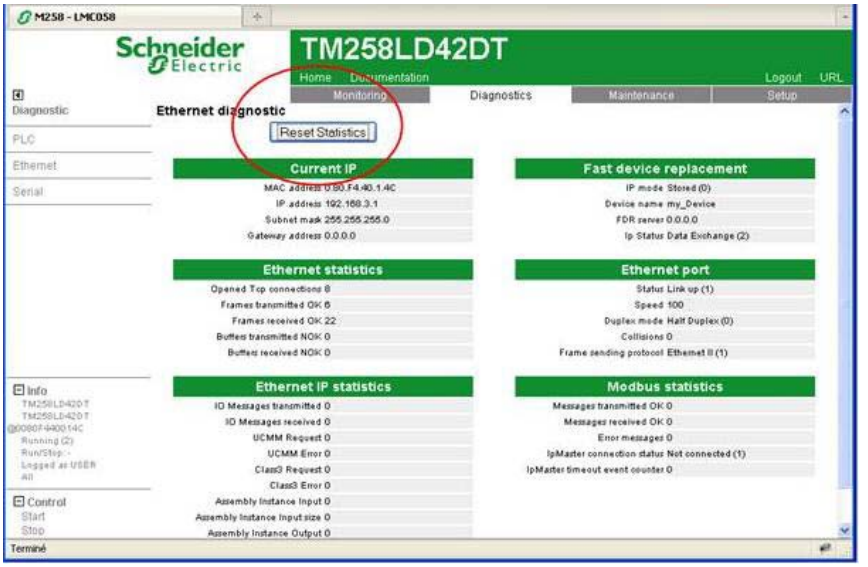
Status	
Application status	Running (2)
Boot project status	Same boot project (65535)
IO Status 1	Ok (FFFF)
IO Status 2	Ok (FFFF)
Clock Battery Status	Ok (FFFF)
Application signature 1	D2D163B1
Application signature 2	4E1C64CB
Application signature 3	8493398C
Application signature 4	B12421EC
Last stop cause	Reset (2)
Last application error	Hardware watchdog expired (17)
System Fault 1	No error
System Fault 2	No error
Last stop time	Fri, 9 Oct 2009 17:03:41
Last power-off time	Fri, 9 Oct 2009 16:03:55
Events counter	0
Host: USB Host status	Not connected (0)
Prg Port: Terminal prg port status	Connected (2)

Extension bus	
Bus status	0b0000000011110000 : Driver for this connector is available TM5 Bus hardware found TM5 Bus configuration done TM5 Bus is active and can be used
Sync error count	114
ASync error count	6
Break count	1
Topology change count	17
Cycle count	61482

File	
File system free handle	18
File system total bytes	127795200 (122 MB)
File system free bytes	126644224 (121 MB)

Diagnostic: Ethernet Submenu

The Ethernet page displays Ethernet communication information:



The **Reset Statistic** button sets the **Ethernet statistics** to 0.

Diagnostics: Serial Submenu

The Serial page displays the serial line communication information:

The **Reset Statistics** button sets the statistics of the serial connections to 0.

Diagnostics: Profibus Submenu

The Profibus page is available for controllers with PCI module. It displays the Profibus communication information:

Maintenance Tab

The Maintenance page provides access to the `/usr`, `/bd0` and `/sys` folders of the controller flash memory *(see page 30)*:

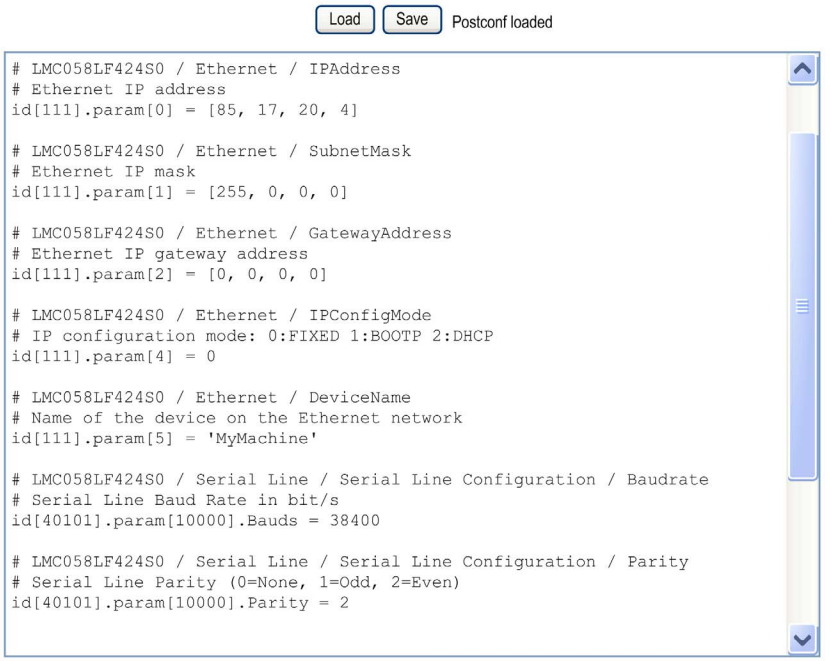
Index of `/usr` or `/bd0`:

Index of `/sys`:

NOTICE
UNINTENDED WEB SERVER AND CONTROLLER BEHAVIOR Do not modify any of the files in the `/usr` and `/sys` directories. **Failure to follow these instructions can result in equipment damage.**

Setup: Post Conf Submenu

The **Post Conf** page allows you to update the PostConf *(see page 245)* file saved on the controller:



Step	Action
1	Click **Load**.
2	Modify the parameters *(see page 249)*.
3	Click **Save**. **NOTE:** The new parameters will be considered at next Post Configuration file reading *(see page 246)*.

Setup: Ethernet IP configurations File Submenu

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

Index of `/usr` or `/bd0`:

My Machine Controller.gz

My Machine Controller.ico

My Machine Controller.eds

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

Setup: Security Submenu

The Security page allows you to modify the password to access the controller Web Server / FTP Server page.

The password is case-sensitive and can be a mix of up to 10 alphanumerical characters (a...Z, 0...9).

If you have lost or forgotten the password, you will need to connect directly to the controller with SoMachine and perform a reset origin to re-establish the default password. After doing so, set up a new, secure password.

NOTE: Do not use the Security page to modify the password if User Rights are activated. For more details about User Rights, refer to the SoMachine Programming Guide.

⚠ WARNING
UNAUTHORIZED DATA ACCESS ● Immediately change the default password to a new, secure password. ● Do not distribute the password to unauthorized or otherwise unqualified personnel. ● Disable the 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.**

NOTE: A secure password is one that has not been shared or distributed to any unauthorized personnel and does not contain any personal or otherwise obvious information. Further, a mix of upper and lower case letters, numbers and special characters offer the best security possible. You should choose a password length of at least 7 characters.

FTP Server

Introduction

Any FTP client installed on a computer that is connected to the controller (Ethernet or through USB port), without SoMachine 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 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 available even if the controller is empty (no user application and no User Rights are enabled).

FTP Access

Access to the FTP server is controlled by User Rights when they are enabled in the controller. For more information, refer to **Users and Groups** Tab Description *(see page 76)*.

If User Rights are not activated in the controller, you are prompted for a user name and password unique to the FTP/Web server. The default user name is USER and the default password is also USER.

NOTE: For compatibility reasons, the access to the FTP/Web server has been maintained from previous versions of SoMachine. That is to say, if you convert your application to the current version of SoMachine, the unique login to the FTP/Web server continues to function as it had. However, it is preferable to use the newly implemented User Rights to help protect your controller as a whole. If you implement User Rights, this unique login to the FTP/Web server is disabled in favor of the more robust method exercised by the User Rights implementation.

Access to the Web Server site requires a login on first prompt with a **User** and a **Password** (by default USER for both).

For reasons of security for your installation, you must immediately change the default password upon first login if User Rights are not enabled in the controller.

⚠ WARNING

UNAUTHORIZED DATA ACCESS

- Immediately change the default password to a new, secure password.
- Do not distribute the password to unauthorized or otherwise unqualified personnel.
- 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.

NOTE: A secure password is one that has not been shared or distributed to any unauthorized personnel and does not contain any personal or otherwise obvious information. Further, a mix of upper and lower case letters and numbers offer greater security. You should choose a password length of at least 7 characters.

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 SoMachine Programming Guide). This effectively removes the existing application from the controller memory, but restores the ability to access the controller.

If you have not enabled User Rights and if you have lost or forgotten the password, you will need to connect directly to the controller with SoMachine and perform a reset origin to reestablish the default password. After doing so, set up a new, secure password.

NOTE: For users who have a controller with a firmware version less than or equal to 2.0.2.0, the log into the FTP server is anonymous and without password.

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 server objects:

Object	Description	Access	Default Value
sysDescr	Text description of the device	Read	SCHNEIDER LMC058 Fast Ethernet TCP/IP
sysObjectID	Points to the product reference in the private MIB	Read	1.3.6.1.4.1.3833.1.7.37
sysUpTime	Time elapsed since the controller was last turned on	Read	-
sysContact	Data item used to contact the manager of this node	Read/Write	-
sysName	Node administrative name	Read/Write	LMC058LF42
sysLocation	Physical location of the product	Read/Write	-
sysService	Indicates the type of service provided by this product	Read	79

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. For more details, refer to www.schneider-electric.com.

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

Section 11.2
Firewall Configuration

Introduction

This section describes how to configure the firewall of the Modicon LMC058 Motion Controller.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction	175
Dynamic Changes Procedure	177
Firewall Behavior	178
Firewall Script Commands	180

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 important part of an overall, system level strategy.

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 3 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 2 ways to load these dynamic changes:

- Using a physical USB memory key,
- Using a function block *(see page 177)* in the application.

Application Settings

Refer to Ethernet Configuration *(see page 143)*.

Dynamic Changes Procedure

Using a USB Memory Key

This table describes the procedure to execute a script file from a USB memory key:

Step	Action
1	Create a valid script file *(see page 180)*. For instance, name the script file *FirewallMaintenance.cmd*.
2	Load the script file on the USB memory key. For instance, 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 instance, the code line is
	`Firewall_install "/bd0/Usr/cfg/FirewallMaintenance.cmd"`

4	Insert the USB memory key 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 180)*. For instance, name the script file *FirewallMaintenance.cmd*.
2	Load the script file in the controller memory. For instance, load the script file in the *Usr/Syslog* folder with FTP.
3	Use an ExecuteScript *(see Modicon LMC058 Motion Controller, System Functions and Variables, LMC058 PLCSystem Library Guide)* function block. For instance, 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 5 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 been already 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 a USB memory key containing a cybersecurity script is plugged into the controller, booting is blocked. First remove the USB key to correctly boot the controller.	

Firewall Script Commands

Overview

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

Script File Syntax

The syntax of script files is described in Script Syntax Guidelines.

General Firewall Commands

The following commands are available to manage the Ethernet firewall of the LMC058 Motion 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`	IP addresses are allowed access to the controller on the Ethernet interfaces.
`FireWall Eth1 Default Allow`	Frames are accepted by the controller.
`FireWall Eth1 Default Reject`	Frames are rejected by the controller. **NOTE:** By default, if this line is not present, it corresponds to the command `FireWall Eth1 Default Reject`.

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 182)*)	Frames with the specified destination port number are allowed.

Command	Range	Description
`Firewall Eth1 Reject port_type port Y`	Y = (destination port numbers *(see page 182)*)	Frames with the specified destination port number are allowed.
`Firewall Eth1 Allow port_type ports Y1 to Y2`	Y = (destination port numbers *(see page 182)*)	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 182)*)	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 182)*)	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 182)*)	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 182)*)	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 182)*)	Frames from the specified IP address and with a destination port number in the specified range are rejected.
`Firewall Eth1 Allow IPs •1.•1.•1.•1 to •2.•2.•2.•2 on port_type port Y`	• = 0...255 Y = (destination port numbers *(see page 182)*)	Frames from an IP address in the specified range and with the specified destination port number are rejected.
`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 182)*)	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 182)*)	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 182)*)	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.
`Firewall Eth1 Reject MAC ••:••:••:••:••:••`	• = 0...F	Frames with the specified MAC address ••:••:••:••:•• are rejected.

Script Example

```
; Enable firewall on Ethernet 1. All frames are rejected;
FireWall Enable;
; Block all Modbus Requests on all IP address
Firewall Eth1 Reject tcp port 502;
; Allow FTP active connection for IP address 85.16.0.17
Firewall Eth1 Allow IP 85.16.0.17 on tcp port 20 to 21;
```

Used Ports

Protocol	Destination Port Numbers
SoMachine	UDP 1740, 1741, 1742, 1743 TCP 1105
FTP	TCP 21, 20
HTTP	TCP 80
Modbus	TCP 502
Discovery	UDP 27126, 27127
SNMP	UDP 161, 162
NVL	UDP Default value: 1202
EtherNet/IP	UDP 2222 TCP 44818

Section 11.3
Ethernet Optional Devices

What Is in This Section?

This section contains the following topics:

Topic	Page
Ethernet Manager	184
EtherNet/IP Device	185
Modbus TCP Slave Device	207

Ethernet Manager

Adding an Ethernet Manager

The controller supports the following Ethernet managers:

- EthernetIP (for CIP Device)
- ModbusTCP Slave Device

To add an Ethernet manager to your controller, select in the **Hardware Catalog**:

- For an EthernetIP: **EthernetIP**
- For a ModbusTCP: **ModbusTCP Slave Device**

Drag it to the **Devices tree** and drop it on one of the highlighted nodes.

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

• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

EtherNet/IP Device

Introduction

This section describes the configuration of the EtherNet/IP Device (CIP) to the controller.

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

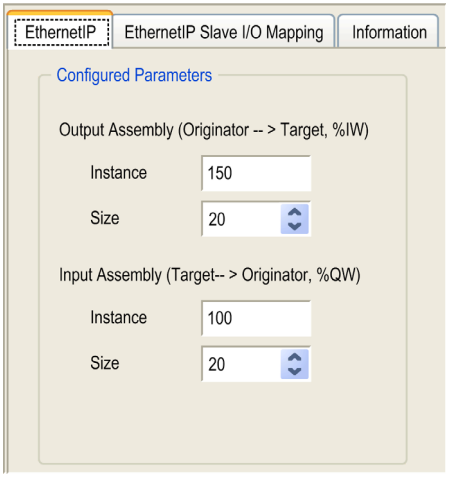
Adding an EtherNet/IP Device

See Adding an EtherNet Manager *(see page 184)*.

EtherNet/IP Device Configuration

To configure the EtherNet/IP device parameters, double-click **Ethernet → EthernetIP** in the **Devices tree**.

The following 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	SoMachine Default Value
Output Assembly	**Instance**	150...189	150
	Size	2...40	20
Input Assembly	**Instance**	100...149	100
	Size	2...40	20

EDS File Generation

You can generate an EDS file to facilitate configuring 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**.

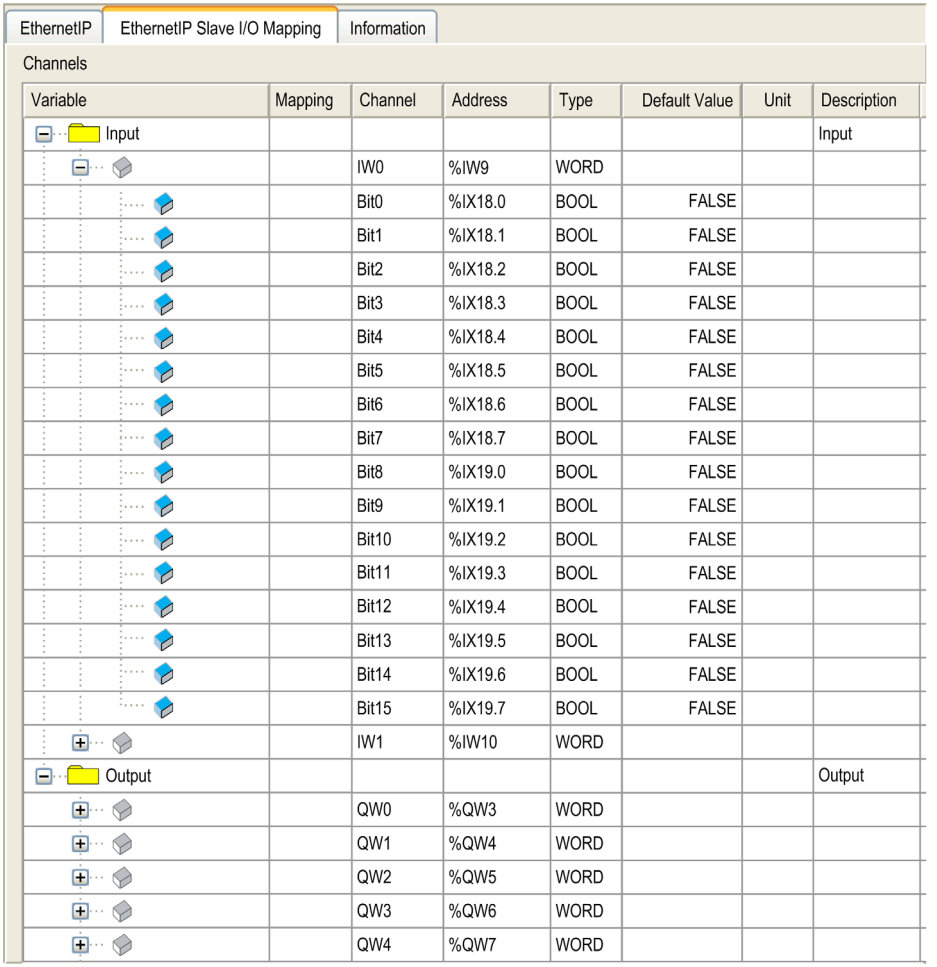
NOTE: The **Major Revision** and **Minor Revision** objects in the EDS file are used to ensure uniqueness of the EDS file. The values of these objects do not reflect the actual controller revision level.

The EDS file is generated automatically in the "/usr/Eip" directory within the controller when an application is downloaded, or at start-up when a boot application exists, according to the parameters above.

NOTE: The EDS file is generated when the control network is working correctly on the controller (cable connected and the IP address is acquired).

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.



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			

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 113)*.

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 Device Configuration *(see page 185)*.

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, open a connection (global name used by EtherNet/IP protocol level) 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

Profile

The controller supports the following objects:

Object class	Class ID	Cat.	Number of Instances	Effect on Interface Behavior
Identity Object *(see page 189)*	01 hex	1	1	Supports the reset service
Message Router Object *(see page 192)*	02 hex	1	1	Explicit message connection
Assembly Object *(see page 195)*	04 hex	2	2	Defines I/O data format

Object class	Class ID	Cat.	Number of Instances	Effect on Interface Behavior
Connection Manager Object *(see page 197)*	06 hex		1	-
File Object *(see page 199)*	37 hex		2	Allows to exchange EDS file
Modbus Object *(see page 202)*	44 hex		1	-
TCP/IP Interface Object *(see page 203)*	F5 hex	1	1	TCP/IP configuration
Ethernet Link Object *(see page 205)*	F6 hex	1	1	Counter and status information

Identity Object (Class ID = 01 hex)

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

Attribute ID	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	01h	Implementation revision of the Identity Object
2	Get	Max Instances	UINT	01h	The largest instance number
3	Get	Number of Instances	UINT	01h	The number of object instances
4	Get	Optional Instance Attribute List	UINT, UINT [ ]	00h	The first 2 bytes contain the number of optional instance attributes. Each following pair of bytes represents the number of other optional instance attributes.
6	Get	Max Class Attribute	UINT	07h	The largest class attributes value
7	Get	Max Instance Attribute	UINT	07h	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 value is the default value if this parameter is omitted.
1	Reset Warm.
2	Not supported.
3...99	Reserved
100...199	Vendor specific
200...255	Reserved

The following table describes the Instance attributes:

Attribute ID	Access	Name	Data Type	Value	Details
1	Get	Vendor ID	UINT	243h	Schneider Automation ID
2	Get	Device type	UINT	0Eh	Controller
3	Get	Product code	UINT	806h	Controller product code
4	Get	Revision	Struct of USINT, USINT	-	Product revision of the controller (1) Equivalent to the 2 low bytes of controller version
5	Get	Status	WORD (1)	-	See definition in the table below

Attribute ID	Access	Name	Data Type	Value	Details
6	Get	Serial number	UDINT	-	Serial number of the controller XX + 3 LSB of MAC address
7	Get	Product name	Struct of USINT, STRING	-	Example: LMC258LF42DT.

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

Example: 0205h means revision V5.2.

Status Description (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 error detected • 3: no I/O connections established • 4: non-volatile configuration invalid • 5: non recoverable error detected • 6: at least one I/O connection in RUNNING state • 7: at least one I/O connection established, all in idle mode • 8: reserved • 9...15: unused
8	Minor Recoverable Error	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 Error	TRUE indicates the device detected an error, which, under most circumstances, is not recoverable. This type of event does not lead to a change in the device state.
10	Major Recoverable Error	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.

Bit	Name	Description
11	Major Unrecoverable Error	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	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	01h	Implementation revision of the Message Router Object
2	Get	Max Instances	UINT	01h	The largest instance number
3	Get	Number of Instance	UINT	01h	The number of object instances
4	Get	Optional Instance Attribute List	Struct of UINT, UINT [ ]	20	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	00h	The number and list of any implemented optional services attribute (0: no optional services implemented)
6	Get	Max Class Attribute	UINT	07h	The largest class attributes value
7	Get	Max Instance Attribute	UINT	119	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	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 two bytes that follow represent 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	20h	Maximum number of concurrent CIP (Class1 or Class 3) connections supported
100	Get	Total incoming Class1 packets received during the last second	UINT	-	Total number of incoming packets received for all implicit (Class1) connections during the last second
101	Get	Total outgoing Class1 packets sent during the last second	UINT	-	Total number of outgoing packets sent for all implicit (Class1) connections during the last second
102	Get	Total incoming Class3 packets received during the last second	UINT	-	Total number of incoming packets received for all explicit (Class 3) connections during the last second
103	Get	Total outgoing Class3 packets sent during the last second	UDINT	-	Total number of outgoing packets sent for all explicit (Class 3) connections during the last second
104	Get	Total incoming unconnected packets received during the last second	UINT	-	Total number of incoming unconnected packets received during the last second

Attribute ID	Access	Name	Data Type	Value	Description
105	Get	Total outgoing unconnected packets sent during the last second	UINT	-	Total number of outgoing unconnected packets sent during the last second
106	Get	Total incoming EtherNet/IP packets received during the last second	UINT	-	Total unconnected Class1 or Class 3 packets received during the last second
107	Get	Total outgoing EtherNet/IP packets sent during the last second	UINT	-	Total unconnected Class1 or Class 3 packets sent during the last second
108	Get	Total incoming Class1 packets received	UINT	-	Total number of incoming packets received for all implicit (Class1) connections
109	Get	Total outgoing Class1 packets sent	UINT	-	Total number of outgoing packets sent for all implicit (Class1) connections
110	Get	Total incoming Class3 packets received	UINT	-	Total number of incoming packets received for all explicit (Class 3) connections. This number includes the packets that would be returned if an error had been detected (listed in the next two rows).
111	Get	Total incoming Class3 packets Invalid Parameter Value	UINT	-	Total number of incoming Class 3 packets that targeted a non-supported service/class/instance/attribute/member
112	Get	Total incoming Class3 packets Invalid Format	UINT	-	Total number of incoming Class 3 packets that had an invalid format
113	Get	Total outgoing Class3 packets sent	UINT	-	Total number of packets sent for all explicit (Class 3) connections
114	Get	Total incoming unconnected packets received	UINT	-	Total number of incoming unconnected packets. This number includes the packets that would be returned if an error had been detected (listed in the next two rows).

Attribute ID	Access	Name	Data Type	Value	Description
115	Get	Total incoming unconnected packets Invalid Parameter Value	UINT	-	Total number of incoming unconnected packets that targeted a non-supported service/class/instance/attribute/member
116	Get	Total incoming unconnected packets Invalid Format	UINT	-	Total number of incoming unconnected packets that had an invalid format
117	Get	Total outgoing unconnected packets sent	UINT	-	Total number of all unconnected packets sent
118	Get	Total incoming EtherNet/IP packets	UINT	-	Total unconnected, Class 1, or Class 3 packets received
119	Get	Total outgoing EtherNet/IP packets	UINT	-	Total unconnected, Class 1, or Class 3 packets sent

Assembly Object (Class ID = 04 hex)

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

Attribute ID	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	2	Implementation revision of the Assembly Object
2	Get	Max Instances	UINT	189	The largest instance number
3	Get	Number of Instances	UINT	2	The number of object instances
4	Get	Optional Instance Attribute List	Struct of: UINT UINT [ ]	 1 4	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	00h	The number and list of any implemented optional services attribute (0: no optional services implemented)
6	Get	Max Class Attribute	UINT	07h	The largest class attributes value
7	Get	Max Instance Attribute	UINT	04h	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
18	Get Member	Reads a member of an Assembly object instance
19	Set Member	Modifies a member of an Assembly object instance

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	Access	Name	Data Type	Value	Description
1	Get	Number of Member Object List	UINT	2...40	Always 1 member for the controller
2	Get	Member List	ARRAY of Struct	-	Array of 1 structure where each structure represents one member
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

Member list content:

Name	Data Type	Value	Type of Reset
Member data size	UINT	4...40	Member data size in bits
Member path size	UINT	6	Size of the EPATH (see table below)
Member path	EPATH	-	EPATH to the Member

EPATH is:

Word	Value	Semantic
0	2004 hex	Class 4
1	24xx hex	Instance xx where xx is the instance value (example: 2464 hex = instance 100).
2	30 hex	Attribute 3

Connection Manager Object (Class ID = 06 hex)

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

Attribute ID	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	2	Implementation revision of the Connection Manager Object
2	Get	Max Instances	UINT	189	The largest instance number
3	Get	Number of Instances	UINT	2	The number of object instances

Attribute ID	Access	Name	Data Type	Value	Details
4	Get	Optional Instance Attribute List	Struct of: UINT UINT [ ]	-	The number and list of the optional attributes. The first word contains the number of attributes to follow and each following word contains another attribute code. Following optional attributes include: ● total number of incoming connection open requests ● the number of requests rejected because of the non-conforming format of the Forward Open ● the number of requests rejected because of insufficient resources ● the number of requests rejected because of the parameter value sent with the Forward Open ● the number of Forward Close requests received ● the number of Forward Close requests that had 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	07h	The largest class attributes value
7	Get	Max Instance Attribute	UINT	08h	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

Service Code (hex)	Name	Description
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	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

File Object (Class ID = 37 hex)

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

Attribute ID	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	1	Implementation revision of the File Object
2	Get	Max Instances	UINT	C9h	The largest instance number
3	Get	Number of Instances	UINT	2	The number of object instances

Attribute ID	Access	Name	Data Type	Value	Details
6	Get	Max Class Attribute	UINT	20h	The largest class attributes value
7	Get	Max Instance Attribute	UINT	0Bh	The largest instance attributes value
32	Get	Instance List	-	-	Returns information on all configured instances including Instance Number, Instance Name and Instance File Name

The following table describes the Class Services:

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

Instance Code

The File object provides download functionality for the EDS and EDS Icon files. The following instances of the File object are implemented:

- Instance C8 hex returns an uncompressed version of the EDS text file. The instance name attribute is returned as "EDS and Icon Files". The file name attribute returns "LMC058xxx.eds" where LMC058xxx is the exact reference of the controller. The contents of the EDS file are adjusted dynamically by the controller. The connection data sizes in the EDS file are adjusted to reflect the actual standard Assembly instance sizes.
- Instance C9 hex returns a compressed version of the device EDS icon file. The instance name is returned as "Related EDS and Icon Files". The file name attribute returns "LMC058xxx.gz" where M258xxx is the exact reference of the controller. This is zip encoded file containing only one file: LMC058x.ico. The file is encoded with the ZLIB compression file format. ZLIB is a free, general purpose, legally unencumbered, loss-less compression library. The specifications are available from the Internet Engineering Task Force (http://www.ietf.org).

The following table describes the Instance Services:

Service Code (hex)	Name	Description
0E	Get Attribute Single	Returns the value of the specified instance attribute
4B	Initiate Upload	Start uploading process. Request contains the Maximum File Size the Client can accept on Upload. Response contains the actual File Size, which will never be more than the Maximum File Size and the Transfer Size, which is the number of bytes transferred with each Upload Transfer request.

Service Code (hex)	Name	Description
4F	Upload Transfer	Upload another section of file data. Request contains the Transfer Number, which is incremented with each subsequent transfer. Response contains the matching Transfer Number, Transfer Type, File Data, and for the last transfer, the Checksum word. Transfer Type indicates if this is the first, intermediate or last packet, if it is the only packet, or if the transfer should be aborted.

The following table describes the Instance Attributes:

Attribute ID	Access	Name	Data Type	Value	Description
1	Get	State	USINT	-	One of the following: ● 0: non existent ● 1: file empty - file should not have any content until it is downloaded from the remote client. When set, File name, Revision, Checksum and File Save Flag values have no meaning, and File Size is 0. ● 2: file loaded - file contents are pre-loaded by the application (file size > 0) or file data has been downloaded and stored into the non-volatile memory area ● 3: upload initiated ● 4: download initiated ● 5: upload in progress ● 6: download in progress ● 7: storing to non-volatile area is in progress
2	Get	Instance Name	STRING	-	Unique Name assigned to the File Object Instance. For the C8 hex instance, it is "EDS and Icon Files". For the C9 hex instance, it is "Related EDS and Icon Files".
3	Get	Instance Format Revision	UINT	-	Revision number assigned for this instance by the application, to differentiate between different file formats.
4	Get	File Name	STRING	-	Unique name for File Storage
5	Get	File Revision	USINT	Major Minor	File Revision is updated every time file content is changed.
6	Get	File Size	UDINT	-	File Size in bytes

Attribute ID	Access	Name	Data Type	Value	Description
7	Get	File Checksum	UINT	-	Two's complement of the 16 bit sum of all bytes
8	Get	Invocation Method	USINT	-	Defines what should happen after the file is downloaded. Possible options include: ● 0: No Action ● 2: Power Cycle, etc.
9	Get	File Save Parameters	BYTE	-	If bit 1 is set, the file should be explicitly saved to non-volatile storage after download is complete.
10	Get	File Type	USINT	-	● 0: Read/Write access ● 1: Read Only access
11	Get	File Encoding Format	UINT	-	● 0: no encoding ● 1: encoded using ZLIB

Modbus Object (Class ID = 44 hex)

The Modbus object provides an additional method to access the Modbus table data. A single explicit request will either read or write 1 or more contiguous registers. An additional Pass-through service allows the user to specify an actual Modbus message data.

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

Attribute ID	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	1	Implementation revision of the Modbus Object

The following table describes the Class Services:

Service Code (hex)	Name	Description
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
4B	Read Digital Inputs	Returns the value of one or several contiguous Digital Input registers
4C	Read Coils	Returns the value of one or several contiguous Coils
4E	Read Holding Registers	Returns the value of one or several contiguous Holding Registers

Service Code (hex)	Name	Description
4F	Write Coils	Updates the value of one or several contiguous Coils
50	Write Holding Registers	Updates the value of one or several contiguous Holding Registers

NOTE: The Read Register service requires 4 bytes of data: the first word contains the starting register address and the second word contains the number of registers to read. The Write service request requires the same 4 bytes, followed by the actual data.
The Modbus Pass-through service indicates a specific Modbus function. The translation function will not perform any Indian conversion on the request or response data. Both request and response contain 1 byte of the Modbus Function code followed by the Modbus message data, including a sub-function code if present.

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	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	1	Implementation revision of the TCP/IP Interface Object
2	Get	Max Instances	UINT	1	The largest instance number
3	Get	Number of Instance	UINT	1	The number of object instances
6	Get	Max Class Attribute	UINT	07h	The largest class attributes value
7	Get	Max Instance Attribute	UINT	06h	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

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	Access	Name	Data Type	Value	Description
1	Get	Status	DWORD	Bit level	• 0: The interface configuration attribute has not been configured. • 1: The interface configuration contains a valid configuration. • 2...15: Reserved.
2	Get	Configuration Capability	DWORD	Bit level	• 0: BOOTP Client • 1: DNS Client • 2: DHCP Client • 3: DHCP-DNS capable • 4: interface configuration set table 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.

Attribute ID	Access	Name	Data Type	Value	Description
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. Otherwise, the name server address shall be set to a valid Class A, B, or C address.
			STRING	Default Domain Name	ASCII characters. Maximum length is 48 characters. Padded to an even number of characters (pad not included in length). 0: no Domain Name is configured
6	Get	Host Name	STRING	-	ASCII characters. Maximum length is 64 characters. Shall be padded to an even number of characters (pad not included in length). 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	Access	Name	Data Type	Value	Details
1	Get	Revision	UINT	2	Implementation revision of the Ethernet Link Object
2	Get	Max Instances	UINT	1	The largest instance number
3	Get	Number of Instances	UINT	1	The number of object instances
6	Get	Max Class Attribute	UINT	07h	The largest class attributes value
7	Get	Max Instance Attribute	UINT	03h	The largest instance attribute 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

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
10	Set Attribute Single	Modifies the value of the specified attribute
0E	Get Attribute Single	Returns the value of the specified instance attribute

The following table describes the Instance Attributes:

Attribute ID	Access	Name	Data Type	Value	Description
1	Get	Interface Speed	UDINT	-	Speed in Mbps (10 or 100)
2	Get	Interface Flags	DWORD	Bit level	• 0: link status • 1: half/full duplex • 2...4: negotiation status • 5: manual setting / requires reset • 6: local hardware error detected All other bits are reserved and set to 0.
3	Get	Physical Address	ARRAY of 6 USINT	-	This array contains the MAC address of the product. Format: XX-XX-XX-XX-XX-XX

Modbus TCP Slave Device

Overview

This section describes the configuration of **Modbus TCP Slave Device** to the controller.

The **Modbus TCP Slave Device** is a privileged Modbus device on the network. It serves as a gateway for an external I/O scanner configured as the Modbus master, and allows this scanner to exchange data with the controller without interfering with the operation of the Modbus server on the controller. Essentially, the Modbus TCP Slave allows two Modbus masters to coexist and exchange data.

While the **Modbus TCP Slave Device** uses standard Modbus commands (3 h, 6 h, and so on), these commands do not have their standard meaning. As this device is acting as a gateway for an external I/O scanner (Modbus master), the schema where %IW registers are associated with inputs (read-only) and %QW registers are associated with outputs (read-write) is reversed when considered from the perspective of the external Modbus master.

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

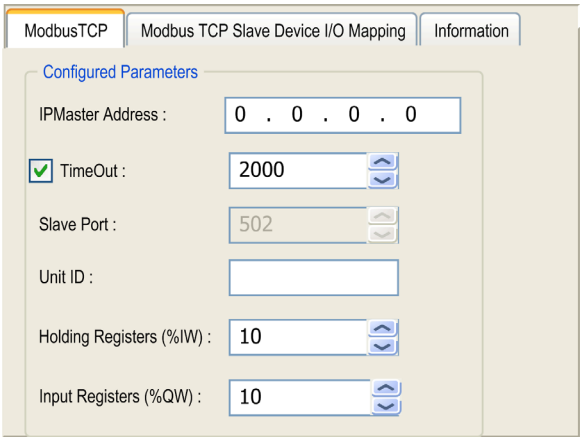
Adding a Modbus TCP Slave Device

See Adding an Ethernet Manager *(see page 184)*.

Modbus TCP Configuration

To configure the **Modbus TCP Slave Device**, double-click **Modbus → Modbus TCP** 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.
TimeOut	Timeout in 500 ms increments **NOTE:** The timeout applies to the IP master Address unless the address is 0.0.0.0.
Slave Port	Modbus communication port (502)
Unit ID	Sends the requests to the Modbus TCP slave device (1...247), instead of to the embedded Modbus server (255).
Holding Registers (%IW)	Size of the %IW registers in bytes (2...40 bytes)
Input Registers (%QW)	Size of the %QW registers in bytes (2...40 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).

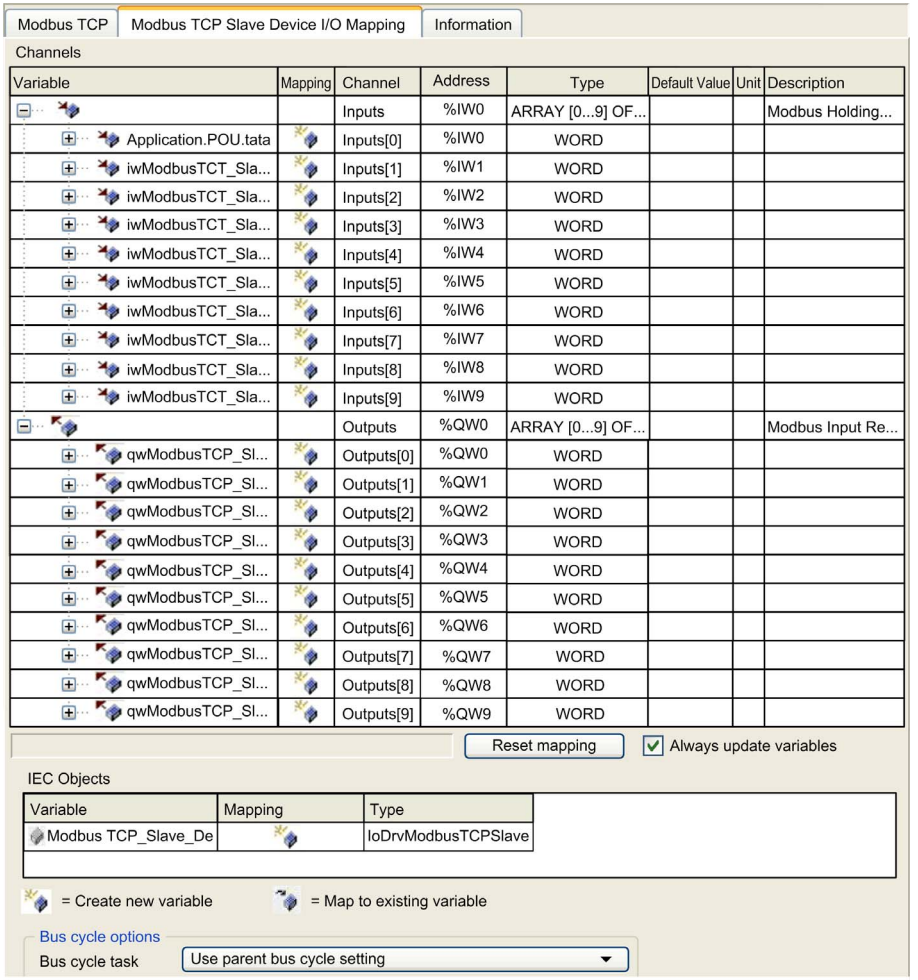
Once a **Modbus TCP Slave Device** has been configured, Modbus commands sent to its Unit ID (Modbus address) are handled differently than the same command would be when addressed to any other Modbus device on the network. For example, when the Modbus command 3 (3 hex) is sent to a standard Modbus device, it reads and returns the value of one or more registers. When this same command is sent to the Modbus TCP *(see page 146)* Slave, it facilitates a read operation by the external I/O scanner.

The **Modbus TCP Slave Device** responds to a subset of the Modbus commands, but does so in a way that differs from Modbus standards, and 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**.

Bus Cycle Options

Select the **Bus cycle task** to use:

- **Use parent bus cycle setting** (the default),
- **MAST**
- **An existing task of the project**

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.

Chapter 12
CANopen Configuration

Introduction

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

The LMC058 has 2 CAN connections allowing you to declare either a CANmotion master and a CANopen master or 2 CANopen masters:

- One connection (CAN1) supports a CANopen manager or a CANmotion manager.
- The other (CAN0) only supports a CANopen manager, which does not support motion devices.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
CANmotion Principle	212
CANmotion Interface Configuration	217
CANopen Interface Configuration	222

CANmotion Principle

Introduction

The Modicon LMC058 Motion Controller is able to synchronize up to 8 of the following devices on CANmotion:

- Lexium 05 Servo drive
- Lexium 23 Servo drive
- Lexium 32 Servo drive
- Lexium SD3 Stepper drive

Do not exceed 8 motion control devices on the CANmotion bus or install CAN devices not related to motion control. If you do, the CANmotion bus communications may be interrupted, resulting in a loss of synchronization or communication with the controlled devices and unintended operation.

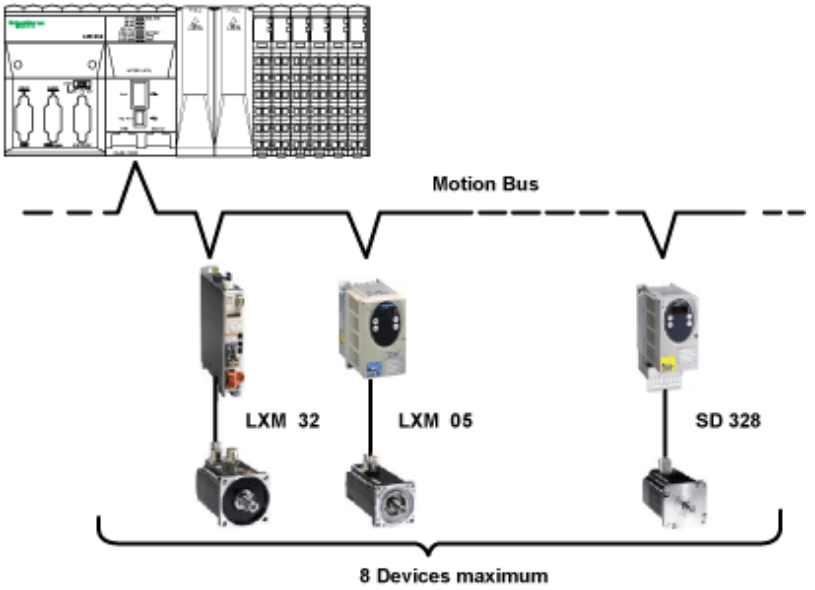
⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only connect motion devices such as drive controllers to the CANmotion bus.
- Do not connect more than 8 motion devices to the CANmotion bus.

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

Architecture Example with CANmotion



NOTE: In many cases, a daisy-chain network topology that does not use any tap-offs will provide improved performance. If you use a different topology and tap-offs to connect the motion devices on your CANmotion bus, these alternate topologies can exceed the capabilities of the CANmotion bus to provide synchronized motion control. It is essential to conduct thorough testing and commissioning before placing your CANmotion bus into service.

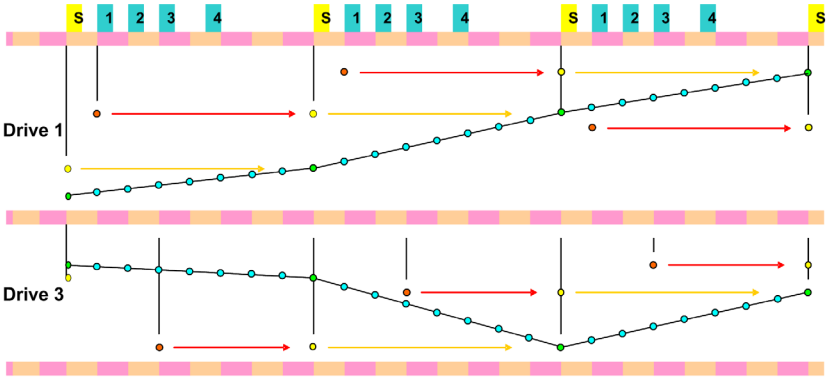
Cyclic Synchronous Modes

3 cyclic synchronous modes are available with CANmotion:

- Cyclic Synchronous Position mode (CSP) – by default
- Cyclic Synchronous Torque mode (CST)
- Cyclic Synchronous Velocity mode (CSV)

During every motion task cycle (**Sync cycle period (µs)**), a new Setpoint (position, torque or velocity) is calculated and sent to each drive by Receive Process Data Object (RPDO). The synchronization mechanism causes all drives to update their Setpoint at the same time. The new drive actual value is sent to the controller by Transmit Process Data Object (TPDO).The maximal jitter of the synchronization mechanism is 75 µs.

CANmotion Bus Principle for Axis Synchronization



- New setpoint (calculated by the controller) is sent to the drive (TPDO)
- All drives take into account the new setpoint at reception of the Synchronous signal
- New drive actual value is sent to the controller (RPDO)
- Intermediate setpoints are calculated inside the drive every 250 µs (Linear interpolation)

S Synchronous signal

1 Data exchange with drive 1

The 3 cyclic synchronous modes can be switched using the function block **SMC_SetControllerMode**. Before using this function block to set the cyclic synchronous mode of a drive controller on the CANmotion bus, confirm that the target device supports the mode. Do not command a drive controller to use an unsupported mode or unintended operation may result.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use the CST and CSV modes or the optional TPDO with Lexium 05 and Lexium SD3 drive controllers.

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

For more information, refer to Compatibility *(see page 216)* and theSoMachine Compatibility and Migration User Guide.

Cyclic Synchronous Torque and Cyclic Synchronous Velocity modes requires the configuration of 1 extra RPDO and 1 extra TPDO.

For the configuration, see Adding a CANmotion Device.

Asynchronous messaging

Asynchronous messaging is possible via SDO between the Modicon LMC058 Motion Controller and the drives on CANmotion. Only one SDO exchange per cycle is possible for all CANmotion slaves.

Optional TPDO

In addition to the TPDO and RPDO used to provide the cyclic synchronous modes, it is possible to map 1 additional TPDO per Lexium 32 CANmotion slave. This allows you to perform a cyclic update on an additional drive parameters. This option must not be used with the Lexium 05 or Lexium SD3 drive controllers.

For more information on configuring your motion controller, see Adding a CANmotion Device.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use the CST and CSV modes or the optional TPDO with Lexium 05 and Lexium SD3 drive controllers.

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

Compatibility

The following compatibility table indicates the minimum version required for proper functioning on CANmotion:

Lexium 32A	V01.03.22 or greater
Lexium 32M	V01.01.31 or greater
Lexium 05	V1.502 or greater
Lexium SD3	V1.502 or greater
Lexium 23	V1.010 Sub4

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Ensure that the firmware version of your drives controller conforms to the requirements of the table above.
- Do not use a drive controller with a lesser firmware version.

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

NOTE: The Cyclic Synchronous Torque (CST), Cyclic Synchronous Velocity (CSV) modes and the optional TPDO must not be used with Lexium SD3 and Lexium 05.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not attempt to use CST and CSV modes and the optional TPDO in the specified drives controller.

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

CANmotion Interface Configuration

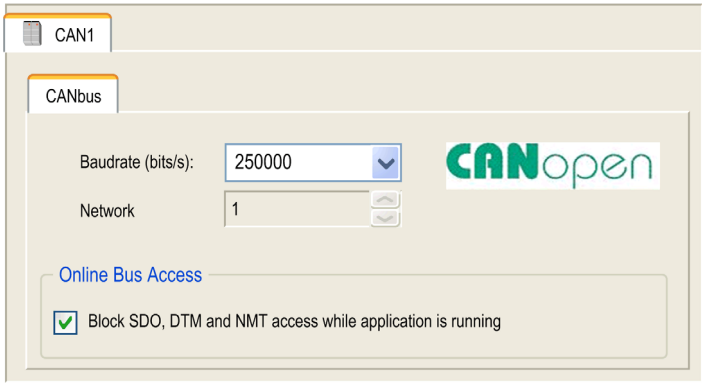
Introduction

On the CAN1 connector, you can connect a CANmotion manager:

- It accepts only the following motion devices (with a specific interface for CANmotion bus):
  - Lexium 05 Servo drive
  - Lexium 23 Servo drive
  - Lexium 32 Servo drive
  - Lexium SD3 Stepper drive
- Generic CAN devices cannot be added.

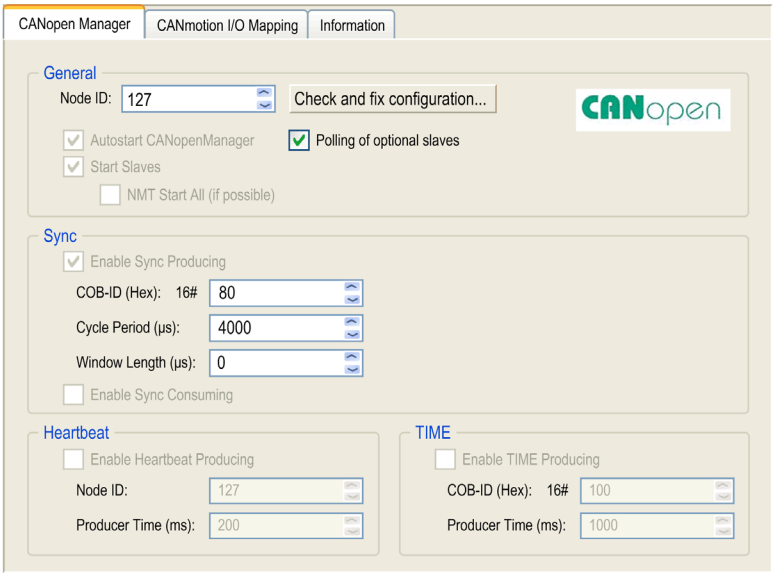
CANmotion Configuration

To configure the communication port of your controller, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **CAN1**. **Result:** The configuration dialog box for CAN communications will be displayed.
2	Configure the baud rate (by default: 250.000 bit/s):  **NOTE:** On CANmotion, only the following baud rates are supported: 250, 500 and 1000 kBaud

CANmotion Manager

To add the CANmotion Manager, proceed as follows:

Step	Action
1	Select the **CANmotion** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method *(see SoMachine, Programming Guide)* • Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*
2	Double-click the CANmotion Manager you added. **Result:** The CANmotion Manager configuration window appears: 

CANmotion Cycle Time Configuration

The CANmotion cycle time is configured with parameter **Cycle period (µs)**. Configure it from 1 to 20 ms in 1 ms steps.

Configure the **Cycle period (µs)** value to allow sufficient time for both of the following to be true:

- The program processing defined in your Motion task must have enough time to execute in full. Test the execution time of your Motion task under all operating conditions to determine this value.
- The **Cycle period (µs)** must be of sufficient length to allow the physical exchange of all PDO and SDO data between the controller and all of the configured devices.

NOTE: If you do not configure a sufficient **Cycle period (µs)**, this can result in a task or system watchdog exception or even a detected synchronization error for the controlled devices.

Calculating the Minimum Physical Data Exchange Period

This minimum time necessary to physically exchange PDO and SDO objects depends on:

- The baud rate (transmission speed)
- The number of axes declared
- The number of CANmotion services used (the TPDO and RPDO associated with the chosen cyclic synchronous mode, and the asynchronous messaging SDO and optional TPDO if selected).

For example: Using an axis in CSV or in CST mode will require more time as 1 extra RDO and 1 extra TPDO are exchanged compared to the CSP mode.

The table below explains how to calculate the CANmotion theoretical minimum time for the physical exchange of PDO and SDO data:.

Baud rate		**250 Kbit/s**	**500 Kbit/s**	**1 Mbit/s**
CANmotion service	**Basis**	**Time consumption (ms)**		
Default traffic (Synchronous bit + reserved SDO exchange + others)	per CANmotion	1.01	0.66	0.48
Drive in CSP mode only	per drive	0.92	0.46	0.23
Drive in CST or CSV mode	per drive	1.89	0.95	0.47
Optional TPDO	header (per TPDO per drive)	0.19	0.10	0.05
	per byte per TPDO	0.04	0.02	0.01

Example of calculation:

Description	**Value**
CANmotion with a baud rate of 1 Mbit/s	0.48
Drive 1 and drive 2 are in CSP. Drive 3 in CST	2x0.23 + 0.47
1 optional TPDO is configured to read torque value (2 bytes) of drive 1	0.05 + 2x0.01
1 optional TPDO is configured to read torque value (4 bytes) of drive 3	0.05 + 4x0.01
Minimum time required to physically exchange data between the controller and motion devices	**1.57 ms.**

In the example above, the minimum time required for the physical data exchange is 1.57 ms. This value must be compared to the tested duration of the Motion Task processing. Define a **Cycle period** at least 20% longer than the larger of the two values to accommodate variations in the duration of the Motion Task and of data exchanges.

NOTE: The performance values provided are calculated based upon the assumption that you have complied with all installation instructions for the equipment. The actual performances can vary depending on such factors as electromagnetic interference, wiring quality and conformance to CAN wiring guides, and a daisy chain topology.

Adding a CANmotion Device

To add a CANmotion slave device, proceed as follows:

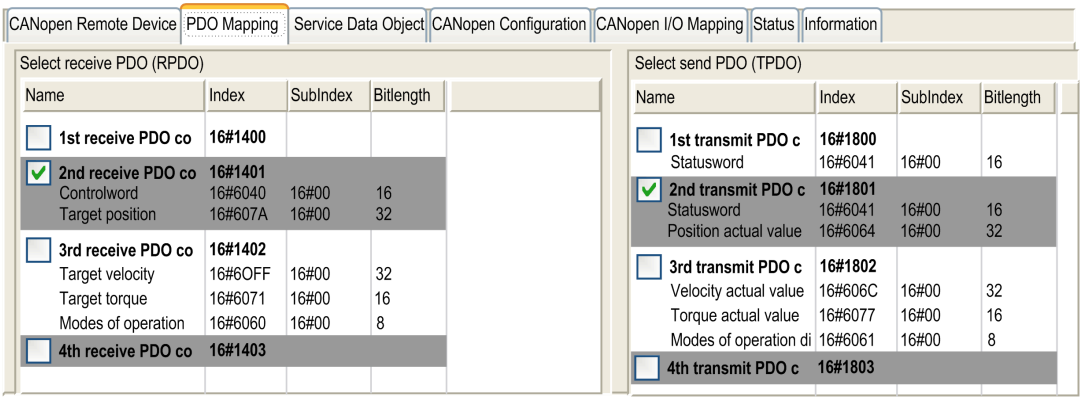
Step	Action
1	Select the **Devices & Modules** tab in the **Hardware Catalog**.
2	Select **Communication → CANopen** and choose the **CANmotion** slave device to add, drag-and-drop it onto the **CANx** node of the **Devices tree**.

CANmotion Lexium 32 Device Configuration

To add and configure the CANmotion Lexium32 Device, proceed as follows:

Step	Action
1	Select the **Devices & Modules** tab in the **Hardware Catalog** and click **Motor Control**.
2	Select **Servos → Lexium 32 A** in the list, drag-and-drop the item onto the **CANmotion** node of the **Devices tree**.
3	In the **Devices tree**, double-click **Lexium 32 A**. **Result:** The **Lexium_32_A** configuration window appears.

PDO mapping screen:



The default **PDO mapping** configuration of CANmotion slave is suitable for most implementations of the Cyclic Synchronous Position (CSP). If cyclic synchronous torque mode, cyclic synchronous velocity mode or optional TPDO are needed the PDO mapping has to be modified as follows:

	Default status	Function
1st Receive PDO	Not activated by default	Not used. Do not activate these options.
1st Transmit PDO		
2nd Receive PDO	Activated by default	Used for all modes (CSP, CSV & CST) Cannot be deactivated Mapping cannot be modified
2nd Transmit PDO		
3rd Receive PDO	Not activated by default	Used for CST and CSV only Should be activated only if CST or CSV are needed Mapping cannot be modified
4th Transmit PDO	Not activated by default	Optional TPDO should be activated only if an optional TPDO is needed Mapping can be modified in the window **Send PDO Mapping** only accessible if **Enable Expert PDO Settings** is enabled: For more information, refer to SoMachine online help, chapter Programming with SoMachine / Device Editors / CANbus Configuration Editor / CANopen Device.

NOTE: The first Receive PDO and first Transmit PDO mapping options are deactivated by default. If activated, they can cause saturation of the CANmotion bus traffic and delay or prevent motion commands to devices on the bus.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

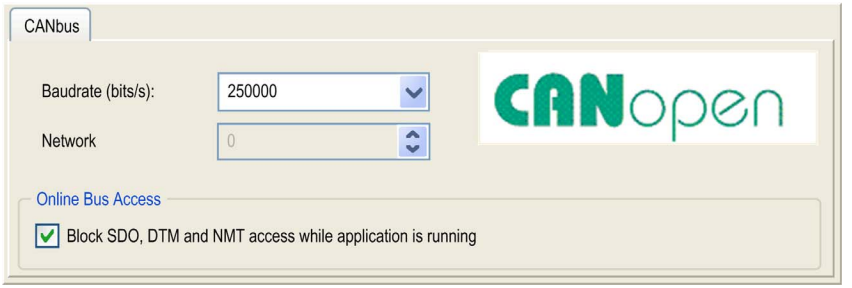
Do not activate the first Receive PDO or first Transmit PDO options.

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

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 **CAN0** or **CAN1**.
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

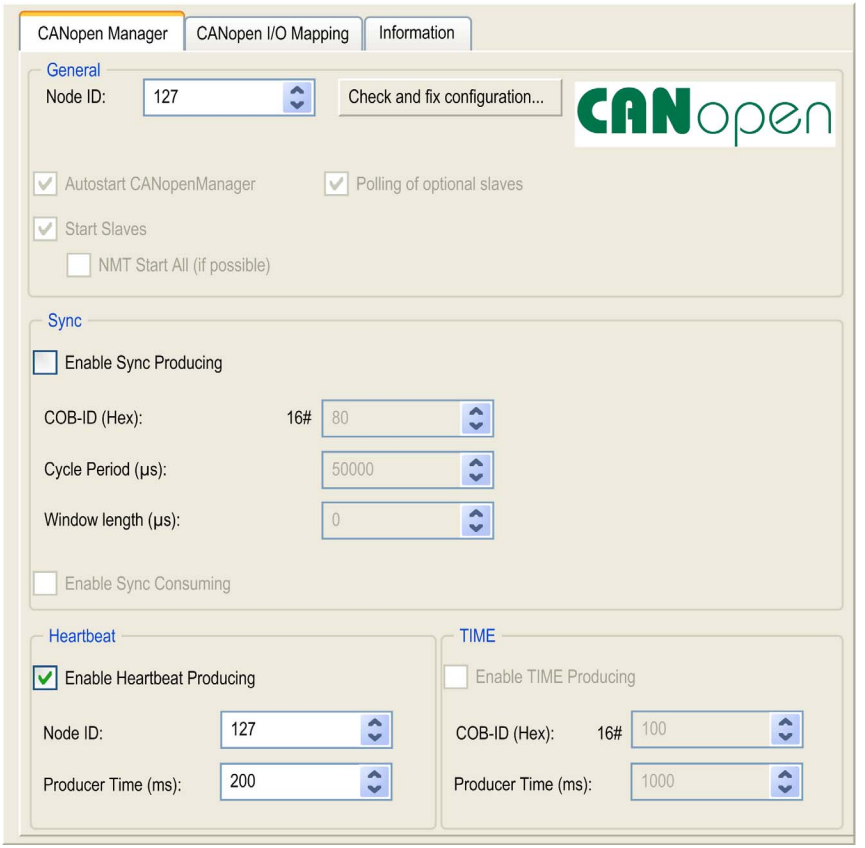
UNINTENDED EQUIPMENT OPERATION

You must consider the impact of DTM connections on the CANopen fieldbus load.

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	Select **CANopen Optimized** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes. For more information on adding a device to your project, refer to: ● Using the Drag-and-Drop Method *(see SoMachine, Programming Guide)* ● Using the Contextual Menu or Plus button *(see SoMachine, Programming Guide)*
2	Double-click **CANopen_Performance**. **Result:** The **CANopen Manager** configuration window appears: 

NOTE: If **Enable Sync Producing** is checked, the **CANx_Sync** task is added.

Do not delete or change the **Name**, **Type**, or **External event** attributes of **CANx_Sync** tasks. If you do so, SoMachine 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** sub-tab of the **CANopen_Performance** tab, the **CANx_Sync** task will automatically be deleted from your program.

Adding a CANopen Device

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

CANopen Operating Limits

The Modicon LMC058 Motion Controller CANopen master has the following operating limits:

Maximum number of slave devices	63
Maximum number of Received PDO (RPDO)	126
Maximum number of Transmitted PDO (TPDO)	126

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not connect more than 63 CANopen slave devices to the controller.
- Program your application to use 126 or fewer Transmit PDO (TPDO).
- Program your application to use 126 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 13
Serial Line Configuration

Introduction

This chapter describes how to configure the serial line communication of the Modicon LMC058 Motion Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Serial Line Configuration	226
ASCII Manager	228
SoMachine Network Manager	230
Modbus Serial IOScanner	231
Adding a Device on the Modbus Serial IOScanner	233
Modbus Manager	240
Adding a Modem to a Manager	244

Serial Line Configuration

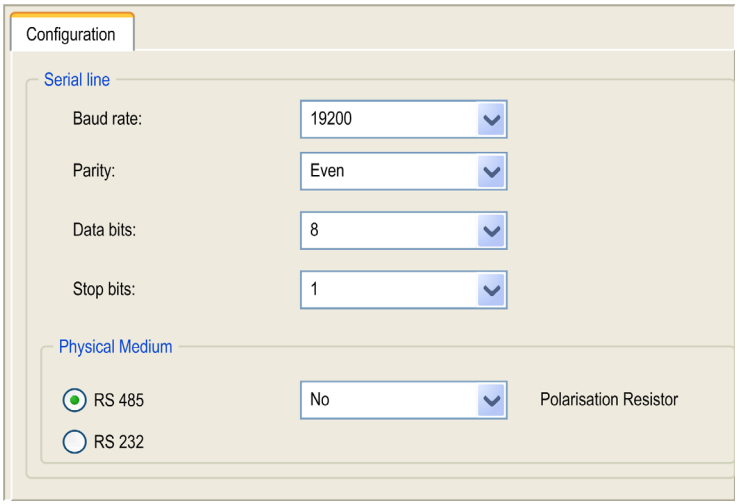
Introduction

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 port(s) of your controller are configured for the SoMachine protocol by default when new or when you update the controller firmware.

The serial line ports of your controller are configured for the SoMachine protocol by default when new or when you update the controller firmware. The SoMachine 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)
SoMachine Network Manager	115200
Modbus Manager	38400
ASCII Manager	
Modbus IOScanner	

ASCII Manager

Introduction

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

Adding the Manager

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

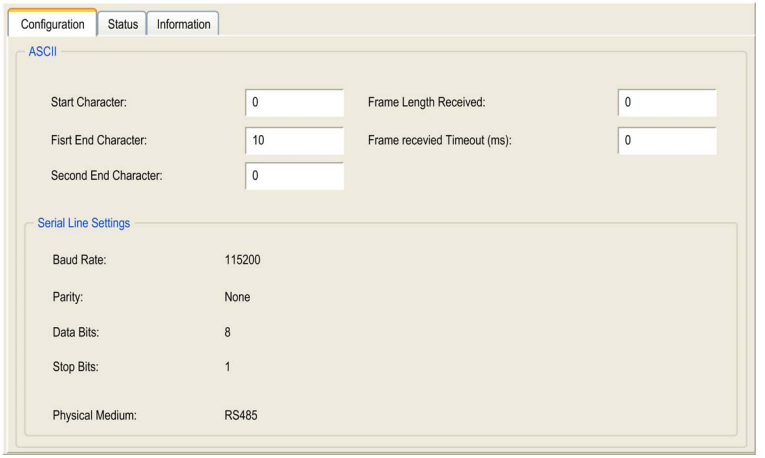
For more information on adding a device to your project, refer to:

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

ASCII Manager Configuration

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

The ASCII Manager configuration window is displayed as below:



Set the parameters as described in this table:

Parameter	Description
Start Character	If 0, no start character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the beginning of a frame. In **Sending Mode**, this character is added at the beginning of the frame.
First End Character	If 0, no first end character is used in the frame. Otherwise, in **Receiving Mode**, the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.

Parameter	Description
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 226)*.

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

Adding a Modem

To add a Modem to the ASCII manager, refer to Adding a Modem to a Manager *(see page 244)*.

SoMachine Network Manager

Introduction

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

Adding the Manager

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

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

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

Configuring the Manager

There is no configuration for SoMachine Network Manager.

Adding a Modem

To add a modem to the SoMachine Network Manager, refer to Adding a Modem to a Manager *(see page 244)*.

Modbus Serial IOScanner

Introduction

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

Add a Modbus IOScanner

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

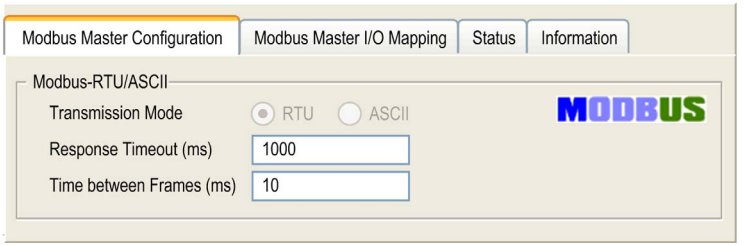
For more information on adding a device to your project, refer to:

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

Modbus IOScanner Configuration

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

The configuration window is displayed as below:



Set the parameters as described in this table:

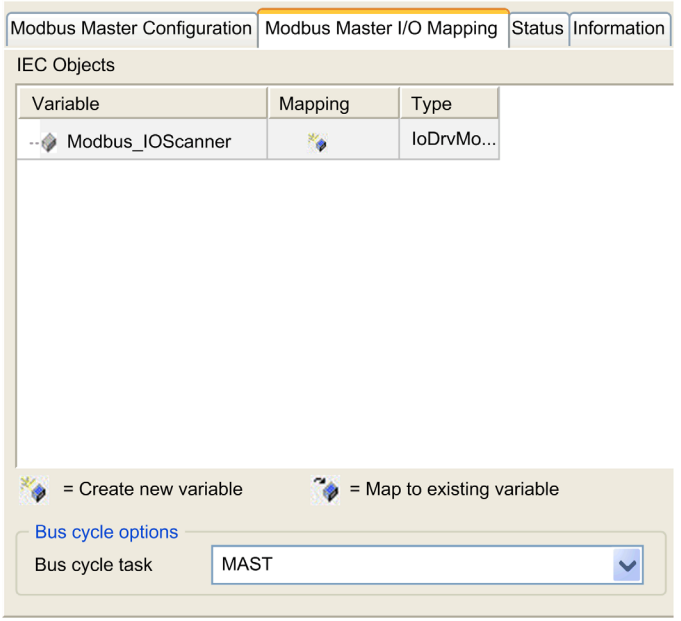
Element	Description
Transmission Mode	Specifies the transmission mode to use: ● RTU: uses binary coding and CRC error-checking (8 data bits) ● ASCII: messages are in ASCII format, LRC error-checking (7 data bits) Set this parameter identical for each Modbus device on the network.
Response Timeout (ms)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

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

Bus Cycle Task Selection

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

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



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

- **Use parent bus cycle setting**: associate the scanner with the application task that manages the controller.
- **MAST**: associate the scanner with the MAST task.
- Another existing task: you can select an existing task and associate it to the scanner. For more information about the application tasks, refer to the SoMachine 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.

Add a Device on the Modbus IOScanner

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

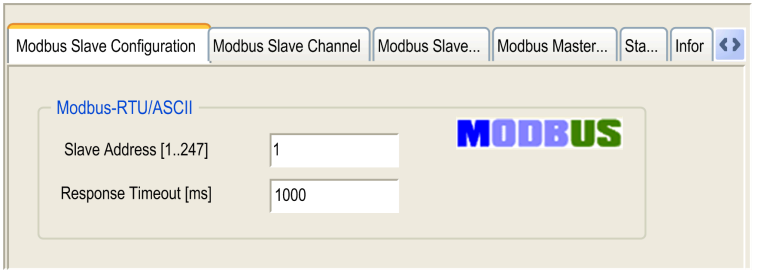
For more information on adding a device to your project, refer to:

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

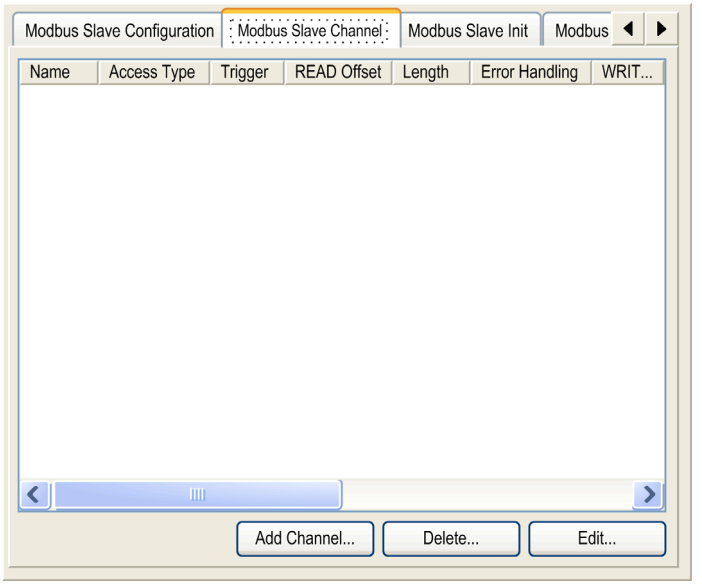
NOTE: The variable for the exchange is automatically created in the `%IWx` and `%QWx` of the **Modbus Serial Master I/O Mapping** tab.

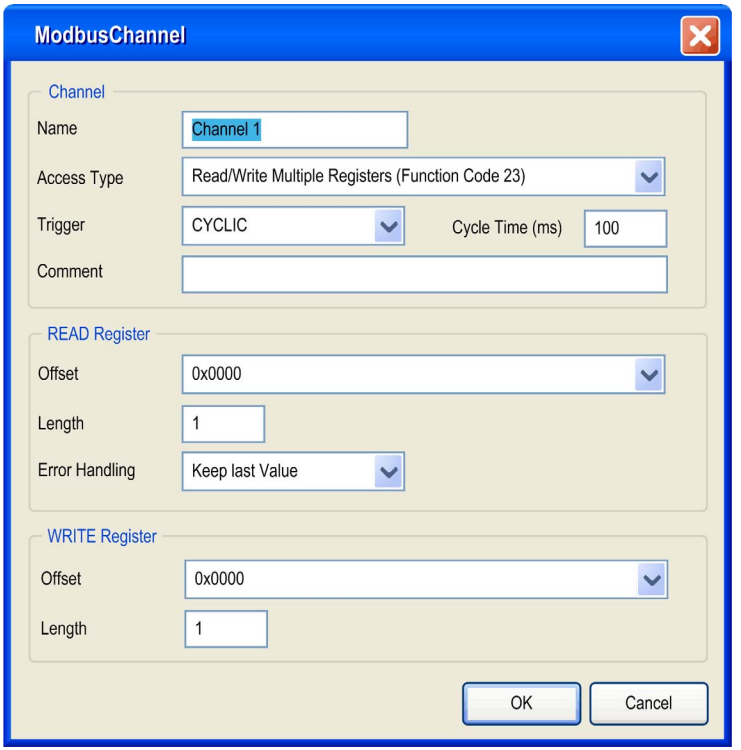
Configure a Device Added on the Modbus IOScanner

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

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).

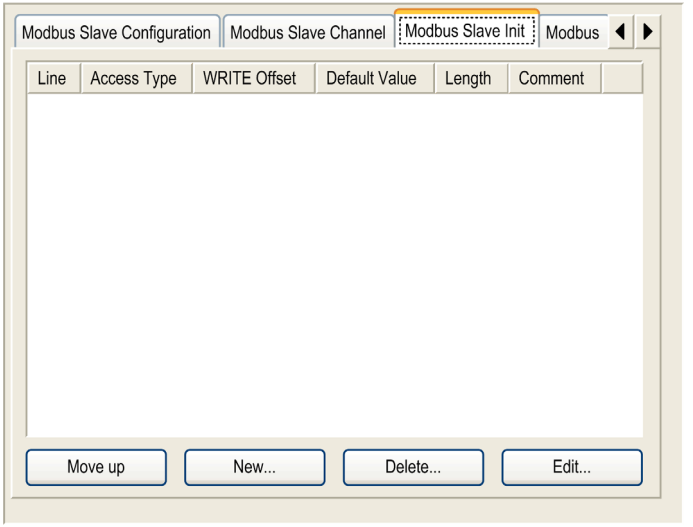
To configure the **Modbus Channels**, proceed as follow:

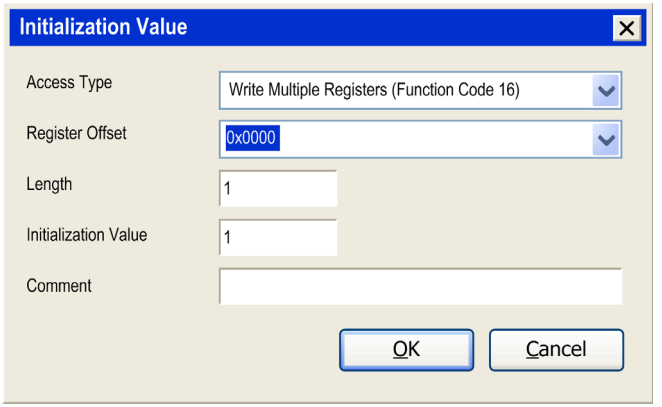
Step	Action
1	Click the **Modbus Slave Channel** tab: 

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

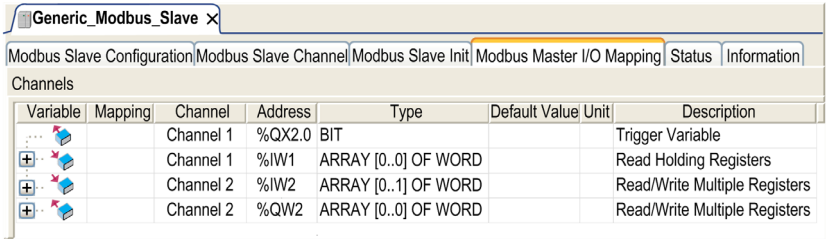
Step	Action
3	Configure an exchange: In the field **Channel**, you can add the following values: ● **Channel**: Enter a name for your channel. ● **Access Type**: Choose the exchange type: Read or Write or Read/Write multiple registers (i.e. %MW) *(see page 239)*. ● **Trigger**: Choose the trigger of the exchange. It can be either **CYCLIC** with the period defined in **Cycle Time (ms)** field or started by a **RISING EDGE** on a boolean variable (this boolean variable is then created in the **Modbus Master I/O Mapping** tab). ● **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 will be mapped on %IW (see Modbus Master I/O Mapping tab): ● **Offset**: Offset of the %MW to read. 0 means that the first object that will be read will be %MW0. ● **Length**: Number of %MW to be read. For example, if 'Offset' = 2 and 'Length' = 3, the channel will read %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 will be mapped on %QW (see Modbus Master I/O Mapping tab): ● **Offset**: Offset of the %MW to write. 0 means that the first object that will be written will be %MW0. ● **Length**: Number of %MW to be written. For example, if 'Offset' = 2 and 'Length' = 3, the channel will write %MW2, %MW3 and %MW4.
5	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 follow:

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

Step	Action
2	Click **New** to create a new initialization value:  The **Initialization Value** window contains the following parameters: • **Access Type**: Choose the exchange type: Read or Write or Read/Write multiple registers (that is, %MW) *(see page 239)*. • **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 will read %MW2, %MW3 and %MW4. • **Initialization Value**: Value the registers are initialized with. • **Comment**: Add a comment about this channel.
4	Click **OK** to create a new **Initialization Value**. **NOTE:** You can also: • Click **Move up** 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 follow:

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 SoMachine 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**

Modbus Manager

Introduction

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

Adding the Manager

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

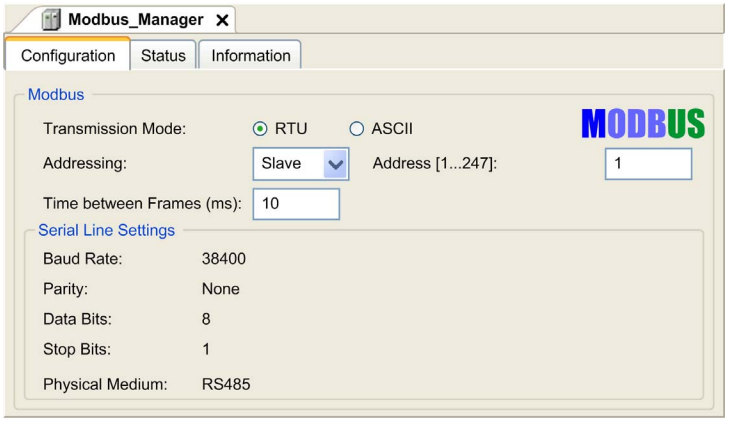
For more information on adding a device to your project, refer to:

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

Modbus Manager Configuration

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

The Modbus Manager configuration window is displayed as below:



Set the parameters as described in this table:

Element	Description
Transmission Mode	Specify the transmission mode to use: ● RTU: uses binary coding and CRC error-checking (8 data bits) ● ASCII: messages are in ASCII format, LRC error-checking (7 data bits) Set this parameter identical for each Modbus device on the link.
Addressing	Specify the device type: ● Master ● Slave

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 SoMachine, 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)
5 (5 hex)	–	Write single coil (%M)
6 (6 hex)	–	Write single register (%MW)
8 (8 hex)	–	Diagnostic
15 (F hex)	–	Write multiple digital outputs (%Q)
16 (10 hex)	–	Write multiple registers (%MW)
23 (17 hex)	–	Read/write multiple registers (%MW)
43 (2B hex)	14 (E hex)	Read device identification

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

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

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

Object ID	Object Name	Type	Value
00 hex	Vendor code	ASCII String	Schneider Electric
01 hex	Product code	ASCII String	Controller reference eg: LMC058LF42
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

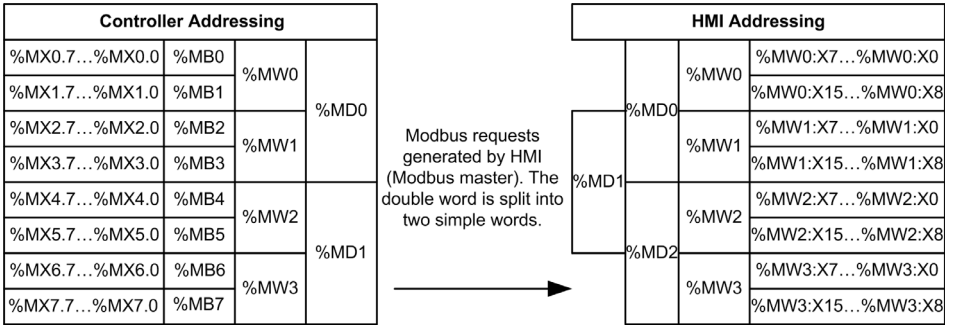
UNINTENDED EQUIPMENT OPERATION

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 244)*.

Adding a Modem to a Manager

Introduction

A modem can be added to the following managers:

- ASCII Manager
- Modbus Manager
- SoMachine Network Manager

NOTE: Use Modem TDW-33 (which implements AT & A1 commands) if you need a modem connexion with SoMachine Network Manager.

Adding a Modem to a Manager

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

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

• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

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

Chapter 14
Post Configuration

Introduction

This chapter describes how to generate and configure the post configuration file of the Modicon LMC058 Motion Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Post Configuration Presentation	246
Post Configuration File Management	248
Post Configuration Example	250

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). All parameters in the Post Configuration file without corresponding hardware are ignored (for example, a PCI module configuration without a PCI module).

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
-

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 (PCI 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 device.
For example, if the IP address used by an HMI is updated in the configuration with a Post Configuration file, the HMI will still use 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 67)*
- after a Reset Cold command *(see page 68)*
- after a reboot *(see page 69)*
- after an application download *(see page 71)*

Refer to Controller States and Behaviors *(see page 51)* 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 specified by a variable type, variable ID, and value. The format is:

```
id[moduleType].param[paramId].paramField=value
```

where:

- `moduleType` is a numerical value, for example 111.
- `paramId` is a numerical value specifying the parameter to be modified, for example 10000.
- `paramField` is a string value that must be used in addition to the `paramId` to specify serial line parameters, for example, “Bauds”.
- `value` is the value assigned to the parameter. Its type depends on the parameter data type.

Each parameter will be defined on 3 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 SoMachine.

To generate the file, proceed as follows:

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

NOTE: When you use SoMachine to create a Post Configuration file, it reads the value of each parameter currently assigned in your application program and then writes the new files using these values. This automatically generated a file explicitly assigns a value to every parameter that can be specified via Post configuration. 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. Retain only those parameters assignments that you wish changed by the Post configuration function that are necessary to make you application portable.

Post Configuration File Transfer

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

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

- USB Memory key *(see page 258)* (with the proper script)
- Download through the FTP server *(see page 169)*
- Download with SoMachine controller device editor *(see page 76)*

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 148)*.

To modify the Post Configuration file in the controller with SoMachine 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**.

Deleting the Post Configuration File

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

- USB Memory key *(see page 255)* (with the delete script)
- Through the FTP server *(see page 169)*
- Online with SoMachine controller device editor *(see page 76)*, **Files** tab

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

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

- A Reset Warm command *(see page 67)*
- A Reset Cold command *(see page 68)*
- A reboot *(see page 69)*
- An application download *(see page 71)*

Post Configuration Example

Post Configuration File Example

```
# LMC058LF424 / Ethernet / IPAddress
# Ethernet IP address
id[111].param[0] = [0, 0, 0, 0]

# LMC058LF424 / Ethernet / SubnetMask
# Ethernet IP mask
id[111].param[1] = [0, 0, 0, 0]

# LMC058LF424 / Ethernet / GatewayAddress
# Ethernet IP gateway address
id[111].param[2] = [0, 0, 0, 0]

# LMC058LF424 / Ethernet / IPConfigMode
# IP configuration mode: 0:FIXED 1:BOOTP 2:DHCP
id[111].param[4] = 0

# LMC058LF424 / Ethernet / DeviceName
# Name of the device on the Ethernet network
id[111].param[5] = 'my Device'

# LMC058LF424 / Serial Line / Serial Line Configuration / Baudrate
# Serial Line Baud Rate in bit/s
id[40101].param[10000].Bauds = 115200

# LMC058LF424 / Serial Line / Serial Line Configuration / Parity
# Serial Line Parity (0=None, 1=Odd, 2=Even)
id[40101].param[10000].Parity = 0

# LMC058LF424 / Serial Line / Serial Line Configuration / Data bits
# Serial Line Data bits (7 or 8)
id[40101].param[10000].DataFormat = 8
```

```
# LMC058LF424 / Serial Line / Serial Line Configuration / Stop bits
# Serial Line Stop bits (1 or 2)
id[40101].param[10000].StopBit = 1
# LMC058LF424 / PCI Slots / BusAddr
# Profibus station address
id[42000].pos[1].id[34].param[100] = 2
# LMC058LF424 / PCI Slots / BaudRate
# Profibus Baud Rate (0: 9.6, 1: 19.2, 11: 45.45, 2: 93.75, 3: 187.5, 4:
500, 6: 1500, 7: 3000, 8: 6000, 9: 12000, 15: Auto)
id[42000].pos[1].id[34].param[101] = 15
```

Chapter 15
Connecting a Modicon LMC058 Motion 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 SoMachine 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 Mini-B Port Connection

TCSXCNAMUM3P: This USB cable is suitable for short duration connections such as quick updates or retrieving data values.

BMXXCAUSBH045: Grounded and shielded, this USB cable is suitable for long duration connections.

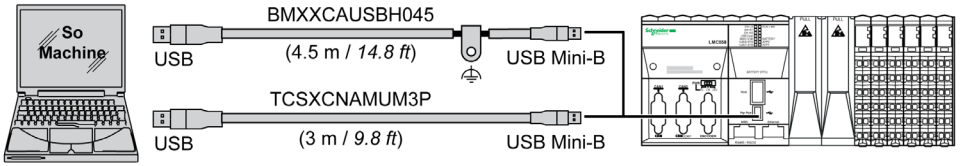
NOTE: You can only connect 1 controller or any other device associated with SoMachine and its component to the PC at any one time.

The USB Mini-B Port is the programming port you can use to connect a PC with a USB host port using SoMachine 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 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.

The following illustration shows the USB connection to a PC:



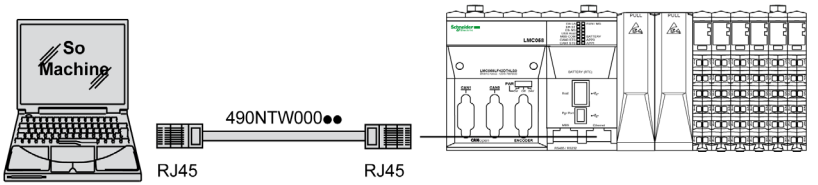
To connect the USB cable to your controller, follow the steps below:

Step	Action
1	**1a** If making a long-term connection using the cable BMXXCAUSBH045, or other cable with a ground shield connection, 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 the USB cable connector to the PC.
3	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.

The following illustration shows the Ethernet connection to a PC:



To connect the controller to the PC, do the following:

Step	Action
1	Connect the Ethernet cable to the PC.
2	Connect the Ethernet cable to the Ethernet port on the controller.

Chapter 16
Transfer by USB memory Key

Introduction

This chapter describes how to transfer firmware, application, using an USB memory key to the Modicon LMC058 Motion Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Changing Modicon LMC058 Motion Controller Firmware	256
File Transfer with USB Memory Key	258

Changing Modicon LMC058 Motion Controller Firmware

Introduction

The firmware updates for Modicon LMC058 Motion Controller are available on the http://www.schneider-electric.com website (in .zip format).

You can change the firmware using the USB memory key (with compatible script file).

NOTE: Changing the firmware is also possible using **SoMachine V4 → Central → Maintenance → Controller Assistant**. Refer to *SoMachine Controller Assistant User Guide*.

NOTE: The controller can be in RUNNING state during firmware download.

Performing a firmware update will delete 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.**

The serial line port(s) of your controller are configured for the SoMachine protocol by default when new or when you update the controller firmware. The SoMachine 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. Ensure 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 port(s) 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.**

Changing by USB Memory Key Management

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 (logic controller, motion controller, HMI controller or drive) into service until the file transfer has completed successfully. **Failure to follow these instructions can result in equipment damage.**

Step	Action
1	Extract the zip file on the root of the USB memory key. **NOTE:** The folder \sys\CMD\ contains the download script file.
2	Power OFF
3	Insert the USB memory key into the controller.
4	Power ON **NOTE:** The USB Host LED is flashing green and the other controller LEDs are switched OFF during download. **NOTE:** All controller LEDs may be switched OFF for up to 4 minutes during the download. Do not interrupt the procedure. If your USB memory key has an integrated activity LED, this LED flashes during the procedure to indicate normal activity.
5	Wait until the end of the download (USB Host LED is solid ON): • If an error has been detected, the USB Host LED is red. • If the download ended successfully, the USB Host LED is solid green.
6	Remove the USB memory key. The controller reboots automatically with new firmware if the download ended successfully. **NOTE:** If the controller reboots with its LEDs OFF, go back to Step 2.

NOTE: If you use exclusively the USB memory key to upgrade either the firmware or the application in memory, you need to have pre-configured and wired the Run/Stop input to restart your controller after the download. After the download and re-applying power, the controller will be in a STOPPED state, provided the other conditions of the boot sequence allows this to occur.

File Transfer with USB Memory Key

Introduction

The Modicon LMC058 Motion Controller allows file transfers with a USB memory key. Using this key, it is not necessary to use SoMachine or an FTP Server.

To upload or download files to the controller with a USB memory key, use one of the following methods:

- The clone function (use of an empty USB memory key)
- A script stored in the USB memory key

When a USB memory key is inserted into the USB Data Port of the controller, the firmware searches and executes the script contained in the USB memory key (/sys/CMD/Script.cmd).

NOTE: The controller operation is not modified during file transfer.

The **Mass Storage (USB or SDCard)** editor lets you generate and copy the script and all necessary files into the USB memory key.

NOTE: The Modicon LMC058 Motion Controller accepts only USB key formatted in FAT or FAT32.

⚠ 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 (logic controller, motion controller, HMI controller or drive) 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)*. However, for security reasons, it does not duplicate the Web Server/FTP password, nor any user access-rights, on any targeted machine.

NOTE: Retained and retained persistent data are not duplicated when using a controller firmware version prior to V3.1.

NOTE: Ensure access-rights are disabled in the source controller before doing a clone operation. For more details about Access Rights, refer to the SoMachine Programming Guide.

The following procedure describes how to upload on your USB memory key the current application stored in the controller:

Step	Action
1	Remove power from the controller.
2	Plug the USB memory key into the controller. **NOTE:** The USB memory key must be empty to perform this procedure.
3	Restore power to the controller.
4	The clone operation is in progress. **NOTE:** The USB LED is green and flashing during upload. At the end of the upload the USB LED is fixed green, if an error was detected the LED is red.
5	Remove the USB memory key.

The following procedure describes how to download to your controller the current application stored in the USB memory key:

Step	Action
1	Remove power from the controller.
2	Insert the USB memory key into the controller.
3	Restore power to the controller.
4	The clone operation is in progress. **NOTE:** The USB LED is green and flashing during download. At the end of the download the USB LED is fixed green, if an error was detected the LED is solid red.
5	Remove the USB memory key to reboot the controller. **NOTE:** The controller model must match the **<HardwareRef>.srd** file located on the USB memory key (`Usr/dta`) to restore retained and persistent data.

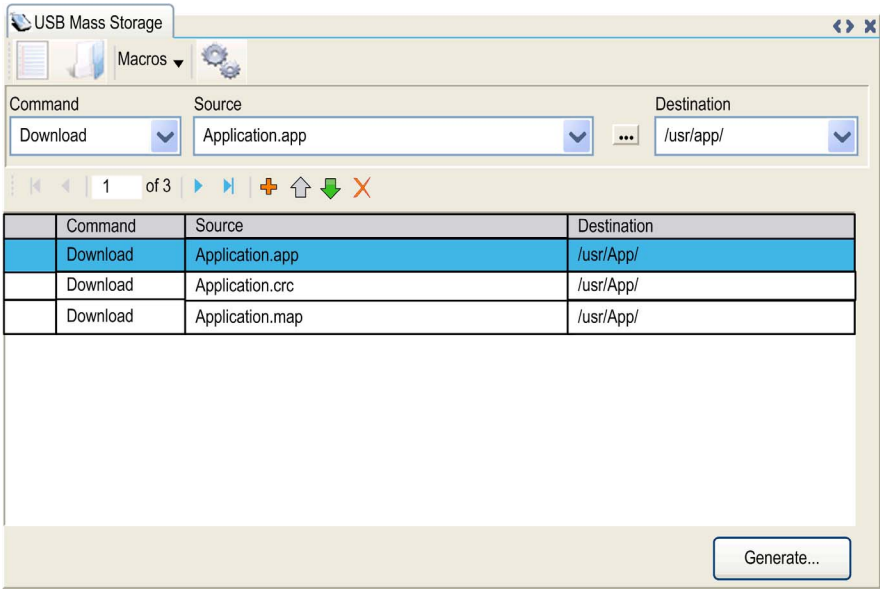
NOTE: The controller model must match the **<HardwareRef>.srd** file located on the USB memory key (`Usr/dta`) to restore retained and persistent data.

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 SoMachine 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.
Generate	Generate the script and all necessary files on the USB memory key.
Command	List of basic instructions.
Source	Source directory on the PC or the controller.

Element	Description
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.

Command descriptions:

Command	Description	Source	Destination	Syntax
Download	Download a file from the USB memory key 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"'`
Upload	Upload files contained in a controller directory to the USB memory key.	Select the directory.	-	`'Upload "/usr/*"'`
Delete	Delete files contained in a controller directory. **NOTE:** Delete "*" does not delete system files.	Select the directory and enter a specific file name **Important:** by default, all directory files are selected.	-	`'Delete "/usr/SysLog/*"'`
	Removes the user access-rights from the controller	-	-	`'Delete "/usr/*"'`
Reboot	Reboot the controller (only available at the end of the script).	-	-	`'Reboot'`

NOTE: When Access 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 will be disabled (this includes the clone operation). For more details about Access Rights, refer to the SoMachine Programming Guide.

Macros description

Macros	Description	Directory/Files

Download App	Download the application from the USB memory key to the controller.	`/usr/App/*.app`
		`/usr/App/*.crc`
		`/usr/App/*.map`

Upload App	Upload the application from the controller to the USB memory key.	

Macros	Description	Directory/Files
Download Sources	Download the project archive from the USB memory key to the controller.	`/usr/App/*.prj`
Upload Sources	Upload the project archive from the controller to the USB memory key.	
Download Multi-files	Download multiple files from the USB memory key to a controller directory.	Defined by user
Upload Log	Upload the log files from the controller to the USB memory key.	`/usr/Log/*.log`

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 USB memory key root directory. **Result**: The script and files are transferred on the USB memory key.
3	Plug the USB memory key into the controller. **NOTE:** The USB LED blinks green during transfer. At the end of the transfer the USB LED is solid green. If an error was detected the LED is solid red. When the controller has executed the script, the result is logged on the USB memory key (file `/sys/CMD/Cmd.log`).
4	Remove the USB memory key. **NOTE:** A reboot is required to record the new application.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Consult the controller state and behavior diagram *(see page 53)* 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.

Chapter 17
Compatibility

Software and Firmware Compatibilities

SoMachine Compatibility and Migration

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

Appendices

Overview

This appendix lists the documents necessary for technical understanding of the Modicon LMC058 Motion Controller Programming Guide.

What Is in This Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	Functions to Get/Set Serial Line Configuration in User Program	269
B	How to change the IP address of the controller	275
C	Controller Performance	279

Appendix A
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 SoMachine Programming Guide.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
GetSerialConf: Get the Serial Line Configuration	270
SetSerialConf: Change the Serial Line Configuration	271
SERIAL_CONF: Structure of the Serial Line Configuration Data Type	273

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 SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide)*	`Link` is the communication port number.
`PointerToSerialConf`	POINTER TO `SERIAL_CONF` *(see page 273)*	`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 272)* 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 UNEXPECTED 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 273)*	`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/Somachine 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 SoMachine (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: SoMachine (when Protocol = 0)
CodesysNetType	BYTE	not used

Appendix B
How to change the IP address of the controller

changeIPAddress: Change the IP address of the controller

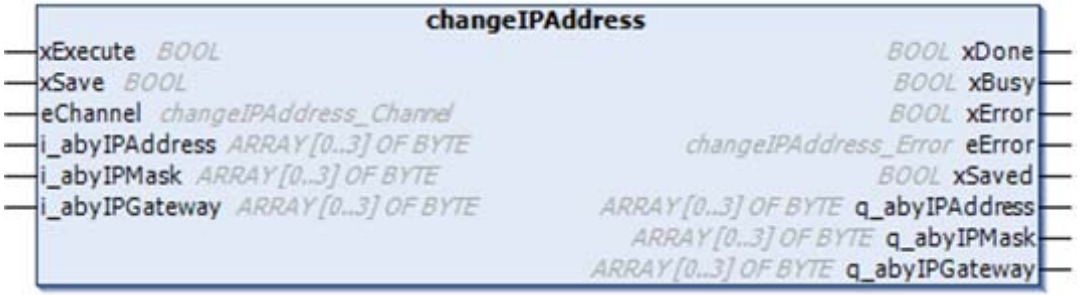
Function Block Description

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 141)*.

NOTE: For more information on the function block, use the **Documentation** tab of SoMachine Library Manager Editor. For the use of this editor, refer SoMachine Programming 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.

Input	Type	Comment
eChannel	changeIPAddress_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 2 values *(see page 276)* 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 144)* is configured.
i_abyIPMask	ARRAY[0..3] OF BYTE	The new subnet mask. Format: 0.0.0.0
i_abyIPGateway	ARRAY[0..3] OF BYTE	The new gateway IP address. Format: 0.0.0.0

Output	Type	Comment
xDone	BOOL	TRUE: if IP Addresses have been successfully configured or if default IP Addresses have been successfully configured because input i_abyIPAddress is set to 0.0.0.0.
xBusy	BOOL	Function block active.
xError	BOOL	• TRUE: error detected, function block aborts action. • FALSE: no error has been detected.
eError	changeIPAddress_Error	Error code of the detected error *(see page 277)*.
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.

Appendix C
Controller Performance

Processing Performance

Introduction

This chapter provides information about the LMC058 processing performance.

Logic Processing

This table presents logic processing performance for various logical instructions:

IL Instruction Type	Duration for 1000 Instructions
Addition/subtraction/multiplication of INT	42 µs
Addition/subtraction/multiplication of DINT	41 µs
Addition/subtraction/multiplication of REAL	336 µs
Division of REAL	678 µs
Operation on BOOLEAN, for example, Status:= Status and value	75 µs
LD INT + ST INT	64 µs
LD DINT + ST DINT	49 µs
LD REAL + ST REAL	50 µs

Communication and System Processing Time

The communication processing time varies, depending on the number of sent/received requests.

Response Time on Event

The response time presented in the following table represents the time between a signal rising edge on an input triggering an external task and the edge of an output set by this task. The event task also process 100 IL instructions before setting the output:

Minimum	Typical	Maximum
120 µs	200 µs	500 µs

Glossary

!

%I

According to the IEC standard, %I represents an input bit (for example, a language object of type digital IN).

%Q

According to the IEC standard, %Q represents an output bit (for example, a language object of type digital OUT).

A

analog input

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

analog output

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

application

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

CAN

(*controller area network*) A protocol (ISO 11898) for serial bus networks, designed for the interconnection of smart devices (from multiple manufacturers) in smart systems and for real-time industrial applications. Originally developed for use in automobiles, CAN is now used in a variety of industrial automation control environments.

CANmotion

A CANopen-based motion bus with an additional mechanism that provides synchronization between the motion controller and the drives.

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.

CIP

(*common industrial protocol*) When a CIP is implemented in a network application layer, it can communicate seamlessly with other CIP-based networks without regard to the protocol. For example, the implementation of CIP in the application layer of an Ethernet TCP/IP network creates an EtherNet/IP environment. Similarly, CIP in the application layer of a CAN network creates a DeviceNet environment. In that case, devices on the EtherNet/IP network can communicate with devices on the DeviceNet network through CIP bridges or routers.

compact I/O module

An inseparable group of 5 analog and/or digital I/O electronic modules in a single reference.

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).

CPDM

(*controller power distribution module*) The connection of the controller to the external 24 Vdc power supplies and the beginning of the power distribution for the local configuration.

CRC

(*cyclical redundancy check*) A method used to determine the validity of a communication transmission. The transmission contains a bit field that constitutes a checksum. The message is used to calculate the checksum by the transmitter according to the content of the message. Receiving nodes, then recalculate the field in the same manner. Any discrepancy in the value of the 2 CRC calculations indicates that the transmitted message and the received message are different.

cyclic task

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

D

data log

The controller logs events relative to the user application in a *data log*.

DHCP

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

digital I/O

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

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.

electronic module

In a programmable controller system, most electronic modules directly interface to the sensors, actuators, and external devices of the machine/process. This electronic module is the component that mounts in a bus base and provides electrical connections between the controller and the field devices. Electronic modules are offered in a variety of signal levels and capacities. (Some electronic modules are not I/O interfaces, including power distribution modules and transmitter/receiver modules.)

encoder

A device for length or angular measurement (linear or rotary encoders).

equipment

A part of a machine including sub-assemblies such as conveyors, turntables, and so on.

Ethernet

A physical and data link layer technology for LANs, also known as IEEE 802.3.

EtherNet/IP

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

expansion bus

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

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.

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

G

GVL

(*global variable list*) Manages global variables within a project.

H

HSC

(*high-speed counter*) A function that counts pulses on the controller or on expansion module inputs.

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.

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

LED

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

LINT

(*long integer*) A whole number encoded in a 64-bit format (4 times `INT` or 2 times `DINT`).

LRC

(*longitudinal redundancy checking*) An error-detection method for determining the correctness of transmitted and stored data.

LREAL

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

LWORD

(*long word*) A data type encoded in a 64-bit format.

M

MAC address

(*media access control address*) A unique 48-bit number associated with a specific piece of hardware. The MAC address is programmed into each network card or device when it is manufactured.

MAST

A processor task that is run through its programming software. The MAST task has 2 sections:

- **IN:** Inputs are copied to the IN section before execution of the MAST task.
- **OUT:** Outputs are copied to the OUT section after execution of the MAST task.

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).

minimum I/O update time

The time needed by the module or block to update I/O on the bus. If the bus cycle time is shorter than this minimum value, the I/O is updated on the bus at the next bus cycle time.

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.

O

OS

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

P

PCI

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

PDO

(*process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

PE

(*Protective Earth*) A common grounding connection to help avoid the hazard of electric shock by keeping any exposed conductive surface of a device at earth potential. To avoid possible voltage drop, no current is allowed to flow in this conductor (also referred to as *protective ground* in North America or as an equipment grounding conductor in the US national electrical code).

persistent data

Value of persistent data is used at next application change or cold start. Only get re-initialized at a reboot of the controller or reset origin. Especially, they maintain their values after a download.

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.

Profibus DP

(*Profibus decentralized peripheral*) An open bus system uses an electrical network based on a shielded 2-wire line or an optical network based on a fiber-optic cable. DP transmission allows for high-speed, cyclic exchange of data between the controller CPU and the distributed I/O devices.

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.

PWM

(*pulse width modulation*) A fast output that oscillates between off and on in an adjustable duty cycle, producing a rectangular wave form (though you can adjust it to produce a square wave).

R

REAL

A data type that is defined as a floating-point number encoded in a 32-bit format.

reflex output

Among the outputs of HSC are the reflex outputs associated to a threshold value that is compared to the counter value depending on the configuration of the HSC. The reflex outputs switch to either on or off depending on the configured relationship with the threshold.

RPDO

(*receive process data object* An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

RS-232

A standard type of serial communication bus, based on 3 wires (also known as EIA RS-232C or V.24).

RS-485

A standard type of serial communication bus, based on 2 wires (also known as EIA RS-485).

RTC

(*real-time clock*) A battery-backed time-of-day and calender clock that operates continuously, even when the controller is not powered for the life of the battery.

run

A command that causes the controller to scan the application program, read the physical inputs, and write to the physical outputs according to solution of the logic of the program.

S

scan

A function that includes:

- reading inputs and placing the values in memory
- executing the application program 1 instruction at a time and storing the results in memory
- using the results to update outputs

SDO

(*service data object*) A message used by the field bus master to access (read/write) the object directories of network nodes in CAN-based networks. SDO types include service SDOs (SSDOs) and client SDOs (CSDOs).

SFC

(*sequential function chart*) A language that is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

SINT

(*signed integer*) A 15-bit value plus sign.

SNMP

(*simple network management protocol*) A protocol that can control a network remotely by polling the devices for their status and viewing information related to data transmission. You can also use it to manage software and databases remotely. The protocol also permits active management tasks, such as modifying and applying a new configuration.

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

STOP

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

string

A variable that is a series of ASCII characters.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

TCP

(*transmission control protocol*) A connection-based transport layer protocol that provides a simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

TPDO

(*transmit process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

U

UDINT

(*unsigned double integer*) Encoded in 32 bits.

UDP

(*user datagram protocol*) A connectionless mode protocol (defined by IETF RFC 768) in which messages are delivered in a datagram (data telegram) to a destination computer on an IP network. The UDP protocol is typically bundled with the Internet protocol. UDP/IP messages do not expect a response, and are therefore ideal for applications in which dropped packets do not require retransmission (such as streaming video and networks that demand real-time performance).

UINT

(*unsigned integer*) Encoded in 16 bits.

V

variable

A memory unit that is addressed and modified by a program.

W

watchdog

A watchdog is a special timer used to ensure that programs do not overrun their allocated scan time. The watchdog timer is usually set to a higher value than the scan time and reset to 0 at the end of each scan cycle. If the watchdog timer reaches the preset value, for example, because the program is caught in an endless loop, an error is declared and the program stopped.

WORD

A type encoded in a 16-bit format.

Index

A

Add an Encoder
- Motion Encoder, *104*
- Standard Encoder, *104*

ASCII Manager, *228*

C

CANmotion
- Architecture with CANmotion, *213*
- configuration, *217*

CANmotion Cycle Time Configuration, *218*
CANmotion Device Configuration, *220*
CANmotion Manager, *218*
changeIPAddress, *275*
- changing the controller IP address, *275*

Controller Configuration
- Controller Selection, *78*
- PLC Settings, *80*
- Services, *82*

cyclic data exchanges, generating EDS file for, *186*

D

Download application, *71*

E

EDS file, generating, *186*
Ethernet
- changeIPAddress function block, *275*

EtherNet
- EtherNet/IP device, *185*

Ethernet
- FTP Server, *169*
- Modbus TCP Client/Server, *146*
- Modbus TCP slave device, *207*
- Services, *139*
- SNMP, *173*
- Web server, *148*

expansion modules and blocks
- check resources, *108*
- TM5 manager, *108*

External Event, *43*

F

firewall
- configuration, *178*
- default script file, *178*
- script commands, *180*

FTP client, *172*
FTP Server
- Ethernet, *169*

FTPRemoteFileHandling library, *172*

G

GetSerialConf
- getting the serial line configuration, *270*

H

Hardware Initialization Values, *64*

I

IP address
- changeIPAddress, *275*

L

libraries, *21*

Libraries
FTPRemoteFileHandling, *172*

M

M2•• communication
GetSerialConf, *270*
SetSerialConf, *271*
Memory Mapping, *25*
Modbus
Protocols, *146*
Modbus Ioscanner, *231*
Modbus Manager, *240*
Modbus TCP Client/Server
Ethernet, *146*
Motion
Performance, *46*
Programming Requirements, *46*

O

Output Behavior, *64*, *64*, *64*
Output Forcing, *64*

P

Post Configuration, *245*
baud rate, *246*, *246*
data bits, *246*
device name, *246*
Example, *250*
file management, *248*
gateway address, *246*
IP address, *246*
IP configuration mode, *246*
IP master name, *246*
parity, *246*
presentation, *246*
station address, *246*
stop bit, *246*
subnet mask, *246*
transfer rate, *246*
Protocols, *139*
IP, *141*
Modbus, *146*
protocols
SNMP, *173*

R

Reboot, *69*
Remanent variables, *73*
Reset cold, *68*
Reset origin, *68*
Reset warm, *67*
Run command, *66*

S

script commands
firewall, *180*
serial line
ASCII Manager, *228*
GetSerialConf, *270*
Modbus Manager, *240*
SetSerialConf, *271*
SERIAL_CONF, *273*
SetSerialConf, *271*
setting the serial line configuration, *271*
SNMP
Ethernet, *173*
protocols, *173*
Software Initialization Values, *64*
State diagram, *53*
Stop command, *66*

T

Task
Cyclic task, *41*
Event task, *43*
External Event Task, *43*
Freewheeling task, *42*
Types, *41*
Watchdogs, *47*
TM5 expansion modules
general description, *127*

W

Web server
Ethernet, *148*