Modicon M258 Logic Controller

Programming Guide

05/2010

EIO0000000402.01

www.schneider-electric.com

Schneider Electric

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

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

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

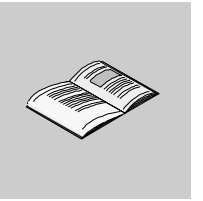
When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

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

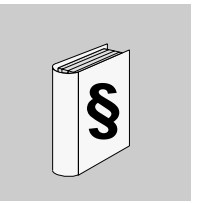
Failure to observe this information can result in injury or equipment damage.

© 2010 Schneider Electric. All rights reserved.

Table of Contents



Safety Information



Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

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

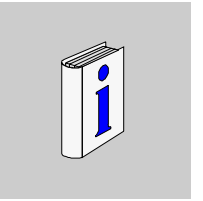
CAUTION
CAUTION, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** equipment damage.

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 the 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 M258 Logic Controller.

NOTE: Read and understand this document and all related documents *(see page 10)* before installing, operating, or maintaining your Modicon M258 Logic Controller.

The Modicon M258 Logic Controller users should read through the entire document to understand all features.

Validity Note

This document has been updated with the release of SoMachine V2.0.

Related Documents

Title of Documentation	Reference Number
Modicon M258 Logic Controller Hardware Guide	EIO0000000432 (ENG); EIO0000000433 (FRE); EIO0000000434 (GER); EIO0000000435 (SPA); EIO0000000436 (ITA); EIO0000000437 (CHS)
Modicon TM5 Expansion Modules Configuration Programming Guide	EIO0000000420 (ENG); EIO0000000421 (FRE); EIO0000000422 (GER); EIO0000000423 (SPA); EIO0000000424 (ITA); EIO0000000425 (CHS)
Modicon TM5 PCI Modules Configuration Programming Guide	EIO0000000590 (ENG); EIO0000000591 (FRE); EIO0000000592 (GER); EIO0000000593 (SPA); EIO0000000594 (ITA); EIO0000000595 (CHS)
Modicon M258 Motion Controller System Functions and Variables M258 PLCSystem Library Guide	EIO0000000584 (ENG); EIO0000000585 (FRE); EIO0000000586 (GER); EIO0000000587 (SPA); EIO0000000588 (ITA); EIO0000000589 (CHS)
Modicon M258 Motion Controller High Speed Counting M258 Expert I/O Library Guide	EIO0000000572 (ENG); EIO0000000573 (FRE); EIO0000000574 (GER); EIO0000000575 (SPA); EIO0000000576 (ITA); EIO0000000577 (CHS)
Modicon M258 Motion Controller Pulse Width Modulation M258 Expert I/O Library Guide	EIO0000000578 (ENG); EIO0000000579 (FRE); EIO0000000580 (GER); EIO0000000581 (SPA); EIO0000000582 (ITA); EIO0000000583 (CHS)

SoMachine Modbus and ASCII Read/Write Functions PLCCommunication Library Guide	EIO0000000361 (ENG); EIO0000000362 (FRE); EIO0000000363 (GER); EIO0000000364 (SPA); EIO0000000365 (ITA); EIO0000000366 (CHS)
SoMachine Data Logging Functions DataLogging Library Guide	EIO0000000551 (ENG); EIO0000000551 (FRE); EIO0000000551 (GER); EIO0000000551 (SPA); EIO0000000551 (ITA); EIO0000000551 (CHS)
SoMachine Modem Functions Modem Library Guide	EIO0000000552 (ENG); EIO0000000552 (FRE); EIO0000000552 (GER); EIO0000000552 (SPA); EIO0000000552 (ITA); EIO0000000552 (CHS)

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

Product Related Information

⚠ 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.

⚠ 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.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

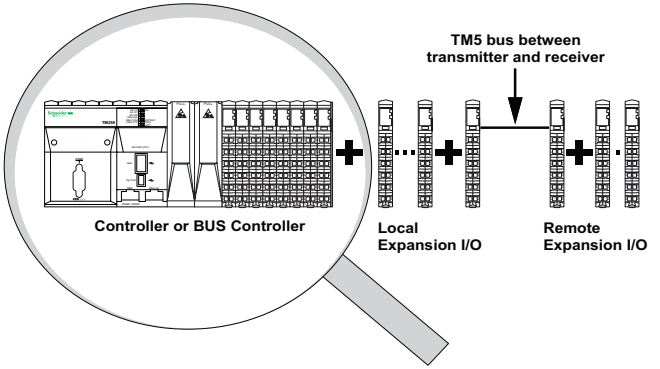
General Information

1

About the Modicon M258 Logic Controller

Overview

The Schneider Electric Modicon M258 Logic Controller is a controller with a variety of powerful features. It can control a wide range of applications.



The Software configuration is described in the SoMachine Programming Guide *(see page)*.

Key Features

The SoMachine software compatible with the controller provides the following IEC61131-3 programming languages:

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

All controllers include:
- CANopen Master
- Ethernet
- Serial Line
- Expert functions (counting, reflex outputs...)
- Embedded I/Os

All controllers support up to 21 tasks with the following limits:
- 4 cyclic tasks: one is configured by default (Mast)
- 1 freewheeling task
- 8 software event driven tasks
- 8 hardware event driven tasks

Controller Range

	PCI	CAN	USB A	USB Pgr	Ethernet	Serial Line
TM258LD42DT *(see M258 Logic Controller, Hardware Guide)*	0	0	1	1	1	1
TM258LD42DT4L *(see M258 Logic Controller, Hardware Guide)*	2	0	1	1	1	1
TM258LF42DT•• *(see M258 Logic Controller, Hardware Guide)*	0	1	1	1	1	1
TM258LF42DT4L•• *(see M258 Logic Controller, Hardware Guide)*	2	1	1	1	1	1
TM258LF66DT4L•• *(see M258 Logic Controller, Hardware Guide)*	2	1	1	1	1	1
TM258LF42DR•• *(see M258 Logic Controller, Hardware Guide)*	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**
TM258LD42DT *(see M258 Logic Controller, Hardware Guide)*	2x	5	2	2	1x	12	12	0
TM258LD42DT4L *(see M258 Logic Controller, Hardware Guide)*	2x	5	2	2	1x	12	12	4
TM258LF42DT•• *(see M258 Logic Controller, Hardware Guide)*	2x	5	2	2	1x	12	12	0
TM258LF42DT4L•• *(see M258 Logic Controller, Hardware Guide)*	2x	5	2	2	1x	12	12	4
TM258LF66DT4L•• *(see M258 Logic Controller, Hardware Guide)*	2x	5	2	2	2x	12	12	4
TM258LF42DR•• *(see M258 Logic Controller, Hardware Guide)*	2x	5	2	2	2x	6	6 Relays	0

How to Configure the Controller

2

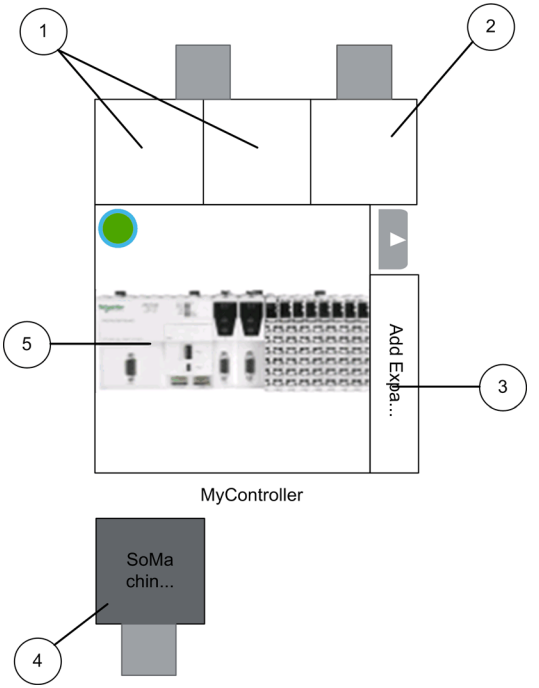
How to Configure the Controller

Introduction

Before configuring the controller, you must first create a new machine in the SoMachine software *(see SoMachine, Programming Guide)*.

Graphical Configuration Editor

In the Graphical Configuration Editor *(see SoMachine, Programming Guide)*, the controller is displayed as below:



Click on the following element to add (if empty) or replace objects:

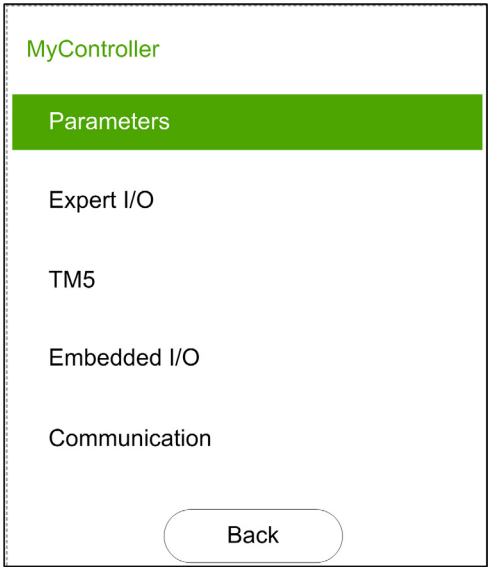
Element	Description
1	Ethernet port manager
2	CANopen port manager (CAN 0)
3	Expansion modules
4	Serial Line port manager (SoMachine_Network_Manager by default)
5	Access to the controller configuration screen (double click on the controller)

Controller Configuration Screen

To access to the controller configuration screen, proceed as follow:

Step	Action
1	Select the **Configuration** tab.
2	Double click the controller.

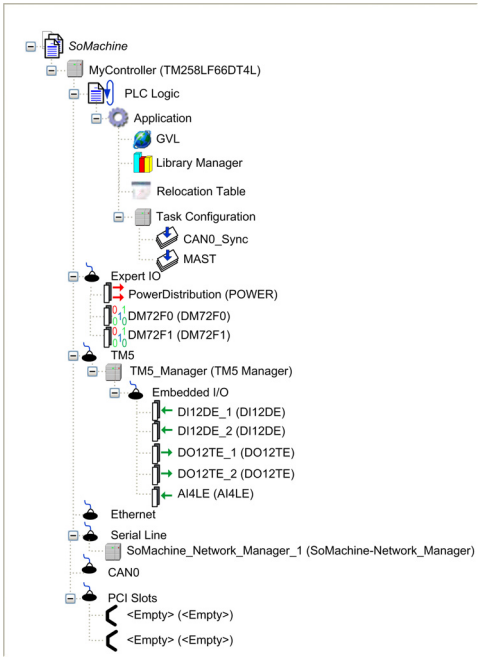
In the left hand side, entries and sub-entries let you access the different item configuration windows:



Entry	Sub-entry	Refer to...
Parameters	-	Controller Device Editor *(see page 77)*
Expert I/O	Power Distribution	Expert I/O configuration *(see page 83)*
	DM72F0	
	DM72F1	
TM5	TM5 Manager	TM5 Manager configuration *(see page 102)*
Embedded I/O	*	Embedded regular I/O modules configuration *(see page 106)*
Communication	Ethernet	Ethernet configuration *(see page 133)*
	CAN0	CANopen configuration *(see page 189)*
	Serial Line	Serial Line configuration *(see page 193)*
	PCI slots	PCI expansion module configuration *(see page 129)*
*The sub-entry depends on the controller selected.		

Device Tree

Many functions of the **Configuration** tab are also accessible from the **Program** tab. In the **Program** tab, the device tree describes the hardware configuration (for example, the following device tree is the default one when the controller is added):



Item	Description
PLC Logic	This part shows everything related to the application: ● Tasks including motion task ● Programming ● Library manager ● etc.
Expert IO	This representation shows the Embedded Expert I/O.
TM5	TM5 contains the embedded regular I/O modules and the expansion modules in the controller.
Ethernet Serial Line CAN0	These are the embedded communications.
PCI slots	Communication interfaces on the bus are presented as slots.

Content of Device Tree

The device tree represents the objects managed by a specific target (controller or HMI). These objects are:

- application objects (Tasks, etc.),
- programming objects (POU, GVL, etc.),
- hardware-related objects (Embedded functions, CAN, Expansion modules, etc.)

By default, the device tree includes the following hardware-related objects:

Reference	Expert IO	TM5 Manager	Embedded communications	PCI
TM258LD42DT	PowerDistribution DM72F0 DM72F1	DI12DE DO12TE	Ethernet Serial Line	-
TM258LD42DT4L		DI12DE DO12TE AI4LE	Ethernet Serial Line	2 slots
TM258LF42DT••		DI12DE DO12TE	Ethernet Serial Line CAN0 (CANopen)	-
TM258LF42DT4L••		DI12DE DO12TE AI4LE		2 slots
TM258LF66DT4L••		DI12DE DI12DE_1 DO12TE DO12TE_1 AI4LE		
TM258LF42DR••		DI6DE DI6DE_1 DO6RE D000E DO6RE_1		

Libraries

3

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. Refer to CoDeSys online help for more information on the **Library Manager**.

Modicon M258 Logic Controller

When you select a Modicon M258 Logic 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 all functions and function blocks which are required matching IEC61131-3 as standard POUs for an IEC programming system. The standard POUs must be tied to the project (standard.library).
Util	Analog Monitors, BCD Conversions, Bit/Byte Functions, Controller Datatypes, Function Manipulators, Mathematical Functions, Signals.
M258 PLCSystem *(see Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 Logic Controller PLCSystem Library Guide)*	Contains functions and variables to get information and send commands to the controller system.

Library name	Description
PLCCommunication *(see Communication Functions:, PLCCommunication Library)*	SysMem, Standard, SE_PLCSystem. 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.
M258 Relocation Table *(see page 34)*	The relocation table allows the user to organize data to optimize exchanges between the Modbus client and the controller, by regrouping non-contiguous data into a contiguous table of registers.

Supported Standard Data Types

4

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	-	-	16 bit

Memory Mapping

5

Introduction

This chapter describes the memory maps and sizes of the different memory areas in the Modicon M258 Logic Controller. These memory areas are used to store user program logic, data and the programming libraries.

What's in this Chapter?

This chapter contains the following topics:

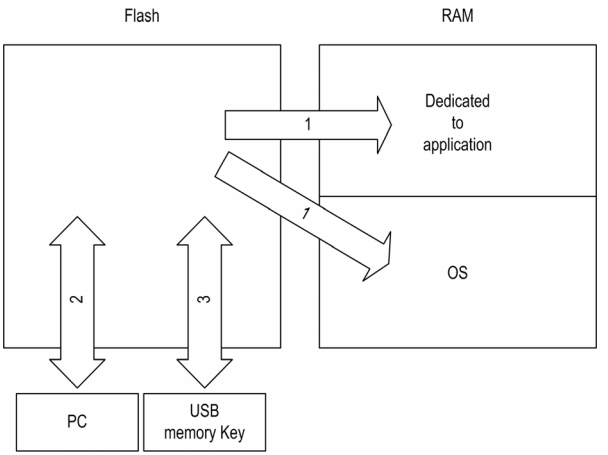
Controller Memory Organization

Introduction

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

- The Flash memory *(see page 31)* contains files (application, configuration files, etc.).
- The RAM (Random Access Memory) *(see page 29)* 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 147)* ● FTP server *(see page 159)* ● SoMachine *(see page 78)*
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 217)*

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 by the memory.

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 M258 Logic Controller.

Memory Mapping

The RAM size is 64 Mbytes.

The RAM is composed of 2 areas:
- dedicated application memory
- OS memory

This table describes the dedicated application memory:

Area	Element	Size (bytes)
System area 128 Kbytes	%MW0...MW59999	125000
	System and diagnostic variables *(see page 30)* (60000...60199)	
	Dynamic Memory Area: Read Relocation Table *(see page 34)* (60200...61999)	
	Reserved Memory Area (62000...62199)	
	Dynamic Memory Area: Write Relocation Table *(see page 34)* (62200...63999)	
	Reserved	3000
User area 10 Mbytes	Symbols	2000000
	Retains Data *(see page 31)*	10000
	Persistent Data *(see page 31)*	20000
	Variables	-
	Application	
	Libraries	

System and Diagnostic Variables

Variables
PLC_R
PLC_W
ETH_R]
ETH_W
SERIAL_R
SERIAL_W
TM5_MODULE_R

For more information on System and Diagnostic Variables, refer to *M258 PLCSystem Library Guide*.

Flash Memory Organization

Introduction

The Flash memory contains the files used by the controller.

File Type

The Modicon M258 Logic 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 in the case that the source file is not residing on the PC (2).
Post configuration	File that contains Ethernet and Serial parameters. The parameters specified in the file overide 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 served by the Webserver 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 not automatic. When you download an application from SoMachine to the controller, you are transferring only the binary Executable application directly to RAM.

There are two ways to create the Boot application:

- Select the option during the application download process.
- Do this in the **Online** menu at any point after download.

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

(2) SoMachine does not support uploading of the Executable application nor the Boot application to a PC for modification. Program modifications must be made to the application source. When you download your application, you have the option to store the source file to Flash memory.

File Organization

The following table shows the file organization of the flash memory:

Disk	Directory	File	Content	Access	Up/Downloaded data type
/sys	OS	M258FW1v_XX.YY (1)	Firmware of core 1	Read/Write	Firmware
		M258FW2v_XX.YY (1)	Firmware of core 2		
		M258_top_Vxx.bit	Firmware		
		Version.ini	Control file for firmware version		
	Cmd	Cmd.log	Result of last script executed by the USB memory Key *(see page 217)*	Read/Write	log file
		Script.cmd	Script executed by the USB memory Key		
	Web	Index.htm	HTML pages served by the Webserver for the website embedded in the controller.	Read/Write	Website
		Conf.htm			
		...			
/usr	App	Application.app	Boot application	Read/Write	Application
		Application.crc			
		Application.map			
		Archive.prj (2)	Application source		
	App/MFW	DeviceID_X.fw (2)	Expansion modules Firmware	Read/Write	Firmware
	Cfg	Machine.cfg	Post configuration file *(see page 38)*	Read/Write	Configuration
		CodesysLateConf.cfg	• Name of application to launch • Routing table (main/sub net)		Configuration
	Log	*UserDefinedLogName_1*.log	All log files created using the data logging functions *(see SoMachine, Data Logging Functions, DataLogging Library Guide)*. You must specified the total number of files created and the names and contents of each log file.	Read/Write	log file
		...			
		UserDefinedLogName_n.log			

Disk	Directory	File	Content	Access	Up/Downloaded data type
/usr	SysLog	CrashCx.txt	This file contains a record of detected system errors. For use by Schneider Electric Technical Support.	Read/Write	log file
		PLClog.txt	This file contains system event data that is also visible in SoMachine by viewing **Program →MyController →Log**.		
		FWLogBoot.txt	This file contains a log of firwmare Boot event. For use by Schneider Electric Technical Support.		
		FWLogCx.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.	Read/Write	Configuration and icon files

(1): v_XX.YY represents the version

Relocation Table

Introduction

The Relocation Table allows the user to organize data to optimize communication between the controller and other equipment by regrouping non-contiguous data into a contiguous table of registers.

NOTE: A Relocation Table is considered as an object. Only one Relocation Table object can be added to a controller.

Relocation Table Description

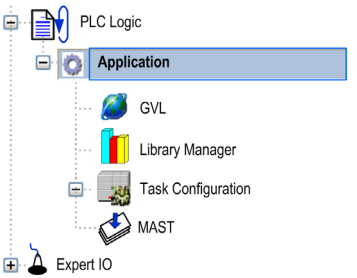
This table describes the Relocation Table organization:

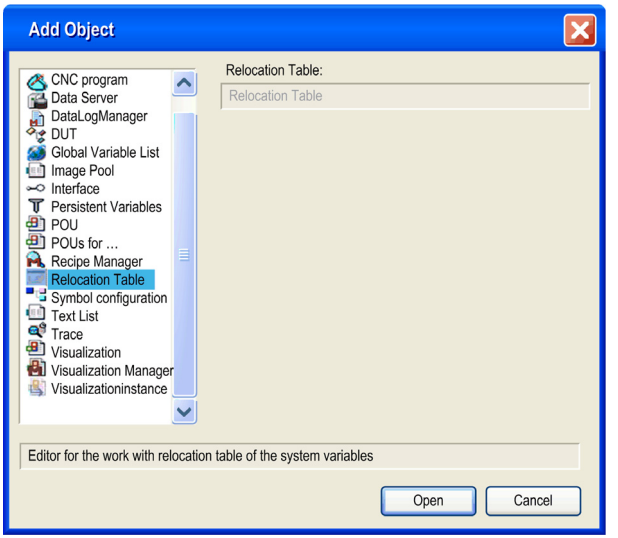
Register	Description
60200...61999	Dynamic Memory Area: Read Relocation Table
62200...63999	Dynamic Memory Area: Write Relocation Table

For further information refer to *M258 PLCSystem Library Guide*.

Adding a Relocation Table

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

Step	Action
1	Select the **Program** tab: 
2	In the Device tree of the **Devices** window, right click the **Application** node and click **Add Object...** in the contextual menu: 

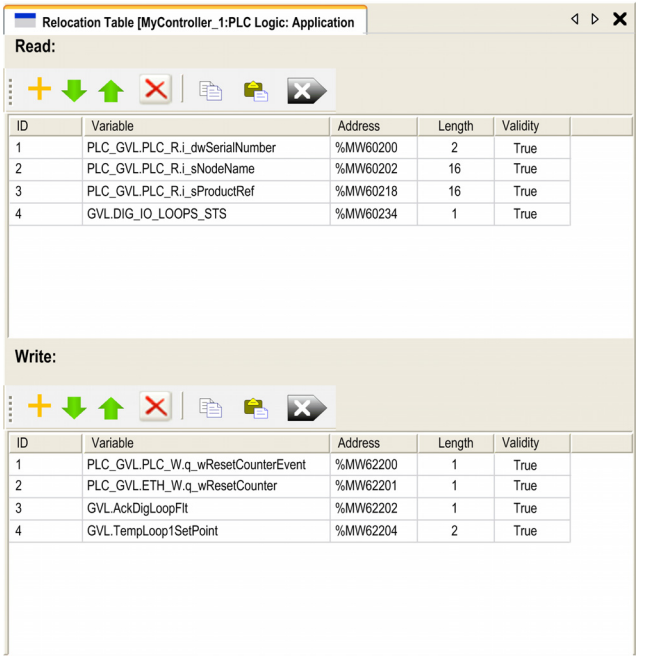
Step	Action
3	Select **Relocation Table** in the list and click the **Open** button:  **Result:** The new Relocation Table is created and initialized. **NOTE:** As a Relocation Table must be unique for a controller, its name is **Relocation Table** and cannot be changed.

Relocation Table Editor

The **Relocation Table Editor** allows to organize your variables under the **Relocation Table**.

To access to the **Relocation Table Editor**, double-click the **Relocation Table** node in the Device tree of the **Devices** window:

The following 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.

Icon	Element	Description
-	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 the entered variable is undefined, then the content of the cell is displayed in red, the related **Validity** cell is False, and **Address** is set to -1.

Post Configuration

Introduction

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

By default, all 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

Post Configuration file allows you to change some network parameters.

- Ethernet parameters:
  - IP Address
  - Subnet Mask
  - Gateway Address
  - Transfer Rate
  - IP Config Mode
  - Device Name
  - IPMaster Address *(see page 184)*
- Serial Line parameters, for each Serial Line in the application (embedded port or PCI module):
  - Baud rate
  - Parity
  - Data bits
  - Stop bit

Operating Mode

The Post Configuration file is read:

- at power up
- after an application download
- on reboot

Format

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	Select the **PROGRAM** tab.
2	In the menu bar, click **Build** →**Generate Post Configuration...** **Result:** an explorer window appears.
3	Select the destination folder of the Post Configuration file.
4	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, you must 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 220)* (with the proper script)
- download through the FTP server *(see page 159)*
- download with SoMachine controller device editor *(see page 78)*

Modifying a Post Configuration File

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

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

To directly modify the Post Configuration file in the controller, use the **Setup** menu of the Web server *(see page 147)*.

Post Configuration File Example

```
# TM258LD42DT / Ethernet / IPAddress
# Ethernet IP address
id[111].param[0] = [0, 0, 0, 0]

# TM258LD42DT / Ethernet / SubnetMask
# Ethernet IP mask
id[111].param[1] = [0, 0, 0, 0]

# TM258LD42DT / Ethernet / GatewayAddress
# Ethernet IP gateway address
id[111].param[2] = [0, 0, 0, 0]

# TM258LD42DT / Ethernet / TransferRate
# Transfer Rate: 0:Auto, 1:10 MBit full, 2:10 MBit half, 3:100
MBit full, 4:100 MBit half
id[111].param[3] = 0

# TM258LD42DT / Ethernet / IPConfigMode
# IP configuration mode: 0:FIXED 1:BOOTP 2:DHCP
id[111].param[4] = 0
```

```
# TM258LD42DT / Ethernet / DeviceName
# Name of the device on the Ethernet network
id[111].param[5] = 'my Device'

# TM258LD42DT / Serial Line / Serial Line Configuration /
Baudrate
# Serial Line Baud Rate in bit/s
id[40101].param[10000].Bauds = 38400

# TM258LD42DT / Serial Line / Serial Line Configuration /
Parity
# Serial Line Parity (0=None, 1=Odd, 2=Even)
id[40101].param[10000].Parity = 0

# TM258LD42DT / Serial Line / Serial Line Configuration / Data
bits
# Serial Line Data bits (7 or 8)
id[40101].param[10000].DataFormat = 8

# TM258LD42DT / Serial Line / Serial Line Configuration / Stop
bits
# Serial Line Stop bits (1 or 2)
id[40101].param[10000].StopBit = 1
```

Tasks

6

Introduction

The Task Configuration node in the SoMachine device tree allows you to define one or several tasks to control the execution of your application program.

The task types available are:
- Cyclic
- Freewheeling
- Event
- External Event

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

What's in this Chapter?

This chapter contains the following topics:

Maximum Number of Tasks

Maximum Number of Tasks

The maximum number of tasks you can define for the Modicon M258 Logic Controller are:

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

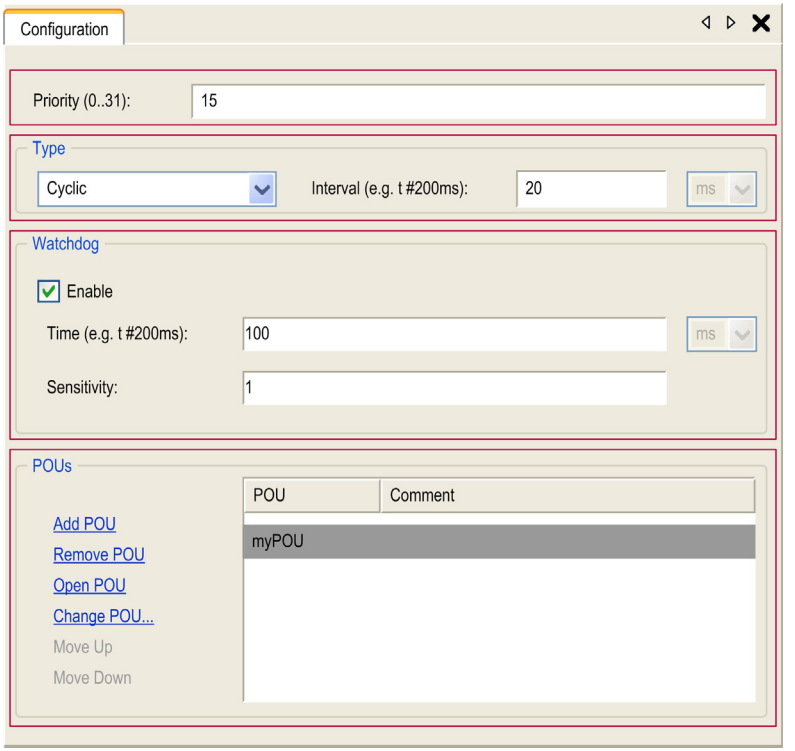
Task Configuration Screen

Screen Description

The following screen allows configuring the tasks. Double click on the task that you want to configure in the device tree of the **Devices** window to access this screen.

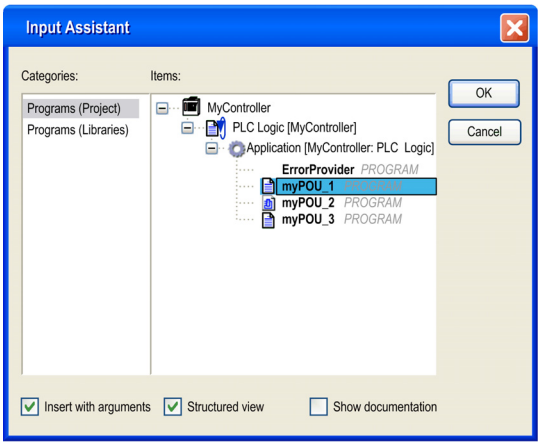
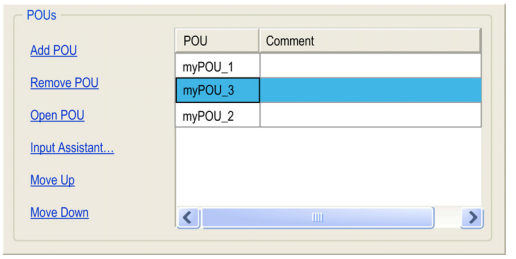
Each configuration task has its own parameters which are independent of the other tasks.

The **task configuration** window is composed of 4 parts:



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

Field Name	Definition
Priority	You can configure the priority of each task with a number between 0 and 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 preempt 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 preempt tasks with the same priority, the result could be indeterminate and unpredicable. For more information, refer to Task Priorities *(see page 52)*.
Type	4 types of task are available: ● Cyclic *(see page 48)* ● Freewheeling *(see page 49)* ● Event *(see page 49)* ● External event *(see page 50)*
Watchdog *(see page 51)*	To configure the watchdog, you must define two 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 into a HALT state *(see page 56)*.

Field Name	Definition
POUs	The list of **POU**s (Programming Organization Unit) controlled by the task is defined in the task configuration window. To add a POU linked to the task, use the command **Add POU**. To remove a POU from the list, use the command **Remove POU**. You can create as many POUs as you want. An application with several small POUs, as opposed to one large POU, improves the refresh time of the variables in online mode. The command **Open POU** opens the currently selected POU in the appropriate editor. To access an item already stated in the system, use the **Change POU...**:  **POU**s are executed in the order shown in the list below. To rearrange the **POU**s in the list, click on **Move Up** or **Move Down**: 

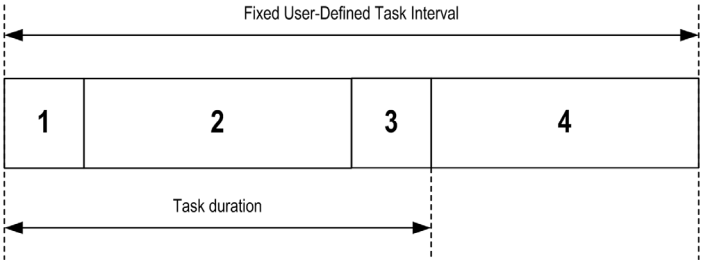
Task Types

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 duration using the Interval setting in the Type section of Configuration sub-tab for that task. Each Cyclic task type executes as follows:

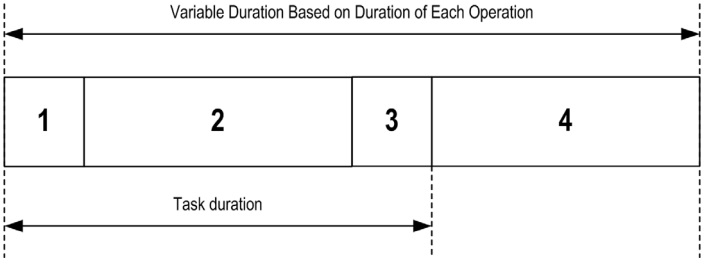


1. **Read Inputs:** The input states are written to the %I input memory variable and other system operations are executed.
2. **Task Processing:** The user code (POU, etc.) defined in the task is processed. The %Q output memory variable is updated according to your application program instructions but not written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variable is 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 Modicon M258 Logic Controller Settings *(see page 80)* and CoDeSys online help.
   For more information on I/O behavior, refer to Controller States Detailed Description *(see page 60)*.
4. **Remaining Interval time:** The controller OS carries out system processing and any other lower priority tasks.

NOTE: If you define too short a period for the cycle task, it will repeat immediately after the write of the outputs and without executing other lower priority task 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.

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 input states are written to the %I input memory variable and other system operations are executed.
2. **Task Processing:** The user code (POU, etc.) defined in the task is processed. The %Q output memory variable is updated according to your application program instructions but not written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variable is 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 Modicon M258 Logic Controller Settings *(see page 80)* and CoDeSys online help.
   For more information on I/O behavior, refer to Controller States Detailed Description *(see page 60)*.
4. **System Processing:** The controller OS carries out system processing and any other lower priority tasks. The length of the system processing period is set to 30 % of the total duration of the 3 previous operations (4 = 30 % x (1 + 2 + 3)). In any case, the system processing period won't be lower than 3 ms.

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 preempted 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, select the **Event type** on the **Configuration** sub-tab and click on the **Input Assistant** button ... to the right of the **Event name** field. This will cause the **Input Assistant dialog** box to appear. In the **Input Assistant dialog** box, you navigate the tree to find and assign the `my_Var` variable.

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 preempted 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 dropdown list on the **Configuration** sub-tab.

Depending of the related product, there are up to 4 types of events that can be associated with an External Event task:

- Rising edge on Fast input (%IX1.0 ... %IX1.3 and %IX3.0 ... %IX3.3 inputs)
- HSC thresholds
- HSC Stop
- CAN Sync

NOTE: CAN Sync is a specific event object, dependant 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.

System and Task Watchdogs

Introduction

Two types of watchdog functionality are implemented for the Modicon M258 Logic Controller. These are:

- **System Watchdogs**: These watchdogs are defined in and managed by the controller OS (firmware). These are not configurable by the user.
- **Task Watchdogs**: Optional watchdogs that can be defined for each task. These are managed by your application program and are configurable in SoMachine.

System Watchdogs

Two system watchdogs are defined for the Modicon M258 Logic Controller. They are managed by the controller OS (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, a system error is detected.

The threshold conditions for the 2 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 EMPTY 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.

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 and application error.

A task watchdog is configured on the Configuration sub-tab of the Task Configuration tab for the individual task. To access this tab, double-click on the task in the device tree.

NOTE: For more details on watchdogs, refer to the CoDeSys online help.

Task Priorities

Introduction

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.

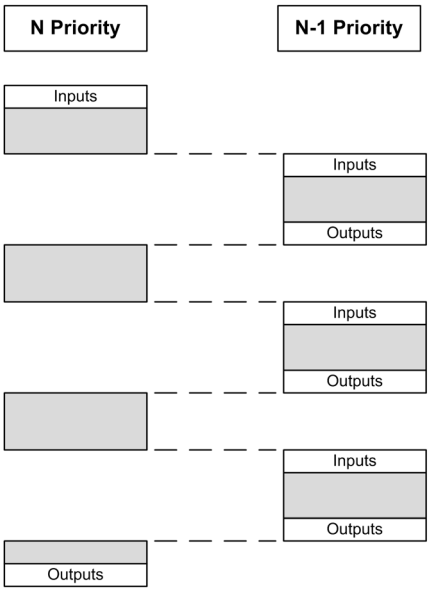
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Task Priority Recommendations

- Priority 0 to 24: Controller tasks. Assign these priorities to tasks with a high real-time requirement.
- Priority 25 to 31: Background tasks. Assign these priorities to tasks with a low real-time 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, an error is detected 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

For the Modicon M258 Logic Controller:

- The MAST task that 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 52)* for more information on priority settings. Refer to System and Task Watchdogs *(see page 51)* 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 52)* 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 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 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.

Controller States and Behaviors

7

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's in this Chapter?

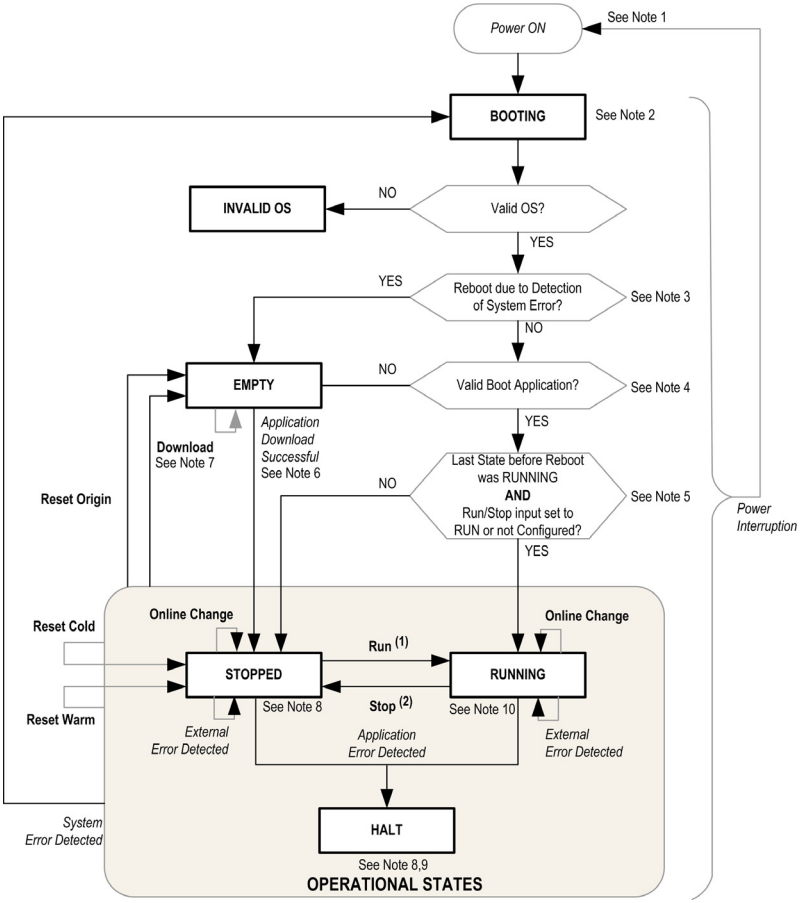
This chapter contains the following sections:

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 67)*.

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

Note 1: The Power Cycle (Power Interruption followed by a Power ON) deletes all output forcing settings. Refer to Controller State and Output Behavior *(see page 65)* 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 38)* file settings (if any) are applied.

Note 5: 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.

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 38)* 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.

- **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.

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 60)* 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 application or cycle power.

Note 10: The RUNNING state has two exceptional conditions.

They are:

- RUNNING with External Error: This exceptional condition is indicated by the MS Status LED, which will appear solid green with 1 red flash. You may exit this state by clearing the external error. No controller commands are required.
- RUNNING with Breakpoint: This exceptional condition is indicated by the MS Status LED, which will show 3 green flashes. Refer to Controller States Description *(see page 60)* for further details.

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, checking for 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) **Note:** The controller states can be read in the PLC_R.i_wStatus system variable of the M258 PLCSystem library *(see Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 Logic Controller 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 checks the checksum of the firmware and user applications. It does not execute the application nor does it communicate.	Flashing green / red
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 flashing red
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 M258 Motion Controller Firmware *(see page 218)*.	Flashing red
EMPTY	There is no or an invalid application. PCI expansion modules are inactive.	Single flash green
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 flashing red
RUNNING	The controller is executing a valid application.	Solid 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. See online CoDeSys help in SoMachine for details on breakpoints management.	3 flash green
RUNNING with detection of an *External Error*	This state is the same as the normal RUNNING state except the LED indications are different.	Solid green / single flash red
STOPPED	The controller has a valid application that is stopped. See Details of the STOPPED State *(see page 62)* for an explanation of the behavior of outputs and field buses in this state.	Flashing green

Controller State	Description	RUN/MS LED
STOPPED with detection of an *External Error*	This state is the same as the normal STOPPED state except the LED indications are different.	Flashing green / single flash red
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: • The task responsible for the Application Error always behaves as if the **Update IO while in stop** option was not selected. All other tasks follow the actual setting. • The LED indications are different	Single flash red

Details of the STOPPED State

The following statements are always 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, etc.), 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 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 variable.
- The Task Processing operation is not executed.
- The Write Outputs operation continues. The %Q output memory variable is 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 variable is frozen at its 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.

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's in this Section?

This section contains the following topics:

Controller States and Output Behavior

Introduction

The Modicon M258 Logic Controller defines output behavior in response to commands and system events in a way that allows for greater flexibility. An understanding of this behavior is necessary before discussing the commands and events that affect controller states. For example, typical controllers define only two options for output behavior in stop: fallback to default value or keep current value.

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

- Managed by Application Program
- Keep Current Values
- Set All Outputs to Default
- Initialization Values
- Output Forcing

Managed by Application Program

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

Keep Current Values

You can select this option by choosing **Keep current values** in the **Behaviour for outputs in Stop** dropdown menu of the **PLC Settings** sub-tab 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 and HALT controller states. Outputs are set to and maintained in their current state, although the details of the output behavior varies greatly depending on the setting of the **Update IO while in stop** option and the actions commanded via configured fieldbuses. Refer to Controller States Description *(see page 60)* for more details on these variations.

Set All Outputs to Default

You can select this option by choosing **Set all outputs to default** in the **Behaviour for outputs in Stop** dropdown menu of the **PLC Settings** sub-tab 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 and HALT controller states. Outputs are set to their user-defined default values, although the details of the output behavior varies greatly depending on the setting of the **Update IO while in stop** option and the actions commanded via configured fieldbuses. Refer to Controller States Description *(see page 60)* for more details on these variations.

Initialization Values

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

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

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

Output Forcing

The controller allows you to force the state of selected outputs to a defined value for the purposes of system testing and commissioning. Output forcing overrides all other commands to an output irrespective of task programming. You are only able to force the value of an output while your controller is connected to SoMachine. To do so you use the Force Values command in the Debug/Watch menu. 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.

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. The Run/Stop input must be 1 for all of the subsequent options to be effective. Refer to Run/Stop Input *(see page 85)* for more information.
- SoMachine Online Menu: Select the **Start** command.
- By an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M258 PLCSystem library *(see Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 Logic Controller 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 56)* 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 Run Command:

- Run/Stop Input: If configured, command a value of 0 to the Run/Stop input. Refer to Run/Stop Input *(see page 85)* for more information.
- SoMachine Online Menu: Select the **Stop** command.
- By an internal call by the application or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M258 PLCSystem library *(see Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 Logic Controller 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 222)* and Reboot *(see page 70)* for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram *(see page 56)* 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 or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M258 PLCSystem library *(see Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 Logic Controller PLCSystem Library Guide)*.

Effects of the Reset Warm Command:
1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for detected 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 briefly reset to their initialization values and then to their user-configured default values.

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

Reset Cold

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

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Cold Command:

- SoMachine Online Menu: Select the **Reset cold** command.
- By an internal call by the application or an external call via Modbus request using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the M258 PLCSystem library *(see Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 Logic Controller PLCSystem Library Guide)*.

Effects of the Reset Cold Command:

1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for detected 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 briefly reset to their initialization values and then to their user-configured default values.

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

Reset Origin

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

Starting Conditions: RUNNING, STOPPED, or HALT states.

Methods for Issuing a Reset Origin Command:

- 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 detected 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 75)*.

Reboot

Effect: Commands a reboot of the controller.

Starting Conditions: Any state.

Methods for Issuing the Reboot Command:
- Power cycle.
- 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 222)* for further details.

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
      - If configured, the Run/Stop input is set to RUN, and
      - Controller state was RUNNING prior to the power cycle.
   b. The controller state will be STOPPED if:
      - The Reboot was provoked by a Reboot by script, or
      - The boot application is different than the application loaded prior to the reboot, or
      - If configured, the Run/Stop input is set to STOP, or
      - Controller state was STOPPED prior to a power cycle, or
      - The previously saved context is invalid.
   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 a detected System Error.
   d. The controller state will be INVALID_OS if there is no valid OS.
2. Forcing is maintained if the boot application is loaded successfully. If not, forcing is erased.
3. Diagnostic indications for detected 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.

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

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

NOTE: If you provide power to the Run/Stop input from the same source as the controller, the loss of power to this input 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.

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:
  Two 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 56)*.
- 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 220)* 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 detected 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.

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

Effects of the FTP or USB key Download Command:

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

Error Detection, Types, and Management

Detected Error Management

The controller manages 3 types of detected errors:
- external detected errors
- application detected errors
- system detected errors

The following table describes the types of errors that may be detected

Type of Error Detected	Description	Resulting Controller State
External Error Detected	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 whether or not it reports an error, 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 loss of communication with a device • The controller is configured for a module that is not present or not detected • The boot application in Fash memory is not the same as the one in RAM. Examples: • output short circuit • missing expansion module • communication lost • etc.	RUNNING with External Error Detected Or STOPPED with External Error Detected
Application Error Detected	An application error is detected when improper programming is encountered or when a task watchdog threshold is exceeded. Examples: • task (software) watchdog exception • execution of an unknown function • etc.	HALT

Type of Error Detected	Description	Resulting Controller State
System Error Detected	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. Examples: ● System (hardware) watchdog overflow ● exceeding the defined size of an array ● etc.	BOOTING → EMPTY

NOTE: refer to the M258 PLCSystem library Guide *(see Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 Logic Controller PLCSystem Library Guide)* for more detailed information on diagnostics.

Remanent Variables

Remanent Variables

Remanent variables can 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.

The following table describes the behavior of remanent variables in each case:

Action	**VAR**	**VAR RETAIN**	**VAR PERSISTENT and RETAIN-PERSISTENT**
Online change to application program	X	X	X
Stop	X	X	X
Power cycle	-	X	X
Reset warm	-	X	X
Reset cold	-	-	X
Reset origin	-	-	-
Download of application program	-	-	X
X The value is maintained **-** The value is re initialized			

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 others %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).

Controller Device Editor

8

Introduction

This chapter describes how to configure the controller.

What's in this Chapter?

This chapter contains the following topics:

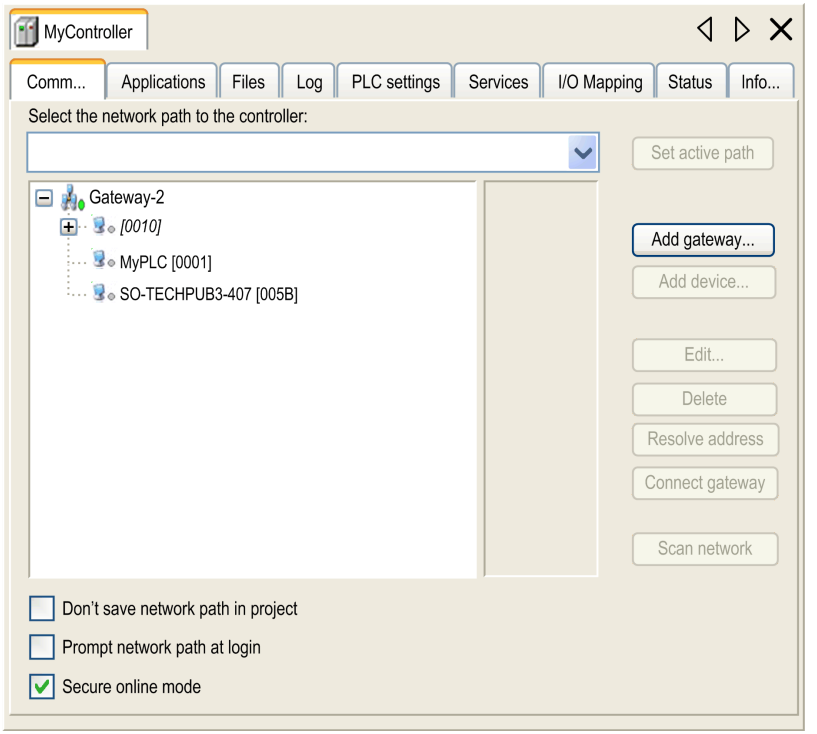
Controller Device Editor

Introduction

The Controller Device Editor lets you configure the controller.

Controller Device Editor

To open the controller device editor, select the **Configuration** tab and double-click on the controller:



Tab Descriptions

The following table contains the Controller Device Editor tabs description:

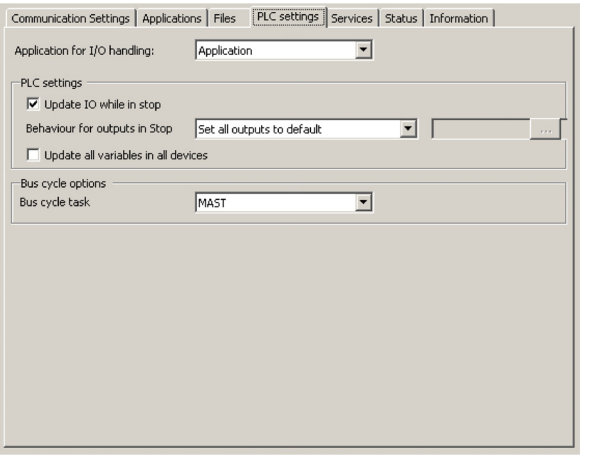
Tab	Description	Restriction
Communication Settings	Allows configuring the connection between SoMachine and the controller.	-
Applications	Shows the application currently running on the controller and allows removing the application from the controller.	Online mode only
Files	File management between the 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 81)*	Lets you configure the on-line 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.	-
Status	No information delivered.	-
Information	Displays general information about the device (name, description, provider, version, image).	-

For more details, see SoMachine Online Help CoDeSys part.

Modicon M258 Logic Controller Controller Settings

Overview

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



The following table describes the elements of 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.

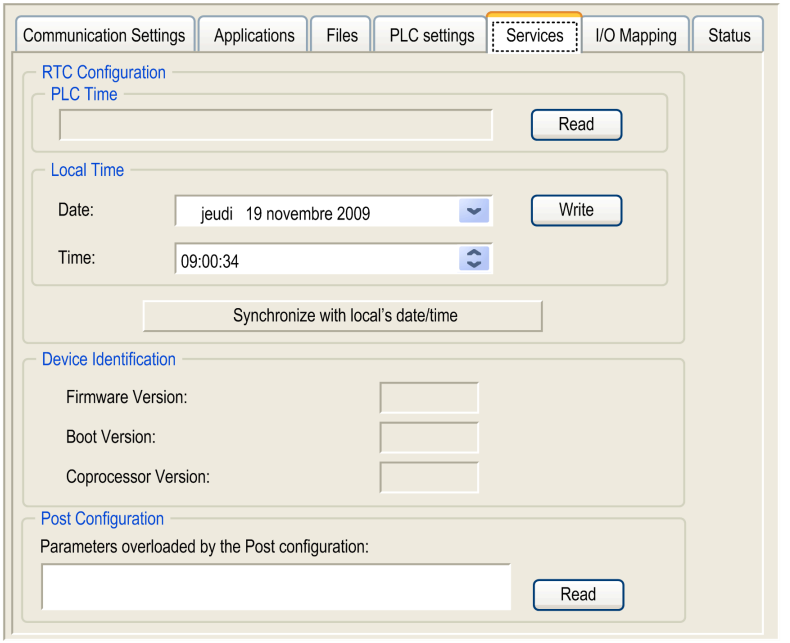
Modicon M258 Logic Controller Controller Services

Services Tab

The Services tab is divided in three parts:

- RTC Configuration
- Device Identification
- Post Configuration

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



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

Element		Description
RTC Configuration	PLC time	Displays the date/time read from the controller. This read-only field is initially empty. To read and display the date/time saved on the controller, click on the **Read** button.
	Local time	Lets you define a date and a time which are sent to the controller by a click on the **Write** button. A message box informs the user on the success of the command. Local time fields are initialized with the current PC settings.
	Synchronize with local date/time	Lets you send directly the current PC settings. A message box informs the user of the success of the command.
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.

Embedded Expert I/O

9

Introduction

This chapter describes how to configure Modicon M258 Logic Controller Embedded Expert I/O.

The controller base provides:

- 2 embedded expert I/O modules (DM72F0 and DM72F1) with:
  - 5 fast inputs
  - 2 regular inputs
  - 2 fast outputs
- 1 Controller Power Distribution Module (CPDM)

Each embedded expert I/O module (DM72F•) can support expert functions *(see page 92)*.

What's in this Chapter?

This chapter contains the following sections:

9.1 Overview

Expert I/O Overview

Introduction

The controller base provides:

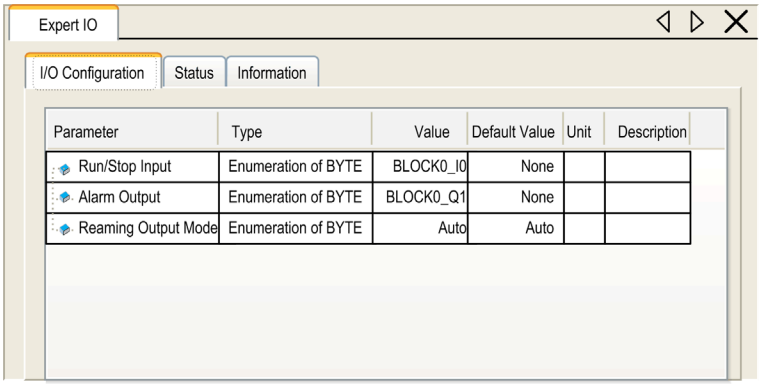
- 2 embedded expert I/O modules (DM72F0 and DM72F1) with:
  - 5 fast inputs
  - 2 regular inputs
  - 2 fast outputs
- 1 Controller Power Distribution Module (CPDM)

Each embedded expert I/O module (DM72F•) can support expert functions *(see page 92)*.

Embedded Expert I/O Configuration

To configure the Expert I/O, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Expert I/O** entry on the left hand side.



Parameter	Function
Run/Stop Input	Define one input to be used as Run/Stop input *(see page 85)*.
Alarm Output	Define one output to be used as alarm output *(see page 86)*.
Rearming Output Mode	Define the rearming output mode *(see page 86)*.

Run/Stop Input

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

NOTE: Run/Stop input is managed even if the option **Update I/O while in stop** is not selected in Controller Device Editor (PLC setting tab) *(see page 78)*.

Inputs of expert function cannot be configured as Run/Stop.

The controller will automatically restart into the RUNNING state if the Run/Stop Input is configured and set to 1 or not configured and one or more of the following statements is true:

- The controller state before the reboot or power cycle was RUNNING.
- The restart was initiated by an online change to the application program.
- You performed a **Multiple download** and the **Start all application after download or online change** option was selected.

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

⚠ WARNING

UNINTENDED MACHINE START UP

- Before using the Automatic Start in Run setting, confirm that the automatic reactivation of the outputs does not produce unintended consequences.
- Use the Run/Stop input to help avoid an unwanted restart in Run mode.
- Use the Run/Stop input to help prevent the unintentional start up from a remote location.
- Be sure of the state of security of your machine or process environment before applying power to the Run/Stop input.
- Be sure of the state of security or your machine or process environment before issuing a Run command 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.

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

NOTE: When a task is stopped at a breakpoint, the alarm output signals that the controller has stopped executing the application (Alarm output value is 0).

Rearming Output Mode

Fast outputs of DM72F• modules are push/pull technology. In case of detected error (short-circuit or over temperature), the output is put in tri-state and the condition is signaled by status bit (DM72F• channel IB1.0) and PLC_R.i_wLocalIOStatus *(see Modicon LMC058 Motion Controller, System Functions and Variables, Modicon LMC058 Motion Controller 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 enter into thermal protection mode (all outputs 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 STARTUP

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.

9.2 DM72F0 and DM72F1

What's in this Section?

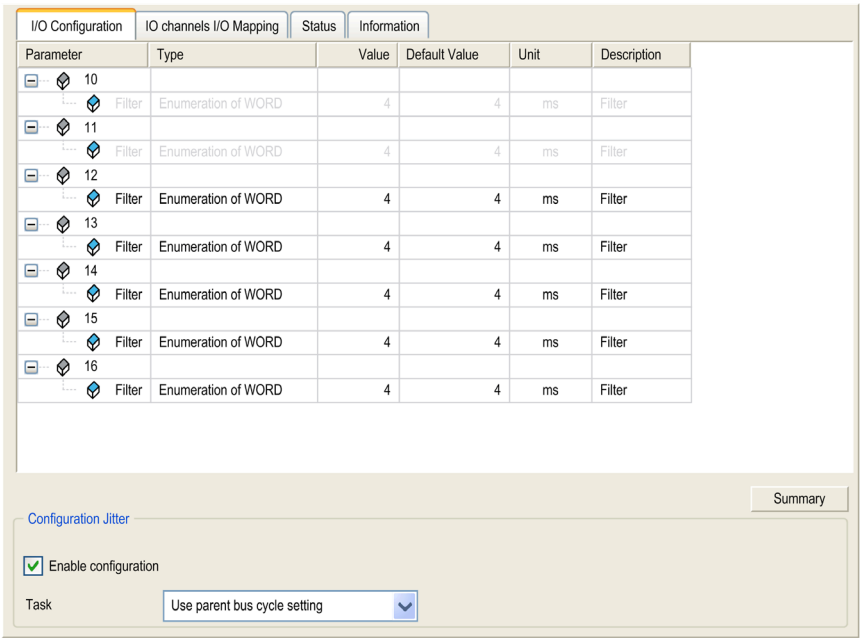
This section contains the following topics:

DM72F• Configuration

DM72F• I/O Configuration

The DM72F• Editor allows to configure the I/Os when they are not used by an expert function.

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Expert I/O** →**DM72F•** entry on the left hand side.
3	Select the **I/O Configuration** tab.



User can configure the following parameters:

Parameter	Value	Unit	Description	Constraint
Filter	No 1.5 4(default) 12	ms	Filtering value reduces the effect of noise on a controller input.	Enable if input is not used by expert function.
Enable Minimized jitter	Yes: Enabled No: Disabled (default)		Minimizes jitter on outputs by delaying the write to the physical ouputs until the read inputs operation of the next **Bus cycle task** commences. (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 M258 Logic Controller, Hardware Guide)*.

When inputs are used by an expert function (Event_Latch, HSC, PWM,…) the corresponding lines are disable 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 DM72F• level is ignored. Output management depends on expert function configuration.

I/O Management

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

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

If **Enable Minimized jitter** 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 **Enable Minimized jitter** is enabled, the physical output is updated from the %Q memory variable value at the begining of the subsequent **Bus cycle task**.

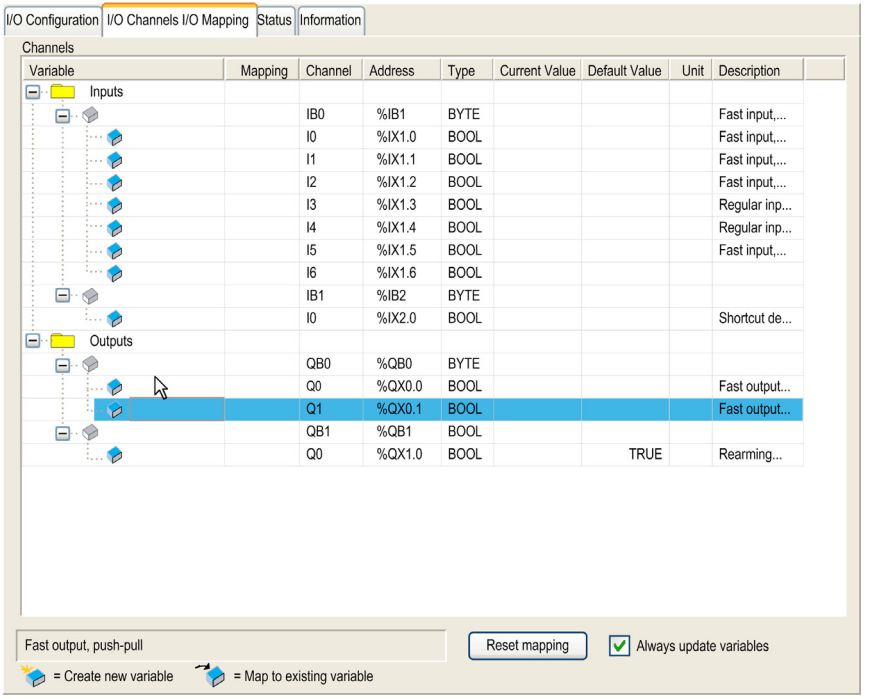
NOTE: The interest is to synchronize the effective activation of output with command or motion control on network.

For further details about **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	Select the **Configuration** tab and double-click the controller.
2	Click the **Expert I/O** →**DM72F•** entry on the left hand side.
3	Select the **IO Channels I/O Mapping** tab.



The table below describes the DM72F• modules I/O Mapping configuration:

Channel			Type	Default value	Description
Inputs	IB0		BYTE	-	State of all inputs (bit 8 = 0, not used)
		I0	BOOL	-	State of input 0
		...			...
		I6			State of input 6
	IB1		BYTE	-	Status byte of all outputs (bits 2-8 = 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 3-8 = 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 2-8 = 0, not used)
		Q0	BOOL	- TRUE FALSE	Rearming output command *(see page 84)* bit

User can associate variables with listed inputs and outputs.

For more details, refer to Online Help Codesys part.

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:
  - HSC Simple
  - Event_Latch I/O
- complex functions:
  - HSC Main
  - Encoder
  - PWM Generator
  - Frequency generator

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

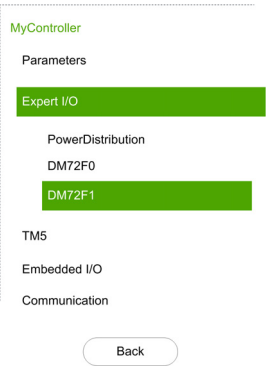
NOTE:

- 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 84)*.

Adding an Expert Function

To add an Expert function, proceed as follow:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Expert I/O** entry on the left hand side.
3	Click the **DM72F0** or **DM72F1** sub-entry on the left hand side. 

Step	Action
4	Click the **Add Device** button.
5	In the **Add Device** dialog box, select the expert function and click the **Add and Close** button.

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 97)*
HSC	The HSC functions can execute fast counts of pulses from sensors, encoders, switches, etc. that are connected to dedicated fast inputs.	M258 HSC Library *(see Modicon M258 Logic Controller, High Speed Counting, M258 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%).	M258 PWM library *(see Modicon M258 Logic Controller, Pulse Width Modulation, M258 Expert I/O Library Guide)*.
Encoder	The goal of this function is to connect an encoder to acquire a position. This function can be installed on an Embedded Expert I/O interface and supports only an incremental encoder. You can configure a linear or rotary axis.	M258 HSC Library *(see Modicon M258 Logic Controller, High Speed Counting, M258 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**	**Encoder**	**PWM**	**Frequency Generator**
DM72F0	Up to 4	1	1	1	1
DM72F1	Up to 4	1	1	1	1

For more details, refer to Expert I/O Mapping *(see page 95)*.

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 cannot be configured in an expert function if it has already been configured as a Run/Stop.
- An Output cannot be configured in an expert function if it has already been configured as a 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 function are available as fast or regular I/O.

When inputs are used in expert function (Latch, HSC,…), integrator filter is replaced by anti-bounce filter *(see M258 Logic Controller, Hardware Guide)*. Filter value will be configured in expert function screen.

Embedded Expert I/O mapping

I/O mapping for Expert Function on DM72F•

Embedded Expert IO mapping by expert function (M = Mandatory, C = depend on Configuration):

		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

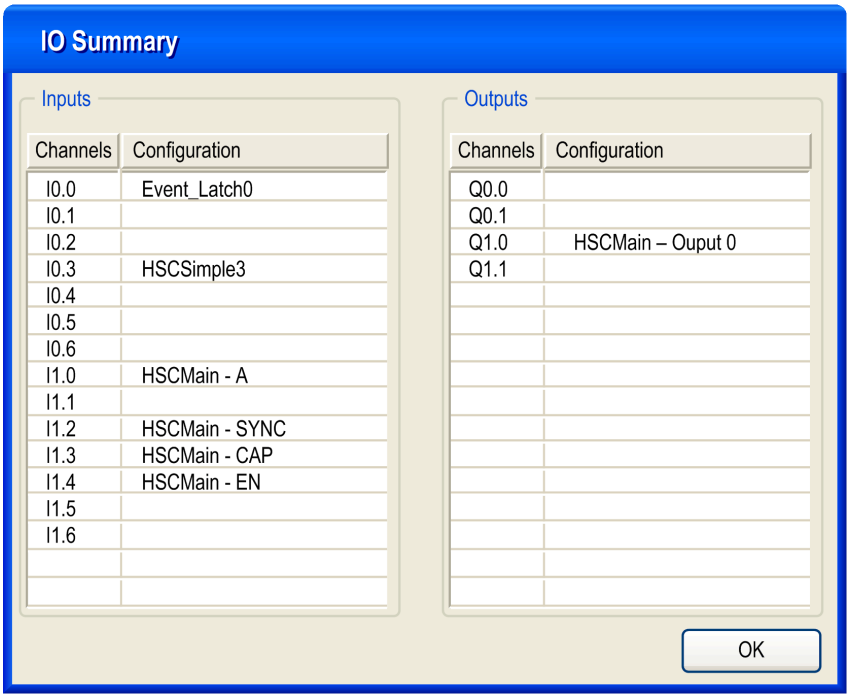
IO Summary

The IO summary window display the DM72F• IO mapping. You can see the I/O used by expert function.

The IO summary window is accessible from Expert I/O or DM72F• entries:

Step	Action
1	Select the **Configuration** tab and double-click on your controller.
2	Click the **Expert I/O** entry in the left hand side. **or** Click the **Expert I/O** →**DM72F•** entry in the left hand side.
3	Click the **Summary** button.

Example of IO summary:



Event_Latch Function

Introduction

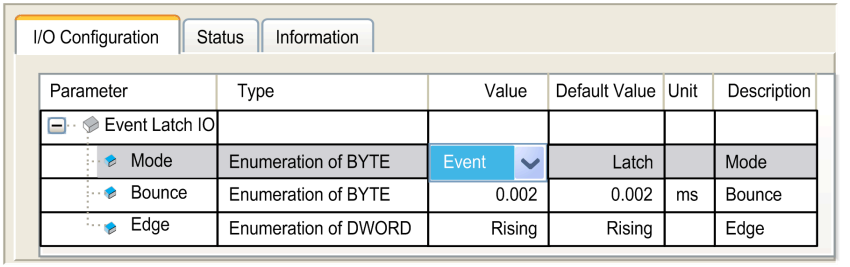
With the Event_Latch function, the Embedded Expert inputs can be configured as event or latch.

Adding a Event_Latch Function

To add an Event_Latch function, proceed as explained in Add an Expert Function *(see page 92)*.

Event_Latch Function Configuration

To configure the Event_Latch function, click on 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 Task reads the input.	
	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 more priority task is running).

Parameter	Value	Description	Constraint
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.	
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 84)*.

Standard Encoder

Introduction

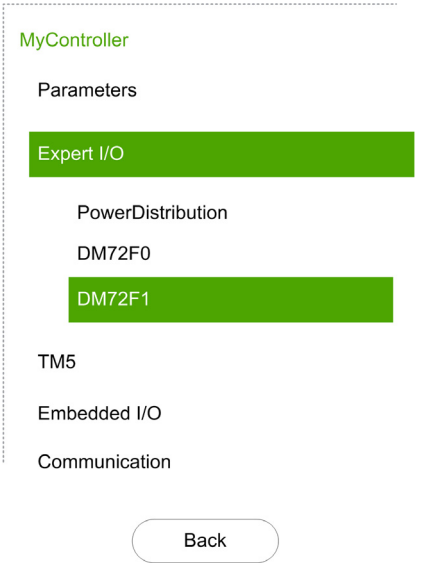
The goal of this function is to connect an encoder to acquire a position. So this function can be used as Master Axis for motion drives on CAN.

This function can be installed on an Embedded Expert I/O interface and supports only an incremental encoder. You can configure a linear or rotary axis.

For further information about Standard Encoder, see M258 HSC Library *(see Modicon M258 Logic Controller, High Speed Counting, M258 Expert I/O Library Guide)*.

Adding an Encoder

To add an Encoder, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Expert I/O** entry in the left hand side.
3	Click the **DM72F0** or **DM72F1** sub-entry. 
4	Click the **Add Device** button
5	In the **Add Device** dialog box, select the Encoder and click the **Add and Close** button.

9.3 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 channels I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

The table below describes the Controller Power Distribution Module I/O Mapping configuration:

Channel			**Type**	**Default value**	**Description**
Inputs	IB0		BYTE	-	State of all inputs (bits 4-8 = 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 are present, the memory variable = 00h.

TM5 Modules

10

Introduction

The TM5 Bus contains:
- Embedded I/O modules
- TM5 expansion modules

This chapter describes how to configure the TM5 Bus.

What's in this Chapter?

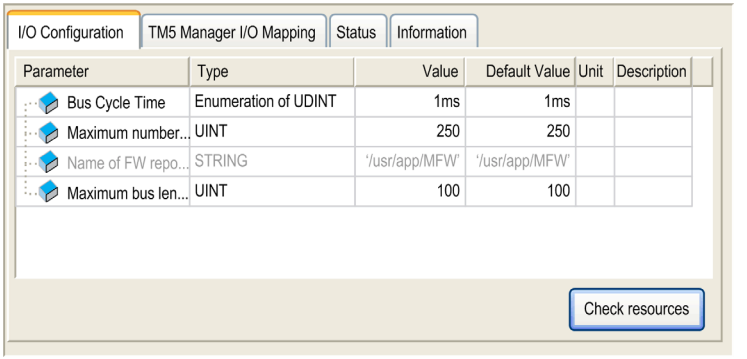
This chapter contains the following sections:

10.1 TM5 Manager Configuration

TM5 Manager Configuration

TM5 Manager Configuration

To configure the TM5 Manager, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click on the controller.
2	Click the **TM5** →**TM5_Manager** entry on the left-hand side. **Result**: The TM5 Manager configuration window is displayed: 

Parameters of the TM5 Manager:

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	TM5 Bus Cycle Time
Maximum number of physical slots	No of Embedded modules...250	250	-	Maximum number of modules on the TM5 bus.
Name of FW repository	Not configurable	-	-	This parameter indicates the Flash memory repository where are the modules firmware.
Maximum bus length in meters	1...2500	100	m	Total cable length used on the TM5 bus (cable between Transmitter/Receiver modules)

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 TM5 Bus Cycle Time follows 2 rules:

- The TM5 Bus Cycle Time must be longer than the **Minimum Cycle Time** of **EACH** expansion modules.
- The TM5 Bus Cycle Time must be long enough to permit the data exchange with all the modules. The calculation of this minimum bus cycle time is made by the function Check Resources *(see page 104)*.

Minimum Cycle Time of a module

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

For further information, refer to the Modicon TM5 Expansion Modules Configuration Programming Guide *(see Modicon TM5, Expansion Modules Configuration, Programming Guide)*.

Minimum I/O Update Time of a module

The Minimum I/O Update Time of a module is the time needed by the module 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 used %I memory variable for the inputs is updated from physical information.

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

On the next TM5 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.

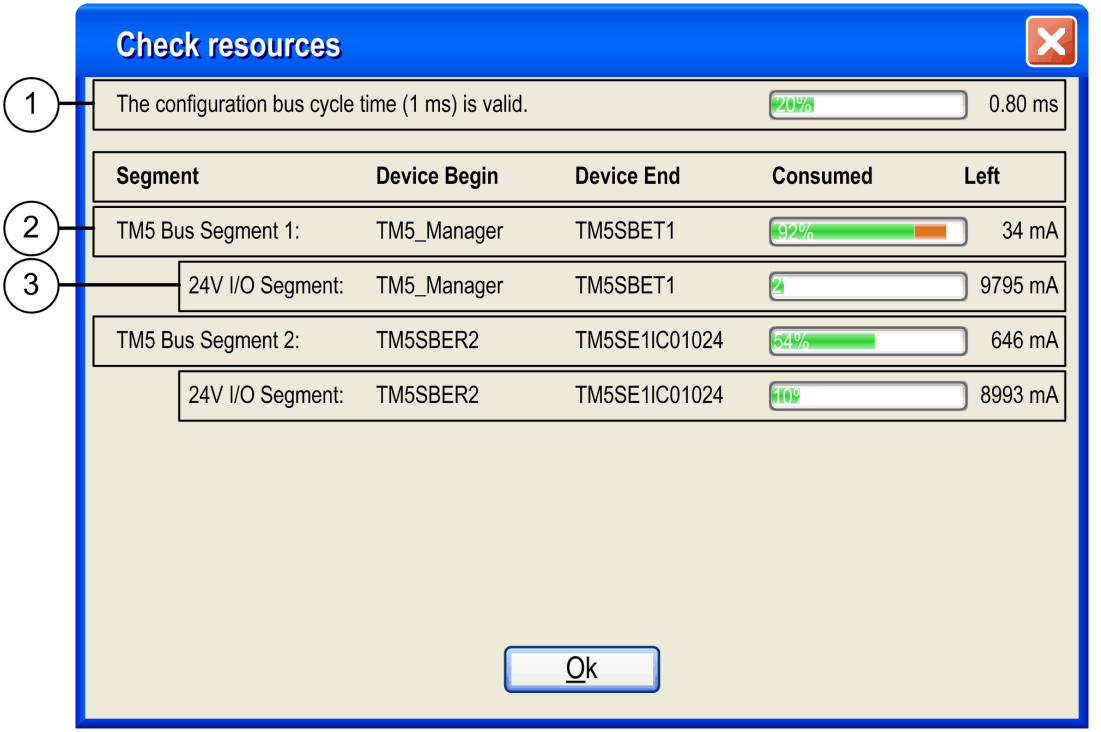
For more details on **Bus cycle task**, refer to Controller PLC Settings *(see page 80)*.

Check Resources

You can check if the Bus Cycle Time is valid and the power supply of the expansion modules.

To check resources of the expansion modules, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Embedded I/O** entry on the left-hand side.
3	Click the **Check Resources** button.



A segment is a group of I/O modules that are supplied by the same Power Distribution Module.

To check resources of the expansion modules, proceed as follows:

Item	Description
1	Indicates if the configured TM5 bus cycle time is valid.
2	Indicates, in this segment, the calculated current consumption of the modules. Indicates also if the PDM will be able to supply the modules.
3	Indicates, in this segment, the calculated current consumption of the 24Vdc I/O. Indicates also if the PDM will be able to supply the I/O.

NOTE: The current consumption figures presented by the Check Resources function are based on assumed values, and not on actual current measurements. The assumed values for the outputs are based on classical loads but can be adjusted using the 24 Vdc I/O segment external current setting in the I/O Configuration 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 Check Resources function to test the power budget is required, it is no substitute for actual and complete system testing and commissioning, refer to the System Planning and Installation Guide *(see Modicon Flexible TM5 System, System Planning and Installation Guide)*.

10.2 Embedded Regular I/O Modules Configuration

Introduction

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

What's in this Section?

This section contains the following topics:

Embedded Regular I/O Configuration

Introduction

The following table shows the Embedded Regular I/O modules and their associated controller reference:

Controller	Embedded Regular I/O	Description
TM258LD42DT TM258LF42DT••	DI12DE	12 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
	DO12TE	12 Outputs 24 VDC / 0.5 A
TM258LD42DT4L TM258LF42DT4L••	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
TM258LF66DT4L••	DI12DE	12 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
	DI12DE_1	12 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
	DO12TE	12 Outputs 24 VDC / 0.5 A
	DO12TE_1	12 Outputs 24 VDC / 0.5 A
	AI4LE	4 Inputs ±10 V / 0... 20 mA
TM258LF42DR••	DI6DE	6 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
	DI6DE_1	6 Digital Inputs 24 VDC, Sink, IEC 61131-2, Type 1
	DO6RE	6 Outputs, 30 VDC / 250 VAC / 5 A, relay C/O
	D000E	Dummy module
	DO6RE_1	6 Outputs, 30 VDC / 250 VAC / 5 A, relay C/O

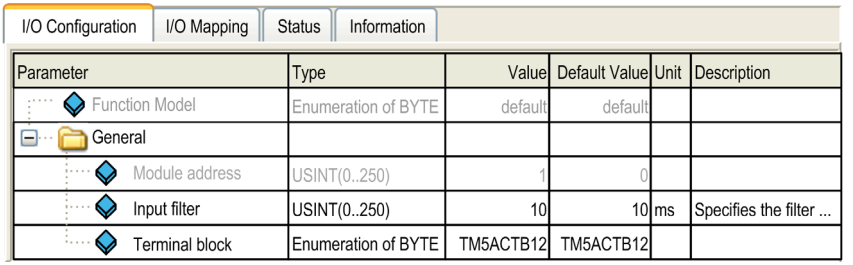
Embedded Regular I/O Configuration

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

Step	Action
1	Select the **Configuration** tab and double-click on the controller.
2	Click the **Embedded I/O** entry on the left-hand side.
3	Click the module you want to configure. **Result**:The **I/O Configuration** tab is displayed.

I/O Configuration Tab Description

The embedded regular I/O modules are configurable from the **I/O Configuration** tab:

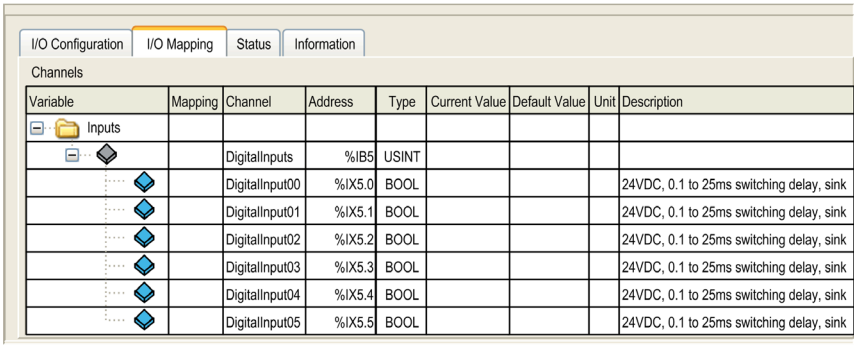


The **I/O Configuration** tab contains the following columns:

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 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
Current Value	Current value of the channel, displayed in online mode
Default Value	Value taken by the Output when the controller is in a STOPPED or HALT state *(see page 55)*. Double-click to change the default value.
Unit	Unit of the channel value
Description	Description of the channel

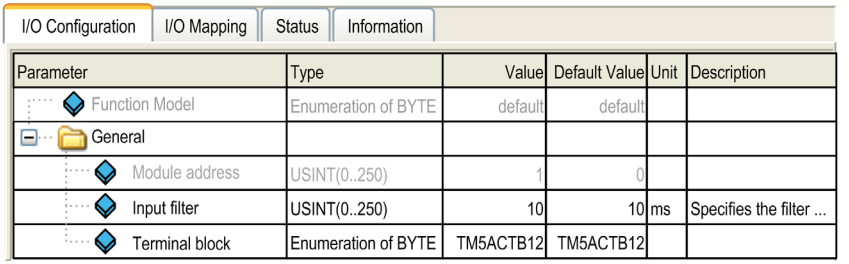
DI6DE Embedded Regular I/O Module

Introduction

The DI6DE embedded regular I/O module is a 24 Vdc Digital Inputs module with 6 inputs.

I/O Configuration Tab

To configure the DI6DE modules, select the **I/O Configuration** tab:



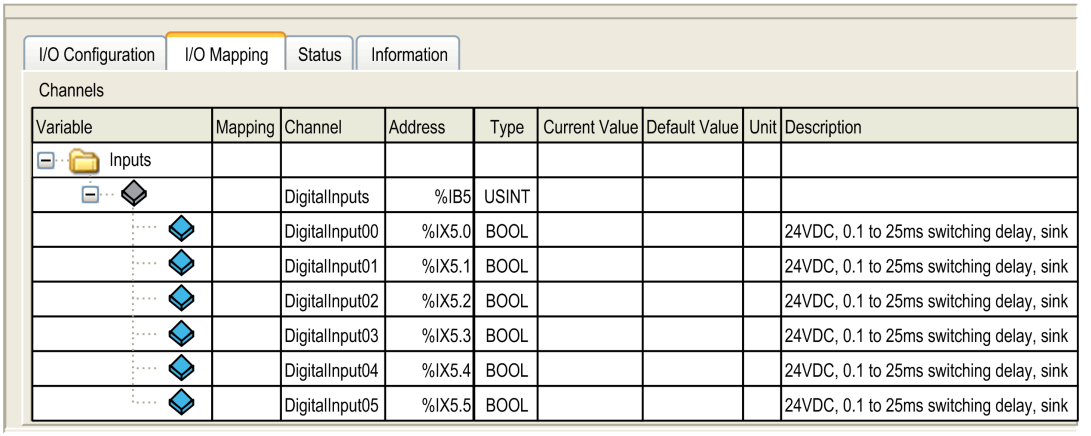
For further generic descriptions, refer to I/O Configuration Tab Description *(see page 108)*.

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
Terminal block	TM5ACTB12	TM5ACTB12	-	Terminal block associated with the 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.



Variable	Mapping	Channel	Address	Type	Current Value	Default Value	Unit	Description
Inputs								
		DigitalInputs	%IB5	USINT				
		DigitalInput00	%IX5.0	BOOL				24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput01	%IX5.1	BOOL				24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput02	%IX5.2	BOOL				24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput03	%IX5.3	BOOL				24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput04	%IX5.4	BOOL				24VDC, 0.1 to 25ms switching delay, sink
		DigitalInput05	%IX5.5	BOOL				24VDC, 0.1 to 25ms switching delay, sink

For further generic descriptions, refer to I/O Mapping Tab Description *(see page 108)*.

The table below describes the DI6DE I/O Mapping configuration:

Channel			**Type**	**Default value**	**Description**
Inputs	DigitalInputs		USINT	-	State of all inputs (bits 7-8 = 0, not used)
		DigitalInput00	BOOL	-	State of input 0
		...			...
		DigitalInput05			State of input 5

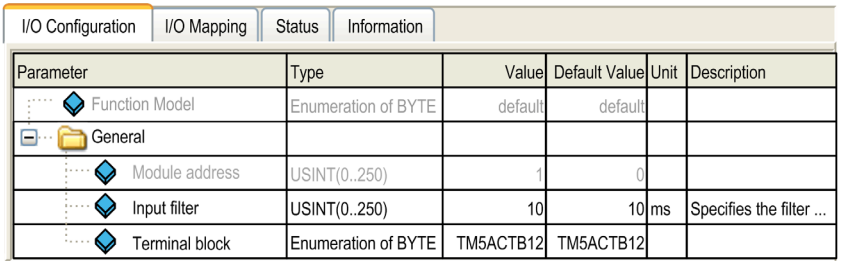
DI12DE Embedded Regular I/O Module

Introduction

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

I/O Configuration Tab

To configure the DI12DE module, select the **I/O Configuration** tab:



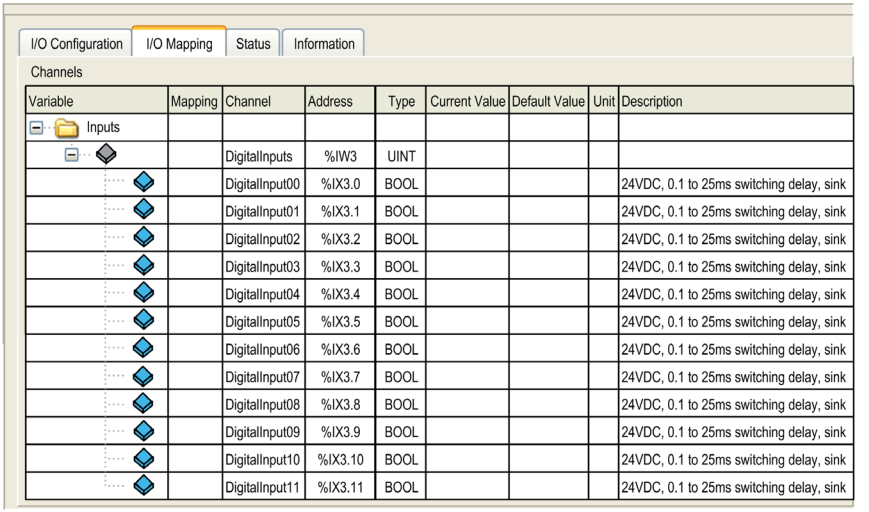
For further generic descriptions, refer to I/O Configuration Tab Description *(see page 108)*.

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
Terminal block	TM5ACTB12	TM5ACTB12	-	Terminal block associated with the 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.



Variable	Mapping	Channel	Address	Type	Current Value	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 108)*.

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

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

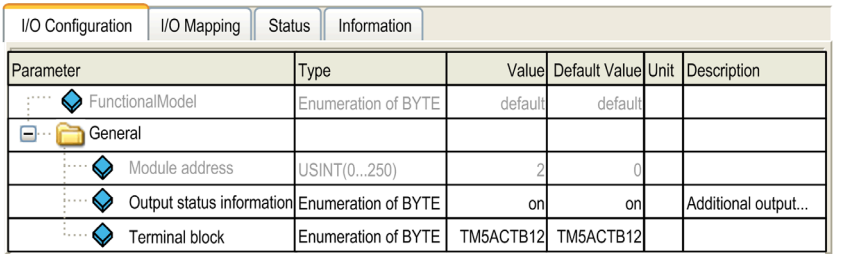
DO12TE Embedded Regular I/O Module

Introduction

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

I/O Configuration Tab

To configure the DO12TE module, select the **I/O Configuration** tab:



For further generic descriptions, refer to I/O Configuration Tab Description *(see page 108)*.

The table below describes the module parameters configuration:

Parameter	Value	Default Value	Description
Output status information	On Off	On	Additional output status information. **On**: the **StatusDigitalOutputs** word is added to the**I/O Mapping** tab.
Terminal block	TM5ACTB12	TM5ACTB12	Terminal block associated with the 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.

I/O Configuration | I/O Mapping | Status | Information

Channels

Variable	Mapping	Channel	Address	Type	Current Value	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 108)*.

The table below describes the I/O Mapping configuration:

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

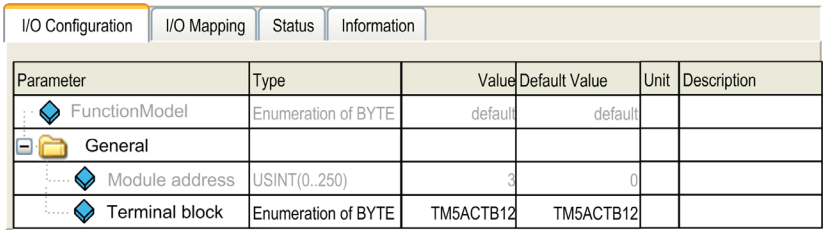
DO6RE Embedded Regular I/O Module

Introduction

The DO6RE embedded regular I/O module is a 30 Vdc/250 Vac Digital Outputs module with 6 relay outputs.

I/O Configuration Tab

To configure the DO6RE embedded regular I/O module, select the **I/O Configuration** tab:



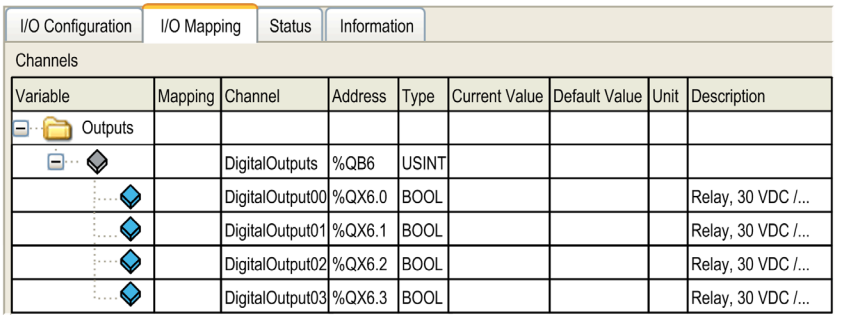
For further generic descriptions, refer to I/O Configuration Tab Description *(see page 108)*.

The table below describes the module parameters configuration:

Parameter	Value	Default Value	Description
Terminal block	TM5ACTB12	TM5ACTB12	Terminal block associated with the 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.



For further generic descriptions, refer to I/O Mapping Tab Description *(see page 108)*.

The table below describes the I/O Mapping configuration:

Channel			**Type**	**Default value**	**Description**
Outputs	DigitalOuputs		USINT	-	Command word of all outputs (bits 7-8: not used)
		DigitalOuput00	BOOL	None* TRUE FALSE	Command bit of output 0
		...			...
		DigitalOuput05			Command bit of output 5

*By default, the value is unspecified.

AI4LE Embedded Regular I/O Module

Introduction

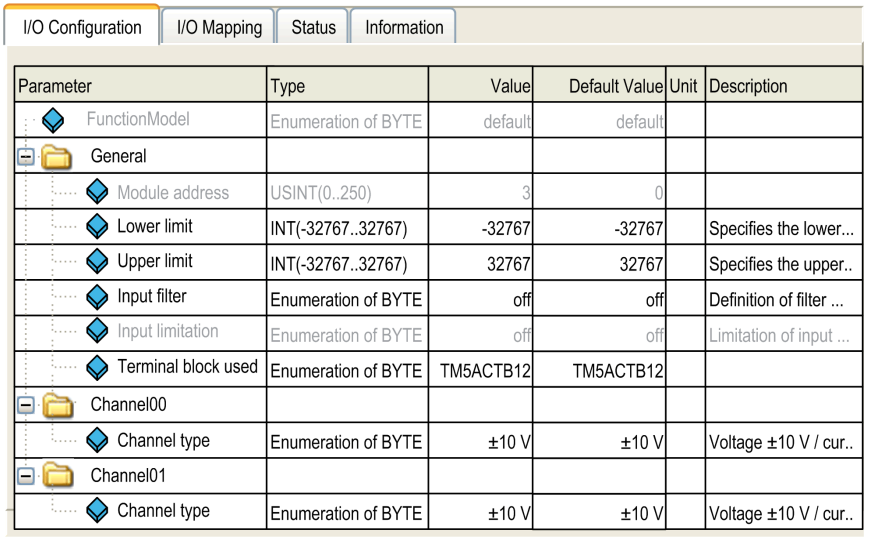
The AI4LE embedded regular I/O module is a ±10 Vdc/0...20 mA/4...20 mA analog input module with 4 inputs.

If you have wired your input for a voltage measurement, and you configure SoMachine for a current type of configuration, you may permanently damage the electronic module.

CAUTION
INOPERABLE EQUIPMENT Be sure that the physical wiring of the module is compatible with the software configuration for the module. **Failure to follow these instructions can result in equipment damage.**

I/O Configuration Tab

To configure the AI4LE module, select the **I/O Configuration** tab:



Parameter	Type	Value	Default Value	Unit	Description
FunctionModel	Enumeration of BYTE	default	default		
General					
Module address	USINT(0..250)	3	0		
Lower limit	INT(-32767..32767)	-32767	-32767		Specifies the lower...
Upper limit	INT(-32767..32767)	32767	32767		Specifies the upper..
Input filter	Enumeration of BYTE	off	off		Definition of filter ...
Input limitation	Enumeration of BYTE	off	off		Limitation of input ...
Terminal block used	Enumeration of BYTE	TM5ACTB12	TM5ACTB12		
Channel00					
Channel type	Enumeration of BYTE	±10 V	±10 V		Voltage ±10 V / cur..
Channel01					
Channel type	Enumeration of BYTE	±10 V	±10 V		Voltage ±10 V / cur..

For further generic descriptions, refer to I/O Configuration Tab Description *(see page 108)*.

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 121)*
	Upper limit	-32768...32767	32767	Specifies the upper measurement limit *(see page 121)*
	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 122)*
	Input limitation	Off 16383 8191 4095 2047 1023 511 255	Off	Specifies the limitation of input ramp *(see page 123)* **NOTE:** Parameter available if an input filter is selected.
	Terminal block	TM5ACTB12	TM5ACTB12	Lets you choose the terminal block associated with the electronic module
Channel 00	Channel type	±10 V 0 to 20 mA 4 to 20 mA	±10 V	Specifies the channel type
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

The input signal is monitored at the upper and lower limit value:

Limit value (default)	**Voltage signal ±10 V**		**Current signal 0...20 mA**		**Current signal 4...20 mA**	
Upper limit value	+10 V	+32767	20 mA	+32767	20 mA	+32767
Lower limit value	- 10 V	-32767	0 mA	0[1]	4 mA	0[2]

1. The analog value is limited down to 0.
2. The analog value is limited down to 0 at currents <4 mA. The status bit for the lower limit is set.

Other limit values can be defined if necessary. The limit values are valid for all channels. These are activated automatically by writing to the limit value register. From this point on, the analog values are monitored and limited according to the new limits. The monitor information is displayed in the status register.

Limit Analog Value

In addition to the status information, the analog value is set to the values listed below, by default, when an detected error occurs

NOTE: The reported analog value when an error is detected is either the default limit value as presented in the table below, or the user-defined limit value if the limit was changed from the default.

Detected error type	**Digital value**
Wire break	+32767 (7FFF hex)
Above upper limit value	+32767 (7FFF hex)
Below lower limit value	-32767 (8001 hex)
Invalid value	-32768 (8000 hex)

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:

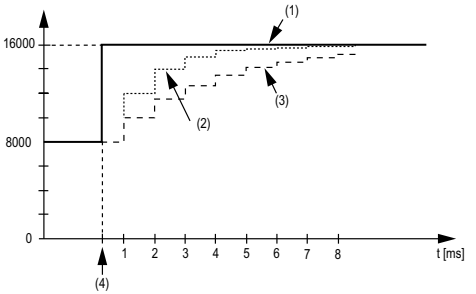
Filter level
Filter switched off
Filter level 2
Filter level 4
Filter level 8
Filter level 16
Filter level 32
Filter level 64
Filter level 128

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

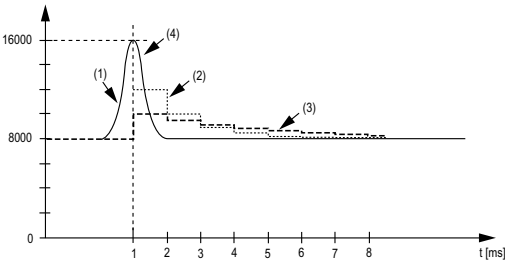


1 Input value.
2 Evaluated value: Filter level 2.
3 Evaluated value: Filter level 4.
4 Input jump.

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 make sure 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 below shows the adjustable limit values:

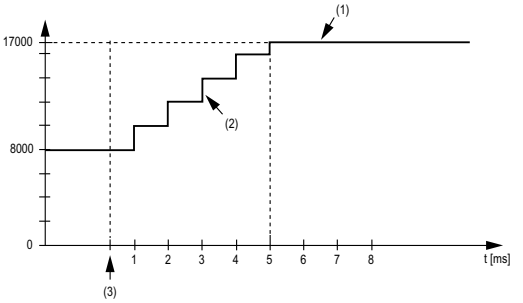
Limit value
The input value is used without limitation.
3FFF hex = 16383
1FFF hex = 8191
0FFF hex = 4095
07FF hex = 2047
03FF hex = 1023
01FF hex = 511
00FF hex = 255

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

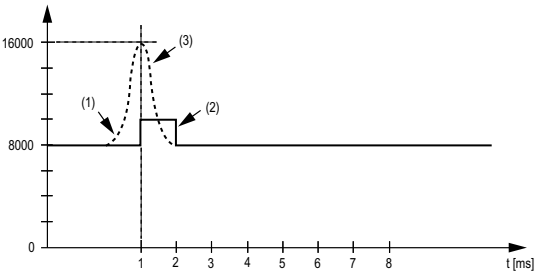


1 Input value.
2 Internal adjusted input value before filter.
3 Input jump.

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

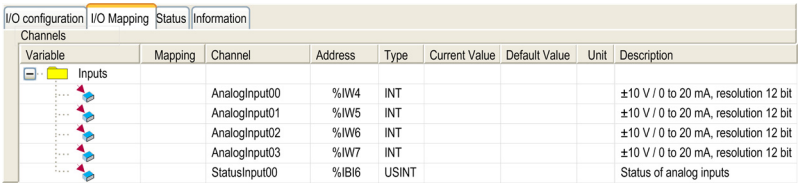
Filter level = 2



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 108)*.

The table below describes the I/O Mapping configuration:

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

The table below 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 102)*.

10.3 TM5 Expansion Modules Configuration

TM5 Expansion Module Configuration

Introduction

The Modicon M258 Logic Controller supports the following TM5 expansion modules:
- analog/digital modules
- specialized modules (HSC)
- transmitter and receiver modules
- power and common distribution modules
- dummy modules

For further information about the TM5 expansion modules configuration, refer to TM5 I/O Expansion Modules Configuration *(see Modicon TM5, Expansion Modules Configuration, Programming Guide)*.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Adding an Expansion Module

To add an expansion module, proceed as follows:

Step	Action
1	Select the **Configuration** tab.
2	In the Graphical Configuration Editor *(see page 17)*, click the **Add Expansion** button.
3	In the **Add Device** dialog box, select the expansion module and click the **Add Device** button.

PCI Expansion Modules Configuration

11

Introduction

This chapter describes the PCI Expansion Modules configuration of the Modicon M258 Logic Controller.

What's in this Chapter?

This chapter contains the following topics:

General Description

Introduction

The controller accepts 2 PCI Expansion Modules:

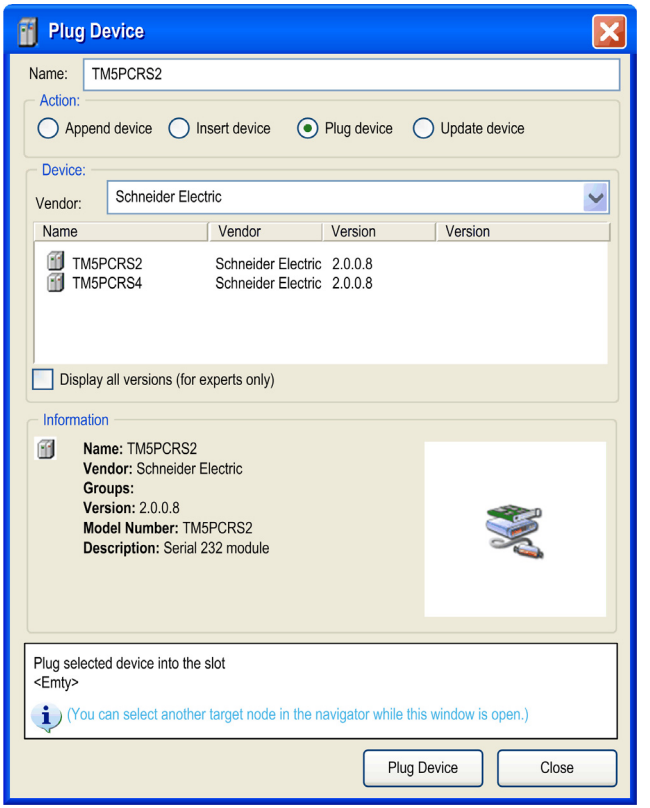
Reference	Description
TM5PCRS2	Serial Line RS232
TM5PCRS4	Serial Line RS485

NOTE: It is not possible to have more than one PCI Serial Com Module. The additional slot is reserved for the future PCI expansions.

Add a PCI Expansion Module

Add a PCI Expansion Module

To add a PCI Expansion Module on your controller, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** entry in the left hand side.
3	Click the **PCI Slots** →**<Empty>** entry.
4	Click the **Plug Device** button. Choose the PCI Expansion Module and click **Plug Device** button: 

For further information, refer to PCI expansion module configuration *(see Modicon TM5, PCI Modules Configuration, Programming Guide)*.

Ethernet Configuration

12

Introduction

This chapter describes how to configure the Ethernet network interface of the Modicon M258 Logic Controller.

What's in this Chapter?

This chapter contains the following sections:

12.1 Ethernet Services

What's in this Section?

This section contains the following topics:

Ethernet Services

Ethernet Services

Ethernet Services

The controller supports the following services:
- Modbus TCP Server *(see page 142)*
- Modbus TCP Client *(see page 142)*
- Web Server *(see page 147)*
- FTP Server *(see page 159)*
- SNMP *(see page 160)*
- EthernetIP Device *(see page 163)*
- Modbus Device *(see page 184)*

Ethernet Protocol

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)

TCP Server Connection

This table shows the maximum number of TCP server connections:

Connection Type	Maximum number of server connection
Modbus Server	8
Modbus Device	2
EthernetIP Device	16
FTP Server	4
Web Server	10

Each server based on TCP manages its own pool of connections.

When a client tries to open a connection that exceeds the poll size, the controller closes the oldest connection.

If all connections are busy (exchange in progress) when a client tries to open a new one, the new connection is denied.

All server connections stay open as long as the controller stays in operational states (RUN, STOP, HALT).

All server connections are closed when leaving or entering operational states (RUN, STOP, HALT), except in case of power outage (because the controller did 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 a 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 details, refer to Online Help CoDeSys part.

IP Address Configuration

Introduction

There are four 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 38)*. If a post configuration file exits, this assignment method has priority over the others.

NOTE: If the attempted addressing method is unsuccessful, the controller will start using a default IP address *(see page 140)* derived from the MAC address.

You must carefully manage the modules' IP addresses because each device on the network requires a unique address. Having multiple devices with the same IP address can cause unpredictable operation of your network and associated equipment.

⚠ WARNING

UNINTENTED EQUIPMENT OPERATION

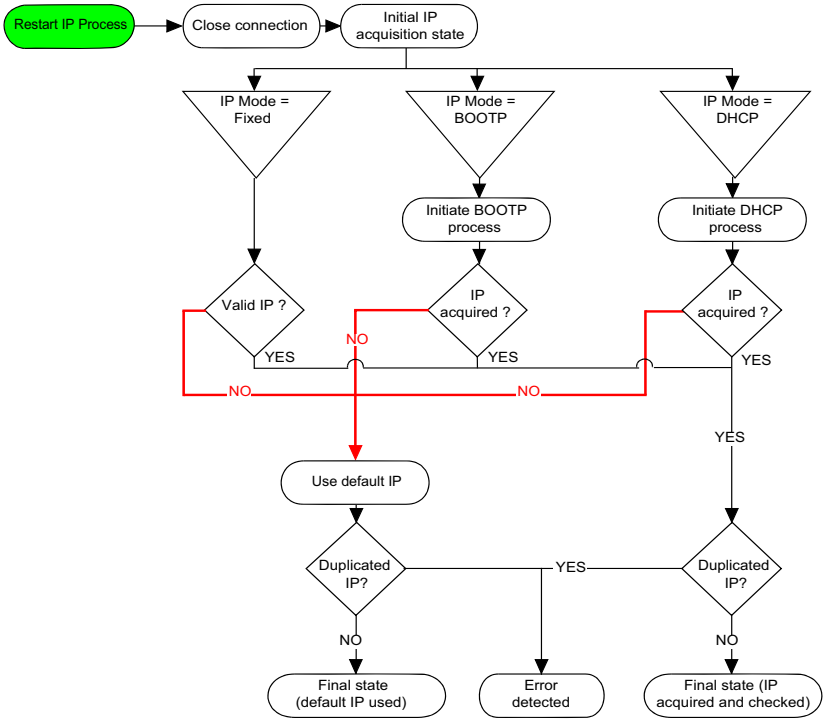
- Be sure that there is only one master controller configured on the network or remote link.
- Be sure that all slave devices have unique addresses. Be sure that all slave devices have unique addresses. You cannot have duplicated addresses.
- Obtain your IP address from your system administrator.
- Confirm that the device's IP address is unique before placing the system into service.
- Do not assign the same IP address to any other equipment on the network.

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

NOTE: It is good practice to ensure that your system administrator maintains a record of all assigned IP addresses on the network and subnetwork, and to inform the system administrator of all configuration changes performed.

Address Management

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



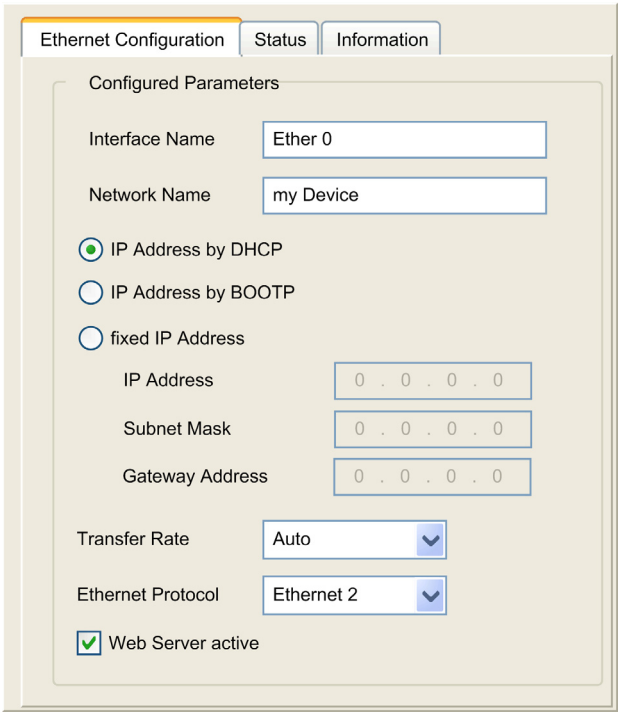
NOTE: If a device programmed to use the DHCP or BOOTP addressing methods is unable to contact its respective server, the controller uses the default IP address. It will, however, constantly reiterate its request.

The IP process automatically 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 on the **Ethernet** item:



Element	Description
Interface Name	Name for the network link
Network Name	Used as device name to retrieve IP address through DHCP, max 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.
Transfer Rate	Transfer rate and direction on the bus are automatically configured.
Ethernet Protocol	Protocol type used (Ethernet2 or IEEE 802.3)
Web Server active	Enable/disable Web Server

Default IP Address

The default IP address is based on the device's MAC address. The first two bytes are 10 and 10. The last two bytes are the last two bytes of the device's MAC address.

The default subnet mask is 255.0.0.0.

NOTE: A MAC address is always written in hexadecimal format, and an IP address in decimal format. You must 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 (known as 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 the following table:

Address Class	Byte 1					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 sub-network and the device address in the host ID.

The subnet address is obtained by retaining the bits of the IP address which 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 which 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 sub-network when there is no gateway.

Gateway

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

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

Modbus TCP Server/Client

Introduction

The Modbus protocol is widely used in industry. Unlike Modbus serial link, Modbus TCP/IP is not based on a hierarchical structure, but on a client / server model.

The transfer of information between a Modbus client and server is initiated when the client sends a request to the server to transfer information, to execute a command, or to perform one of many other possible functions.

After the server receives the request, it executes the command or retrieves the required data from its memory. The server then responds to the client by either acknowledging that the command is complete or by providing the requested data.

The Modicon M258 Logic Controller implements both client and server services so that it can initiate communications to other controllers and I/O devices, and to respond to requests from other controllers, SCADA, HMIs and other devices.

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

The Modbus Server/Client 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, see Function Block Descriptions *(see Communication Functions:, PLCCommunication Library)* of the PLCCommunication Library.

Modbus TCP Server

The Modbus Server supports the following Modbus requests:

Function Code Dec (Hex)	Sub-function Dec (Hex)	Function
1 (1h)		Read digital outputs (%Q)
2 (2h)		Read digital inputs (%I)
3 (3h)		Read holding register (%MW)

Function Code Dec (Hex)	Sub-function Dec (Hex)	Function
6 (6h)		Write single register (%MW)
8 (8h)		Diagnostic *(see page 143)*
15 (Fh)		Write multiple digital outputs (%Q)
16 (10h)		Write multiple registers (%MW)
23 (17h)		Read/write multiple registers (%MW)
43 (2Bh)	14 (Eh)	Read device identification *(see page 146)*

Diagnostic Request

The following table contains the Data Selection Code list:

Data Selection Code	Description
0x00	Reserved
0x01	Basic Network Diagnostics *(see page 143)*
0x02	Ethernet Port Diagnostic *(see page 144)*
0x03	Modbus TCP/Port 502 Diagnostics *(see page 145)*
0x04	Modbus TCP/Port 502 Connection Table *(see page 146)*
0x05 - 0x7E	Reserved for other public codes
0x7F	Data Structure Offsets

Basic Network Diagnostics

Basic Network Diagnostics

Field Name	Bytes	TR Designation
Basic NW Diag Validity	4	-
Communication Global Status	2	-
Supported Communication Services	2	-
Status of Communication Services	2	-
IP Address	4	IP Address
Subnet Mask	4	Subnet Mask
Default Gateway	4	Default Gateway
MAC Address	6	MAC Address
Ether Frame Format Capability / Configuration / Operational	6	Ethernet Frame Format
Ether Rcv Frames OK	4	Total number of Ethernet Frames received OK

Field Name	Bytes	TR Designation
Ether Xmit Frames OK	4	Total number of Ethernet Frames Transmitted OK
Reserved	2	-
Num MB Open Server Connections	2	Num_Open_ServerCnx
Num MB Error Msgs Sent	4	Num_MB_Error_Msgs_Sent
Num MB Msgs Sent	4	Num_MB_Msgs_Sent
Num MB Msgs Rcvd	4	Num_MB_Msgs_Rcvd
Device Name	16	Device Name
IP Assignment Mode Capability / Operational	4	IPAssignment ModeCapability; IPAssignmentModeOperational
Total:	78	

Ethernet Port Diagnostic

Ethernet Port Diagnostic: Port Diagnostics Data Validity

Field Name	Bytes	TR Designation
Port Diagnostics Data Validity	2	-
Logical/Physical Port Number	2	-
Ether Control Capability	2	Cable Type - Duplex Status
Link Speed Capability	2	Speed
Ether Control Configuration	2	-
Link Speed Configuration	2	Speed
Ether Control Operational	2	-
Link Speed Operational	2	Speed
Port MAC Address	6	MAC Address
Media Counters	72	-
Reserved	46	-
Total:	140	

Ethernet Port Diagnostic: Media Counters Diagnostic Data Validity

Field Name	Bytes	TR Designation
Media Counters Data Validity	4	-
Num Frames Xmit OK	4	Frames transmitted OK
Num Frames Received OK	4	Frames received OK
Reserved	60	-
Total:	72	

Modbus TCP/Port 502 Diagnostics

Modbus TCP/Port 502 Diagnostics:

Field Name	Bytes	TR Designation
Modbus TCP/Port 502 Diag Validity	4	-
Port 502 Status	2	-
Num Open Connections	2	Num_Open_Cnx
Num MB Msgs Sent	4	Num_MB_Msgs_Xmit
Num MB Msgs Received	4	Num_MB_Msgs_Rcvd
Num Open Client Connections	2	Num_Open_ClientCnx
Reserved	2	-
Max Num Connections	2	Max_Num_Cnx
Max Num Client Connections	2	Max_Num_ClientCnx
Reserved	2	-
Num MB Error Msgs Sent	4	Num_MB_Error_Msgs_Sent
Reserved	102	-
Total:	34 + 6*N + 2	

Modbus TCP/Port 502 Connection Table

Modbus TCP/Port 502 Connection Table:

Field Name	Bytes	TR Designation
Connection Table Validity	2	-
Number of Entries (NE)	2	-
Starting Entry Index (SE)	2	-
Connection Table Entry 1	16	-
Connection Table Entry 2	16	-
Reserved	...	-
Connection Table Entry N	16	-
Total:	6 + 16 * N	

Read Device Identification Request

The table below list the objects that can be read with a read device identification request (basic identification level):

Object ID	Object Name	Type	Value
00h	Vendor name	ASCII string	Schneider Electric
01h	Product code	ASCII string	Controller reference eg: TM258LD42DT
02h	Major / minor revision	ASCII string	aa.bb.cc.dd (same as device descriptor)

Web Server

Introduction

The controller provides as 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. They 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 135)*.

NOTE: The Web server can be disabled by setting the **Web Server active** parameter in the Ethernet Configuration tab *(see page 139)*.

The Web server is a powerful tool for both reading and writing data to your controller, with full access to all data in your application. If, however, there are security concerns over data writing, you must disable this service to prevent unauthorized access to your application. By enabling the Web server, you enable the writing of data.

⚠ WARNING

UNAUTHORIZED DATA ACCESS

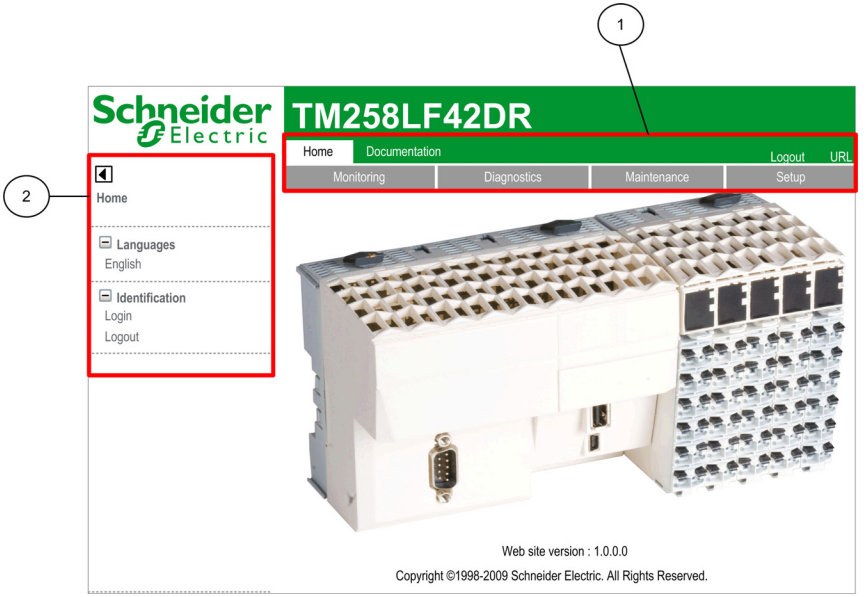
Disable the Web server to prevent any unwanted or unauthorized access to data in your application.

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

NOTE: In this section the screenshots are given as examples, the Web server is identical to the range of performance controllers.

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 an USB connection:



Item	Description
1	Generic menu bar *(see page 149)*
2	Active page Sub-menu

NOTE: Access to the home page does not require any access rights.

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 148)*	Allow login and password entry
Documentation	References	Link to the brand site.
Monitoring	PLC Viewer *(see page 150)*	• Serial Number • Version (firmware, boot...) • Configuration status
	Expansion Viewer *(see page 151)*	See status of expansion modules.
	I/O Viewer *(see page 152)*	See module by module I/O values.
	Oscilloscope *(see page 152)*	Display of two variables in the form of a recorder-type time chart.
	Data parameters *(see page 154)*	Display and modify controller variables.
Diagnostics	PLC *(see page 155)*	Controller status
	Ethernet *(see page 155)*	Ethernet status
	Serial *(see page 156)*	Serial line status
Maintenance	FTP *(see page 156)*	Link to the file system server (/Usr and /Sys folders)
Setup	Post configuration setup *(see page 157)*	Set the Ethernet and serial line parameters.
	EthernetIP configurations files *(see page 158)*	Set the EthernetIP configuration files.

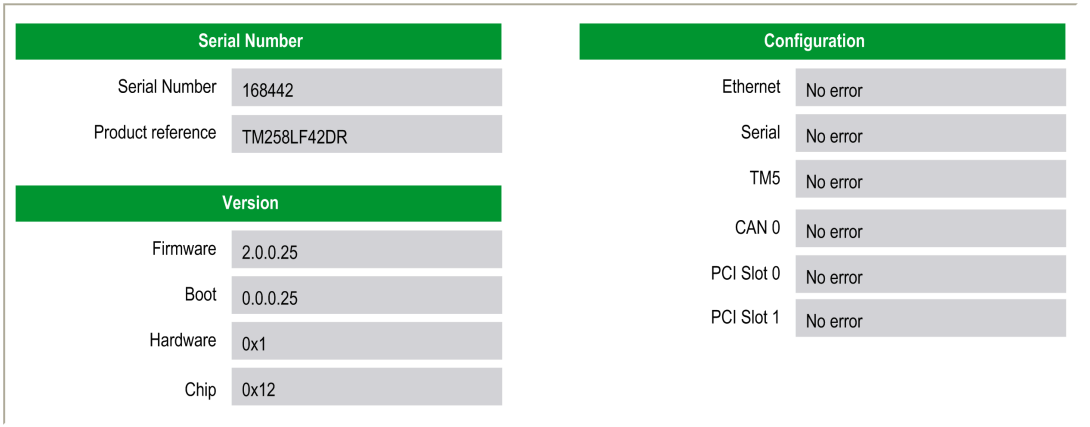
Page Access

This table lists the status and user rights necessary to access pages:

Menu	Page	Controller State			
		EMPTY	STOPPED	RUNNING	HALT
Home	Home	X	X	X	X
Documentation	References	X	X	X	X
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	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 PLC Viewer Page

The PLC Viewer page shows the controller status:

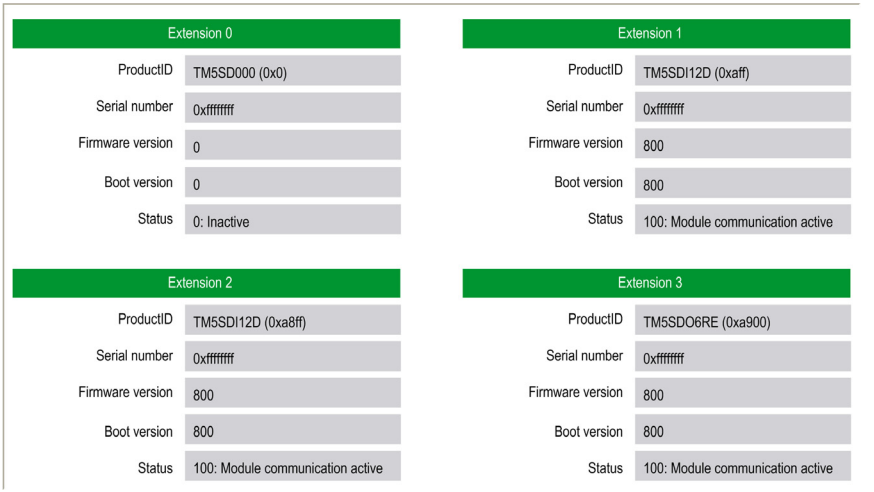


The following table describes the Configuration status fields:

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

Monitoring Expansion Viewer Page

The Expansion Viewer page shows the expansion module status:

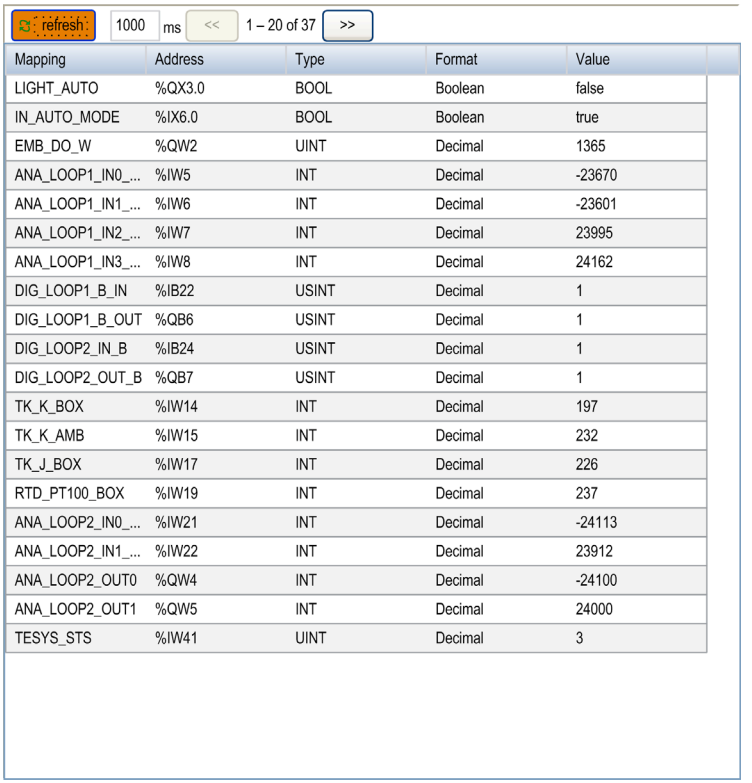


The following table describes each status code:

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 Page

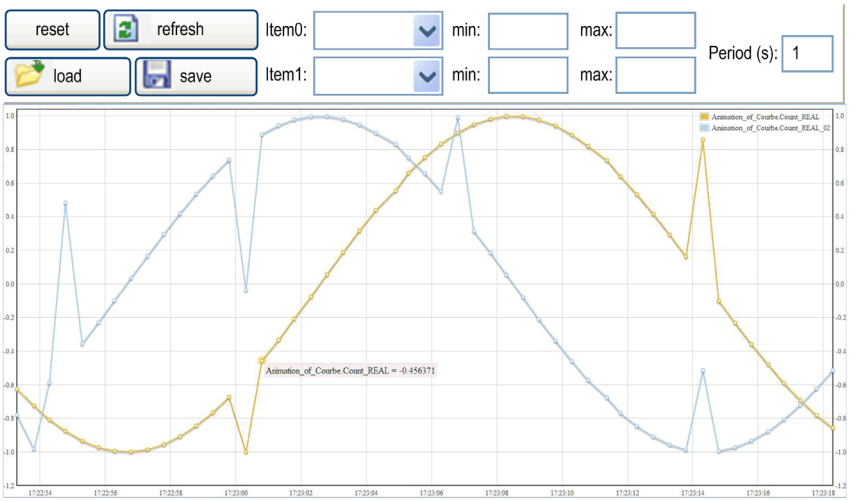
The IO Viewer lets you display and modify the I/O:



Element	Description
Refresh	Enable I/O refreshing: ● gray button: refreshing disabled ● orange button: refreshing enabled
1000 ms	I/O refreshing period in ms
<<	Go to previous I/O list page
>>	Go to next I/O list page

Monitoring Oscilloscope Page

The oscilloscope page lets you display two variables in the form of a recorder time chart:



Element	Description
Reset	Erase the memorization
Refresh	Start/stop refreshing
Load	Load parameter configuration of Item0 and Item1
Save	Save 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

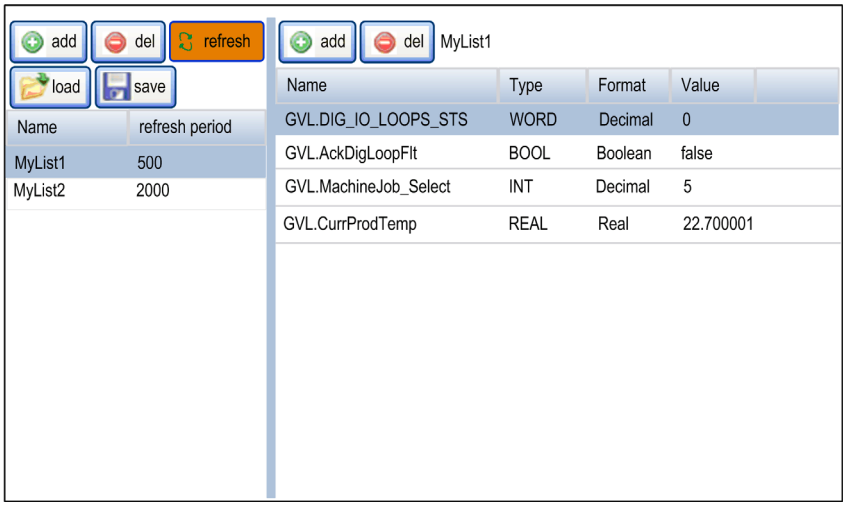
NOTE: The variables list is updated only during the boot application creation, that is done automatically during the download if the **Login with download** option is checked.

If the **Login with online change** option is checked, the new variables will not appear.

For further information refer to Transferring Applications *(see SoMachine, Programming Guide)*.

Monitoring Data Parameters Page

The Data Parameters page lets you display and modify variables values:



Element	Description
Load	Load saved lists
Save	Save the selected list description in the controller (*/usr/web* directory)
Add	Add a list description or a variable
Del	Delete a list description or a variable
Refresh period	Refreshing period of the variables contained in the list description (in ms)
Refresh	Enable I/O refreshing: ● gray button: refreshing disabled ● orange button: refreshing enabled

NOTE: IEC objects (%IW, %M,...) are not accessible.

NOTE: The variables list is updated only during the boot application creation, that is done automatically during the download if the **Login with download** option is checked.

If the **Login with online change** option is checked, the new variables will not appear.

For further information refer to Transferring Applications *(see SoMachine, Programming Guide)*.

Diagnostic PLC Page

The PLC page enables to display controller information:

Identification	
VendorID	0x101a
Vendor name	Schneider Electric
ProductID	0x205
Product reference	TM258LF42DR
Serial Number	168442
Node name	(TM258LF42DR) EthMAC 00-80-F4-4

Version	
Firmware	2.0.0.19
Boot	0.0.0.147
Hardware	0x1
Chip	0x12

Status	
Application status	0 : Empty
Boot project status	2 : Different boot project
Local IO status	1 : No init
Remote IO status	0
Clock Battery Status	FFFF : Ok
Last stop cause	0: Unknown
Last application error	0 : No errror
System Fault 1	No error
System Fault 2	No error
Last stop time	Thu, 1 Jan 1970 00:00:00
Last power-off time	Fri, 9 Oct 2009 17:03:41
Events counter	0
Terminal port status	2 : connected
USB Host status	0 : not connected

Extension bus	
Bus status	0b0000000000000000 :
Sync error count	0
ASync error count	0
Break count	0
Topology change count	0
Cycle count	0

File	
File system free handle	25
File system total bytes	127795200 (122 MB)
File system free bytes	126644224 (121 MB)

Diagnostic Ethernet Page

The Ethernet page enables to display Ethernet information:

Current IP	
MAC address	0.80.F4.40.0.21
IP address	0.0.0.0
Subnet mask	0.0.0.0
Gateway address	0.0.0.0

Fast device replacement	
IP mode	255 : Default IP
Device name	
FDR server	0.0.0.0

Ethernet statistics	
Opened Tcp connections	2
Frames transmitted OK	0
Frames received OK	0
Buffers transmitted NOK	0
Buffers received NOK	0

Ethernet port	
Status	0 : Link down
Speed	10
Duplex mode	0 : Half Duplex
Collisions	0
Frame sending protocol	0 : Ethernet II

Ethernet IP statistics	
IO Messages transmitted	0
IO Messages received	0
UCMM Request	0
UCMM Error	0
Class3 Request	0
Class3 Error	0
Assembly Instance Input	0
Assembly Instance Input size	0
Assembly Instance Output	0
Assembly Instance Output size	0
Connection timeouts	0

Modbus statistics	
Messages transmitted OK	0
Messages received OK	0
Error messages	0

Diagnostics Serial Page

The Serial page enables to display Serial information:

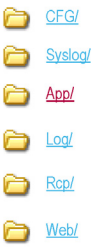
Serial 0	
Frames transmitted OK	0
Frames received OK	0
RX Message error	0
Slave exception count	0
Slave no response count	0
Slave Nak count	0
Slave busy count	0

Serial 1	
Frames transmitted OK	0
Frames received OK	0
RX Message error	0
Slave exception count	0
Slave no response count	0
Slave Nak count	0
Slave busy count	0

Maintenance Page

The Maintenance page lets you access the **/usr** and **/sys** folders of the controller flash memory *(see page 31)*:

Index of /usr

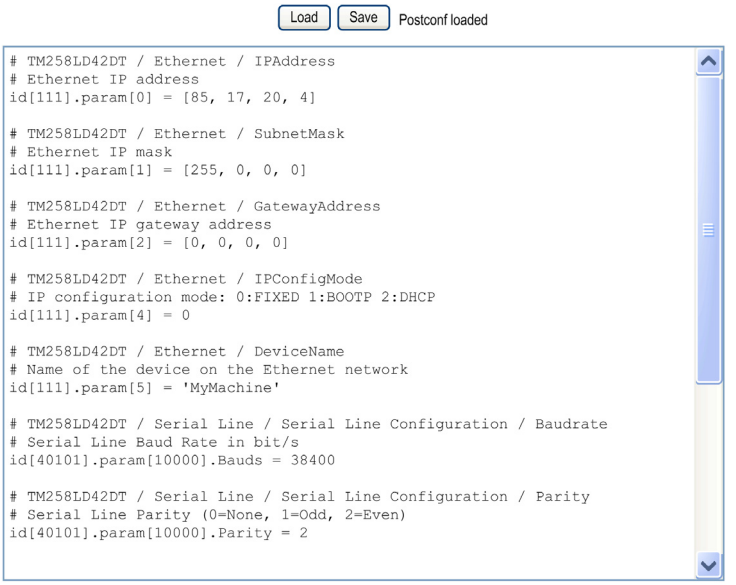


NOTE: Do not access the Visu or Web directories in the index /usr. The files in these directories should not be altered.

Index of /sys

Setup Post Configuration Page

Select the **Setup** menu to access the **Postconf** page:



The following table describes the procedure allowing parameters modification:

Step	Action
1	Click **Load**
2	Modify the parameters *(see page 38)*
3	Click **Save** **NOTE:** The new parameters will be considered at next reboot.

Setup EthernetIP configurations files

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

Index of /usr


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

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 controller's data storage area.

NOTE: The FTP server is available even if the controller is empty (no user application).

FTP Access

When the controller is connected through the USB port, the FTP server is accessible at address 90.0.0.1.

The login is anonymous without any password.

Files Access

See File Organization *(see page 32)*.

SNMP

Introduction

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

The data is stored in an MIB (Management Information Base). 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 Object Handle

Object	Description	Access	Default Value
SysDescr	Text description of the device	Read	SCHNEIDER M258 Fast Ethernet TCP/IP
SysObjectID	Points to the product reference in the private MIB	Read	1.3.6.1.4.1.3833.1.7.36
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	TM258LF42DT••
SysLocation	Physical location of the product	Read/Write	-
SysService	Indicates the type of service provided by this product	Read	79

NOTE: The SysContact, SysName and SysLocation objects can be modified by the user.

The values written are saved to the controller via SNMP client tool software. The Schneider Electric software 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.

12.2 Ethernet Optional Devices

What's in this Section?

This section contains the following topics:

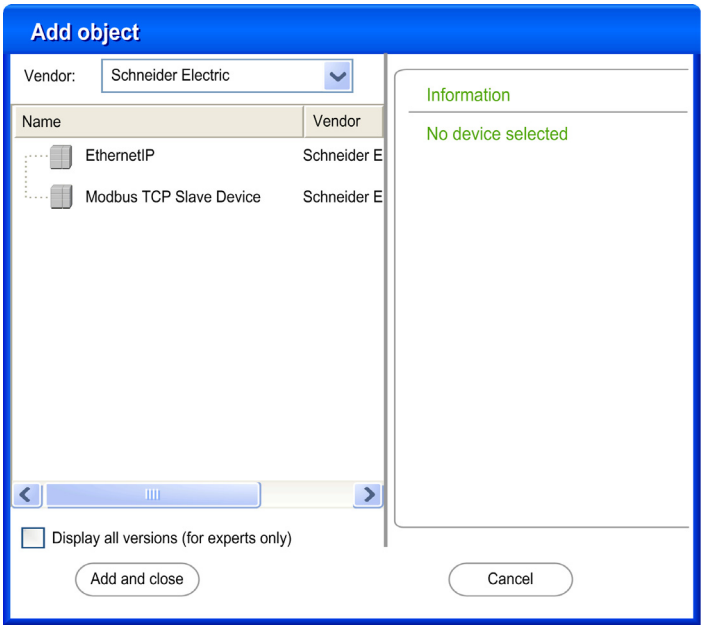
Ethernet Manager

Adding an Ethernet Manager

The controller supports the following Ethernet managers:

- EthernetIP (for CIP Device)
- ModbusTCP Slave Device

To add an Ethernet manager, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double click on the controller.
2	Click the **Communication** entry on the left hand side of the screen.
3	Click the **Ethernet** →**Protocol Settings** sub-entries.
4	Click the **No manager defined** sub-entry. **Result:** The dialog box for Ethernet manager will be displayed.  **NOTE:** You can also open this dialog box by clicking on the port of the controller in the graphical configuration editor *(see page 17)*. Note however that this leads to a new configuration of the port and removes the configured settings which are already available.
5	Select the Ethernet manager in the list and click on the **Add and close** button.

EtherNet/IP Device

Introduction

This section describes the connection 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

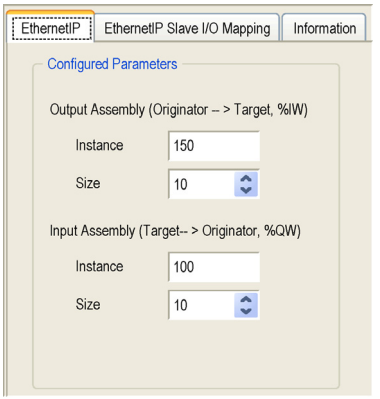
See Adding an Ethernet Manager *(see page 162)*.

EtherNet/IP Device Configuration

To configure the EtherNet/IP Device parameters, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** entry on the left hand side of the screen.
3	Click the sub-entries **Ethernet** →**Protocol Settings** →**EthernetIP**.

The following dialog box is displayed:



Element	Admissible Controller Range	SoMachine Default Value
Output Assembly Instance	150...189	150
Output Assembly Size	2...40	10
Input Assembly Instance	100...149	100
Input Assembly Size	2...40	10

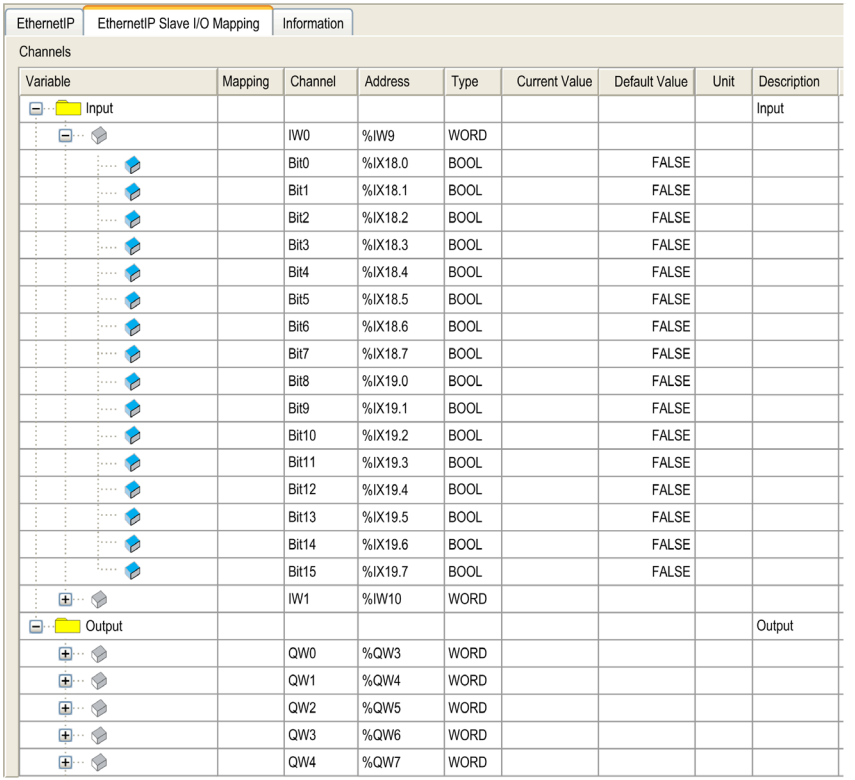
EDS File Generation

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 Ethernet network is working correctly on the controller (cable connected and the IP address is acquired).

EtherNet/IP Device 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.


EthernetIP | EthernetIP Slave I/O Mapping | Information

Channels

Variable	Mapping	Channel	Address	Type	Current Value	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 108)*.

The table below describes the EtherNet/IP Device 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 word depends on the size parameter configurated in EtherNet/IP Device Configuration *(see page 163)*.

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

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

Connections on EtherNet/IP

To access a slave, it is necessary to 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 connections).

One I/O connection uses 2 sessions.

The following table shows the EtherNet/IP connections limitations:

Characteristic	Description
Maximum explicit connection	8 (Class 3)
Maximum I/O connection	1 (Class 1)
Maximum connection	8
Maximum session	16
Maximum simultaneous request	32

Profile

The controller supports the following objects:

Object class	Class ID	Cat.	Number of instances	Effect on interface behavior
Identity Object *(see page 166)*	01h	1	1	Supports the reset service
Message Router Object *(see page 169)*	02h	1	1	Explicit message connection

Object class	Class ID	Cat.	Number of instances	Effect on interface behavior
Assembly Object *(see page 173)*	04h	2	2	Defines I/O data format
Connection Manager Object *(see page 175)*	06h		1	-
File Object *(see page 177)*	37h		2	Allows to exchange EDS file
Modbus Object *(see page 179)*	44h		1	-
TCP/IP Interface Object *(see page 180)*	F5h	1	1	TCP/IP configuration
Ethernet Link Object *(see page 182)*	F6h	1	1	Counter and status information

Identity Object

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	Name	Description
01h	Get Attribute All	Returns the value of all class attributes
0Eh	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

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

(1) Reset Service description:

When the Identity Object receives a Reset request, it:
- determines if 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	Emulate as closely as possible cycling power. Simulate Reboot command. **NOTE:** This value is the default value if this parameter is omitted.
1	Emulate as closely as possible the removal and reapplication of supply power to the controller and a restoration of the I/O to initialization values.
2	Return as closely as possible to the out-of-box configuration, with the exception of communication link parameters, and emulate cycling power as closely as possible. The communication link parameters that are to be preserved are defined by each network type. See the Reset service of the network specific link object(s) for complete information. Simulate Reset origin command.
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	PLC
3	Get	Product code	UINT	805h	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
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: TM258LD42DT.

(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	-

Bit	Name	Description
4...7	Extended Device Status	• 0: self-testing or unknown • 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 Fault	TRUE indicates the device detected an error, which is thought to be recoverable. This type of event does not lead to a change in the device state.
9	Minor Unrecoverable Fault	TRUE indicates the device detected an error, which is thought to be unrecoverable. This type of event does not lead to a change in the device state.
10	Major Recoverable Fault	TRUE indicates the device detected an error, which requires the device to report an exception and enter into the HALT state.
11	Major Unrecoverable Fault	TRUE indicates the device detected an error, which requires the device to report an exception and enter into the HALT state.
12...15	Reserved	-

Message Router Object

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).

Attribute ID	Access	Name	Data Type	Value	Details
5	Get	Optional Service List	UINT	00h	The number and list of any implemented optional services attribute (0: no optional services supported)
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	Name	Description
01h	Get Attribute All	Returns the value of all class attributes
0Eh	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code	Name	Description
01h	Get Attribute All	Returns the value of all class attributes
0Eh	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
105	Get	Total outgoing unconnected packets sent during the last second	UINT	-	Total number of outgoing unconnected packets sent during the last second

Attribute ID	Access	Name	Data Type	Value	Description
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 no-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).
115	Get	Total incoming unconnected packets Invalid Parameter Value	UINT	-	Total number of incoming unconnected packets that targeted a no-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

Attribute ID	Access	Name	Data Type	Value	Description
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

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 supported)
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	Name	Description
0Eh	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code	Name	Description
10h	Get Attribute Single	Modifies the value of the specified attribute
0Eh	Get Attribute Single	Returns the value of the specified attribute
18h	Get Member	Reads a member of an assembly object instance
19h	Set Member	Modifies a member of an assembly object instance

Instances Supported

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

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

The controller supports 2 assemblies:

Name	Instance	Data size
Native Controller Output (%IW)	Configurable: must be between 100 and 149	2...40 words
Native 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 Native 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	2004h	Class 4
1	24xxh	Instance xx where xx is the instance value (example: 2464h = instance 100).
2	30h	Attribute 3

Connection Manager Object

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
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 every next word contains another attribute code. Following optional attributes include: ● total number of incoming connection open requests ● the number of requests rejected because of the unexpected 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	Name	Description
01h	Get Attribute All	Returns the value of all class attributes
0Eh	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

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

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
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	Name	Description
0Eh	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 0xC8 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 "M258xxx.eds" where M258xxx 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 0xC9 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 "M258xxx.gz" where M258xxx is the exact reference of the controller. This is zip encoded file containing only one file: M258xxx.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	Name	Description
0Eh	Get Attribute Single	Returns the value of the specified instance attribute
4Bh	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.
4Fh	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 0xC8 instance it is "EDS and Icon Files". For the 0xC9 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	-	Define 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

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	Name	Description
0Eh	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	Name	Description
4Bh	Read Digital Inputs	Returns the value of one or several contiguous Digital Input registers
4Ch	Read Coils	Returns the value of one or several contiguous Coils
4Eh	Read Holding Registers	Returns the value of one or several contiguous Holding Registers

Service Code	Name	Description
4Fh	Write Coils	Updates the value of one or several contiguous Coils
50h	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

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	Name	Description
01h	Get Attribute All	Returns the value of all class attributes
0Eh	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	Name	Description
01h	Get Attribute All	Returns the value of all instance attributes
0Eh	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 for future use.
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 others bits are reserved and set to 0.
3	Get	Configuration	DWORD	Bit level	• 0: The interface configuration is valid. • 1: The interface configuration must be obtained with BOOTP. • 2: The interface configuration must be obtained with DHCP. • 3: reserved • 4: DNS Enable All others 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

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	Name	Description
01h	Get Attribute All	Returns the value of all class attributes
0Eh	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	Name	Description
01h	Get Attribute All	Returns the value of all instance attributes
10h	Set Attribute Single	Modifies the value of the specified attribute
0Eh	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 fault All others 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

Introduction

This section describes the connection 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 (3h, 6h, etc.), these commands do not have their standard meaning. Because this device is acting as a gateway for an external I/O scanner (Modbus Master), the normal 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.modbus.org website.

Adding a Modbus TCP Slave Device

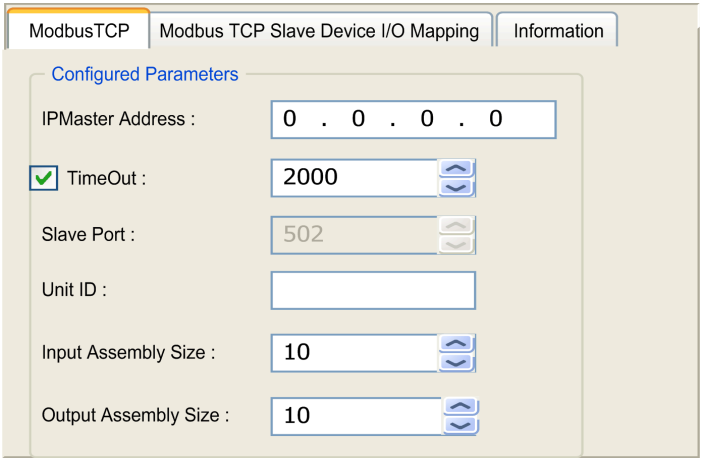
See Adding an Ethernet Manager *(see page 162)*.

Modbus TCP Configuration

To configure the Modbus TCP slave device, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** entry on the left hand side.
3	Click **Ethernet** →**Protocol Settings** →**Modbus TCP Slave Device**.

The following 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 ms (step 500 ms) **NOTE:** The timeout applies to the IP Master Address unless if the address is 0.0.0.0.
Slave Port	Modbus communication port (502 by default)
Unit ID	Modbus slave address (1...255)
Holding Registers (%IW)	Size of the input assembly in bytes (2...40 bytes)
Input Registers (%QW)	Size of the output assembly in bytes (2...40 bytes)

I/O Mapping Tab

The I/Os are mapped to Modbus registers from Master point of view as following:

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

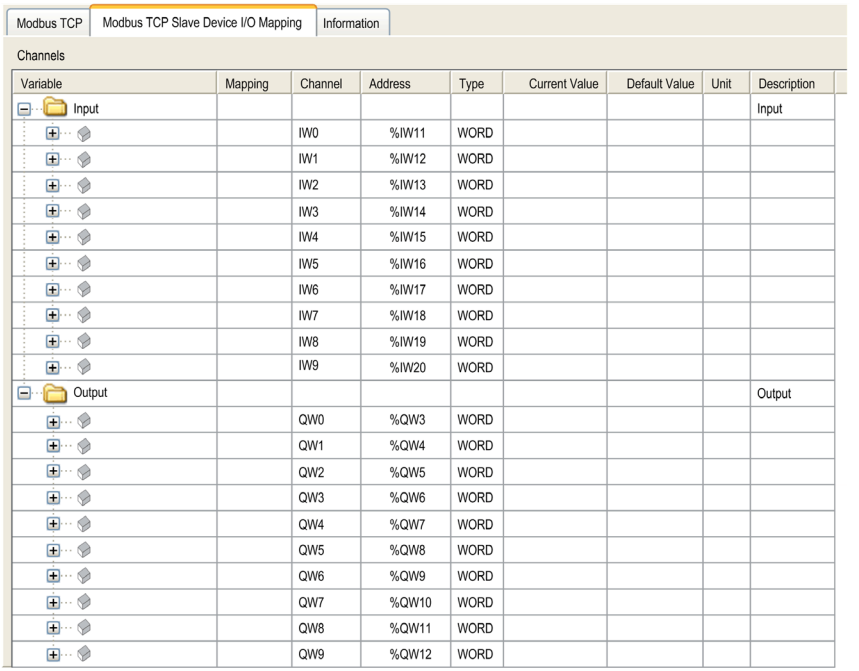
Once a Modbus TCP Slave device has been configured, Modbus commands sent to its Unit ID (Modbus address) are handled differently than the same command would be when addressed to any other Modbus device on the network. For example, when the Modbus command 3 (3h) is sent to a standard Modbus device, it reads and returns the value of one or more registers. When this same command is sent to the Modbus TCP *(see page 142)* Slave, it facilitates a read operation by the external I/O scanner.

The Modbus TCP Slave device responds to a subset of the normal Modbus commands, but does so in a way that differs from normal Modbus standards, and with the purpose of exchanging data with the external I/O scanner. The following 4 Modbus commands may be issued to the Modbus TCP Slave device:

Function Code Dec (Hex)	Function	Comment
3 (3h)	Read holding register	Allow Master IO Scanner to read %IW and %QW of the device
6 (6h)	Write single register	Allow Master IO Scanner to Write %IW of the device
16 (10h)	Write multiple registers	Allow Master IO Scanner to Write %IW of the device
23 (17h)	Read/write multiple registers	Allow Master IO Scanner to read %IW and %QW of the device and Write %IW 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 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	IW0	WORD	Input register 0
	...	...	...
	IWy	WORD	Input register y

The number of word depends on the **Holding Registers (%IW)** and **Input Registers (%QW)** parameters of the ModbusTCP tab.

NOTE: Output means OUTPUT from Master controller (= %IW for the controller).

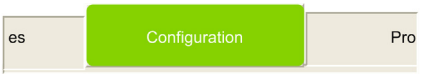
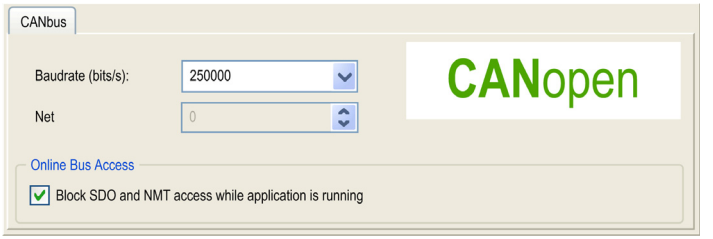
Input means INPUT from Master controller (= %QW for the controller).

CANopen Configuration

13

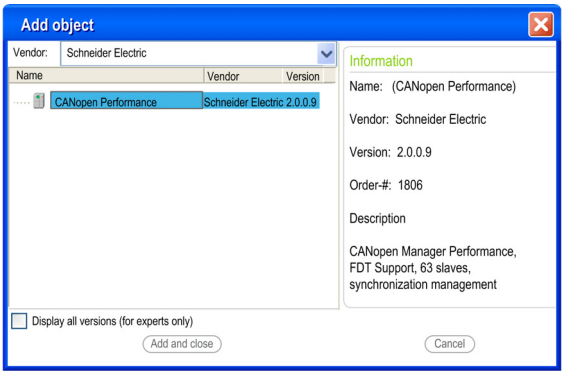
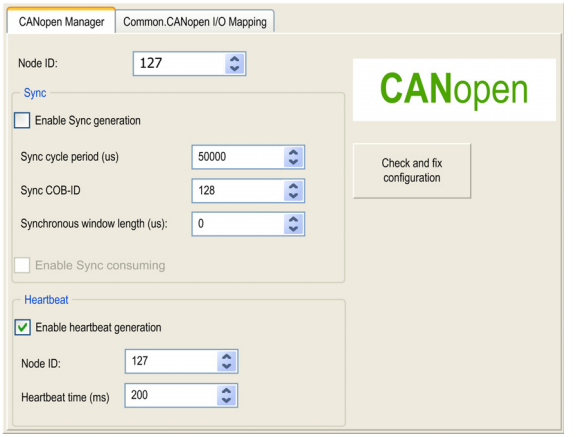
CANopen Interface Configuration

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

Step	Action
1	Select the **Configuration** tab and double-click the controller: 
2	Click the **Communication** entry on the left hand side of the screen.
3	Click the **CAN0** entry.
4	Click the **Physical Settings** entry. **Result:** The tabbed configuration dialog box for CANopen network is displayed on the right hand side of the screen.
5	Configure the baudrate (by default: 250000 bits/s):  **NOTE:** The **Online Bus Access** option allows you block SDO and NMT sending through the status screen.

CANopen Manager Creation and Configuration

To create and configure the **CANopen Manager**, proceed as follows:

Step	Action
1	Click the **Protocol Settings** entry and select **CANopen Performance**: 
2	Click the **Add and close** button. **Result:** The **CANopen Manager** configuration window appears:  **NOTE:** If **Enable Sync generation** is checked, the **CAN0_Sync** task is added. Do not delete or change the **Name**, **Type**, or **External event** attributes of **CAN0_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 generation** option on the **CANopen Manager** sub-tab of the **CANopen_Performance** tab, the **CAN0_Sync** task will automatically be deleted from your program.

Refer to the online help CoDeSys part.

Adding a CANopen Device

To add a CANopen slave device, refer to Adding Slave Devices to a Communication Manager *(see SoMachine, Programming Guide)*. Refer to the online help CoDeSys part.

CANopen Limitations

The Modicon M258 Logic Controller CANopen master has the following limitations:

Maximum number of slave devices	32
Maximum number of Received PDO (RPDO)	64
Maximum number of Transmitted PDO (TPDO)	64

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not connect more than 32 CANopen slave devices to the controller
- Program your application to use 64 or fewer Transmit PDO (TPDO)
- Program your application to use 64 or fewer Receive PDO (RPDO)

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

Serial Line Configuration

14

Introduction

This chapter describes how to configure the serial line communication of the Modicon M258 Logic Controller.

What's in this Chapter?

This chapter contains the following topics:

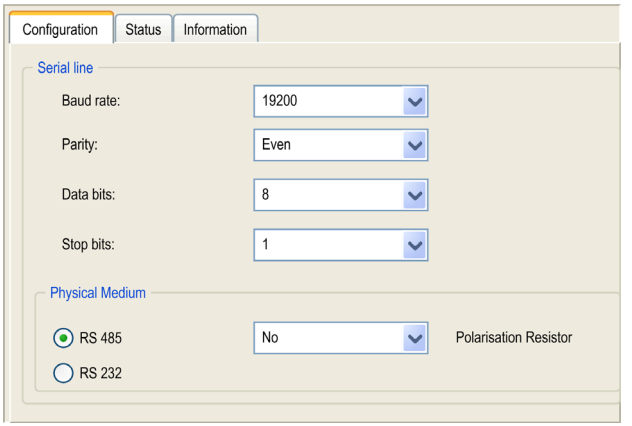
Serial Line Configuration

Introduction

The Serial Line configuration window allows to configure the physical parameters of serial line (baud rate, parity, etc...).

Serial Line Configuration

To configure the Serial Line, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Physical Settings** entry. **Result:** The configuration window is displayed. 

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 **NOTE:** Two line polarisation 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 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.

CAUTION

UNINTENDED EQUIPMENT OPERATION

Be sure 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.

The following table indicates the Managers baud rate characteristic:

Manager	Maximum Baud Rate (bits/s)
SoMachine Network Manager	115200
Modbus Manager	38400
ASCII Manager	
Modbus IOScanner	

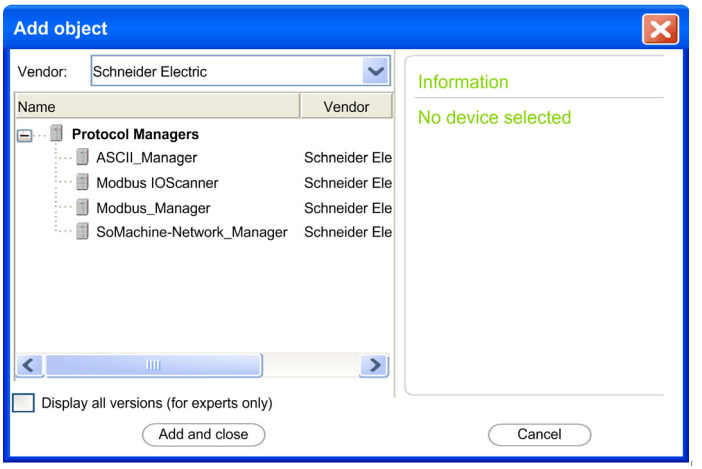
ASCII Manager

Introduction

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

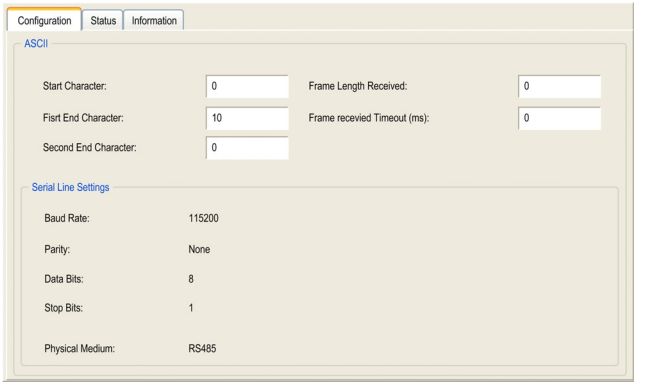
Adding the Manager

To add the Manager on Serial Line, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Protocol Settings** entry.
4	Click the **Remove/Change Protocol** button. Choose the **ASCII_Manager** object and click **Add and close**: 

ASCII Manager Configuration

To configure the ASCII Manager of your controller, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Protocol Settings** entry. **Result:** The ASCII Manager configuration window is displayed. 

Set the parameters as described in the following table:

Parameter	Description
Start Character	If 0, no start character is used in the frame. Otherwise, in **Receiving Mode** the corresponding character in ASCII is used to detect the beginning of a frame. In **Sending Mode**, this character is added at the beginning of the frame.
First End Character	If 0, no first end character is used in the frame. Otherwise, in **Receiving Mode** the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Second End Character	If 0, no second end character is used in the frame. Otherwise, in **Receiving Mode** the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Frame Length Received	If 0, this parameter is not used. This parameter allows the system to conclude an end of frame at reception, when the controller received the specified number of characters. **Note:** This parameter cannot be used simultaneously with **Frame Received Timeout (ms)**.
Frame Received Timeout (ms)	If 0, this parameter is not used. This parameter allows the system to conclude the end of frame at reception after a silence of the specified number of ms.
Serial Line Settings	Parameters specified in the Serial Line configuration window *(see page 194)*.

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 212)*.

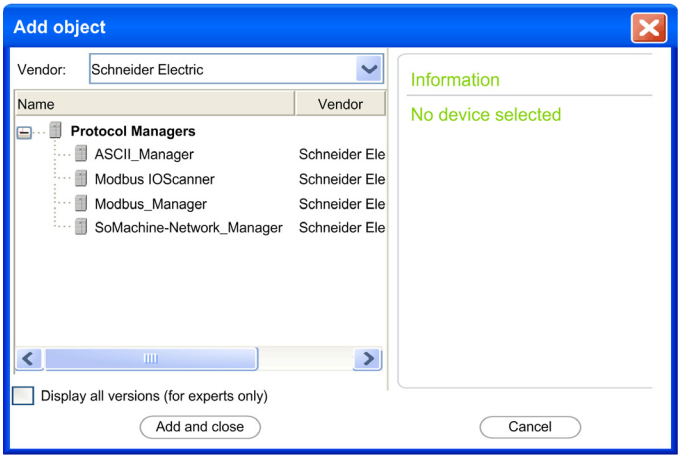
SoMachine Network Manager

Introduction

The SoMachine Network Manager must be used if you want to exchange variables with a XBTGT/XBTGK device with SoMachine software protocol, or when the Serial Line is used for SoMachine programming.

Adding the Manager

To add the Manager on Serial Line, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Protocol Settings** entry.
4	Click the **Remove/Change Protocol** button. Choose the **SoMachine-Network_Manager** object and click **Add and close**: 

Configure 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 212)*.

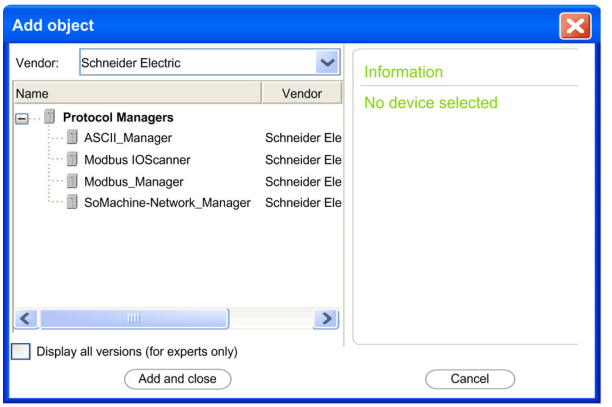
Modbus IOScanner

Introduction

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

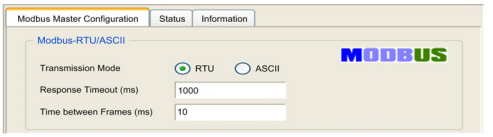
Add a Modbus IOScanner

To add a Modbus IOScanner on the Serial Line, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Protocol Settings** entry.
4	Click the **Remove/Change Protocol** button. Choose the Modbus IOScanner and click **Add and close**: 

Modbus IOScanner Configuration

To configure a Modbus IOScanner on Serial Line, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Protocol Settings** entry. **Result:**The configuration window is displayed: 

Set the parameters as described in the following 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) This parameter must be identical for each Modbus device on the link.
Response Timeout (ms)	Timeout used in the exchanges.
Time between frames (ms)	Time to help avoid bus-collision. This parameter must be identical for each Modbus device on the link.

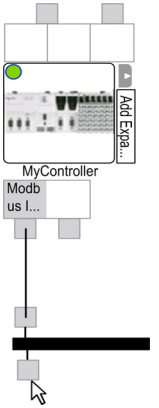
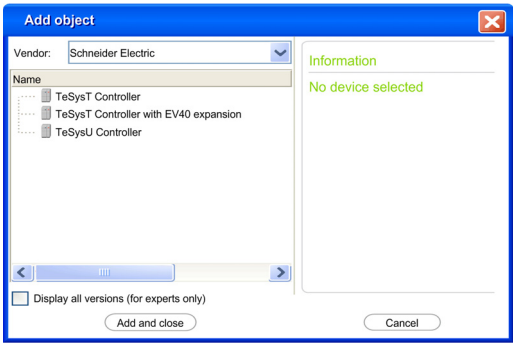
Adding a Device on the Modbus IOScanner

Introduction

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

Add a Device on the Modbus IOScanner

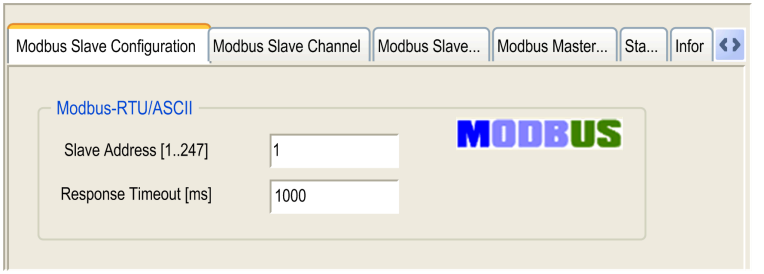
To add a device on the Modbus IOScanner, proceed as follow:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the free port of the **Modbus IOScanner** Fieldbus in the graphical configuration editor: 
3	The Add Object window appears:  Click the device to add and click the **Add and close** button.

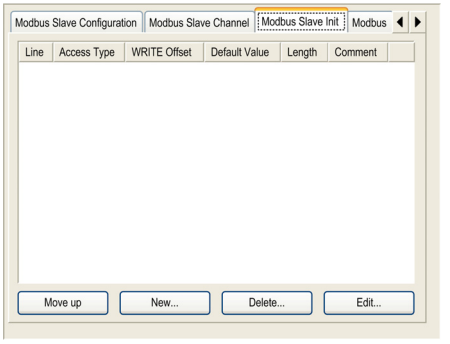
NOTE: The variable for the exchange is automatically created in the %IWx and %QWx of the **Modbus Serial Master I/O Mapping** tab.

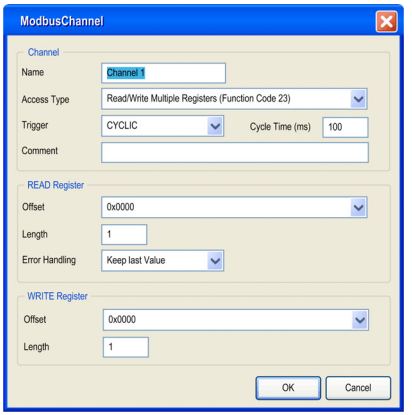
Configure a Device Added on the Modbus IOScanner

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

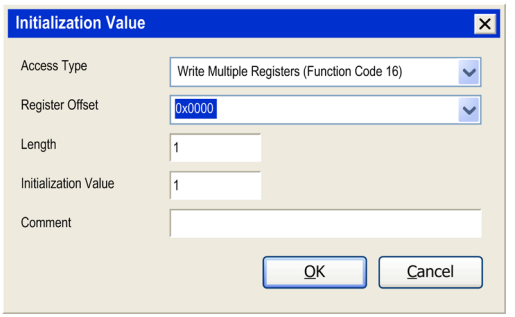
Step	Action
1	Select the **Configuration** tab.
2	In the graphical configuration editor, double-click the device. **Result:** The configuration window will be displayed. 
3	Enter a **Slave Address** value for your device (choose a value from 1 to 247).
4	Choose a value for the **Response Timeout** (in ms).

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

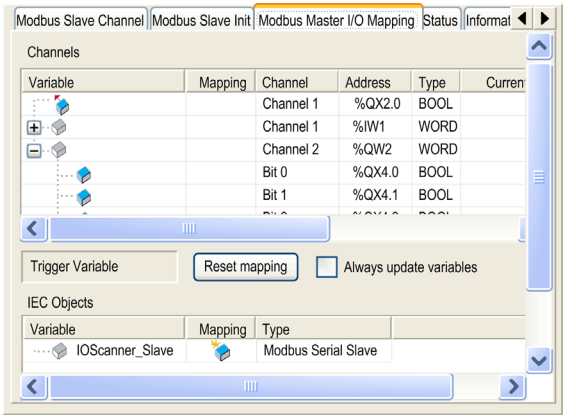
Step	Action
1	Click the **Modbus Slave Channel** tab: 

Step	Action
2	Click the **Add Channel** button: 
2	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) ● **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 a Read or a Read/Write one), you can configure the %MW to be read on the Modbus slave. Those ones 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 a Write or a Read/Write one), you can configure the %MW to be written to the Modbus slave. Those ones 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
3	Click the **Delete** button to remove a channel. Click the **Edit** button to change the parameters of a channel.
4	Click **OK** to validate the configuration of this channel.

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

Step	Action
1	Click the **Modbus Slave Init** tab: 
2	Click **New** to create a new initialization value:  The **Initialization Value** window contains the following parameters: ● **Access Type**: Choose the exchange type: Read or Write or Read/Write multiple registers (i.e. %MW) ● **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
3	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.
4	Click **OK** to create a new **Initialization 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 on the browse button **[...]** and chose a variable with the **Input Assistant**
3	For more details about I/O mapping, refer to the online help CoDeSys part.

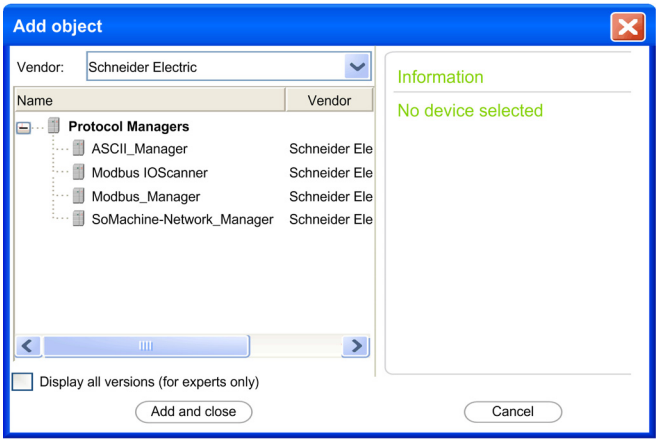
Modbus Manager

Introduction

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

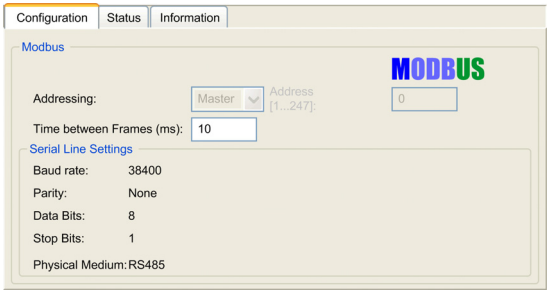
Adding the Manager

To add the Manager on Serial Line, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Protocol Settings** entry.
4	Click the **Remove/Change Protocol** button. Choose the **Modbus_Manager** object and click **Add and close**: 

Modbus Manager Configuration

To configure the Modbus Manager of your controller, proceed as follows:

Step	Action
1	Select the **Configuration** tab and double-click the controller.
2	Click the **Communication** →**Serial Line** entry on the left hand side.
3	Click the **Protocol Settings** entry. **Result:** The Modbus Manager configuration window will be displayed. 

Set the parameters as described in the following 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) This parameter must be identical for each Modbus device on the link.
Addressing	Specify the device type: ● Master ● Slave
Address	Modbus address of the device.
Time between Frames (ms)	Time to avoid bus-collision. This parameter must be 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 Block 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 Communication Functions:, PLCCommunication Library)* of the PLCCommunication Library.

Modbus Slave

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

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

The following table contains the Sub-function codes supported by the diagnostic Modbus request 08:

Sub-Function Code		Function
Dec	Hex	
10	0A	Clear Counters and Diagnostic Register
11	0B	Return Bus Message Count
12	0C	Return Bus Communication Error Count
13	0D	Return Bus Exception Error Count
14	0E	Return Slave Message Count

Sub-Function Code		Function
15	0F	Return Slave No Response Count
16	10	Return Slave NAK Count
17	11	Return Slave Busy Count
18	12	Return Bus Character Overrun Count

The table below 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: TM258LD42DT
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 and it will be possible for incorrect values to be written to memory areas responsible for output operations.

⚠ WARNING

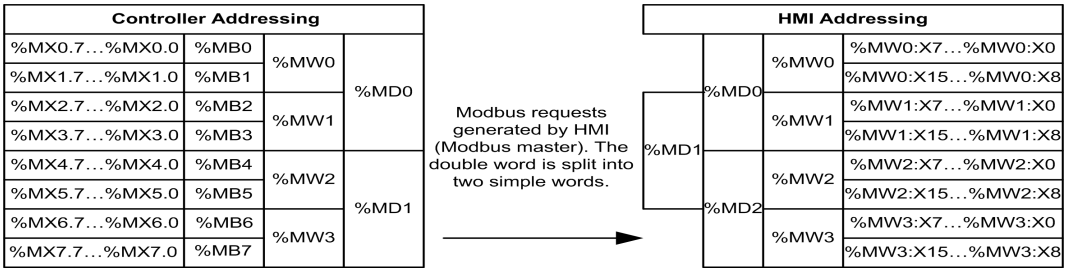
UNINTENDED EQUIPMENT OPERATION

Program your application to translate between the Modbus memory mapping used by the controller and that used by 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 212)*.

Adding a Modem to a Manager

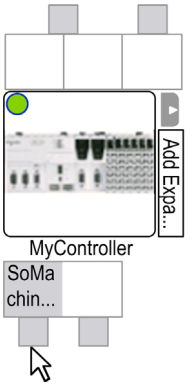
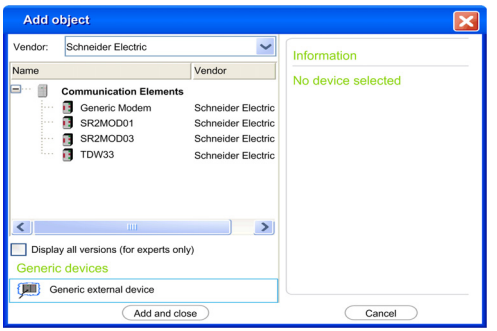
Introduction

A modem can be added to the following managers:

- ASCII Manager
- Modbus Manager
- SoMachine Network Manager

Adding a Modem to a Manager

To add a modem, proceed as follows:

Step	Action
1	Select the **Configuration** tab.
2	Click the free port of the Manager in the graphical configuration editor. 
3	The Add object window is displayed:  Click the modem to add and click the **Add and close** button.

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

Connecting a Modicon M258 Logic Controller to a PC

15

Introduction

This chapter shows how to connect a Modicon M258 Logic Controller to a PC.

What's in this Chapter?

This chapter contains the following topics:

Connecting the Controller to a PC

Overview

To transfer and run applications, you must connect your Modicon M258 Logic Controller to a computer that has SoMachine installed.

You also need a specific communication cable to connect your controller to the computer. This cable depends on the connection type you want to use:

- USB programming port: TCS XCNAMUM3P cable
- Ethernet: Ethernet cable

NOTE: Connect only one controller to a PC at a time. If you attempt to make multiple USB connections to multiple controllers from the same PC, SoMachine will recognize the first controller that it finds, and not recognize the other controllers.

USB Mini-B Port Connection

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.

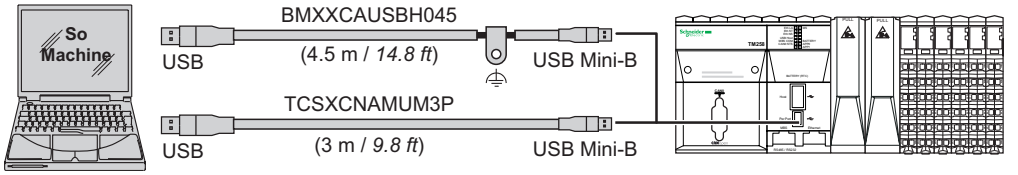
⚠ WARNING

INOPERABLE EQUIPMENT OR UNINTENDED EQUIPMENT OPERATION

- 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.

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

The following figure shows the USB connection to a PC:

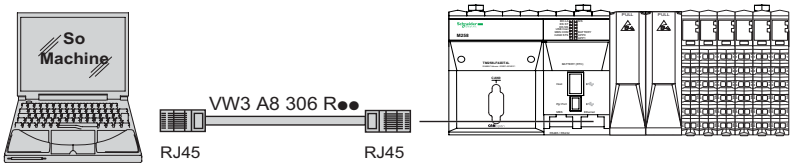


To connect the USB cable to your controller, do the following:

Step	Action
1	Connect your USB cable that allows its shield to be secured to the functional ground (FE) *(see Modicon Flexible TM5 System, System Planning and Installation Guide)* of the TM5 System.
2	Connect your USB cable to the PC on the USB port.
3	Connect your USB cable to the USB programming port on the controller.

Ethernet Port Connection

The following figure shows the Ethernet connection to a PC:



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

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

Active Path of the Controller

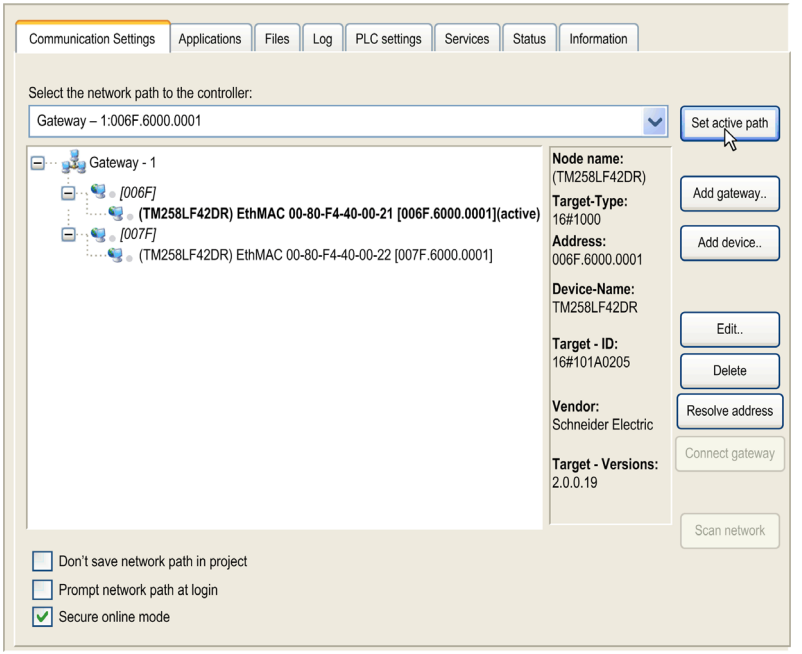
Introduction

After Connecting the controller to the PC *(see page 214)*, you must configure the Active Path of the controller in SoMachine.

NOTE: SoMachine cannot log multiple controllers simultaneously.

Active Path

To set the Active Path of the controller, proceed as follows:

Step	Action
1	In the **Configuration** tab, double-click the controller.
2	Select the **Communication Settings** tab: 
3	Click on the **Add gateway** button.
4	Click on the **Scan network** button.
5	Select the controller from the list by checking its Serial Number (the 6 last numbers on the controller) and clicking on the **Set active path** button.
6	Press ALT+F when the dialog box appears.

Transfer by USB memory Key

16

Introduction

This chapter describes how to transfer firmware, application, using an USB memory key to the Modicon M258 Logic Controller.

What's in this Chapter?

This chapter contains the following topics:

Upgrading Modicon M258 Logic Controller Firmware

Introduction

The firmware updates for Modicon M258 Logic Controller are available on the http://www.schneider-electric.com website (in .zip format).

Upgrading the firmware is possible by USB memory Key (with compatible script file).

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.

⚠ CAUTION

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 injury or equipment damage.

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

CAUTION

INOPERABLE EQUIPMENT

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

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. 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.

CAUTION
UNINTENDED EQUIPMENT OPERATION Be sure 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.**

Upgrading by Quick Key Management

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	Plug the USB memory Key into the controller. **NOTE:** The USB LED blinks green during download.
4	Power ON
5	Wait until the USB LED is solid green (the end of the download) and: ● Unplug the USB memory Key. ● The controller reboots automatically with new firmware. ● If an error is detected the USB LED is red.

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

File Transfer with USB Memory key

Introduction

The Modicon M258 Logic 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 **USB Mass Storage** editor lets you generate and copy the script and all necessary files into the USB memory key.

NOTE: The Modicon M258 Logic Controller accepts only USB key formatted in FAT or FAT32.

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

CAUTION
INOPERABLE EQUIPMENT ● Do not interrupt the transfer of the application program or a firmware update once the transfer has begun. ● Do not place the device into service until the transfer is completed successfully. **Failure to follow these instructions can result in equipment damage.**

Clone Function

The clone function lets you upload or download the application without any software and any controller competency.

NOTE: The USB memory key must be empty to perform this procedure.

Automatic upload procedure:

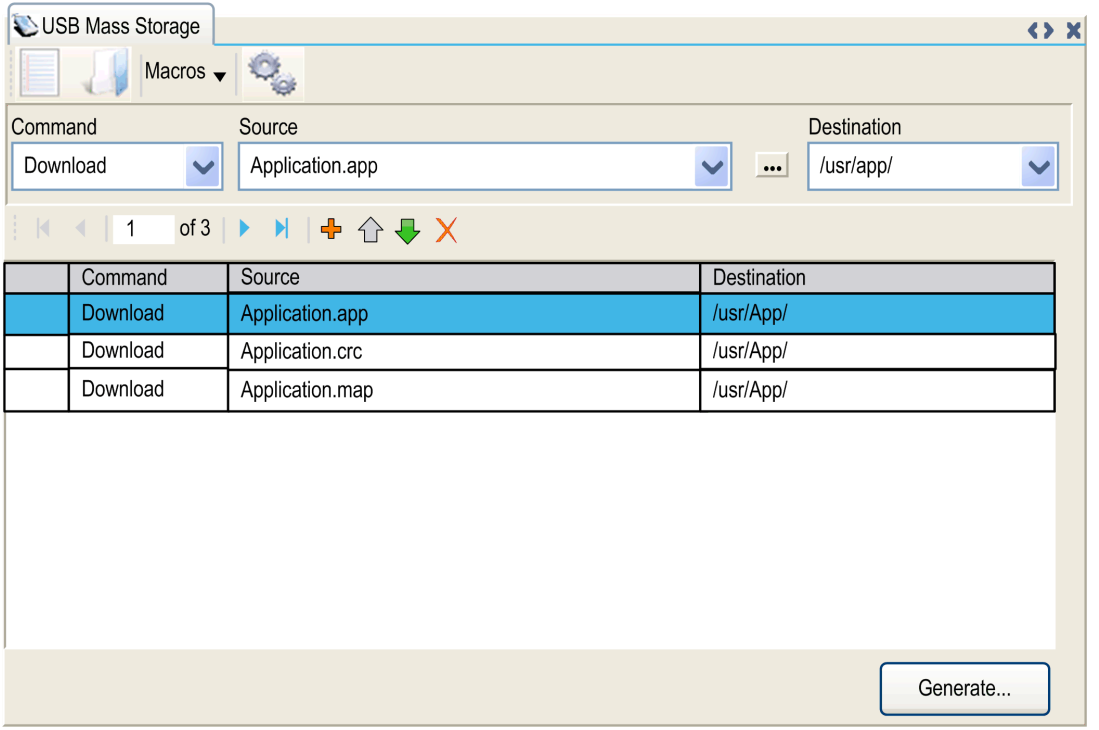
Step	Action
1	Plug the USB memory key into the controller (the controller may be in RUNNING state): The application upload is in progress. **NOTE:** The USB LED is green blinking during upload. At the end of the upload the USB LED is fixed green, if an error was detected the LED is red.
2	Unplug the USB memory key.

Automatic download procedure:

Step	Action
1	Plug the USB memory key into the controller (the controller may be in RUNNING state): The application download is in progress. **NOTE:** The USB LED is green blinking during download. At the end of the download the USB LED is fixed green, if an error was detected the LED is solid red.
2	Unplug ìthe USB memory key. **NOTE:** A reboot is required to record the new application.

Script and Files Generation with USB Mass Storage

Click on **USB Mass Storage** in the **Online** 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.
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
Download	Download a file from the USB memory key to the controller.	Select the file to download.	Select the controller destination directory.
Upload	Upload files contained in a controller directory to the USB memory key.	Select the directory.	-
Delete	Delete a controller directory.	Select the directory.	-
Reboot	Reboot the controller (only available at the end of the script).	-	-
Stop	Switch controller to STOPPED state.	-	-
Run	Switch controller to RUNNING state.	-	-

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.	
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.	
Upload Recipes	Upload the recipes from the controller to the USB memory key.	/usr/Rcp/*.rcp /usr/Rcp/*.rsi
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

Step	Action
1	Create the script with the **USB Mass Storage** 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	Unplug the USB memory key. **NOTE:** A reboot is required to record the new application.

Appendices

Functions to get/set serial line configuration in user program

A

Overview

This section describes the functions to get/set the serial line configuration in your program.

To use these functions, you must add the **M2xx Communication** library.

For further information on adding a library, refer to the SoMachine Programming Guide *(see SoMachine, Programming Guide)*.

What's in this Chapter?

This chapter contains the following topics:

GetSerialConf: Get the Serial Line Configuration

Function Description

`GetSerialConf` returns the configuration parameters for a specific serial line communication port.

Graphical Representation

Parameter Description

Input	Type	Comment
`Link`	LinkNumber	`Link` is the communication port number.
`PointerToSerialConf`	POINTER TO `SERIAL_CONF` *(see page 231)*	`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 230)* 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 on going communications with other connected devices.

⚠ WARNING

LOSS OF CONTROL DUE TO UNEXPECTED CONFIGURATION CHANGE

Be sure to 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`	`LinkNumber` is the communication port number.
`PointerToSerialConf`	POINTER TO `SERIAL_CONF` *(see page 231)*	`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

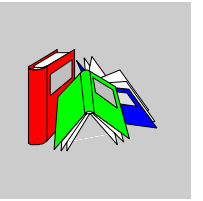
Structure Description

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

Glossary



A

application source

The *application source* file can be uploaded to the PC to reopen a SoMachine project. This source file can support a full SoMachine project (for example, one that includes HMI application).

ARP

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

ASCII

The *american standard code for information interchange* is a communication protocol for representing alphanumeric characters (letters, numbers, and certain graphic and control characters).

B

BOOTP

The *bootstrap protocol* is 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's MAC address. The server—which maintains a pre-configured table of client device MAC addresses and associated IP addresses—sends the client its pre-configured IP address. BOOTP was originally used as a method that enabled diskless hosts to be remotely booted over a network. The BOOTP process assigns an infinite lease of an IP address. The BOOTP service utilizes UDP ports 67 and 68.

C

CAN

The *controller area network* protocol (ISO 11898) for serial bus networks is designed for the interconnection of smart devices (from multiple manufacturers) in smart systems for real-time industrial applications. CAN multi-master systems ensure high data integrity through the implementation of broadcast messaging and advanced diagnostic mechanisms. Originally developed for use in automobiles, CAN is now used in a variety of industrial automation control environments.

CANmotion

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

CANopen

CANopen is an open industry-standard communication protocol and device profile specification.

CFC

The *continuous function chart* (an extension of the IEC61131-3 standard) is a graphical programming language that works like a flowchart. By adding simple logicals blocks (AND, OR, etc.), each function or function block in the program is represented in this graphical format. For each block, the inputs are on the left and the outputs on the right. Block outputs can be linked to inputs of other blocks in order to create complex expressions.

CiA

CAN in automation is a non-profit group of manufacturers and users dedicated to developing and supporting CAN-based higher layer protocols.

CIP

When the *common industrial protocol* is implemented in a network's 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.

controller

A *controller* (or “programmable logic controller,” or “programmable controller”) is used to automate industrial processes.

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

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

E

EEPROM

Electrically erasable programmable read-only memory is a type of non-volatile memory used to store data that must be saved when power is removed.

EIA rack

An *electronic industries alliance rack* is a standardized (EIA 310-D, IEC 60297 and DIN 41494 SC48D) system for mounting various electronic modules in a stack or rack that is 19 inches (482.6 mm) wide.

EtherNet/IP

The *ethernet industrial protocol* is an open communications protocol for manufacturing automation solutions in industrial systems. EtherNet/IP is in a family of networks that implements Common Industrial Protocol at its upper layers. The supporting organization (ODVA) specifies EtherNet/IP to accomplish global adaptability and media independence.

expansion bus

The *expansion bus* is an electronic communication bus between expansion modules and a CPU.

expansion I/O module

An *expansion input or output module* is either a digital or analog module that adds additional I/O to the base controller.

expert I/O

*Expert I/O*s are dedicated modules or channels for advanced features. These features are generally embedded in the module in order to not use the ressources of the PLC Controller and to allow a fast response time, depending of the feature. Regarding the function, it could be considered as a “stand alone” module, because the function is independant of the Controller processing cycle, it just exchanges some information with the Controller CPU.

F

FAST I/O

*FAST I/O*s are specific I/Os with some electrical features (response time, for example) but the treatment of these channels is done by the Controller CPU.

FB

A *function block* performs a specific automation function, such as speed control, interval control, or counting. A function block comprises configuration data and a set of operating parameters.

FBD

A *function block diagram* is a graphically oriented programming language, compliant with IEC 61131-3. It works with a list of networks whereby 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.

FG

frequency generator

firmware

The *firmware* represents the operating system on a controller.

Flash Memory

Flash memory is nonvolatile memory that can be overwritten. It is stored on a special EEPROM that can be erased and reprogrammed.

FTP

File transfer protocol is a standard network protocol (built on a client-server architecture), to exchange and manipulate files over TCP/IP based networks.

function block

See *FB*.

function block diagram

See *FBD*.

G

GVL

The *global variable list* manages global variables that are available in every application POU.

H

HSC

high-speed counter

I

ICMP

The *internet control message protocol* reports errors and provides information related to datagram processing.

IEC 61131-3

The IEC 61131-3 is an *international electrotechnical commission* standard for industrial automation equipment (like controllers). IEC 61131-3 deals with controller programming languages and defines 2 graphical and 2 textual programming language standards:

- **graphical:** ladder diagram, function block diagram
- **textual:** structured text, instruction list

IL

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

instruction list language

Refer to IL.

IP

The *internet protocol* is part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

IP 20

Ingress protection rating according to IEC 60529. IP20 modules are protected against ingress and contact of objects larger than 12.5 mm. The module is not protected against harmful ingress of water.

L

Ladder Diagram Language

See *LD*.

latching input

A *latching input* module interfaces with devices that transmit messages in short pulses. Incoming pulses are captured and recorded for later examination by the application.

LD

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

located variable

A *located variable* has an address. (See *unlocated variable*.)

M

MAC address

The *media access control address* is 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 master (MAST) task is a processor task that is run through its programming software. The MAST task has two 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.

master/slave

The single direction of control in a network that implements the master/slave model is always from a master device or process to one or more slave devices.

MIB

The *management information base* is an object database that is monitored by a network management system like SNMP. SNMP monitors devices that are defined by their MIBs. Schneider has obtained a private MIB, groupeschneider (3833).

Modbus

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

N

NEMA

The *national electrical manufacturers association* publishes standards for the performance of various classes of electrical enclosures. The NEMA standards cover corrosion resistance, ability to protect from rain and submersion, etc. For IEC member countries, the IEC 60529 standard classifies the ingress protection rating for enclosures.

network

A network includes interconnected devices that share a common data path and protocol for communications.

node

A *node* is an addressable device on a communication network.

O

ODVA

The *open deviceNet vendors association* supports the family of network technologies that are built on CIP (EtherNet/IP, DeviceNet, and CompoNet).

OS

Operating system. Can be used for Firmware that can be uploaded/downloaded by the user.

P

PDO

A *process data object* is transmitted as 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.

periodic execution

The master task is executed either cyclically or periodically. In periodic mode, you determine a specific time (period) in which the master task must be executed. If it is executed under this time, a waiting time is generated before the next cycle. If it is executed over this time, a control system indicates the overrun. If the overrun is too high, the controller is stopped.

persistent data

Value of persistent data that will be 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.

PLCopen

The PLCopen standard brings efficiency, flexibility, and manufacturer independence to the automation and control industry through the standardization of tools, libraries, and modular approaches to software programming.

post configuration

Post-configuration files contain machine-independent parameters, including:

- machine name
- device name or IP address
- Modbus serial line address
- routing table

POU

A *program organization unit* includes a variable declaration in source code and the corresponding instruction set. POUs facilitate the modular reuse of software programs, functions, and function blocks. Once declared, POUs are available to one another. SoMachine programming requires the utilization of POUs.

protocol

A *protocol* is a convention or standard that controls or enables the connection, communication, and data transfer between two computing endpoints.

PTO

Pulse train outputs are used to control for instance stepper motors in open loop.

PWM

Pulse width modulation is used for regulation processes (e.g. actuators for temperature control) where a pulse signal is modulated in its length. For these kind of signals, transistor outputs are used.

R

real-time clock (RTC)

See RTC

reflex output

In a counting mode, the high speed counter's current value is measured against its configured thresholds to determine the state of these dedicated outputs.

retained data

A *retained data* value is used in the next power-on or warm start. The value is retained even after an uncontrolled shutdown of the controller or a normal switch-off of the controller.

RFID

Radio-frequency identification is an automatic identification method that relies on the storage and remote retrieval of data using RFID tags or transponders.

RPDO

A *receive PDO* sends data to a device in a CAN-based network.

RTC

The *real-time clock* option keeps the time for a limited amount of time even when the controller is not powered.

S

scan

A controller's scanning program performs 3 basic functions: [1] It reads inputs and places these values in memory; [2] it executes the application program 1 instruction at a time and stores results in memory; [3] It uses the results to update outputs.

SDO

A *service data object* message is used by the fieldbus 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).

sequential function chart

See *SFC*.

SFC

A program written in the *sequential function chart* language can be used for processes that can be split into steps. SFC 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.)

SNMP

The *simple network management protocol* can control a network remotely by polling the devices for their status, performing security tests, and viewing information relating to data transmission. It can also be used to manage software and databases remotely. The protocol also permits active management tasks, such as modifying and applying a new configuration

Structured Text

A program written in the *structured text* (ST) language includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

symbol

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

system variable

A system variable structure provides controller data and diagnostic information and allows sending commands to the controller.

T

task

A group of sections and subroutines, executed cyclically or periodically for the MAST task, or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in consequence.

A controller can have sereval tasks.

TCP

A *transmission control protocol* is a connection-based transport layer protocol that provides a reliable simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

threshold output

Threshold outputs are controlled directly by the HSC according to the settings established during configuration.

TPDO

A *transmit PDO* reads data from a device in a CAN-based system.

U

UDP

The *user datagram protocol* is 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).

unlocated variable

An *unlocated variable* does not have an address. (See *located variable*.)

Index