EcoStruxure Machine Expert Modbus TCP

User Guide

05/2019

EIO0000003826.00

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

Use this document to configure the Modbus TCP connection of the Modicon devices.

NOTE: Read and understand this document and all related documents before installing, operating, or maintaining your controller.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.1.

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert Industrial Ethernet Overview - User Guide	*EIO0000003053 (ENG)* *EIO0000003054 (FRE)* *EIO0000003055 (GER)* *EIO0000003056 (SPA)* *EIO0000003057 (ITA)* *EIO0000003058 (CHS)* *EIO0000003816 (POR)* *EIO0000003817 (TUR)*
EcoStruxure Machine Expert EtherNet/IP - User Guide	*EIO0000003818 (ENG)* *EIO0000003819 (FRE)* *EIO0000003820 (GER)* *EIO0000003821 (SPA)* *EIO0000003822 (ITA)* *EIO0000003823 (CHS)* *EIO0000003824 (POR)* *EIO0000003825 (TUR)*
Modicon M241 Logic Controller - Programming Guide	*EIO0000003059 (ENG)* *EIO0000003060 (FRE)* *EIO0000003061 (GER)* *EIO0000003062 (SPA)* *EIO0000003063 (ITA)* *EIO0000003064 (CHS)*

Title of Documentation	Reference Number
Modicon M251 Logic Controller - Programming Guide	*EIO0000003089 (ENG)* *EIO0000003090 (FRE)* *EIO0000003091 (GER)* *EIO0000003092 (SPA)* *EIO0000003093 (ITA)* *EIO0000003094 (CHS)*
Modicon TM4 Expansion Modules - Programming Guide	*EIO0000003149 (ENG)* *EIO0000003150 (FRE)* *EIO0000003151 (GER)* *EIO0000003152 (SPA)* *EIO0000003153 (ITA)* *EIO0000003154 (CHS)*
Modicon M262 Logic/Motion Controller - Programming Guide	*EIO0000003651 (ENG)* *EIO0000003652 (FRE)* *EIO0000003653 (GER)* *EIO0000003654 (SPA)* *EIO0000003655 (ITA)* *EIO0000003656 (CHS)* *EIO0000003657 (POR)* *EIO0000003658 (TUR)*
Modicon TM3 Bus Coupler - Programming Guide	*EIO0000003643 (ENG)* *EIO0000003644 (FRE)* *EIO0000003645 (GER)* *EIO0000003646 (SPA)* *EIO0000003647 (ITA)* *EIO0000003648 (CHS)* *EIO0000003649 (POR)* *EIO0000003650 (TUR)*
Modicon TMS Expansion Modules - Programming Guide	*EIO0000003691 (ENG)* *EIO0000003692 (FRE)* *EIO0000003693 (GER)* *EIO0000003694 (SPA)* *EIO0000003695 (ITA)* *EIO0000003696 (CHS)* *EIO0000003697 (POR)* *EIO0000003698 (TUR)*
EcoStruxure Machine Expert - Programming Guide	*EIO0000002854 (ENG)* *EIO0000002855 (FRE)* *EIO0000002856 (GER)* *EIO0000002858 (SPA)* *EIO0000002857 (ITA)* *EIO0000002859 (CHS)*

Title of Documentation	Reference Number
Motion Control Library Guide	*EIO0000002221 (ENG)* *EIO0000002222 (GER)* *EIO0000002223 (CHS)*
TcpUdpCommunication Library Guide	*EIO0000002803 (ENG)* *EIO0000002804 (FRE)* *EIO0000002805 (GER)* *EIO0000002807 (SPA)* *EIO0000002806 (ITA)* *EIO0000002808 (CHS)*
Distributed Modbus TCP Logic Controller M251 - System User Guide	*EIO0000002902 (ENG)*
Compact EtherNet/IP Logic Controller M251 - System User Guide	*EIO0000002903 (ENG)*

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

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

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

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

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

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

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

Chapter 1
Modbus TCP Overview

What Is in This Chapter?

This chapter contains the following topics:

Principles

Modbus TCP Overview

The Modbus TCP protocol uses a Client/Server architecture for data exchange.

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

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

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

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

The communication between the Modbus TCP IOScanner and the slave device is accomplished using Modbus TCP channels *(see page 40)*.

Setup Procedure Overview

Overview

This document is structured in accordance with the phases of a machine life cycle.

The chapters that follow contain information and procedures to follow to set up a use case system:

- Device network configuration *(see page 19)*
- Device network commissioning *(see page 49)*
- Device network operating *(see page 55)*
- Device network diagnostics *(see page 67)*
- Device network maintenance *(see page 75)*

Chapter 2
Device Network Configuration

Overview

This chapter contains the information and procedures necessary to configure the device network.

The device network configuration is prepared in EcoStruxure Machine Expert.

At the end of this phase, you can perform the device network commissioning *(see page 49)*.

What Is in This Chapter?

This chapter contains the following sections:

Section 2.1
Network Planning

Network Planning

Purpose

A planned network helps increase the installation efficiency and decrease the installation time and costs. The preliminarily interfacing of materials (switches, cables, ports) must be designed to plan the network.

Network Design

To design and plan the Industrial Ethernet network, refer to the corresponding documentation, such as the *Media Planning and Installation Manual*, by ODVA. You can download this manual from the *ODVA website*.

Switch Types

Depending on the specific needs of your network, use the appropriate switch type:

If you need…	then plan to use…
network diagnostics and operation information	manageable switches
communication availability in case of a physical connection loss	redundant switches
long range network (fiber optic)	switch with duplex SC connector

Hubs can reduce the available bandwidth. This can result in lost requests and devices no longer being managed.

NOTICE
LOSS OF DATA Do not use a hub to set up an Industrial Ethernet network. **Failure to follow these instructions can result in equipment damage.**

For more information about switches, refer to the *Essential Guide: Networks, connectivity and Web servers*.

Cable Types

These tables present cable references that can be used in the network.

In a typical installation, you can use these cables:

Reference	Description	Details	Length
490NTW000••	Ethernet shielded cable for DTE connections	Regular cable, equipped with RJ45 connectors at each end for DTE. CE compliant	2, 5, 12, 40, or 80 m (6.56, 16.4, 39.37, 131.23, or 262.47 ft)
490NTW000••U		Regular cable, equipped with RJ45 connectors at each end for DTE. UL compliant	2, 5, 12, 40, or 80 m (6.56, 16.4, 39.37, 131.23, or 262.47 ft)
TCSECE3M3M••S4		Cable for harsh environments, equipped with RJ45 connectors at each end. CE compliant	1, 2, 3, 5, or 10 m (3.28, 6.56, 9.84, 16.4,or 32.81 ft)
TCSECU3M3M••S4		Cable for harsh environments, equipped with RJ45 connectors at each end. UL compliant	1, 2, 3, 5, or 10 m (3.28, 6.56, 9.84, 16.4, or 32.81 ft)
TCSECL1M1M••S2••		Cable for harsh environments. 2 M12 connectors. CE compliant	1, 3, 10, 25, or 40 m (3.28, 9.84, 32.8, 82.02, or 131.23 ft)
TCSECL1M3M••S2••		Cable for harsh environments. 1 M12 connector 1 RJ45 connector CE compliant	1, 3, 10, 25, or 40 m (3.28, 9.84, 32.8, 82.02, or 131.23 ft)

In fiber optic networks, you can use these cables:

Reference	Description	Details	Length
490NOC00005	Glass fiber optic cable for DTE connections	1 SC connector 1 MT-RJ connector	5 m (16.4 ft)
490NOT00005		1 ST connector (BFOC) 1 MT-RJ connector	5 m (16.4 ft)
490NOR00003		2 MT-RJ connectors	3 m (9.8 ft)
490NOR00005		2 MT-RJ connectors	5 m (16.4 ft)

Section 2.2
IP Address Assignment Strategy

What Is in This Section?

This section contains the following topics:

IP Address Assignment Strategy

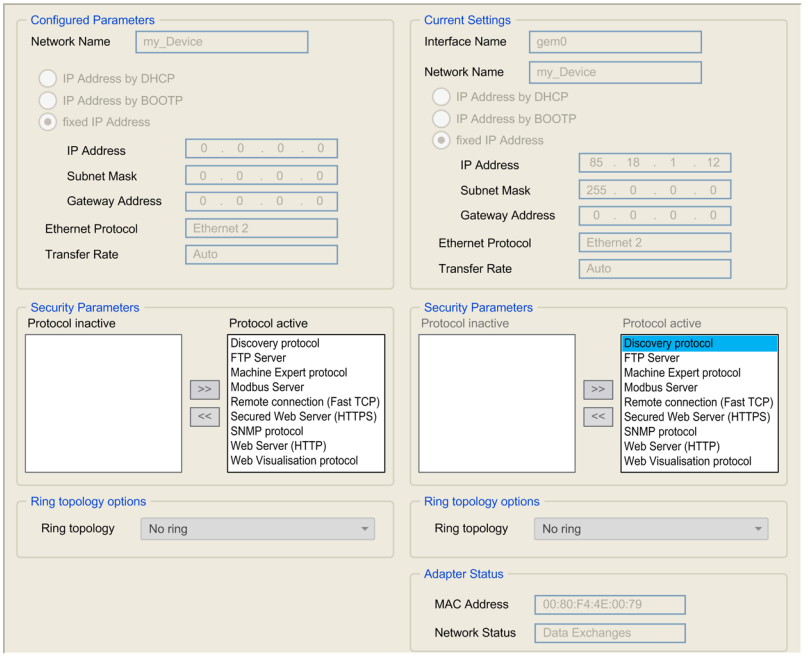
Overview

This section describes the steps to follow to implement the strategy for IP address assignment of the network devices:

- Configure the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* of the controller:
  - Network settings: IP address, subnet mask, and gateway address.
  - Choose the IP addressing method *(see page 25)* to use.
- Configure the protocol manager *(see page 26)*.

Industrial Ethernet Port Configuration

To configure the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)*, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click the Industrial Ethernet port node. The configuration tab is displayed, for example:  **Note:** If you are in online mode, you see the two windows. You cannot edit them. If you are in offline mode, you see the **Configured Parameters** and **Ring topology options** windows depending on the controller reference. You can edit them.

Step	Action
2	Select **fixed IP Address**.
3	Set the **IP Address**. This IP address is used for the network planning *(see page 30)*.
4	Set the **Subnet Mask**.
5	Verify that the **Gateway Address** is set by default to `0.0.0.0`. The gateway address allows a message to be routed to a device that is not on the local network. If there is no gateway, the gateway address is `0.0.0.0`.
6	Select the **Security Parameters** check boxes: ● **Web Server active**: used during configuration and maintenance phases ● **FTP Server active**: used by FDR service *(see page 37)*
7	Select the **DHCP Server active** check box if using a DHCP server to assign IP addresses. For more details, see IP Addressing Methods *(see page 25)*.

IP Addressing Methods

Presentation

This table presents the IP addressing methods:

Method	Description	Details
DHCP	The DHCP server uses the **DHCP device name** of the device to send it its IP address: The **DHCP device name** is also used by the FDR service.	New devices use the DHCP addressing method by default. By using DHCP, the FDR service is available. To replace a device: ● Install the new device ● Define the DHCP device name in the device ● Power up the device and launch the application. At power up, the new device is recognized and the controller loads the previously stored configuration into the new device.
BOOTP	The BOOTP server uses the **MAC Address** of the device to send its IP address:	To replace a device: ● Install the new device ● In EcoStruxure Machine Expert, enter the MAC address of the new device ● Build the application and load it in the controller. ● Configure the parameters in the device. ● Power up the device and launch the application.
Fixed	The IP address is fixed in the application.	To replace a device: ● Install the new device ● Configure in the device the network settings (IP address, subnet mask, and gateway address). ● Configure the parameters in the device directly or using EcoStruxure Machine Expert. ● Power up the device and launch the application.

Activating the DHCP Server

When using the DHCP addressing method, the DHCP server assigns IP addresses to devices upon request.

To activate the DHCP server, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* node.
2	Select the **DHCP Server active** check box. When active, devices added to the fieldbus can be configured to be identified by DHCP device name instead of by MAC address or fixed IP address.

Protocol Manager Configuration

Overview

The controller uses a protocol manager to manage the device network:

Controllers/Protocol Managers	Industrial Ethernet Manager	Ethernet/IP Scanner	Modbus TCP IO Scanner	Sercos Master
M241	✓	–	–	–
M251	✓	–	–	–
M262	–	✓	✓	✓(1)
(1) On Ethernet_1 on TM262M•				

Protocol Manager Settings for M241/M251 Controllers

To configure the protocol manager, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **Industrial_Ethernet_Manager**. **NOTE:** The **Network Settings** are automatically generated in accordance with the Industrial Ethernet port network settings *(see page 23)*.
2	Select the **Preferred protocol Modbus TCP**. Your selection becomes the device protocol set by default for each device declaration *(see page 27)*.

NOTE: When the Modbus TCP IOScanner is configured, the post configuration file for the Industrial Ethernet network is ignored.

Protocol Manager Settings for M262 Controllers

To see the configuration of the protocol manager, in the **Devices tree**, double-click **Modbus_TCP_IO_Scanner**.

NOTE: The settings are automatically generated in accordance with the Industrial Ethernet port network settings *(see page 23)*.

Section 2.3
Network Device Declaration

Network Device Declaration

Overview

This section describes how to add a device on the protocol manager node.

The available Schneider Electric devices, as well as devices supplied with EDS files, are listed in the **Hardware Catalog**. These devices are supplied with predefined connection configurations *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)*. For other devices not listed in the catalog, use **Generic slave device**.

Automatic Settings

During each device declaration, EcoStruxure Machine Expert automatically:

- Sets the network settings (IP address, subnet mask, gateway address) in accordance with the Industrial Ethernet scanner settings.
- Sets a unique DHCP device name, normally compatible with the internal rules of the device (each **DHCP device name** must be unique).
- Creates predefined data exchanges for predefined devices.

NOTE: If the proposed **DHCP device name** is not compatible with the device, you may edit it.

Add a Device

To add a device on the protocol manager node, select the device in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* node.

Once a device is added, it appears in the **Network Manager** or **Ethernet Services** tab. Refer to Adapting Network Planning and Device Identification *(see page 30)*.

When you use the Drag-and-Drop method, the devices are defined with the preferred protocol, if possible.

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

• Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Add a Device with a Protocol Other Than the Preferred Protocol

When you use the Drag-and-Drop method:

- If the device cannot be defined with the preferred protocol, the default supported protocol of the device is used instead.
- If the preferred protocol is not set, a list appears to select the protocol to use.

To add a slave device with a protocol other than the preferred protocol, refer to Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*.

For example, when you add an OTB1EODM9LP device, it is configured with Modbus TCP even if the preferred protocol is set to EtherNet/IP.

Add Device from Template

For devices that do not have key features but support TVDA *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)*, it is possible to declare them using a template. This imports additional elements to facilitate program writing.

Use this method for OsiSense XGCS, XUW, and Preventa XPSMCM devices.

To add a device from a template to the protocol manager, proceed as follows:

Step	Action
1	In the **Hardware Catalog**, select the **Device Template** check box.
2	Select the device in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)*.

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

• Using the Drag-and-drop Method *(see EcoStruxure Machine Expert, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see EcoStruxure Machine Expert, Programming Guide)*

Add a TCP/UDP Device

To add a TCP/UDP device on the protocol manager node, select **Generic TCP/UDP equipment** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* node.

Section 2.4
Adapting Network Planning and Device Identification

What Is in This Section?

This section contains the following topics:

Adapting Network Planning and Device Identification

Overview

When devices are added in the protocol manager, use the **Network Manager** or **Ethernet Services** tab to edit the network planning.

Editing the Network Planning

In the **Devices tree**, double-click the **Industrial_Ethernet_Manager** node.

If you use a M262 controller, double-click the controller node in the **Devices tree → Ethernet Services**.

The **Network Manager** or **Ethernet Services** tab shows the devices defined on the device network:

Column	Use	Comment
Device Name	Click to open the device settings	Name of the device. A name is given by default. To rename your device, type a name in the **Name** box. Do not use spaces within the name. Do not use an underscore (“_”) at the end of the name. Give the device a meaningful name to facilitate the organization of your project.
Device Type	-	Device type
IP Address	Modify the IP address	An IP address is shown as invalid if it has already been assigned to another device that uses the same protocol and DHCP address assignment. If the IP address is invalid, the icon  is displayed.
MAC Address	Enter the MAC address	Used to retrieve an IP address through BOOTP. Each IP address must be unique for a particular protocol and for DHCP/BOOTP. For example, you can add the same device for both Modbus TCP and Ethernet/IP protocols, but if you use BOOTP or DHCP to obtain an IP address for one of the protocols, you must enter that same IP address for the other protocol as a **Fixed** IP address.
Device name	Modify the DHCP device name	Used as device name to retrieve an IP address through DHCP, maximum 16 characters. The **DHCP device name** must be the same as that defined in the device. Each **DHCP device name** must be unique. The default **DHCP device name** is normally compatible with the internal rules of the device. For details on **DHCP device name** internal rules of the device, refer to the documentation of the device. **NOTE:** If the proposed **DHCP device name** is not compatible with the device, you may edit it.

Column	Use	Comment
Identified by	Modify the IP addressing method: • **DHCP** • **BOOTP** • **Fixed**	**DHCP**: The **DHCP device name** must be the same as defined in the device. This method is mandatory for the FDR service.
		BOOTP: The **MAC Address** of the device must be entered.
		Fixed: The **IP Address** must be the same as defined in the device.
Protocol	–	Protocol used
Subnet Mask	Modify the subnet mask	Click **Expert Mode** to hide/show the column.
Gateway Address	Modify the gateway address	Click **Expert Mode** to hide/show the column. For operational details, refer to Out of Process Data Exchanges *(see page 61)*.
Identification Mode	–	**IP Address**
Operation Mode	–	–

The modifications made in this tab are applied in the **Modbus TCP settings** tab *(see page 33)*.

IP Addressing Methods

By default, the added devices use DHCP.

This table presents the IP addressing methods:

Method	Description	Details
DHCP	The DHCP server uses the **DHCP device name** of the device to send it its IP address: The **DHCP device name** is also used by the FDR service.	By using DHCP, the FDR service is available. To replace a device, you have to: • Install the new device • Define the DHCP device name in the device • Power up the device and launch the application. At power-up, the new device is recognized and the controller loads the previously stored configuration into the new device.
BOOTP	The BOOTP server uses the **MAC Address** of the device to send its IP address:	To replace a device, you have to: • Install the new device • In EcoStruxure Machine Expert, enter the MAC address of the new device • Build the application and load it in the controller. • Configure the parameters in the device. • Power up the device and launch the application.

Method	Description	Details
Fixed	The IP address is fixed in the application.	To replace a device, you have to: ● Install the new device ● Configure in the device the network settings (IP address, subnet mask, and gateway address). ● Configure the parameters in the device directly or using EcoStruxure Machine Expert. ● Power up the device and launch the application.

Reinitialize IP Address Plan

Click **Regenerate IP address** to reinitialize the IP address plan associated with the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* (for example, after a change of IP address on the Industrial Ethernet port).

EcoStruxure Machine Expert reads the IP address configured on the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* and assigns the next available IP addresses to the devices. For example, if the IP address configured on the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* is `192.168.0.11`, the IP addresses attributed to the devices are `192.168.0.12`, `192.168.0.13`, and so on.

Out of Process Data Exchanges

Out of process data exchanges are often data exchanges between the control network and the device network. For example, you may use a supervision software or a third-party configuration tool to communicate with a target on the device network.

For more operational details, refer to Out of Process Data Exchanges *(see page 61)*.

If you need an out of process data exchange, set the correct device gateway address parameter.

The gateway address parameter of the network devices must be set to the IP address of the Industrial Ethernet port *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)* of the controller.

A configuration tool must be able to communicate with the network devices in order to set their parameters.

If the configuration tool...	Then...
is connected on the control network	update the network device gateway parameter, see below.
is connected on the device network	the gateway parameter is not used.
uses a protocol other than TCP/IP	the gateway parameter is not used.

To configure the gateway parameter in the network device, refer to the documentation of the device.

NOTE: If the DHCP service is used to address the network devices, the gateway parameter is set in the controller network tab *(see page 30)*.

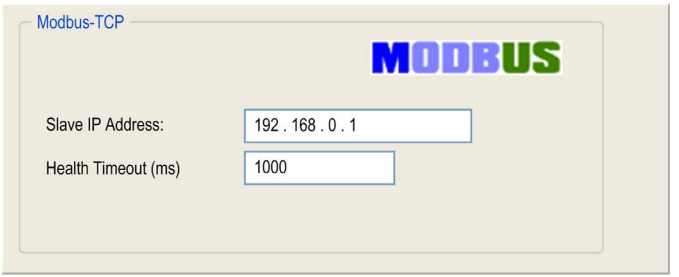
Modbus TCP Settings

Overview

Once the devices are added in the protocol manager, use its **Network manager** or **Ethernet Services** tab to edit the network planning.

Modbus TCP Settings

To configure pre-defined slave devices added on the Modbus TCP IOScanner, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click a Modbus TCP slave device node. **Result:** The configuration window is displayed: 
2	Enter a **Slave IP Address** value. The **Address settings** values are the same as those defined in the protocol manager *(see page 30)*.
3	Enter a **Health Timeout (ms)** value (by default 1000). This represents the delay (in ms) between a request of the Modbus TCP IOScanner and a response from the slave. When the health timeout expires, the associated health bit values change to 0. Health bit values can be visualized in the **IOScanner I/O Mapping** tab *(see page 43)* or through the Web server. The health timeout applies to the channels of the slave device.
4	For devices with advanced settings, some additional settings can be required: ● **Repetition rate (ms)**: Time value expressed in ms. This represents the delay between two sendings of a request. This value must be lower than the **Health Timeout (ms)**. ● **Unit ID**: unit ID of the Modbus TCP slave device (by default 255). Refer to the Device Type Manager User Guide.

Section 2.5
Network Device Configuration

Network Device Configuration

Overview

Once the network devices are defined on the device network, you can configure them with:
- DTM
- Specific editors
- Third-party tools

Description	Advantages
DTM	Can manage complex configurations.
Specific editors	Good transparency. Specifically designed for EcoStruxure Machine Expert.
Third-party tools	Tools specifically designed for the device.

Devices with DTM

Some devices have a DTM. Refer to supported devices *(see EcoStruxure Machine Expert Industrial Ethernet Overview, User Guide)*.

The DTM allows you to modify the parameters of the device.

To configure a device with its DTM, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click the device.
	Click **Start offline**.
2	Select the **Configuration** tab.
3	Click **OK**. **Result:** The content of the tab is updated by the DTM.
4	Modify the device configuration. For more information, refer to Device Type Manager User Guide.

NOTE: The use of a DTM may require a particular routing and IP forwarding *(see EcoStruxure Machine Expert, Device Type Manager (DTM), User Guide)* configuration on the controller.

Devices with Plugins

Depending on the plugin, the User Parameters may not be available. In that case, it is up to the plugin to manage the configuration of the device.

Example: Advantys OTB1EODM9LP

The Advantys OTB1EODM9LP is supported in EcoStruxure Machine Expert by a library. One function block is a configuration function block that allows you to send the configuration to the device. For more details, refer to the Distributed Modbus TCP Logic Controller M251 System User Guide.

To configure an OTB1EODM9LP, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click the OTB1EODM9LP node.
2	Configure the I/Os of the Advantys OTB device in the **OTB I/O Configuration** tab.
3	Add and configure any TM2 expansion modules attached to the OTB.
4	Stop the communication using the `IOS_STOP` function.
5	Call a `CONFIGURE_OTB` function block to update the Advantys OTB configuration with the data created on the previous steps.
6	Restart the communication using the `IOS_START` function.

NOTE: The expert functions of the Advantys OTB such as counters, fast counters, and pulse generators, cannot be directly used in the Industrial Ethernet scanner.

Specific Editors

The specific editors allow you to configure the TM2 and TM3 expansion modules on a TM3 Ethernet bus coupler. The configuration applies to the modules automatically after download.

Third-party Tools

Some devices are configured outside of EcoStruxure Machine Expert (specific software, keypad, Web server, and so on).

For more details, refer to the documentation of the device.

Master IP Address Parameter

Some devices have a **Master IP address** parameter so that only one, declared Master, controller has access to the device.

If the device...	Then...
is configured to use the protocol manager	configure the **Master IP address** parameter inside the device, see below.
is not configured to use the protocol manager	use `0.0.0.0` for the **Master IP address** parameter in the device.

The **Master IP address** parameter of the device has to be set to the IP address of the controller supporting the protocol manager.

To configure this parameter in the device, refer to the documentation of the device.

Section 2.6
Network Device Replacement

Device Replacement with FDR

FDR Overview

Some devices support the Fast Device Replacement (FDR) service.

The FDR service stores network and operating parameters of devices on the network. If a device is replaced, the service automatically configures the replacement device with parameters identical to those of the removed device.

In order to configure this service in the device, refer to the documentation of the device.

The FDR server relies on the following advanced services embedded in the controller (depending on the reference):

- DHCP server for device address assignment
- FTP server for device parameter files. This optional service is used only by devices that contain parameters.
- TFTP server for device parameter files. This optional service is used only by devices that contain parameters.

The DHCP server allows the configuration of the new device with the same addressing parameters.

Devices that contain parameters use the FTP or TFTP server to save their parameter files.

The replacing device requests the FTP or TFTP server to restore the parameter files.

Section 2.7
Cyclic Data Exchanges Configuration

What Is in This Section?

This section contains the following topics:

Cyclic Data Exchanges Overview

Overview

The protocol manager supports cyclic data exchanges between the controller and the slave devices.

The cyclic data exchange requests are supported by a channel for Modbus TCP.

Predefined devices have predefined data exchanges, for which the cyclic data exchanges are automatically defined. To configure the Generic devices, you must add the channel in **Modbus TCP Channel Configuration**.

If necessary, you can configure these data exchanges using the dedicated DTM or the appropriate third-party tool. For details, refer to the documentation of the device.

You can add and configure new requests for these devices and generic slave devices.

For all data exchanges, you can map variables to be used by the program.

Modbus TCP Cyclic Data Exchanges Configuration

Overview

To configure the Modbus TCP cyclic data exchanges, you have to:

- Configure for each Modbus TCP slave devices the data exchanges request (on channels) and the I/O Mapping.
- Configure the I/O scanner for Modbus TCP slave devices.

Modbus TCP Channel

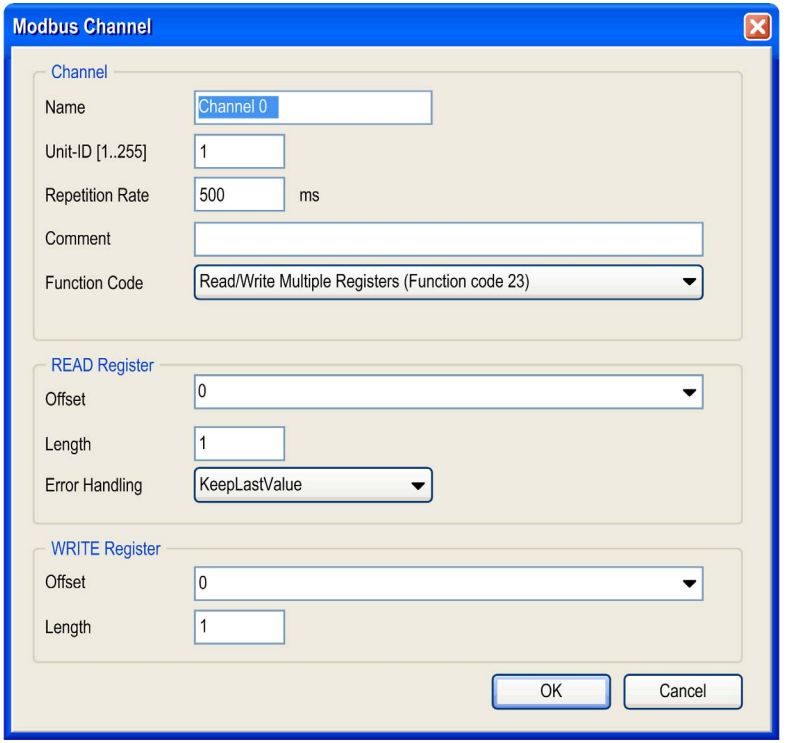
A Modbus channel carries a Modbus request between the master and a slave.

For a generic slave device, you can use multiple channels. To send several different requests to a device, create several channels.

Configure the Modbus TCP Slave Device Channels

To configure the data exchanges (on channels) of a Modbus TCP slave device, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click a Modbus TCP slave device. **Result**: Its configuration window is displayed.
2	Click the **Modbus TCP Channel Configuration** tab: 
3	To remove a non-predefined channel, select it and click **Delete**.
4	To change the parameters of a channel, select the channel and click **Edit**. **NOTE:** For the devices that provide predefined channels, only the **Repetition Rate** value can be modified.

Step	Action
5	To add a channel, click **Add Channel**. This dialog box is displayed: 

Step	Action
6	In the **Channel** area, you can define: ● **Name**: optional string for naming the channel. ● **Unit-ID [1..255]**: unit ID (1) of the Modbus TCP slave device (by default 255). ● **Repetition Rate**: polling interval of the Modbus request (by default 20 ms). ● **Comment**: optional field to describe the channel. ● **Function Code**: type of Modbus request: ❍ **Read/Write Multiple Registers (Function code 23)** (by default). ❍ **Read Holding Registers (Function code 03)**. ❍ **Write Multiple Registers (Function code 16)**. In the **READ register** area, you can define: ● **Offset**: starting register number to read from 0 to 65535. ● **Length**: number of the registers to be read (depending on the function code). ● **Error Handling**: define the fallback value in the case of a communication interruption: ❍ **Keep Last Value** (by default) holds the last valid value. ❍ **SetToZero** resets the values to 0. In the **WRITE register** area, you can define: ● **Offset**: starting register number to write from 0 to 65535. ● **Length**: number of the registers to be written (depending on the function code).
7	Click **OK** to validate the configuration of the channel.
8	Repeat steps 5 to 7 to create other channels that define the Modbus communication with the device. For each Modbus request, you must create a channel.

(1) Unit identifier is used with Modbus TCP devices which are composed of several Modbus devices, for example, on Modbus TCP to Modbus RTU gateways. In such case, the unit identifier allows reaching the slave address of the device behind the gateway. By default, Modbus/TCP-capable devices ignore the unit identifier parameter.

Read/Write Register Length

The read/write register length depends on the Modbus function code.

This table contains, for 1 channel, the maximum length of the read/write registers:

Modbus function code	Maximum length	
	Read register	Write register
Read/write multiple registers (function code 23)	125	121
Read registers (function code 03)	125	-
Write registers (function code 16)	-	123

NOTE: Due to these limitations and the maximum input/output words of the scanner (2048), verify the scanner resources overload *(see page 45)*.

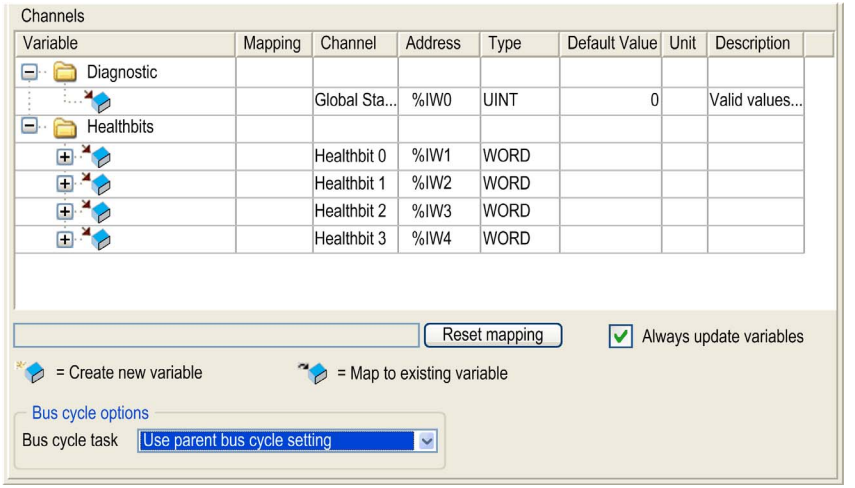
Modbus TCP I/O Mapping

Prerequisites

A Modbus TCP channel must exist.

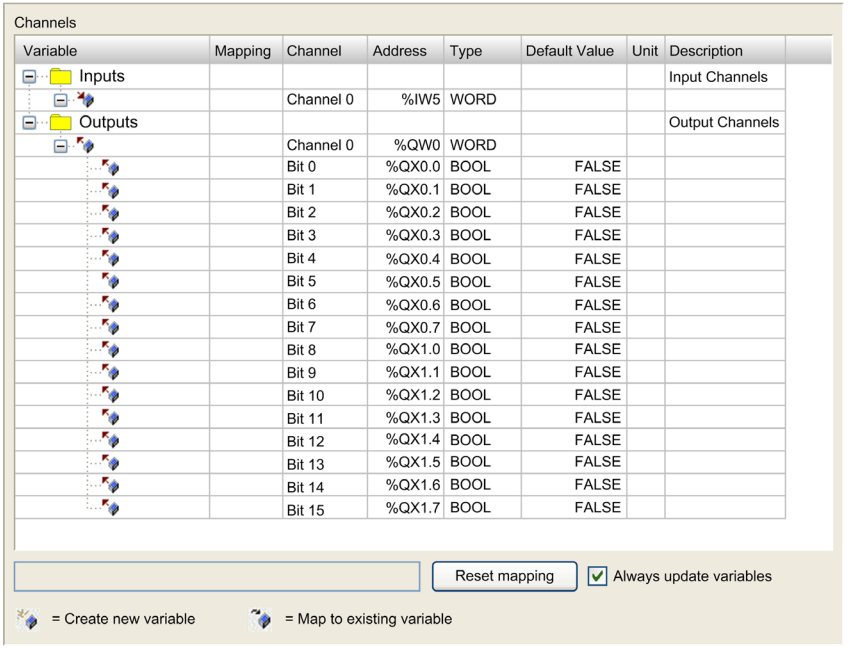
Configure the Modbus TCP IOScanner

To configure a Modbus TCP IOScanner, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click the protocol manager. **Result**: The configuration window is displayed.
2	Select the **IOScanner I/O Mapping** tab: 
3	Select the **Bus cycle task** in the list: • **Use parent bus cycle setting** (by default), • **MAST** **NOTE:** The **Bus cycle task** parameter inside the I/O mapping editor of the device that contains the Modbus TCP IOScanner defines the task responsible for the refresh of the I/O images (%QW, %IW). These I/O images correspond to the Modbus request sent to the Modbus slaves and the health bits.
4	Double-click in a cell of the **Variable** column to open a text field. Enter the name of a variable or click the browse button **[...]** and chose a variable with the **Input Assistant**.

Configure a Modbus TCP Slave Device I/O Mapping

To configure a Modbus TCP slave device I/O Mapping, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click a Modbus TCP slave device. **Result**: Its configuration window is displayed.
2	Select the **ModbusTCPSlave I/O Mapping** tab.  Channels Reset mapping Always update variables = Create new variable = Map to existing variable
3	Select the **Bus cycle task** in the list: • **Use parent bus cycle setting** (by default), • **MAST** **NOTE:** The **Bus cycle task** parameter inside the I/O mapping editor of the device that contains the Modbus TCP IOScanner defines the task responsible for the refresh of the I/O images (%QW, %IW). These I/O images correspond to the Modbus request sent to the Modbus slaves and the health bits.
4	Double-click in a cell of the **Variable** column to open a text field. Enter the name of a variable or click the browse button **[...]** and chose a variable with the **Input Assistant**.

Variable	Mapping	Channel	Address	Type	Default Value	Unit	Description
Inputs							Input Channels
		Channel 0	%IW5	WORD			
Outputs							Output Channels
		Channel 0	%QW0	WORD			
		Bit 0	%QX0.0	BOOL	FALSE		
		Bit 1	%QX0.1	BOOL	FALSE		
		Bit 2	%QX0.2	BOOL	FALSE		
		Bit 3	%QX0.3	BOOL	FALSE		
		Bit 4	%QX0.4	BOOL	FALSE		
		Bit 5	%QX0.5	BOOL	FALSE		
		Bit 6	%QX0.6	BOOL	FALSE		
		Bit 7	%QX0.7	BOOL	FALSE		
		Bit 8	%QX1.0	BOOL	FALSE		
		Bit 9	%QX1.1	BOOL	FALSE		
		Bit 10	%QX1.2	BOOL	FALSE		
		Bit 11	%QX1.3	BOOL	FALSE		
		Bit 12	%QX1.4	BOOL	FALSE		
		Bit 13	%QX1.5	BOOL	FALSE		
		Bit 14	%QX1.6	BOOL	FALSE		
		Bit 15	%QX1.7	BOOL	FALSE		

Protocol Manager Load Verification

Purpose

If the load on the protocol manager exceeds 100%, cyclic data exchanges might not be processed at the configured rate.

The **Ethernet Resources** tab allows you to estimate the load on the protocol manager.

Verify this load before operating the machine.

To manage the load, you can manipulate one or more of the following load factors:

- Number of slaves
- With Modbus TCP:
  - Number of channels (on the Modbus TCP IOScanner)
  - The repetition rate of the channels

Load Estimation

This equation allows estimation of the load on the protocol manager if it manages exclusively Modbus TCP IOScanner devices:

$$\text{IOScanner load (\%)} = \sum_{\text{channel}=1}^{\text{Nb Channels}} \frac{50}{\text{Repetition Rate}_{\text{channel}}}$$

This equation allows estimation of the load on the protocol manager of the TM262L10MESE8T and TM262M15MESS8T if it manages EtherNet/IP or Modbus TCP IOScanner device:

$$\sum_{TCPch=1}^{NbTcpChannels} 25/RepetitiveRate(TCPch) + \sum_{EIPch=1}^{NbEIPChannels} load/RPI\ (EIPch)$$

$$if\ RPI(EIPch) < 5\ then\ load = 100,\ else\ load = 62.5$$

This equation allows estimation of the load on the protocol manager of the TM262L20MESE8T, TM262M25MESS8T and TM262M35MESS8T if it manages EtherNet/IP or Modbus TCP IOScanner device:

$$\sum_{TCPch=1}^{NbTcpChannels} 15/RepetitiveRate(TCPch) + \sum_{EIPch=1}^{NbEIPChannels} load/RPI\ (EIPch)$$

$$if\ RPI(EIPch) < 3\ then\ load = 50,\ else\ load = 32$$

NOTE: If you use Sercos communication, the resources are not calculated.

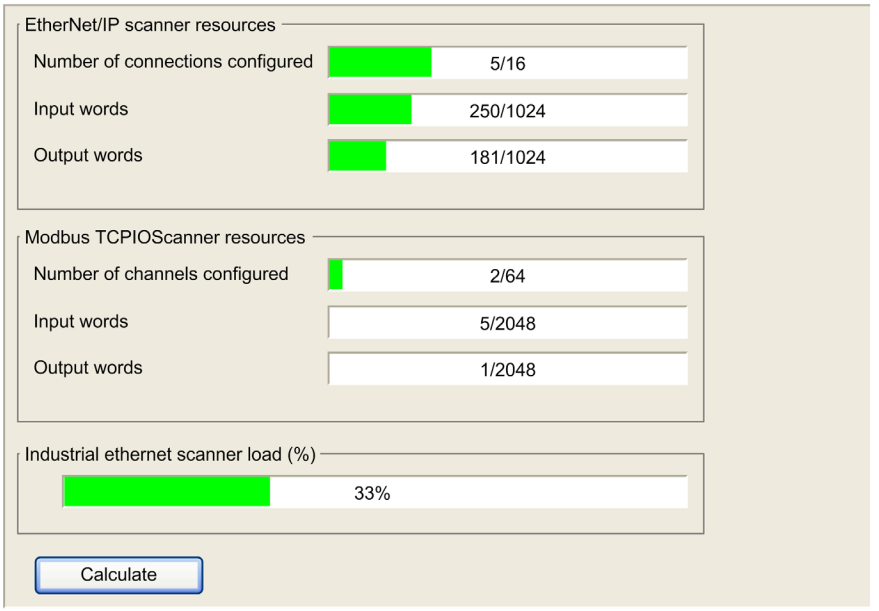
This load estimate does not take into account increases in load resulting from out of process data exchanges *(see page 61)* such as:

- DTM, Web server, and Modbus TCP requests.
- fieldbus communications (DTM, Web server communications when the PC is on the fieldbus)
- TCP UDP communications generated by the TcpUdpCommunications library.

In EcoStruxure Machine Expert, an automatic load calculation is available:

Step	Action
1	In the **Devices tree**, double-click the protocol manager node.
2	If you use a M262 controller, select **Ethernet Services → Ethernet Resources**.
3	Click **Calculate**.

This picture presents the **Ethernet Resources** tab:



Section 2.8
Programming Over Industrial Ethernet

Programming Over Industrial Ethernet

Overview

When the protocol manager is added, the Modbus TCP IOScanner library is automatically instantiated.

In addition, most Industrial Ethernet slave devices have a dedicated library containing function and function blocks.

Use these elements to facilitate the program writing.

EcoStruxure Machine Expert contains TVDA templates that can be used.

Manage the Operating Modes of the Devices

The Modbus TCP IOScanner library contains these functions:

- IOS_GETSTATE: Read the state of the Modbus TCP IOScanner
- IOS_START: Start the Modbus TCP IOScanner
- IOS_GETHEALTH: Read the Health Bit value
- IOS_STOP: Stop the Modbus TCP IOScanner
- CONFIGURE_OTB: Send the software configuration of the Advantys OTB

For more details, refer to Modbus TCP IOScanner Library *(see page 79)*.

For operational details, refer to Mastering slave devices operating modes *(see page 56)* and Impact of the controller States on the Industrial Ethernet *(see page 63)*.

Send Commands and Read Status from Devices

Cyclic data exchanges are used with generic devices that require deterministic data exchanges. Cyclic data exchanges are managed by the protocol manager.

In addition, you can send explicit messages.

For Modbus TCP devices, you can use READ_VAR and WRITE_VAR.

For operational details, refer to Slave devices configuration on start *(see page 60)* and Data exchanges on demand *(see page 58)*.

Use TVDA Templates

Most Industrial Ethernet slave devices are parts of TVDA.

EcoStruxure Machine Expert proposes to add a device from a template *(see page 28)*.

By this, the device is added with several already parametrized blocks and/or function blocks.

Chapter 3
Device Network Commissioning

Overview

This chapter describes how to perform the commissioning of your Industrial Ethernet network.

This phase follows the device network configuration *(see page 19)*.

At the end of this phase, the application can be started *(see page 55)*.

What Is in This Chapter?

This chapter contains the following topics:

Commissioning

Overview

During the commissioning, you have to:

- Perform the machine (controller and slave devices) first power-up.
- Perform network tests.
- Download the configuration to the network devices.
- Adjust controller and network devices configuration (online or directly on the devices).
- Complete the FDR on each available device.
- Back up your application.

First Machine Power-up

To realize the first power-up, proceed as follow:

Step	Action
1	Transfer the application to the controller. Refer to Downloading an Application *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Prepare each device to be recognized on the device network by referring to the network planning *(see page 30)*: BOOTP, DHCP, fixed IP, network name. For details, refer to Prepare the Device to Be Recognized *(see page 52)*.
3	Perform a machine power cycle. This may be necessary for some devices to acquire the correct network settings.
4	Perform network tests *(see page 68)*.

Download the Configuration to the Network Devices

Refer to Apply the Correct Device Configuration *(see page 54)*.

Adjust Controller and Devices Application

Once the first machine power-up performed and the configuration downloaded to the devices, you can adjust the system with:

- Embedded DTMs online modification, such as:
  - parameters adjustment,
  - autotuning for performances and energy efficiency,
  - oscilloscope for fine dynamic tuning
  - …
- For devices not having DTM, manual adjustment directly done on the devices. Refer to the documentation of the device.

Complete the FDR Service

Once the system is configured, you have to complete the FDR service. This step consists in saving the device configuration in the controller FTP server.

Depending on the device, several tools can be used:

- EcoStruxure Machine Expert,
- Third-party tools (for example: SoMove),
- Device Web server,
- Directly on the device (with embedded HMI),
- …

For more details, refer to the documentation of the device.

Back Up Application

Once the machine commissioning is completed, and before operation phase, upload and save project for further use.

Depending of the controller, several methods are available:

- EcoStruxure Machine Expert: Perform a backup of the application program to the hard disk of the PC.
- Controller Web server
- Controller clone function (with SD card).
- …

For more details, refer to the documentation of the device.

Prepare the Device to Be Recognized

Overview

The aim of this step is to configure the IP address assignment method of the device to be in accordance with that configured for the network planning *(see page 30)*.

This can be done during:
- the commissioning phase *(see page 49)*.
- a device replacement *(see page 75)*.

Depending on the device, different tools can be used:
- Machine Assistant *(see Modicon M262 Logic/Motion Controller, Programming Guide)*
- Screwdriver: for devices with rotary switch, dip switch, …(example: OTB)
- Keypad (example: ATV)
- PC, for devices that have to be configured with:
  - EcoStruxure Machine Expert
  - Third-party software
  - Its Web server (example: OsiSense XGCS)

Depending on the IP address assignment, different actions can be done:
- DHCP: configure the DHCP device name in the device.
- BOOTP: refer to Device Configured in BOOTP *(see page 53)*.
- Fixed IP: configure the IP address in the device.

If using EtherNet/IP, and using Electronic Keying, verify whether it is correctly configured.

Main Device Configuration Method

Tool	IP address assignment method	Description
None	DHCP	The device is pre-configured in DHCP with the correct DHCP device name
Screwdriver	DHCP	Use a screwdriver on the device (rotary switch, dip switch, …) to configure the DHCP device name. Example: Advantys OTB.
	BOOTP	Use a screwdriver on the device (rotary switch, dip switch, …) in BOOTP. Example: XPSMCM.
	Fixed IP	Use a screwdriver on the device (rotary switch, dip switch, …) to configure the IP address.
Keypad	DHCP	Use the keypad of the device to configure DHCP device name. Example: ATV32.
	BOOTP	Use the keypad of the device to configure the device in BOOTP.
	Fixed IP	Use the keypad of the device to configure IP address.
PC, tablet, …	DHCP BOOTP Fixed IP	Use PC or tablet to connect to the **Device Web server** and configure the network settings. Choose a connection method: ● Connect the PC to an Ethernet port of the device The current IP address of the device must be known. ● Connect a WIFER TCSEGWB13FA0 to an Ethernet port of the device. Connect the PC to the WIFER.
PC	DHCP BOOTP Fixed IP	Use EcoStruxure Machine Expert (via DTM) to configure the network settings. Connect the PC to a dedicated communication port of the device. Example: Modbus serial line port of ATV32. For details, refer to Using DTMs to Configure Devices on Modbus Serial Line *(see EcoStruxure Machine Expert, Device Type Manager (DTM), User Guide).*
	DHCP BOOTP Fixed IP	Use **third-party software** to configure the network settings. Choose a connection method: ● Connect the PC to an Ethernet port of the device The current IP address of the device must be known. ● Connect the PC to a dedicated communication port of the device.
You may have to perform a device power cycle for parameter modifications to take affect.		

Device Configured in BOOTP

If the device IP address assignment is BOOTP, you must use EcoStruxure Machine Expert:

- Set the MAC address of the new device *(see page 30)*,
- Load the new application in the controller.

Apply the Correct Device Configuration

Overview

Once the device is recognized on the device network, you may have to configure it.

This can be done during:

- the commissioning phase *(see page 49)*.
- a device replacement *(see page 75)*.

Description

It may be necessary to perform different actions, depending on the device, to apply the correct device configuration. In addition, a power cycle of the device may also be required before any configuration information is taken into account by the device.

Action	Description
No manual modification	The device is provided pre-configured. Everything is automated.
	For Advantys OTB, the configuration download can be performed by program only. For more details, refer to Services Configuration on Start *(see page 60)*.
SD card, USB memory key, keypad …	Often, the media used to store the configuration is already prepared for operation. However, inserting the media into the new device may require some manual actions.
Multiloader	Use the multiloader tool to load a previously saved configuration file in the device.
FDR (through keypad menus)	In some cases, you must explicitly ask the device to get its configuration from the FDR server, and then switch the FDR service back to IDLE. For more details, refer to the documentation of the device. For FDR details, refer to Device Replacement with FDR *(see page 37)*.
FDR (through Web server)	Use an external tool such as a PC, smart phone, tablet, etc. that supports the use of a web browser to affect the device replacement. In some cases, you must explicitly ask the device to get its configuration from the FDR server, then switch the FDR service back to IDLE.
Device Web server (parameter by parameter)	Use an external tool such as a PC, smart phone, tablet, etc. that supports the use of a web browser to affect the configuration.
EcoStruxure Machine Expert	Use EcoStruxure Machine Expert to download the configuration to the device. For devices that support DTM, refer to Using DTMs to Configure Devices on Modbus TCP or EtherNet/IP *(see EcoStruxure Machine Expert, Device Type Manager (DTM), User Guide)*.
Third-party software	Use a third-party software.
You may have to perform a device power cycle for parameter modifications to take affect.	

For more information concerning the device configuration, refer to the documentation of the device.

Chapter 4
Device Network Operation

Overview

This chapter describes the functionalities, data exchange process, and security for operating modes.

What Is in This Chapter?

This chapter contains the following topics:

Managing Slave Devices Operating Modes

Overview

The operating modes of slave devices are managed by the protocol manager with the following scanners and their dedicated libraries:

- Modbus TCP IOScanner: Modbus TCP IOScanner library *(see page 79)*

These libraries contain function blocks that allow you to:

- Control the Modbus TCP IOScanner,
- Manage cyclic data exchanges (implicit messages),
- Manage the status variables,
- Send non-cyclic data exchange requests (explicit messages).

Other libraries can be used depending on the devices.

Status Variables of the Modbus TCP IOScanner

There are two status variable types:

- **Health bits**: variables to indicate the communication state of the channels. There is one health bit per channel.
- **Global scanner status**: variable to indicate the Modbus TCP IOScanner state.

This table presents the health bit values:

Health bit value	Communication state of the channel
0	Health timeout expired without receiving a reply.
1	No errors detected. Request and reply are received.

I/O Image Variables

The scanners collect and write data from/to the devices. These variables constitutes the I/O image.

Variables Addresses

Each variable gets its own address:

Variable	Type	Amount
I/O image variables	**%IW** for inputs **%QW** for outputs	A table of words is created per channel/connection.
Health bit	**%IW**	Four consecutive words for Modbus TCP
Global scanner status	**%IW**	One word for Modbus TCP

Function Blocks to Control the Modbus TCP IOScanner

Modbus TCP IOScanner library contains function blocks *(see SoMachine Modbus TCP IOScanner, User Guide)* used by the application to communicate with the controller and the Modbus TCP slave devices:

- `CONFIGURE_OTB`: Send the software configuration of the Advantys OTB
- `IOS_GETSTATE`: Read the state of the Modbus TCP IOScanner
- `IOS_START`: Launch the Modbus TCP IOScanner
- `IOS_GETHEALTH`: Read the health bit value
- `IOS_STOP`: Stop the Modbus TCP IOScanner

For more details, refer to Modbus TCP IOScanner *(see page 79)*.

Function Blocks to Control ATV and Lexium Devices

Use the PLC Open and other function blocks dedicated to drives to control ATV and Lexium devices. These function blocks can be accessed in the GMC Independent PLCopen MC library, GMC Independent Altivar library, and GMC Independent Lexium library. For more information, refer to the Motion Control Library Guide.

Bus Cycle Task

The protocol manager and the slave devices exchange data at each cycle of an application task.

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

- **Use parent bus cycle setting**: associate the scanner with the application task that manages the controller.
- **MAST**: associate the scanner with the MAST task.
- Another existing task: you can select an existing task and associate it to the scanner.

For more information about the application tasks, refer to the EcoStruxure Machine Expert Programming Guide *(see EcoStruxure Machine Expert, Programming Guide)*.

Data Exchanges on Demand

Description

The Cyclic (implicit) data exchanges are managed by the chosen Industrial Ethernet scanner.

To perform data exchanges on demand, you must use explicit messages.

Explicit messages are initiated by the application with function blocks:

- For Modbus TCP devices, you can use `READ_VAR` and `WRITE_VAR` function blocks.
- For TCP/UDP devices, you can use function blocks *(see page 95)*.

Custom Cyclic Data Exchanges

Description

When predefined devices are added in the project, cyclic data exchanges are created automatically.

Furthermore, you can create additional cyclic data exchanges on each slave device *(see page 38)*.

Slave Devices Configuration on Start

Description

To simplify device maintenance, you can send configuration data to slave devices.

In addition, configuration of Advantys OTB devices can be sent on demand by the application using the CONFIGURE_OTB function block *(see page 85)*.

Out of Process Data Exchanges

Overview

Out of process data exchanges are often data exchanges between control network and device network. For example, you may use a supervision software or a third-party configuration tool to communicate with a target on the device network.

The Industrial Ethernet network permits out of process data exchanges.

To enable out of process data exchanges:

- Configure the gateway address in the devices *(see page 32)*.
- Ensure that the IP forwarding service *(see Modicon M262 Logic/Motion Controller, Programming Guide)* is enabled.
- Check the PC routing (see below).

NOTE:
Out of process data exchanges originating from any of the following sources may impact the performance of the controller:

- DTM, Web server, and Modbus TCP requests.
- Network communications (DTM, Web server communications when the PC is on the network).
- TCP UDP communications generated by the TcpUdpCommunications library.

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Do not connect DTMs that communicate across the device network on a running application if the DTM causes deleterious effect on performance.

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

PC Routing

The PC supporting the supervision software or configuration tool must be configured to communicate with the slave devices. The PC must be in the same subnet as one of the Ethernet ports of the controller.

If the slave device is configured...	Then...
As a predefined slave through FDT/DTM	No specific PC parameterization is needed. **NOTE:** The PC configuration is not altered.
Using another tool	If the PC is not in the same subnet as the slave devices, you must update the routing table of the PC (see below).

To update the routing table of the PC, stop every connection from the PC to the controller and/or other devices. Then, in a Windows command prompt, execute the command:

`route ADD` *destination* `MASK` *subnet_mask gateway*

Where:

Parameter	Value
destination	IP address of the Industrial Ethernet network
subnet_mask	Subnet mask of the Industrial Ethernet network
gateway	IP address of the controller port connected to the control network

For example, for a TM251MESE, if:

- IP address of the PC: 192.168.0.2
- Subnet mask of the PC: 255.255.0.0
- IP address of the Industrial Ethernet network: 10.10.0.0
- Subnet mask of the Industrial Ethernet network: 255.255.252.0
- IP address of the control network port “Ethernet_1”: 192.168.0.5
- Subnet mask of the control network port “Ethernet_1”: 255.255.0.0

The corresponding command would be:

`route ADD` 10.10.0.0 `MASK` 255.255.252.0 192.168.0.5

To verify the parameters, execute the command:

`route PRINT`

To remove the route from the PC, execute the command:

`route DELETE` *destination*

Where *destination* is the IP address of the Industrial Ethernet network entered previously.

Protocol Manager Operating Modes

Protocol Manager States

To manage the operating modes of the devices, protocol manager is composed by Modbus TCP IOScanner.

The protocol manager state defines the behavior of the different devices in the device network. For each state, monitoring information (health bit, communication states, and so on) is specific.

The scanners states depend on the controller state:

Controller state	Modbus TCP IOScanner state
EMPTY	IDLE
CONFIGURED	STOPPED
STOPPED	STOPPED
HALT	STOPPED
RUNNING	OPERATIONAL
RUNNING with breakpoint	OPERATIONAL with a specific behavior

Controller EMPTY State

TCP/IP connections are closed.

Device states are managed according to their individual mode of operation.

The Modbus TCP IOScanner is not created (IDLE state).

Therefore, health bits and I/O images are not available.

Controller CONFIGURED State

TCP/IP connections are closed.

Controller enters in CONFIGURED state after:

- an application load.
- a reset (cold/warm) command sent by EcoStruxure Machine Expert.

The Modbus TCP IOScanner is in STOPPED state, all channels with the Modbus TCP slave devices are closed in half-sided mode.

Controller STOPPED State

The Modbus TCP IOScanner is in STOPPED state. All channels with the Modbus TCP slave devices are closed in half-sided mode.

Slave devices are managed according to their individual mode of operation.

This table presents the EcoStruxure Machine Expert variables for Modbus TCP IOScanner:

Variable	Value	Comments
Health bit value	0	-
Input image	0 or the last read value	Input values depend on the **Error Handling** parameter. Input values are those when the controller entered in the STOPPED state and therefore may not reflect the actual state of the input thereafter.
Output image	0 or the last written value	Output values depend on the **Behavior for outputs in Stop** parameter. Output values may not reflect the actual state of the output thereafter.

⚠ WARNING

OUTPUT VALUES IN MEMORY MAY BE DIFFERENT THAN THEIR PHYSICAL STATE

Do not rely on the memory values for the state of the physical outputs when the controller is not in the RUNNING state.

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

Controller HALT State

For Modbus TCP IOScanner, same behavior as the controller STOPPED state.

⚠ WARNING

OUTPUT VALUES IN MEMORY MAY BE DIFFERENT THAN THEIR PHYSICAL STATE

Do not rely on the memory values for the state of the physical outputs when the controller is not in the RUNNING state.

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

Controller RUNNING State

TCP/IP connections are open.

Slave devices are managed by the controller.

This table presents the EcoStruxure Machine Expert variables:

Variable	Value	Comments
Health bit value	0...1	0: No reply from the device before the timeout expired. 1: Requests are sent and replied before the timeout expires.
Input image	Last read value	Values are refreshed synchronously with the task *(see SoMachine Modbus TCP IOScanner, User Guide)* which drives the scanners.
Output image	Last written value	Values are managed by the application.

Controller RUNNING with Breakpoint State

TCP/IP connections are open.

Slave devices are managed by the controller.

⚠ WARNING

OUTPUT VALUES IN MEMORY MAY BE DIFFERENT THAN THEIR PHYSICAL STATE

Do not rely on the memory values for the state of the physical outputs when the controller is not in the RUNNING state.

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

Security

Overview

The **Master IP address** feature can increase the system security level for device replacement.

Master IP Address Description

Some devices have a **Master IP address** parameter so that only one, declared Master, controller has access to the devices.

For more details, refer to Master IP Address Parameter *(see page 36)*.

Chapter 5
Device Network Diagnostics

Overview

This chapter contains troubleshooting information.

What Is in This Chapter?

This chapter contains the following topics:

Network Test

Purpose

Before operating the protocol manager, test the network.

Verify the following:

- The address configuration of each device conforms to the network planning.
- Each device is correctly wired.

Some standard testing methods are presented below.

Status LED

Depending on your devices, verify that the status LEDs display a correct wiring.

Verification Using a PC

With a PC, verify that each network device is connected and addressed:

Step	Action
1	Connect the PC in the Industrial Ethernet network.
2	Access the command prompt.
3	Use a `ping xxx.xxx.xxx.xxx` command to reach each network device, where.`xxx.xxx.xxx.xxx` is the IP address of the device to test. **NOTE:** The command `ping -h` displays the help for the `ping` command.

Verification Using a Web Server

With the controller Web server, verify that the controller can communicate with each network device:

Step	Action
1	Access the controller Web server.
2	Open the **Ethernet Diagnostic** page.
3	Use the **Remote ping** service on each device.

Diagnostics: Web Server

Overview

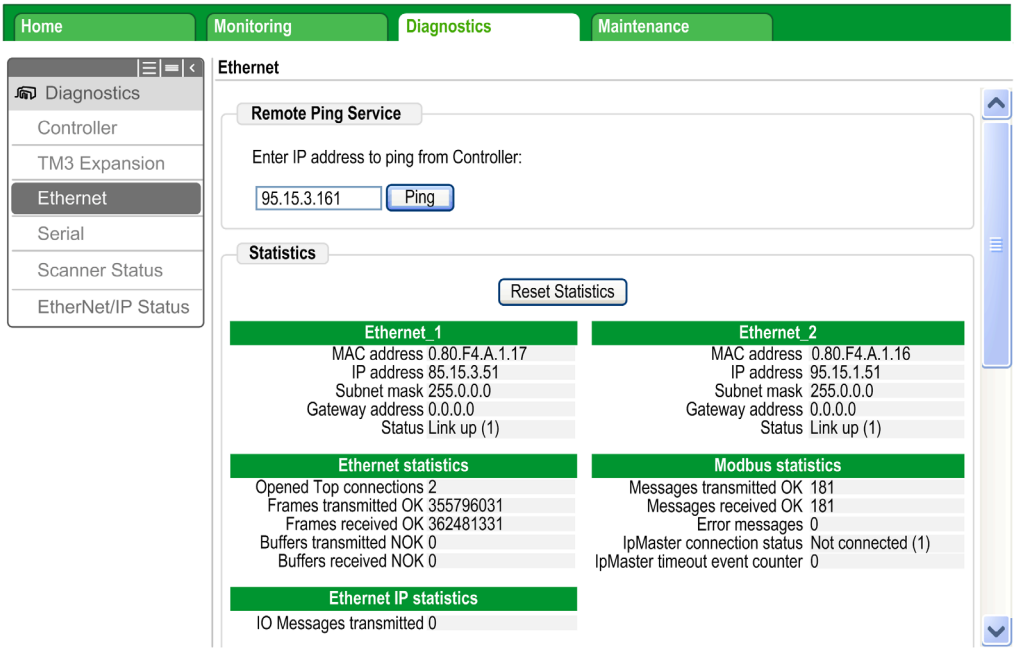
The Web server of the controller has a diagnostic tab.

In this tab, you can access to Industrial Ethernet diagnostic pages:

- **Ethernet** diagnostic page *(see page 69)*
- **Modbus TCP** diagnostic page *(see page 70)*

Ethernet Page

Click **Ethernet** to display Ethernet information of the controller and to allow you to test communication with a specific IP address:

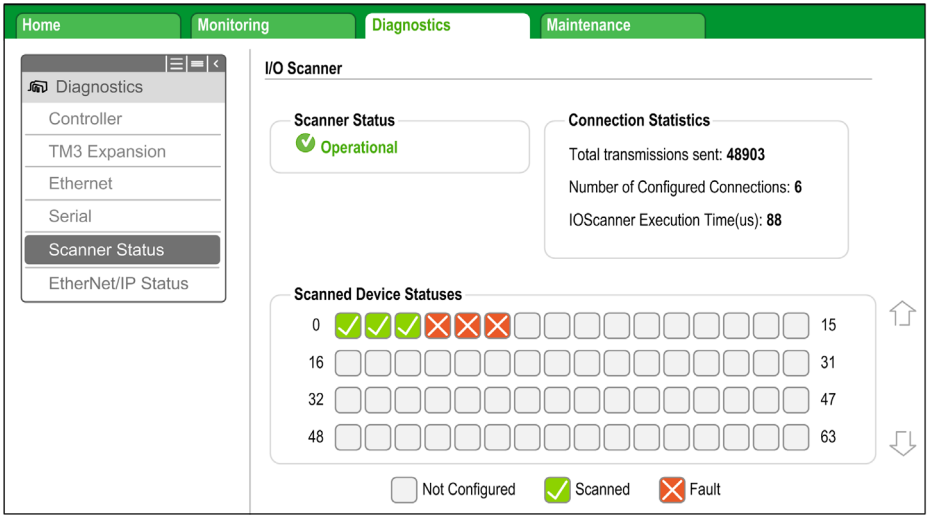


This table presents the ping test result on the **Ethernet** page:

Icon	Meaning
	The communication test is successful.
	The controller is unable to communicate with the defined IP address.

Modbus TCP Status Page

Click **Scanner Status** to display the Modbus TCP IOScanner status (IDLE, STOPPED, OPERATIONAL) and the health bit of up to 64 Modbus TCP slave devices:



0…63 corresponds to the channel ID.

This table presents the status of each channel presented on the **Scanner Status** page:

Icon	Health bit value	Meaning	Scanner status
	1	Request and reply are ongoing on time.	OPERATIONAL
	0	An error is detected, the communications are closed.	OPERATIONAL
	–	This ID does not correspond to a configured channel.	OPERATIONAL
	0	The communications are closed.	STOPPED

NOTE: Click any icon to open the device Web server (if existing). To access this Web server, the computer must be able to communicate with the device. For more information, refer to PC routing *(see page 62)*.
If the Modbus TCP IOScanner status is IDLE, no icon is displayed; **No scanned device reported** is displayed.

Diagnostics: EcoStruxure Machine Expert Online Mode

Overview

In online mode, you can monitor the protocol manager in EcoStruxure Machine Expert using the following methods:

- Icons in the **Devices tree**
- Status tab of the protocol manager and the devices
- **IOScanner I/O Mapping** tab of the protocol manager for Modbus TCP IOScanner
- I/O mapping tab of the devices
- The protocol manager resources tab

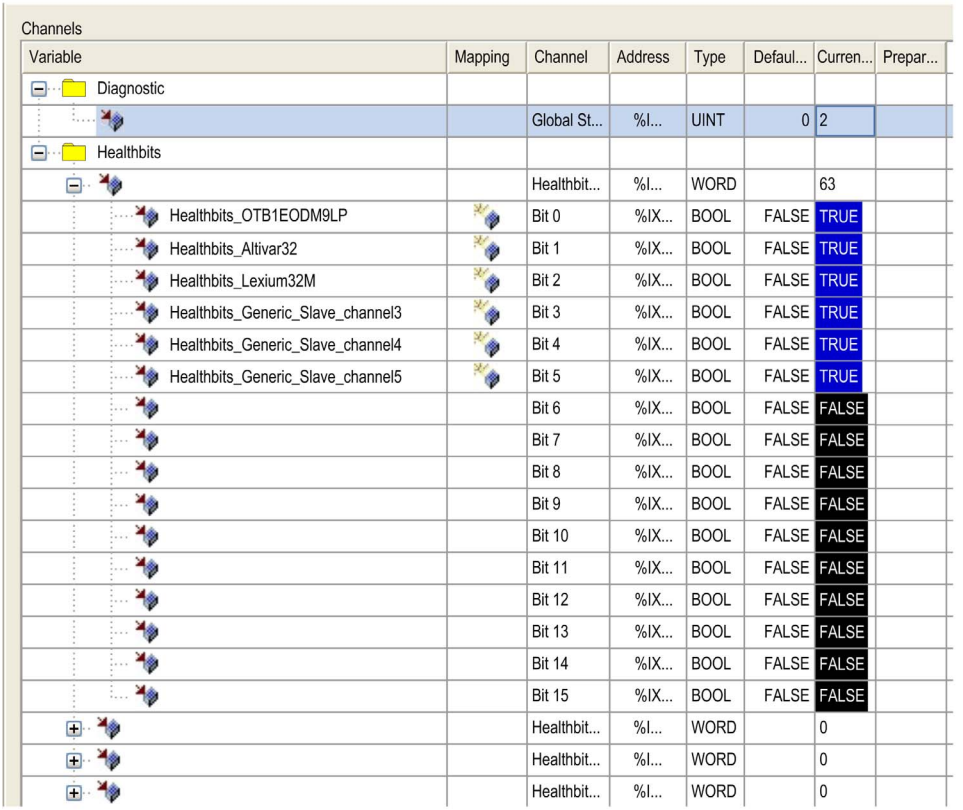
Devices Tree

The communication status of the protocol manager and the devices is presented with icons in the **Devices Tree**:

Icon	Meaning
	The communication with the device is normal. **NOTE:** The protocol manager is always presented with this icon.
	The controller is unable to communicate with the device. **NOTE:** When the protocol manager is STOPPED, all devices show this icon.

Protocol Manager I/O Mapping

The **IOScanner I/O Mapping** tab of the protocol manager allows you to monitor the Modbus TCP IOScanner status and the health bit of the Modbus TCP slave devices:



Channels

Variable	Mapping	Channel	Address	Type	Defaul...	Curren...	Prepar...
Diagnostic							
		Global St...	%I...	UINT	0	2	
Healthbits							
		Healthbit...	%I...	WORD		63	
Healthbits_OTB1EODM9LP		Bit 0	%IX...	BOOL	FALSE	TRUE	
Healthbits_Altivar32		Bit 1	%IX...	BOOL	FALSE	TRUE	
Healthbits_Lexium32M		Bit 2	%IX...	BOOL	FALSE	TRUE	
Healthbits_Generic_Slave_channel3		Bit 3	%IX...	BOOL	FALSE	TRUE	
Healthbits_Generic_Slave_channel4		Bit 4	%IX...	BOOL	FALSE	TRUE	
Healthbits_Generic_Slave_channel5		Bit 5	%IX...	BOOL	FALSE	TRUE	
		Bit 6	%IX...	BOOL	FALSE	FALSE	
		Bit 7	%IX...	BOOL	FALSE	FALSE	
		Bit 8	%IX...	BOOL	FALSE	FALSE	
		Bit 9	%IX...	BOOL	FALSE	FALSE	
		Bit 10	%IX...	BOOL	FALSE	FALSE	
		Bit 11	%IX...	BOOL	FALSE	FALSE	
		Bit 12	%IX...	BOOL	FALSE	FALSE	
		Bit 13	%IX...	BOOL	FALSE	FALSE	
		Bit 14	%IX...	BOOL	FALSE	FALSE	
		Bit 15	%IX...	BOOL	FALSE	FALSE	
		Healthbit...	%I...	WORD		0	
		Healthbit...	%I...	WORD		0	
		Healthbit...	%I...	WORD		0	

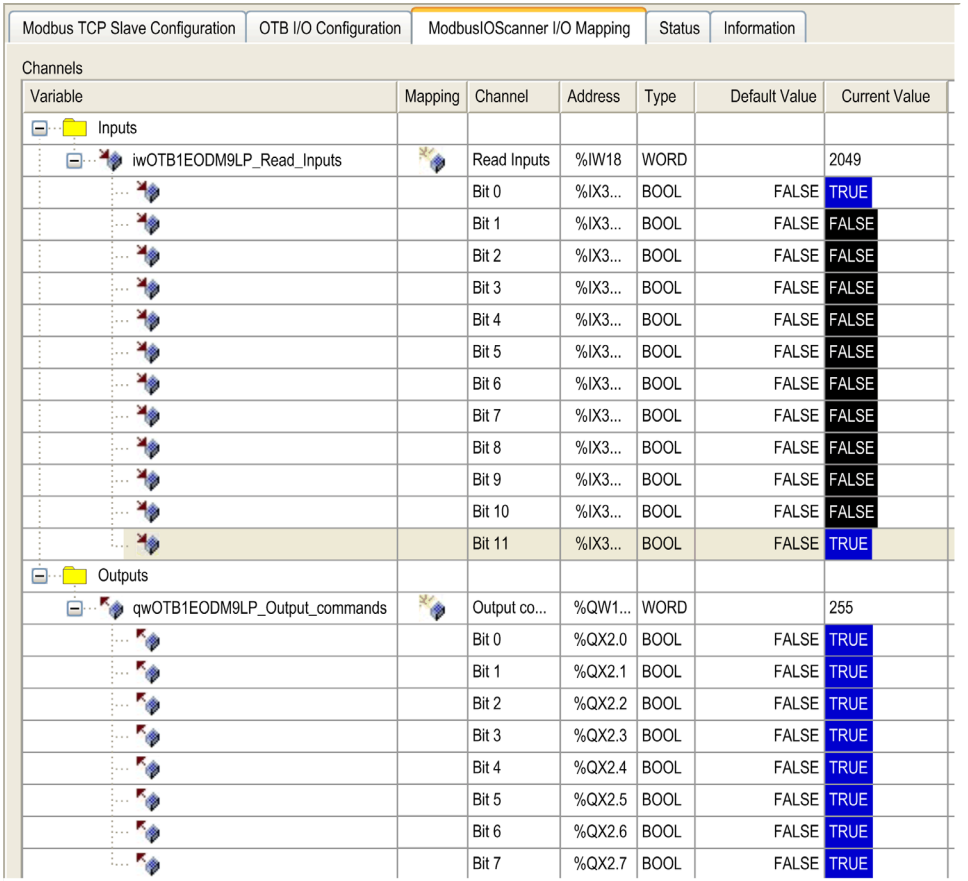
Column		Use	Comment
Variable	**Diagnostic**	Assign a name to the global scanner status variable.	-
	Healthbits	Assign a name to each health bit. For example, name a health bit with the associated device name.	Health bits are grouped in 4 subfolders of 16 bits.
Address		Retrieve the address of each variable.	Addresses may be modified when the configuration is changed.
Current value		Monitor the Modbus TCP devices	For boolean values (health bit): • **TRUE** = 1 • **FALSE** = 0

Slave Device Mapping

Industrial Ethernet devices have an **I/O Mapping** tab containing their I/Os.

NOTE: Generic TCP/UDP does not have an I/O mapping tab;

This figure presents an example of an **I/O Mapping** tab for an Advantys OTB slave device:



Column		Use	Comment
Variable	**Inputs**	Assign a name to each input of the device.	Each bit can also be mapped.
	Outputs	Assign a name to each output of the device.	
Channel		–	Symbolic name of the input or output channel of the device.
Address		Retrieve the address of each variable.	Addresses may be modified when the configuration is changed.
Type		–	Data type of the input or output channel.

Troubleshooting

Main Issues

Symptom	Possible cause	Resolution
Industrial Ethernet manager or Modbus TCP IOScanner is presented with a red triangle in the **Devices tree**.	The configuration is not compliant with the controller version.	• **Build → Clean all** • **Build → Rebuild all** • Ensure that the controller has the latest firmware version.
A device is presented with a red triangle in the **Devices tree**.	The controller is unable to communicate with the device.	• Verify device wiring and powering. • Verify device IP address (by using the Remote ping service *(see SoMachine Modbus TCP IOScanner, User Guide)* on the IP address of the device.). • Verify whether the device supports the read/write request. • Verify whether the accessed registers are relevant for this device. • Verify whether the accessed registers are not write-protected. • Verify that the FDR (Fast Device Replacement) service is properly configured inside the device. • Verify that the **Master IP address** parameter is properly configured inside the device.
A device/channel is temporarily presented in red.	The wiring is unstable.	Verify the wiring.
	Configuration requires adjustment.	• Increase the health timeout value. • Increase the repetition rate value.
	The load is too important for the protocol manager.	Verify the **Scanner Resources** tab *(see page 45)*.
Some states of the device are not presented in the application.	For Modbus TCP slave device: The repetition rate is too slow (the value is too high).	Decrease the repetition rate value for the channels associated to this device.
Some states of the device are not presented in the application.	The bus cycle task is not fast enough.	• Associate the scanner to a different task (Modbus TCP IOScanner). • Decrease the cycle value of the associated task.

Chapter 6
Maintenance

Maintenance Overview

Main Steps

In case of device replacement, the main steps are:

- Power off the machine or the part of the machine affected
- Unmount the device
- Mount the new device
- Power on the new device
- Prepare the device to be recognized by the system *(see page 52)*
- Apply the correct device configuration *(see page 54)*
- Acknowledge the device replacement (depends on your application)

Appendices

What Is in This Appendix?

The appendix contains the following chapters:

Appendix A
Modbus TCP IOScanner Library

Overview

This chapter describes the `ModbusTCPIOScanner` library.

What Is in This Chapter?

This chapter contains the following sections:

Section A.1
Modbus TCP IOScanner Functions

Overview

This section describes the functions included in the `ModbusTCPIOScanner` library.

What Is in This Section?

This section contains the following topics:

IOS_GETSTATE: Read the State of the Modbus TCP IOScanner

Function Description

This function returns the value corresponding to the state of the Modbus TCP IOScanner.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to Function and Function Block Representation *(see page 97)*

I/O Variable Description

This table describes the output variable:

Output	Type	Comment
`IOS_GETSTATE`	IosStateCodes *(see page 89)*	Return values: `IosStateCodes enum`

Example

This is an example of a call of this function:

```
mystate := IOS_GETSTATE() ; (* 0=NOT CONFIGURED 2=OPERATIONAL or
3=STOPPED. *)
```

IOS_START: Launch the Modbus TCP IOScanner

Function Description

This function starts the Modbus TCP IOScanner.

It allows runtime control of the Modbus TCP IOScanner execution. By default, the Modbus TCP IOScanner starts automatically when the application starts.

This function call waits for the Modbus TCP IOScanner to be physically started, so it can last up to 5 ms.

Starting a Modbus TCP IOScanner already started has no effect.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to Function and Function Block Representation *(see page 97)*.

I/O Variable Description

This table describes the output variable:

Output	Type	Comment
IOS_START	UDINT	• 0 = successful start • Other value = start unsuccessful

Example

This is an example of a call of this function:

```
rc := IOS_START() ;
IF rc <> 0 THEN (* Abnormal situation to be processed at application level
*)
```

IOS_GETHEALTH: Read the Health Bit Value

Function Description

This function returns the health bit value of a specific channel.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to Function and Function Block Representation *(see page 97)*.

I/O Variable Description

This table describes the input variable:

Input	Type	Comment
channelID	UINT	Channel ID *(see SoMachine Modbus TCP IOScanner, User Guide)* of the channel to monitor.

This table describes the output variable:

Output	Type	Comment
IOS_GETHEALTH	UINT	• 0: Channel I/O values are not updated • 1: Channel I/O values are updated

Example

This is an example of a call of this function:

```
chID:=1 ;
channelHealth := IOS_GETHEALTH(chID)(* Get the health value (1=OK, 0=Not
OK) of the channel number chID. The channel ID is displayed in the
configuration editor of the device *)
```

IOS_STOP: Stop the Modbus TCP IOScanner

Function Description

This function stops the Modbus TCP IOScanner.

It allows runtime control of the Modbus TCP IOScanner execution. By default, the Modbus TCP IOScanner stops when the controller is `STOPPED`.

The Modbus TCP IOScanner has to be stopped, from the first cycle, until all network devices are operational.

This function call may take as long as 5 ms as it waits for the Modbus TCP IOScanner to physically stop.

Stopping an already stopped Modbus TCP IOScanner has no effect.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to Function and Function Block Representation *(see page 97)*.

I/O Variable Description

This table describes the output variable:

Output	Type	Comment
`IOS_STOP`	UDINT	• 0 = successful stop • Other value = stop unsuccessful

Example

This is an example of a call of this function:

```
rc := IOS_STOP() ;
IF rc <> 0 THEN (* Abnormal situation to be processed at application level
*)
```

CONFIGURE_OTB: Send the Software Configuration of the Advantys OTB

Function Block Description

This function block sends the EcoStruxure Machine Expert configuration data of an Advantys OTB to the physical device through Modbus TCP.

It allows the update of the configuration parameters of an I/O island without third-party software.

The Modbus TCP IOScanner must be stopped before calling this function.

The execution of this function block is asynchronous. In order to check the configuration completion, the `Done`, `Busy`, and `Error` output flags must be tested at each application cycle.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to Function and Function Block Representation *(see page 97)*.

I/O Variable Description

This table describes the input variables:

Input	Type	Comment
`Execute`	BOOL	Activation entry. Start the configuration on rising edge.
`sAddr`	STRING	OTB IP address. The format of the string must be `3{xx.xx.xx.xx}`

This table describes the output variables:

Output	Type	Comment
`Done`	BOOL	Set to TRUE when the configuration completion succeeded.
`Busy`	BOOL	Set to TRUE when the configuration is in progress.
`Error`	BOOL	Set to TRUE when the configuration ended with an error detected.
`ConfError`	configurationOTBErrorCodes *(see page 91)*	Return values: `configurationOTBErrorCodes`
`CommError`	CommunicationError Codes *(see page 90)*	Return values: `CommunicationErrorCodes`

Example

This is an example of a call of this function:

```
VAR
```

(*Function Block to configure OTB , need to stop the IOscanner before the execution of the FB*)

```
configure_OTB1: CONFIGURE_OTB;
```

(*init value different than 16#00000000 , IO_start_done=0 when we have a successful start*)

```
IO_start_done: UDINT := 1000;
```

(*init value different than 16#FFFFFFFF , IO_start_done=16#FFFFFFFF when we have a successful stop*)

```
IO_stop_done: UDINT := 1000;
```

(*Configure_OTB_done= true when we configure with success the OTB, then we can start the IO scanner*)

```
Configure_OTB_done: BOOL;
myBusy: BOOL;
myError: BOOL;
myConfError: configurationOTBErrorCodes;
myCommError: UINT;
myExecute: BOOL;
END_VAR
```

(* First, stop the IOScanner, before configuring OTB *)

```
IF NOT myExecute THEN
IO_stop_done:=IOS_STOP();
END_IF
```

```
(* Send the configuration data to OTB, at IP address 95.15.3.1, when myExecute is TRUE *)
configure_OTB1(
Execute:= myExecute,
sAddr:='3{95.15.3.1}' ,
Done=> Configure_OTB_done,
Busy=> myBusy,
Error=&gt; myError,
ConfError=&gt; myConfError,
CommError=&gt; myCommError);
(* After OTB is successfully configured, start the IOScanner *)
IF Configure_OTB_done THEN
IO_start_done:=IOS_START();
END_IF
```

Section A.2
Modbus TCP IOScanner Data Types

Overview

This section describes the data types of the `ModbusTCPIOScanner` library.

What Is in This Section?

This section contains the following topics:

IosStateCodes: Modbus TCP IOScanner Status Values

Enumeration Type Description

The `IosStateCodes` enumeration data type contains these values:

Enumerator	Value	Comment
`IosErr`	0	Modbus TCP IOScanner is in an error state.
`IosIdle`	1	Modbus TCP IOScanner is in IDLE state. The configuration is empty or not compliant.
`IosOperationnal`	2	Modbus TCP IOScanner is in OPERATIONAL state.
`IosStopped`	3	Modbus TCP IOScanner is in STOPPED state.

CommunicationErrorCodes: Error Detected Codes

Enumeration Type Description

The `CommunicationErrorCodes` enumeration data type contains these values:

Enumerator	Value	Comment
`CommunicationOK`	hex 00	Exchange is correct.
`TimedOut`	hex 01	Exchange stopped because of timeout.
`Canceled`	hex 02	Exchange stopped on user request.
`BadAddress`	hex 03	Address format is incorrect.
`BadRemoteAddr`	hex 04	Remote address is incorrect.
`BadMgtTable`	hex 05	Management table format is incorrect.
`BadParameters`	hex 06	Specific parameters are incorrect.
`ProblemSendingRq`	hex 07	Error detected on sending request to destination.
`RecvBufferTooSmall`	hex 09	Size of reception buffer is too small.
`SendBufferTooSmall`	hex 0A	Size of transmission buffer is too small.
`SystemResourceMissing`	hex 0B	System resource is missing.
`BadTransactionNb`	hex 0C	Transaction number is incorrect.
`BadLength`	hex 0E	Length is incorrect.
`ProtocolSpecificError`	hex FE	The detected operation error contains protocol-specific code.
`Refused`	hex FF	Transaction is refused.

configurationOTBErrorCodes: Error Detected Codes in the OTB Configuration

Enumeration Type Description

The `configurationOTBErrorCodes` enumeration data type contains these values:

Enumerator	Value	Comment
`ConfigurationOK`	hex 00	OTB configuration is done successful.
`IPAddrErr`	hex 01	`sAddr` input parameter is incorrect.
`ChannelNbErr`	hex 02	There is no OTB channel initialization value for this IP address.
`ChannelInitValueErr`	hex 03	Cannot get the OTB channel initialization value.
`CommunicationErr`	hex 04	OTB configuration stopped because of an error detected.
`IosStateErr`	hex 05	The Modbus TCP IOScanner is running. The Modbus TCP IOScanner must be stopped before executing the `CONFIGURE_OTB` function block.

Appendix B
Motion Control Library

Motion Control Library

Overview

This document describes function blocks that are used to control ATV32, ATV320, ATV340 drives, ATV6••, ATV71, ATV9••, LXM32M, ILA, ILE and ILS drives in fieldbus under the EcoStruxure Machine Expert software environment.

For more details, refer to Motion Control Library Guide *(see SoMachine, Motion Control Library Guide)*.

Appendix C
Generic TCP UDP Library

Generic TCP UDP Library

Overview

The TcpUdpCommunication library provides implementing TCP and UDP using IPv4.

The library provides the core functionality for implementing socket-based network communication protocols using TCP (client and server) or UDP (including broadcast and multicast if supported by the platform). Only IPv4-based communication is supported.

The application protocol used by the remote side (which can be hardware such as barcode scanners, vision cameras, industrial robots, or computer systems running software like database servers) has to be implemented using this library. While this requires extensive knowledge of socket-based communication and the protocol used, the TcpUdpCommunication library allows you to concentrate on the application layers.

For more details, refer to TcpUdpCommunication Library Guide *(see SoMachine Motion, TcpUdpCommunication, Library Guide)*.

Appendix D
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

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

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

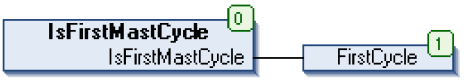
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

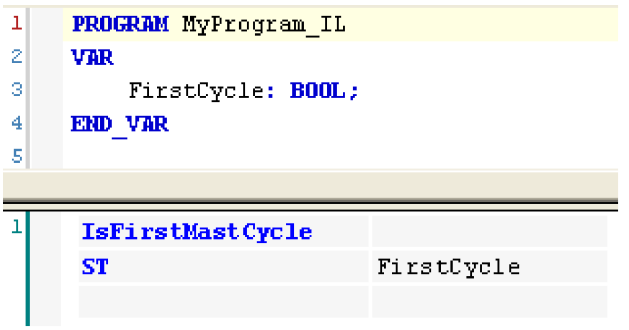
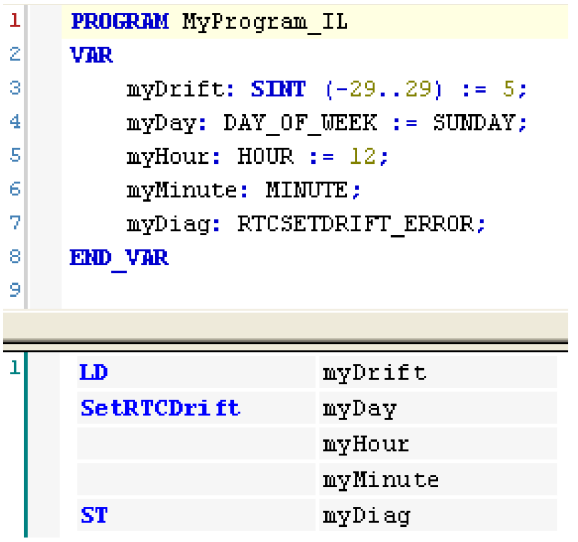
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with `???` in the fields on the right. Replace the `???` with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

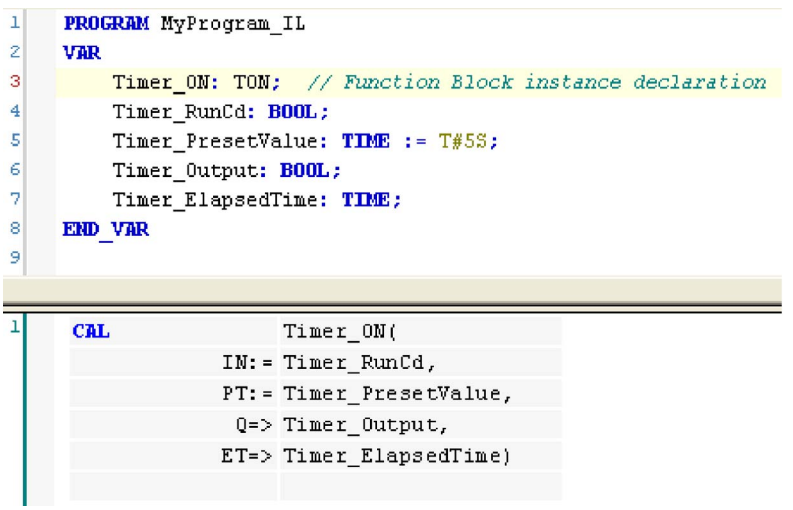
This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a `CAL` instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the `CAL` instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the `CAL` right-side field, replace `???` with the instance name.
5	Replace other `???` with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the `TON` Function Block graphically presented below:

Function Block	Graphical Representation
`TON`	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	

```
PROGRAM MyProgram_IL
VAR
    Timer_ON: TON;  // Function Block instance declaration
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
CAL          Timer_ON(
        IN:= Timer_RunCd,
        PT:= Timer_PresetValue,
         Q=> Timer_Output,
        ET=> Timer_ElapsedTime)
```

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

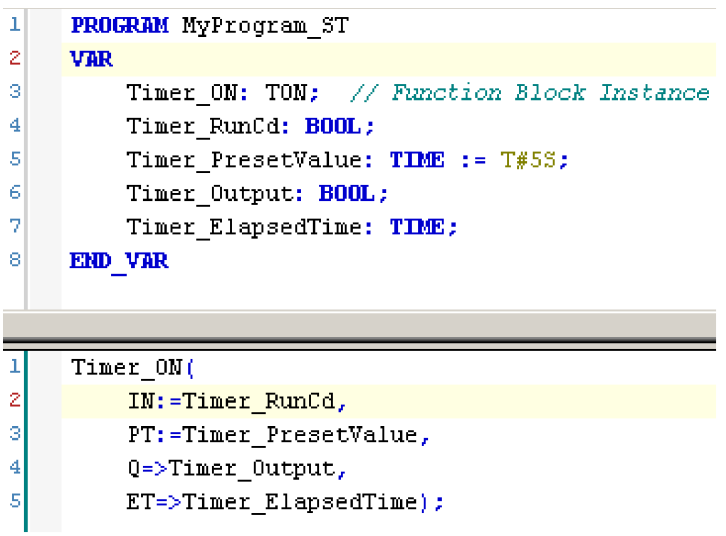
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

!

%IW

According to the IEC standard, %IW represents an input word register (for example, a language object of type analog IN).

%QW

According to the IEC standard, %QW represents an output word register (for example, a language object of type analog OUT).

A

ATV

The model prefix for Altivar drives (for example, ATV312 refers to the Altivar 312 variable speed drive).

B

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

D

device network

A network that contains devices connected to a specific communication port of a logic controller. This controller is seen as a master from the devices point of view.

DHCP

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

DTM

(*device type manager*) Classified into 2 categories:

- Device DTMs connect to the field device configuration components.
- CommDTMs connect to the software communication components.

The DTM provides a unified structure for accessing device parameters and configuring, operating, and diagnosing the devices. DTMs can range from a simple graphical user interface for setting device parameters to a highly sophisticated application capable of performing complex real-time calculations for diagnosis and maintenance purposes.

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

FDR

(*fast device replacement*) A service supported by the device, that facilitate the replacement of an inoperable equipment.

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

H

health bit

Variable that indicates the communication state of the channels.

health timeout

Represents the maximal time (in ms) between a request of the Modbus IO scanner and a response of the slave.

I

IL

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

INT

(*integer*) A whole number encoded in 16 bits.

L

LD

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

M

MAC address

(*media access control address*) A unique 48-bit number associated with a specific piece of hardware. The MAC address is programmed into each network card or device when it is manufactured.

O

ODVA

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

P

post configuration

(*post configuration*) An option that allows to modify some parameters of the application without changing the application. Post configuration parameters are defined in a file that is stored in the controller. They are overloading the configuration parameters of the application.

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

R

repetition rate

Polling interval of the Modbus request that is sent.

RJ45

A standard type of 8-pin connector for network cables defined for Ethernet.

S

ST

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

U

UL

(*underwriters laboratories*) A US organization for product testing and safety certification.

V

variable

A memory unit that is addressed and modified by a program.

Index

O

P

S

T