



# Process Expert - General Purpose Library Classic

## Communication Control Services Reference Manual

EIO0000001312.15  
03/2023



# Legal Information

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

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

Schneider Electric does not grant any right or license for commercial use of the guide or its content, except for a non-exclusive and personal license to consult it on an "as is" basis. Schneider Electric products and equipment should be installed, operated, serviced, and maintained only by qualified personnel.

As standards, specifications, and designs change from time to time, information contained in this guide may be subject to change without notice.

To the extent permitted by applicable law, no responsibility or liability is assumed by Schneider Electric and its subsidiaries for any errors or omissions in the informational content of this material or consequences arising out of or resulting from the use of the information contained herein.

As part of a group of responsible, inclusive companies, we are updating our communications that contain non-inclusive terminology. Until we complete this process, however, our content may still contain standardized industry terms that may be deemed inappropriate by our customers.

© 2023 – Schneider Electric. All rights reserved.

---

# Table of Contents

Safety Information .....	7
Qualification of Personnel .....	7
Proper Use .....	8
Before You Begin .....	8
Start-up and Test .....	9
Operation and Adjustments .....	9
About the Book .....	11
<b>General Overview .....</b>	<b>15</b>
General Overview of Communication Library .....	16
Introduction .....	16
General Overview of Communications Resources .....	17
Communications Resources Overview .....	17
General Concepts .....	18
Logical Architecture – Communication .....	18
Communication Process Diagram .....	19
<b>Modbus and Modbus TCP Ethernet Communication .....</b>	<b>20</b>
Modbus Client Profile .....	21
Description .....	21
DFB Representation .....	22
Inputs .....	22
Outputs .....	24
Inputs/Outputs .....	25
Modbus Port Profile .....	26
Description .....	26
DFB Representation .....	28
Inputs .....	28
Outputs .....	30
Inputs/Outputs .....	31
Modbus Client1 Profile .....	33
Description .....	33
DFB Representation .....	34
Inputs .....	34
Outputs .....	35
Inputs/Outputs .....	37
Modbus Port1 Profile .....	40
Description .....	40
DFB Representation .....	41
Inputs .....	42
Outputs .....	43
Inputs/Outputs .....	44
Scanner Profile .....	46
Description .....	46
DFB Representation .....	47
Inputs .....	48
Outputs .....	49
Inputs/Outputs .....	50
Calculating the Array Size of the Input and Output Parameters .....	50
ModBusGateway - Serial Modbus-Ethernet Gateway .....	51

Description .....	51
DFB Representation .....	51
Inputs.....	52
Outputs .....	53
Inputs/Outputs .....	54
Ethernet IP Communication .....	56
EthernetIPPortMxx - Ethernet IP Port Profile .....	57
Description .....	57
DFB Representation .....	57
Inputs.....	58
Outputs .....	58
Inputs/Outputs .....	58
EthernetIPClient - Ethernet IP Client Profile .....	60
Description .....	60
DFB Representation .....	61
Inputs.....	61
Outputs .....	62
Inputs/Outputs .....	62
StatisticCounter1 - Statistic Counter Profile .....	64
Description .....	64
DFB Representation .....	64
Inputs.....	64
Outputs .....	65
Profibus .....	66
PRMMgt - PRM Management .....	67
Description .....	67
DFB Representation .....	67
Inputs.....	68
Outputs .....	69
Inputs/Outputs .....	70
Public Variables .....	70
Diagnostic Information Management .....	71
Diagnostic Information Management Codes .....	72
Description .....	72
Read_Var and Write_Var Diagnostic Codes .....	73
MBP_MSTR Diagnostic Codes .....	74
Client Parameter Diagnostic Codes .....	76
Scanner Parameter Diagnostic Codes .....	77
EthernetIP Communication Diagnostic Codes .....	78
Communication Technologies .....	82
Supported Architectures .....	83
Device/Communication Port Architectures .....	83
Ethernet Technology .....	88
Ethernet Communication Architecture .....	88
Addressing Example for the M340/M580 Platform .....	89
Addressing Example for the Quantum Platform .....	90
Configuring the EMClient1 and EMPort1M .....	90
Gateway Technology .....	95
Gateway Communication Architecture .....	95
Addressing Example for the M340/M580 Platform .....	96

---

Addressing Example for the Quantum Platform .....	98
Modbus Technology.....	100
Modbus Communication Architecture .....	100
Addressing Example for the Modicon M340/M580 Platform.....	101
EthernetIP Technology.....	103
Ethernet IP Communication Architecture .....	103
EIPPort Client Configuration .....	103
Appendices .....	106
Editing the <code>WorkMemory</code> Array Size.....	107
.....	107
Index .....	109



# Safety Information

## Important Information

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



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



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

### **DANGER**

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

### **WARNING**

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

### **CAUTION**

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

### **NOTICE**

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

## Please Note

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

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

## Qualification of Personnel

A qualified person is one who has the following qualifications:

- Skills and knowledge related to the construction and operation of electrical equipment and the installation.
- Knowledge and experience in industrial control programming.
- Received safety-related training to recognize and avoid the hazards involved.

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical,

electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

## Proper Use

This product is a library to be used together with the automation control systems and is intended solely for the purposes described in the present documentation as applied in the industrial sector.

Always observe the applicable safety-related instructions, the specified conditions, and the technical data.

Perform a risk evaluation concerning the specific use before using the product. Take protective measures according to the result.

Since the product is used as a part of an overall system, you must ensure the safety of the personnel by means of the concept of this overall system (for example, machine concept).

Any other use is not intended and may be hazardous.

## Before You Begin

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

### **WARNING**

#### **UNGUARDED EQUIPMENT**

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

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

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before



placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

**NOTE:** Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

## Start-up and Test

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check are made and that enough time is allowed to perform complete and satisfactory testing.

### **⚠ WARNING**

#### **EQUIPMENT OPERATION HAZARD**

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

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

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

**Software testing must be done in both simulated and real environments.**

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

## Operation and Adjustments

The following precautions are from the NEMA Standards Publication ICS 7.1-1995:

(In case of divergence or contradiction between any translation and the English original, the original text in the English language will prevail.)

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.

- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

# About the Book

## Document Scope

This document describes the function blocks (DFBs) and variables that are encapsulated in the Control facets referenced by the communication control module templates to provide Control services.

For a list of templates and the services that they provide, refer to the user guides mentioned in this document.

This document does not cover any development procedures and internal functionality details of EcoStruxure Process Expert.

This document is for users with knowledge of EcoStruxure Process Expert, and of the Control Participants.

## Validity Note

This document has been updated for the release of EcoStruxure™ Process Expert 2023.

## Related Documents

The characteristics that are described in the present document, as well as those described in the documents included in the Related Documents section below, can be found online. To access the information online, go to the Schneider Electric home page [www.se.com/ww/en/download/](http://www.se.com/ww/en/download/).

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

Title of Documentation	Reference Number
EcoStruxure Process Expert User Guide	EIO0000001114
EcoStruxure™ Process Expert - General Purpose Library Classic Communication Templates Reference Manual	EIO0000001311

## Technical Support

Visit <https://www.se.com/myschneider/> for support, software updates, and latest information.

## Product Related Information

### **⚠ WARNING**

#### **LOSS OF CONTROL**

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

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

<sup>1</sup> 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.

Examples described in this manual are provided for information only.

### **⚠ WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Adapt examples that are given in this manual to the specific functions and requirements of your industrial application before you implement them.

**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.

Standard	Description
	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.



---

# General Overview

## What's in This Part

General Overview of Communication Library .....	16
General Overview of Communications Resources .....	17
General Concepts .....	18

## Overview

This part provides:

- overview of Communication library
- overview of Communication resources
- basic concepts used to implement the communications control resources

# General Overview of Communication Library

## What's in This Chapter

Introduction..... 16

## Introduction

### Overview

The EcoStruxure Process Expert automation system provides resources that have been pre-configured and tested by Schneider Electric and that are specifically designed for automating systems with networks or communication buses.

The control resources for Control (function blocks, components, and libraries) provide the commonly required functions facilitating the development of control systems under the M340, M580 and Quantum automation platforms and the development of the devices connected to them (for example, variable speed drives, starters, power meters, servo drives, and so on). The physical connection between these devices and the controllers is established with a Modbus, Profibus, or Ethernet fieldbus.

The system provides Control function blocks (DFB) that can be used jointly with the tools for code generation and that complement resources from the Device library.

This document describes the basic concepts behind the platform resources for communications and includes a detailed overview of the corresponding function blocks.



# General Overview of Communications Resources

## What's in This Chapter

Communications Resources Overview..... 17

## Communications Resources Overview

### Overview

This chapter lists the resources designed for network or communication bus-based communications.

### List of Function Blocks for Communications

The DFBs have been specifically designed for the automation of Modbus and Modbus TCP Ethernet communications systems.

The DFBs have been classified based on the families being used.

The following table lists the function blocks and its description:

Family name	Profile name	Function blocks	Description
Modbus and Modbus TCP Ethernet	Client	ModBusClientBasic, page 21	Modbus client basic.
		ModBusEthernetClient, page 21	Ethernet client.
	Port	ModbusPortM, page 26	M340 Modbus port.
		ModbusPortQx80, page 26	ModbusPortQx80 Modbus port.
		ModbusPortM58x80, page 26	M58x80 Modbus port.
		ModbusEthernetPortM, page 26 or EthernetPortM340	M340 Modbus TCP Ethernet port.
		ModbusEthernetPortQ, page 26 or EthernetPortQuantum	Quantum Modbus TCP Ethernet port.
	Scanner	ModBusScanner, page 46	Modbus scanner.
		ModBusEthernetScanner, page 46	Ethernet scanner.
	ModBusGateway	ModBusGateway, page 51	Serial modbus-Modbus TCP/IP gateway (Ethernet) or Modbus – Ethernet gateway.
EthernetIP Communication	Port	EthernetIPPortMxx, page 57	Port for Ethernet IP communication.
	Client	EthernetIPClient, page 60	Client for Ethernet IP communication.
	Statistic counter profile	StatisticCounter1, page 64	Communication client statistics.
Profibus	PRM management	PRMMgtM	PRM management M340/M580
		PRMMgtQ	PRM management Quantum

# General Concepts

## What's in This Chapter

Logical Architecture – Communication .....	18
Communication Process Diagram.....	19

## Overview

This chapter describes the basic concepts behind the communication Control Expert components.

## Logical Architecture – Communication

### Basics

The basic operation of the communication functions consists of various clients or scanners storing issued requests in a memory that is managed by the ports.

The ports are linked to a physical port on the controller and they send the requests to the correct destination based on a defined algorithm, managed priorities, and waiting times. Finally, the ports return the corresponding response to the client or scanner that generated the request.

### Memory Management

To provide appropriate memory management, the data exchange zone consists of a dynamic structure that adapts to the needs of each program.

The necessary work memory (`WorkMemory`) is calculated during the first scanning cycle.

**NOTE:** To enable the port to carry out an optimal calculation of the work memory, configure clients and scanners and execute program instances when the first scanning cycle of the controller occurs. This action maintains the correct order of execution. Clients or scanners serialize the requests, followed by the port.

### Gateway

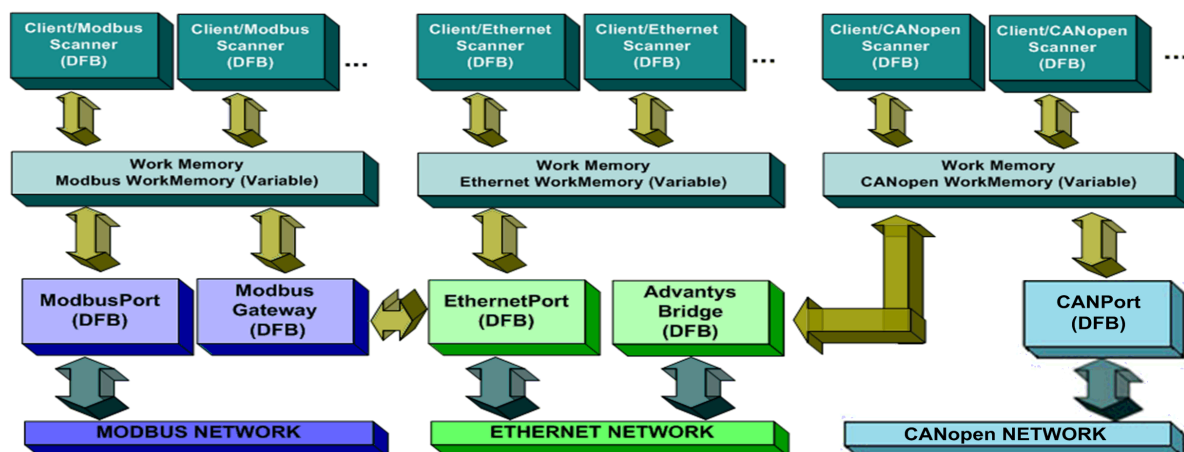
A Gateway is a hardware device that connects Modbus Ethernet networks with serial Modbus networks. To use this device, use a DFB that converts requests from serial clients to Ethernet client requests.

After the Gateway receives these requests, it converts the requests into serial requests again and sends them to the corresponding devices. The main difference between serial requests and Ethernet requests is how the device is addressed.

Internally, a Gateway is made up of a modified serial Modbus port and an Ethernet client. Instead of calling the serial communication functions (that is what a normal port does), the serial Modbus port enters the request data into the Ethernet client, waits for its response, and returns the data like a normal port would.

## Communication Process Diagram

The following diagram represents the communication process.



# Modbus and Modbus TCP Ethernet Communication

## What's in This Part

Modbus Client Profile .....	21
Modbus Port Profile .....	26
Modbus Client1 Profile .....	33
Modbus Port1 Profile .....	40
Scanner Profile .....	46
ModBusGateway - Serial Modbus-Ethernet Gateway .....	51

## Overview

This part provides the detailed description of the Modbus and Modbus TCP Ethernet communication-based DFBs.

These function blocks do not reflect any specific installation.

**⚠ WARNING**

**LOSS OF CONTROL**

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

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

<sup>1</sup> 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.

# Modbus Client Profile

## What's in This Chapter

Description .....	21
DFB Representation .....	22
Inputs .....	22
Outputs .....	24
Inputs/Outputs .....	25

## Overview

This chapter describes the DFBs of the `Modbus Client` profile.

## Description

### General

A communication client allows device data to be written or read through Modbus, and Ethernet Communication Protocols.

By using a communication client, you can access remote device data that cannot be accessed normally with components generated with the Control Expert solution. For example, this enables you to read/write from/to a variable speed drive parameter if this parameter is not available on the respective control block of the speed drive.

Access to this data is explicit, that is, you need to program the access to this data. This is different from the implicit access used in other communication, as with Ethernet IO Scanning, in which access needs to be configured but not programmed.

The *ModBusClientBasic* and *ModBusEthernetClient* DFBs send a read or write request for  $n$  registers on a Modbus communication bus and an Ethernet communication network.

The *ModBusClientBasic* and *ModBusEthernetClient* DFBs belong to the Modbus communication and Modbus TCP Ethernet profile.

## Function Description

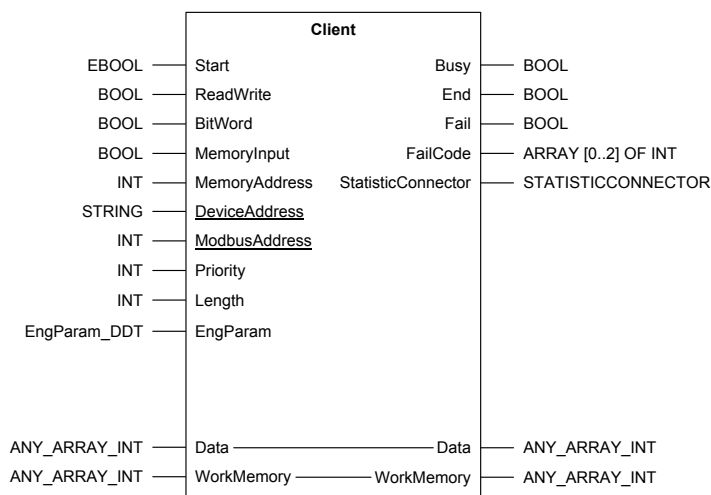
The main functions of the client DFB are described in the following table:

Function	Description
Read/Write	Enables you to select whether a parameter has read or write access.
Multiple register reading/writing	Enables several consecutive registers to be read (available only in <i>ModBusEthernetClient</i> and <i>ModBusClientBasic</i> ).
Diagnostic information management	Monitors detected transaction problems and identifies them on 3-levels to determine the source of the error detected.
Priorities	Enables you to define priorities for systems with multiple client accesses.
Statistics	Obtains the transactions OK/NOK status and their access times.

# DFB Representation

## Representation

The following figure represents the functional module of *Client*:



**NOTE:** The underlined parameters are specific for some components.

The table shows the parameters available for specific DFBs:

Parameters		Components	
		Modbus	Ethernet
		<i>ModBusClientBasic</i>	<i>ModBusEthernetClient</i>
Inputs	DeviceAddress	–	X
	ModbusAddress	X	–
X: Parameter is available			
–: Parameter is not available			

## Inputs

### Input Parameter Description

Parameter	Type	Description
Start	EBOOL	<p>1 = Resets detected errors. Indicates to the client that the data on the inputs is valid and a request to a server needs to be issued.</p> <p>0 = Triggers an ACK of the end-of-operation notification and the client is ready for the next cycle.</p> <p>When the signal is activated, it copies the parameters to the function so modifying the parameters has no effect.</p>
ReadWrite	BOOL	<p>1 = Write operation.</p> <p>0 = Read operation.</p>
BitWord	BOOL	<p>1 = Operation performed on words.</p> <p>0 = Operation performed on bits.</p> <p><b>NOTE:</b> Bit operation is not available if the client block is used with <i>ModBusEthernetPortQ</i>, <i>ModBusEthernetPortQ16</i> and <i>ModBusPortQx80</i> port block.</p>

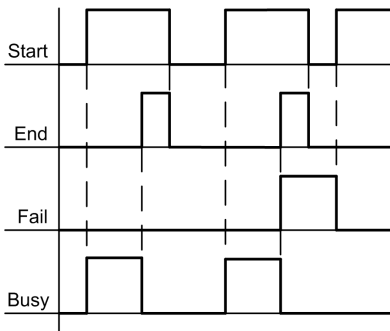
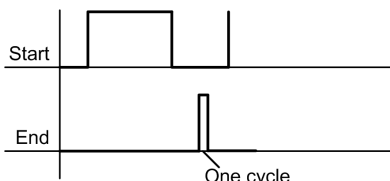
Parameter	Type	Description
MemoryInput	BOOL	<p>1 = Selects input zones (30000 or %IW registers).</p> <p>0 = Selects memory zones (40000 or %MW registers).</p> <p><b>NOTE:</b> CPU embedded ethernet ports cannot be used to read input registers through the client blocks.</p>
MemoryAd- dress	INT	<p>Memory address where the requested operation starts.</p> <p>Any value is valid for the master because the slave needs to validate the address based on its memory map. You can enter the values as hexadecimal or decimal values.</p> <p>These indexes are expressed in decimal values in device manuals. To express the index as a hexadecimal value, use the string 16# before the index, that is, 16#index.</p>
DeviceAd- dress*	STRING [26]	<p>Device address. Refer to the Ethernet Technology, page 88 to see examples.</p> <p>Depending on the platform, the following definitions apply:</p>
	<b>Platform</b>	<b>IP addressing DeviceAddress (variable)</b>
	M340	'{ IP } ID'
	Quantum	'{ IP } ID'
		<b>NOTE:</b> Do not omit punctuation marks.
ModbusAd- dress	INT	Address of the Modbus slave that the client needs to access. The range of possible values goes from 1 to 255.
Priority	INT	<p>Command priority. The lower the value, the higher the priority, that is, 0 is the maximum priority. Any value is valid.</p> <p>If you integrate a Modbus client together with any element from the Control Device library, consider the following priorities to not interfere with these elements:</p>
	<b>Variable value</b>	<b>Description</b>
	1	Control commands. They have maximum priority.
	2	<p>Command confirmation. The read operations are carried out to confirm that the commands have been executed.</p> <p>Medium priority.</p>
	3	Data reading.
		<p>If these priority levels are not observed, the client requests do not execute or the client does not allow other requests to be transmitted.</p> <p>If you do not integrate it together with other Control Device library elements, you can define any priority levels that you want. Otherwise, use priority levels greater than or equal to 4.</p>
Length	INT	<p>Length of the data involved in the request that has been carried out.</p> <p>For Modbus (ModBusClientBasic) or Ethernet (ModBusEthernetClient) communication, the length can be in bits or words depending on the BitWord variable. The maximum length for words is 120.</p>
EngParam	EngPar- am_DDT, page 24  EngPar- amMB- Client/ EngPara- mEM- Client	Engineering parameters.
* This parameter is available only with <i>ModBusEthernetClient</i> .		

## EngParam\_DDT

Parameter	Type	Description
MaxReadSize	INT	<p>This configuration parameter indicates how many words are read in the request issued by this client. If no value is specified, the maximum possible value is used by default (125 words for Ethernet/Modbus). This parameter is used for calculating and managing the work memory area (<i>WorkMemory</i>).</p> <p><b>NOTE:</b> If several requests are made with the same client, use the maximum length of the read requests to be issued as the value of the parameter.</p>
MaxWriteSize	INT	<p>This configuration parameter indicates how many words are written in the largest request issued by this client. If no value is specified, the maximum possible value is used by default (125 words for Ethernet/Modbus). This parameter is used for calculating and managing the work memory area (<i>WorkMemory</i>).</p> <p><b>NOTE:</b> If several requests are made with the same client, use the maximum length of the write requests to be issued as the value of the parameter.</p>

## Outputs

### Output Parameter Description

Parameter	Type	Description
Busy	BOOL	<p>Activated while a request is taking place.</p> <p>1 = Indicates that the client is busy, and that you cannot make new requests.</p>
End	BOOL	<p>Activated when a request cycle ends.</p> <p>1 = Indicates that the operation has ended (with or without a detected error). This signal is acknowledged (ACK) by setting the <i>Start</i> signal to Low.</p>
Fail	BOOL	<p>Activated when a detected error occurs in the request transmission.</p> <p>1 = Indicates that the operation is unsuccessful.</p> <p>To reset the detected error, activate the client again by setting the <i>Start</i> signal to High.</p> <p>Timing diagram:</p>  <p><b>NOTE:</b> If for any reason the <i>Start</i> signal is FALSE when the <i>End</i> signal is activated, only an edge for one cycle of the <i>End</i> signal is produced. This can result in the program (depending on how it is designed) not detecting the <i>End</i> signal, which is why this should not be allowed. The corresponding behavior would be as follows:</p> 



Parameter	Type	Description
FailCode	ARRAY [0..2] OF INT	Indicates the last detected error that took place according to 3 detected error levels, page 71.
StatisticConnector	STATISTICCONNECTOR	<p>The information data is used to obtain network statistics (requests carried out, time between requests, and so on). This structure has been created for its use together with the <code>StatisticCounter</code> module in Communication library.</p> <p>The following table describes the <code>StatisticConnector</code>:</p>
	<b>Parameter</b>	<b>Type</b> <b>Description</b>
	Start	BOOL      1 = Operation has started.
	EndOk	BOOL      1 = Operation has ended correctly.
	EndNOk	BOOL      1 = Operation has ended with a detected error.
	PartialTime	DINT      Partial time.

## Inputs/Outputs

### Input/Output Parameter Description

Parameter	Type	Description
Data	ANY_ ARRAY_ INT	Holds the write data or the read data depending on the <code>ReadWrite</code> input parameter.
WorkMemory	ANY_ ARRAY_ INT	Array is used for communication. This variable is used in a <code>CanPort/ModBusPort/ModBusEthernetPort</code> , which serializes client requests in an optimum manner.

# Modbus Port Profile

## What's in This Chapter

Description .....	26
DFB Representation .....	28
Inputs .....	28
Outputs .....	30
Inputs/Outputs .....	31

## Overview

This chapter describes the DFBs of the `Modbus Port` profile.

## Description

### General

A Port DFB is a function which is capable of serializing and managing requests sent to a physical medium working on Modbus TCP/IP, Modbus serial on local rack or Modbus serial on X80 rack.

A client is a Modbus communication (ATV, PM, ATS, and so on) device DFB or a generic read/write DFB used to communicate a device on the physical medium (clients and scanners).

The basic operation consists of various clients storing requests in a memory that is managed by the port (Modbus, or Ethernet). The port takes out requests from the queue based on a defined algorithm that manages priorities and waiting times and sends them to the appropriate destination after the port returns the response of the destination to the client that generated the request.

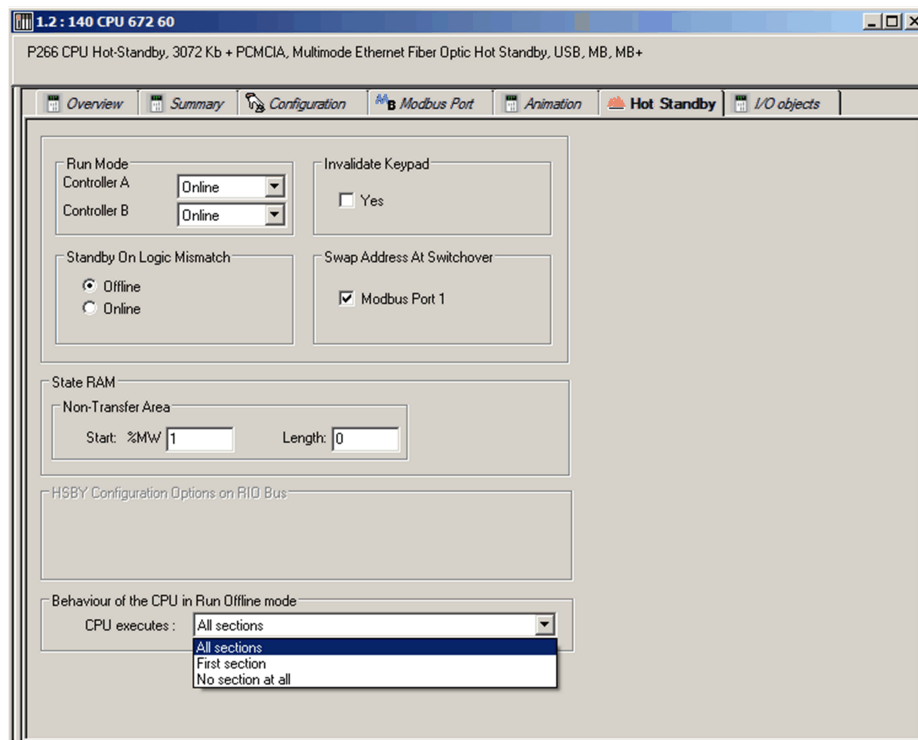
Use a Port DFB for any messaging requests. A minimum of 1 port per controller hardware is necessary to effect communication of this type.

#### NOTE:

- In a serial Modbus (RTU) project, use at least 1 Port DFB because communication on a Modbus takes place through messaging by definition.
- If the General Purpose library for device is used with Modbus TCP/IP networks (IOScanning), Port DFBs are not required for their control because these devices exchange cyclic data through IOScanning technology on Modbus TCP/IP. Any information that is not included in the Device library DFBs needs to be requested through acyclic exchange, that is, through requests sent to a Port DFB.
- New device DFB cannot be added to the existing explicit communication (Port DFB) while controller is running.

To add a new device DFB user can instantiate a new port DFB which is mapped to the same physical port.

**NOTE:** For Port and its client DFBs to work as expected in HSBY controller, it is mandatory that these DFBs should not run in standby controller. This configuration can be done in the HSBY controller configuration as shown below.



Depending on the technology used, the function name varies to enable you to identify the product range to which the function belongs during generation:

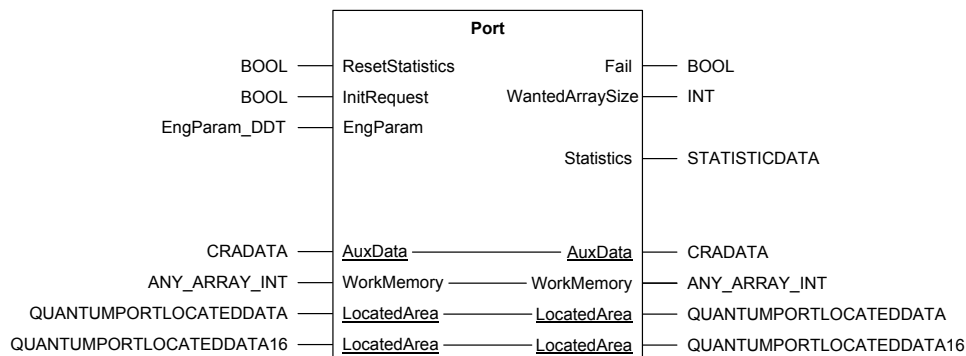
These functions are further classified to support, maximum number of concurrent operations based on Communication or CPU module. The table describes which port function to be used with the controller.

Communica- tion	M340	M580	Quantum	X80	Maximum Simultaneous Sends
Modbus RTU			–	<i>Modbus-PortQX80</i>	4
	–	–	–	<i>Modbus-PortM58X80</i>	4
ModbusTCP/IP	<i>ModbusEthernetPortM</i>		<i>ModbusEther- netPortQ</i>	–	4
	<i>ModbusEthernetPortM16</i>		<i>ModbusEther- netPortQ16</i>	–	16
	–	<i>ModbusEthernet- PortM32</i>	–	–	32
	–	<i>ModbusEthernet- PortM48</i>	–	–	48
	–	<i>ModbusEthernet- PortM64</i>	–	–	64
	–	<i>ModbusEthernet- PortM80</i>	–	–	80

# DFB Representation

## Representation

The following figure represents the functional module of *Port*:



**NOTE:** The underlined parameters are specific for some components.

The table shows the parameters available for specific DFBs:

Parameters		Components				
		Modbus			Ethernet	
		<i>ModBusPortM</i>	<i>ModBus-PortQx80</i>	<i>ModBusPortM58x80</i>	<i>ModbusEthernet-PortM/16/32/48/64/80</i>	<i>ModbusEthernetPortQ/16</i>
Inputs/ Outputs	AuxData	–	X	–	–	–
	LocatedArea	–	–	–	–	X
X: Parameter is available						
–: Parameter is not available						

## Inputs

### Input Parameter Description

#### ⚠ WARNING

##### UNINTENDED EQUIPMENT OPERATION

Use *InitRequest* pin only when the devices connected to *Port* function block are not running.

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

Parameter	Type	Description
ResetStatistics	BOOL	1 = Resets the data on the <code>Statistics</code> output variable to its default values (0).  The input is level-based, that is, <code>Statistics</code> remains 0 as long as the input is TRUE.
InitRequest	BOOL	Reset all the current processing request of the Ethernet client. On triggering this pin, client connected to the port block may go into fail state. This input pin works on rising edge.  <b>NOTE:</b> This input pin has to be used only when the PLC data restore operation is performed.
EngParam	EngParam-DDT, page 29  EngParam-MB-Port/ EngParam-Port-M58x80/ EngParam-MBPQ-x80/ EngParam-EM-Port/ EngParam-EM-PortQ	Engineering parameters.

## EngParam\_DDT

Parameter	Type	Description
TimeOut	TIME	Time to wait for a response, after this port will send a communication timeout to the requesting device.
SimultaneousSends	INT	Indicates how many requests are queued (serialized) in the Modbus master.  The value ranges from 1 to 4. The more requests in the card queue, the better the general performance of the system. You need to consider the following: <ul style="list-style-type: none"> <li>There is a maximum number of queued requests per CPU. This means that if a high number is selected, limit the number of simultaneous transmissions so that the total of these values does not exceed the limit.</li> <li>Priority management does not consider requests that are queued in the hardware buffer, which enables a high-priority command to wait until lower-priority commands that are already in the queue are sent first. This action results in a longer response time for high-priority commands.</li> </ul> For detail description, refer to <code>SimultaneousSends</code> table, page 26.
PortAddress*	STRING	Defines the physical port that the DFB uses to send the Modbus TCP/IP request to (M340/M580/Quantum/NOE/NOC) Modbus TCP/IP port. To define, refer to <code>Ethernet communication architecture</code> , page 88.
EthPortAddress**	STRING	Defines the physical port that the DFB uses to send the Modbus TCP/IP request to (M580CPU/CRP/NOE/NOC) Modbus TCP/IP port. To define, refer to <code>Ethernet communication architecture</code> , page 88.
CRAIPAddress**	STRING	Defines the physical port that the DFB uses to send the Modbus TCP/IP request to CRA. To define, configure ip address as {ip.ip.ip}.
MaxRetryAfterSwitch***	INT	Maximum number of retry for first request is to be successful after the controller switchover. If the response is not received from the field devices after a specified number of retry then, port block will send the detected error received in the last retry to the client which was requesting the data.

Parameter	Type	Description
MBPortAddress**	STRING	Defines the physical port that the DFB uses to send the Modbus serial request to Modbus serial port. It is calculated automatically by the template. To define, configure Modbus port address as <i>Rack.Slot.Channel</i>
<p>* This parameter is available only with <i>ModbusPortM</i>, <i>ModbusEthernetPortM</i> and <i>ModbusEthernetPortQ</i>.</p> <p>** These parameters are available only with <i>ModbusPortQx80</i> and <i>ModbusPortM58x80</i>.</p> <p>*** This parameter is not applicable to <i>ModbusPortM</i>.</p>		

## Outputs

### Output Parameter Description

Parameter	Type	Description	
Fail	BOOL	<p>1 = Indicates that a detected error has occurred while transmitting a request. It is only an indication, and the port process continues. The bit does not have to be reset, and it cannot be reset. As soon as a request is issued without a detected error, it is set to 0.</p> <p>If Fail bit is set to 1, following can be the possibilities:</p> <ul style="list-style-type: none"><li>• If Workmemory area is less than the WantedArraySize</li><li>• If the PortAddress is not correctly configured.</li></ul> <p>Check the diagnostic codes in each DFB belonging to devices or clients/scanners.</p>	
WantedArraySize	INT	<p>Indicates the size necessary for the WorkMemory array to function correctly. Declare the array with the minimum size of [0..WantedArraySize-1] because arrays start on 0.</p> <p>If you add more Modbus, or Modbus TCP/IP clients/scanners or more serial Modbus (RTU) devices to the program during programming, the size of this variable increases making it necessary to monitor the growing size of the WorkMemory array.</p> <p><b>NOTE:</b> This variable is calculated during the first scanning cycle so adding serial Modbus (RTU) clients or devices in the online mode has the following effects:</p> <ul style="list-style-type: none"><li>• The size of the array will not increase.</li><li>• The ModbusGateway DFB does not consider requests from these new clients.</li></ul>	
LostMessages	INT	<p>Indicates the number of messages that have been lost (that the client has not received).</p> <p>This value needs to be 0. Otherwise, the user has written an incorrect program (a client/scanner instance is probably not executed every cycle).</p> <p>If messages are lost, the user has to diagnose the reason (either hardware wiring or incorrect program).</p>	
Statistics	STATISTICDATA	<p>Holds a structure with statistical data on the operation of the Port DFB. This data is useful for debugging the application.</p> <p>The information obtained in this data structure is the statistic for the requests managed by this Port DFB. The clients associated with this DFB through the WorkMemory array send the requests.</p> <p>The following table describes the StatisticData type:</p>	
	Name	Type	Description
	RequestsSended	DINT	Total number of requests sent.
	RequestsOk	DINT	Total number of requests that ended correctly.
	RequestsError	DINT	Total number of requests that ended in a detected error.
	MinTime	DINT	Minimum time required to end a request (msec).
	AvgTime	REAL	Average time required to end a request (msec).
	MaxTime	DINT	Maximum time required to end a request (msec).

Parameter	Type	Description	
	LastTime	DINT	Time required to end the last request (msec).
	CurrentTime	DINT	Time spent on processing the current request (0 if no request is in progress).
	RequestsSecond	DINT	Number of requests that can be sent per second (calculated with the AvgTime value).
	LastCycleNumber	DINT	Number of controller cycles that the last request took to be executed.
	CurrentCycleNumber	DINT	Number of controller cycles that the current request has taken so far (0 if no request is in progress).
	TimeOnQueue	DINT	Time that the last request has been waiting in the queue before being sent (only useful for client statistics).

## Inputs/Outputs

### Input/Output Parameter Description

Parameter	Type	Description	
AuxData*	CRADATA	This structure is used for managing CRA data requests.	
	<b>Parameter</b>	<b>Type</b>	<b>Description</b>
	CRADDataSubPort1	ARRAY	CRA data for SubPort1.
	CRADDataSubPort2	ARRAY	CRA data for SubPort2.
	CRADDataSubPort3	ARRAY	CRA data for SubPort3.
	CRADDataSubPort4	ARRAY	CRA data for SubPort4.
WorkMemory	ANY_ARRAY_INT	<p>This array is for use with a Modbus-RTU, or Modbus TCP/IP Port DFB holds the read/write requests of the clients/scanners.</p> <p>The <code>WorkMemory</code> array is the work memory or link between the serial Modbus clients/scanners and an Ethernet port DFB.</p> <p>During the first scanning cycle, the clients/scanners or devices reserve read/write %Mwords from this array based on the exchange words required by each DFB. After they have been reserved, the Port DFB counts them, and as a result specifies and identifies how many pieces of data are to be serialized.</p> <p>Adjust the <code>WorkMemory</code> parameter to set:</p> <ul style="list-style-type: none"> <li>The size of the data exchange area between the device and client DFBs.</li> <li>The Port DFB to the size needed for the elements that are being used.</li> </ul> <p>The system is not limited by fixed sizes or unused memory.</p>	
LocatedArea**	QUANTUMPORTLOCATEDDATA	<p>Due to internal requirements, the Quantum port needs some variables to be mapped.</p> <p>As a result, this input provides the port with a memory area that has the internal structure necessary for it to work. The internal data does not have to be checked, and do not modify this internal data.</p> <p>Declare this variable in the variable table and allocate an address to it; otherwise the port does not work. To do this, specify the variable address.</p> <p>The necessary data consists of 436 words. If, for example, the variable has been addressed on %MW100, this is equivalent of reserving a %MW100:436 array. If you use this address again, it leads to a detected error when the project is compiled.</p>	

Parameter	Type	Description
		<p>This has been done to force the programmer to address the variable within the PLC working range. If this input is not mapped, diagnostic code 16#200E (Hex) appears at level 0.</p> <p>Use memory maps to reserve memory area for this variable (refer to memory segment figures in this section).</p>
LocatedArea***	QUANTUMPORTLOCATEDDATA16	<p>Due to internal requirements, the Quantum port needs some variables to be mapped.</p> <p>As a result, this input provides the port with a memory area that has the internal structure necessary for it to work. The internal data does not have to be checked, and do not modify this internal data.</p> <p>Declare this variable in the variable table and allocate an address to it; otherwise the port does not work. To do this, specify the variable address.</p> <p>The necessary data consists of 436 words. If, for example, the variable has been addressed on %MW100, this is equivalent of reserving a %MW100:436 array. If you use this address again, it leads to a detected error when the project is compiled. This has been done to force the programmer to address the variable within the PLC working range. If this input is not mapped, diagnostic code 16#200E (Hex) appears at level 0.</p> <p>Use memory maps to reserve memory area for this variable (refer to memory segment figures in this section).</p>
<p>* This parameter is available only with <i>ModbusPortQx80</i> and <i>ModbusPortM58X80</i>.</p> <p>** This parameter is available only with <i>ModbusEthernetPortQ</i>.</p> <p>*** This parameter is available only with <i>ModbusEthernetPortQ16</i>.</p>		

## Calculating the workMemory Array Size

For more details, refer to [Editing the WorkMemory Array Size](#), page 107.



# Modbus Client1 Profile

## What's in This Chapter

Description .....	33
DFB Representation .....	34
Inputs .....	34
Outputs .....	35
Inputs/Outputs .....	37

## Overview

This chapter describes the DFBs of the `Modbus Client1` profile.

## Description

### General

A communication client allows device data to be written or read through Modbus, and Ethernet Communication Protocols.

By using a communication client, you can access remote device data that cannot be accessed normally with components generated with the Control Expert solution. For example, this enables you to read/write from/to a variable speed drive parameter if this parameter is not available on the respective control block of the speed drive.

Access to this data is explicit, that is, you need to program the access to this data. This is different from the implicit access used in other communication, as with Ethernet IO Scanning, in which access needs to be configured but not programmed.

The *ModBusClient1Basic* and *ModBusEthernetClient1* DFBs send a read or write request for  $n$  registers on a Modbus communication bus and an Ethernet communication network.

The *ModBusClient1Basic* and *ModBusEthernetClient1* DFBs belong to the Modbus communication and Modbus TCP Ethernet profile.

## Function Description

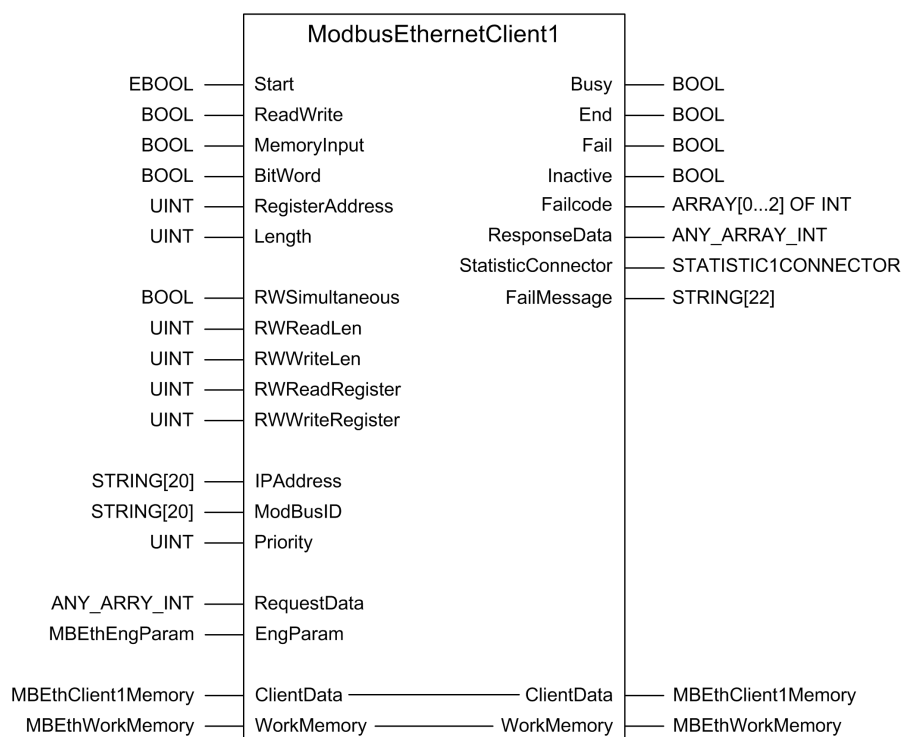
The main functions of the client DFB are described in the following table:

Function	Description
Read/Write	Enables you to select whether a parameter has read or write access.
Multiple register reading/writing	Enables several consecutive registers to be read (available only in <i>ModBusEthernetClient</i> and <i>ModBusClientBasic</i> ).
Diagnostic information management	Monitors detected transaction problems and identifies them on 3-levels to determine the source of the error detected.
Priorities	Enables you to define priorities for systems with multiple client accesses.
Statistics	Obtains the transactions OK/NOK status and their access times.

# DFB Representation

## Representation

The following figure represents the functional module of *Client*:



## Inputs

### Input Parameter Description

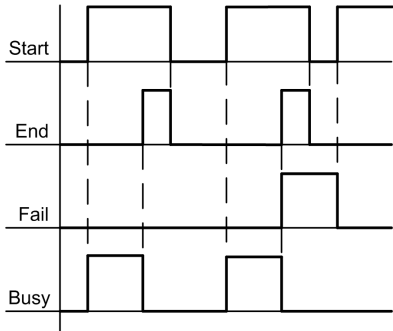
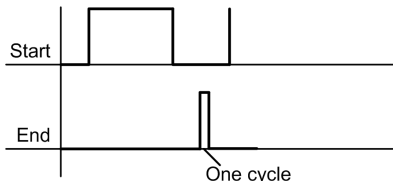
Parameter	Type	Description
Start	EBOOL	<p>1 = Resets detected errors. Indicates to the client that the data on the inputs is valid and a request to a server needs to be issued.</p> <p>0 = Triggers an ACK of the end-of-operation notification and the client is ready for the next cycle.</p> <p>When the signal is activated, it copies the parameters to the function so modifying the parameters has no effect.</p>
ReadWrite	BOOL	<p>1 = Write operation.</p> <p>0 = Read operation.</p>
MemoryInput	BOOL	<p>1 = Selects input zones (30000 or %IW registers).</p> <p>0 = Selects memory zones (40000 or %MW registers).</p> <p><b>NOTE:</b> CPU embedded ethernet ports cannot be used to read input registers through the client blocks.</p>
BitWord	BOOL	<p>1 = Operation performed on words.</p> <p>0 = Operation performed on bits.</p> <p><b>NOTE:</b> Bit operation is not available if the client block is used with <i>ModBusEthernetPortQ</i>, <i>ModBusEthernetPortQ16</i> and <i>ModBusPortQx80</i> port block.</p>
RegisterAddress	UINT	The address of the Register on which operation to be performed.
Length	UINT	Length of the data involved in the request that has been carried out.

Parameter	Type	Description	
		For Modbus ( <code>ModBusClientBasic</code> ) or Ethernet ( <code>ModBusEthernetClient</code> ) communication, the length can be in bits or words depending on the <code>BitWord</code> variable. The maximum length for words is 120.	
<code>RWSimultaneous</code>	BOOL	1= The operation considered is read and write simultaneously in one request.	
<code>RWReadLen</code>	UINT	When RW both is 1 then length of Registers to read max 125.	
<code>RWWriteLen</code>	UINT	When RW both is 1 then length of Registers to write max 100.	
<code>RWReadRegister</code>	UINT	When RWBoth is 1 then address of starting Register to read.	
<code>RWWriteRegister</code>	UINT	When RWBoth is 1 then address of starting Register to write.	
<code>IPAddress</code>	STRING[20]	IP address of device in format {'IP1.IP2.IP3.IP4'}.	
<code>ModbusID</code>	STRING[20]	Modbus address. Refer to the <a href="#">Ethernet Technology</a> , page 88 to see examples.  Depending on the platform, the following definitions apply:	
<code>Priority</code>	UINT	Command priority. The lower the value, the higher the priority, that is, 0 is the maximum priority. Any value is valid.  If you integrate a Modbus client together with any element from the Control Device library, consider the following priorities to not interfere with these elements:	
	<b>Variable value</b>	<b>Description</b>	
	1	Control commands. They have maximum priority.	
	2	Command confirmation. The read operations are carried out to confirm that the commands have been executed.  Medium priority.	
	3	Data reading.	
		If these priority levels are not observed, the client requests do not execute or the client does not allow other requests to be transmitted.  If you do not integrate it together with other Control Device library elements, you can define any priority levels that you want. Otherwise, use priority levels greater than or equal to 4.	
<code>RequestData</code>	ANY_ARRAY_INT	Data to be written	
<code>EngParam</code>	<code>MBEthEng-Param</code>	Engineering parameters	
	<b>Parameter name</b>	<b>Type</b>	<b>Description</b>
	<code>CommFailRetries</code>	INT	Number of retries in case of a communication detected failure
	<code>Timeout</code>	TIME	Time for which client will wait for a response, once the request is initiated, before moving to detected fail state.
	<code>InActivityTime</code>	TIME	The period for which the client will not initiate a request after the retries are expired.
	<code>ResetDataOnFail</code>	BOOL	Reset the data on detected fail of request instead of holding.

## Outputs

### Output Parameter Description

Parameter	Type	Description
<code>Busy</code>	BOOL	Activated while a request is taking place.  1 = Indicates that the client is busy, and that you cannot make new requests.
<code>End</code>	BOOL	Activated when a request cycle ends.

Parameter	Type	Description												
		1 = Indicates that the operation has ended (with or without a detected error). This signal is acknowledged (ACK) by setting the <i>Start</i> signal to Low.												
Fail	BOOL	<p>Activated when a detected error occurs in the request transmission.</p> <p>1 = Indicates that the operation is unsuccessful.</p> <p>To reset the detected error, activate the client again by setting the <i>Start</i> signal to High.</p> <p>Timing diagram:</p>  <p><b>NOTE:</b> If for any reason the <i>Start</i> signal is FALSE when the <i>End</i> signal is activated, only an edge for one cycle of the <i>End</i> signal is produced. This can result in the program (depending on how it is designed) not detecting the <i>End</i> signal, which is why this should not be allowed. The corresponding behavior would be as follows:</p>  <p>One cycle</p>												
Inactive	BOOL	The DFB is in inactive period (Requests will not be generated when this pin is logical high).												
FailCode	ARRAY [0..2] OF INT	Indicates the last detected error that took place according to 3 detected error levels, <a href="#">page 71</a> .												
	<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>FailCode[0]</td><td>INT</td><td>Error code</td></tr><tr><td>FailCode[1]</td><td>INT</td><td>Type of error</td></tr><tr><td>FailCode[2]</td><td>INT</td><td>Type of error in the device (To be used by the devices)</td></tr></table>	Parameter	Type	Description	FailCode[0]	INT	Error code	FailCode[1]	INT	Type of error	FailCode[2]	INT	Type of error in the device (To be used by the devices)	
Parameter	Type	Description												
FailCode[0]	INT	Error code												
FailCode[1]	INT	Type of error												
FailCode[2]	INT	Type of error in the device (To be used by the devices)												
ResponseData	ANY_ARRAY_INT	<p>Response received from the device.</p> <p><b>NOTE:</b> The data returned by the device (if there is any data), will be available in the <i>ResponseData</i>, irrespective of the status of the request (success or detected failure).</p>												

Parameter	Type	Description
StatisticConnector	STATISTIC1CONNECTOR	The information data is used to obtain network statistics (requests carried out, time between requests, and so on). This structure has been created for its use together with the <code>StatisticCounter</code> module in Communication library.  The following table describes the <code>StatisticConnector</code> :
	<b>Parameter</b>	<b>Type</b> <b>Description</b>
	Start	BOOL      1 = Operation has started.
	EndOK	BOOL      1 = Operation has ended correctly.
	EndNOK	BOOL      1 = Operation has ended with a detected error.
	TotalTime	DINT      Total time taken for the current request.
FailMessage	STRING[22]	Indicates the cause of detected failure.  <b>NOTE:</b> The data returned by the device (if there is any data), will be available in the <code>ResponseData</code> , irrespective of the status of the request (success or detected failure).

## Inputs/Outputs

### Input/Output Parameter Description

Parameter	Type	Description
ClientData	MBEth-Client1Data	All the client details are available within this structure  <b>NOTE:</b> Modifying the <code>ClientData</code> input/output pin will result in unexpected behavior of the DFB.
WorkMemory	MBEthWorkMemory	Array is used for communication. This variable is used in a <code>CanPort/ModBusPort/ModBusEthernetPort</code> , which serializes client requests in an optimum manner.

### MBEthClient1Data DDT Structure

Parameter	Type	Description
Data	ARRAY[0..125] OF INT	Data buffer.
IPAddress	string[20]	IPAddress of the device {IP1.IP2.IP3.IP4}.
ModbusID	string[20]	ModBusID
Length	UINT	Length of data to be written.
Priority	UINT	Priority of the request.
ErrorCode	INT	Detected error Information.
ErrorCode1	INT	Type of detected error.
Retries	INT	Number of retries.
ReadLength	INT	Length of data received from port in bytes.
OrderNumber	INT	Order number of request.
Status	WORD	The list of statuses are: <ul style="list-style-type: none"> <li>0- Idle</li> <li>1 -Client has placed a request.</li> <li>2 - Port is processing the request.</li> <li>3 - Port has processed without a detected error.</li> </ul>

Parameter	Type	Description
		<ul style="list-style-type: none"> <li>4 - Port has processed with detected error.</li> </ul>
ServiceID	BYTE	MBTCP ServiceID.
ReadWrite	BOOL	0 = Read operation 1 = Write operation.
MemoryInput	BOOL	0 = Operation on memory type register (%M) 1= Operation on input type register (%I).
RegisterAddress	UINT	The address of the register on which operation to be performed.
BitWord	BOOL	0= Operation on Bits (%I,%M), 1= Operation on registers (%IW,%MW).
RWSimultaneous	BOOL	1= The operation considered is read and write simultaneously in one request.
RWReadLen	UINT	When RW both is 1 then length of registers to read max 125.
RWWriteLen	UINT	When RW both is 1 then length of registers to write max 100.
RWReadRegister	UINT	When RW both is 1 then address of starting register to read.
RWWriteRegister	UINT	When RW both is 1 then address of starting register to write.

## MBEthWorkMemory DDT Structure

Parameter	Type	Description
<i>Header</i>	ARRAY [0..15] OF INT	Common data.
	<b>Parameter name</b>	<b>Type</b>
	Header[0]	INT
	Header[1]	INT
	Header[2]	INT
	Header[3]	INT
	Header[4]	INT
	Header[5]	INT
	Header[6]	INT
	Header[7]	INT
	Header[8]	INT
	Header[9]	INT
	Header[10]	INT
	Header[11]	INT
	Header[12]	INT
	Header[13]	INT
	Header[14]	INT
	Header[15]	INT
ClientDataReference	ARRAY[0..3] OF MBEthSocketData	Array of client references.

## MBEthSocketData DDT Structure

Parameter	Type	Description	
<i>ClientDataReference[0]</i>  <b>NOTE:</b> ClientDataReference parameters are available from ClientDataReference[0]...ClientDataReference[3].	<i>ClientData</i>	Reference of the client data.	
	<b>Parameter name.</b>	<b>Type.</b>	<b>Description.</b>
	<i>ClientData</i>	REF_TO MBEthClient1Data	MBTCP Client Data.
	Status	WORD	Status of the socket( 00 - Idle, 01- Client has sent data, 11 -Port has processed, 10 - Port is processing the request).

# Modbus Port1 Profile

## What's in This Chapter

Description .....	40
DFB Representation .....	41
Inputs .....	42
Outputs .....	43
Inputs/Outputs .....	44

## Overview

This chapter describes the DFBs of the `Modbus Port1` profile.

## Description

### General

A Port DFB is a function which is capable of serializing and managing requests sent to a physical medium working on Modbus TCP/IP, Modbus serial on local rack or Modbus serial on X80 rack.

A client is a Modbus communication (ATV, PM, ATS, and so on) device DFB or a generic read/write DFB used to communicate a device on the physical medium (clients and scanners).

The basic operation consists of various clients storing requests in a memory that is managed by the port (Modbus, or Ethernet). The port takes out requests from the queue based on a defined algorithm that manages priorities and waiting times and sends them to the appropriate destination after the port returns the response of the destination to the client that generated the request.

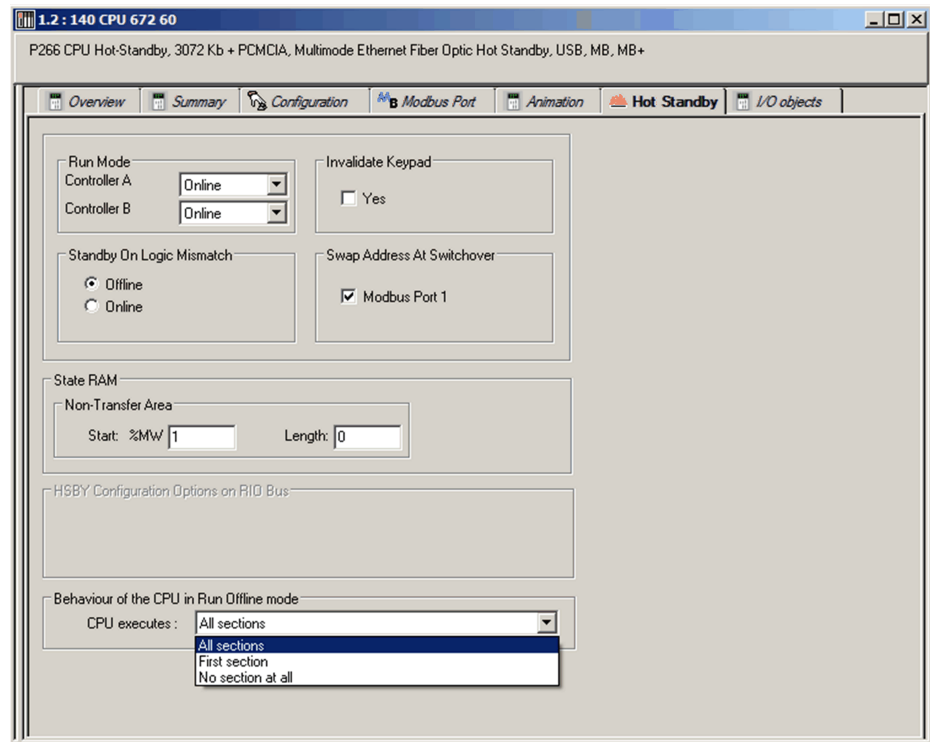
Use a Port DFB for any messaging requests. A minimum of 1 port per controller hardware is necessary to effect communication of this type.

#### NOTE:

- In a serial Modbus (RTU) project, use at least 1 Port DFB because communication on a Modbus takes place through messaging by definition.
- If the General Purpose library for device is used with Modbus TCP/IP networks (IOScanning), Port DFBs are not required for their control because these devices exchange cyclic data through IOScanning technology on Modbus TCP/IP. Any information that is not included in the Device library DFBs needs to be requested through acyclic exchange, that is, through requests sent to a Port DFB.
- New device DFB can be added to the existing explicit communication (Port DFB) while controller is running. To add a new device DFB user can instantiate a new port DFB which is mapped to the same physical port.

**NOTE:** For Port and its client DFBs to work as expected in HSBY controller, it is mandatory that these DFBs should not run in standby controller. This configuration can be done in the HSBY controller configuration as shown below.





Depending on the technology used, the function name varies to enable you to identify the product range to which the function belongs during generation:

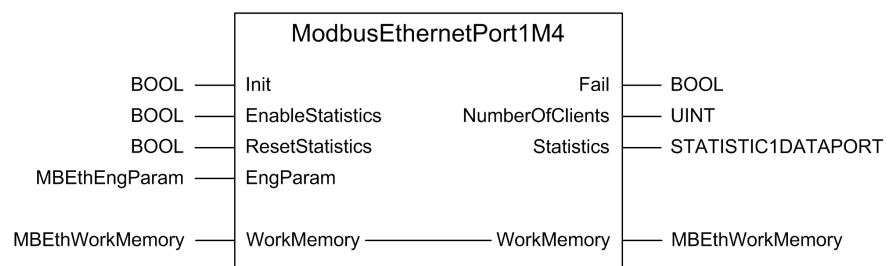
These functions are further classified to support, maximum number of concurrent operations based on Communication or CPU module. The table describes which port function to be used with the controller.

Communica- tion	M340	M580	Quantum	X80	Maximum Simultaneous Sends
Modbus RTU			—	<i>Modbus- Port1QX80</i>	4
	—	—	—	<i>Modbus- Port1M58X80</i>	4
ModbusTCP/IP	<i>ModbusEthernetPort1M</i>		—	—	4
	<i>ModbusEthernetPort1M16</i>		—	—	16

## DFB Representation

## Representation

The following figure represents the functional module of *ModbusEthernetPort*:



# Inputs

## Input Parameter Description

### ⚠ WARNING

#### UNINTENDED EQUIPMENT OPERATION

Use *InitRequest* pin only when the devices connected to *Port* function block are not running.

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

Parameter	Type	Description	
Init	BOOL	Reset all the current processing request of the Ethernet client. On triggering this pin, client connected to the port block may go into detected fail state. This input pin works on rising edge. <b>NOTE:</b> This input pin has to be used only when the PLC data restore operation is performed.	
EnableStatistics	BOOL	Enable the statistics counters	
ResetStatistics	BOOL	1 = Resets the data on the <i>Statistics</i> output variable to its default values (0).  The input is level-based, that is, <i>Statistics</i> remains 0 as long as the input is TRUE.	
EngParam	MBEthPortEngParam	Engineering parameters	
	Parameter name	Type	Description
	PortAddress	STRING [14]	Communication network name (Ex. 'Ethernet_1') or hardware address ('rack.module.channel').
	SimultaneousSends	INT	Number of simultaneous sends.  The maximum and default simultaneous client request processed by <i>ModbusEthernetPort1M4</i> is '4'.
	Timeout	TIME	Time to wait a response, after this port will send a detected timeout error to the requesting client.

#### NOTE:

- As per the design of the port block, there is no limit on number of client instances.
- The response time is proportional to the number of client instances.
- Increase in the client instances will have an impact on the response.
- If the port *EngParam* parameter is not configured by the user, it will consider the default CPU port address (0.0.3), Simultaneous sends '3' and Port timeout #t3s.

# Outputs

## Output Parameter Description

Parameter	Type	Description
Fail	BOOL	<p>1 = Indicates that a detected error has occurred while transmitting a request. It is only an indication, and the port process continues. The bit does not have to be reset, and it cannot be reset. As soon as a request is issued without a detected error, it is set to 0.</p> <p>If <code>Fail</code> bit is set to 1, following can be the possibilities:</p> <ul style="list-style-type: none"> <li>If <code>Workmemory</code> area is less than the <code>WantedArraySize</code></li> <li>If the <code>PortAddress</code> is not correctly configured.</li> </ul> <p>Check the diagnostic codes in each DFB belonging to devices or clients/scanners.</p>
NumberOfClients	UINT	Number of clients connected to port.
Statistics	STATISTIC1DATA-PORT	<p>Holds a structure with statistical data on the operation of the Port DFB. This data is useful for debugging the application.</p> <p>The information obtained in this data structure is the statistic for the requests managed by this Port DFB. The clients associated with this DFB through the <code>WorkMemory</code> array send the requests.</p> <p>The following table describes the <code>StatisticData</code> type:</p>
	<b>Name</b>	<b>Type</b> <b>Description</b>
	RequestsSent	UDINT      Total number of requests sent.
	RequestsOk	UDINT      Total number of requests that ended correctly.
	RequestsError	UDINT      Total number of requests that ended in a detected error.
	MinTime	UDINT      Minimum time required to end a request (msec).
	AvgTime	UDINT      Average time required to end a request (msec).
	MaxTime	UDINT      Maximum time to process a request in the communication port (ms).
	MinCycleNumber	UDINT      Minimum number of PLC cycles to process a request in the communication port.
	AvgCycleNumber	UDINT      Average number of PLC cycles to process a request in the communication port.
	MaxCycleNumber	UDINT      Maximum number of PLC cycles to process a request in the communication port.
	MinTotalTime	DINT      Minimum amount of time spent on the requests sent (ms).
	AvgTotalTime	DINT      Average amount of time spent on the requests sent (ms).
	MaxTotalTime	DINT      Maximum amount of time spent on the requests sent (ms).
	RequestsperSecond	UINT      Number of requests sent per second.
	SimultaneousRequests	UINT      Number of simultaneous requests.

# Inputs/Outputs

## Input/Output Parameter Description

Parameter	Type	Description
WorkMemory	MBEthWorkMemory	<p>This array is for use with a Modbus-RTU, or Modbus TCP/IP Port DFB holds the read/write requests of the clients/scanners.</p> <p>The <code>WorkMemory</code> array is the work memory or link between the serial Modbus clients/scanners and an Ethernet port DFB.</p> <p>During the first scanning cycle, the clients/scanners or devices reserve read/write %Mwords from this array based on the exchange words required by each DFB. After they have been reserved, the Port DFB counts them, and as a result specifies and identifies how many pieces of data are to be serialized.</p> <p>Adjust the <code>WorkMemory</code> parameter to set:</p> <ul style="list-style-type: none"> <li>The size of the data exchange area between the device and client DFBs.</li> <li>The Port DFB to the size needed for the elements that are being used.</li> </ul> <p>The system is not limited by fixed sizes or unused memory.</p>

## MBEthWorkMemory DDT Structure

Parameter	Type	Description
<i>Header</i>	ARRAY [0..15] OF INT	Common data.
	<b>Parameter name</b>	<b>Type</b>
	Header[0]	INT
	Header[1]	INT
	Header[2]	INT
	Header[3]	INT
	Header[4]	INT
	Header[5]	INT
	Header[6]	INT
	Header[7]	INT
	Header[8]	INT
	Header[9]	INT
	Header[10]	INT
	Header[11]	INT
	Header[12]	INT
	Header[13]	INT
	Header[14]	INT
	Header[15]	INT
ClientDataReference	ARRAY[0..3] OF MBEthSocketData	Array of client references.

## MBEthSocketData DDT Structure

Parameter	Type	Description
<i>ClientData</i>	REF_TO MBEthClient1Data	Reference of the client data.
Status	WORD	Status of the socket (00 - Idle, 01 - Client has sent data, 11 - Port has processed, 10 - Port is processing the request).

# Scanner Profile

## What's in This Chapter

Description .....	46
DFB Representation .....	47
Inputs .....	48
Outputs .....	49
Inputs/Outputs .....	50
Calculating the Array Size of the Input and Output Parameters .....	50

## Overview

This chapter describes the DFBs of the `Scanner` profile.

## Description

### General

A communication scanner is a function block that uses an internal client to send several sequential requests to different memory positions in the remote device. It enables you to program up to 10 read/write requests that will be sent sequentially, activated either cyclically or with 1 single cycle.

By using a communication scanner, you can access a set of memory addresses (consecutive or non-consecutive) in remote devices which cannot be accessed normally with components generated with the Control Expert solution – just like with a client, but by programming several accesses to different memory positions in the device sequentially and individually, without the use of multiple clients.

Access to this data is explicit, that is, you need to program access to this data. This is different from the implicit access used in other communication, as with Ethernet IO Scanning, in which access needs to be configured but not programmed.

The *ModBusScanner* and *ModBusEthernetScanner* DFBs periodically refreshes read or write requests for *n* registers issued to a single node on a Modbus communication bus and Ethernet communication.

The *ModBusScanner* and *ModBusEthernetScanner* DFBs belong to the Modbus communication profile.

## Function Description

The main functions of the scanner DFB are described in the following table

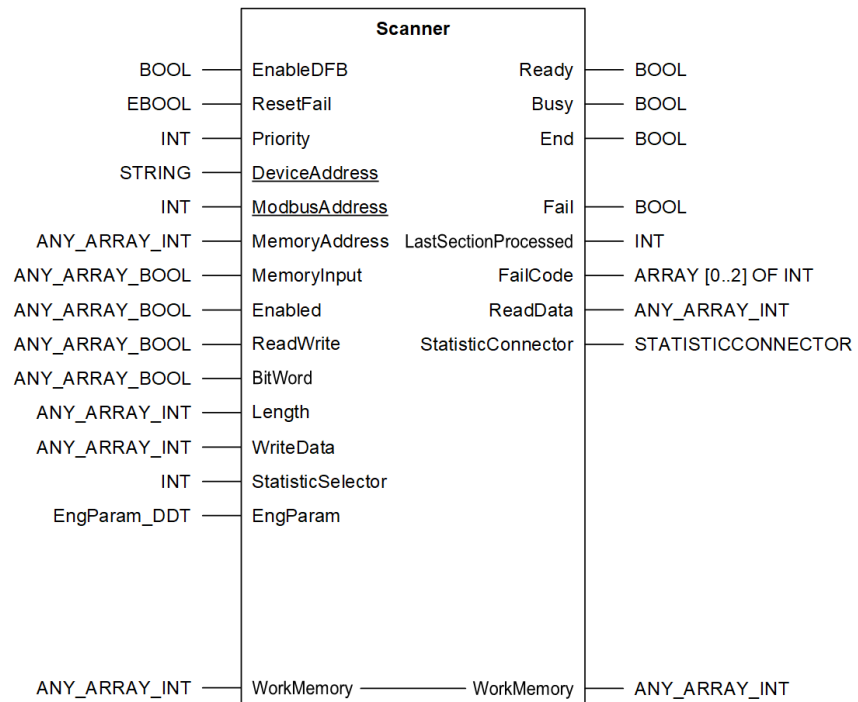
Function	Description
Cyclic reading/writing	Allows reading or writing with cycling timing.
Multiple register reading/writing	Allows several consecutive registers to be read with each line. Available only in <i>ModBusEthernetClient</i> and <i>ModBusClientBasic</i> .
Individual activation of scanner requests	Enables you to select individually which requests need to be activated (out of the 10 available ones).
Diagnostic information management	Monitors incorrect transactions and identifies them on 3-levels to determine the source of the error detected.

Function	Description
Priorities	Enables you to define priorities for systems with multiple scanner accesses.
Statistics	Obtains the OK/NOK status of the transactions and their access times.

## DFB Representation

### Representation

The following figure represents functional module of *Scanner*:



**NOTE:** The underlined parameters are specific for some components.

The table shows the parameters available for specific DFBs:

Parameters		Components	
		Modbus	Modbus TCP Ethernet
		<i>ModBusScanner</i>	<i>ModBusEthernetScanner</i>
Inputs	DeviceAddress	–	X
	ModbusAddress	X	–
X: Parameter is available			
–: Parameter is not available			

# Inputs

## Input Parameter Description

Parameter	Type	Description
EnableDFB	BOOL	1 = Enables communication. 0 = Initializes scanner.
ResetFail	EBOOL	1 = Resets a communication interruption or incorrect parameter configuration.
Priority	INT	Command priority. The lower the value, the higher the priority, that is, 0 is the maximum priority. Any value is valid.
DeviceAddress*	STRING [26]	Device address. Refer to the Ethernet Technology, page 88 to see an example.  Depending on the platform, the following definitions apply:
	Platform	IP addressing DeviceAddress (variable)
	M340/M580/Quantum	'{IP}ID'
		<b>NOTE:</b> Do not omit punctuation marks.
ModbusAddress**	INT	Address of the Modbus slave that the client needs to access. The range of possible values goes from 1 to 255.
MemoryAddress	ANY_ARRAY_INT	Each position in the array holds the memory address where the requested scanner operation starts.  Any value is valid for the master because the slave needs to validate the address based on its memory map.  You can enter values as hexadecimal or decimal values. These indexes are expressed in decimal values in device manuals. To express the index as a hexadecimal value, use the string 16# before the index, that is, 16#index.
MemoryInput	ANY_ARRAY_BOOL	Array of memory or input operations <ul style="list-style-type: none"> <li>MemoryInput[x] = 0: Selects memory zones (40000 or %MW registers)</li> <li>MemoryInput[x] = 1: Selects input zones (30000 or %IW registers)</li> </ul>
Enabled	ANY_ARRAY_BOOL	Array of enabled operations <ul style="list-style-type: none"> <li>Enabled[x] = 0: Operation is skipped</li> <li>Enabled[x] = 1: Operation is performed</li> </ul>
ReadWrite	ANY_ARRAY_BOOL	Array of read or write operations <ul style="list-style-type: none"> <li>ReadWrite[x] = 0: Read operation</li> <li>ReadWrite[x] = 1: Write operation</li> </ul>
BitWord	ANY_ARRAY_BOOL	Array of operations <ul style="list-style-type: none"> <li>BitWord[x] = 0: Operation is performed in bits</li> <li>BitWord[x] = 1: Operation is performed in words</li> </ul>
Length	ANY_ARRAY_INT	Integer array. Length of the data involved in the request that has been carried out.  For Modbus (ModBusClientBasic) or Ethernet (ModBusEthernetClient) communication, the length can be in bits or words depending on the BitWord variable. The maximum length for words is 120 (100 words in Quantum platform).  <b>NOTE:</b> Modifying the lengths of the write operations for each request misalign the WriteData array.
WriteData	ANY_ARRAY_INT	Integer array. The integers hold the data to be written in each scanner line. This table holds the data to be written for the read requests. The size of the variables used for each request is marked by the Length variable. As a result, each write request reserves its zone in this array.
StatisticSelector	INT	Variable used to obtain network statistics (requests carried out, time between requests, and so on). This data provides information for using the StatisticConnector with the StatisticCounter module in the Communication library.
	Variable value	Description
	1	Statistics of complete scanner cycles.



Parameter	Type	Description
	2	Statistics of each scanner line
EngParam	EngParam_DDT, page 49 EngParamEMScanner/ EngParamMBScanner	Engineering parameters.
* This parameter is available only with <i>ModbusEthernetScanner</i> .		
** This parameter is available only with <i>ModbusScanner</i> .		

## EngParam\_DDT

Parameter	Type	Description
Refresh	TIME	Time that the scanner waits before it repeats an operation cycle.
MaxReadSize	INT	This configuration parameter indicates how many words are read in the request issued by this client. If no value is specified, the maximum possible value is used by default (125 words for Ethernet/Modbus). This parameter is used for calculating and managing the work memory area ( <i>WorkMemory</i> ).  <b>NOTE:</b> If several requests are made with the same client, use the maximum length of the read requests to be issued as the value of the parameter.
MaxWriteSize	INT	This configuration parameter indicates how many words are written in the largest request issued by this client. If no value is specified, the maximum possible value is used by default (125 words for Ethernet/Modbus). This parameter is used for calculating and managing the work memory area ( <i>WorkMemory</i> ).  <b>NOTE:</b> If several requests are made with the same client, use the maximum length of the write requests to be issued as the value of the parameter.

## Outputs

### Output Parameter Description

Parameter	Type	Description
Ready	BOOL	1 = The scanner is ready for operation.
Busy	BOOL	1 = The scanner is busy waiting for the response to a request.
End	BOOL	1 = The operation in progress has ended.
Fail	BOOL	1 = Detected error in the function during the current request.  If communication with the slave does not respond, it stops requests until a <i>ResetFail</i> is carried out.
LastSectionProcessed	INT	If <i>End</i> is true, this output shows the index of the scanner operation that has ended.
FailCode	ARRAY [0..2] OF INT	Indicates the last detected error that took place according to 3 detected error levels, page 71.
ReadData	ANY_ARRAY_INT	Data read by the scanner if the last operation was a read operation. This zone is common to read requests that have been configured in the parameters. As a result, the data is overwritten with each new processed read request.  <b>NOTE:</b> <i>ReadData</i> holds the read data for <i>LastSectionProcessed</i> hence, this data is updated every time a new read operation is processed. The program unloads this data from this array if needed after the next request. Use <i>End</i> variable to unload data when read request ended ( <i>LastSectionProcessed</i> AND <i>End</i> ).

Parameter	Type	Description
StatisticConnector	STATISTICCONNECTOR	The information data is used to obtain network statistics (requests carried out, time between requests, and so on). This structure has been created for its use together with the <code>StatisticCounter</code> module in Communication library.  The following table describes the <code>StatisticConnector</code> :
	<b>Parameter</b>	<b>Type</b> <b>Description</b>
	Start	BOOL 1 = Operation has started.
	EndOk	BOOL 1 = Operation has ended correctly.
	EndNok	BOOL 1 = Operation has ended with a detected error.
	PartialTime	DINT Partial time.

## Inputs/Outputs

### Input/Output Parameter Description

Parameter	Type	Description
WorkMemory	ANY_ ARRAY_ INT	Array is used for communication. This variable is used in a <code>ModBusPort/ModBusEthernetPort</code> , which serializes client requests in an optimum manner.

## Calculating the Array Size of the Input and Output Parameters

### Description

Declare all the configurable array parameters with the array dimensions starting from 0.

`MemoryAddress`, `MemoryInput`, `Enabled`, `ReadWrite`, `BitWord`, and `Length` parameters: The size of these parameters depends on the number of requests configured. If the scanner is programmed for 2 requests their array size should be [0..1], if programmed for 3 requests then the array size should be [0..2].

The configuration of properties of a request is at the same index position across all the above mentioned parameters.

`WriteData` parameter: Each request reserves its zone in this array, hence the size of this input parameter is the summation of the length of all the requests programmed, regardless of any of the requests being Read or Write type.

For Example, If there are 3 requests with the following configuration: `ReadWrite[0]=0`, `ReadWrite[1]=0`, `ReadWrite[2]=1` (The first two requests are Read, the third request is Write type), `Length[0]=3`, `Length[1]=5`, and `Length[2]=8` Then the total length is 16, and the `WriteData` array size should be [0..15], which is 16 elements. The data for the write request (third request) should be assigned from `WriteData[8..15]`.

`ReadData` parameter: The size of this array should be the size of response from the largest read request programmed.

`WorkMemory` Parameter: Refer to the topic `Editing the WorkMemory Array Size`, page 107.

# ModBusGateway - Serial Modbus-Ethernet Gateway

## What's in This Chapter

Description .....	51
DFB Representation .....	51
Inputs .....	52
Outputs .....	53
Inputs/Outputs .....	54

## Overview

This chapter describes the `ModBusGateway` DFB.

## Description

### General

The `ModBusGateway` is the DFB that functions as a bridge between serial devices and Ethernet buses.

By using a gateway (`TSXETG100`), you can establish communication between a CPU with Ethernet ports and devices that implement only an RS485 interface.

## Function Description

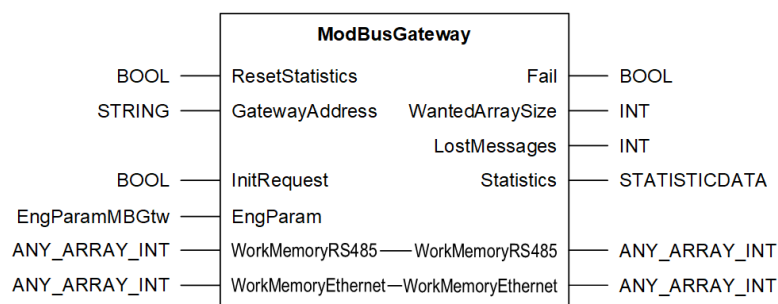
The main functions of the DFB are described in the following table:

Function	Description
Request conversion	Enables read/write requests to be issued to devices on a 485 bus through an Ethernet port (available only for <code>ModBusClientBasic</code> and <code>ModBusScanner</code> ).
Statistics	Obtains the OK/NOK status of the transactions and their access times.

## DFB Representation

### Representation

The following figure represents the functional module of `ModBusGateway`:



# Inputs

## Input Parameter Description

### ⚠ WARNING

#### UNINTENDED EQUIPMENT OPERATION

Use *InitRequest* pin only when the devices connected to *Port* function block are not running.

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

Parameter	Type	Description
ResetStatistics	BOOL	1 = Resets the data on the <i>Statistics</i> output variable to its default values (0). The input is level-based, that is, statistics remains 0 as long as the input is TRUE.
GatewayAddress	STRING[26]	IP address for the gateway where the serial Modbus devices are physically located. Depending on the platform used and the position in which the Ethernet port is located inside the PLC rack, enter IP addressing in the following formats:
	Platform	Gateway addressing
	M340/M580/ Quantum	'{IP}'
InitRequest	BOOL	Reset all the current processing request of the Modbus client. On triggering this pin, client connected to the gateway block may go into fail state. This input pin works on rising edge. <b>NOTE:</b> This input pin has to be used only when the PLC data restore operation is performed.
EngParam	EngParamMBGtw, page 52	Engineering parameters

The slot indicates the position occupied by the Ethernet port inside the PLC rack.

**NOTE:** Follow the naming conventions (periods, brackets) to configure the Modbus slave address.

## EngParamMBGtw

Parameter	Type	Description
SimultaneousSends	INT	Indicates how many requests are in the gateway queue. You can configure this input.  The value ranges from 1 to 4. The more requests in the gateway queue, the better the general performance of the system. You need to consider the following: <ul style="list-style-type: none"> <li>The maximum number of requests in the CPU buffer.</li> <li>The maximum number of requests for a gateway queue.</li> <li>Priority management does not consider requests that are queued in the hardware buffer, which enables a high-priority command to wait until lower-priority commands that are already in the queue are sent first. This action results in a longer response time for high-priority commands.</li> </ul>

# Outputs

## Output Parameter Description

Parameter	Type	Description	
Fail	BOOL	<p>1 = Indicates that a detected error has occurred while transmitting a request. It is only an indication, and the port process continues.</p> <p>The bit does not have to be reset, and it cannot be reset. As soon as a request is issued without an error detected, it is set to 0.</p> <p>Check the diagnostic codes in each DFB belonging to devices or serial Modbus clients.</p>	
WantedArraySize	INT	<p>Indicates the size necessary for the <code>WorkMemoryRS485</code> array to function correctly.</p> <p>Declare the array with the minimum size of <code>[0..WantedArraySize-1]</code> because arrays start on 0.</p> <p>If you add additional serial Modbus (RTU) clients or devices to the program, the size of this variable increases making it necessary to monitor the <code>WorkMemoryRS485</code> array.</p> <p><b>NOTE:</b> This variable is calculated during the first scanning cycle so adding serial Modbus (RTU) clients or devices during online mode has the following effects:</p> <ul style="list-style-type: none"><li>• The size of the array does not increase.</li><li>• The <i>ModbusGateway</i> DFB does not consider requests from these new clients.</li></ul>	
LostMessages	INT	<p>Indicates the number of messages that have been lost (that the client has not received).</p> <p>This value needs to be 0. Otherwise, the user has written an incorrect program (a <code>ModbusClientBasic/ModbusScanner</code> instance is probably not executed each cycle).</p> <p>If messages are lost, the user has to diagnose the reason (either hardware wiring or incorrect programming).</p>	
Statistics	STATISTICDATA	<p>Holds a structure with statistical data on the operation of the <code>ModBusGateway</code> DFB. This data is useful for debugging the application.</p> <p>The information obtained in this data structure is the statistic for the requests managed by this <code>ModBusGateway</code> DFB. The clients that are associated with this DFB by using the <code>WorkMemoryRS485</code> array send requests.</p> <p>The following table describes the <code>StatisticData</code> type:</p>	
	Name	Type	Description
	RequestsSended	DINT	Total number of requests sent.
	RequestsOk	DINT	Total number of requests that ended correctly.
	RequestsError	DINT	Total number of requests that ended in an error detected.
	MinTime	DINT	Minimum time required to end a request (msec).
	AvgTime	REAL	Average time required to end a request (msec).
	MaxTime	DINT	Maximum time required to end a request (msec).
	LastTime	DINT	Time required to end the last request (msec).
	CurrentTime	DINT	Time spent on processing the current request (0 if no request is in progress).
	RequestsSecond	DINT	Number of requests can be sent per second (calculated with the <code>AvgTime</code> value).
	LastCycleNumber	DINT	Number of controller cycles that the last request took to be executed.
	CurrentCycleNumber	DINT	Number of controller cycles that the current request has taken so far (0 if no request is in progress).
	TimeOnQueue	DINT	Time that the last request has been waiting in the queue before sending (only useful for client statistics).

# Inputs/Outputs

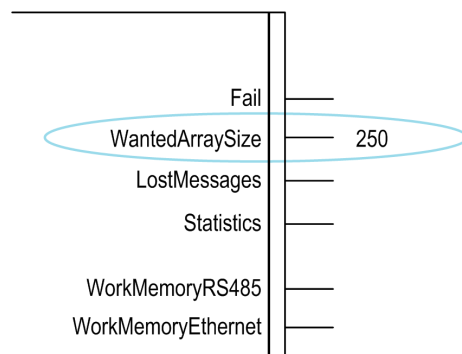
## Input/Output Parameter Description

Parameter	Type	Description
WorkMemoryRS485	ANY_ ARRAY_ INT	<p>This array holds the read/write requests of the clients/scanners.</p> <p>The <code>WorkMemoryRS485</code> array is the work memory or link between the serial Modbus clients/scanners and an Ethernet port DFB.</p> <p>During the first scanning cycle, the devices reserve read/write %Mwords from this array based on the exchange words required by each DFB. The <code>ModbusGateway</code> DFB counts them, and as a result specifies and identifies how many pieces of data are to be serialized.</p> <p>Adjust the <code>WorkMemoryRS485</code> variable previously to set:</p> <ul style="list-style-type: none"> <li>The size of the data exchange area between the device and client DFB.</li> <li>The <code>ModbusGateway</code> DFB to the size needed for the elements that are used.</li> </ul> <p>The system is not limited by fixed sizes or unused memory.</p>
WorkMemoryEthernet	ANY_ ARRAY_ INT	<p>This variable holds the requests that are in the <code>WorkMemoryRS485</code> in Modbus TCP/IP requests.</p> <p>As a result, this variable needs to have a bigger size than the <code>WorkMemoryRS485</code> variable because each request holds more data, such as the destination IP address, and so on.</p> <p><b>NOTE:</b> The <code>WantedArraySize</code> output variable only indicates the size that is required for the <code>WorkMemoryRS485</code> variable. To know the appropriate size for the variable, check the <code>WantedArraySize</code> output variable of the Ethernet port DFB to which the <code>ModbusGateway</code> DFB is linked.</p>

## Calculating the `WorkMemoryRS485` Array Size

The necessary size for the array of the variable associated to `WorkMemoryRS485` is automatically calculated by the DFBs of the port shown by the `WantedArraySize` output.

Check if the size of the array for the `WorkMemoryRS485` variable of the `ModbusGateway` DFB, which has been calculated by the code generator, has a size greater than or equal to the one that the `WantedArraySize` variable requires.



The following steps explain how to check the value:

Step	Action
1	Execute the program with the calculated array.
2	Check the value returned by the <code>ModbusEthernetPortM</code> DFB.
3	Resize the array to the <code>WantedArraySize</code> value (the array needs to have a size of [0... <code>WantedArraySize</code> -1] as a maximum).

EGtwMB_GatewayAddr	string[25]
EGtwMB_LostMessages	INT
EGtwMB_MBWorkMemory	ARRAY[0..250] OF INT
EGtwMB_ResetStatistics	BOOL

The DFB rebuilds the serial Modbus requests that are in `WorkMemoryRS485` as Modbus TCP/IP requests, which are copied to the `WorkMemoryEthernet` variable.


# Ethernet IP Communication

## What's in This Part

EthernetIPPortMxx - Ethernet IP Port Profile .....	57
EthernetIPClient - Ethernet IP Client Profile .....	60
StatisticCounter1 - Statistic Counter Profile.....	64

## Overview

This part explains about Ethernet IP communication technology.

 **WARNING**

**LOSS OF CONTROL**

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

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



# EthernetIPPortMxx - Ethernet IP Port Profile

## What's in This Chapter

Description .....	57
DFB Representation .....	57
Inputs .....	58
Outputs .....	58
Inputs/Outputs .....	58

## Overview

This chapter describes the DFB of the Ethernet IP port profile.

## Description

### Overview

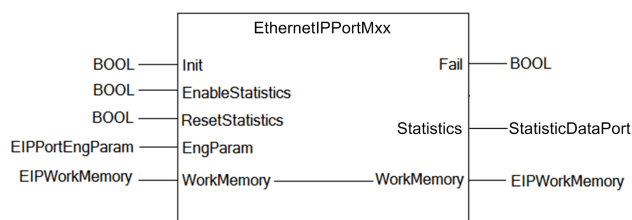
EthernetIPPortMxx DFB is a function block which is capable of serializing and managing requests sent to a physical device communicating on explicit messaging using Ethernet IP protocol.

The function of this DFB is to serialize requests generated by various clients connected to this port via the work memory to the appropriate physical destination. The `DATA_EXCH` function of Control Expert is used to generate EIP requests within this DFB. The received response is sent back to the respective clients who have generated the requests. This serialization of requests enables you to connect more number of clients to the same physical port.

## DFB Representation

### Description

The following figure represents the functional module of *EthernetIPPortMxx*



**NOTE:** **xx** can have the values of 4, 16, 32, 48, 64, 80 and 96. The values indicate the maximum number of **SimultaneousSends** supported by the function block.

## Inputs

### Input Parameter Description

Parameter	Type	Description
<code>Init</code>	BOOL	Initialize the <code>EthernetIPPortMxx</code> DFB.
<code>EnableStatistics</code>	BOOL	Enable the statistics counters.
<code>ResetStatistics</code>	BOOL	Reset the statistics.
<code>EngParam</code>	EIPPortEng-Param	Engineering parameters of the Ethernet IP port. <b>NOTE:</b> <ul style="list-style-type: none"> <li>If <b>SimultaneousSends</b> is 0, then the default value of 1 is considered by the DFB.</li> <li>If <b>Timeout</b> is configured as 0, then a default value of t#3s is considered by the DFB.</li> </ul>

### EIPPortEngParam DDT structure

Parameter	Type	Description
<code>PortAddress</code>	string[14]	Communication network name. For example <code>Ethernet_1</code> or hardware address <code>rack.module.channel</code> . <b>NOTE:</b> If <code>PortAddress</code> is empty, then the nearest Ethernet port to CPU will be considered for communication.
<code>SimultaneousSends</code>	INT	Number of simultaneous sends.
<code>Timeout</code>	TIME	Time to wait a response. Request time in the network/bus.

## Outputs

### Output Parameter Description

Parameter	Type	Description
<code>Fail</code>	BOOL	Displays if there are errors detected during communication.
<code>Statistics</code>	StatisticDataPort	Statistics of the data sent.

## Inputs/Outputs

### Input/Output Parameter Description

Parameter	Type	Description
<code>WorkMemory</code>	EIPWorkMemory	Memory Area which contains the client reference.

## EIPWorkMemory DDT Structure

Parameter	Type	Description
<i>Header</i>	Array [0..15] OF WORD	Common data.
ClientDataReference	Array[0..95] OF EIPSocketData	Array of client references.

## StatisticDataPort DDT Structure

Parameter	Type	Description
RequestsSent	UDINT	Number of requests sent including retries.
RequestsOk	UDINT	Number of requests that ended successfully.
RequestsError	UDINT	Number of requests that ended with detected failure.
MinTime	UDINT	Minimum time to process a request in the communication port (ms).
AvgTime	UDINT	Average time to process a request in the communication port (ms).
MaxTime	UDINT	Maximum time to process a request in the communication port (ms).
MinCycleNumber	UDINT	Minimum number of PLC cycles to process a request in the communication port.
AvgCycleNumber	UDINT	Average number of PLC cycles to process a request in the communication port.
MaxCycleNumber	UDINT	Maximum number of PLC cycles to process a request in the communication port.
MinTotalTime	DINT	Minimum amount of time spent on the requests sent (ms).
AvgTotalTime	DINT	Average amount of time spent on the requests sent (ms).
MaxTotalTime	DINT	Maximum amount of time spent on the requests sent (ms)
RequestspersSecond	UINT	Number of requests sent per second.
SimultaneousRe- quests	UINT	Number of active requests.

# EthernetIPClient - Ethernet IP Client Profile

## What's in This Chapter

Description .....	60
DFB Representation .....	61
Inputs .....	61
Outputs .....	62
Inputs/Outputs .....	62

## Overview

This chapter describes the DFB of the Ethernet IP client profile.

## Description

### Overview

The `EthernetIPClient` DFB facilitates EthernetIP services to be executed on the devices via Ethernet IP communication protocol. This function has provision to capture the parameters for the execution of the EIP service on specific device.

Access to the client data is explicit, that is, you need to program the access to this data. This is different from the implicit access used in other communication, as with Ethernet IO Scanning, in which access needs to be configured but not programmed.

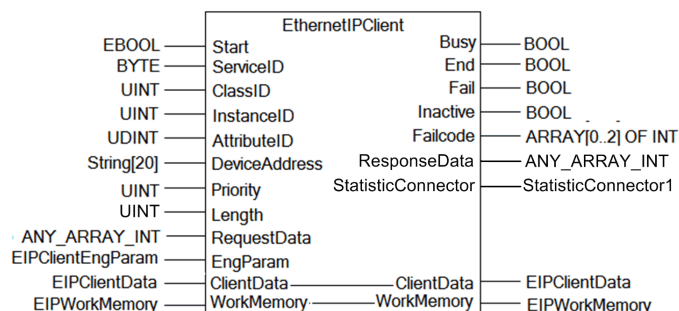
## Function Description

Function	Description
EIP Services	Enables user to execute various EIP services on devices.
Diagnostic information management	Monitors detected transaction problems and identifies them on two levels to determine the source of the error detected.
Priorities	Enables user to define priority for the client requests.
Statistics	Obtains the status of the request (OK/NOK) and its execution time.
Timeout	If the client is waiting after generating a request for a time greater than the Timeout configured, the client is declared inoperable.
Retries	If client is interrupted due to any detected error generated from <code>DATA_EXCH</code> block, the request gets retried for the configured retry count in the client.
Inactivity period	If the number of retries are expired then the client waits for the inactivity time configured before generating the next request.

# DFB Representation

## Description

The following figure represents the functional module of *EthernetIPClient*



## Inputs

## Input Parameter Description

Parameter	Type	Description
Start	EBOOL	Rising edge enables client to create request.
ServiceID	BYTE	EIP service ID.
ClassID	UINT	Class ID of the device on which the EIP service has to be executed.
InstanceID	UINT	Instance ID held by the <code>ClassID</code> on which the EIP service has to be executed.
AttributeID	UDINT	Attribute ID held by the <code>InstanceID</code> on which the EIP service has to be executed.
DeviceAddress	String[20]	IPAddress of device in format { IP1 . IP2 . IP3 . IP4 }.
Priority	UINT	Priority of the request. Lesser the value higher the priority.
Length	UINT	Length of the write data in bytes.
RequestData	ANY_ARRAY_INT	Data to be written along with EIP header.
EngParam	EIPClientEngParam	EIP Client Engineering parameters.

## EIPPortEngParam DDT structure

Parameter	Type	Description
CommFailRetries	INT	Number of retries in case of a communication interruption.
Timeout	TIME	Time for which client will wait for a response, once a request is initiated before moving to <i>Fail</i> state.  <b>NOTE:</b> If the <i>Timeout</i> is either configured t#0s or not configured, then client will be in <i>Busy</i> state until it gets a response from <i>EthernetIPPortMxx</i> DFB. In order to move to next state, you have to either initialize the <i>EthernetIPPortMxx</i> DFB or configure <i>Timeout</i> online.
InactivityTime	TIME	The period for which the client will not initiate a request after the retries are expired.

## Outputs

### Output Parameter Description

Parameter	Type	Description
<i>Busy</i>	BOOL	Indicates that the DFB is busy processing a request.
<i>End</i>	BOOL	Indicates that the DFB has completely the requested EIP service.
<i>Fail</i>	BOOL	Indicates the request has interrupted.  <b>NOTE:</b> All detected failed clients will be in ready state after port is initialized.
<i>Inactive</i>	BOOL	DFB is in inactive period (Request will not be generated when this pin is logical high).
<i>Failcode</i>	ARRAY[0..2] OF INT	<i>Failcode</i> of the last request which got interrupted.
<i>ResponseData</i>	ANY_ARRAY_INT	Response received from the device.  <b>NOTE:</b> The data returned by the device (if there is any data), will be available in the <i>ResponseData</i> , irrespective of the status of the request (success or detected failure).
<i>StatisticConnector</i>	StatisticConnector1	Statistic connector.

## Inputs/Outputs

### Input/Output Parameter Description

Parameter	Type	Description
<i>ClientData</i>	EIPClientData	All the client details are available within this structure.  <b>NOTE:</b> Modifying the <i>ClientData</i> input/output pin will result in unexpected behavior of the DFB.
<i>WorkMemory</i>	EIPWorkMemory	Memory area which contains the client reference.

## EIPWorkMemory DDT Structure

Parameter	Type	Description
<i>Header</i>	Array [0..15] OF WORD	Common data.
<i>ClientDataReference</i>	Array[0..95] OF EIPSocketData	Array of client references.

## EIPSocketData DDT Structure

Parameter	Type	Description
<i>ClientRef</i>	REF_TO EIPClientData	Reference of the client data.
<i>Status</i>	WORD	Status of the socket.

## EIPClientData DDT Structure

Parameter	Type	Description
<i>Data</i>	ARRAY[0..251] OF INT	Data buffer.
<i>IPAddress</i>	string[20]	IPAddress of the device { IP1 . IP2 . IP3 . IP4 } .
<i>AttributeID</i>	UDINT	Attribute ID of the request.
<i>InstanceID</i>	UINT	Instance ID of the request.
<i>ClassID</i>	UINT	Class ID of the request.
<i>Length</i>	UINT	Length of data to be written.
<i>Priority</i>	UINT	Priority of the request.
<i>ErrorCode</i>	INT	Detected error Information.
<i>ErrorCode1</i>	INT	Type of detected error.
<i>Retries</i>	INT	Number of retries.
<i>ReadLength</i>	INT	Length of data received from port in bytes.
<i>OrderNumber</i>	INT	Order number of request.
<i>Status</i>	WORD	The list of statuses are: <ul style="list-style-type: none"> <li>• 0- Idle</li> <li>• 1 -Client has placed a request.</li> <li>• 2 - Port is processing the request.</li> <li>• 3 - Port has processed without a detected error.</li> <li>• 4 - Port has processed with detected error.</li> </ul>
<i>ServiceID</i>	BYTE	EIP Service ID.

# StatisticCounter1 - Statistic Counter Profile

## What's in This Chapter

Description .....	64
DFB Representation .....	64
Inputs .....	64
Outputs .....	65

## Overview

This chapter describes the DFB of Statistic counter profile.

## Description

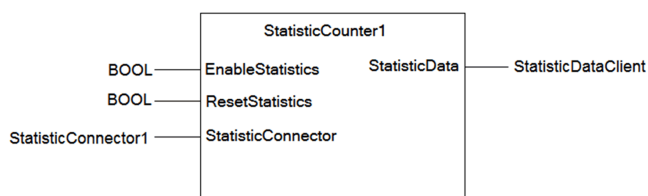
### General

`StatisticCounter1` DFB is used to log the statistical data of the requests handled by the client over Ethernet IP protocol.

## DFB Representation

### Representation

The following figure represents the functional module of `StatisticCounter1`.



## Inputs

### Input Parameter Description

Parameter	Type	Description
EnableStatistics	BOOL	Enable the statistic counter.
ResetStatistics	BOOL	Reset the statistics.
StatisticConnector	StatisticConnector1	Statistic Connector structure. To be connected from the client.



## StatisticConnector1 DDT Structure

Name	Type	Description
Start	BOOL	Start command.
EndOK	BOOL	Request ended successfully.
EndNOK	BOOL	Request ended with detected fail.
TotalTime	DINT	Total time taken for current request.

## Outputs

### Output Parameter Description

Parameter	Type	Description
StatisticData	StatisticDataClient	Client statistic data.

## StatisticDataClient DDT structure


Name	Type	Description
RequestsSent	UDINT	Number of requests sent.
RequestsOK	UDINT	Number of requests that ended successfully.
RequestsError	UDINT	Number of requests that ended with a detected fail.
MinCycleNumber	UINT	Minimum number of PLC cycles to process a request.
AvgCycleNumber	UINT	Average number of PLC cycles to process a request.
MaxCycleNumber	UINT	Maximum number of PLC cycles to process a request.
CurrentCycle- Number	UINT	Number of PLC cycles spent on the current request.
MinTotalTime	UINT	Minimum amount of time spent on the requests sent.
AvgTotalTime	UINT	Average amount of time spent on the requests sent.
MaxTotalTime	UINT	Maximum amount of time spent on the requests sent.
LastTotalTime	UINT	Amount of time spent on the last request sent.
CurrentTotal- Time	UINT	Time spent on current request.
Requestspers- Second	UINT	Number of requests sent per second.

What’s in This Part

PRMMgt - PRM Management.....67

Overview

This part provides the detailed description of the Profibus based DFBs.  
These function blocks do not reflect any specific installation.

 **WARNING**

**LOSS OF CONTROL**

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

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

<sup>1</sup> 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.

# PRMMgt - PRM Management

## What's in This Chapter

Description .....	67
DFB Representation .....	67
Inputs .....	68
Outputs .....	69
Inputs/Outputs .....	70
Public Variables .....	70

## Overview

This chapter describes the DFBs of the *PRM Management* profile.

## Description

### Overview

This block is mandatory and has to be used in the application to start the PRM on a M340/M580/Quantum controller platform.

While enabled (FDB, LD language) or continuously executed (ST language), *PRM Management* automatically starts the PRM using the information given by the *Master\_Info* structure.

The start is not possible in case of an error detected inside this structure. The block is configured to restart the PRM, 3 times automatically. You can still start the PRM but after rectifying the detected error logged in the *Master\_Info.ddt*. The *RETRY* input needs to be triggered. The block again tries to start the PRM, 3 times and trigger *RETRY* input.

In addition, activate the input *EnableFullStatus* using *PRM Management* to get the PRM status (complete diagnostic information).

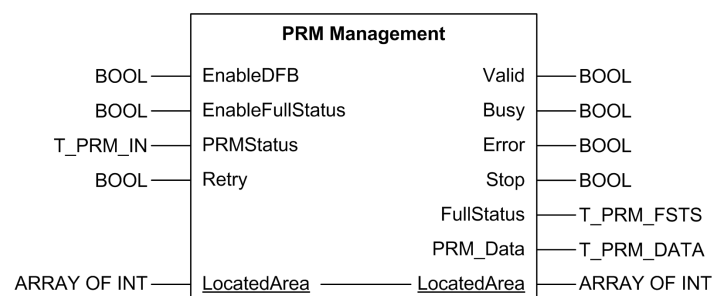
The conditions to read are:

- DFB enable input = TRUE
- *ERROR* output = FALSE

## DFB Representation

### Description

The following figure represents the functional module of *PRM Management*:



**NOTE:** The underlined parameters are specific for some components.

The table shows the parameters available for specific DFBs:

Parameters		Components	
		<i>PRMMgtM</i>	<i>PRMMgtQ</i>
Inputs/Outputs	LocatedArea	–	X
X: Parameter is available			
–: Parameter is not available			

## Inputs

### Input Parameter Description

Parameter	Type	Description
<i>EnableDFB</i>	BOOL	1 = Enables the DFB function.
<i>EnableFullStatus</i>	BOOL	1 = Request to read the PRM status and update the FSTS output accordingly.
<i>PRMStatus</i>	T_PRM_IN, page 68	Dynamic information, PRM implicit status To be connected to the variable named: <PRM Master alias name>_IN.
<i>Retry</i>	BOOL	1 = Request to retry to start the PRM.

### T\_PRM\_IN Type

Parameter	Type	Description
<i>PRM_MASTER_STATUS</i>	BYTE	<ul style="list-style-type: none"> <li>0 = No IO exchange</li> <li>1 = No configuration</li> <li>2 = No link</li> <li>3 = Stop</li> <li>4 = Run</li> </ul>
<i>PROFIBUS_STATUS</i>	BYTE	<ul style="list-style-type: none"> <li>0 = Unknown</li> <li>9 = Inoperable devices</li> <li>10 = Devices in diagnostics</li> <li>11 = Inoperable devices and devices in diagnostics</li> <li>12 = Detected error in Profibus</li> <li>16 = OK</li> </ul>
<i>DPM1_STATUS</i>	BYTE	<ul style="list-style-type: none"> <li>0 = Unknown</li> <li>1 = Offline</li> <li>2 = Stopped</li> <li>3 = Operate</li> <li>4 = Clear</li> </ul>
<i>DPM2_STATUS</i>	BYTE	Number of pending Profibus acyclic requests.
<i>DIAG_LIST</i>	ARRAY OF WORD	List of configured devices with pending diagnosis.
<i>MAILBOX_STATUS</i>	ARRAY OF BYTE	Status of the mailboxes used by DFBs.

## Outputs

### Output Parameter Description

Parameter	Type	Description
<i>Valid</i>	BOOL	1 = Valid PRM diagnostic (FSTS) received.
<i>Busy</i>	BOOL	1 = DFB working – other outputs are undefined.
<i>Error</i>	BOOL	1 = No link established between the PRM and the PLC.  <i>PRM_Master_Status</i> in the INIT, NO CONF, or NO LINK state.
<i>Stop</i>	BOOL	1 = The PRM is ready, the link is established with the PLC, but the PRM is waiting for a start from the PLC.  <i>PRM_Master_Status</i> in the STOP state.
<i>FullStatus</i>	T_PRM_FSTS, page 69	Record of the PRM complete status.
<i>PRM_Data</i>	T_PRM_DATA, page 70	DDT contains information on communication path to the PRM and the PRM status.

### T\_PRM\_FSTS Type

Parameter	Type	Description
<i>Name</i>	STRING [32]	Product name assigned in the DTM browser.
<i>IP</i>	ARRAY [0..3] of BYTE	Current IP address.
<i>Rotary</i>	ARRAY [0..1] of BYTE	Position of rotary switch <ul style="list-style-type: none"> <li><i>Rotary</i>[0] = Lower switch</li> <li><i>Rotary</i>[1] = Upper switch</li> </ul>
<i>CRC</i>	UDINT	<i>CRC_IOMAPPING</i> expected by the PRM.
<i>PRM_MASTER_STATUS</i>	BYTE	<ul style="list-style-type: none"> <li>0 = No IO exchange</li> <li>1 = No configuration</li> <li>2 = No link</li> <li>3 = Stop</li> <li>4 = Run</li> </ul>
<i>PROFIBUS_STATUS</i>	BYTE	<ul style="list-style-type: none"> <li>0 = Unknown</li> <li>9 = Inoperable devices</li> <li>10 = Devices in diagnostics</li> <li>11 = Inoperable devices and devices in diagnostics</li> <li>12 = Detected error in Profibus</li> <li>16 = OK</li> </ul>
<i>DPM1_STATUS</i>	BYTE	<ul style="list-style-type: none"> <li>0 = Unknown</li> <li>1 = Offline</li> <li>2 = Stopped</li> <li>3 = Operate</li> <li>4 = Clear</li> </ul>
<i>DPM2_STATUS</i>	BYTE	Number of pending Profibus acyclic requests.
<i>IOscanner_Requests</i>	BYTE	Number of Modbus IO scanner requests received in an IO scanner cycle.
<i>PROFIBUS_IO_Bandwidth</i>	BYTE	Profibus bandwidth used by PRM for data exchange (%).

## T\_PRM\_DATA Type

Parameter	Type	Description
<i>PRM_Address</i>	T_PRM_INFO_M	Address of PRM.
<i>Rack_number</i>	BYTE	Rack number of the Ethernet module linked to the PRM.
<i>Module_number</i>	BYTE	Position of Ethernet module in the rack.
<i>Channel_number</i>	BYTE	Channel number of the Ethernet port into the Ethernet module.
<i>IP</i>	ARRAY [1..4] of BYTE	Current IP address. <ul style="list-style-type: none"> <li><i>IP4</i> = Most significant byte of IP address of the PRM</li> <li><i>IP1</i> = Least significant byte of IP address of the PRM</li> </ul>

## Inputs/Outputs

### Input/Output Parameter Description

Parameter	Type	Description
<i>LocatedArea</i> *	ARRAY of INT	Array of located variables (%MW) required for Quantum.
* This parameter is available only with PRMMgtQ.		

## Public Variables

### Public Variable Description

Parameter	Type	Description
<i>PortAddress</i>	STRING	Entered as <i>Rack Number.Module Number.Channel Number</i> .  Position of the Ethernet module linked to the PRM (254 if CPU and slot number if NOE).
<i>IPAddress</i>	STRING	IP address of the PRM.

---

# Diagnostic Information Management

## What's in This Part

Diagnostic Information Management Codes ..... 72

# Diagnostic Information Management Codes

## What's in This Chapter

Description .....	72
Read_Var and Write_Var Diagnostic Codes .....	73
MBP_MSTR Diagnostic Codes .....	74
Client Parameter Diagnostic Codes .....	76
Scanner Parameter Diagnostic Codes .....	77
Ethernet/IP Communication Diagnostic Codes .....	78

## Overview

This chapter describes the diagnostic information management codes.

## Description

### General

The client and scanner function blocks manage information related to detected errors in communication and reporting the status of the last transaction with a detected problem result.

The `FailCode[.]` output, which has 3 detected error levels to specify the detected error source, is provided in the following table for this function:

For `FailCode[2]`:

<code>FailCode[2]</code>	Meaning
16#0000	Client/scanner requests
16#0001	Read requests from Device library blocks
16#0002	Write requests from Device library blocks

For `FailCode[1]`:

<code>FailCode[1]</code>	Meaning
16#0001	A <code>Read_Var</code> request
16#0002	A <code>Write_Var</code> request
16#0004	A read <code>MBP_MSTR</code> request
16#0005	A write <code>MBP_MSTR</code> request
16#0006	Client parameter error detected
16#0007	Scanner parameter error detected

## Serial Modbus-Modbus TCP/IP gateway

If the client is connected to the network through a serial Modbus-Modbus TCP/IP gateway, the detected error is the same on level 0 and on level 1. The high byte is used for the level 1 diagnostic code of the gateway client (that is, of the gateway itself) and the low byte is used for the diagnostic code of the actual client:

Level 1: 16#ppcc	Where pp represents the gateway detected error and cc the client detected error
------------------	---



If the detected error comes from the Gateway, the low byte of the word is FF.

**Example:**

FailCode [1]	16#02FF
FailCode [0]	16#xxxx

In this case, the client is reporting that the code for level 0 (xxxx) was returned by the `Write_Var` function through the Gateway.

**Another example:**

FailCode [1]	16#0006
FailCode [0]	16#xxxx

In this case, the system is normal. The client returns the level 0 code due to a parameter detected error.

## Read\_Var and Write\_Var Diagnostic Codes

### Description

For FailCode [1]:

FailCode [1]	Meaning
16#0001	A Read_Var request
16#0002	A Write_Var request

For FailCode [0] (without Advantys bridge):

FailCode [0]	Meaning
16#0001	Exchange stopped due to timeout
16#0002	Exchange stopped following a user request (CANCEL) / Init request initiated in the port block.
16#0003	Incorrect address format
16#0004	Incorrect destination address
16#0005	Incorrect management parameter format
16#0006	Incorrect specific parameters
16#0007	Interruption while transmitting to the destination
16#0009	Insufficient reception buffer size
16#000A	Insufficient transmission buffer size
16#000B	Not enough processor system resources
16#000C	Incorrect exchange number
16#000D	No frames have been received
16#000E	Incorrect length
16#000F	Frame service not configured
16#0010	Missing network coupler
16#0011	No request present
16#0012	Application server already active
16#0013	Incorrect UNI-TE V2 transaction number
16#00FF	Message rejected

FailCode [0]	Meaning
16#0100	Request not processed
16#0200	Incorrect response
16#01FF	No processor resources
16#02FF	No line resources
16#04FF	Incorrect line
16#05FF	Incorrect length
16#06FF	Inoperable communication channel
16#07FF	Invalid addressing
16#08FF	Invalid application
16#0BFF	No system resources
16#0CFF	Inactive communication function
16#0DFF	Destination not present
16#0FFF	Non-authorized access between stations or non-configured channel
16#11FF	Non-managed address format
16#12FF	No destination resources
16#14FF	Non-operational connection (for example, Ethernet TCP/IP)
16#15FF	No local channel resources
16#16FF	Non-authorized access (for example, Ethernet TCP/IP)
16#17FF	Inconsistent network configuration (for example, Ethernet TCP/IP)
16#18FF	Connection temporarily unavailable
16#21FF	Stopped application server
16#300B	Timeout in Gateways serial bus
16#30FF	Incorrect transmission

## MBP\_MSTR Diagnostic Codes

### Description

For FailCode [1]:

FailCode [1]	Meaning
16#0004	A read MBP_MSTR request
16#0005	A write MBP_MSTR request

For FailCode [0]:

FailCode [0]	Meaning
16#0002	Init request initiated in port block.
16#1001	Interruption on behalf of user.
16#2001	An operation type that is not supported is defined in the control block.
16#2002	One or several control block parameters are modified while the MSTR element is active (only valid for operations that require several cycles to end). You can modify the control block parameters when the MSTR element is inactive.
16#2003	Invalid value in the control block length field.
16#2004	Invalid value in the control block offset field.

FailCode [0]	Meaning
16#2005	Invalid value in the control block length and offset fields.
16#2006	Data field not allowed on slave.
16#2007	Network field not allowed on slave.
16#2008	Network access path not allowed on slave.
16#2009	Access path equivalent to its own address.
16#2010	Attempt to receive a larger number of global data words than is available.
16#2014	The control block is not assigned, or parts of the control block are located outside of the configured %MW (4x) area.
16#3001	Slave does not support requested operation.
16#3002	Registers from a non-existent slave have been requested.
16#3003	A data value that is not supported was requested.
16#3005	The slave has received a long program command.
16#3006	The function cannot be executed: A long command is being executed.
16#3007	The slave has rejected a long program command.
16#3011	The request has timed out on the gateway.
16#4001	Inconsistent response through Modbus slave.
16#5001	Inconsistent response through network.
16#5056	Non-operational TCP connection.
16#6001	No response has been received.
16#6002	Access to a program denied.
16#6003	Disconnected node. Communication is impossible.
16#6004	Extraordinary response received.
16#6005	Busy router node data paths.
16#6006	The slave is in detected error state.
16#6007	Bad destination address.
16#6008	Non-supported node type in access path.
16#6010	The slave rejected the command.
16#6020	The slave has forgotten the activated transaction.
16#6040	An unexpected master output path has been received.
16#6080	An unexpected response has been received.
16#6101	Detected error in first device on the path. No response has been received.
16#6102	Detected error in first device on the path. Access to a program denied.
16#6103	Detected error in first device on the path. Disconnected node. Communication is impossible.
16#6104	Detected error in first device on the path. Extraordinary response received.
16#6105	Detected error in first device on the path. Busy router node data paths.
16#6106	Detected error in first device on the path. The slave was not responding.
16#6107	Detected error in first device on the path. Bad destination address.
16#6108	Detected error in first device on the path. Non-supported node type in access path.
16#6110	Detected error in first device on the path. The slave has rejected the command.
16#6120	Detected error in first device on the path. The slave has forgotten the activated transaction.
16#6140	Detected error in first device on the path. An unexpected master output path has been received.

FailCode [0]	Meaning
16#6180	Detected error in first device on the path. An unexpected response has been received.
16#6201	Detected error in second device on the path. No response has been received.
16#6202	Detected error in second device on the path. Access to a program denied.
16#6203	Detected error in second device on the path. Disconnected node. Communication is impossible.
16#6204	Detected error in second device on the path. Extraordinary response received.
16#6205	Detected error in second device on the path. Busy router node data paths.
16#6206	Detected error in second device on the path. The slave was not responding.
16#6207	Detected error in second device on the path. Bad destination address.
16#6208	Detected error in second device on the path. Non-supported node type in access path.
16#6210	Detected error in second device on the path. The slave has rejected the command.
16#6220	Detected error in second device on the path. The slave has forgotten the activated transaction.
16#6240	Detected error in second device on the path. An unexpected master output path has been received.
16#6280	Detected error in second device on the path. An unexpected response has been received.
16#6301	Detected error in third device on the path. No response has been received.
16#6302	Detected error in third device on the path. Access to a program denied.
16#6303	Detected error in third device on the path. Disconnected node. Communication is impossible.
16#6304	Detected error in third device on the path. Extraordinary response received.
16#6305	Detected error in third device on the path. Busy router node data paths.
16#6306	Detected error in third device on the path. The slave was not responding.
16#6307	Detected error in third device on the path. Bad destination address.
16#6308	Detected error in third device on the path. Non-supported node type in access path.
16#6310	Detected error in third device on the path. The slave has rejected the command.
16#6320	Detected error in third device on the path. The slave has forgotten the activated transaction.
16#6340	Detected error in third device on the path. An unexpected master output path has been received.
16#6380	Detected error in third device on the path. An unexpected response has been received.
16#6144	A wrong destination node was specified for the MSTR operation.

## Client Parameter Diagnostic Codes

### Description

For FailCode[1]:

FailCode[1]	Meaning
16#0006	Client parameter detected error

For FailCode[0]:

FailCode [0]	Meaning
16#2001	Invalid request length
16#2003	Invalid slave address
16#2010	No data buffer
16#2011	Data buffer size insufficient to execute the request
16#2012	Request length larger than specified <code>MaxReadSize</code>
16#2020	No data buffer
16#2021	Data buffer size insufficient to execute the request
16#2022	Invalid request length
16#2023	Request length larger than specified <code>MaxWriteSize</code>
16#2100	Insufficient <code>WorkMemory</code> size
16#2101	There are clients called after the communication port, or not called continuously

## Scanner Parameter Diagnostic Codes

### General

For `FailCode [1]`:

FailCode [1]	Meaning
16#0007	Scanner parameter detected error

For `FailCode [0]`:

FailCode [0]	Meaning
16#1001	Data buffer size insufficient to execute read request 1
16#1002	Data buffer size insufficient to execute read request 2
16#1003	Data buffer size insufficient to execute read request 3
16#1004	Data buffer size insufficient to execute read request 4
16#1005	Data buffer size insufficient to execute read request 5
16#1006	Data buffer size insufficient to execute read request 6
16#1007	Data buffer size insufficient to execute read request 7
16#1008	Data buffer size insufficient to execute read request 8
16#1009	Data buffer size insufficient to execute read request 9
16#1010	Data buffer size insufficient to execute read request 10
16#2001	Data buffer size insufficient to execute write request 1
16#2002	Data buffer size insufficient to execute write request 2
16#2003	Data buffer size insufficient to execute write request 3
16#2004	Data buffer size insufficient to execute write request 4
16#2005	Data buffer size insufficient to execute write request 5
16#2006	Data buffer size insufficient to execute write request 6
16#2007	Data buffer size insufficient to execute write request 7
16#2008	Data buffer size insufficient to execute write request 8
16#2009	Data buffer size insufficient to execute write request 9

FailCode[0]	Meaning
16#2010	Data buffer size insufficient to execute write request 10
16#3000	The input parameter arrays have different lengths

## EthernetIP Communication Diagnostic Codes

### Description

For FailCode[1]:

FailCode [1]	Meaning
16#0001	Data_Exch Communication error.
16#0002	Data_Exch Report error
16#0003	System Event
16#0004	Protocol Error
16#0005	Generic request error
16#0006	Parameter fault

Below given tables give the information of values of FailCode [0] for different values of FailCode[1].

For FailCode[1] = 1

FailCode [0]	Meaning
16#01	Exchange stop on timeout <b>NOTE:</b> OK value returned when a Modicon M340 CPU sends a MODBUS BROADCAST request.
16#02	Exchange stop on user request.
16#03	Incorrect address format.
16#04	Incorrect destination address.
16#05	Incorrect management parameter format.
16#06	Incorrect specific parameters.
16#07	Problem in sending to the destination.
16#08	Reserved.
16#09	Insufficient receive buffer size.
16#0A	Insufficient send buffer size.
16#0B	No system resources: The number of simultaneous communication EFs exceeds the maximum that can be managed by the processor.
16#0C	Incorrect Exchange number.
16#0D	No telegram received.
16#0E	Incorrect length.
16#0F	Telegram service is not configured.
16#10	Network module is missing.
16#11	Request missing.
16#12	Application server already active.

FailCode [0]	Meaning
16#13	UNI-TE V2 transaction number incorrect.
16#FF	Message refused. <b>NOTE:</b> OK value returned when a TSX SCP*** or TSX SCY*** sends a MODBUS BROADCAST request.

For FailCode[1] = 2

FailCode [0]	Meaning
16#01	No resources toward the processor.
16#02	No line resources.
16#03	No device or device without resources.
16#04	Line error.
16#05	Length error.
16#06	Faulty communication channel.
16#07	Detected addressing error.
16#08	Detected application error.
16#0B	No system resources: The number of simultaneous communication EFs exceeds the maximum that can be managed by the processor.
16#0C	Communication function not active.
16#0D	Destination missing
16#0F	Intra-station routing problem or channel not configured.
16#11	Address format not managed.
16#12	No destination resources.
16#14	Non-operational connection. For example, Ethernet TCP/IP.
16#15	No resource on the local channel.
16#16	Access not authorized. For example, Ethernet TCP/IP
16#17	Inconsistent network configuration. For example, Ethernet TCP/IP.
16#18	Connection temporarily unavailable.
16#21	Application server stopped.
16#30	Detected transmission error.

For FailCode[1] = 3

FailCode [0]	Meaning
16#800D	Timeout on the explicit message request.
16#8012	Inoperable device.
16#8015	Either: <ul style="list-style-type: none"> <li>Neither resources to handle the message, nor</li> <li>Internal detected error: No buffer available, no link available, not possible to send to the TCP task.</li> </ul>
16#8018	Either: <ul style="list-style-type: none"> <li>Another explicit message for this device is in progress, or</li> <li>TCP connection or encapsulation session in progress.</li> </ul>
16#8030	Timeout on the Forward_Open request.
<b>NOTE:</b> The following 16#81xx detected errors are Forward_Open response detected errors that originate at the remote target and are received via the CIP connection.	
16#8100	Faulty communication channel.
16#8103	Transport class and trigger combination not supported.

FailCode [0]	Meaning
16#8106	Ownership conflict.
16#8107	Target connection not found.
16#8108	Invalid network connection parameter.
16#8109	Invalid connection size.
16#8110	Target for connection not configured.
16#8111	RPI not supported.
16#8113	Out of connections.
16#8114	Vendor ID or product code mismatch.
16#8115	Product type mismatch.
16#8116	Revision mismatch.
16#8117	Invalid produced or consumed application path.
16#8149	Secondary resources available.
16#814A	Rack connection already established.
16#814B	Module connection already established.
16#814C	Miscellaneous
16#814D	Redundant connection mismatch.
16#814E	No more user-configurable link consumer resources: The configured number of resources for a producing application has reached the limit.
16#814F	No more user-configurable link consumer resources: There are no consumers configured for a producing application to use.
16#8160	Vendor specific.
16#8170	No target application data available.
16#8171	No originator application data available.
16#8173	Not configured for off-subnet multicast.
16#81A0	Detected error in data assignment.
16#81B0	Optional object state detected error.
16#81C0	Optional device state detected error.
<b>NOTE:</b> All 16#82xx detected errors are register session response detected errors.	
16#8200	Target device does not have sufficient resources.
16#8208	Target device does not recognize message encapsulation header.
16#820F	Reserved or unknown detected error from target.

For FailCode[1] = 4

FailCode [0]	Meaning
16#00	Success.
16#01	Connection unsuccessful.
16#02	Resource unavailable.
16#03	Invalid parameter value.
16#04	Path segment error.
16#05	Path destination unknown.
16#06	Partial transfer.
16#07	Connection lost.
16#08	Service not supported.
16#09	Invalid attribute value.



FailCode [0]	Meaning
16#0A	Attribute list error.
16#0B	Already in requested mode/state.
16#0C	Object state conflict.
16#0D	Object already exists.
16#0E	Attribute not settable.
16#0F	Privilege violation.
16#10	Device state conflict.
16#11	Reply data too large.
16#12	Fragmentation of primitive value.
16#13	Not enough data.
16#14	Attribute not supported.
16#15	Too much data.
16#16	Object does not exist.
16#17	Service fragmentation sequence not in progress.
16#18	No stored attribute data.
16#19	Store operation unsuccessful.
16#1A	Routing unsuccessful, request packet too large.
16#1B	Routing unsuccessful, response packet too large.
16#1C	Missing attribute list entry data.
16#1D	Invalid value list.
16#1E	Embedded detected service error.
16#1F	Vendor specific error.
16#20	Invalid parameter.
16#21	Write-once value or medium written.
16#22	Invalid reply received.

For FailCode[1] = 5

FailCode [0]	Meaning
16#3001	Client timeout error.
16#3002	Port was initialized while processing the request.

For FailCode[1] = 6

FailCode [0]	Meaning
16#2050	Client timeout error.
16#2001	Invalid request length.
16#2010	Insufficient request data buffer.
16#2011	Insufficient response data buffer.

---

# Communication Technologies

## What's in This Part

Supported Architectures .....	83
Ethernet Technology.....	88
Gateway Technology .....	95
Modbus Technology .....	100
EthernetIP Technology.....	103

## Overview

This part explains about communication technologies.

# Supported Architectures

## What's in This Chapter

Device/Communication Port Architectures..... 83

## Overview

This chapter provides the device/communication port matrix.

## Device/Communication Port Architectures

### Device/Communication Port Matrix

The following table describes the architectures that EcoStruxure Process Expert supports for managing communication between controllers and devices.

Device family	Device	Required device template	Controller family	Required communication port templates	
Circuit Breakers	Compact NSX	<i>CompactNSXMB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	Masterpact	<i>MasterpactMB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	Masterpact Draw-Out	<i>MasterpactCMB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
Digital Protective Relays	Sepam 20	<i>SEPAM20CSTMMB</i> <i>SEPAM20CBMB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	Sepam 40	<i>SEPAM40MB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	Sepam 80	<i>SEPAM80MB</i> <i>SEPAM80E</i>	M340/M580	<i>ModbusPortM</i>	

Device family	Device	Required device template	Controller family	Required communication port templates	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
Motor Controllers and Starters	Tsys T	TsysTEM	M340/M580	ModbusEthernetPortM	
			Quantum	ModbusEthernetPortQ	
		I/O TsysTE (I/O scanning)	M340/M580	EIO TsysTHW <sup>(1)</sup>	
			Quantum		
		TsysTE (fast I/O scanning)	M340/M580	ETsysTHW <sup>(1)</sup>	
			Quantum		
		TsysTMB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
		TsysTAS	M340/M580	–	
			Quantum	–	
Motor Controllers and Starters	Tsys U	TsysUStdStMB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
		TsysUAdvStMB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
		TsysUMfStMB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
		TsysUAdvCtlMB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
		TsysUMfCtlMB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ

Device family	Device	Required device template	Controller family	Required communication port templates	
Motor Controllers and Starters	TesyS U	<i>TesySUAAdvCtrlAS</i>	M340/M580	–	
			Quantum	–	
		<i>TesySUMfCtrlAS</i>	M340/M580	–	
			Quantum	–	
		<i>TesySUMFStarterAS</i>	M340/M580	–	
			Quantum	–	
		<i>TesySUAAdvStarterAS</i>	M340/M580	–	
			Quantum	–	
Power Monitoring Devices	PM 700	<i>PM700MB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	PM 800	<i>PM800MB</i> <i>PM800EM</i> <i>PM800E</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
Power Monitoring Devices	PM 1200	<i>PM1200MB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	PM 5350	<i>PM5350MB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	PM 9C	<i>PM9CMB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	MBSMART-UPS	<i>SmartUPSMB</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>
			Quantum	<i>Mod-BusGateway</i>	<i>ModbusEthernetPortQ</i>
	ACCUSINE	<i>AccusineE</i>	M340/M580	<i>ModbusPortM</i>	
				<i>Mod-BusGateway</i>	<i>ModbusEthernetPortM</i>

Device family	Device	Required device template	Controller family	Required communication port templates	
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
Progressive Starters	ATS 22	ATS22MB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
	ATS 48	ATS48MB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
Variable Speed Drives	ATV 12	ATV12MB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
	ATV 31	ATV31MB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
		ATV31AS	M340/M580	–	
			Quantum	–	
	ATV 312	ATV312MB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
	ATV 61	ATV61EM	M340/M580	ModbusEthernetPortM	
			Quantum	ModbusEthernetPortQ	
		ATV61E (I/O scanning)	M340/M580	EATV61HW <sup>(1)</sup>	
			Quantum		
		ATV61MB	M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
		ATV61AS	M340/M580	–	
			Quantum	–	
Variable Speed Drives	ATV 71	ATV71EM	M340/M580	ModbusEthernetPortM	
			Quantum	ModbusEthernetPortQ	
		ATV71E (I/O scanning)	M340/M580	EATV71HW <sup>(1)</sup>	

Device family	Device	Required device template	Controller family	Required communication port templates	
		ATV71MB	Quantum		
			M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
		ATV71AS	Quantum	Mod-BusGateway	ModbusEthernetPortQ
			M340/M580	–	
			Quantum	–	
	ATV 6xx/ ATV9xx	ATV6xxE	M340/M580	ModbusEthernetPortM	
			Quantum	ModbusEthernetPortQ	
			M340/M580	EATV71HW <sup>(1)</sup>	
			Quantum		
			M340/M580	ModbusPortM	
				Mod-BusGateway	ModbusEthernetPortM
			Quantum	Mod-BusGateway	ModbusEthernetPortQ
	PBAT-V7161	ATV61PB	M340/M580	PRMMgtM	
			Quantum	PRMMgtQ	
		ATV71PB	M340/M580	PRMMgtM	
			Quantum	PRMMgtQ	

**1** No Communication port is required for I/O scanning. Instead, in addition to the application device template, instantiate the topological template that corresponds to the device to create the communication channels between the controller and the device.

# Ethernet Technology

## What's in This Chapter

Ethernet Communication Architecture .....	88
Addressing Example for the M340/M580 Platform .....	89
Addressing Example for the Quantum Platform .....	90
Configuring the EMClient1 and EMPort1M .....	90

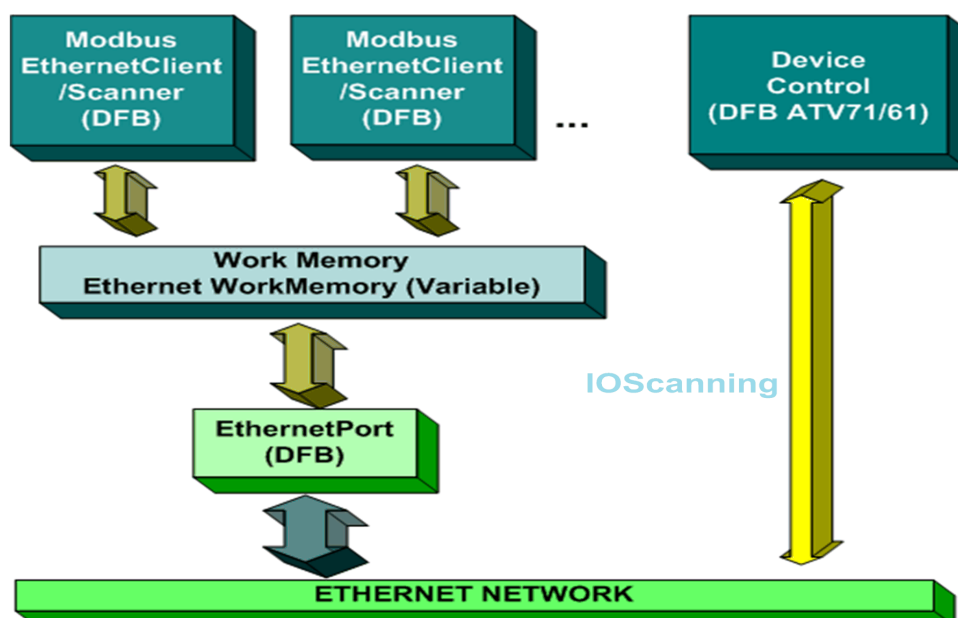
## Overview

This chapter describes the Ethernet technology.

## Ethernet Communication Architecture

### Architecture Diagram

The following diagram represents Ethernet communication architecture.



## Description

This first part allows to correctly address the Ethernet client DFB components that have to carry out non-cyclic messaging through the various Ethernet port DFB components.

Depending on the PLC platform used, IP addressing for Ethernet clients varies and needs to be handled separately. Independently of the PLC platform used, acyclic messaging requires an Ethernet port. IP addressing is implemented with the `DeviceAddress` input variable.

The Ethernet port DFB component carries out the Modbus TCP/IP request by means of `Read_Var`, `Write_Var`, or `MBP_MSTR`-type instructions in Quantum. Hence, the clients message is sent through `WorkMemory`.

Depending on the platform, the following definitions apply:



Platform		Device Addressing	Gateway Addressing	Port Addressing
M340/M580	CPU	'{IP}ID'	'{IP}'	'RackNumber.SlotNumber.ChannelNumber'
	NOE and NOC			
Quantum	CPU	'{IP}ID'	'{IP}'	'254'
	NOE and NOC			'Slot or Rack Number.SlotNumber.ChannelNumber' (In case of <i>ModbusPortQX80</i> )

If the destination device is another CPU with an Ethernet port, the above table applies.

## Addressing Example for the M340/M580 Platform

### ⚠ WARNING

#### UNINTENDED EQUIPMENT OPERATION

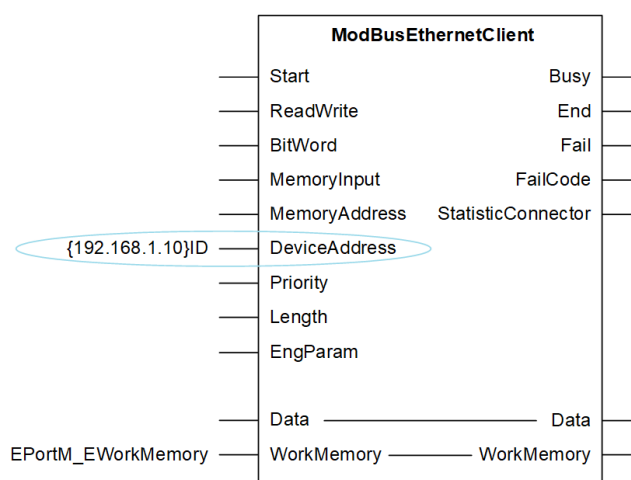
Adapt the below examples to configure device or communication network parameters before you implement them.

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

## Description

The current platform allows addressing with the name of the channel through which the Modbus TCP/IP requests are made.

Addressing for the Ethernet client with M340/M580 needs to be as in the following example (addressing for an `ModBusScanner`, `DeviceAddress` needs to be implemented the same way):



The ID ({IP}ID) is necessary and is based on the slave addressing and ranges from 0 to 255. For example, if it is an `ATV71`; 0 = `ATV71` variant, 251 = Ethernet card, 252 = Controller inside, 255 = IO Scanning.

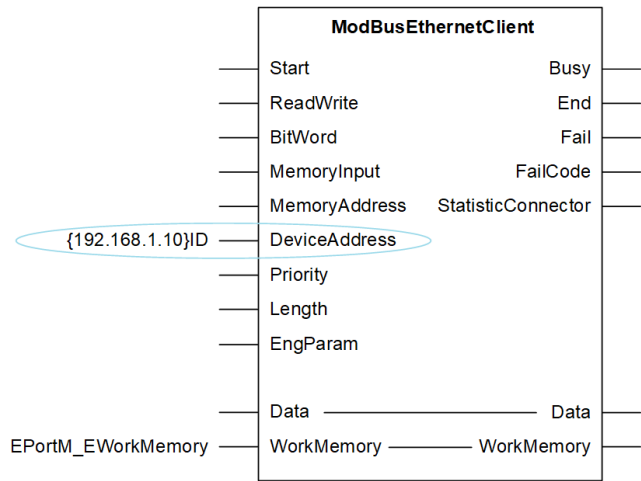
Regarding the Ethernet port DFB component to/from which the client transmits/receives external requests with `WorkMemory`, you need to define the public variable based on address table, page 88.

# Addressing Example for the Quantum Platform

## Introduction

Quantum platforms do not have XWAY addressing because the instructions they use for Modbus TCP/IP communication do not require this type of addressing.

Depending on the physical location of the Ethernet port, communication is to be established.



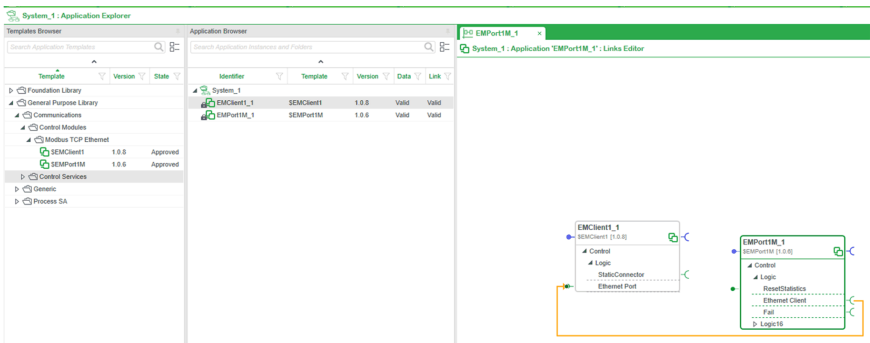
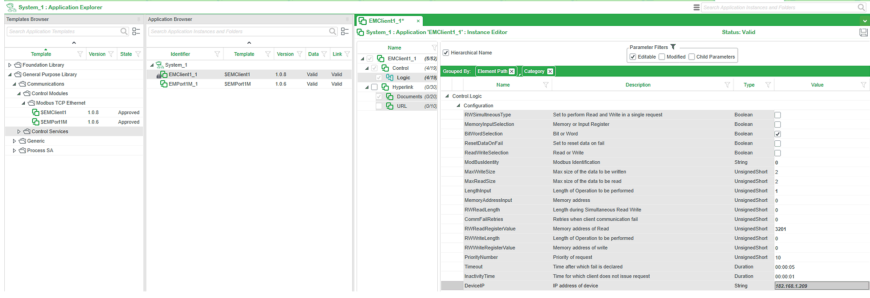
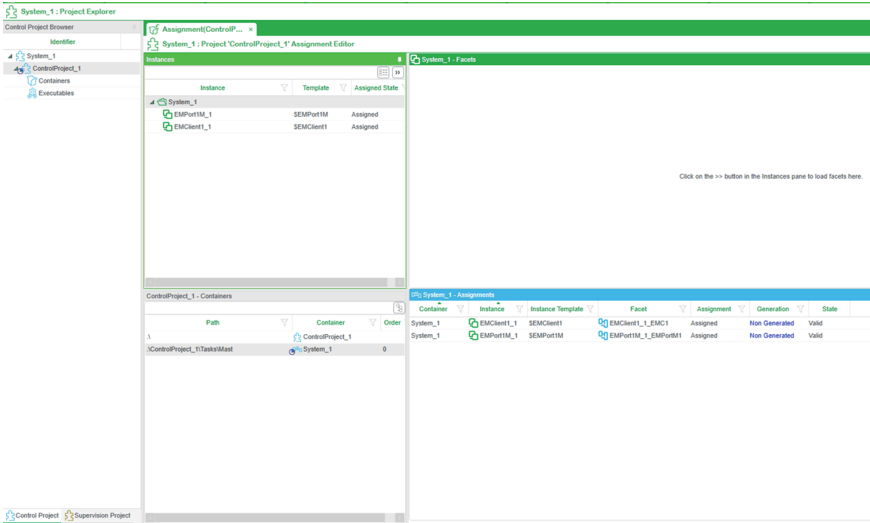
The ID ({IP}ID) is necessary and is based on the slave addressing and ranges from 0 to 255.

## Configuring the EMClient1 and EMPort1M

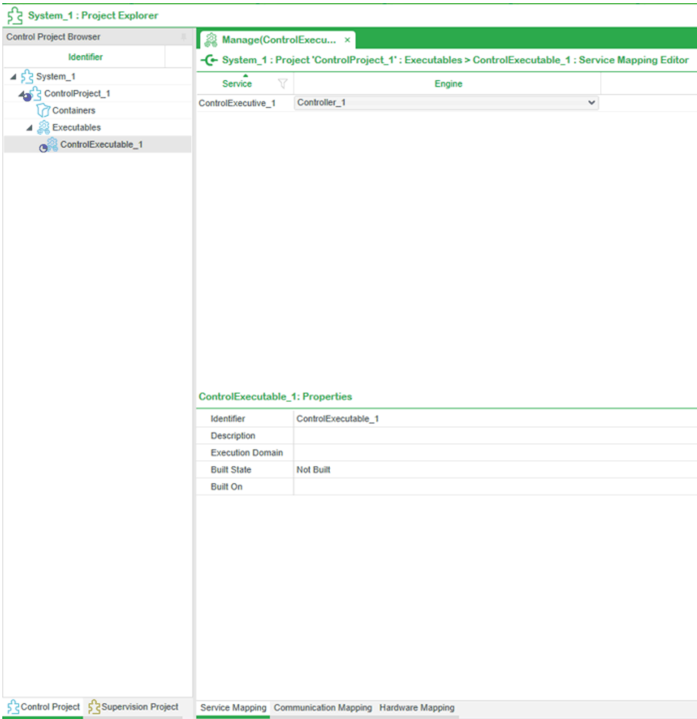
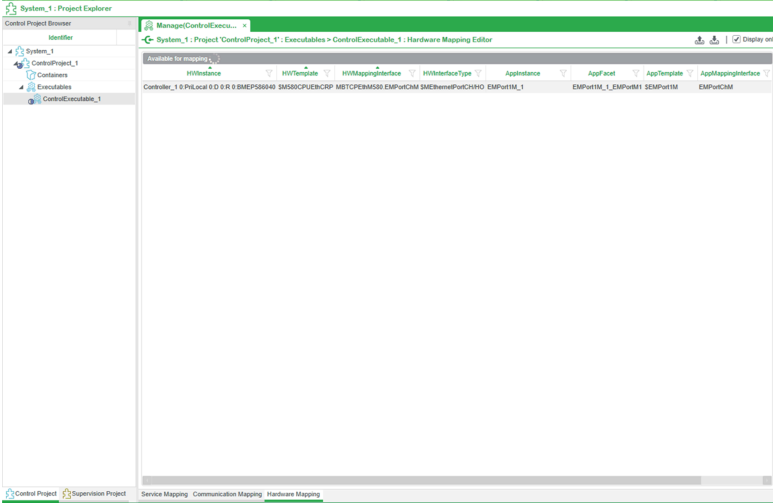
## Configuring the EMClient1 and EMPort1M

Perform the following steps to configure the EMClient1 and EMPort1M:

Step	Action
1	<div>Instantiate the EMClient1 and EMPort1M.</div> <div></div>
2	Perform <b>Edit</b> link.

Step	Action
	<div></div>
3	<div><p>Configure instance to read the register 3201 of length 1 from device with IP 182.168.1.209, as shown in image.</p><div></div></div>
4	<div><p>Assign the instances and generate.</p><div></div></div>
5	<div><p>Configure the Controller in Topology Explorer.</p></div>

Step	Action
	<div><div>System_1 : Topology Explorer</div><div><div>System_1</div><div><div>Create Folder</div><div>Open Application</div><div>Open Project</div><div>Create Ethernet Network</div><div>Create Station Node</div><div>Create Controller</div><div>Create STB Island</div><div>Create PRM Profibus DP</div><div>Create Device-IO</div><div>Update Template</div><div>Export</div><div>Import</div></div><div><div>Quantum</div><div>M340</div><div>M580</div></div></div></div>
6	<div>Link the created Topology to Ethernet network.</div> <div><div>System_1 : Topology Explorer</div><div><div>System_1</div><div><div>Instance</div><div>Template</div><div>Version</div><div>State</div><div>Description</div></div><div><div>EthernetNetwork_1</div><div>SEthernetNetwork</div><div>1.0.1</div><div>Valid</div><div></div></div><div><div>Controller_1</div></div></div><div><div>Controller_1: Physical Connections</div><div><div>Only Ethernet networks on which the IP address of a communication module is unique are listed.</div><div>Only communication modules that have a valid IP address are listed.</div><div><div>Communication Module</div><div>Allowed Network(s)</div></div><div><div>Controller_1 0 PnLocal 0 D 0 R 0 BMEPS66040</div><div>EthernetNetwork_1</div></div><div><div>Controller_1 2 EbrRIO</div><div>EthernetNetwork_1</div></div><div><div>OK</div><div>Cancel</div></div></div></div></div>

Step	Action
7	<div>Perform the mapping in executable in Project Explorer.</div> <div></div> <div></div>
8	<div>Deploy project to controller.</div>

Step	Action
	<div><div>System_1 : Topology Explorer</div><div><div>System_1</div><div><div>Instance</div><div>Template</div><div>Version</div><div>State</div><div>Description</div></div><div><div>EthernetNetwork_1</div><div>SEthernetNetwork</div><div>1.0.1</div><div>Valid</div></div></div><div><div>Controller_1</div><div><div>Expand</div><div>Expand All</div><div>Configure</div><div>Physical Connections</div><div>Update Template</div><div>Deploy Built Project</div><div>Deploy Changes / Undo Online Changes</div><div>Re-Deploy Last Project</div><div>Deploy Data</div><div>Start</div><div>Stop</div><div>Refine Online</div><div>Sync (Primary-&gt;Standby)</div><div>Update Control Project</div><div>Back Up Data</div><div>Manage Password</div><div>Clear Password</div><div>Forgot Password</div><div>Delete</div><div>Rename</div><div>Export</div><div>Properties</div></div></div></div>

**NOTE:** To perform Online add client repeat step 1,2,3,4 and add changes after build up the executable. (there is no need to stop the controller).

# Gateway Technology

## What's in This Chapter

Gateway Communication Architecture .....	95
Addressing Example for the M340/M580 Platform.....	96
Addressing Example for the Quantum Platform.....	98

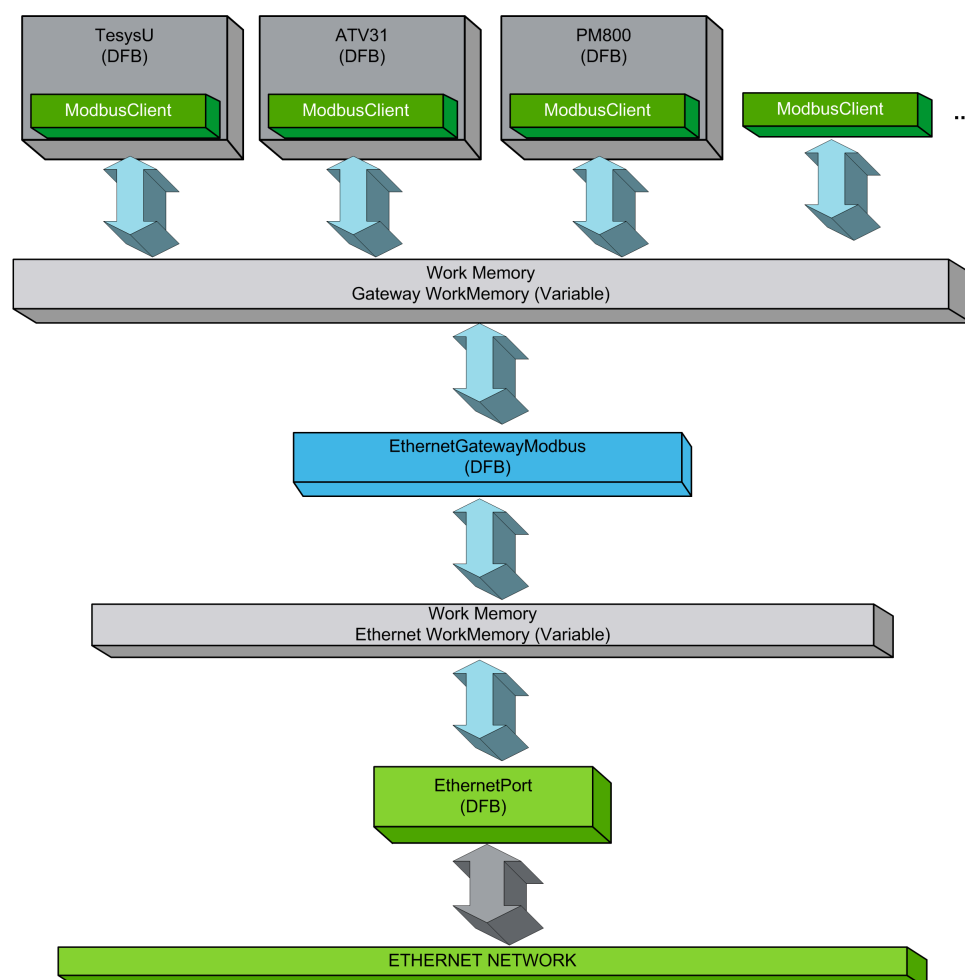
## Overview

This chapter describes the Gateway technology.

## Gateway Communication Architecture

### Architecture Diagram

The following diagram represents the architecture used in communication implemented with a Serial Modbus-Modbus TCP/IP Gateway:



## Description

This part enables you to correctly address the Modbus client/scanner components that have to carry out non-cyclic messaging through Modbus Ethernet gateways and the various Ethernet port instances.

Depending on the PLC platform to be used, IP addressing for Ethernet clients varies and needs to be handled separately. Independent of the PLC platform used, acyclic messaging requires an Ethernet gateway instance (ModBusGateway). IP addressing is implemented with the DeviceAddress input variable.

The Ethernet Modbus TCP/IP-Serial Modbus gateway component receives requests from the Modbus slaves (clients/scanners) through WorkMemoryRS485 and converts them into serial Modbus requests for the Ethernet port.

The programmed Ethernet port instance sends the Modbus TCP/IP request by means of Read\_Var, Write\_Var, or MBP\_MSTR-type instructions in Quantum. Because of this, the clients message is sent through EthernetWorkMemory.

Depending on the platform, the following definitions apply:

Platform	Gateway Addressing	Client/Scanner Addressing ModbusAddress (variable)
M340/M580/Quantum	'{IP}'	Modbus slave number

**NOTE:** Do not omit the punctuation marks.

## Addressing Example for the M340/M580 Platform

### **⚠ WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

Adapt the below examples to configure device or communication network parameters before you implement them.

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

## Introduction

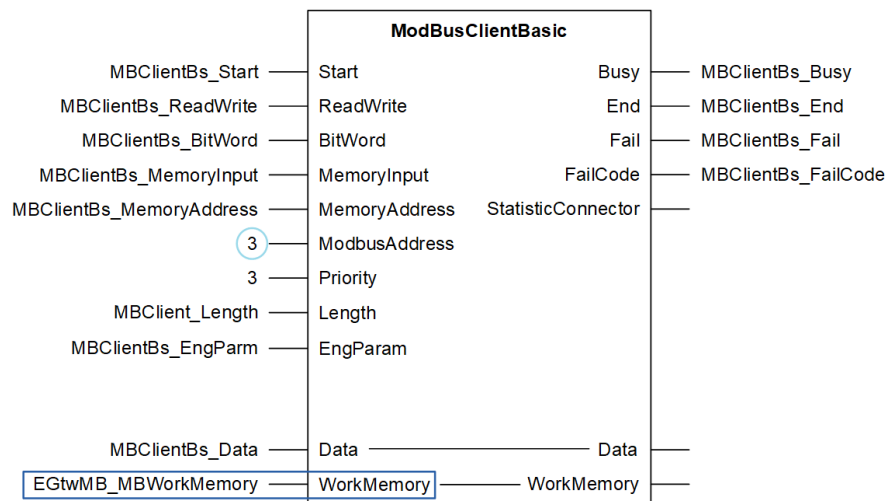
In this programming example, an M340/M580 series PLC sends requests to a Modbus slave (slave number 3) through a Serial Modbus-Modbus TCP/IP Gateway. This example has a client on Modbus (addressing in ModbusAddress and an ModBusScanner with various WorkMemorys need to be implemented the same way).

The name of the EGtwMB\_MBWorkMemory variable needs to be same in the ModBusClientBasic and ModBusGateway.

The name of the EMPortM\_EWorkMemory variable needs to be same in the ModBusGateway and EPortP.



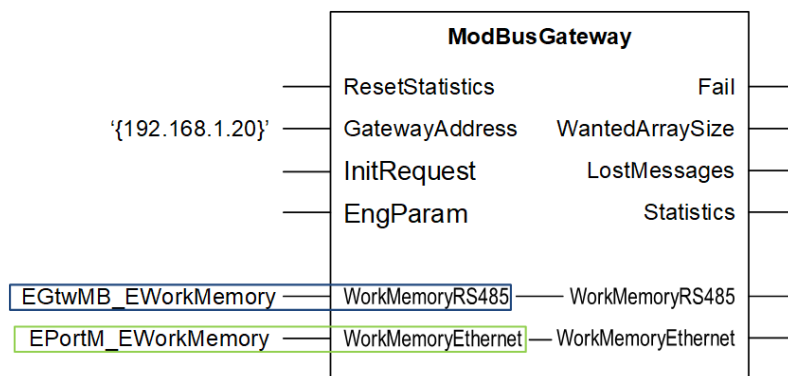
## Modbus Slave with Address 3



The result of the read operation will be found in `MBClientBs_Data` after `MBClientBs_End` has been activated. To carry out a write operation, the values to be written are found in `MBClientBs_Data`.

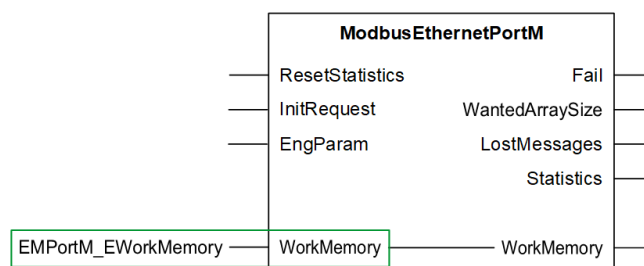
## Modbus TCP/IP – Serial Modbus Gateway

The IP address needs to be entered into the `GatewayAddress` input variable.



## Ethernet Port

The DFB has a public variable `PortAddress` in which the number of the slot needs to be entered. The PLC sends requests to the slave through this channel, page 88.



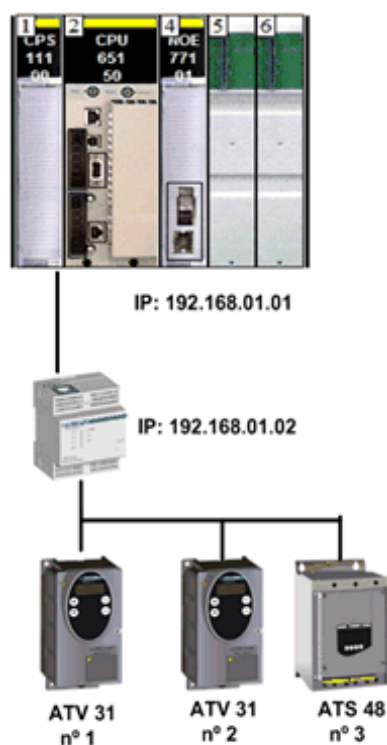
The name of the `EGtwMB_MBWorkMemory` variable needs to be same in the **ModBusClientBasic** and **ModBusGateway**.

The name of the `EMPortM_EWorkMemory` variable needs to be same in the **ModBusGateway** and **ModbusEthernetPortM**.

# Addressing Example for the Quantum Platform

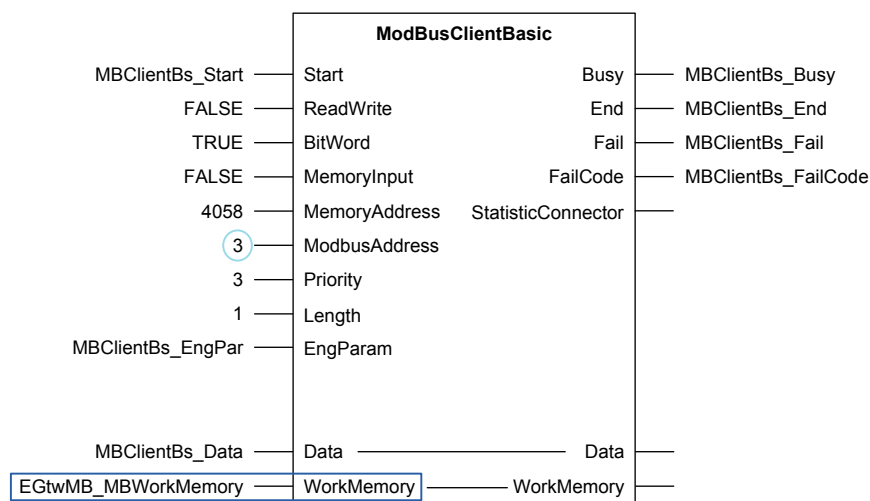
## General

The following figure represents the addressing example for the Quantum platform.



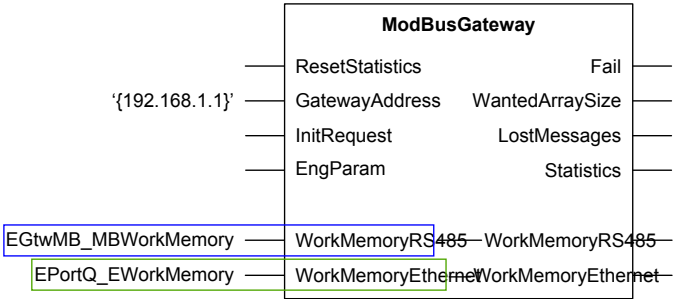
The objective is to establish communication between a Quantum PLC with an **ATS48** Modbus slave (Modbus slave number 3) through a Modbus Ethernet-Serial Modbus Gateway to write to the cascade function activation register (register 4058).

## Modbus Slave - ATS48 Speed Driver with Address 3

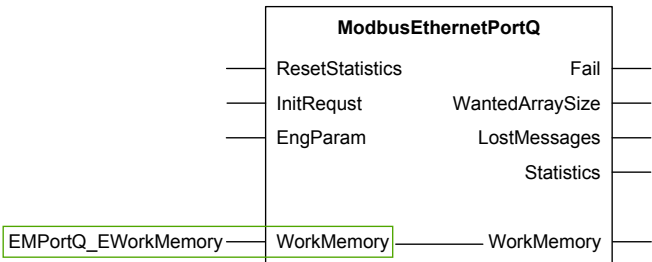


# Modbus TCP/IP – Serial Modbus Gateway

The Ethernet address needs to be addressed correctly. Communication is carried out through the PLC integrated port.



## Ethernet Port



The name of the EGtwMB\_MBWorkMemory variable needs to be same in the ModBusClientBasic and ModBusGateway.

The name of the EPortQ\_EWorkMemory variable needs to be same in the ModBusGateway and ModbusEthernetPortQ.

# Modbus Technology

## What's in This Chapter

Modbus Communication Architecture..... 100  
Addressing Example for the Modicon M340/M580 Platform ..... 101

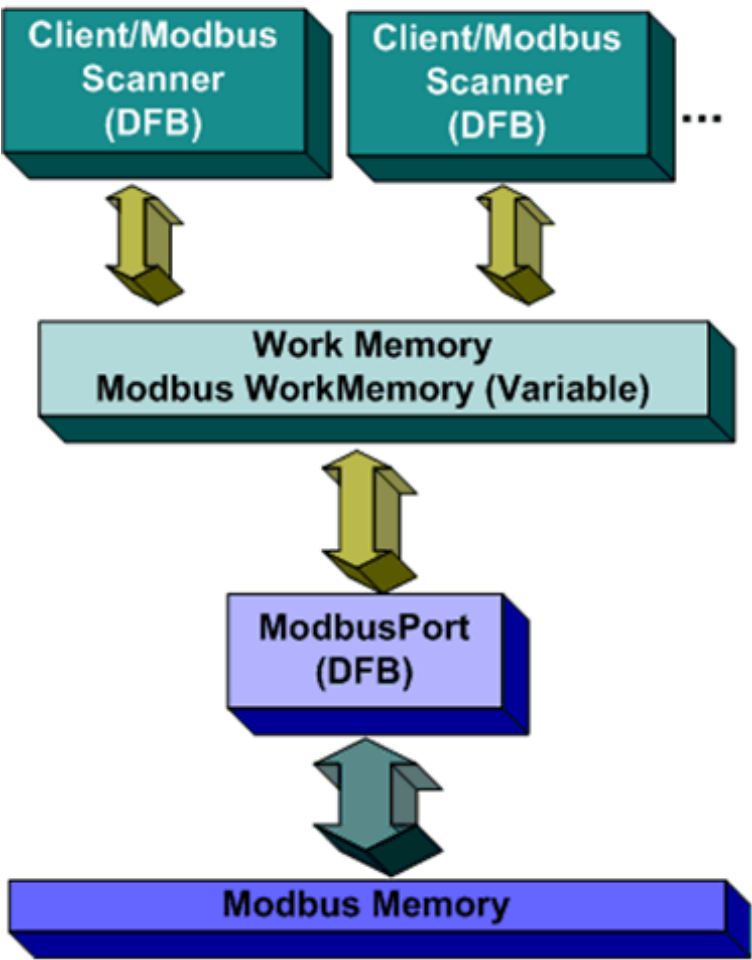
## Overview

This chapter describes the Modbus technology.

## Modbus Communication Architecture

### Architecture Diagram

The following figure represents the Modbus communication architecture.



# Addressing Example for the Modicon M340/M580 Platform

## ⚠ WARNING

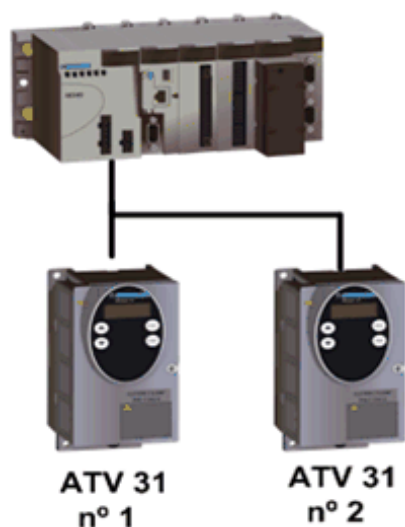
### UNINTENDED EQUIPMENT OPERATION

Adapt the below examples to configure device or communication network parameters before you implement them.

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

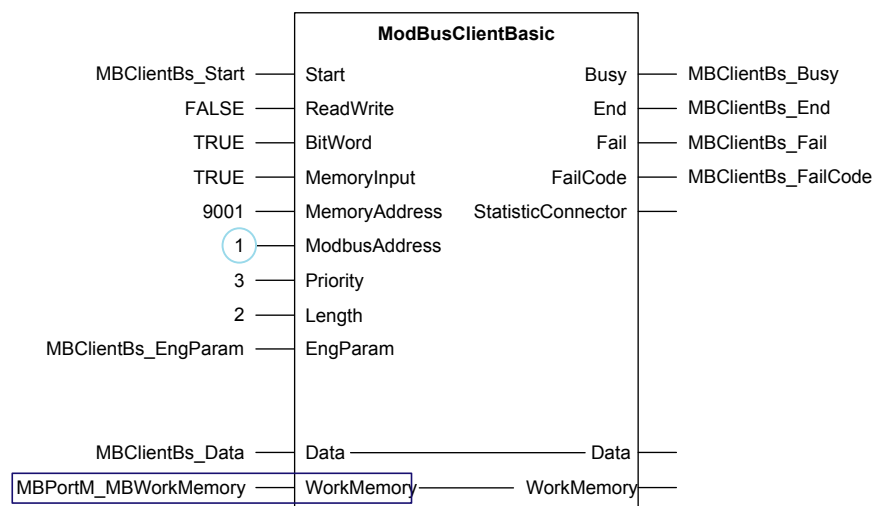
## Introduction

A Modicon M340/M580 PLC is used in this example to read the value of the ACC and DEC (registers 9001 and 9002) of the ATV 31 speed driver with slave number 1.



**NOTE:** This example can be applied to any CPU of the Modicon M340/M580 families that can communicate on Modbus.

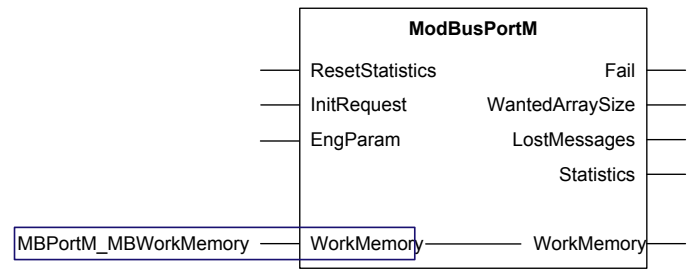
## ModBusClientBasic - Client with Address 1



### Related Topics:

The first register of the `MBClientBs_Data[0]` array holds the acceleration value. The second register holds the deceleration read from the device every time that `MBClientBs_Start` is `TRUE` and the `MBClientBs_End` reading operation ends in `TRUE`.

Modbus Port



The name of the `MBPortM_MBWorkMemory` variable needs to be same in the `ModBusClientBasic` and `ModbusPortM`.

# EthernetIP Technology

## What's in This Chapter

Ethernet IP Communication Architecture .....	103
EIPPort Client Configuration.....	103

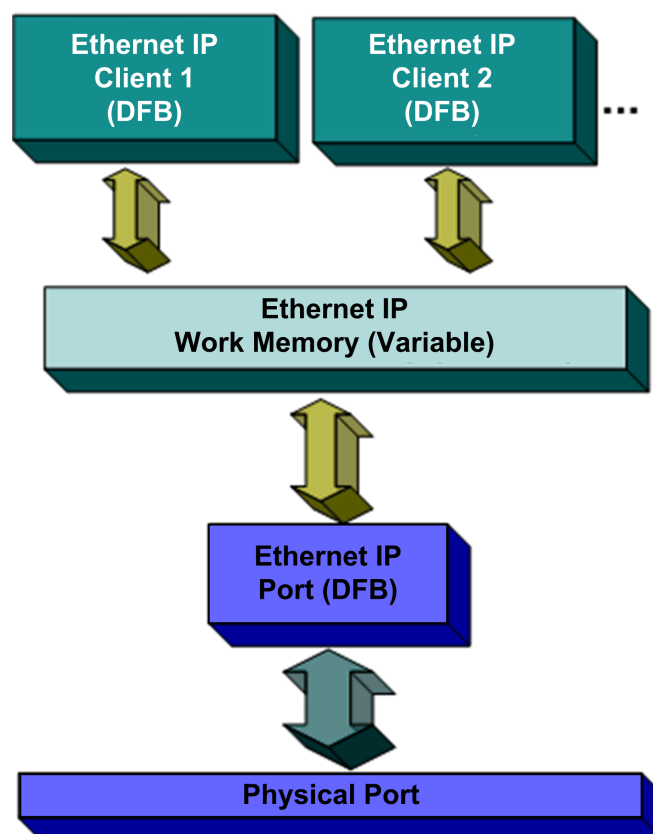
## Overview

This chapter describes the EthernetIP technology.

## Ethernet IP Communication Architecture

### Architecture Diagram

The following figure represents the Ethernet IP communication architecture.

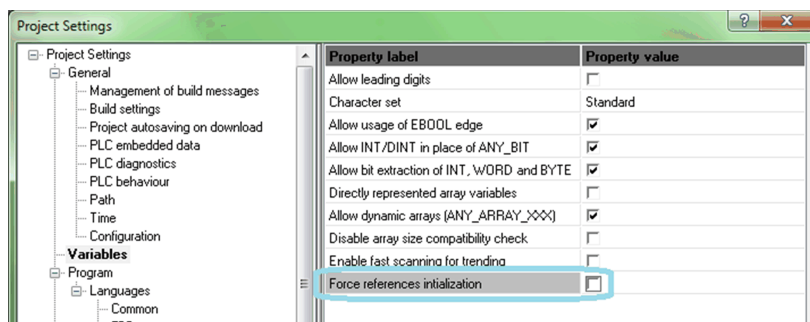


## EIPPort Client Configuration

### Overview

In this section, the configuration required for establishing communication on EIP unconnected explicit messaging is detailed. The configuration is detailed with an example using the EIP Port and client block which will execute a `Get_Single` attribute service.

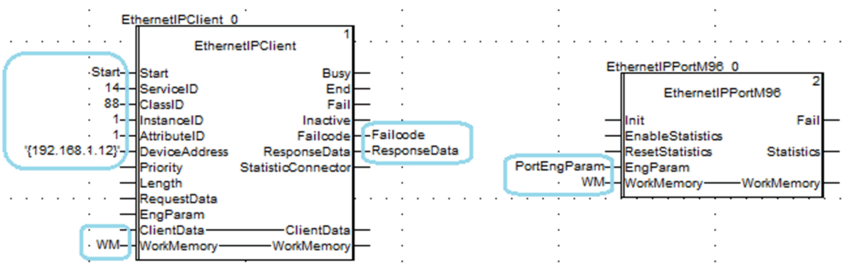
**NOTE:** The project setting **Force References Initialization** should be unchecked while using EIP port and client



## Instantiation













Step	Action
1	Instantiate <code>EthernetIPClient</code> DFB and <code>EthernetIPPortxx</code> DFB. <b>NOTE:</b> The Port DFB should be selected based on the controller family.
2	Declare <code>Failcode</code> variable and the <code>WorkMemory</code> variable and link them to the <code>Failcode</code> and <code>WorkMemory</code> pin respectively of the <code>EthernetIPClient</code> DFB (mandatory).
3	Link the same <code>WorkMemory</code> variable to the <code>EthernetIPPortxx</code> DFB.

## Configuration in Client

Step	Action
1	Define the device address (Only IP Address). For example, {192.168.1.25}
2	Configure the service ID to be executed. <b>NOTE:</b> If the service ID is in Hexadecimal format, prefix <b>16#</b> has to be used while configuring. For example <b>16#0E</b> .
3	Define a variable of required array size and connect to the <code>ResponseData</code> pin of <code>EthernetIPClient</code> DFB
4	Configure the <code>ClassID</code> , <code>InstanceID</code> and <code>AttributeID</code> on which the service has to be executed.  <b>NOTE:</b> If the service ID is in Hexadecimal format, prefix <b>16#</b> has to be used while configuring. For example <b>16#0E</b> .



## Configuration in Port

Step	Action																				
1	<p>Configure the PortAddress, SimultaneousSends and Timeout in the EngParam of Port, as shown in the below example.</p> <table><tr><td> PortEngParam</td><td>EIPPortEngineeringParam</td><td></td><td></td><td></td></tr><tr><td> PortAddress</td><td>string[14]</td><td></td><td>'0.0.3'</td><td>Communicatio...</td></tr><tr><td> SimultaneousSends</td><td>INT</td><td></td><td>79</td><td>Number of sim...</td></tr><tr><td> Timeout</td><td>TIME</td><td></td><td>t#3s</td><td>Time to wait a ...</td></tr></table>	 PortEngParam	EIPPortEngineeringParam				 PortAddress	string[14]		'0.0.3'	Communicatio...	 SimultaneousSends	INT		79	Number of sim...	 Timeout	TIME		t#3s	Time to wait a ...
 PortEngParam	EIPPortEngineeringParam																				
 PortAddress	string[14]		'0.0.3'	Communicatio...																	
 SimultaneousSends	INT		79	Number of sim...																	
 Timeout	TIME		t#3s	Time to wait a ...																	

### NOTE:

- On initialization all statistics data will be reset.
- Valid range of **SimultaneousSends** is 1 to **xx**. The value of **SimultaneousSends** remains 1 if the entered value is less than 1, similarly the value of **SimultaneousSends** remains **xx** if the entered value is greater than **xx**.
- In case of Set\_Attribute\_List (Service Code is 03<sub>hex</sub>) and Get\_Attribute\_List (Service Code is 04<sub>hex</sub>) services, if reading/ writing of any attribute in the list interrupts, the whole request is considered as a success and no detected error is notified by the protocol. The user application has to take care of this scenario and interpret the data.

---

# Appendices

## What's in This Part

Editing the `WorkMemory` Array Size ..... 107

# Editing the WorkMemory Array Size

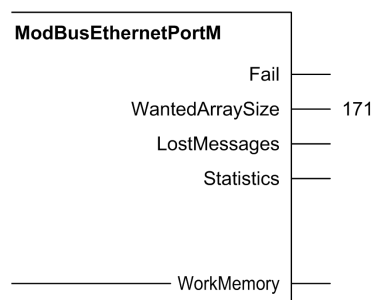
## What's in This Chapter

..... 107

The necessary size for the array of the variable associated to `WorkMemory` is automatically calculated by the DFBs of the port shown by the `WantedArraySize` output.

Check if the size of the array for the `WorkMemory` variable of the Port DFB, has a size  $\geq$  `WantedArraySize`, else edit the size of the array as described in the following example.


**Example with a `ModbusEthernetPortM` DFB:**



The following table describes the procedure to check the value of the `WantedArraySize` output:

Step	Action																																								
1	Execute the program with the calculated array.																																								
2	<p>In the Control Participant <b>Project Browser</b>, click <b>Variables &amp; FB Instances</b>→<b>Derived FB Instances</b>→<b>EMPortM</b> and check the value returned by the ModbusEthernetPortM DFB.</p> <table><tr><th>Name</th><th>no.</th><th>Type</th><th>Value</th><th>Comment</th></tr><tr><td>EMPortM</td><td></td><td>ModBusEt...</td><td></td><td></td></tr><tr><td>    <input type="checkbox"/> &lt;inputs&gt;</td><td></td><td></td><td></td><td></td></tr><tr><td>    <input type="checkbox"/> &lt;outputs&gt;</td><td></td><td></td><td></td><td></td></tr><tr><td>        <input checked="" type="radio"/> Fail</td><td>1</td><td>BOOL</td><td></td><td>Shows if there are errors with the communications</td></tr><tr><td>        <input checked="" type="radio"/> WantedArraySize</td><td>2</td><td>INT</td><td>171</td><td>Requested size of the WorkMemory array</td></tr><tr><td>        <input checked="" type="radio"/> LostMessages</td><td>3</td><td>INT</td><td></td><td>If this value is not 0 there is a DFB that is not called always. C...</td></tr><tr><td>        <input checked="" type="radio"/> Statistics</td><td>4</td><td>StatisticD...</td><td></td><td>Statistics of the data sent</td></tr></table>	Name	no.	Type	Value	Comment	EMPortM		ModBusEt...			<input type="checkbox"/> <inputs>					<input type="checkbox"/> <outputs>					<input checked="" type="radio"/> Fail	1	BOOL		Shows if there are errors with the communications	<input checked="" type="radio"/> WantedArraySize	2	INT	171	Requested size of the WorkMemory array	<input checked="" type="radio"/> LostMessages	3	INT		If this value is not 0 there is a DFB that is not called always. C...	<input checked="" type="radio"/> Statistics	4	StatisticD...		Statistics of the data sent
Name	no.	Type	Value	Comment																																					
EMPortM		ModBusEt...																																							
<input type="checkbox"/> <inputs>																																									
<input type="checkbox"/> <outputs>																																									
<input checked="" type="radio"/> Fail	1	BOOL		Shows if there are errors with the communications																																					
<input checked="" type="radio"/> WantedArraySize	2	INT	171	Requested size of the WorkMemory array																																					
<input checked="" type="radio"/> LostMessages	3	INT		If this value is not 0 there is a DFB that is not called always. C...																																					
<input checked="" type="radio"/> Statistics	4	StatisticD...		Statistics of the data sent																																					

The following table describes the procedure to edit the size of the array for the `WorkMemory` variable of the Port DFB:

Step	Action
1	In the Control Participant <b>Project Browser</b> , click <b>Variables &amp; FB Instances</b> → <b>Elementary Variables</b> → <b>EMPortM</b> .
2	<p>Edit the value of <code>WorkMemory</code> in the <b>Type</b> column of the variable.</p>  <p><b>NOTE:</b> The array should have a size of <code>[0... WantedArraySize-1]</code> as a minimum).</p>



---

# Index

## A

architectures	
supported communication architectures .....	83

## C

communication	
device-communication matrix .....	83
supported architectures .....	83
communication technologies	
Ethernet technology .....	88
Gateway technology .....	95
Modbus technology .....	100

## D

diagnostic information management	
diagnostic information management codes .....	72

## E

Ethernet communication architecture	
description .....	88
diagram .....	88
Ethernet technology	
addressing example for M340/M580 platform .....	89
addressing example for Quantum platform .....	90
Ethernet communication architecture .....	88

## G

Gateway communication architecture	
description .....	95
diagram .....	95
Gateway technology	
addressing example for the M340/M580	
platform .....	96
addressing example for the Quantum platform .....	98
Gateway communication architecture .....	95
general	
Communication library overview .....	16
communications resources overview .....	17
general concepts .....	18
general concepts	
communication process diagram .....	19
logical architecture - communication .....	18

## L

logical architecture - communication	
basics .....	18
gateway .....	18
memory management .....	18

## M

memory	
WorkMemory array size .....	107
Modbus communication	
ModBusScanner .....	46
Port Profile .....	26, 40
Modbus communication architecture	

diagram .....	100
Modbus TCP Ethernet communication	
ModBusGateway .....	51
Modbus technology	
addressing example for the Modicon M340/M580	
platform .....	101
Modbus communication architecture .....	100

## P

Profibus	
PRMMgt .....	67

## W

WorkMemory	
checking array size .....	107
editing array size .....	107

Schneider Electric  
35 rue Joseph Monier  
92500 Rueil Malmaison  
France

+ 33 (0) 1 41 29 70 00

[www.se.com](http://www.se.com)

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

© 2023 Schneider Electric. All rights reserved.

EIO0000001312.16