Modicon TM5

PCI Modules Configuration Programming Guide

09/2020

EIO0000003167.00

www.schneider-electric.com

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

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

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

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

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

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This document describes the configuration of the PCI expansion modules. For further information, refer to the separate documents provided in the EcoStruxure Machine Expert online help.

Validity Note

This document has been updated for the release of EcoStruxure^{TM} Machine Expert V1.2.5.

The technical characteristics of the devices described in the present document also appear online. To access the information online, go to the Schneider Electric home page *https://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.

Related Documents

Title of Documentation	Reference Number
Modicon M258 Logic Controller Programming Guide	*EIO0000004135 (Eng)* *EIO0000004136 (Fre)* *EIO0000004137 (Ger)* *EIO0000004138 (Spa)* *EIO0000004139 (Ita)* *EIO0000004140 (Chs)*
Modicon M258 Logic Controller System Functions and Variables M258 PLCSystem Library Guide	*EIO0000004141 (ENG)* *EIO0000004142 (FRE)* *EIO0000004143 (GER)* *EIO0000004144 (SPA)* *EIO0000004145 (ITA)* *EIO0000004146 (CHS)*
Modicon LMC058 Motion Controller Programming Guide	*EIO0000004165 (ENG)* *EIO0000004166 (FRE)* *EIO0000004167 (GER)* *EIO0000004168 (SPA)* *EIO0000004169 (ITA)* *EIO0000004170 (CHS)*

Title of Documentation	Reference Number
Modicon LMC058 Motion Controller System Functions and Variables LMC058 PLCSystem Library Guide	EIO0000004171 (ENG) EIO0000004172 (FRE) EIO0000004173 (GER) EIO0000004174 (SPA) EIO0000004175 (ITA) EIO0000004176 (CHS)
Modicon TM5 PCI Modules Hardware Guide	EIO0000003173 (ENG) EIO0000003174 (FRE) EIO0000003175 (GER) EIO0000003176 (SPA) EIO0000003177 (ITA) EIO0000003178 (CHS)

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

Product Related Information

⚠ **WARNING**

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.

Standard	Description
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Chapter 1
General Description

Introduction

This chapter provides the general description for configuring PCI expansion modules.

What Is in This Chapter?

This chapter contains the following topics:

TM5 PCI Expansion Modules General Description

Introduction

The controller accepts the following PCI Expansion Modules:

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

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

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

- Modicon M258 Logic Controller Hardware Guide *(see Modicon LMC058, Motion Controller, Hardware Guide)*,
- Modicon LMC058 Motion Controller Hardware Guide *(see Modicon M258, Logic Controller, Hardware Guide)*.

Add a PCI Expansion Module

Add a PCI Expansion Module

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

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

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

For further information, refer to PCI expansion module configuration.

Chapter 2
Serial Line PCI Expansion Module

Introduction

This chapter describes how to configure the Serial Line PCI Expansion Modules.

What Is in This Chapter?

This chapter contains the following sections:

Section 2.1
Serial Line PCI Expansion Module Configuration

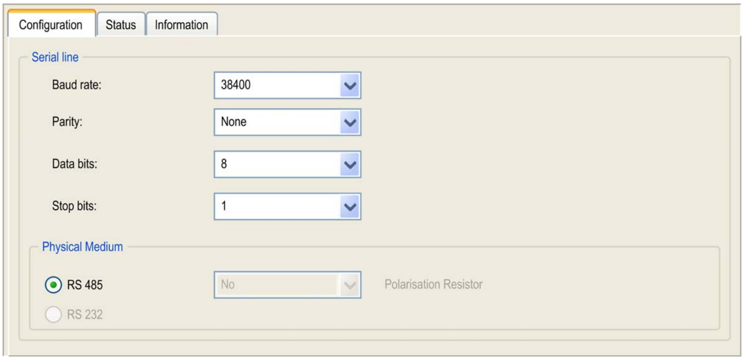
Serial Line PCI Expansion Module Configuration

Introduction

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

Serial Line Configuration

To configure the Serial Line, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click the Serial Line PCI Expansion module. **Result:** The configuration window is displayed.  Serial line Baud rate: 38400 Parity: None Data bits: 8 Stop bits: 1 Physical Medium ● RS 485 No Polarisation Resistor ○ RS 232

The following parameters must be identical for each Serial device connected to the port:

Element	Description
Baud rate	Transmission speed
Parity	Used for error detection
Data bits	Number of bits for transmitting data
Stop bits	Number of stop bits
Physical Medium	Specify the medium to use (automatically selected depending on the module): ● RS485 (for TM5PCRS4) ● RS232 (for TM5PCRS2)

The following table indicates the maximum baud rate according to the Manager:

Manager	Maximum Baud Rate
EcoStruxure Machine Expert Network Manager	115200
ASCIIManager	38400
Modbus IOScanner	
Modbus Manager	

Section 2.2
Devices on Serial Line PCI Expansion Modules

Introduction

This section describes the managers and devices of Serial Line PCI Expansion Modules.

What Is in This Section?

This section contains the following topics:

ASCII Manager

Introduction

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

Adding the Manager

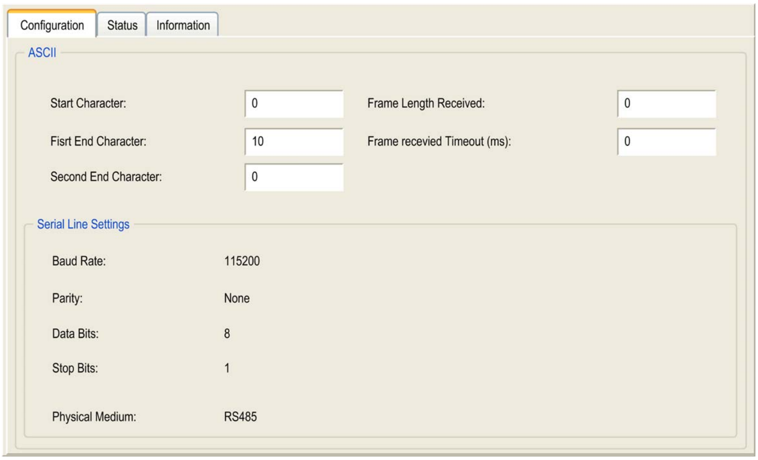
To add an ASCII Manager to your controller, select the **ASCII_Manager** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

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

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

ASCII Manager Configuration

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

Step	Action
1	In the **Devices tree**, double-click **ASCII_Manager**. **Result:** The **ASCII_Manager** configuration window is displayed.  Configuration Status Information ASCII Serial Line Settings Start Character: 0 Fisrt End Character: 10 Second End Character: 0 Frame Length Received: 0 Frame recevied Timeout (ms): 0 Baud Rate: 115200 Parity: None Data Bits: 8 Stop Bits: 1 Physical Medium: RS485

Set the parameters as described in the following table:

Parameter	Description
Start Character	If 0, no start character is used in the frame. In **Receiving Mode** the corresponding character in ASCII is used to detect the beginning of a frame. In **Sending Mode**, this character is added at the beginning of the frame.
First End Character	If 0, no first end character is used in the frame. In **Receiving Mode** the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Second End Character	If 0, no second end character is used in the frame. In **Receiving Mode** the corresponding character in ASCII is used to detect the end of a frame. In **Sending Mode**, this character is added at the end of the frame.
Frame Length Received	If 0, this parameter is not used. This parameter allows the system to conclude an end of frame at reception, when the controller receives the specified number of characters. **Note:** This parameter cannot be used simultaneously with **Frame Received Timeout (ms)**.
Frame Received Timeout (ms)	If 0, this parameter is not used. This parameter allows the system to conclude the end of frame at reception after a silence of the specified number of ms. **Note:** This parameter cannot be used simultaneously with **Frame Length Received**.
Serial Line Settings	Parameters specified in the Serial Line configuration window *(see page 16)*.

NOTE: In the case of using **First End Character** and **Second End Character** simultanously, these 2 conditions are considered as one frame termination condition. The frame termination condition becomes TRUE when the 2 characters are recognized.
In the case of using several frame termination conditions, the first condition to be TRUE will terminate the exchange.

Adding a Modem

To add a Modem to the ASCII Manager, refer to Adding a Modem to a Manager *(see page 32)*.

EcoStruxure Machine Expert Network Manager

Introduction

The EcoStruxure Machine Expert Network Manager must be used if you want to exchange variables with HMI Terminal Range with EcoStruxure Machine Expert software protocol, or when the Serial Line is used for EcoStruxure Machine Expert programming.

Adding the Manager

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

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

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

Configure the Manager

There is no configuration for EcoStruxure Machine Expert Network Manager.

Adding a Modem

To add a Modem to the EcoStruxure Machine Expert Network Manager, refer to Adding a Modem to a Manager *(see page 32)*.

Modbus IOScanner

Introduction

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

Add a Modbus IOScanner

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

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

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

Modbus IOScanner Configuration

To configure a Modbus IOScanner on a PCI Expansion Module, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **Modbus IOScanner**. **Result:** The configuration window is displayed.

Set the parameters as described in the following table:

Element	Description
Transmission Mode	The transmission mode to use is RTU. RTU uses binary coding and CRC error-checking (8 data bits). This parameter must be identical for each Modbus device on the link.
Response Timeout (ms)	Timeout used in the exchanges
Time between frames (ms)	Time to avoid bus-collision This parameter must be identical for each Modbus device on the link.

Adding a Device on the Modbus IOScanner

Introduction

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

Add a Device on the Modbus IOScanner

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

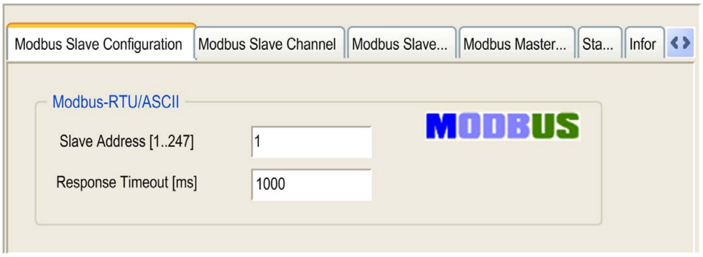
For more information on adding a device to your project, refer to:

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

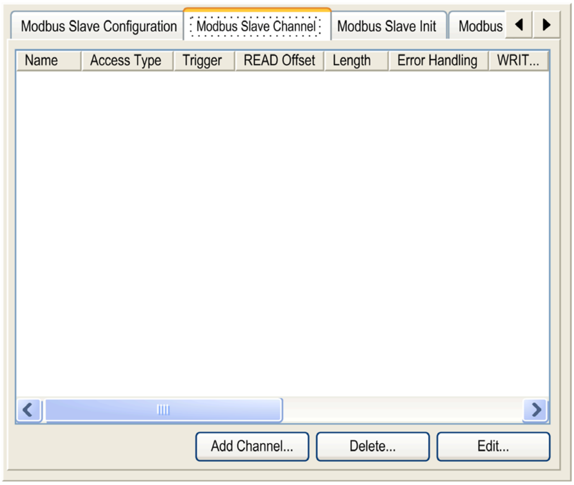
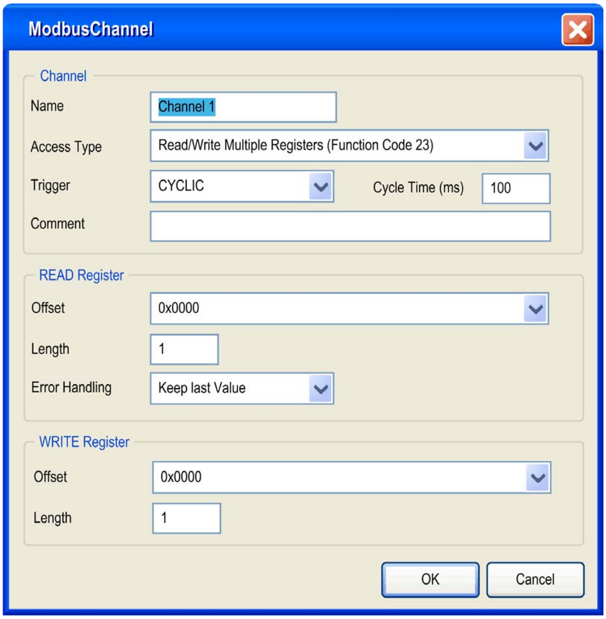
NOTE: The variable for the exchange is automatically created in the %IWx and %QWx of the **Modbus Serial Master I/O Mapping** tab.

Configure a Device Added on the Modbus IOScanner

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

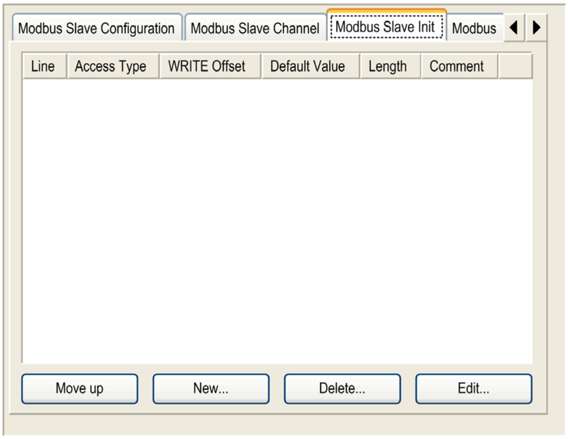
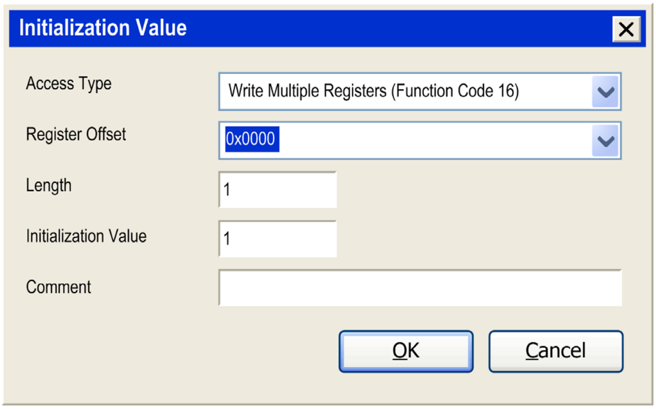
Step	Action
1	In the **Devices tree**, double-click **Generic_Modbus_Slave**. **Result:** The configuration window will be displayed.  Modbus-RTU/ASCII Slave Address [1..247] 1 Response Timeout [ms] 1000
2	Enter a **Slave Address** value for your device (choose a value from 1 to 247).
3	Choose a value for the **Response Timeout** (in ms).

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

Step	Action
1	Click the **Modbus Slave Channel** tab:  Modbus Slave Configuration Modbus Slave Channel Modbus Slave Init Modbus Name Access Type Trigger READ Offset Length Error Handling WRIT... Add Channel... Delete... Edit...
2	Click the **Add Channel** button:  ModbusChannel Channel Name: Channel 1 Access Type: Read/Write Multiple Registers (Function Code 23) Trigger: CYCLIC Cycle Time (ms): 100 Comment: READ Register Offset: 0x0000 Length: 1 Error Handling: Keep last Value WRITE Register Offset: 0x0000 Length: 1

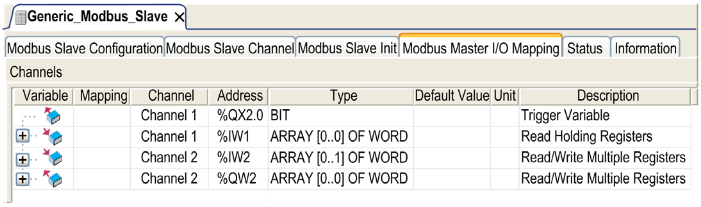
Step	Action
3	Configure the exchange: In the field **Channel**, you can add the following values: ● **Channel**: Enter a name for your channel. ● **Access Type** *(see page 29)*: Choose the exchange type: Read or Write or Read/Write multiple registers. ● **Trigger**: Choose the trigger of the exchange. It can be either **CYCLIC** with the period defined in **Cycle Time (ms)** field or started by a RISING EDGE on a boolean variable (this boolean variable is then created in the 'Modbus Master I/O Mapping' tab). ● **Comment**: Add a comment about this channel. In the field **READ Register** (if your channel is a Read or a Read/Write one), you can configure the %MW to be read on the Modbus slave. These registers will be mapped on %IW (see 'Modbus Master I/O Mapping' tab): ● **Offset**: Offset of the %MW to read. 0 means that the first object that will be read will be %MW0. ● **Length**: Number of %MW to be read. For example if 'Offset' = 2 and 'Length' = 3, the channel will read %MW2, %MW3 and %MW4. ● **Error Handling**: choose the behavior of the related %IW in case of loss of communication. In the field **WRITE Register** (if your channel is a Write or a Read/Write one), you can configure the %MW to be written to the Modbus slave. These registers will be mapped from %QW (see 'Modbus Master I/O Mapping' tab): ● **Offset**: Offset of the %MW to write. 0 means that the first object that will be written to will be %MW0. ● **Length**: Number of %MW to be written. For example if 'Offset' = 2 and 'Length' = 3, the channel will write %MW2, %MW3 and %MW4.
4	Click the **Delete** button to remove a channel. Click the **Edit** button to change the parameters of a channel.
5	Click **OK** to validate the configuration of this channel.

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

Step	Action
1	Click the **Modbus Slave Init** tab:  Modbus Slave Configuration Modbus Slave Channel Modbus Slave Init Modbus Line Access Type WRITE Offset Default Value Length Comment Move up New... Delete... Edit...
2	Click **New** to create a new initialization value:  Initialization Value Access Type: Write Multiple Registers (Function Code 16) Register Offset: 0x0000 Length: 1 Initialization Value: 1 Comment: The **Initialization Value** window contains the following parameters: ● **Access Type** *(see page 29)*: Choose the exchange type: Write multiple registers. ● **Register Offset**: Register number of register to be initialized. ● **Length**: Number of %MW to be written. For example if 'Offset' = 2 and 'Length' = 3, the channel will write %MW2, %MW3 and %MW4. ● **Initialization Value**: Value the registers are initialized with. ● **Comment**: Add a comment about this channel.

Step	Action
3	Click **Move up** to change the position of a value in the list. Click **Delete** to remove a value in the list. Click **Edit** to change the parameters of a value.
4	Click **OK** to create a new **Initialization Value**.

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

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Click the <b>Modbus Master I/O Mapping</b> tab:


<table>
<tr><th>Variable</th><th>Mapping</th><th>Channel</th><th>Address</th><th>Type</th><th>Default Value</th><th>Unit</th><th>Description</th></tr>
<tr><td></td><td></td><td>Channel 1</td><td>%QX2.0</td><td>BIT</td><td></td><td></td><td>Trigger Variable</td></tr>
<tr><td>⊞</td><td></td><td>Channel 1</td><td>%IW1</td><td>ARRAY [0..0] OF WORD</td><td></td><td></td><td>Read Holding Registers</td></tr>
<tr><td>⊞</td><td></td><td>Channel 2</td><td>%IW2</td><td>ARRAY [0..1] OF WORD</td><td></td><td></td><td>Read/Write Multiple Registers</td></tr>
<tr><td>⊞</td><td></td><td>Channel 2</td><td>%QW2</td><td>ARRAY [0..0] OF WORD</td><td></td><td></td><td>Read/Write Multiple Registers</td></tr>
</table></td></tr>
<tr><td>2</td><td>Double-click in a cell of the <b>Variable</b> column to open a text field.
Enter the name of a variable or click on the browse button <b>[...]</b> and chose a variable with the <b>Input Assistant</b>.</td></tr>
<tr><td>3</td><td>For more details about I/O mapping, refer to the EcoStruxure Machine Expert Programming Guide.</td></tr>
</table>

Modbus Manager

Introduction

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

Adding the Manager

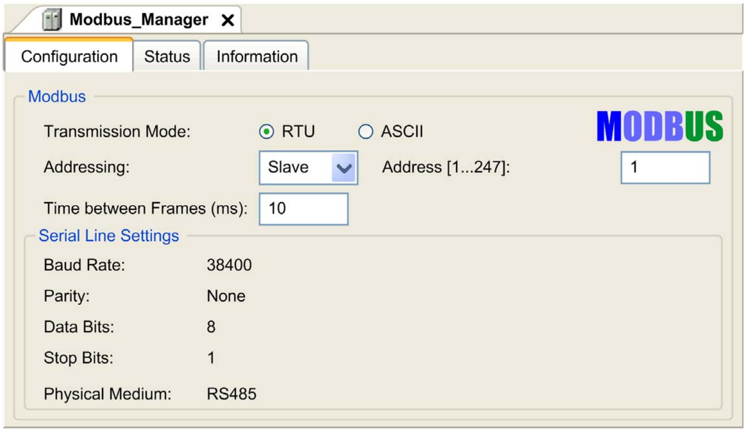
To add a Modbus Manager to your controller, select **Modbus_Manager** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

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

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

Modbus Manager Configuration

To configure the Modbus Manager, proceed as follows:

Step	Action
1	In the **Devices tree**, double-click **Modbus_Manager**. **Result:** The **Modbus_Manager** configuration window will be displayed.  Modbus_Manager Configuration / Status / Information Modbus Transmission Mode: ● RTU ○ ASCII Addressing: Slave Address [1...247]: 1 Time between Frames (ms): 10 Serial Line Settings Baud Rate: 38400 Parity: None Data Bits: 8 Stop Bits: 1 Physical Medium: RS485

Set the parameters as described in the following table:

Element	Description
Addressing	Specify the device type: ● Master ● Slave
Address	Modbus address of the device
Time between Frames (ms)	Time to avoid bus-collision This parameter must be identical for each Modbus device on the link.
Serial Line Settings	Parameters specified in the Serial Line configuration window.

Modbus Master

When the module is configured as a Modbus Master, the following Function Blocks are supported from the PLCCommunication Library:

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

For further information, see Function Block Descriptions of the PLCCommunication Library.

Modbus Slave

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

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

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

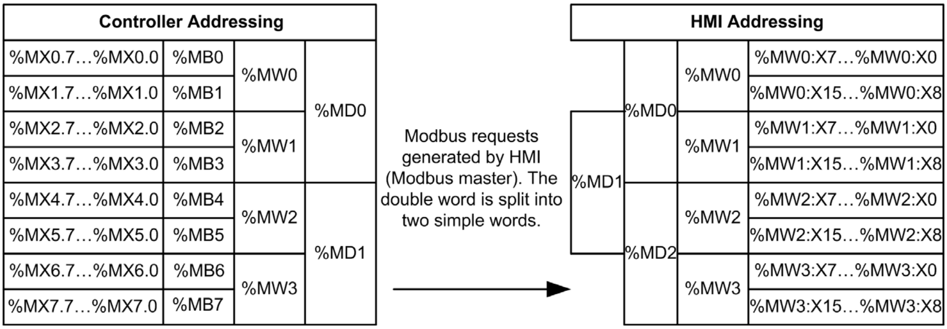
Sub-Function Code		Function
Dec	Hex	
10	0A	Clear Counters and Diagnostic Register
11	0B	Return Bus Message Count
12	0C	Return Bus Communication Error Count
13	0D	Return Bus Exception Error Count
14	0E	Return Slave Message Count
15	0F	Return Slave No Response Count
16	10	Return Slave NAK Count
17	11	Return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

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

Function Code Dec (Hex)	Sub-Function Code Dec (Hex)	Object ID	Object Name	Type	Value
43 (2B hex)	14 (E hex)	00 hex	Vendor code	ASCII String	Schneider Electric
		01 hex	Product code	ASCII String	Module reference eg:TM5PCRS2
		02 hex	Major / Minor revision	ASCII String	aa.bb.cc.dd (same as device descriptor)

NOTE: The following section describes the differences between Controller Modbus mapping and HMI Modbus mapping.
When the controller and the Magelis HMI are connected via Modbus (HMI is master of Modbus requests), the data exchange uses simple word requests.

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



Controller Addressing			
%MX0.7…%MX0.0	%MB0	%MW0	%MD0
%MX1.7…%MX1.0	%MB1		
%MX2.7…%MX2.0	%MB2	%MW1	
%MX3.7…%MX3.0	%MB3		
%MX4.7…%MX4.0	%MB4	%MW2	%MD1
%MX5.7…%MX5.0	%MB5		
%MX6.7…%MX6.0	%MB6	%MW3	
%MX7.7…%MX7.0	%MB7		

Modbus requests generated by HMI (Modbus master). The double word is split into two simple words.

HMI Addressing			
	%MD0	%MW0	%MW0:X7…%MW0:X0
			%MW0:X15…%MW0:X8
%MD1		%MW1	%MW1:X7…%MW1:X0
			%MW1:X15…%MW1:X8
	%MD2	%MW2	%MW2:X7…%MW2:X0
			%MW2:X15…%MW2:X8
		%MW3	%MW3:X7…%MW3:X0
			%MW3:X15…%MW3:X8

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

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

The following gives examples of memory match for the bits:

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

Adding a Modem

To add a Modem to the Modbus Manager, refer to Adding a Modem to a Manager *(see page 32)*.

Adding a Modem to a Manager

Introduction

A modem can be added to the following managers:

- ASCII Manager
- Modbus Manager
- EcoStruxure Machine Expert Network Manager

Adding a Modem to the Manager

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

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

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

For further information, refer to Modem Library.

Section 2.3
SerialConf Functions

Introduction

This section describes the **SerialConf** functions. These functions can be used for Serial Line management.

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

For further information on adding a library, refer to the EcoStruxure Machine Expert Programming Guide.

What Is in This Section?

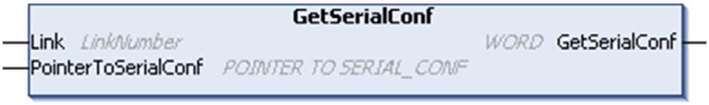
This section contains the following topics:

GetSerialConf: Get the Serial Line Configuration

Function Description

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

Graphical Representation



Parameter Description

Input	Type	Comment
Link	LinkNumber	Link is the communication port number.
PointerToSerialConf	POINTER TO SERIAL_CONF *(see page 37)*	PointerToSerialConf is the address of the configuration structure (variable of SERIAL_CONF type) in which the configuration parameters are stored. The ADR standard function must be used to define the associated pointer. (See the example below.)

Output	Type	Comment
GetSerialConf	WORD	This function returns: ● 0: The configuration parameters are returned ● 255: The configuration parameters are not returned because: ❍ the function was not successful ❍ the function is in progress

Example

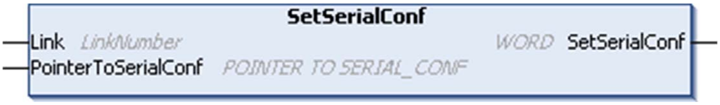
Refer to the SetSerialConf *(see page 36)* example.

SetSerialConf: Change the Serial Line Configuration

Function Description

SetSerialConf is used to change the serial line configuration.

Graphical Representation



NOTE: Changing the configuration of the Serial Line(s) port(s) during programming execution can interrupt ongoing communications with other connected devices.

⚠ **WARNING**

LOSS OF CONTROL DUE TO CONFIGURATION CHANGE

Validate and test all the parameters of the SetSerialConf function before putting your program into service.

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

Parameter Description

Input	Type	Comment
Link	LinkNumber	LinkNumber is the communication port number.
PointerToSerialConf	POINTER TO SERIAL_CONF *(see page 37)*	PointerToSerialConf is the address of the configuration structure (variable of SERIAL_CONF type) in which the new configuration parameters are stored. The ADR standard function must be used to define the associated pointer. (See the example below.) If 0, set the application default configuration to the serial line.

Output	Type	Comment
SetSerialConf	WORD	This function returns: ● 0: The new configuration is set ● 255: The new configuration is refused because: ❍ the function is in progress ❍ the input parameters are not valid

Example

```
VAR
  MySerialConf: SERIAL_CONF
  result: WORD;
END_VAR

(*Get current configuration of serial line 1*)
GetSerialConf(1, ADR(MySerialConf));

(*Change to modbus RTU slave address 9*)
MySerialConf.Protocol := 0;        (*Modbus RTU/Machine Expert protocol
(in this case CodesysCompliant selects the protocol)*)
MySerialConf.CodesysCompliant := 0; (*Modbus RTU*)
MySerialConf.address := 9;          (*Set modbus address to 9*)

(*Reconfigure the serial line 1*)
result := SetSerialConf(1, ADR(MySerialConf));
```

SERIAL_CONF: Structure of the Serial Line Configuration Data Type

Structure Description

The `SERIAL_CONF` structure contains configuration information about the serial line port. It contains these variables:

Variable	Type	Description
`Bauds`	`DWORD`	baud rate
`InterframeDelay`	`WORD`	minimum time (in ms) between 2 frames in Modbus (RTU, ASCII)
`FrameReceivedTimeout`	`WORD`	In the ASCII protocol, `FrameReceivedTimeout` allows the system to conclude the end of a frame at reception after a silence of the specified number of ms. If 0 this parameter is not used.
`FrameLengthReceived`	`WORD`	In the ASCII protocol, `FrameLengthReceived` allows the system to conclude the end of a frame at reception, when the controller received the specified number of characters. If 0, this parameter is not used.
`Protocol`	`BYTE`	0: Modbus RTU or Machine Expert (see `CodesysCompliant`)
		1: Modbus ASCII
		2: ASCII
`Address`	`BYTE`	Modbus address 0 to 255 (0 for Master)
`Parity`	`BYTE`	0: none
		1: odd
		2: even
`Rs485`	`BYTE`	0: RS232
		1: RS485
`ModPol (polarizartion resistor)`	`BYTE`	0: no
		1: yes
`DataFormat`	`BYTE`	7 bits or 8 bits
`StopBit`	`BYTE`	1: 1 stop bit
		2: 2 stop bits
`CharFrameStart`	`BYTE`	In the ASCII protocol, 0 means there is no start character in the frame. Otherwise, the corresponding ASCII character is used to detect the beginning of a frame in receiving mode. In sending mode, this character is added at the beginning of the user frame.
`CharFrameEnd1`	`BYTE`	In the ASCII protocol, 0 means there is no second end character in the frame. Otherwise, the corresponding ASCII character is used to detect the end of a frame in receiving mode. In sending mode, this character is added at the end of the user frame.

Variable	Type	Description
CharFrameEnd2	BYTE	In the ASCII protocol, 0 means there is no second end character in the frame. Otherwise, the corresponding ASCII character is used (along with CharFrameEnd1) to detect the end of a frame in receiving mode. In sending mode, this character is added at the end of the user frame.
CodesysCompliant	BYTE	0: Modbus RTU
		1: Machine Expert (when Protocol = 0)
CodesysNetType	BYTE	not used

Chapter 3
Profibus DP Slave PCI Expansion Module

Introduction

This chapter describes how to configure the TM5PCDPS Profibus DP Slave PCI Expansion Module.

What Is in This Chapter?

This chapter contains the following sections:

Section 3.1
Profibus DP Slave PCI Expansion Module Configuration

Introduction

This section describes the configuration of Profibus DP Slave PCI expansion modules.

What Is in This Section?

This section contains the following topics:

Add a Profibus DP Slave PCI Expansion Module

Overview

With the Profibus protocol the data is exchanged according to the master-slave principle. Only the master can initialize communication. The slaves respond to requests from masters. Several masters can coexist on the same bus. In this case, the slave I/O can be read by all the masters. However, a single master has write access to the outputs. The number of data items exchanged is defined during the configuration.

For the Profibus Master, the GSD file of the TM5 Profibus DP PCI module is available on *www.schneider-electric.com*.

There are 2 types of exchange services supported by this module:

- I/O cyclic frames exchanges *(see page 46)*
- acyclic data exchanges with Profibus DPV1 function *(see page 48)*

Add a Profibus DP Slave PCI Expansion Module

Add a Profibus DP slave PCI expansion module to your project, as described in the Add a PCI Expansion Module chapter *(see page 13)*, by selecting the TM5PCDPS module.

NOTE: Adding Profibus increases the associated task cycle time by several milliseconds and the starting time by several seconds.

Configure the Profibus DP Slave PCI Expansion Module

Profibus DP Slave PCI Expansion Module Configuration

To open the configuration window of a Profibus DP slave PCI expansion module:

Step	Action
1	Select the **TM5PCDPS** (Profibus DP Slave PCI) expansion module in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes. For more information on adding a device to your project, refer to: • Using the Drag-and-drop Method • Using the Contextual Menu or Plus Button
2	Double-click **TM5PCDPS** node in the **Devices tree**. **Result**: The configuration window for Profibus DP Slave is displayed.

The following parameters are provided in the **Profibus Configuration** tab:

Parameter	Value	Default Value	Description
BusAddr	1...126	2	Profibus DP slave address. The address 126 is reserved.
BaudRate (KBaud)	9.6 19.2 45.45 93.75 187.5 500 1500 3000 6000 12000 Auto	Auto	Profibus transmission rate
DPV1Enable	TRUE FALSE	TRUE	TRUE = Profibus DPV1 functions for acyclic communication *(see page 48)* enable
SyncSupported	TRUE FALSE	TRUE	TRUE = sync mode, that supports the sync command, enable
FreezeSupported	TRUE FALSE	TRUE	TRUE = freeze mode, that supports the freeze command, enable
NoAddrChangeSupported	TRUE FALSE	TRUE	TRUE = blocks a Profibus master from changing the address

Input / Output Devices Objects

Introduction

To exchange data between the controller and a Profibus master, it is important to understand the role of the TM5PCDPS PCI expansion module.

The TM5PCDPS is an intermediate between the Profibus master and the controller, and data is exchanged through the use of virtual I/O devices that you define when configuring the PCI expansion module. The virtual devices are not physical I/O modules, but are simply logical input and output objects within the expansion module that you can then map to memory within the controller. These input and output objects are read from and written to by the Profibus master. In turn, the PCI module reads and writes this data to I/O memory locations in the controller so that you can use the data within your application program.

Virtual I/O Devices

The virtual I/O devices you define within the TM5PCDPS PCI expansion module can be either input or output, and can vary in size as defined by the table:

Name	Number of I/O	Format
12 word input (0x5B)	12	word
12 word output (0x6B)	12	word
16 byte input (0x1F)	16	byte
16 byte output (0x2F)	16	byte
2 byte input (0x11)	2	byte
2 byte output (0x21)	2	byte
2 word input (0x51)	2	word
2 word output (0x61)	2	word
20 word input (0x40, 0x53)	20	word
20 word output (0x80, 0x53)	20	word
32 word input (0x40, 0x5F)	32	word
32 word output (0x80, 0x5F)	32	word
4 word input (0x53)	4	word
4 word output (0x63)	4	word
8 byte input (0x17)	8	byte
8 byte output (0x27)	8	byte
8 word input (0x57)	8	word
8 word output (0x67)	8	word

Once you have defined these virtual input and/or output devices within the TM5PCDPS PCI expansion module, you can then map these devices to memory locations within the controller. The type of memory objects you map these virtual I/O devices to depends on the type of exchange you define between the master and the slave.

Section 3.2
Data Exchange

Introduction

This section provides further information on the exchange of data between the TM5PCDPS Profibus DP Slave PCI expansion module and the Profibus master.

What Is in This Section?

This section contains the following topics:

I/O Cyclic Exchange

Introduction

In order to exchange input / output data between the Profibus DP slave PCI expansion module and the Profibus master in a cyclic way, define the variables in the **Profibus-Modules I/O Mapping** tab.

The `%IW` addresses of the controller are the output values supplied by the Profibus DP master.

The `%QW` addresses of the controller are applied to the input of the Profibus DP master.

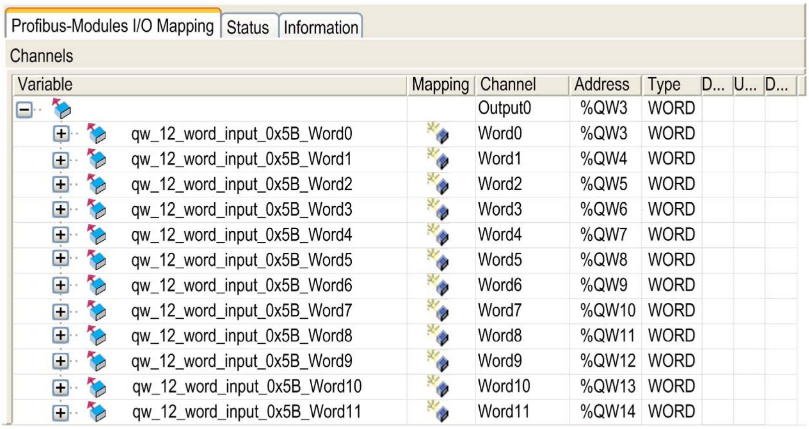
Create Your I/O Mapping Table for the TM5PCDPS PCI Module

To create your I/O mapping table for the TM5PCDPS, select the **Profibus I/O** module you want in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on the **TM5PCDPS** node of the **Devices tree**.

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

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

The variables for the exchange are automatically created in the `%IWx` and `%QWx` of the **Profibus-Module I/O Mapping** tab. Double-click the I/O device you added to access this screen.



Profibus-Modules I/O Mapping | Status | Information

Channels

Variable	Mapping	Channel	Address	Type	D...	U...	D...
		Output0	%QW3	WORD			
qw_12_word_input_0x5B_Word0		Word0	%QW3	WORD			
qw_12_word_input_0x5B_Word1		Word1	%QW4	WORD			
qw_12_word_input_0x5B_Word2		Word2	%QW5	WORD			
qw_12_word_input_0x5B_Word3		Word3	%QW6	WORD			
qw_12_word_input_0x5B_Word4		Word4	%QW7	WORD			
qw_12_word_input_0x5B_Word5		Word5	%QW8	WORD			
qw_12_word_input_0x5B_Word6		Word6	%QW9	WORD			
qw_12_word_input_0x5B_Word7		Word7	%QW10	WORD			
qw_12_word_input_0x5B_Word8		Word8	%QW11	WORD			
qw_12_word_input_0x5B_Word9		Word9	%QW12	WORD			
qw_12_word_input_0x5B_Word10		Word10	%QW13	WORD			
qw_12_word_input_0x5B_Word11		Word11	%QW14	WORD			

Configure a Virtual I/O Device Added to the TM5PCDPS PCI Module

The tabs of the configuration window are described in the table below:

The configuration window contains the following tabs:

Tab Name	Description
Profibus-Modules I/O Mapping	This tab contains the variables for data exchange.
Status	This tab provides diagnostic information *(see page 50)*.
Information	This tab provides further information on the selected input or output module.

Profibus Virtual I/O Behavior

The following table describes the status of the Profibus I/O depending on:

- the Controller status
- the Profibus communication state (value of **PROFIBUS_R.i_CommState** of **PLCSystem**)

Controller State	Controller Profibus I/O State
STOPPED	The `%QW` addresses are managed as it is configured in PLC Settings (refer to the chapter Controller States and Behaviors) The `%IW` addresses are managed as it is configured in PLC Settings (refer to the chapter Controller States and Behaviors)
RUNNING	The `%IW` addresses are updated by the master The `%QW` addresses are sent to the master
HALT	The `%QW` addresses are managed as it is configured in PLC Settings (refer to the chapter Controller States and Behaviors) The `%IW` addresses keep the last correct value sent by the master

Communication Status	Value of PROFIBUS_R.i_CommState	Controller Profibus I/O State
Profibus Master is stopped	4 (Operate mode)	The `%IW` addresses are set to 0 by the master The `%QW` addresses are sent to the master
Watchdog is detected	2 (Stop)	The `%QW` addresses are not sent to the master The `%IW` addresses keep the last correct value sent by the master

Acyclic Exchange with Profibus DPV1 Functions

Introduction

The Profibus DPV1 enhancement additionally supports acyclic data exchange between a Profibus DPV1 master and DPV1 slaves. It allows access to `%MW` variables.

To use these functions between a Profibus DPV1 master and the TM5PCDPS PCI module, the parameter **DPV1Enable** must be set to TRUE (default value) *(see page 42)*.

Data Addressing

Data addressing in the logic controller is `%MW`.

The **Profibus status** of the controller must be in **Operate** state, therefore it can be updated even if the logic controller is not running.

The `%MW` variables are automatically updated by the I/O driver whenever a DPV1 message is received.

It is based on Profibus DPV1 read and write functions.

The logic address is the number of the `%MW` addressed.

Addressing

2 different types of addressing are available for acyclic exchange:

Addressing Type	Number of Requests for Read/Write `%MW` Variables	Description
Direct Addressing	1	The address of the `%MW` variable is coded directly by **Slot** and **Index** fields. See restrictions in the Note below.
Indirect Addressing	2	● The first request sends the address of the first `%MW` that the master will read or write. ● The second request reads or writes one or several values of the `%MW` variable.

NOTE:
The following restrictions apply to direct addressing:

- **Slot** field (**DU1**): value 0xFF is not allowed
- **Index** field (**DU2**): values 0xFF, 0xE9 and 0xEA are not allowed

The table shows how to create requests for accessing the %MW from the Profibus DPV1 master:

Addressing		DU0: DPV1 Function Number	DU1: Slot	DU2: Index	DU3: Length (in bytes)	DPV1 Data Frame
		1 byte	1 byte	1 byte	1 byte	N byte
Direct Addressing	write	5F hex (write)	MSB of the %MW address	LSB of the %MW address	length to read	values to write
	read	5E hex (read)	MSB of the %MW address	LSB of the %MW address	length to write	–
Indirect Addressing	send address (Step 1)	5F hex (write)	1	E9 hex	2	%MW address
	read (Step 2)	5E hex (read)	1	EA hex	length to read	–
	write (Step 2)	5F hex (write)	1	EA hex	length to write	values to write

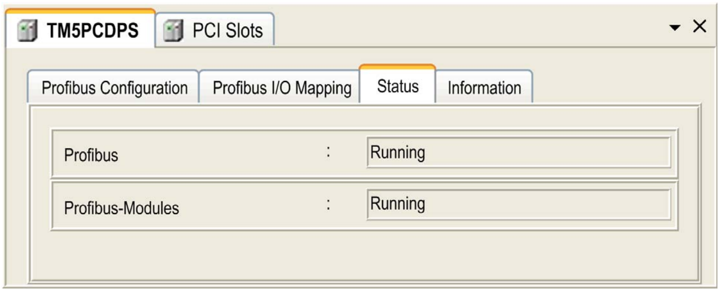
NOTE: The Length field has to have an even value (the length in byte of one %MW is 2).

Section 3.3
Diagnostic

Diagnostic Information

Displaying General Diagnostics Data

To display general diagnostic data, open the **Status** tab of the TM5PCDPS configuration window.



Profibus : Running

Profibus-Modules : Running

Monitoring the Status of the TM5PCDPS PCI Module

You can monitor the status of the TM5PCDPS PCI module with the `PROFIBUS_R` system data type described in the M258 Controller PLCSystem Library Guide or LMC058 Controller PLCSystem Library Guide depending on your controller.

Fallback Management

When there is a Profibus communication interruption (`i_CommState`=0), the outputs of the TM5PCDPS are maintained to the last state transmitted by the Profibus master.

The Fail Safe Mode as defined by the Profibus DP standard is not supported by the TM5PCDPS module.

Messages on Detected Errors

Use `i_CommError` of the `PROFIBUS_R` system data type to visualize the error displayed.

No error has been detected:

Name	Value	Meaning
SUCCESS	0 hex	No error detected.

Runtime error has been detected:

Name	Value	Meaning
WATCHDOG_TIMEOUT	C000000C hex	The watchdog time has been exceeded.

Initialization errors have been detected:

Name	Value	Meaning
INIT_FAULT	C0000100 hex	The initialization was not successful.
DATABASE_ACCESS_FAILED	C0000101 hex	Access to data memory was not successful.

Configuration errors have been detected:

Name	Value	Meaning
NOT_CONFIGURED	C0000119 hex	The TM5PCDPS PCI module is not configured.
CONFIGURATION_FAULT	C0000120 hex	A configuration error has been detected.
INCONSISTENT_DATA_SET	C0000121 hex	Inconsistent set data have been detected.
DATA_SET_MISMATCH	C0000122 hex	A mismatch of set data has been detected.
INSUFFICIENT_LICENSE	C0000123 hex	An insufficient license has been detected.
PARAMETER_ERROR	C0000124 hex	A parameter error has been detected.
INVALID_NETWORK_ADDRESS	C0000125 hex	The network address is not correct.
SECURITY_MEMORY	C0000126 hex	The security memory is not available.

Network errors have been detected:

Name	Value	Meaning
COMM_NETWORK_FAULT	C0000140 hex	A network communication error has been detected.
COMM_CONNECTION_CLOSED	C0000141 hex	The communication connection has been closed.
COMM_CONNECTION_TIMEOUT	C0000142 hex	A communication connection timeout has been detected.
COMM_DUPLICATE_NODE	C0000144 hex	A duplicate node has been detected.
COMM_CABLE_DISCONNECT	C0000145 hex	A disconnected cable has been detected.
PROFIBUS_CONNECTION_TIMEOUT	C009002E hex	A Profibus connection timeout has been detected.

Glossary

A

ASCII

(*American standard code for Information Interchange*) A protocol for representing alphanumeric characters (letters, numbers, certain graphics, and control characters).

C

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

control network

A network containing logic controllers, SCADA systems, PCs, HMI, switches, ...

Two kinds of topologies are supported:

- flat: all modules and devices in this network belong to same subnet.
- 2 levels: the network is split into an operation network and an inter-controller network.

These two networks can be physically independent, but are generally linked by a routing device.

CRC

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

E

electronic module

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

F

function block

A programming unit that has 1 or more inputs and returns 1 or more outputs. FBs are called through an instance (function block copy with dedicated name and variables) and each instance has a persistent state (outputs and internal variables) from 1 call to the other.

Examples: timers, counters

H

HMI

(*human machine interface*) An operator interface (usually graphical) for human control over industrial equipment.

I

I/O

(*input/output*)

L

LSB

(*least significant bit/byte*) The part of a number, address, or field that is written as the right-most single value in conventional hexadecimal or binary notation.

M

Magelis

The commercial name for Schneider Electric's range of HMI terminals.

Modbus

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

ms

(*millisecond*)

MSB

(*most significant bit/byte* The part of a number, address, or field that is written as the left-most single value in conventional hexadecimal or binary notation.

N

NAK

(*negative acknowledge*)

network

A system of interconnected devices that share a common data path and protocol for communications.

node

An addressable device on a communication network.

P

PCI

(*peripheral component interconnect*) An industry-standard bus for attaching peripherals.

Profibus DP

(*Profibus decentralized peripheral*) An open bus system uses an electrical network based on a shielded 2-wire line or an optical network based on a fiber-optic cable. DP transmission allows for high-speed, cyclic exchange of data between the controller CPU and the distributed I/O devices.

R

RS-232

A standard type of serial communication bus, based on 3 wires (also known as EIA RS-232C or V.24).

RS-485

A standard type of serial communication bus, based on 2 wires (also known as EIA RS-485).

S

string

A variable that is a series of ASCII characters.

Index