

Modicon M340

Serial Link

User Manual

(Original Document)

12/2018

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.

© 2018 Schneider Electric. All rights reserved.

Table of Contents



	Safety Information	7
	About the Book	11
Part I	Hardware Installation for Modbus Serial and Character Mode Communications	15
Chapter 1	Introduction to Serial Communications	17
	Modbus Serial and Character Mode Communications	18
	Presentation of the Serial Link on Modicon M340 Processors	19
	Standards and Certifications	24
	Wiring Considerations	25
Chapter 2	Serial Communication Architectures	27
	Modbus Line Termination and Polarization (RS485)	28
	Connecting Modbus Devices (RS485)	30
	Connecting Data Terminal Equipment (DTE) (RS232)	33
	Connecting Data Circuit-terminating Equipment (DCE) (RS232)	35
	Cabling	37
Part II	Software Implementation of Modbus Serial and Character Mode Communications	41
Chapter 3	Installation Methodology	43
	Introduction to the Installation Phase	43
Chapter 4	Modbus Serial Communication for Modicon M340 Processors	45
4.1	Generalities	46
	About Modbus Serial	47
	Performance	48
	How to Access the Serial Link Parameters	50
4.2	Modbus Serial Communication Configuration	53
	Modbus Serial Communication Configuration Screen	54
	Application-linked Modbus Parameters	56
	Signal and Physical Line Parameters in Modbus	58
	Transmission-linked Modbus Parameters	60
4.3	Modbus Serial Communication Programming	62
	Services Supported by a Modbus Link Master Processor	63
	Services Supported by a Modbus Link Slave Processor	65

4.4	Debugging Modbus Serial Communication	67
	Modbus Serial Communication Debug Screen	67
Chapter 5	Character Mode Communication for Modicon M340 Processors	69
5.1	Generalities	70
	About Character Mode Communication	71
	Performance	72
5.2	Character Mode Communication Configuration	73
	Character Mode Communication Configuration Screen	74
	Message End Detection Parameters in Character Mode	76
	Signal and Physical Line Parameters in Character Mode	78
	Transmission Parameters in Character Mode	80
5.3	Character Mode Communication Programming	82
	Character Mode Communication Functions	82
5.4	Debugging Character Mode communication	84
	Character Mode Communication Debug Screen	84
Chapter 6	Language Objects of Modbus and Character Mode Communications	87
6.1	Language Objects and IODDTs of Modbus and Character Mode Communications	88
	Introduction to the Language Objects for Modbus and Character Mode Communications	89
	Implicit Exchange Language Objects Associated with the Application-Specific Function	90
	Explicit Exchange Language Objects Associated with the Application-Specific Function	91
	Management of Exchanges and Reports with Explicit Objects	93
6.2	General Language Objects and IODDTs for Communication Protocols	96
	Details of IODDT Implicit Exchange Objects of Type T_COM_STS_GEN	97
	Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN	98

6.3	Language Objects and IODDTs Associated with Modbus Communication	100
	Details concerning Explicit Exchange Language Objects for a Modbus Function.	101
	Details of the IODDTs Implicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT	102
	Details of the IODDTs Explicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT	103
	Details of language objects associated with configuration Modbus mode	106
6.4	Language Objects and IODDTs associated with Character Mode Communication	108
	Details concerning Explicit Exchange Language Objects for Communication in Character Mode	109
	Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX.	110
	Details of IODDT Explicit Exchange Objects of Type T_COM_CHAR_BMX.	111
	Details of language objects associated with configuration in Character mode	113
6.5	The IODDT Type T_GEN_MOD Applicable to All Modules	115
	Details of the Language Objects of the IODDT of Type T_GEN_MOD	115
Chapter 7	Dynamic Protocol Switching	117
	Changing Protocol with Modicon M340 Processors	117
Part III	Quick Start : Example of Serial Link Implementation	119
Chapter 8	Description of the Application	121
	Overview of the Application	121
Chapter 9	Installing the Application Using Control Expert	123
9.1	Presentation of the Solution Used	124
	The Different Steps in the Process Using Control Expert	124
9.2	Developing the Application	125
	Creating the Project	126
	Declaration of Variables	132
	Using a Modem	136
	Procedure for Programming	138
	Programming Structure	139
	Programming	142
Chapter 10	Starting the Application	151
	Execution of the Application in Standard Mode.	151
Glossary	155
Index	163

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.

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
<ul style="list-style-type: none">• 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 be 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 (English version prevails):

- 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 actually 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



At a Glance

Document Scope

This manual describes the principle for hardware and software implementation of character mode and Modbus communication for Modicon M340 PLCs.

Validity Note

This documentation is valid for EcoStruxure™ Control Expert 14.0 or later.

The technical characteristics of the devices described in the present document also appear online. To access the information online:

Step	Action
1	Go to the Schneider Electric home page www.schneider-electric.com .
2	In the Search box type the reference of a product or the name of a product range. <ul style="list-style-type: none">• Do not include blank spaces in the reference or product range.• To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the Product Datasheets search results and click on the reference that interests you. If you entered the name of a product range, go to the Product Ranges search results and click on the product range that interests you.
4	If more than one reference appears in the Products search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click Download XXX product datasheet .

The characteristics that are presented 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 X80, BMXNOM0200 Serial Link Module, User Manual	EIO0000002696 (English), EIO0000002697 (French), EIO0000002698 (German), EIO0000002699 (Italian), EIO0000002700 (Spanish), EIO0000002701 (Chinese)
Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications	EIO0000002726 (English), EIO0000002727 (French), EIO0000002728 (German), EIO0000002730 (Italian), EIO0000002729 (Spanish), EIO0000002731 (Chinese)
EcoStruxure™ Control Expert, Operating Modes	33003101 (English), 33003102 (French), 33003103 (German), 33003104 (Spanish), 33003696 (Italian), 33003697 (Chinese)
EcoStruxure™ Control Expert, Communication, Block Library	33002527 (English), 33002528 (French), 33002529 (German), 33003682 (Italian), 33002530 (Spanish), 33003683 (Chinese)
EcoStruxure™ Control Expert, I/O Management, Block Library	33002531 (English), 33002532 (French), 33002533 (German), 33003684 (Italian), 33002534 (Spanish), 33003685 (Chinese)

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

Product Related Information

 **WARNING**

UNINTENDED EQUIPMENT OPERATION

The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise should be allowed to program, install, alter, and apply this product.

Follow all local and national safety codes and standards.

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

Part I

Hardware Installation for Modbus Serial and Character Mode Communications

In This Part

This part provides an introduction to hardware installation for Modbus serial and Character Mode communications.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Introduction to Serial Communications	17
2	Serial Communication Architectures	27

Chapter 1

Introduction to Serial Communications

Subject of this Chapter

This chapter introduces the serial communications on Modicon M340 Platform.

The table below gives a quick overview of the two possibilities for implementing serial link communications:

Using the integrated port of the M340 CPU	Using the BMX NOM 0200 communication module (<i>see Modicon X80, BMXNOM0200 Serial Link Module, User Manual</i>)
<ul style="list-style-type: none">- Limited transmission speed- Non isolated serial lines- Provision of power supply to terminal equipment	<ul style="list-style-type: none">- Increased number of available communication channels- Handling of modem specific RS232 signals- Higher transmission speed- Two isolated RS485 serial lines

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Modbus Serial and Character Mode Communications	18
Presentation of the Serial Link on Modicon M340 Processors.	19
Standards and Certifications	24
Wiring Considerations	25

Modbus Serial and Character Mode Communications

General

The serial links support two communication protocols:

- Modbus Serial
- Character Mode

Modbus Protocol

Modbus is a standard protocol with the following properties:

- Establishes client/server communication between different modules within a bus or serial link. The client is identified by the master and the slave modules represent the servers.
- Is based on a mode of data exchange composed of requests and responses offering services via different function codes.
- Establishes a means of exchanging frames from Modbus-type applications in two types of code:
 - RTU mode
 - ASCII mode

The exchange management procedure is as follows:

- Only one device may send data on the bus.
- Exchanges are managed by the master. Only the master may initiate exchanges. Slaves may not send messages without first being invited to do so.
- In the event of an invalid exchange, the master repeats the request. The slave to which the request is made is declared absent by the master if it does not respond within a given time scale.
- If the slave does not understand or cannot process the request, it sends an exception response to the master. In this case, the master may or may not repeat the request.

Two types of dialogue are possible between master and slave(s):

- The master sends a request to a specific slave number and awaits its response.
- The master sends a request to all the slaves without awaiting a reply (the general broadcast principle).

Character Mode Communication

Character mode is a point-to-point mode of data exchange between two entities. Unlike Modbus Protocol, it does not establish hierarchically structured serial link communications or offer services via function codes.

Character Mode is asynchronous. Each item of textual information is sent or received character by character at irregular time intervals. The time taken by the exchanges can be determined from the following properties:

- One or two end-of-frame characters.
- Timeout.
- Number of characters.

Presentation of the Serial Link on Modicon M340 Processors.

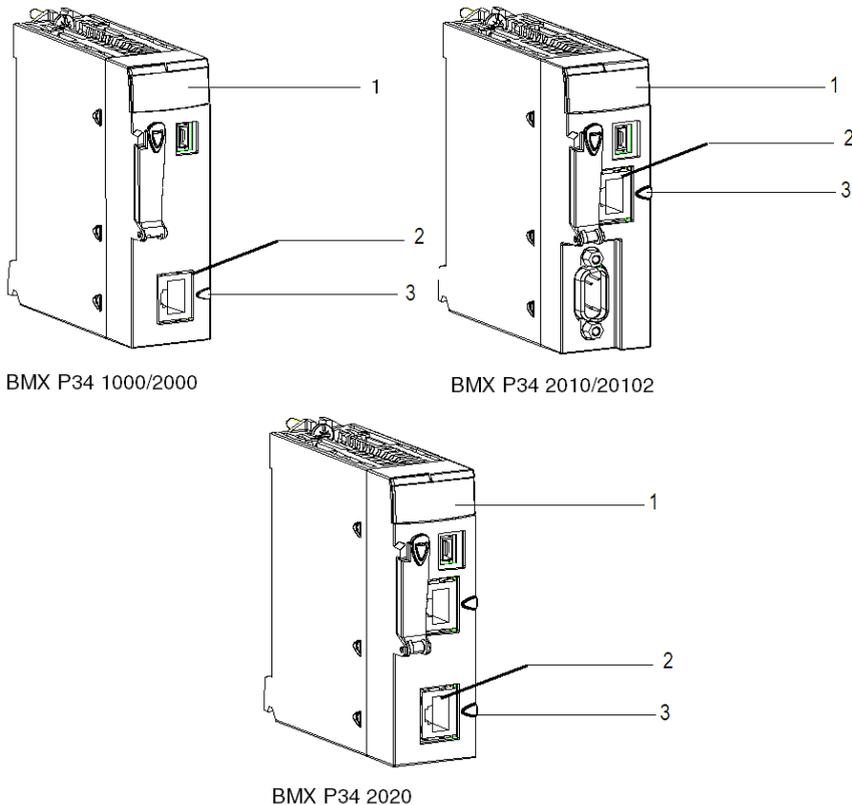
General

The following processors have an integrated communication channel dedicated to serial communications, and enable communication via serial link:

- BMX P34 1000
- BMX P34 2000
- BMX P34 2010
- BMX P34 20102
- BMX P34 2020

Serial Port Localization

The illustration below shows the localization of the serial port of the Modicon M340 processors:



These processors are composed of the following elements:

Address	Description
1	Processor status LEDs on the front
2	Integrated channel (channel 0) dedicated to the serial link.
3	Serial port identification ring (black)

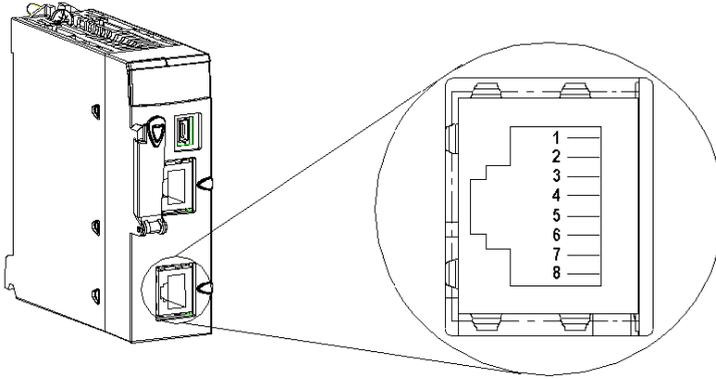
Visual Diagnostic of Serial Communication

The status of the serial communication is indicated by a yellow SER COM LED on the front of these processors:

- LED flashing: Serial communication is in progress.
- LED off: Serial communication is not in progress.

Serial Port Connector Description

The illustration below shows the RJ45 serial port:



The RJ45 connector has eight pins. The pins used vary according to the physical link used.

The pins used by the RS232 serial link are:

- Pin 1: RXD signal
- Pin 2: TXD signal
- Pin 3: RTS signal
- Pin 6: CTS signal
- Pin 8: Potential serial link grounding (0 V)

The pins used by the RS485 serial link are:

- Pin 4: D1 signal
- Pin 5: D0 signal

Pin 7 is used solely to supply power to human-machine interfaces or small devices via the serial link cable:

- Pin 7: Serial link power supply: 5 VDC/190 mA

Detailed characteristics

DC characteristics:

- Maximum stabilized power consumption: 190 mA,
- Minimum voltage on CPU connector for 190 mA: 4.9 V,
- Maximum voltage on CPU connector for 190 mA: 5.25 V,
- Maximum voltage on CPU connector with no load: 5.5 V.

AC characteristics:

- Capacitor charge: (on 5 V)
 - Maximum 1 μ F ceramic capacitor
 - 10 μ F tantalum
- Pump charge startup: (on 5 V)
 - 4 x 1 μ F ceramic capacitor
 - 2 x 10 μ F tantalum

NOTE: The four-wire RS232, the two-wire RS485 and the two-wire RS485 with power supply all use the same female RJ45 connector. Only the signal cabling is different.

Electrical Line Characteristics

The RS232 and the RS485 lines are not isolated.

In case of non equipotential earth between connected equipments (cables equal or longer than 30 m), it is necessary to use a TWDXCAISO isolator module in RS485 mode.

The RS485 line polarisation is integrated into the PLC and automatically enabled or disabled by the system according to the configuration chosen in the Control Expert screen:

- Modbus master : The line polarisation is enabled.
- Modbus slave : The line polarization is disabled.
- Character mode : The line polarization is disabled.

The polarisation is not affected by dynamic protocol switching. The polarization resistors' value is 560 ohms.

In RS232 mode, no polarization is required.

There is no built-in line termination.

Channel Specifications

The channel of these processors includes:

- One non-isolated RS485 physical interface,
- One non-isolated RS232 physical interface,
- Modbus Serial (ASCII and RTU) and Character Mode communication types.

The link specifications for the two protocols are:

	Modbus Serial / RS485	Modbus Serial / RS232	Character Mode / RS485	Character Mode / RS232
Type	Master/Slave	Master/Slave	Half Duplex	Full Duplex
Flow	19200 bauds. Parameters can be set from 300 bauds to 38400 bauds.	19200 bauds. Parameters can be set from 300 bauds to 38400 bauds.	9600 bauds. Parameters can be set from 300 bauds to 38400 bauds.	9600 bauds. Parameters can be set from 300 bauds to 38400 bauds
Number of devices	32	32	–	–
Authorized slave addresses	1 to 247	1 to 247	–	–
Max. length of Bus without branching	1000 m (15 m with Branching)	15 m	1000 m (15 m with Branching)	15 m
Message Size	Modbus Serial: <ul style="list-style-type: none"> • RTU: 256 bytes (252 bytes of data) • ASCII: 513 bytes (2x252 bytes of data) 	Modbus Serial: <ul style="list-style-type: none"> • RTU: 256 bytes (252 bytes of data) • ASCII: 513 bytes (2x252 bytes of data) 	1024 bytes	1024 bytes
Utilities	Read words/bits. Write words/bits. Diagnostics.	Read words/bits. Write words/bits. Diagnostics.	Send character strings. Receive character strings.	Send character strings. Receive character strings.

Standards and Certifications

Online Help

From the Control Expert online help, you can access the standards and certifications that apply to the modules in this product line by referring to the *Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications* guide.

Download

Click the link that corresponds to your preferred language to download the standards and certifications (PDF format) that apply to the modules in this product line:

Language	
English	<i>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</i>
French	<i>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</i>
German	<i>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</i>
Italian	<i>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</i>
Spanish	<i>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</i>
Chinese	<i>Modicon M580, M340, and X80 I/O Platforms, Standards and Certifications</i>

Wiring Considerations

Operational Consideration

 WARNING
UNINTENDED EQUIPMENT OPERATION Although you can connect or disconnect the serial link wire on the BMX P34 20x0 CPUs while the power to the station is on, doing so can interrupt the application in progress. Failure to follow these instructions can result in death, serious injury, or equipment damage.

The Link

The following situations can create a temporary disruption in the application or communications:

- The RJ45 connector is connected or disconnected when the power is on.
- Modules are re-initialized when the power is switched back on.

Chapter 2

Serial Communication Architectures

Subject of this Chapter

This chapter provides an introduction to architectures that use serial communication, as well as cabling requirements.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Modbus Line Termination and Polarization (RS485)	28
Connecting Modbus Devices (RS485)	30
Connecting Data Terminal Equipment (DTE) (RS232)	33
Connecting Data Circuit-terminating Equipment (DCE) (RS232)	35
Cabling	37

Modbus Line Termination and Polarization (RS485)

Overview

A multi-point Modbus network must have line termination and polarization.

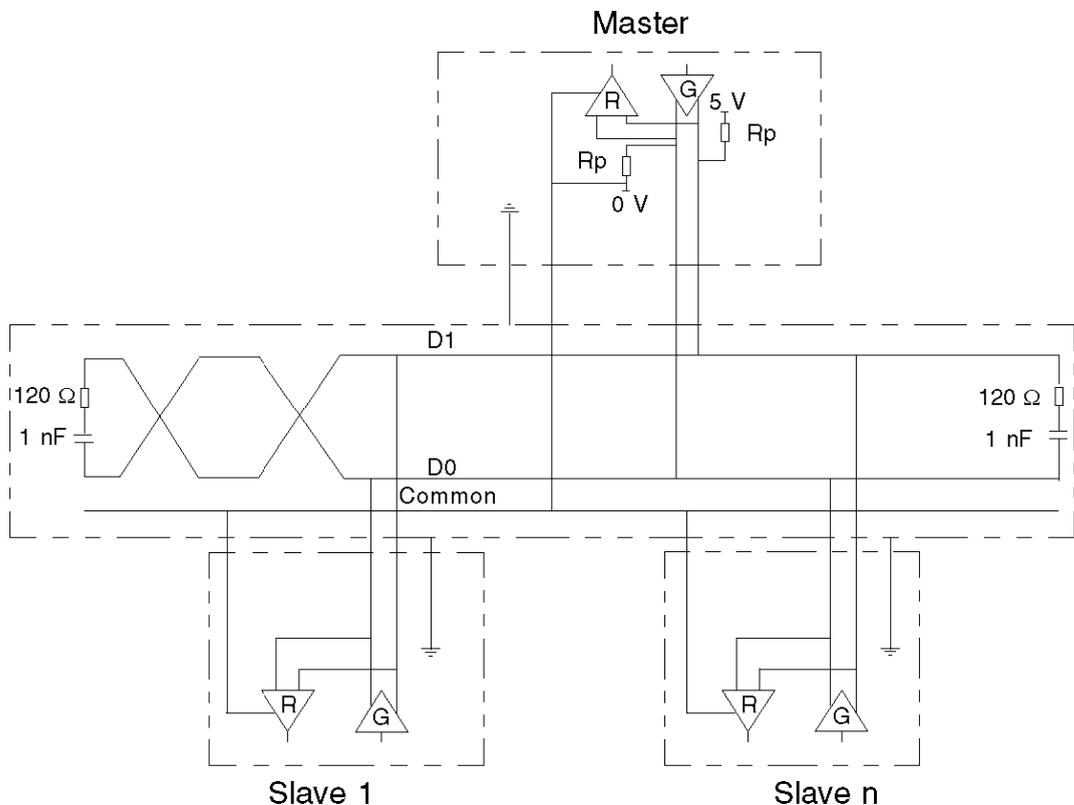
Equipment connectable to this bus are:

- Other PLCs like M340, Premium, Quantum, Twido or Nano
- Schneider Automation devices like Altivar, Security module XPS, SEPAM, XBT or Momentum
- Other Modbus protocol compliant devices
- Modem, Hub

An example of **multi-point Modbus network** (see page 32) including a BMX P34 2010 processor is presented in this manual.

NOTE: A point to point Modbus network can also be performed.

Electrical schema of line termination and polarization:



Line Termination

Line termination is made externally: it consists of two 120 Ω resistors and 1 nF capacitor placed at each end of the network (VW3 A8 306 RC or VW3 A8 306 DRC).

Don't place line termination at the end of a derivation cable.

Line Polarization

On a Modbus line, polarization is needed for an RS485 network.

- If the M340 CPU is used as a master, **it is automatically driven by the system** (*see page 22*) so there is no need of external polarization.
- If the M340 CPU is used as a slave, the polarization must be implemented by two 450 to 650 Ω resistors (R_p) connected on the RS485 balanced pair:
 - a pull-up resistor to a 5 V voltage on the D1 circuit,
 - a pull-down resistor to the common circuit on D0 circuit.

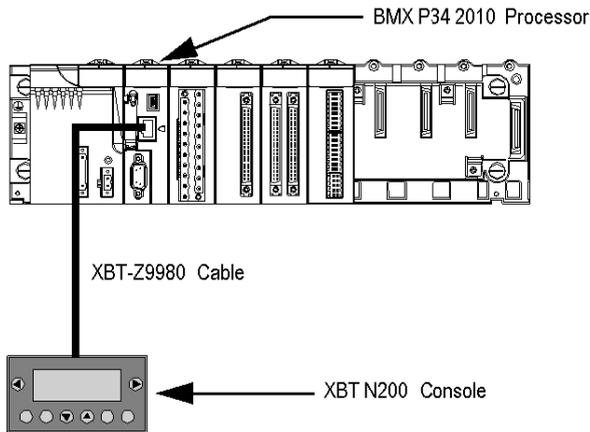
Connecting Modbus Devices (RS485)

General

The following pages present two examples of Modbus devices connection and a Modbus serial link architecture.

Connecting Modbus Devices that are Powered via the Serial Link

The illustration below shows how a BMX P34 2010 processor is connected to an XBT N200 console powered by the Modbus serial link:



The devices are configured as follows:

- The BMX P34 2010 processor is configured as slave,
- The XBT N200 human-machine interface is configured as master.

The XBT-Z9980 cable has the following properties:

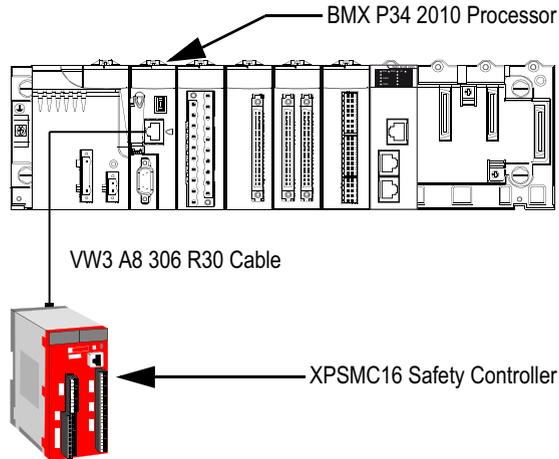
- Connection: 2 male RJ45 connectors
- Wiring: 2 wires for the RS485 physical line and 2 for the serial link power supply

Connecting Modbus Devices that are not Powered via the Serial Link

This architecture consists of the following elements:

- A BMX P34 2010 processor,
- An XPSMC16 safety controller.

The illustration below shows how a BMX P34 2010 processor is connected to an XPSMC16 safety controller:



The devices are configured as follows:

- The BMX P34 2010 processor is configured as master,
- The XPSMC16 safety controller is configured as slave.

The VW3 A8 306 R30 cable has the following properties:

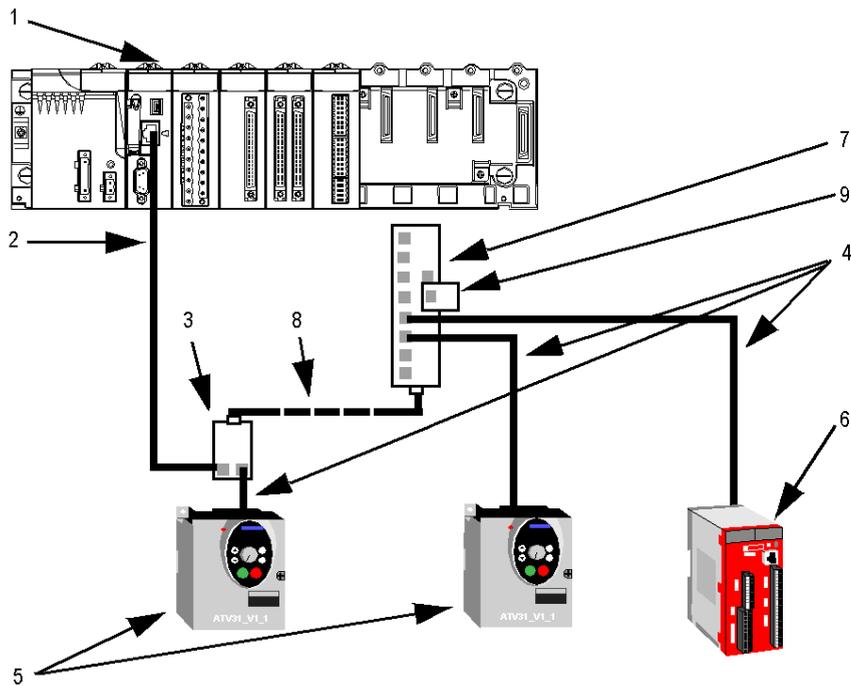
- Connection: 2 male RJ45 connectors
- Wiring: 2 wires for the RS485 physical line

Modbus Serial Link Architecture

The Modbus serial link architecture consists of the following elements:

- A BMX P34 2010/20102 processor configured as master,
- An XPSMC16 safety controller configured as slave,
- A TWDXCAISO isolated splitter block,
- An LU9 GC3 splitter block,
- Two ATV31 drives, configured as slaves.

The diagram below represents the serial link architecture described above:



- 1 BMX P34 2010 processor
- 2 XBT-Z9980 cable
- 3 TWDXCAISO isolated splitter block
- 4 VW3 A8 306 R30 cable
- 5 ATV31 drive
- 6 XPSMC16 safety controller
- 7 LU9 GC3 splitter block
- 8 TSXCSAx00 cable
- 9 VW3 A8 306 RC Modbus Line Terminator

Connecting Data Terminal Equipment (DTE) (RS232)

General

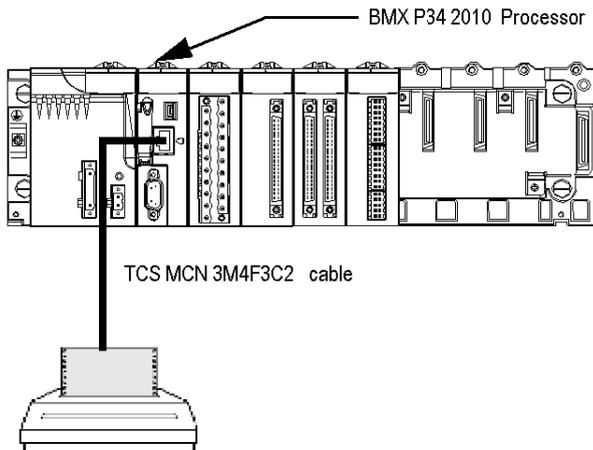
Data terminal equipment is the term used to describe devices such as:

- Common peripherals (printer, keyboard-screen, workshop terminal, etc.),
- Specialized peripherals (barcode readers, etc.),
- PCs.

All data terminal equipments are connected to a BMX P34 1000/2000/2010/20102/2020 processor by a serial cross cable using the RS232 physical link.

Connecting Data Terminal Equipment

The illustration below shows how a printer is connected to a BMX P34 2010 processor:



The communication protocol used is Character Mode.

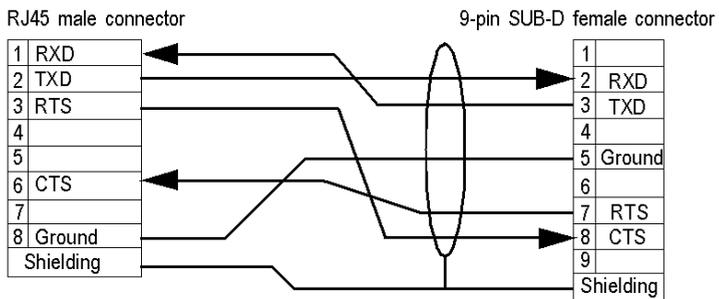
NOTE: Only one item of data terminal equipment may be connected to each BMX P34 1000/2000/2010/20102/2020 processor.

RS 232 Serial Cross Cable

The TCS MCN 3M4F3C2 serial cross cable has two connectors:

- RJ45 male
- Nine-pin SUB-D female

The illustration below shows the pin assignment for a TCS MCN 3M4F3C2 serial cross cable:



Connecting Cables and Accessories

The table below shows the product references of the cables and adapters to be used according to the serial connector used by the data terminal equipment:

Serial Connector for Data Terminal Equipment	Wiring
Nine-pin SUB-D male connector	TCS MCN 3M4F3C2 cable
25-pin SUB-D male connector	<ul style="list-style-type: none"> ● TCS MCN 3M4F3C2 cable ● TSX CTC 07 adapter
25-pin SUB-D female connector	<ul style="list-style-type: none"> ● TCS MCN 3M4F3C2 cable ● TSX CTC 10 adapter

Connecting Data Circuit-terminating Equipment (DCE) (RS232)

General

Data Circuit-terminating Equipment (DCE) is the term used to describe devices such as modems. For a DCE type device, the RTS and CTS pins are connected directly (not crossed).

All data circuit-terminating equipments are connected to a BMX P34 1000/2000/2010/20102/2020 processor by a serial direct cable using an RS232 physical link.

NOTE: The differences between DCE and DTE connections are largely in the plugs and the signal direction of the pins (input or output). For example, a desktop PC is termed as a DTE device while a modem is termed as a DCE device.

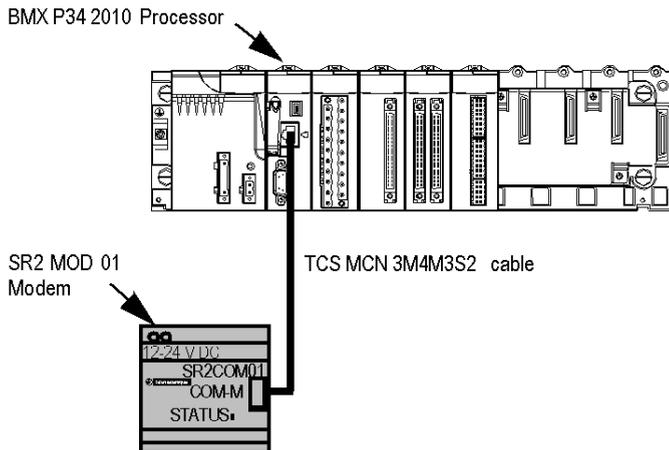
Modem Characteristics

M340 CPUs work with most modems on the market. To connect a modem to the serial port of a BMX P34 1000/2000/2010/20102/2020 processor, the modem must have the following characteristics:

- Support 10 or 11 bits per character if the terminal port is used in Modbus Serial:
 - 7 or 8 data bits
 - 1 or 2 stop bits
 - Odd, even or no parity
- Operate without a data carrier check.

Connecting Data Circuit-terminating Equipment

The illustration below shows how a modem is connected to a BMX P34 2010 processor:



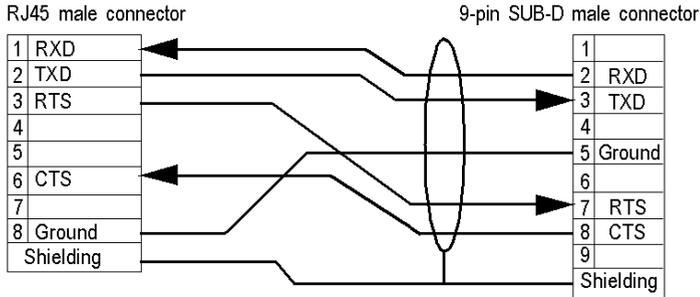
NOTE: In Modbus Serial, the waiting time must be between 100 and 250 ms.

RS 232 Serial Direct Cable

The TCS MCN 3M4M3S2 serial direct cable has two connectors:

- RJ45 male,
- Nine-pin SUB-D male.

The illustration below shows the pin assignment for a TCS MCN 3M4M3S2 serial direct cable:



Connecting Cables and Accessories

The table below shows the product references of the cables and adapters to be used according to the serial connector used by the Data Circuit-terminating Equipment:

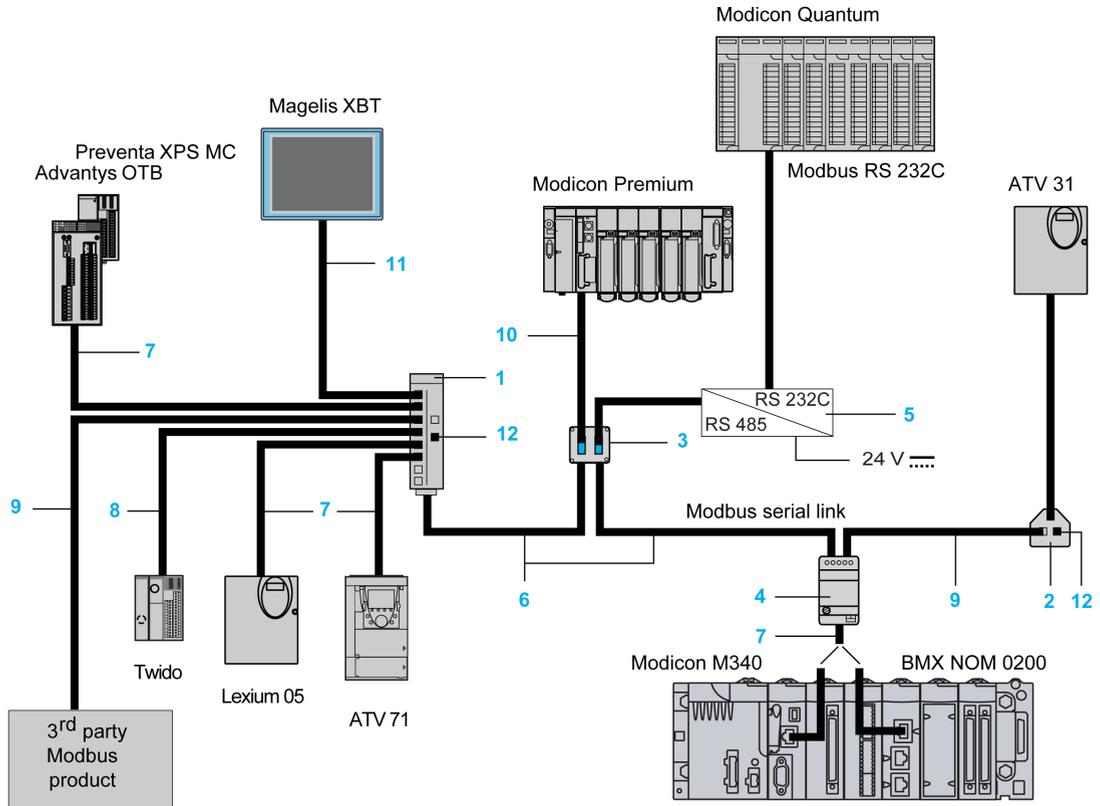
Serial Connector for Data Circuit-terminating Equipment	Wiring
Nine-pin SUB-D female connector	TCS MCN 3M4M3S2 cable
25-pin SUB-D female connector	<ul style="list-style-type: none"> ● TCS MCN 3M4M3S2 cable ● TSX CTC 09 adapter

Cabling

Cabling System

Several cables and accessories are required in order to set up a serial link.

The figure below shows an example of Modbus serial link and character mode cabling system. The **cables** (see page 38) and **connecting accessories** (see page 39) referenced in the figure are described in the next tables:



Cables

The table below shows the available cables that are compatible with serial communication on these processors and module:

Figure Reference	Designation	Characteristics	Length	Product reference
6	RS485 double shielded twisted pair trunk cable	Two bare ends	100 m	TSX CSA 100
			200 m	TSX CSA 200
			500 m	TSX CSA 500
7	Modbus RS485 cable	Two RJ45 male connectors	0.3 m	VW3 A8 306 R03
			1 m	VW3 A8 306 R10
			3 m	VW3 A8 306 R30
-	Modbus RS485 cable	<ul style="list-style-type: none"> One RJ45 male connector One fifteen-pin SUB-D male connector 	3 m	VW3 A8 306
8	Modbus RS485 cable	<ul style="list-style-type: none"> One RJ45 male connector One mini-DIN connector 	0.3 m	TWD XCA RJ003
			1 m	TWD XCA RJ010
			3 m	TWD XCA RJ030
9	Modbus RS485 cable	<ul style="list-style-type: none"> One RJ45 male connector One bare end 	3 m	VW3 A8 306 D30
10	Modbus RS485 cable	<ul style="list-style-type: none"> One miniature connector One 15-pin SUB-D connector 	3 m	TSX SCP CM 4630
11	RS485 cable for Magelis XBT display and terminal	<ul style="list-style-type: none"> One RJ45 male connector One 25-pin SUB-D female connector <p>Note: This cable is not compatible with BMX NOM 0200 module</p>	2.5 m	XBT-Z938
-	RS485 cable for devices that are powered via the serial link	Two RJ45 male connectors Note: This cable is not compatible with BMX NOM 0200 module.	3 m	XBT-Z9980
-	Four-wire RS232 cable for Data Terminal Equipment (DTE)	<ul style="list-style-type: none"> One RJ45 male connector One nine-pin SUB-D female connector 	3 m	TCS MCN 3M4F3C2
-	Four-wire RS232 cable for Data Circuit-terminating Equipment (DCE)	<ul style="list-style-type: none"> One RJ45 male connector One nine-pin SUB-D male connector 	3 m	TCS MCN 3M4M3S2
-	Seven-wire RS232 cable for Data Circuit-terminating Equipment (DCE)	<ul style="list-style-type: none"> One RJ45 male connector One 9-pin SUB-D male connector 	3 m	TCS XCN 3M4F3S4

Connecting Accessories

The table below shows the available connecting accessories that are compatible with serial communication on these processors and module:

Figure Reference	Designation	Characteristics	Product reference
1	Modbus splitter box	<ul style="list-style-type: none"> ● Ten RJ45 connectors ● One screw terminal block 	LU9 GC3
2	T-junction box	<ul style="list-style-type: none"> ● Two RJ45 connectors ● On-board 0.3 m cable with RJ45 connector at end 	VW3 A8 306 TF03
		<ul style="list-style-type: none"> ● Two RJ45 connectors ● On-board 1 m cable with RJ45 connector at end 	VW3 A8 306 TF10
-	Passive T-junction box	<ul style="list-style-type: none"> ● Three screw terminal blocks ● RC line end adapter 	TSX SCA 50
3	Passive 2-channel subscriber socket	<ul style="list-style-type: none"> ● Two fifteen-pin SUB-D female connectors ● Two screw terminal blocks ● RC line end adapter 	TSX SCA 62
4	Isolated RS485 T-junction box	<ul style="list-style-type: none"> ● One RJ45 connectors ● One screw terminal block 	TWD XCA ISO
-	T-junction box	Three RJ45 connectors	TWD XCA T3RJ
-	Modbus / Bluetooth adapter	<ul style="list-style-type: none"> ● One Bluetooth adapter with one RJ45 connector ● One cordset for PowerSuite with two RJ45 connectors ● One cordset for TwidoSuite with one RJ45 connector and one mini-DIN connector ● One RJ45/SUB-D male 9-pin adapter for ATV speed drives 	VW3 A8 114
5	RS232C/RS485 line adapter without modem signals	19.2 kbit/s	XGS Z24
12	Line terminator for RJ45 connector	<ul style="list-style-type: none"> ● Resistance of 120 Ω ● Capacity of 1 nF 	VW3 A8 306 RC
-	Line terminator for screw terminal block	<ul style="list-style-type: none"> ● Resistance of 120 Ω ● Capacity of 1 nF 	VW3 A8 306 DRC
-	Adapter for non-standard devices	<ul style="list-style-type: none"> ● Two 25-pin SUB-D male connectors 	XBT ZG999

Figure Reference	Designation	Characteristics	Product reference
-	Adapter for non-standard devices	<ul style="list-style-type: none">● One 25-pin SUB-D male connector● One 9-pin SUB-D male connector	XBT ZG909
-	Adapter for data terminal equipment	<ul style="list-style-type: none">● One 9-pin SUB-D male connector● One 25-pin SUB-D female connector	TSX CTC 07
-	Adapter for data terminal equipment	<ul style="list-style-type: none">● One 9-pin SUB-D male connector● One 25-pin SUB-D male connector	TSX CTC 10
-	Adapter for Data Circuit-terminating Equipment (DCE)	<ul style="list-style-type: none">● One 9-pin SUB-D female connector● One 25-pin SUB-D male connector	TSX CTC 09

NOTE: This list of cables and accessories is not exhaustive.

Part II

Software Implementation of Modbus Serial and Character Mode Communications

In This Part

This part provides an introduction to the software implementation of Modbus Serial and Character Mode communications using Control Expert software.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	Installation Methodology	43
4	Modbus Serial Communication for Modicon M340 Processors	45
5	Character Mode Communication for Modicon M340 Processors	69
6	Language Objects of Modbus and Character Mode Communications	87
7	Dynamic Protocol Switching	117

Chapter 3

Installation Methodology

Introduction to the Installation Phase

Introduction

The software installation of application-specific modules is carried out from the various Control Expert editors:

- in offline mode
- in online mode

If you do not have a processor to which you can connect, Control Expert allows you to carry out an initial test using a simulator. In this case, the installation is different.

Installation Phases When Using a Processor

The following table shows the various phases of installation using a processor:

Phase	Description	Mode
Configuration of the processor	Processor declaration	Offline
	Processor's serial port configuration	
Configuration of the module (if applicable)	Module declaration	Offline
	Module channel configuration	
	Entry of configuration parameters	
Declaration of variables	Declaration of the IODDT-type variables specific to the processor / module and the project variables	Offline ⁽¹⁾
Association	Association of IODDT variables with the configured channels (variable editor)	Offline ⁽¹⁾
Programming	Project programming	Offline ⁽¹⁾
Generation	Project generation (analysis and editing of links)	Offline
Transfer	Transferring project to PLC	Online
Debug	Project debugging from debug screens and animation tables	Online
Documentation	Creating a documentation file and printing the miscellaneous information relating to the project	Online
How it Works	Displaying of the miscellaneous information required to supervise the project	Online
(1) These phases may also be performed online.		

Installation Phases When Using a Simulator

The following table shows the various phases of installation using a simulator:

Phase	Description	Mode
Configuration of the processor	Processor declaration	Offline
	Processor's serial port configuration	
Configuration of the module (if applicable)	Module declaration	Offline
	Module channel configuration	
	Entry of configuration parameters	
Declaration of variables	Declaration of the IODDT-type variables specific to the processor / module and the project variables	Offline ⁽¹⁾
Association	Association of IODDT variables with the configured channels (variable editor)	Offline ⁽¹⁾
Programming	Project programming	Offline ⁽¹⁾
Generation	Project generation (analysis and editing of links)	Offline
Transfer	Transferring project to simulator	Online
Simulation	Program simulation without inputs/outputs	Online
Adjustment/Debugging	Project debugging from animation tables	Online
	Modifying the program and adjustment parameters	
(1) These phases may also be performed online.		

Configuration of Processor and Module

The configuration parameters may only be accessed from the Control Expert software.

Technical Documentation Creation

Control Expert allows to create a **project technical documentation** (*see EcoStruxure™ Control Expert, Operating Modes*).

The general format of the printout is made of:

- A title: module part number and its position,
- A section with the module identification,
- A section per channel with all parameters of a channel.

The printout is consistent with the configuration: not significant grayed information is not printed.

Chapter 4

Modbus Serial Communication for Modicon M340 Processors

Subject of this Chapter

This chapter presents the software implementation process for Modbus Serial communication for Modicon M340 processors.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
4.1	Generalities	46
4.2	Modbus Serial Communication Configuration	53
4.3	Modbus Serial Communication Programming	62
4.4	Debugging Modbus Serial Communication	67

Section 4.1

Generalities

Subject of this Section

This section presents the general points relating to Modbus Serial communication and its services.

What Is in This Section?

This section contains the following topics:

Topic	Page
About Modbus Serial	47
Performance	48
How to Access the Serial Link Parameters	50

About Modbus Serial

Introduction

Communicating via Modbus enables data exchange between all devices connected to the bus. The Modbus Serial is a protocol that creates a hierarchical structure (one master and several slaves).

The master manages all exchanges in two ways:

- The master exchanges with the slave and awaits a response.
- The master exchanges with all the slaves without waiting for a response (general broadcast).

NOTE: Be careful that two masters (on the same bus) do not send requests simultaneously otherwise the requests are lost and each report will have a bad result which could be 16#0100 (request could not be processed) or 16#ODFF (slave is not present).

 WARNING
CRITICAL DATA LOSS Only use communication ports for non-critical data transfers. Failure to follow these instructions can result in death, serious injury, or equipment damage.

Performance

At a Glance

The tables that follow can be used to evaluate typical Modbus communication exchange times according to different criteria.

The results displayed correspond to the average operation period for the `READ_VAR` function in milliseconds.

Exchange Time Definition

The Exchange Time is the time that passes between the creation of an exchange and the end of that exchange. It includes the serial link communication time.

The exchange is created when the communication function call is made.

The exchange ends when one of the following events occurs:

- Data is received.
- An anomaly occurs.
- Time-out expires.

Exchange Time for One Word

The table below shows exchange times for one word of Modbus communication on a BMX P34 2020 processor:

Exchange time in ms (the Modbus slave is a BMX P34 1000 cyclic)		Cycle time in ms		
		Cyclic	10	50
Baud rate of communication in bits per second	4800	68	72	100
	9600	35	40	50
	19200	20	27	50
	38400	13	20	50

The BMX P34 2000/2010/20102 processor exchange times are similar to the BMX P34 2020 processor. The BMX P34 1000 exchange times are 10% lower than ones.

NOTE: All exchange times listed above come from measures with an accuracy margin of +/-10 ms.

Exchange Time for 100 Words

The table below shows exchange times for 100 words of Modbus communication on a BMX P34 2020 processor:

Exchange time in ms (the Modbus slave is a BMX P34 1000 cyclic)		Cycle time in ms		
		Cyclic	10	50
Baud rate of communication in bits per second	4800	500	540	595
	9600	280	288	300
	19200	142	149	150
	38400	76	80	100

The BMX P34 2000/2010/20102 processor exchange times are similar to the BMX P34 2020 processor. The BMX P34 1000 exchange times are 10% lower than ones.

NOTE: All exchange times listed above come from measures with an accuracy margin of +/-10 ms.

How to Access the Serial Link Parameters

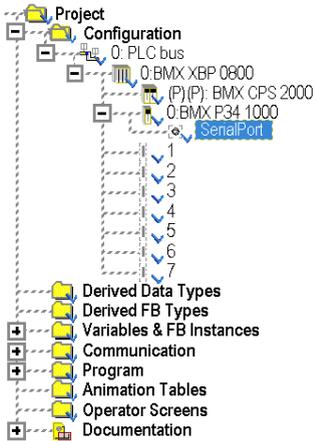
At a Glance

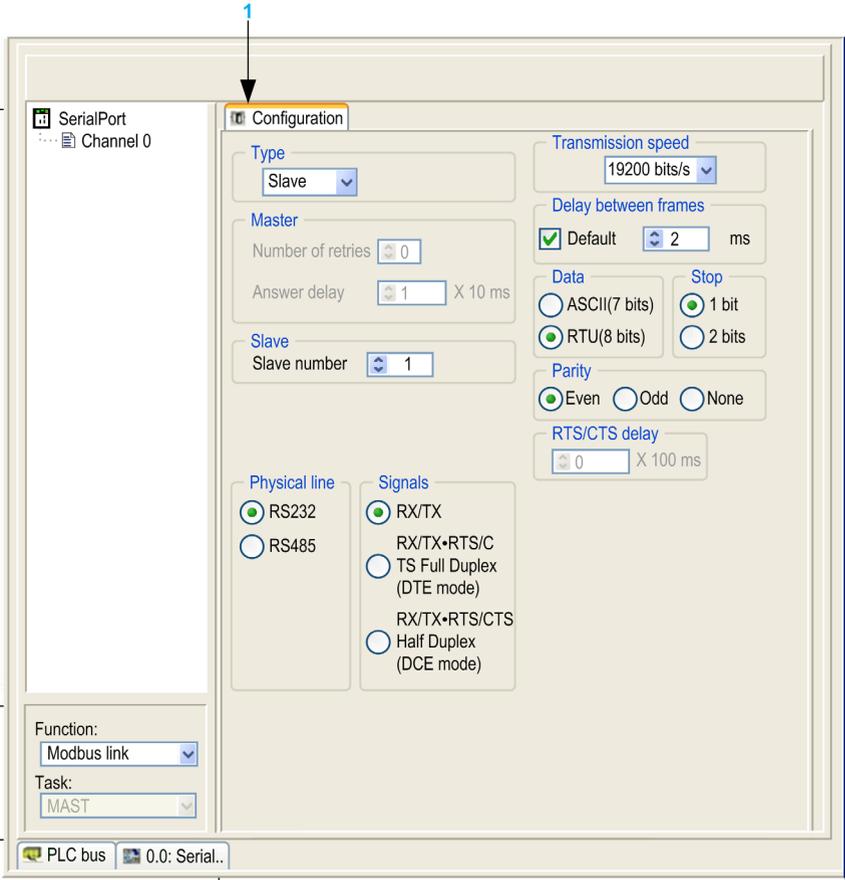
The following pages explain how to access the serial port configuration screen for the following processors as well as the general elements of Modbus and Character Mode link configuration and debug screens:

- BMX P34 1000,
- BMX P34 2000,
- BMX P34 2010/20102,
- BMX P34 2020.

How to Access the Serial Link

The table below describes the procedure for accessing the serial link:

Step	Action
1	<p>In the project browser, open the following directory: <i>Project\Configuration\0: PLC bus\0: BMX XBP ****\0: BMX P34 ****\SerialPort.</i></p> <p>Result: The following screen appears:</p> 

Step	Action
2	<p>Double-click on the SerialPort sub-directory. Result: The configuration screen appears:</p> 

Description of Configuration Screen

The following table shows the different elements of the configuration screen:

Address	Element	Function
1	Tabs	<p>The tab in the foreground indicates the current mode. Each mode can be selected using the corresponding tab. The available modes are:</p> <ul style="list-style-type: none"> ● Configuration ● Debug screen (accessible in online mode only)
2	Channel zone	<p>Enables you to:</p> <ul style="list-style-type: none"> ● Choose between the serial port and channel 0 by clicking on one or the other. ● Display the following tabs by clicking on the serial port: <ul style="list-style-type: none"> ○ "Description", which gives the characteristics of the device. ○ "I/O Objects", (<i>see EcoStruxure™ Control Expert, Operating Modes</i>) which is used to presymbolize the input/output objects. ● Display the following tabs by clicking on the channel: <ul style="list-style-type: none"> ○ Configuration ○ Debugging ● Display the channel name and symbol defined by the user using the variables editor.
3	General parameters zone	<p>This enables you to choose the general parameters associated with the channel:</p> <ul style="list-style-type: none"> ● Function: The available functions are Modbus and Character Mode. The default configuration is with the Modbus function. ● Task: Defines the master task in which the implicit exchange objects of the channel will be exchanged. This zone is grayed out and therefore not configurable.
4	Configuration or debugging zone	<p>In configuration mode, this zone is used to configure the channel parameters. In debug mode, it is used to debug the communication channel.</p>

Section 4.2

Modbus Serial Communication Configuration

Subject of this Section

This section describes the software configuration process for Modbus Serial communication.

What Is in This Section?

This section contains the following topics:

Topic	Page
Modbus Serial Communication Configuration Screen	54
Application-linked Modbus Parameters	56
Signal and Physical Line Parameters in Modbus	58
Transmission-linked Modbus Parameters	60

Modbus Serial Communication Configuration Screen

General

The following pages provide an introduction to the configuration screen for Modbus serial communication.

Access to the Configuration Screen

To access the Modbus serial communication configuration screen, open the Serial Port directory in the project browser (*see page 50*).

Modbus Serial Communication Configuration Screen

The following figure shows the configuration screen for Modbus serial communication:

The configuration screen is titled "Configuration" and is divided into several sections:

- Type:** A dropdown menu set to "Slave".
- Mastrer:**
 - Number of retries: 0
 - Answer delay: 1 X 10 ms
- Slave:**
 - Slave number: 1
- Physical line:**
 - RS232
 - RS485
- Signals:**
 - RX/TX
 - RX/TX + RTS/CTS DTE mode
 - RX/TX + RTS/CTS DCE mode
- Transmission speed:** 19200 bits/s
- Delay between frames:**
 - Default (2 ms)
- Data:**
 - ASCII(7 bits)
 - RTU(8 bits)
- Stop:**
 - 1 bit
 - 2 bits
- Parity:**
 - Even
 - Odd
 - None
- RTS/CTS delay:** 0 X 100 ms

Description

These zones are used to configure channel parameters. In online mode, these zones are accessible. In offline mode, the zone is accessible but some parameters may not be accessible and are grayed out.

The following table shows the different zones of the Modbus link configuration screen:

Element	Comment
Application parameters (<i>see page 56</i>)	These parameters are accessible via three zones: <ul style="list-style-type: none"> ● Type, ● Master, ● Slave.
Signal and physical line parameters (<i>see page 58</i>)	These parameters are accessible via three zones: <ul style="list-style-type: none"> ● Physical line, ● Signals, ● RTS/CTS delay.
Transmission parameters (<i>see page 60</i>)	These parameters are accessible via five zones: <ul style="list-style-type: none"> ● Transmission speed, ● Delay between frames, ● Data, ● Stop bits, ● Parity.

NOTE: When configuring Modbus Serial communication in master mode, the **Slave** zone is grayed out and cannot be modified and conversely.

Default Values

The following table shows the default values for Modbus Serial communication parameters:

Configuration parameters	Value	
Application parameters	Type	Slave
	Slave number	1
Signal and physical line parameters	Physical line	RS485
	Signals	RX/TX
Transmission parameters	Transmission speed	19200 bits/s
	Delay between frames	2 ms
	Data	RTU (8 bits)
	Stop	1 bit
	Parity	Even

Application-linked Modbus Parameters

At a Glance

After configuring the communication channel, you need to enter the application parameters.

These parameters are accessible from three configuration zones:

- The Type zone,
- The Master zone,
- The Slave zone.

The Type Zone

This configuration zone appears on the screen as shown below:



This zone enables you to select the type of Modbus Serial to be used:

- **Master:** When the station concerned is the master.
- **Slave:** When the station concerned is a slave.

The Master Zone

The configuration zone shown below is only accessible when "Master" is selected in the "Type" zone:



This zone enables you to enter the following parameters:

- **Number of retries:** number of connection attempts made by the master before defining the slave as absent.
The default value is 3.
Possible values range from 0 to 15.
A value of 0 indicates no retries by the Master.
- **Answer delay:** the time between the Master's initial request and a repeated attempt if the slave does not respond. This is the maximum time between the transmission of the last character of the Master's request and receipt of the first character of the request sent back by the slave.
The default value is 1 second (100*10 ms).
Possible values range from 10 ms to 10 s.

NOTE: The Answer delay of the Master must be at least equal to the longest Answer delay of the Slaves present on the bus.

The Slave Zone

The configuration zone shown below is only accessible when "Slave" is selected in the "Type" zone:



The image shows a configuration window with a dropdown menu set to 'Slave' and a text input field labeled 'Slave number' containing the value '7'.

This zone enables you to enter the processor's slave number.

The default value is 1.

Possible values range from 1 to 247.

NOTE: In a Modbus Slave configuration, an additional address, number 248, can be used for a point-to-point serial communication.

Signal and Physical Line Parameters in Modbus

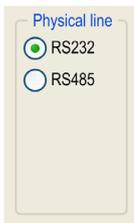
At a Glance

The signal and physical line parameters are accessible via three zones:

- The **Physical line** zone,
- The **Signals** zone,
- The **RTS/CTS delay** zone.

The Physical line Zone

This configuration zone appears on the screen as shown below:

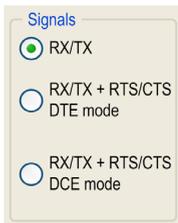


In this zone, you can choose between two types of physical line for the serial port on the BMX P34 1000/2000/2010/20102/2020 processors:

- The RS232 line,
- The RS485 line.

The Signals Zone

This configuration zone appears on the screen as shown below:



In this zone, you can select the signals supported by the RS232 physical line:

- **RX/TX**
- **RX/TX + RTS/CTS DTE mode**
- **RX/TX + RTS/CTS DCE mode**

If the RS485 is configured, the entire zone will be grayed out and the default value is **RX/TX**.

The RTS/CTS delay Zone

This configuration zone appears on the screen as shown below:



RTS/CTS delay zone is available only when both RS232 and RX/TX+RTS/CTS check boxes are selected. An RTS/CTS flow control algorithm is selected if the default value is 0 ms. A value different from 0 enables an RTS/CTS modem control algorithm.

The RTS/CTS flow control algorithm (DTE <-> DTE) is different from the RTS/CTS modem control algorithm (DTE <-> DCE) as follows:

- The RTS/CTS flow control algorithm is related to the overflow reception buffer (full duplex).
- The RTS/CTS modem control algorithm deals with the shared transmission process, e.g. a radio modem.

RTS/CTS Flow Control Algorithm

The aim is to prevent a reception buffer overflow.

The RTS output signal of each device is connected to CTS input signal of other device. The transmitter (M340) is authorized to transmit data when receiving the RTS input signal (e.g. another M340) on its CTS input. This algorithm is symmetric and allows full duplex asynchronous communication.

RTS/CTS Modem Control Algorithm

Before a request is transmitted, the sender (M340) activates the RTS signal and waits for the CTS signal to be triggered by the modem. If the CTS is not activated after the RTS/CTS delay, the request is discarded.

Transmission-linked Modbus Parameters

At a Glance

The transmission parameters are accessible from five zones:

- The **Transmission speed** zone,
- The **Delay between frames** zone,
- The **Data** zone,
- The **Stop** zone,
- The **Parity** zone.

The Transmission speed Zone

This configuration zone appears on the screen as shown below:



You can use it to select the transmission speed of the Modbus Serial. The selected speed has to be consistent with the other devices. The configurable values are 300, 600, 1200, 2400, 4800, 9600, 19200 and 38400 bits per second.

The Delay between frames Zone

This configuration zone appears on the screen as shown below:



The **Delay between frames** is the minimum time separating two frames on reception. This delay is managed when the PLC (master or slave) is receiving messages.

NOTE: The default value depends on the selected transmission speed.

NOTE: The delay between frames should be the Default value in order to be Modbus compliant. In case a Slave is not conform, the value can be changed and should be identical for the Master and all Slaves on the Bus.

The Data Zone

This configuration zone appears on the screen as shown below:



This zone allows you to enter the type of coding used to communicate using Modbus Serial. This field is set according to the other devices connected on the bus. There are two configurable modes:

- RTU mode:
 - The characters are coded over 8 bits.
 - The end of the frame is detected when there is a silence of at least 3.5 characters.
 - The integrity of the frame is checked using a word known as the CRC checksum, which is contained within the frame.
- ASCII mode:
 - The characters are coded over 7 bits.
 - The beginning of the frame is detected when the ":" character is received.
 - The end of the frame is detected by a carriage return and a line feed.
 - The integrity of the frame is checked using a byte called the LRC checksum, which is contained within the frame.

The Stop Zone

This configuration zone appears on the screen as shown below:



The Stop zone allows you to enter the number of stop bits used for communication. This field is set according to the other devices. The configurable values are:

- 1 bit
- 2 bits

The Parity Zone

This configuration zone appears on the screen as shown below:



This zones enables you to determine whether a parity bit is added or not, as well as its type. This field is set according to the other devices. The configurable values are:

- **Even**
- **Odd**
- **None**

Section 4.3

Modbus Serial Communication Programming

Subject of this Section

This section describes the programming process involved in implementing Modbus serial communication.

What Is in This Section?

This section contains the following topics:

Topic	Page
Services Supported by a Modbus Link Master Processor	63
Services Supported by a Modbus Link Slave Processor	65

Services Supported by a Modbus Link Master Processor

Communication Functions

Three specific communication functions are defined for sending and receiving data via a Modbus communication channel:

- `READ_VAR`: To read variables
- `WRITE_VAR`: To write variables
- `DATA_EXCH`: To send Modbus requests to another device over the selected protocol

For detailed information on these communication functions, refer to the chapter *General Information on M340 Communication Functions (see EcoStruxure™ Control Expert, Communication, Block Library)*.

Data Exchanges

Reading or writing of variables are carried out by addressing following requests to the targeted slave device.

These requests use the `READ_VAR`, `WRITE_VAR`, and `DATA_EXCH` communication functions:

Modbus request	Function code	Communication function
Read bits	16#01 or 16#02	<code>READ_VAR</code>
Read words	16#03 or 16#04	<code>READ_VAR</code>
Write bits	16#0F	<code>WRITE_VAR</code>
Write words	16#10	<code>WRITE_VAR</code>
Other request	all	<code>DATA_EXCH</code>

NOTE: `WRITE_VAR` can be used in broadcast mode (`READ_VAR` can't be used in broadcast mode). In this case, the PLC doesn't receive a response. Sending a broadcast request resets the activity bit and the code 16#01 (Exchange stop on timeout) is returned into the EF second management word.

NOTE: The objects read by Modicon M340 PLC can be of the type `%I` and `%IW`. In this case, `READ_VAR` function generates a Modbus request: FC 0x2 or 0x4. In a Quantum PLC, it allows accessing the Input Status or Input Status Registers. More generally, it is possible to send any Modbus requests to a slave device by using the `DATA_EXCH` communication function.

Cancelling an Exchange

There are two ways of programming that enable an exchange executed by the communication functions to be cancelled:

- Using the `CANCEL` function
- Using the communication function cancel bit

For detailed information on cancelling a communication function, refer to *EcoStruxure™ Control Expert, Communication, Block Library*.

Services Supported by a Modbus Link Slave Processor

At a Glance

When used as a slave processor in a Modbus link, the following processors support several services:

- BMX P34 1000,
- BMX P34 2000,
- BMX P34 2010/20102,
- BMX P34 2020.

Data Exchanges

A slave processor manages the following requests:

Modbus request	Function code	PLC object
Read n output bits	16#01	%M
Read n input bits	16#02	%M
Read n output words	16#03	%MW
Read n input words	16#04	%MW
Write an output bit	16#05	%M
Write an output word	16#06	%MW
Write n output bits	16#0F	%M
Write n output words	16#10	%MW

Diagnostics and Maintenance

The diagnostics and maintenance information accessible from a Modbus link is listed below:

Designation	Function code/sub-function code
Echo	16#08 / 16#00
Read the PLC diagnostic registers	16#08 / 16#02
Reset PLC diagnostic registers and counters to 0	16#08 / 16#0A
Read number of messages on the bus	16#08 / 16#0B
Read number of detected communication errors on the bus	16#08 / 16#0C
Read number of detected exception errors on the bus	16#08 / 16#0D
Read number of messages received from the slave	16#08 / 16#0E
Read number of "no responses" from the slave	16#08 / 16#0F
Read number of negative acknowledgements from the slave	16#08 / 16#10
Read number of exception responses from the slave	16#08 / 16#11
Read number of overflowing characters on the bus	16#08 / 16#12

Designation	Function code/sub-function code
Read event counter	16#0B
Read connection event	16#0C
Read identification	16#11
Read device identification	16#2B / 16#0E

Section 4.4

Debugging Modbus Serial Communication

Modbus Serial Communication Debug Screen

General

The Modbus serial communication debug screen can only be accessed in online mode.

Accessing the Debug Screen

The following table describes the procedure for accessing the debug screen for Modbus serial communication:

Step	Action
1	Access the configuration screen for Modbus serial communication. (<i>see page 54</i>)
2	Select the "Debug" tab on the screen that appears.

Description of the Debug Screen

The debug screen is divided into two zones:

- The Type zone,
- The Counters zone.

The Type Zone

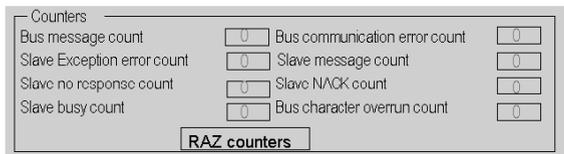
This zone looks like this:



It indicates the type of Modbus function configured (in this case, Master).

The Counters Zone

This zone looks like this:



This zone shows the various debugging counters.

The Reset Counters button resets all the debug mode counters to zero.

Counter Operation

The Modbus serial communication debugging counters are:

- Bus message counter: This counter indicates the number of messages that the processor has detected on the serial link. Messages with a negative CRC check result are not counted.
- Bus communication error counter: This counter indicates the number of negative CRC check results counted by the processor. If a character error (overflow, parity error) is detected, or if the message is less than 3 bytes long, the system that receives the data cannot perform the CRC check. In such cases, the counter is incremented accordingly.
- Slave exception error counter: This counter indicates the number of Modbus exception errors detected by the processor.
- Slave message counter: This counter indicates the number of messages received and processed by the Modbus link.
- Slave "no response" counter: This counter indicates the number of messages sent by the remote system for which it has received no response (neither a normal response, nor an exception response). It also counts the number of messages received in broadcast mode.
- Negative slave acknowledgement counter: This counter indicates the number of messages sent to the remote system for which it has returned a negative acknowledgement.
- Slave busy counter: This counter indicates the number of messages sent to the remote system for which it has returned a "slave busy" exception message.
- Bus character overflow counter: This counter indicates the number of messages sent to the processor that it is unable to acquire because of character overflow on the bus. Overflow is caused by:
 - Character-type data that are transmitted on the serial port more quickly than they can be stored,
 - A loss of data due to a hardware anomaly.

NOTE: For all counters, the count begins at the most recent restart, clear counters operation or processor power-up.

Chapter 5

Character Mode Communication for Modicon M340 Processors

Subject of this Section

This chapter presents the software implementation of communication using Character Mode for Modicon M340 processors.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
5.1	Generalities	70
5.2	Character Mode Communication Configuration	73
5.3	Character Mode Communication Programming	82
5.4	Debugging Character Mode communication	84

Section 5.1

Generalities

Subject of this Section

This section provides an overview of the general points relating to Character Mode communication and its services.

What Is in This Section?

This section contains the following topics:

Topic	Page
About Character Mode Communication	71
Performance	72

About Character Mode Communication

Introduction

Communication in Character Mode enables dialog and communication functions to be carried out with the following devices:

- Regular peripherals (printer, keyboard-screen, workshop terminal, etc.),
- Specialized peripherals (barcode readers, etc.),
- Calculators (checking, production management, etc.),
- Heterogeneous devices (numerical commands, variable speed controllers, etc),
- External modem.

 WARNING
CRITICAL DATA LOSS Only use communication ports for non-critical data transfers. Failure to follow these instructions can result in death, serious injury, or equipment damage.

Performance

At a Glance

The following tables describe typical exchange times in the Character Mode.

The results displayed correspond to the average operation period for the `PRINT_CHAR` function in milliseconds.

Exchange Time Definition

The Exchange Time is the time between the creation of an exchange and the end of that exchange. It includes the serial link communication time.

The exchange is created when the communication function call is made.

The exchange ends when one of the following events occurs:

- Reception of data
- An anomaly
- Time-out expires

Exchange Time Values

The following table gives the exchange times for the transmission of 80 characters in Character Mode on a BMX P34 2020 processor at different baud rates and cycle times:

Exchange time in ms		Cycle time in ms				
		10	20	50	100	255
Baud rate of communication in bits per second	1200	805	820	850	900	980
	4800	210	220	250	300	425
	9600	110	115	145	200	305
	19200	55	60	95	100	250

The BMX P34 2000/2010/20102 processor exchange times are similar to the BMX P34 2020 processor. The BMX P34 1000 exchange times are 10% lower.

NOTE: All exchange times listed above come from measures with an accuracy margin of +/- 10 ms.

Section 5.2

Character Mode Communication Configuration

Subject of this Section

This section describes the configuration process used when implementing Character Mode communication.

What Is in This Section?

This section contains the following topics:

Topic	Page
Character Mode Communication Configuration Screen	74
Message End Detection Parameters in Character Mode	76
Signal and Physical Line Parameters in Character Mode	78
Transmission Parameters in Character Mode	80

Character Mode Communication Configuration Screen

General

The following pages provide an introduction to the configuration screen for Character Mode communication.

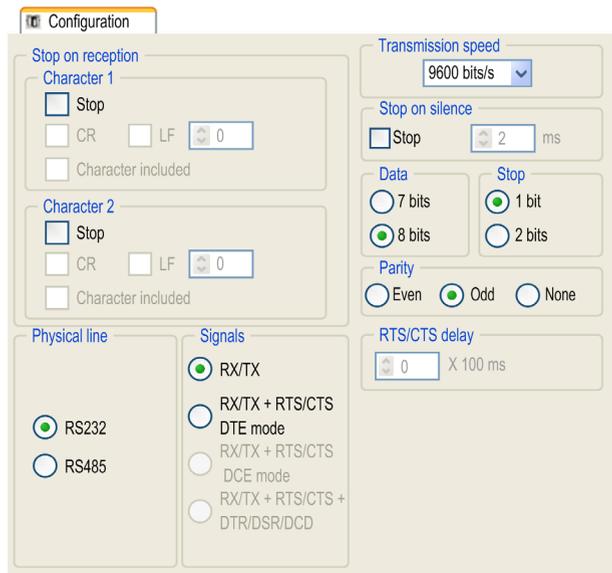
Accessing the Configuration Screen

The following table describes the procedure for accessing the configuration screen for Character Mode communication:

Step	Action
1	Open the Serial Port sub-directory in the project browser (<i>see page 50</i>).
2	On the screen that appears, select Character mode link in the Function box.

Character Mode Communication Configuration Screen

The figure below shows the default configuration screen for Character Mode communication:



Description

These zones are used to configure channel parameters. In the online mode, these zones are accessible. In the offline mode, these zone are accessible but some parameters may not be accessible and are grayed out.

The following table shows the different zones of the Character Mode communication configuration screen:

Element	Comment
Message end detection parameters (<i>see page 76</i>)	These parameters are accessible via two zones: <ul style="list-style-type: none"> ● Stop on reception, ● Stop on silence.
Signal and physical line parameters (<i>see page 78</i>)	These parameters are accessible via three zones: <ul style="list-style-type: none"> ● Physical line, ● Signals, ● RTS/CTS delay.
Transmission parameters (<i>see page 80</i>)	These parameters are accessible via four zones: <ul style="list-style-type: none"> ● Transmission speed, ● Data, ● Stop bits, ● Parity.

Default Values

The following table shows the default values for Character Mode communication parameters:

Configuration parameters		Value
Message end detection parameters	Stop on reception	None
	Stop on silence	None
Signal and physical line parameters	Physical line	RS232
	Signals	RX/TX
Transmission parameters	Transmission speed	9600 bits/s
	Data	8 bits
	Stop	1 bit
	Parity	Odd

Message End Detection Parameters in Character Mode

At a Glance

The message end detection parameters are accessible via two zones:

- The **Stop on reception** Zone: stop on reception of a special character.
- The **Stop on silence** Zone: stop on silence.

Conditions of Use

Selecting **Stop on silence** means that **Stop on reception** is deselected and vice versa.

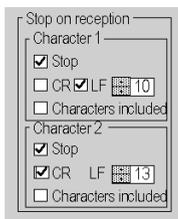
NOTE:

To configure a channel in character mode without stop parameters, uncheck **Stop** check boxes for the following configuration zones:

- **Stop on reception** → **Character 1**
- **Stop on reception** → **Character 2**
- **Stop on silence**

The Stop on reception Zone

This configuration zone appears on the screen as shown below:



A reception request can be terminated once a specific character is received.

By checking the **Stop** option, it is possible to configure **Stop on reception** to be activated by a specific end-of-message character:

- **CR**: enables you to detect the end of the message by a carriage return.
- **LF**: enables you to detect the end of the message by a line feed.
- **Data entry field**: enables you to identify an end-of-message character other than the carriage return or line feed characters, using a decimal value:
 - Between 0 and 255 if the data is coded over 8 bits
 - Between 0 and 127 if the data is coded over 7 bits
- **Character included**: enables you to include the end-of-message character in the reception table of the PLC application.

It is possible to configure two end-of-reception characters. In the above window, the end of reception of a message is detected by a line feed or carriage return character.

The Stop on silence Zone

This configuration zone appears on the screen as shown below:



This zone enables you to detect the end of a message on reception by the absence of message end characters over a given time.

Stop on silence is validated by checking the **Stop** box. The duration of the silence (expressed in milliseconds) is set using the data entry field.

The minimal value of this duration is the time corresponding to the transmission of 1.5 characters. Expressed in number of bits, and depending on the configuration of start and stop bits, the minimal silence duration is as follows:

Total character length (bit)	Minimal silence duration (bit)
8	12
9	12
10	15
11	15

Convert the number in right column in time according to the configured speed transmission.

NOTE: The available values range from 1 ms to 10000 ms and depend on the transmission speed selected.

Signal and Physical Line Parameters in Character Mode

At a Glance

The signal and physical line parameters are accessible via three zones:

- The **Physical line** zone
- The **Signals** zone
- The **RTS/CTS delay** zone

The Physical line Zone

This configuration zone appears on the screen as shown below:

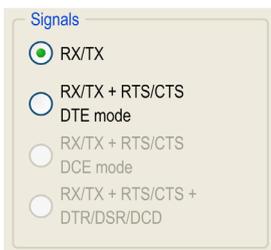


In this zone, you can choose between two types of physical line for the serial port on the BMX P34 1000/2000/2010/20102/2020 processors:

- The RS232 line
- The RS485 line

The Signals Zone

This configuration zone appears on the screen as shown below:



In this zone, you can select the signals supported by the RS232 physical line:

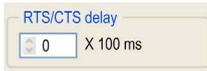
- **RX/TX**
- **RX/TX + RTS/CTS DTE mode**

If the RS485 is configured, the entire zone is grayed out and the default value is **RX/TX**.

NOTE: Only **RX/TX** and **RX/TX + RTS/CTS DTE mode** signals are available when configuring the serial port in Character mode.

The RTS/CTS delay Zone

This configuration zone appears on the screen as shown below:



RTS/CTS delay zone is available only when both RS232 and **RX/TX+RTS/CTS DTE mode** check boxes are selected.

An RTS/CTS flow control algorithm is selected: before a character string is transmitted, the system waits for the CTS (Clear To Send) signal to be activated. This zone enables you to enter the maximum waiting time between the two signals. When this value is timed out, the request is not transmitted on the bus. Configurable values range from 0 s to 10 s.

NOTE: The default value is 0 ms.

NOTE: A value of 0 s indicates that the delay between the two signals has not been managed.

RTS/CTS Flow Control Algorithm

The aim is to prevent a reception buffer overflow.

The RTS output signal of each device is connected to CTS input signal of the other device. The transmitter (M340) is authorized to transmit data when receiving the RTS input signal (e.g. another M340) on its CTS input. This algorithm is symmetric and allows full duplex asynchronous communication.

Transmission Parameters in Character Mode

At a Glance

The transmission parameters are accessible via four zones:

- The **Transmission speed** zone,
- The **Data** zone,
- The **Stop** zone,
- The **Parity** zone.

The Transmission speed Zone

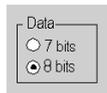
This configuration zone appears on the screen as shown below:



You can use this zone to select the transmission speed of the Character Mode protocol. The selected speed has to be consistent with the other devices. The configurable values are 300, 600, 1200, 2400, 4800, 9600, 19200, and 38400 bits per second.

The Data Zone

This configuration zone appears on the screen as shown below:



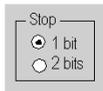
In this zone, you can specify the size of the data being exchanged on the link. The available values are:

- 7 bits
- 8 bits

You are advised to adjust the number of data bits according to the remote device being used.

The Stop Zone

This zone looks like this:



The **Stop** zone allows you to enter the number of stop bits used for communication. You are advised to adjust the number of stop bits according to the remote device being used.

The configurable values are:

- 1 bit
- 2 bits

The Parity Zone

This configuration zone appears on the screen as shown below:



This zone enables you to determine whether a parity bit is added or not, as well as its type. You are advised to adjust parity according to the remote device being used. The configurable values are:

- **Even**
- **Odd**
- **None**

Section 5.3

Character Mode Communication Programming

Character Mode Communication Functions

Available Functions

Three specific communication functions are defined for sending and receiving data via a communication channel in Character Mode:

- `PRINT_CHAR`: send a character string of a maximum of 1,024 bytes.
- `INPUT_CHAR`: read a character string of a maximum of 1,024 bytes.
- `INPUT_BYTE`: read a byte array of a maximum of 1,024 bytes.

For detailed information on communication functions, refer to chapter *General Information on M340 Communication Functions (see EcoStruxure™ Control Expert, Communication, Block Library)*.

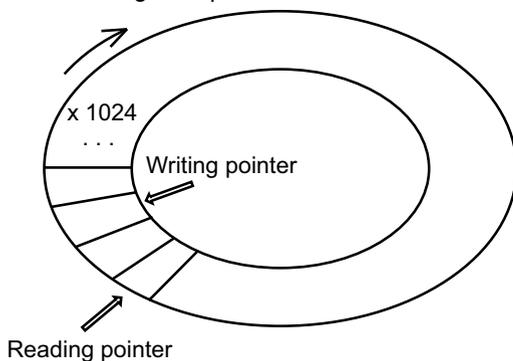
NOTE: For `INPUT_CHAR` function, a configured timeout is necessary if the channel is configured without stop on silence, to acknowledge the activity bit of the function. For `PRINT_CHAR` function, it is advisable but not necessary to configure a timeout.

Internal Mechanism of the CPU

The Modicon M340 PLC serial port is full duplex, so a `PRINT_CHAR` function can be sent even when an `INPUT_CHAR` or `INPUT_BYTE` function has been sent and is still pending.

Two independent pointers allows access for reading and writing the data.

The below figure represents this mechanism:



The received data are stored in a 1024 bits cyclic buffer. Once the buffer has been fully filled, the 1025th bit received overwrites the 1st bit and so on. Each buffer bit read through the `INPUT_CHAR` function is reset.

The CPU save the ECHO of the transmitted data into the same buffer as the received data. Therefore it is mandatory to clear the buffer of the CPU after each `PRINT_CHAR` or before someone send data to the channel. Else the received data from an `INPUT_CHAR` or `INPUT_BYTE` will not be the expected one.

To clear the CPU buffer you can set to 1 the input parameter `RAZ` of the reading function and cancel this reading function before the timeout. The buffer is first reset then the processor is waiting for the reception of data.

NOTE: Using this feature is advised in order to start properly a reception by removing old data that can remain in the buffer.

Cancelling an Exchange

There are two ways of programming that enable an exchange executed by the communication functions to be cancelled:

- Using the `CANCEL` function
- Using the communication function cancel bit

For detailed information on cancelling a communication function, refer to *EcoStruxure™ Control Expert, Communication, Block Library*.

Section 5.4

Debugging Character Mode communication

Character Mode Communication Debug Screen

General

The Character Mode debug screen is accessible in online mode.

Accessing the Debug Screen

The following table describes the procedure for accessing the debug screen for Character Mode communication:

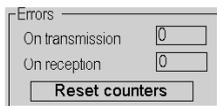
Step	Action
1	Access the configuration screen for Character Mode communication. <i>(see page 74)</i>
2	Select the "Debug" tab on the screen that appears.

Description of the Debug Screen

The debug screen consists of an Error zone and a Signals zone.

The Error Zone

The Error zone looks like this:



This zone indicates the number of communication interruptions counted by the processor:

- **On transmission:** corresponds to the number of interruptions on transmission (image of %MW4 word).
- **On reception:** corresponds to the number of interruptions on reception (image of %MW5 word).

The Reset Counters button resets both counters to zero.

The Signals Zone

The Signals zone looks like this:



This zone indicates the activity of the signals:

- **CTS RS232**: shows the activity of the CTS signal.
- **DCD RS232**: not managed by the processor (no activity on this LED).
- **DSR RS232**: not managed by the processor (no activity on this LED).

Chapter 6

Language Objects of Modbus and Character Mode Communications

Subject of this Chapter

This chapter describes the language objects associated with Modbus and Character Mode communications and the different ways of using them.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Language Objects and IODDTs of Modbus and Character Mode Communications	88
6.2	General Language Objects and IODDTs for Communication Protocols	96
6.3	Language Objects and IODDTs Associated with Modbus Communication	100
6.4	Language Objects and IODDTs associated with Character Mode Communication	108
6.5	The IODDT Type T_GEN_MOD Applicable to All Modules	115

Section 6.1

Language Objects and IODDTs of Modbus and Character Mode Communications

Subject of this Section

This section provides an overview of the general points concerning IODDTs and language objects for Modbus and Character Mode communications.

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction to the Language Objects for Modbus and Character Mode Communications	89
Implicit Exchange Language Objects Associated with the Application-Specific Function	90
Explicit Exchange Language Objects Associated with the Application-Specific Function	91
Management of Exchanges and Reports with Explicit Objects	93

Introduction to the Language Objects for Modbus and Character Mode Communications

General

The IODDTs are predefined by the manufacturer. They contain input/output language objects belonging to the channel of an application-specific module.

Modbus and Character Mode communications have three associated IODDTs:

- T_COM_STS_GEN, which applies to communication protocols except Fipio and Ethernet.
- T_COM_MB_BMX, which is specific to Modbus communication.
- T_COM_CHAR_BMX, which is specific to Character Mode communication.

NOTE: IODDT variables can be created in two different ways:

- Using the I/O objects tab (*see EcoStruxure™ Control Expert, Operating Modes*).
- Using the Data Editor (*see EcoStruxure™ Control Expert, Operating Modes*).

Types of Language Objects

In each IODDT we find a set of language objects that enable us to control them and check that they are operating correctly.

There are two types of language objects:

- Implicit Exchange Objects: These objects are automatically exchanged on each cycle revolution of the task associated with the processor.
- Explicit Exchange Objects: These objects are exchanged on the application's request, using explicit exchange instructions.

Implicit exchanges concern the status of the processors, communication signals, slaves, etc.

Explicit exchanges are used to define the processor settings and perform diagnostics.

Implicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

Use of an integrated, application-specific interface or the addition of a module automatically enhances the language objects application used to program this interface or module.

These objects correspond to the input/output images and software data of the module or integrated application-specific interface.

Reminders

The module inputs (%I and %IW) are updated in the PLC memory at the start of the task, or when the PLC is in RUN or STOP mode.

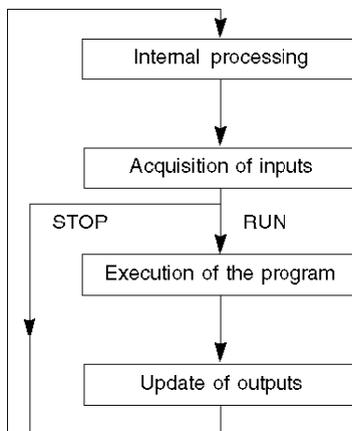
The outputs (%Q and %QW) are updated at the end of the task, only when the PLC is in RUN mode.

NOTE: When the task is in STOP mode, either of the following are possible, depending on the configuration selected:

- Outputs are set to fallback position (fallback mode).
- Outputs are maintained at their last value (maintain mode).

Illustration

The diagram below shows the operating cycle of a PLC task (cyclical execution):



Explicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

Explicit exchanges are exchanges performed at the user program's request, using the following instructions:

- `READ_STS` (see *EcoStruxure™ Control Expert, I/O Management, Block Library*): read status words
- `WRITE_CMD` (see *EcoStruxure™ Control Expert, I/O Management, Block Library*): write command words

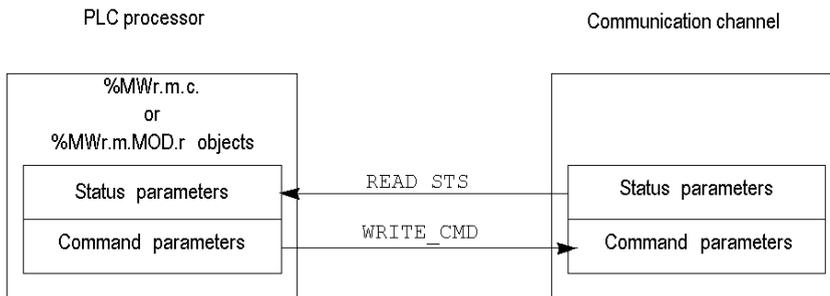
These exchanges apply to a set of `%MW` objects of the same type (status, commands or parameters) belonging to a channel.

NOTE: These objects provide information about the processor or the module, can be used to command them (e.g.: switch command) and to define their operating modes (save and restore adjustment parameters in application).

NOTE: The `READ_STS` and `WRITE_CMD` instructions are executed at the same time as the task that calls them and always correctly. The result of these instructions is available immediately after their execution.

General Principle for Using Explicit Instructions

The diagram below shows the different types of explicit exchanges that can be made between the processor and the communication channel:



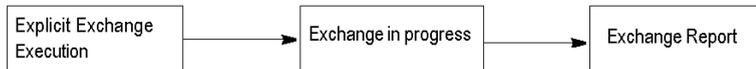
Managing Exchanges

During an explicit exchange, it is necessary to check its performance in order that data is only taken into account when the exchange has been correctly executed.

To this end, two types of information are available:

- Information concerning the exchange in progress (see *EcoStruxure™ Control Expert, I/O Management, Block Library*).
- The exchange report (see *EcoStruxure™ Control Expert, I/O Management, Block Library*).

The following diagram illustrates the management principle for an exchange:



NOTE: In order to avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS ($\%MWr.m.c.0$) of the IODDT associated to the channel before to call any EF using this channel.

Management of Exchanges and Reports with Explicit Objects

At a Glance

When data is exchanged between the PLC memory and the module, the module may require several task cycles to acknowledge this information.

All IODDTs use two words to manage exchanges:

- EXCH_STS (%MWr.m.c.0) : exchange in progress.
- EXCH_RPT (%MWr.m.c.1) : report.

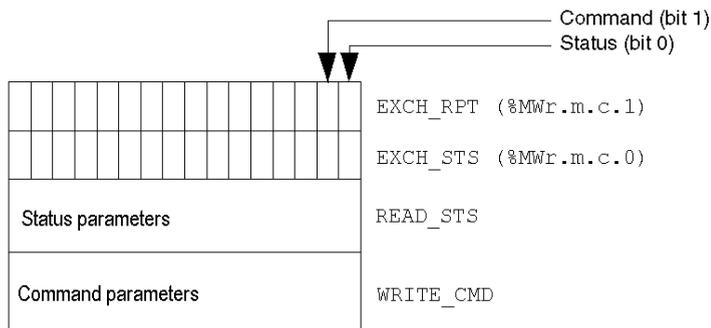
NOTE:

Depending on the localization of the module, the management of the explicit exchanges (%MW0.0.MOD.0.0 for example) will not be detected by the application:

- for in-rack modules, explicit exchanges are done immediately on the local PLC Bus and are finished before the end of the execution task, so the READ_STS, for example, is always finished when the %MW0.0.mod.0.0 bit is checked by the application.
- for remote bus (Fipio for example), explicit exchanges are not synchronous with the execution task, so the detection is possible by the application.

Illustration

The illustration below shows the different significant bits for managing exchanges:



Description of Significant Bits

Each bit of the words `EXCH_STS` (`%MWr.m.c.0`) and `EXCH_RPT` (`%MWr.m.c.1`) is associated with a parameter type:

- Rank 0 bits are associated with the status parameters:
 - The `STS_IN_PROGR` bit (`%MWr.m.c.0.0`) indicates whether a read request for the status words is in progress.
 - The `STS_ERR` bit (`%MWr.m.c.1.0`) specifies whether a read request for the status words is accepted by the module channel.
- Rank 1 bits are associated with the command parameters:
 - The `CMD_IN_PROGR` bit (`%MWr.m.c.0.1`) indicates whether command parameters are being sent to the module channel.
 - The `CMD_ERR` bit (`%MWr.m.c.1.1`) indicates whether or not the command parameters are accepted by the module channel.

NOTE: `r` corresponds to the number of the rack and `m` to the position of the module in the rack, while `c` corresponds to the channel number in the module.

NOTE: Exchange and report words also exist at module level `EXCH_STS` (`%MWr.m.MOD.0`) and `EXCH_RPT` (`%MWr.m.MOD.1`) as per `T_GEN_MOD` type IODDTs.

Explicit Exchange Execution Flags: EXCH_STS

The table below shows the `EXCH_STS` word (`%MWr.m.c.0`) explicit exchange control bits:

Standard symbol	Type	Access	Meaning	Address
<code>STS_IN_PROGR</code>	BOOL	R	Reading of channel status words in progress	<code>%MW_{r.m.c.0.0}</code>
<code>CMD_IN_PROGR</code>	BOOL	R	Command parameters exchange in progress	<code>%MW_{r.m.c.0.1}</code>
<code>ADJ_IN_PROGR</code>	BOOL	R	Adjust parameters exchange in progress	<code>%MW_{r.m.c.0.2}</code>
<code>RECONF_IN_PROGR</code>	BOOL	R	Reconfiguration of the module in progress	<code>%MW_{r.m.c.0.15}</code>

NOTE: If the module is not present or is disconnected, exchanges using explicit objects (`READ_STS`, for example) are not sent to the processor (`STS_IN_PROG` (`%MWr.m.c.0.0`) = 0), but the words are refreshed.

Explicit Exchange Report: EXCH_RPT

The table below shows the EXCH_RPT (%MWr.m.c.1) word report bits:

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Detected error reading channel status words (1 = Detected failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Detected error during a command parameter exchange (1 = Detected failure)	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Interruptions while exchanging adjustment parameters (1 = Detected failure)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Interruptions during reconfiguration of the channel (1 = Detected failure)	%MWr.m.c.1.15

Section 6.2

General Language Objects and IODDTs for Communication Protocols

Subject of this Section

This section presents the general language objects and IODDTs that apply to all communication protocols except Fipio and Ethernet.

What Is in This Section?

This section contains the following topics:

Topic	Page
Details of IODDT Implicit Exchange Objects of Type T_COM_STS_GEN	97
Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN	98

Details of IODDT Implicit Exchange Objects of Type T_COM_STS_GEN

At a Glance

The following table presents the IODDT implicit exchange objects of type T_COM_STS_GEN applicable to all communication protocols except Fipio.

Error bit

The table below presents the meaning of the CH_ERROR error bit (%lr.m.c.ERR):

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Communication channel error bit.	%lr.m.c.ERR

Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN

At a Glance

This section presents the T_COM_STS_GEN type IODDT explicit exchange objects applicable to all communication protocols except Fipio and ethernet. It includes the word type objects whose bits have a specific meaning. These objects are described in detail below.

In this part, the IODDT_VAR1 variable is of type T_COM_STS_GEN.

Observations

In general, the meaning of the bits is given for bit status 1. In specific cases, each bit status is explained.

Not all bits are used.

Explicit Exchange Execution Flags: EXCH_STS

The table below shows the meaning of channel exchange control bits from the EXCH_STS channel (%MWr.m.c.0):

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Read channel status words in progress.	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameter exchange in progress.	%MWr.m.c.0.1

Explicit Exchange Report: EXCH_RPT

The table below presents the meaning of the EXCH_RPT exchange report bits (%MWr.m.c.1):

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Detected read error for channel status words.	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Detected error during command parameter exchange.	%MWr.m.c.1.1

Standard Channel Faults: CH_FLT

The table below shows the meaning of the bits of the status word CH_FLT (%MWr.m.c.2):

Standard symbol	Type	Access	Meaning	Address
NO_DEVICE	BOOL	R	No devices are working on the channel.	%MWr.m.c.2.0
ONE_DEVICE_FLT	BOOL	R	A device on the channel is inoperating.	%MWr.m.c.2.1
BLK	BOOL	R	Terminal block is not connected.	%MWr.m.c.2.2
TO_ERR	BOOL	R	Time out overtaken (analysis needed).	%MWr.m.c.2.3
INTERNAL_FLT	BOOL	R	Detected internal error or channel self-testing.	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Different hardware and software configurations.	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication analysis needed with the channel.	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Application detected error (adjustment or configuration).	%MWr.m.c.2.7

Reading is performed by the READ_STS (IODDT_VAR1) instruction .

Section 6.3

Language Objects and IODDTs Associated with Modbus Communication

Subject of this Section

This section presents the language objects and IODDTs associated with Modbus communication.

What Is in This Section?

This section contains the following topics:

Topic	Page
Details concerning Explicit Exchange Language Objects for a Modbus Function	101
Details of the IODDTs Implicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT	102
Details of the IODDTs Explicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT	103
Details of language objects associated with configuration Modbus mode	106

Details concerning Explicit Exchange Language Objects for a Modbus Function

At a Glance

The table below shows the language objects for Modbus communications in master or slave mode. These objects are not integrated into the IODDTs.

List of Explicit Exchange Objects in Master or Slave mode

The table below shows the explicit exchange objects:

Address	Type	Access	Meaning
%MWr.m.c.4	INT	R	Number of responses received correctly.
%MWr.m.c.5	INT	R	Number of responses received with CRC error.
%MWr.m.c.6	INT	R	Number of responses received with an exception code in slave mode.
%MWr.m.c.7	INT	R	Number of messages sent in slave mode.
%MWr.m.c.8	INT	R	Number of messages sent without response in slave mode.
%MWr.m.c.9	INT	R	Number of responses received with a negative acknowledgement.
%MWr.m.c.10	INT	R	Number of messages repeated in slave mode.
%MWr.m.c.11	INT	R	Number of detected character errors.
%MWr.m.c.24.0	BOOL	RW	Reset of detected error counters.

Details of the IODDTs Implicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT

At a Glance

The tables below show the implicit exchange objects of the IODDTs of types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT that are applicable to Modbus serial communications. They differ in terms of **configuration objects availability** (*see page 105*).

CH_ERROR bit

The following table shows the meaning of the error bit CH_ERROR (%I_{r.m.c}.ERR):

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Communication channel detected error bit	%I _{r.m.c} .ERR

Word object in Modbus Master Mode

The table below shows the meaning of the bit of the INPUT_SIGNALS word (%IW_{r.m.c}.0):

Standard symbol	Type	Access	Meaning	Address
DCD	BOOL	R	Data carrier detect RS232 signal (only applicable to BMX NOM 0200 module)	%IW _{r.m.c} .0.0
CTS	BOOL	R	Clear to send RS232 signal	%IW _{r.m.c} .0.2
DSR	BOOL	R	Data set ready RS232 signal (only applicable to BMX NOM 0200 module)	%IW _{r.m.c} .0.3

NOTE: %IW_{r.m.c}.0.2 is at 1 when the voltage on CTS signal is positive. It is also applicable to DCD and DSR.

Word object in Modbus Slave Mode

The language objects are identical to those of the Modbus master function. Only the objects in the following table differ.

The table below shows the meaning of the bit of the INPUT_SIGNALS word (%IW_{r.m.c}.0):

Standard symbol	Type	Access	Meaning	Address
LISTEN_ONLY	BOOL	R	Listen only mode	%IW _{r.m.c} .0.8

Details of the IODDTs Explicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT

At a Glance

This part presents the explicit exchange objects of the IODDTs of types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT that are applicable to Modbus serial and differ in terms of **configuration objects availability** (*see page 105*). It includes the word type objects whose bits have a specific meaning. These objects are described in detail below.

In this part, the IODDT_VAR1 variable is of the T_COM_STS_GEN type.

Observations

In general, the meaning of the bits is given for bit status 1. In specific cases, each bit status is explained.

Not all bits are used.

Explicit Exchange Execution Flags: EXCH_STS

The following table shows the meanings of the exchange control bits of the EXCH_STS channel (%MWr.m.c.0):

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Reading of channel status words in progress.	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameter exchange in progress.	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjustment parameter exchange in progress (not applicable to the BMX NOM 0200 module).	%MWr.m.c.0.2

Explicit Exchange Report: EXCH_RPT

The table below presents the various meanings of the EXCH_RPT exchange report bits (%MWr.m.c.1):

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Detected read error for channel status words.	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Anomaly during command parameter exchange.	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Anomaly while exchanging adjustment parameters (not applicable to the BMX NOM 0200 module).	%MWr.m.c.1.2

Standard Channel Detected Faults: CH_FLT

The following table explains the various meanings of the CH_FLT status word bits (%MWr.m.c.2):

Standard symbol	Type	Access	Meaning	Address
NO_DEVICE	BOOL	R	No devices are working on the channel.	%MWr.m.c.2.0
ONE_DEVICE_FLT	BOOL	R	A device on the channel is inoperating.	%MWr.m.c.2.1
BLK	BOOL	R	Terminal block is not connected.	%MWr.m.c.2.2
TO_ERR	BOOL	R	Time out overtaken (analysis needed).	%MWr.m.c.2.3
INTERNAL_FLT	BOOL	R	Internal detected error or channel self-testing.	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Different hardware and software configurations.	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication analysis needed with the channel.	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Application detected error (adjustment or configuration error).	%MWr.m.c.2.7

Reading is performed by the READ_STS instruction (IODDT_VAR1).

Specific channel status: %MWr.m.c.3

The table below shows the various meanings of the bits of the PROTOCOL channel status word (%MWr.m.c.3):

Standard symbol	Type	Access	Meaning	Address
PROTOCOL	INT	R	Byte 0 = 16#06 for Modbus Master mode. Byte 0 = 16#07 for Modbus Slave mode. Byte 0 = 16#03 for Character mode.	%MWr.m.c.3

Reading is performed by the READ_STS (IODDT_VAR1) instruction.

Channel command: %MWr.m.c.24

The table below shows the various meanings of the bits of the CONTROL (%MWr.m.c.24) word:

Standard symbol	Type	Access	Meaning	Address
DTR_ON	BOOL	R/W	Set the Data Terminal Ready signal.	%MWr.m.c.24.8
DTR_OFF	BOOL	R/W	Reset the Data Terminal Ready signal.	%MWr.m.c.24.9
TO_MODBUS_MASTER	BOOL	R/W	Change from Character mode or Modbus Slave mode to Modbus Master mode.	%MWr.m.c.24.12
TO_MODBUS_SLAVE	BOOL	R/W	Change from Character mode or Modbus Master mode to Modbus Slave mode.	%MWr.m.c.24.13
TO_CHAR_MODE	BOOL	R/W	Change from Modbus to Character Mode.	%MWr.m.c.24.14

The command is carried out with the `WRITE_CMD (IODDT_VAR1)` instruction.

For further information about how to change protocols, you can refer to **protocol changes** (*see page 117*).

External Configuration Objects of Type `T_COM_MB_BMX_CONF_EXT: %MWr.m.c.24.7` and `%MWr.m.c.25`

The table below shows the meaning of the `CONTROL (%MWr.m.c.24.7)` bit and of the `CONTROL_DATA (%MWr.m.c.25)` word that are specifically intended for the BMX NOM 0200 module programming:

Standard symbol	Type	Access	Meaning	Address
<code>SAVE_SLAVE_ADDR</code>	BOOL	R/W	Save the control data into the FLASH memory	<code>%MWr.m.c.24.7</code>
<code>SLAVE_ADDR</code>	INT	R/W	Modbus slave address to store in the FLASH memory, from 0 to 248 (0 for Master). NOTE: Be aware that this functionality is optional and there is no reason to use it intensively. As the technology involved is a FLASH, it may damage the chip.	<code>%MWr.m.c.25</code>

Details of language objects associated with configuration Modbus mode

At a Glance

The following tables present all configuration language objects for communication Modbus mode. These objects are not integrated in the IODDTs, and may be displayed by the application program.

List of explicit exchange objects for Master mode

The table below shows the explicit exchange objects.

Address	Type	Access	Meaning
%KWr.m.c.0	INT	R	The byte 0 of this word corresponds to the type: <ul style="list-style-type: none"> Value 6 corresponds to Master Value 7 corresponds to Slave
%KWr.m.c.1	INT	R	The byte 0 of this word corresponds to the transmission speed. This byte can take several values: <ul style="list-style-type: none"> Value -2 (0xFE) corresponds to 300 bits/s Value -1 (0xFF) corresponds to 600 bits/s Value 0 (0x00) corresponds to 1200 bits/s Value 1 (0x01) corresponds to 2400 bits/s Value 2 (0x02) corresponds to 4800 bits/s Value 3 (0x03) corresponds to 9600 bits/s Value 4 (0x04) corresponds to 19200 bits/s (default value) Value 5 (0x05) corresponds to 38400 bits/s Value 6 (0x06) corresponds to 57600 bits/s (applicable to BMX NOM 0200 module only) Value 7 (0x07) corresponds to 115200 bits/s (applicable to BMX NOM 0200 module only) The byte 1 of this word corresponds to the format: <ul style="list-style-type: none"> Bit 8: number of bits (1 = 8 bits (RTU), 0 = 7 bits (ASCII)) bit 9 = 1: parity management (1 = with, 0 = without) Bit 10: parity Type (1 = odd, 0 = even) Bit 11: number of stop bits (1 = 1 bit, 0 = 2 bits) Bit 13: physical line (1 = RS232, 0 = RS485) Bit 14: DTR/DSR/DCD modem signals (applicable to BMX NOM 0200 module only and for RS232 physical line only). If this bit is set to 1, modem signals are managed. Bit 15 : RTS/CTS hardware flow management signals. If RS232 is selected this bit can take 2 different values: 0 for RX/TX and 1 for RX/TX + RTS/CTS. If RS485 is selected the default value is 0 and corresponds to RX/TX.
%KWr.m.c.2	INT	R	Delay between frames (in RTU mode only): value in ms from 2 to 10000 ms (depends on the transmission speed and format selected). Its default value is 2 ms if the default box is checked. 10 s corresponds to infinite wait.

Address	Type	Access	Meaning
%KWr.m.c.3	INT	R	In Modbus Master Mode this object corresponds to the answer delay in ms from 10 ms to 1000 ms. 100 ms is the value by default. 10 s corresponds to infinite wait.
%KWr.m.c.4	INT	R	Only available in Modbus Master mode. Byte 0 of this word is the number of retries from 0 to 15. The value by default is 3.
%KWr.m.c.5	INT	R	If RS232 is selected this word corresponds to RTS/CTS delay time in hundreds of ms from 0 to 100. If RS485 is selected the default value is 0.

List of explicit exchange objects for Slave mode

The language objects for the Modbus slave function are identical to those of the Modbus master function. The only difference is for the following objects:

Address	Type	Access	Meaning
%KWr.m.c.3	INT	R	In Modbus Slave Mode the byte 0 of this object corresponds to the slave number [0/1, 247]. For the BMX NOM 0200 module, the value 0 means that the slave number is coded in the FLASH memory
%KWr.m.c.4	INT	R	Used only in Modbus Master mode.

Section 6.4

Language Objects and IODDTs associated with Character Mode Communication

Subject of this Section

This section presents the language objects and IODDTs associated with Character Mode communication.

What Is in This Section?

This section contains the following topics:

Topic	Page
Details concerning Explicit Exchange Language Objects for Communication in Character Mode	109
Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX	110
Details of IODDT Explicit Exchange Objects of Type T_COM_CHAR_BMX	111
Details of language objects associated with configuration in Character mode	113

Details concerning Explicit Exchange Language Objects for Communication in Character Mode

At a Glance

The following tables show all configuration language objects for communication in Character Mode. These objects are not integrated into the IODDTs.

List of Explicit Exchange Objects

The table below shows the explicit exchange objects:

Address	Type	Access	Meaning
%MWr.m.c.4	INT	R	Anomaly in transmitted characters.
%MWr.m.c.5	INT	R	Anomaly in received characters.
%MWr.m.c.24.0	BOOL	RW	Resets error counters when it is set to 1.
%QWr.m.c.0 = 16#DEAD	INT	RW	Reboot the BMX NOM 0200.

Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX

At a Glance

The tables below show the implicit exchange objects of the IODDT of the T_COM_CHAR_BMX type that are applicable to Character Mode communication.

Error bit

The following table shows the meaning of the error bit CH_ERROR (%Ir.m.c.ERR):

Standard symbol	Type	Access	Meaning	Address
CH_ERROR	EBOOL	R	Communication channel error bit.	%Ir.m.c.ERR

Signal object on input

The table below shows the meaning of the bit of the INPUT_SIGNALS word (%IWIr.m.c.0):

Standard symbol	Type	Access	Meaning	Address
DCD	BOOL	R	Data Carrier Detect RS232 signal (applicable to BMX NOM 0200 module only).	%IWIr.m.c.0.0
CTS	BOOL	R	Clear to send RS232 signal.	%IWIr.m.c.0.2
DSR	BOOL	R	Data Set ready RS232 signal (applicable to BMX NOM 0200 module only).	%IWIr.m.c.0.3

NOTE: %IWIr.m.c.0.2 is at 1 when the voltage on the CTS signal is positive. It is also applicable to DCD and DSR.

Details of IODDT Explicit Exchange Objects of Type T_COM_CHAR_BMX

At a Glance

This part presents the explicit exchange objects of the IODDT of the T_COM_CHAR_BMX type that are applicable to Character Mode communication. It includes the word type objects whose bits have a specific meaning. These objects are described in detail below.

In this part, the IODDT_VAR1 variable is of the T_COM_STS_GEN type.

Observations

In general, the meaning of the bits is given for bit status 1. In specific cases, each bit status is explained.

Not all bits are used.

Explicit Exchange Execution Flag: EXCH_STS

The following table shows the meanings of the exchange control bits of the EXCH_STS channel (%MWr.m.c.0):

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Read channel status words in progress.	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameter exchange in progress.	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjustment parameter exchange in progress (not applicable to BMX NOM 0200 module).	%MWr.m.c.0.2

Explicit Exchange Report: EXCH_RPT

The table below presents the meaning of the EXCH_RPT exchange report bits (%MWr.m.c.1):

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Detected read error for channel status words.	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Anomaly during command parameter exchange.	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Anomaly while exchanging adjustment parameters (not applicable to the BMX NOM 0200 module).	%MWr.m.c.1.2

Standard Channel Detected Faults, CH_FLT

The following table explains the various meanings of the CH_FLT status word bits (%MWr.m.c.2):

Standard symbol	Type	Access	Meaning	Address
NO_DEVICE	BOOL	R	No device is working on the channel.	%MWr.m.c.2.0
ONE_DEVICE_FLT	BOOL	R	A device on the channel is inoperating.	%MWr.m.c.2.1
BLK	BOOL	R	Terminal block is not connected.	%MWr.m.c.2.2

Standard symbol	Type	Access	Meaning	Address
TO_ERR	BOOL	R	Time out overtaken (analysis needed).	%MWr.m.c.2.3
INTERNAL_FLT	BOOL	R	Internal detected error or channel self-testing.	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Different hardware and software configurations.	%MWr.m.c.2.5
COM_FLT	BOOL	R	Communication analysis is needed with the PLC.	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Application detected error (adjustment or configuration error).	%MWr.m.c.2.7

Reading is performed by the `READ_STS` instruction (`IODDT_VAR1`).

Specific Channel Status, %MWr.m.c.3

The table below shows the various meanings of the bits of the `PROTOCOL` (%MWr.m.c.3) channel status word:

Standard symbol	Type	Access	Meaning	Address
PROTOCOL	INT	R	Byte 0 = 16#03 for Character Mode function.	%MWr.m.c.3

Reading is performed by the `READ_STS` (`IODDT_VAR1`) instruction.

%MWr.m.c.24 Channel Command

The table below shows the various meanings of the bits of the `CONTROL` (%MWr.m.c.24) word:

Standard symbol	Type	Access	Meaning	Address
DTR_ON	BOOL	R/W	Set the Data Terminal Ready signal.	%MWr.m.c.24.8
DTR_OFF	BOOL	R/W	Reset the Data Terminal Ready signal.	%MWr.m.c.24.9

The command is carried out with the `WRITE_CMD` (`IODDT_VAR1`) instruction.

For further information about how to change protocols, you can refer to protocol changes (*see page 117*).

%QWr.m.c.0 Word Object

The table below shows the meaning of the bit 0 of %QWr.m.c.0 word:

Standard symbol	Type	Access	Meaning	Address
STOP_EXCH	BOOL	R/W	Stop all exchanges on rising edge (available on the BMX NOM 0200 module only).	%QWr.m.c.0.0

Details of language objects associated with configuration in Character mode

At a Glance

The following tables present all configuration language objects for communication Character mode. These objects are not integrated in the IODDTs, and may be displayed by the application program.

List of explicit exchange objects for Character mode

The table below shows the explicit exchange objects.

Address	Type	Access	Meaning
%KWr.m.c.0	INT	R	The byte 0 of this word corresponds to the type. Value 3 corresponds to Character Mode.
%KWr.m.c.1	INT	R	<p>The byte 0 of this word corresponds to the transmission speed. This byte can take several values:</p> <ul style="list-style-type: none"> ● Value -2 (0xFE) corresponds to 300 bits/s ● Value -1 (0xFF) corresponds to 600 bits/s ● Value 0 (0x00) corresponds to 1200 bits/s ● Value 1 (0x01) corresponds to 2400 bits/s ● Value 2 (0x02) corresponds to 4800 bits/s ● Value 3 (0x03) corresponds to 9600 bits/s (default value) ● Value 4 (0x04) corresponds to 19200 bits/s ● Value 5 (0x05) corresponds to 38400 bits/s ● Value 6 (0x06) corresponds to 57600 bits/s (can be taken only for BMX NOM 0200 module) ● Value 7 (0x07) corresponds to 115200 bits/s (can be taken only for BMX NOM 0200 module) <p>The byte 1 of this word corresponds to the format:</p> <ul style="list-style-type: none"> ● Bit 8: number of bits (1 = 8 bits (RTU), 0 = 7 bits (ASCII)) ● bit 9 = 1: parity management (1 = with, 0 = without) ● Bit 10: parity Type (1 = odd, 0 = even) ● Bit 11: number of stop bits (1 = 1 bit, 0 = 2 bits) ● Bit 13: physical line (1 = RS232, 0 = RS485) ● Bit 14: DTR/DSR/DCD modem signals. For BMX NOM 0200 module and if RS232 is selected, this bit can take 2 different values: 1 means that modem signals are managed, 0 means that they are not (default value for BMX P34 or if RS485 is selected) ● Bit 15 : RTS/CTS hardware flow management signals. If RS232 is selected this bit can take 2 different values: 0 for RX/TX and 1 for RX/TX + RTS/CTS. If RS485 is selected the default value is 0 and corresponds to RX/TX
%KWr.m.c.2	INT	R	Entered value in ms of stop on silence (depends on the transmission speed and format selected). Value 0 means no silence detection.

Address	Type	Access	Meaning
%KWr.m.c.3	INT	R	<p>This word corresponds to the polarization type:</p> <ul style="list-style-type: none"> ● Value 0 on both bit 14 and bit 15 corresponds to no polarization (This is the default value for BMX P34 or if RS232 is selected) ● Bit 14: value 1 corresponds to low impedance (Modbus like) polarization and can be taken only for BMX NOM 0200 module and if RS485 is selected ● Bit 15: value 1 corresponds to high impedance polarization and can be taken only for BMX NOM 0200 module and if RS485 is selected
%KWr.m.c.5	INT	R	<p>This word corresponds to RTS/CTS delay time in hundreds of ms from 0 to 100 if RS232 is selected. If RS485 is selected the default value is 0.</p>
%KWr.m.c.6	INT	R	<p>Bit 0 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the stop checkbox in the Stop on reception area for character 1 when checked ● value 0 corresponds to the stop checkbox in the Stop on reception area for character 1 when unchecked <p>Bit 1 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the Character Included checkbox in the Stop on reception area for character 1 when checked ● value 0 corresponds to the Character Included checkbox in the Stop on reception area for character 1 when unchecked <p>Byte 1 of this word corresponds to the entered value of stop on reception of character 1 from 0 to 255.</p>
%KWr.m.c.7	INT	R	<p>Bit 0 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the stop checkbox in the Stop on reception area for character 2 when checked ● value 0 corresponds to the stop checkbox in the Stop on reception area for character 2 when unchecked <p>Bit 1 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the Character Included checkbox in the Stop on reception area for character 2 when checked ● value 0 corresponds to the Character Included checkbox in the Stop on reception area for character 2 when unchecked <p>Byte 1 of this word corresponds to the entered value of stop on reception of character 2 from 0 to 255.</p>

Section 6.5

The IODDT Type T_GEN_MOD Applicable to All Modules

Details of the Language Objects of the IODDT of Type T_GEN_MOD

Introduction

The Modicon X80 modules have an associated IODDT of type T_GEN_MOD.

Observations

In general, the meaning of the bits is given for bit status 1. In specific cases an explanation is given for each status of the bit.

Some bits are not used.

List of Objects

The table below presents the objects of the IODDT.

Standard Symbol	Type	Access	Meaning	Address
MOD_ERROR	BOOL	R	Module detected error bit	%I.r.m.MOD.ERR
EXCH_STS	INT	R	Module exchange control word	%MWr.m.MOD.0
STS_IN_PROGR	BOOL	R	Reading of status words of the module in progress	%MWr.m.MOD.0.0
EXCH_RPT	INT	R	Exchange report word	%MWr.m.MOD.1
STS_ERR	BOOL	R	Event when reading module status words	%MWr.m.MOD.1.0
MOD_FLT	INT	R	Internal detected errors word of the module	%MWr.m.MOD.2
MOD_FAIL	BOOL	R	module inoperable	%MWr.m.MOD.2.0
CH_FLT	BOOL	R	Inoperative channel(s)	%MWr.m.MOD.2.1
BLK	BOOL	R	Terminal block incorrectly wired	%MWr.m.MOD.2.2
CONF_FLT	BOOL	R	Hardware or software configuration anomaly	%MWr.m.MOD.2.5
NO_MOD	BOOL	R	Module missing or inoperative	%MWr.m.MOD.2.6
EXT_MOD_FLT	BOOL	R	Internal detected errors word of the module (Fipio extension only)	%MWr.m.MOD.2.7
MOD_FAIL_EXT	BOOL	R	Internal detected error, module unserviceable (Fipio extension only)	%MWr.m.MOD.2.8
CH_FLT_EXT	BOOL	R	Inoperative channel(s) (Fipio extension only)	%MWr.m.MOD.2.9
BLK_EXT	BOOL	R	Terminal block incorrectly wired (Fipio extension only)	%MWr.m.MOD.2.10

Standard Symbol	Type	Access	Meaning	Address
CONF_FLT_EXT	BOOL	R	Hardware or software configuration anomaly (Fipio extension only)	%MWr.m.MOD.2.13
NO_MOD_EXT	BOOL	R	Module missing or inoperative (Fipio extension only)	%MWr.m.MOD.2.14

Chapter 7

Dynamic Protocol Switching

Changing Protocol with Modicon M340 Processors

General

This part describes how to change the protocol used by a CPU serial communication using the `WRITE_CMD(IODDT_VAR1)` command. This command can be used to switch between the following three protocols:

- Modbus Slave
- Modbus Master
- Character Mode

NOTE: IODDT_VAR1 variable must be a T_COM_MB_BMX type.

Changing Protocol: The Principle

NOTE: In order for changes to be made from one protocol to another, the processor must initially be configured to Modbus Slave mode.

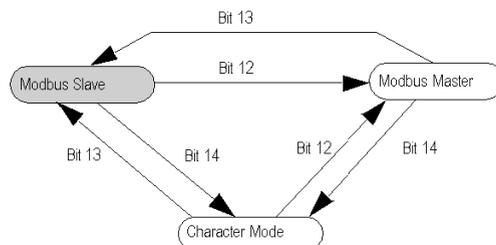
You must create first an IODDT variable linked to the processor's serial channel, then set to 1 the bit of word `IODDT_VAR1.CONTROL (%MWr.m.c.24)` that corresponds to the change of protocol desired:

- `TO_MODBUS_MASTER (Bit 12)`: Current protocol is changed to Modbus Master.
- `TO_MODBUS_SLAVE (Bit 13)`: Current protocol is changed to Modbus Slave.
- `TO_CHAR_MODE (Bit 14)`: Current protocol is changed to Character Mode.

NOTE: `IODDT_VAR1.CONTROL (%MWr.m.c.24)` is part of the IODDT variable `IODDT_VAR1`.

Afterwards, apply the `WRITE_CMD` instruction to the IODDT variable linked to the processor's serial channel.

The diagram below shows the protocol changes to be made according to the bits of the `IODDT_VAR1.CONTROL (%MWr.m.c.24)` word set to 1:



Uses

Three protocol changes are used:

- Transfer to Modbus Master: The protocol change is a two-stage process:
 - Transfer from the Modbus Slave configuration to the Modbus Master configuration
 - Return to the initial Modbus Slave configuration

The aim of Modbus Master configuration is to send information about an event to another PLC. When a change is made from Modbus Slave configuration to Modbus Master configuration, transmission, signal and physical line parameters remain the same. Only the values of the following parameters specific to Modbus Master configuration are changed:

- The Delay Between Frames is set to its default value, which depends on transmission speed.
 - Answer delay is set to 3,000 ms
 - Number of retries set to 3
- Transfer to Character Mode: This protocol change is a two-stage process:
 - Transfer from Modbus Slave configuration to Character Mode configuration
 - Return to the initial Modbus Slave configuration.

The aim of Character Mode configuration is to communicate with a private protocol (a modem, for instance). When a change is made from Modbus Slave configuration to Character Mode configuration, transmission, signal and physical line parameters remain the same. Only the message end parameter specific to Character Mode is set to stop on silence with a timeout of 1000 ms.

- Transfer to the Character Mode and Modbus Master protocols: This protocol change is a three-stage process:
 - Transfer from Modbus Slave configuration to Character Mode configuration.
 - Transfer from Character Mode configuration to Modbus Master configuration.
 - Return to the initial Modbus Slave configuration.

The aim of Character Mode configuration is to communicate with a private protocol (a modem, for instance). Once the exchange has finished, the user switches to the Modbus Master configuration in order to send information about an event to another PLC. Once the message has been sent, the user returns to the initial Modbus Slave configuration.

NOTE: All three cases, the default configuration remains Modbus Slave.

Cold and Warm Starts

Changes in protocol are not affected by the %S0 and %S1 bits (the bits set to 1 during a cold and warm start respectively). However, a cold or warm start of the PLC will configure the serial port to its default values or to values programmed into the application.

Part III

Quick Start : Example of Serial Link Implementation

Overview

This part presents an example of serial link implementation.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
8	Description of the Application	121
9	Installing the Application Using Control Expert	123
10	Starting the Application	151

Chapter 8

Description of the Application

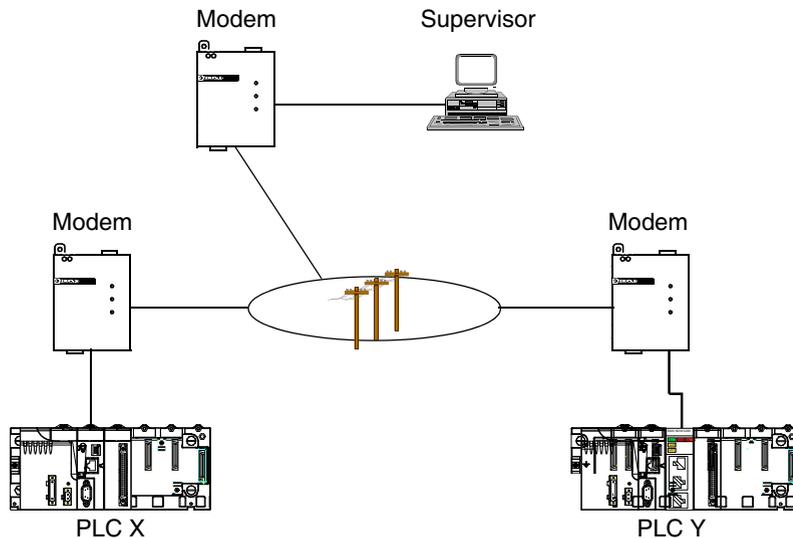
Overview of the Application

At a Glance

The application described in this document is a Modbus communication application via modems.

Example Illustration

The figure below illustrates the example:



The devices communicate with each other using modems. The supervisor is Modbus master whereas the X and Y PLCs are slaves.

The goal of the example is to write the data area values of PLC X to PLC Y.

To do this, the PLC X must become Modbus Master.

Each day, the supervisor communicates with the PLCs to recover information.

If an alarm is raised on the PLC X, it switches in Modbus Master mode and sends data to PLC Y.

To simplify programming, the modems have been initialized with the correct parameters via a programming terminal. These parameters are stored in non-volatile memory by the AT&W commands.

Operating Mode

The operating of the application is as follow:

Step	Action
1	The PLC X port is switched to Character mode.
2	The PLC X sends a dial message to the modem.
3	The PLC X port is switched to Master Modbus mode.
4	The Master PLC (X) sends data to the Slave PLC (Y).
5	The port is switched to character mode.
6	The PLC X sends a disconnection message to the modem.
7	The PLC X port is switched to Slave Modbus mode.

Chapter 9

Installing the Application Using Control Expert

Subject of this Chapter

This chapter describes the procedure for creating the application described. It shows, in general and in more detail, the steps in creating the different components of the application.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
9.1	Presentation of the Solution Used	124
9.2	Developing the Application	125

Section 9.1

Presentation of the Solution Used

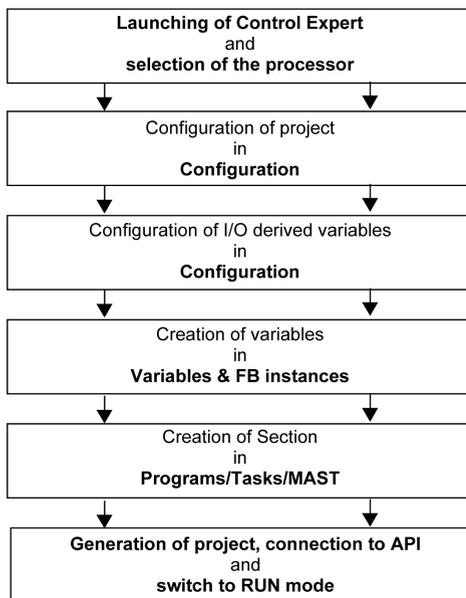
The Different Steps in the Process Using Control Expert

At a Glance

The following logic diagram shows the different steps to follow to create the application. A chronological order must be respected in order to correctly define all of the application elements.

Description

Description of the different types:



Section 9.2

Developing the Application

Subject of this Section

This section gives a step-by-step description of how to create the application using Control Expert.

What Is in This Section?

This section contains the following topics:

Topic	Page
Creating the Project	126
Declaration of Variables	132
Using a Modem	136
Procedure for Programming	138
Programming Structure	139
Programming	142

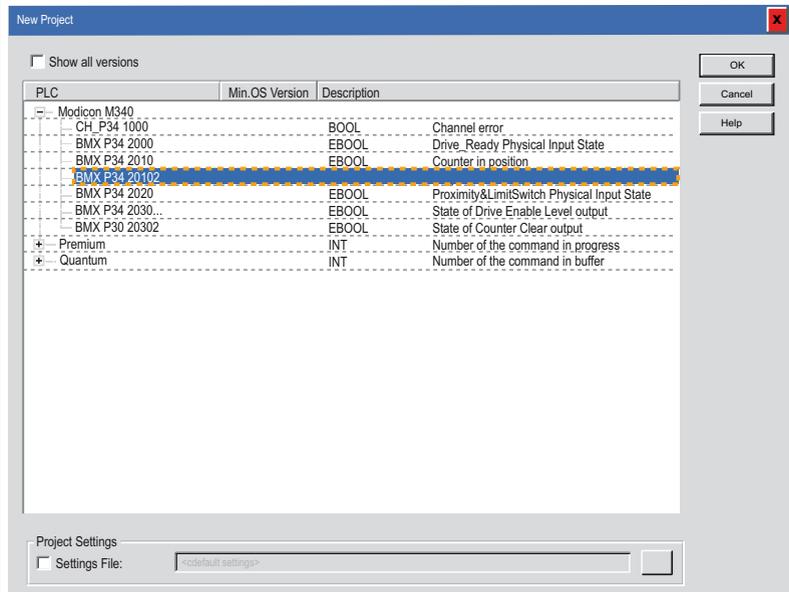
Creating the Project

At a Glance

In order to proceed to the development of the example, a main project associated with the PLC X must be created for configuring the PLC X as well as declaring all needed variables and programming the application. In addition, a separate project must be created for the configuration of PLC Y.

Procedure for Creating a Project

The table below shows the procedure for creating a project using Control Expert.

Step	Action
1	Launch the Control Expert software,
2	Click on File then New to select a BMX P34 20102 processor: 
3	Confirm with OK.

Discrete Input Module Selection

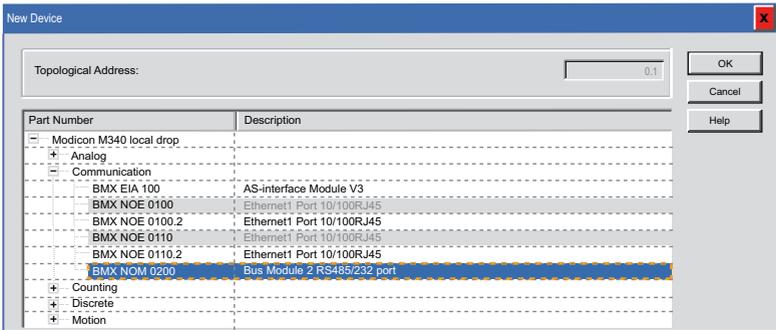
The table below shows the procedure for selecting the discrete module needed by the PLC X.

Step	Action																																																
1	In the Project Browser double-click on Configuration then on 0:PLC Bus and on 0:BMX XBP ... (Where 0 is the rack number),																																																
2	In the PLC Bus window, select a slot (for example slot 1) and double-click on it,																																																
3	Choose the BMX DDI 1602 discrete input module located in the Discrete modules list, <div data-bbox="326 428 1083 1000" style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>New Device</p> <p>Topological address: 0.2</p> <table border="1"> <thead> <tr> <th>Part number</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Modicon M340 local drop</td> <td></td> </tr> <tr> <td> + Analog</td> <td></td> </tr> <tr> <td> + Communication</td> <td></td> </tr> <tr> <td> + Counting</td> <td></td> </tr> <tr> <td> - Discrete</td> <td>Discrete</td> </tr> <tr> <td> BMX DAI 1602</td> <td>Dig 16I 24 Vac/24Vdc Source</td> </tr> <tr> <td> BMX DAI 1603</td> <td>Dig 16I 48 Vac</td> </tr> <tr> <td> BMX DAI 1604</td> <td>Dig 16I 100 to 120 Vac</td> </tr> <tr> <td> BMX DAO 1605</td> <td>Dig 16 O Triacs</td> </tr> <tr> <td> BMX DDI 1602</td> <td>Dig 16I 24 Vdc Sink</td> </tr> <tr> <td> BMX DDI 1603</td> <td>Dig 16I 48 Vdc Sink</td> </tr> <tr> <td> BMX DDI 3202K</td> <td>Dig 32I 24 Vdc Sink</td> </tr> <tr> <td> BMX DDI 6402K</td> <td>Dig 64I 24 Vdc Sink</td> </tr> <tr> <td> BMX DDM 16022</td> <td>Dig 8I 24 Vdc 8Q Source Tr</td> </tr> <tr> <td> BMX DDM 16025</td> <td>Dig 8I 24 Vdc 8Q Relays</td> </tr> <tr> <td> BMX DDM 3202K</td> <td>Dig 16I 24 Vdc 16Q Source Tr</td> </tr> <tr> <td> BMX DDO 1602</td> <td>Dig 16Q Trans Source 0.5A</td> </tr> <tr> <td> BMX DDO 1612</td> <td>Dig 16 O Trans Sink</td> </tr> <tr> <td> BMX DDO 3202K</td> <td>Dig 32Q Trans Source 0.1A</td> </tr> <tr> <td> BMX DDO 6402K</td> <td>Dig 64Q Trans Source 0.1A</td> </tr> <tr> <td> BMX DRA 0805</td> <td>Dig 8Q Isolated Relays</td> </tr> <tr> <td> BMX DRA 1605</td> <td>Dig 16Q Relays</td> </tr> <tr> <td> * Motion</td> <td></td> </tr> </tbody> </table> </div>	Part number	Description	Modicon M340 local drop		+ Analog		+ Communication		+ Counting		- Discrete	Discrete	BMX DAI 1602	Dig 16I 24 Vac/24Vdc Source	BMX DAI 1603	Dig 16I 48 Vac	BMX DAI 1604	Dig 16I 100 to 120 Vac	BMX DAO 1605	Dig 16 O Triacs	BMX DDI 1602	Dig 16I 24 Vdc Sink	BMX DDI 1603	Dig 16I 48 Vdc Sink	BMX DDI 3202K	Dig 32I 24 Vdc Sink	BMX DDI 6402K	Dig 64I 24 Vdc Sink	BMX DDM 16022	Dig 8I 24 Vdc 8Q Source Tr	BMX DDM 16025	Dig 8I 24 Vdc 8Q Relays	BMX DDM 3202K	Dig 16I 24 Vdc 16Q Source Tr	BMX DDO 1602	Dig 16Q Trans Source 0.5A	BMX DDO 1612	Dig 16 O Trans Sink	BMX DDO 3202K	Dig 32Q Trans Source 0.1A	BMX DDO 6402K	Dig 64Q Trans Source 0.1A	BMX DRA 0805	Dig 8Q Isolated Relays	BMX DRA 1605	Dig 16Q Relays	* Motion	
Part number	Description																																																
Modicon M340 local drop																																																	
+ Analog																																																	
+ Communication																																																	
+ Counting																																																	
- Discrete	Discrete																																																
BMX DAI 1602	Dig 16I 24 Vac/24Vdc Source																																																
BMX DAI 1603	Dig 16I 48 Vac																																																
BMX DAI 1604	Dig 16I 100 to 120 Vac																																																
BMX DAO 1605	Dig 16 O Triacs																																																
BMX DDI 1602	Dig 16I 24 Vdc Sink																																																
BMX DDI 1603	Dig 16I 48 Vdc Sink																																																
BMX DDI 3202K	Dig 32I 24 Vdc Sink																																																
BMX DDI 6402K	Dig 64I 24 Vdc Sink																																																
BMX DDM 16022	Dig 8I 24 Vdc 8Q Source Tr																																																
BMX DDM 16025	Dig 8I 24 Vdc 8Q Relays																																																
BMX DDM 3202K	Dig 16I 24 Vdc 16Q Source Tr																																																
BMX DDO 1602	Dig 16Q Trans Source 0.5A																																																
BMX DDO 1612	Dig 16 O Trans Sink																																																
BMX DDO 3202K	Dig 32Q Trans Source 0.1A																																																
BMX DDO 6402K	Dig 64Q Trans Source 0.1A																																																
BMX DRA 0805	Dig 8Q Isolated Relays																																																
BMX DRA 1605	Dig 16Q Relays																																																
* Motion																																																	
4	Confirm with OK.																																																

BMX NOM 0200 Module Selection

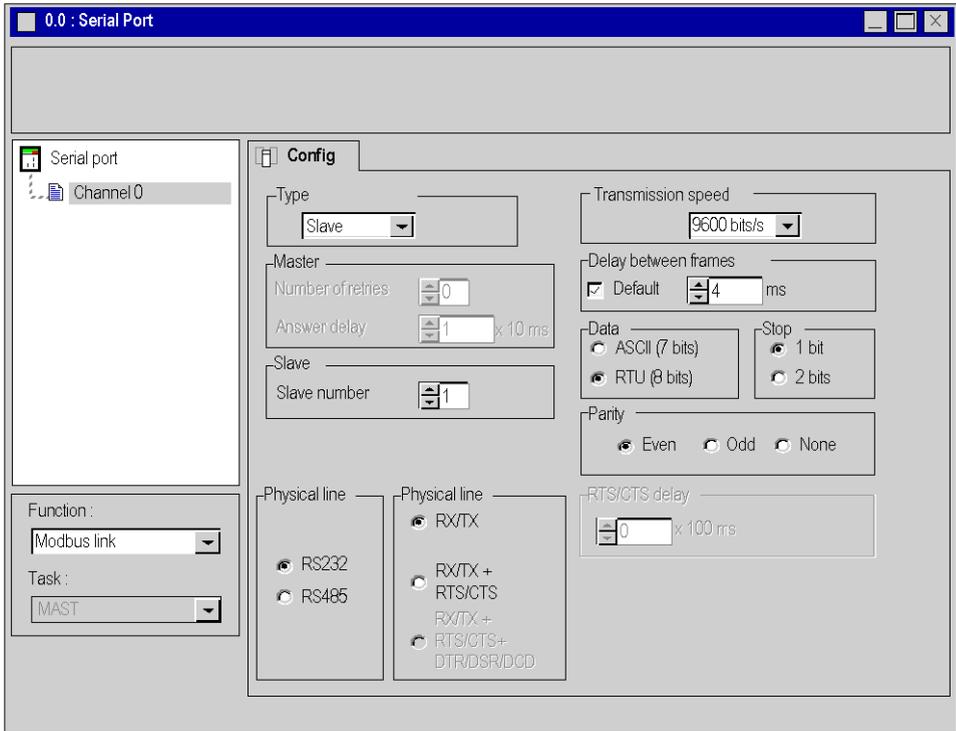
In this example, a BMX NOM 0200 module is used in the PLC Y for the serial link with the modem. Consequently it needs to be added to the project associated with the PLC Y.

The table below shows the procedure for selecting the BMX NOM 0200 module.

Step	Action
1	In the Project Browser double-click on Configuration then on 0:PLC Bus and on 0:BMX XBP ... (Where 0 is the rack number),
2	In the PLC Bus window, select a slot (for example slot 1) and double-click on it,
3	Choose the BMX NOM 0200 communication module located in the Communication modules list, 
4	Confirm with OK.

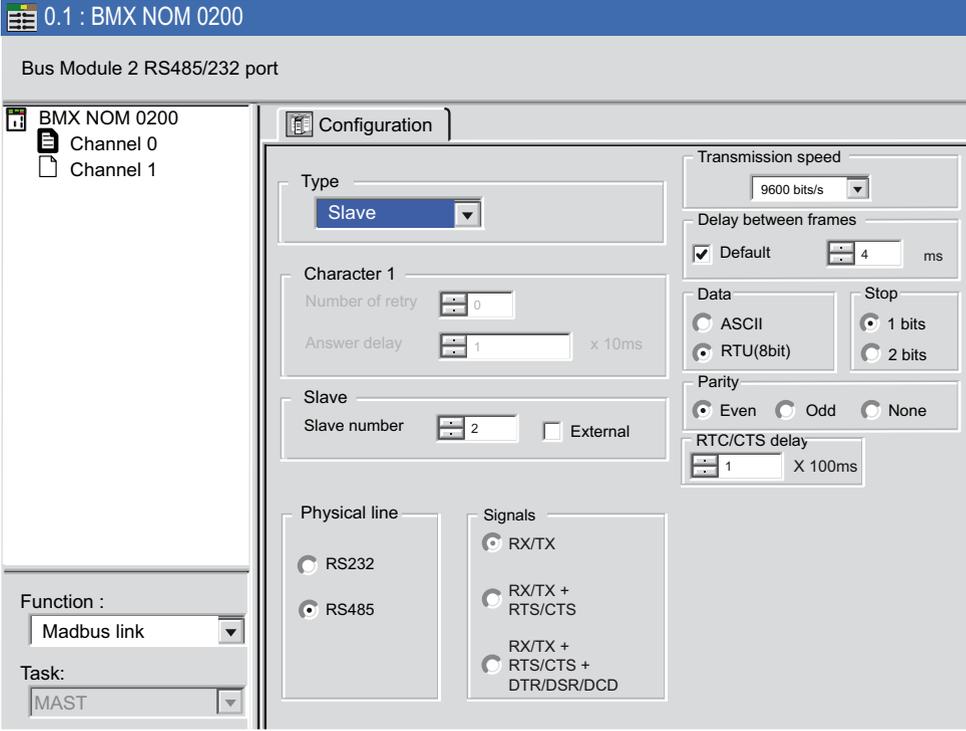
Processor Serial Port Configuration

The table below shows the procedure for configuring the serial port of the PLC X processor as Modbus slave:

Step	Action
1	<p>In the Project Browser double-click on Configuration then on 0:BMX XBP 0800 then on 0:BMX P34 20102. Then double click on Serial Port to access to the 0.0:Serial Port window.</p> 
2	Select the Slave type.
3	Select 9600 bits/s for transmission speed.
4	Select RS232 for physical line.
5	Select RTU (8bits) for data type.
6	Close the window and confirm with OK.

BMX NOM 0200 Serial Channel Configuration

The table below shows the procedure for configuring the serial channel of the PLC Y BMX NOM 0200 module as Modbus slave:

Step	Action
1	<p>In the Project Browser double-click on Configuration then on 0:BMX XBP 0800 then on 0:BMX NOM 0200 to access to the 0.x:BMX NOM 0200 window (where x is the slot number, for example x=1).</p> 
2	Select the Channel 0.
3	Select the Modbus link for function.
4	Select the Slave type.
5	Select 9600 bits/s for transmission speed.
6	Select RS232 for physical line.
7	Select RX/TX + RTS/CTS + DTR/DSR/DCD for signals.
8	Select 100 ms for RTS/CTS delay.

Step	Action
9	Select RTU (8bits) for data type.
10	Close the window and confirm with OK.

Declaration of Variables

At a Glance

All of the variables used in the different sections of the program must be declared.

Undeclared variables cannot be used in the program.

NOTE: For more information, refer to chapter *Data Editor* (see *EcoStruxure™ Control Expert, Operating Modes*).

Procedure for Declaring Variables

The table below shows the procedure for declaring application variables:

Step	Action
1	In Project Browser / Variables & FB instances, double-click on Elementary Variables
2	In the Data Editor window, select the box in the Name column and enter a name for your first variable.
3	Now select a Type for this variable.
4	When all your variables are declared, you can close the window.

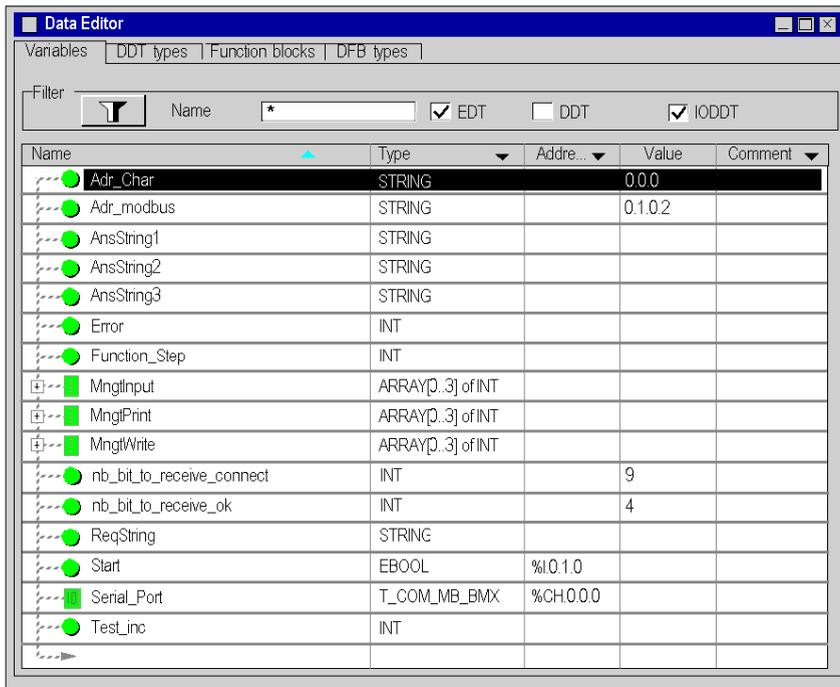
Variables Used for the Application

The following table shows the details of the variables used in the application and declared in the project associated with the PLC X:

Variable	Type	Definition
Adr_Char	STRING	Master PLC serial port address.
Adr_modbus	STRING	Modbus Slave PLC serial channel address (channel 0 of BMX NOM 0200 module).
AnsString1	STRING	First modem answer character string.
AnsString2	STRING	Second modem answer character string.
AnsString3	STRING	Third modem answer character string.
Error	INT	Function error code.
Function_Step	INT	Function step.
MngtInput	ARRAY[0..3] of INT	Array of the communication parameters for the INPUT_CHAR block.
MngtPrint	ARRAY[0..3] of INT	Array of the communication parameters for the PRINT_CHAR block.
MngtWrite	ARRAY[0..3] of INT	Array of the communication parameters for the WRITE_VAR block.

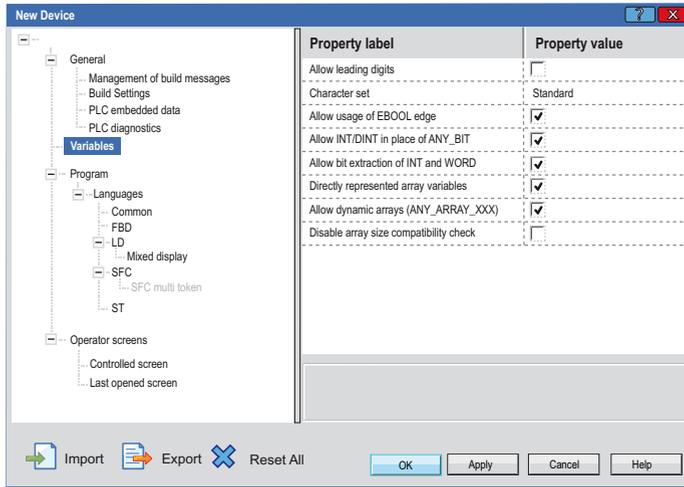
Variable	Type	Definition
nb_charac_to_receive_connect	INT	Number of character to receive: modem connexion
nb_charac_to_receive_ok	INT	Number of character to receive: modem confirmation message
ReqString	STRING	Modem answer.
Start	EBOOL	Starting mode (signal coming from channel 0 of the BMX DDI 1602 module).
Serial_Port	T_COM_MB_BMX	Serial port I/O object
Test_inc	INT	Incrementation value

The following screen shows the application variables created using the data editor:

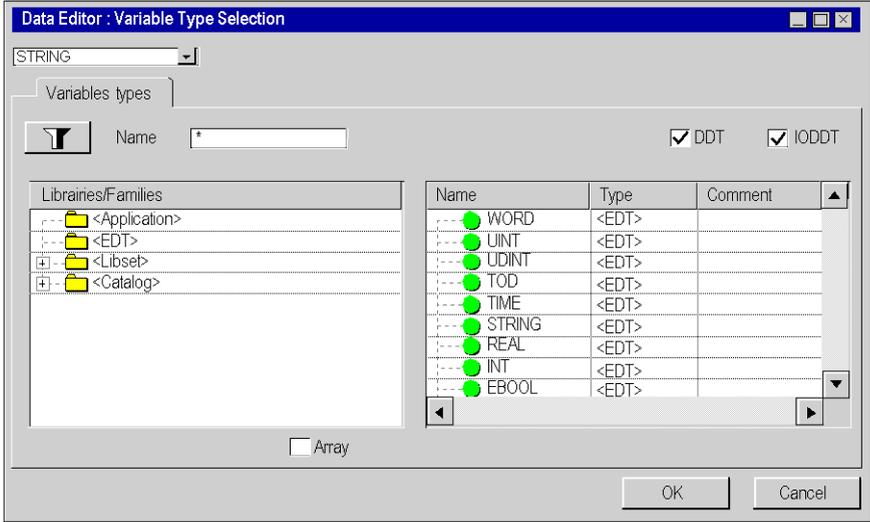


Declaring an Array Type

Before declaring an Array type, click on **Tools/Project Settings/Variables** then check "Directly represented array variables" and "Allow dynamic arrays"



The following table shows how to declare an Array type:

Step	Action
1	In the <code>ProjectBrowser</code> , click on <code>Variables & FB instances</code> .
2	Click in the <code>Name</code> column and enter a name for the variable.
3	<p>Double-click in the <code>Type</code> column and then click on the  button. The <code>Variable Type Selection</code> window opens:</p> 
4	<p>Choose the desired variable type (for example, click on <code><EDT></code> and select <code>INT</code>), then click into the <code>Array</code> checkbox.</p> 
5	Modify the intervalle, then confirm with <code>OK</code> .

Declaration of I/O Objects

For declaring I/O Derived Variables, open the `Variable Type Selection` window as described in the above procedure and click on `<Catalog>` to access the `<IODDT>` type variables (select `T COM MB BMX` for example), then confirm with `OK`.

Using a Modem

Description

It is necessary to know three commands to interface telephonic modems to PLCs. These commands are the following:

- initialize modem,
- renumerate,
- disconnect modem.

It is imperative to send an initialization message followed by a dial message to the modem before sending it an ASCII or Modbus message.

When the connection is successful between the two modems, you may send an unlimited number of ASCII or Modbus messages.

When all the messages have been sent, you must send the disconnection string to the modem.

Initializing the Modem

The two modems must be configured with the same characteristics as the serial ports:

- data rate: 9600 bauds,
- character frame: 8 bits / parity even / 1 stop bit,
- line modulation: V32.

Then define "+" as escape character (command: AT\$S2=43).

Example of initializing command:

```
ATQ0&Q0E0&K0V1
```

with:

- Q0: enable the result code
- &Q0: DTR is always assumed (ON),
- E0: disable the echo of characters,
- &K0: no flow control,
- V1: word result codes.

Dialing the Modem

The dial message is used to send the telephone number to the modem.

Only AT commands relating to dialing should be included in the message.

Example:

- **Frequency dialing:** ATDT6800326<CR><LR>
- **Pulse dialing:** ATDP6800326<CR><LF>
- **Frequency dialing with tone waiting:** ATDTW6800326<CR><LF>

Disconnecting the Modem

The modem is first switched back to the Command Mode by receiving the escape character three times.

Then, the disconnect command "ATH0" can be send.

Escape sequence: "+++" (modem result code: OK),

Disconnect command: "ATH0" (modem result code: OK).

Procedure for Programming

Procedure to Follow

The array below shows the procedure for programming the application.

Step	Action	Details
1	Preparing the communication port.	<ul style="list-style-type: none"> ● Change the Slave Modbus mode to Character mode by sending a WRITE_CMD (<i>see page 138</i>) to the serial port. ● For a modem transmission , send the HAYES command by using the PRINT_CHAR block to configure the modem (<i>see page 136</i>). ● For a modem transmission , send the HAYES command by using the PRINT_CHAR block. The dial message is used to send a telephone number to the modem (<i>see page 136</i>).
2	Master Modbus mode	<ul style="list-style-type: none"> ● Switch to Modbus Master mode using the WRITE_CMD (<i>see page 138</i>) command. ● Send data to write on the Slave PLC.
3	Resetting the communication port.	<ul style="list-style-type: none"> ● Switch to Character mode using the WRITE_CMD (<i>see page 138</i>) command. ● For a modem transmission, send the escape character, then send the disconnect command to send a disconnection message to the modem (<i>see page 137</i>) by using the PRINT_CHAR block. ● Return to the starting mode of the serial port (Slave Modbus) using the WRITE_CMD (<i>see page 138</i>) command.

Writing the Command Words

The following steps should be executed to send a WRITE_CMD to a communication port:

Step	Action	Detail
1	Test to determine whether any command is pending.	Before executing a WRITE_CMD, test whether an exchange is currently in progress using the EXCH_STS language object (%MWr.m.c.0). To refresh this word, use the READ_STS block.
2	Assign the command word.	You must next modify the value of the command language object in order to perform the required command. For a Modbus link, the object language is the internal word CONTROL (%MWr.m.c.24). For example, to switch from Modbus mode to character mode, the bit 14 of the word %MWr.m.c.24 is set to 1. Note: A single command bit must then be switched from 0 to 1 before transmitting the WRITE_CMD.
3	Send the command	Finally, a WRITE_CMD must be executed to acknowledge the command.

Programming Structure

Steps Comments

Step number	Step description	Element
0	Initial state of function When Start bit switches to 1, initialize error to 0 and go to step 5.	Modem
5	Read serial port status and check that no command is active. Switch to Character mode and initialize Test_inc counter to 0. Go to step 10.	
10	Read serial port status and check that no command is active. Reset TO_CHAR_MODE command bit. <ul style="list-style-type: none"> ● If there is no error on the serial port <ul style="list-style-type: none"> ○ and Character mode is active, then go to step 15. ○ and Character mode is not active, then increment Test_inc and retry step 10 up to 1000 times. After 1000 failing retries, set Error to 10 and go to step 130. ● If there is an error on the serial port then <ul style="list-style-type: none"> ○ set Error to 10. ○ Go to step 130. 	
15	Send a dial command to the modem via the PRINT_CHAR block. Go to step 20.	
20	If the result of PRINT_CHAR is conclusive then go to step 25 otherwise set Error to 20 and go to step 130.	
25	Waiting for the response of the modem via the INPUT_CHAR block. Once the response string is fully received, go to step 30.	
30	If the result of INPUT_CHAR is conclusive then go to step 35 otherwise set Error to 30 and go to step 130.	
35	If the modem responds as expected then go to step 40 otherwise set Error at 35 and go to step 130.	
40	Read serial port status and check that no command is active. Switch to Modbus Master mode and initialize Test_inc counter to 0. Go to step 45.	Modbus Master Mode
45	Read serial port status and check that no command is active. Reset TO_CHAR_MODE command bit. <ul style="list-style-type: none"> ● If there is no error on the serial port <ul style="list-style-type: none"> ○ and Character mode is active, then go to step 50. ○ and Character mode is not active, then increment Test_inc and retry step 45 up to 1000 times. After 1000 failing retries, set Error to 45 and go to step 130. ● If there is an error on the serial port then <ul style="list-style-type: none"> ○ set Error to 45. ○ Go to step 130. 	

Step number	Step description	Element
50	Initialization of WRITE_VAR block parameter. Send data to write on the PLC using the WRITE_VAR function. Go to step 55.	Write function
55	If the result of WRITE_VAR is conclusive then go to step 60 otherwise set Error to 55 and go to step 130.	
60	Read serial port status and check that no command is active. Switch to Character mode and initialize Test_inc counter to 0. Go to step 65	Character mode
65	Read serial port status and check that no command is active. Reset TO_CHAR_MODE command bit. <ul style="list-style-type: none"> ● If there is no error on the serial port <ul style="list-style-type: none"> ○ and Character mode is active, then go to step 70. ○ and Character mode is not active, then increment Test_inc and retry step 65 up to 1000 times. After 1000 failing retries, set Error to 65 and go to step 130. ● If there is an error on the serial port then <ul style="list-style-type: none"> ○ set Error to 65. ○ Go to step 130. 	
70	Send an escape sequence to the modem using the PRINT_CHAR block. Go to step 75.	Modem
75	If the result of PRINT_CHAR is conclusive then go to step 80 otherwise set Error at 75 and go to step 130.	
80	Waiting for the response of the modem via the INPUT_CHAR block. Once the response string is fully received, go to step 85.	
85	If the result of INPUT_CHAR is conclusive then go to step 90 otherwise set Error to 85 and go to step 130.	
90	If the modem responds as expected then go to step 95 otherwise set Error to 90 and go to step 130.	
95	Send a disconnection command to the modem using the PRINT_CHAR block. Go to step 100.	
100	If the result of PRINT_CHAR is conclusive then go to step 105 otherwise set Error to 100 and go to step 130.	
105	Waiting for the response of the modem via the INPUT_CHAR block. Once the response string is fully received, go to step 110.	
110	If the result of INPUT_CHAR is conclusive then go to step 115 otherwise set Error to 110 and go to step 130.	
115	If the modem responds as expected then go to step 120 otherwise set Error to 115 and go to step 130.	

Step number	Step description	Element
120	Read serial port status and check that no command is active. Switch to Modbus Slave mode and initialize Test_inc counter to 0. Go to step 125.	Modbus Slave mode
125	Read serial port status and check that no command is active. Reset TO_CHAR_MODE command bit. <ul style="list-style-type: none">● If there is no error on the serial port<ul style="list-style-type: none">○ and Character mode is active, then go to step 130.○ and Character mode is not active, then increment Test_inc and retry step 125 up to 1000 times. After 1000 failing retries, set Error to 125 and go to step 130.● If there is an error on the serial port then<ul style="list-style-type: none">○ set Error to 125.○ Go to step 130.	
130	Return to step 0.	

Programming

Programming in ST Language.

The example is programmed in ST language. The dedicated section is under the same master task (MAST).

```
CASE Function_Step OF
```

```
0: (* Initialization *)
```

```
IF (Start) THEN (* trigger flag *)
```

```
Error := 0;
```

```
Function_Step := 5; (* next step *)
```

```
END_IF;
```

```
5: (* Send command to switch serial port from Slave Modbus mode to Character mode *)
```

```
READ_STS(Serial_port); (* read serial port status *)
```

```
IF (Serial_port.EXCH_STS = 0) THEN (* no active command *)
```

```
Serial_port.CONTROL := 16#00; (* reset control word *)
```

```
(* set TO_CHAR_MODE command bit *)
```

```
SET(Serial_port.TO_CHAR_MODE);
```

```
WRITE_CMD (Serial_port); (* send command *)
```

```
Test_inc := 0; (* initialize retry counter *)
```

```
Function_Step := 10; (* next step *)
```

```
END_IF;
```

```
10: (* Test result of switch command to Character mode*)
```

```
READ_STS(Serial_port); (* read serial port status *)
```

```
IF (Serial_port.EXCH_STS = 0) THEN (* command completed *)
```

```
(* reset TO_CHAR_MODE command bit *)
```

```
RESET(Serial_port.TO_CHAR_MODE);
```

```
IF (Serial_port.EXCH_RPT = 0) THEN (* no error *)
```

```
IF (AND(Serial_port.PROTOCOL, 16#0F) = 03)
```

```
THEN (* Character mode OK *)
```

```
Function_Step := 15; (* next step *)
```

```
ELSE
Test_inc := Test_inc + 1;
IF (Test_inc > 1000) THEN
Error := 10; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;
ELSE (* error in sending command to port *)
Error := 10; (* error *)
Function_Step := 130;
END_IF;
END_IF;

15: (* Send dial command to modem *)
(*Phone number must be inserted between 'ATDT' and '$N'*)
ReqString := 'ATDT4001$N'; (* dial message *)
MngtPrint[2] := 500; (* timeout *)
MngtPrint[9] := 9; (* exchange size in byte *)
PRINT_CHAR(ADDM(Adr_Char), ReqString, MngtPrint);
Function_Step := 20;
20: (* Test PRINT_CHAR function result *)
IF (NOT MngtPrint[0].0) THEN
IF (MngtPrint[1] = 0) THEN
Function_Step := 25; (* success : next step *)
ELSE
Error := 20; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;
25: (* Waiting for the response via INPUT_CHAR *)
MngtInput[2] := 500; (* timeout *)
AnsString1:= ' ';
(* wait modem reply *)
```

```
INPUT_CHAR(ADDM(Adr_Char), 1, nb_charac_to_receive_connect, MngtInput, AnsString1);  
Function_Step := 30; (* next step *)
```

```
30: (* Test INPUT_CHAR function result *)  
IF (NOT MngtInput[0].0) THEN  
IF (MngtInput[1] = 0) THEN  
Function_Step := 35; (* success : next step *)  
ELSE  
Error := 30; (* error *)  
Function_Step := 130; (* next step = end *)  
END_IF;  
END_IF;
```

```
35: (* Test Modem reply *)  
IF (AnsString1 = '$NCONNET') THEN  
Function_Step := 40; (* success : next step *)  
ELSE  
Error := 35; (* error *)  
Function_Step := 130; (* next step = end *)  
END_IF;
```

```
40: (* Send command to switch serial port from character mode to Modbus Master *)  
READ_STS(Serial_port); (* read serial port status *)  
IF (Serial_port.EXCH_STS = 0) THEN (* no active command *)  
Serial_port.CONTROL := 16#00; (* reset control word *)  
(* set TO_MODBUS_MASTER command bit *)  
SET(Serial_port.TO_MODBUS_MASTER);  
WRITE_CMD (Serial_port); (* send command *)  
Test_inc := 0; (* initialize retry counter *)  
Function_Step := 45; (* next step *)  
END_IF;
```

```

45: (* Test result of switch command to Modbus Master mode*)
READ_STS(Serial_port); (* read serial port status *)
IF (Serial_port.EXCH_STS = 0) THEN (* command completed *)
(* TO_MODBUS_MASTER command bit *)
RESET(Serial_port.TO_MODBUS_MASTER);
IF (Serial_port.EXCH_RPT = 0) THEN (* no error *)
IF (AND(Serial_port.PROTOCOL, 16#0F) = 06)
THEN (* Modbus Master mode OK *)
Function_Step := 50; (* next step *)
ELSE
Test_inc := Test_inc + 1;
IF (Test_inc > 1000) THEN
Error := 45; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;
ELSE (* error in sending command to port *)
Error := 45; (* error *)
Function_Step := 130;
END_IF;
END_IF;

50: (*Write information in the second CPU*)
Mngtwrite[2]:=50; (* time outs*)
%MW40:=5; (* value to send *)
WRITE_VAR(ADDM(Adr_modbus),'%MW',100,2,%MW40:2,Mngtwrite);
Function_Step := 55;

55: (* Test WRITE_VAR function result *)
IF (NOT Mngtwrite[0].0) THEN
IF (Mngtwrite[1] = 0) THEN
Function_Step := 60; (* success : next step *)

```

```
ELSE
Error := 55; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;

60: (* Send command to switch serial port from Modbus to character mode *)
READ_STS(Serial_port); (* read serial port status *)
IF (Serial_port.EXCH_STS = 0) THEN (* no active command *)
Serial_port.CONTROL := 16#00; (* reset control word *)
(* set TO_CHAR_MODE command bit *)
SET(Serial_port.TO_CHAR_MODE);
WRITE_CMD (Serial_port); (* send command *)
Test_inc := 0; (* initialize retry counter *)
Function_Step := 65; (* next step *)
END_IF;

65: (* Test result of switch command *)
READ_STS(Serial_port); (* read serial port status *)
IF (Serial_port.EXCH_STS = 0) THEN (* command completed *)
(* reset TO_CHAR_MODE command bit *)
RESET(Serial_port.TO_CHAR_MODE);
IF (Serial_port.EXCH_RPT = 0) THEN (* no error *)
IF (AND(Serial_port.PROTOCOL, 16#0F) = 03)
THEN (* character mode OK *)
Function_Step := 70; (* next step *)
ELSE
Test_inc := Test_inc + 1;
IF (Test_inc > 1000) THEN
Error := 65; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;
```

```
ELSE (* error in sending command to port *)
Error := 65; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;
```

```
70: (* Hangup modem: step 1*)
ReqString := '+++'; (* escape sequence *)
MngtPrint[3] := 3; (* exchange size in byte *)
PRINT_CHAR(ADDM(Adr_Char), ReqString, MngtPrint);
Function_Step := 75; (* next step *)
```

```
75: (* Test PRINT_CHAR function result *)
IF (NOT MngtPrint[0].0) THEN
IF (MngtPrint[1] = 0) THEN
(* Success : next step *)
Function_Step := 80;
ELSE
(* End on error *)
Error := 75;
Function_Step := 130;
END_IF;
END_IF;
```

```
80:
MngtInput[2] := 50; (* timeout *)
INPUT_CHAR(ADDM(Adr_Char), 1, nb_charac_to_receive_ok, MngtInput, AnsString2); (*Wait
modem reply*)
```

```
Function_Step := 85; (*next step*)
```

```
85: (* Test INPUT_CHAR function result *)
IF (NOT MngtInput[0].0) THEN
IF (MngtInput[1] = 0) THEN
(* Success : next step *)
Function_Step := 90;
```

```
ELSE
(* End on error *)
Error := 85;
Function_Step := 130;
END_IF;
END_IF;
90: (* Test Modem reply *)
IF (AnsString2 = '$NOK') THEN
Function_Step := 95; (* success : next step *)
ELSE
Error := 90; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
95: (* Hangup modem: step 2 *)
ReqString := 'ATH0$N'; (* hangup message *)
MngtPrint[3] := 3; (* exchange size in byte *)
PRINT_CHAR(ADDM(Adr_Char), ReqString, MngtPrint);
Function_Step := 100; (* next step *)
100: (* Test PRINT_CHAR function result *)
IF (NOT MngtPrint[0].0) THEN
IF (MngtPrint[1] = 0) THEN
(* Success : next step *)
Function_Step := 105;
ELSE
(* End on error *)
Error := 100;
Function_Step := 130;
END_IF;
END_IF;
105:
MngtInput[2] := 50; (* timeout *)
INPUT_CHAR(ADDM(Adr_Char), 1, nb_charac_to_receive_ok, MngtInput, AnsString3); (*Wait
modem reply*)
Function_Step := 110; (*next step*)
```

```
110: (* Test INPUT_CHAR function result *)
IF (NOT MngtInput[0].0) THEN
IF (MngtInput[1] = 0) THEN
(* Success : next step *)
Function_Step := 115;
ELSE
(* End on error *)
Error := 110;
Function_Step := 130;
END_IF;
END_IF;

115: (* Test Modem reply *)
IF (AnsString3 = '$NOK') THEN
Function_Step := 120; (* success : next step *)
ELSE
Error := 115; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;

120: (* Send command to switch serial port from Character mode to Slave Modbus mode *)
READ_STS(Serial_port); (* read serial port status *)
IF (Serial_port.EXCH_STS = 0) THEN (* no activecommand *)
Serial_port.CONTROL := 16#00; (* reset control word *)
(* set TO_MODBUS_SLAVE command bit *)
SET(Serial_port.TO_MODBUS_SLAVE);
WRITE_CMD (Serial_port); (* send command *)
Test_inc := 0; (* initialize retry counter *)
Function_Step := 125; (* next step *)
END_IF;

125: (* Test result of switch command *)
READ_STS(Serial_port); (* read serial port status *)
```

```
IF (Serial_port.EXCH_STS = 0) THEN (* command completed *)
(* reset TO_MODBUS_SLAVE command bit *)
RESET(Serial_port.TO_MODBUS_SLAVE);
IF (Serial_port.EXCH_RPT = 0) THEN (* no error *)
IF (AND(Serial_port.PROTOCOL, 16#0F) = 07)
THEN (* character mode OK *)
Function_Step := 130; (* next step *)
ELSE
Test_inc := Test_inc + 1;
IF (Test_inc > 1000) THEN
Error := 125; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;
ELSE (* error in sending command to port *)
Error := 125; (* error *)
Function_Step := 130; (* next step = end *)
END_IF;
END_IF;
130: (* End *)
IF (NOT Start) THEN (* trigger flag *)
Function_Step := 0; (* goto waiting state *)
END_IF;
END_CASE;
```

Chapter 10

Starting the Application

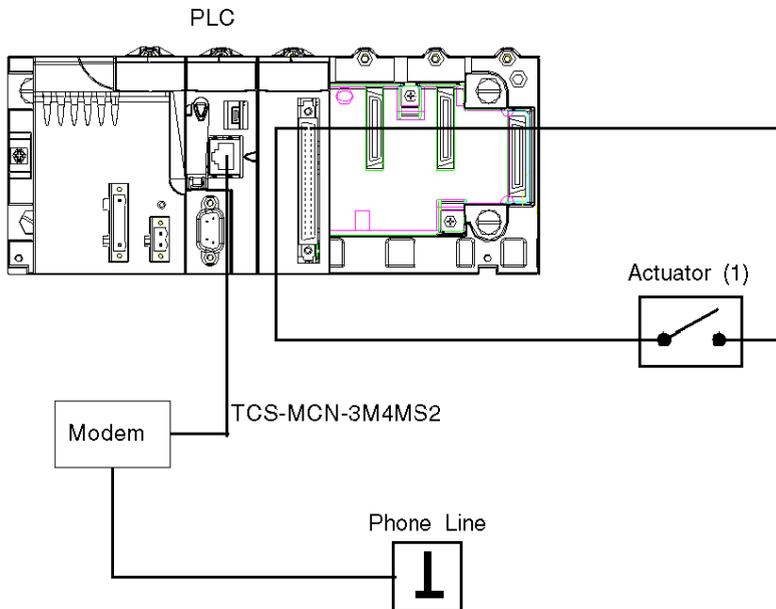
Execution of the Application in Standard Mode

At a Glance

In this example, standard mode working requires the use of two PLCs, a discrete input module, a BMX NOM 0200 module, and 2 SR2MOD01 modems.

First Slave PLC Wiring

The first slave PLC is connected as following:

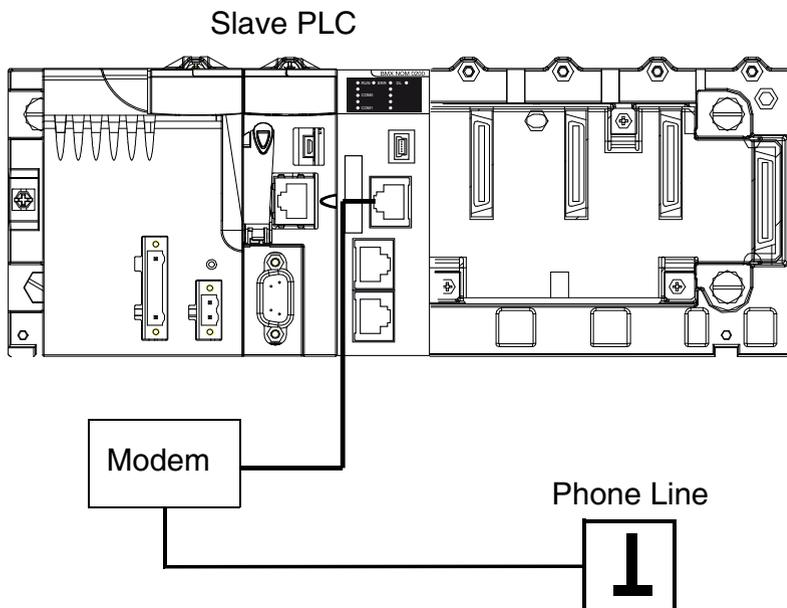


(1): the actuator is connected on the channel 0 of the discrete module.

In this example, the first modem is connected to the processor serial port of the first slave PLC. The actuator state controls the `Start` variable state in the application.

Second Slave PLC Wiring

The second slave PLC is connected as following:



In this example, the second modem is connected to the channel 0 of the BMX NOM 0200 module of the second slave PLC.

For a better communication reliability, the cable TCS XCN 3M4F3S4 is used for DTR/DSR/DCD modem signals handling by the application.

Configuration of the Second Slave PLC

Before transferring the project for configuring the second slave PLC, verify that the second slave PLC is not connected to the modem.

The table below shows the procedure for transferring the project in standard mode:

Step	Action
1	In the PLC menu, click on <code>Standard Mode</code> ,
2	In the <code>Build</code> menu, click on <code>Rebuild All Project</code> . Your project is generated and is ready to be transferred to the PLC.
3	In the PLC menu, click on <code>Connect</code> . You are now connected to the PLC.
4	In the PLC menu, click on <code>Transfer Project to PLC</code> . The <code>Transfer Project to PLC</code> window opens. Click on <code>Transfer</code> . The application is transferred to the PLC.
5	Connect the second slave PLC to a SR2MOD01 modem.

Application Transfer to the First Slave PLC

Before transferring the application, verify that the first slave PLC is not connected to the modem.

The table below shows the procedure for transferring the application in standard mode:

Step	Action
1	In the <code>PLC</code> menu, click on <code>Standard Mode</code> ,
2	In the <code>Build</code> menu, click on <code>Rebuild All Project</code> . Your project is generated and is ready to be transferred to the PLC. When you generate the project, you will see a results window. If there is an error in the program, Control Expert indicates its location (click on the highlighted sequence).
3	In the <code>PLC</code> menu, click on <code>Connect</code> . You are now connected to the PLC.
4	In the <code>PLC</code> menu, click on <code>Transfer Project to PLC</code> . The <code>Transfer Project to PLC</code> window opens. Click on <code>Transfer</code> . The application is transferred to the PLC.

Application Execution on the First Slave PLC

The table below shows the procedure for executing the application in standard mode:

Step	Action
1	In the <code>PLC</code> , click on <code>Run</code> . The <code>Run</code> window opens. Click on <code>OK</code> . The application is now being executed on the PLC.
2	Disconnect the PC which is running Control Expert software from the first slave PLC.
3	Connect the first slave PLC to a SR2MOD01 modem.



!

%I

According to the CEI standard, %I indicates a language object of type discrete IN.

%IW

According to the CEI standard, %IW indicates a language object of type analog IN.

%KW

According to the CEI standard, %KW indicates a language object of type constant word.

%M

According to the CEI standard, %M indicates a language object of type memory bit.

%MW

According to the CEI standard, %MW indicates a language object of type memory word.

%Q

According to the CEI standard, %Q indicates a language object of type discrete OUT.

%QW

According to the CEI standard, %QW indicates a language object of type analog OUT.

A

Address

On a network, the identification of a station. In a frame, a grouping of bits that identifies the frame's source or destination.

Altivar

AC variable speed drive.

ARRAY

An ARRAY is a table containing elements of a single type. The syntax is as follows: ARRAY [<limits>] OF <Type> Example: ARRAY [1..2] OF BOOL is a one-dimensional table with two elements of type BOOL. ARRAY [1..10, 1..20] OF INT is a two-dimensional table with 10x20 elements of type INT.

ASCII

ASCII is the abbreviation of American Standard Code for Information Interchange. This is an American code (but which has become an international standard) that uses 7 bits to define every alphanumerical character used in English, punctuation symbols, certain graphic characters and other miscellaneous commands.

B

BOOL

BOOL is the abbreviation for the Boolean type. This is the basic data type in computing. A BOOL variable can have either of the following two values: 0 (FALSE) or 1 (TRUE). A bit extracted from a word is of type BOOL, for example: %MW10.4.

Broadcast

Broadcast communications send packets from one station to every network destinations. Broadcast messages pertain to every network devices or only one device for which the address is not known.

BYTE

When 8 bits are grouped together, they are called a BYTE. You can enter a BYTE either in binary mode or in base 8. The BYTE type is encoded in an 8 bit format which, in hexadecimal format, ranges from 16#00 to 16#FF.

C

Configuration

The configuration gathers the data which characterizes the machine (invariant) and which is necessary for the module to operate. All this information is stored in the constant PLC %KW zone. The PLC application cannot modify them.

Control Expert

Schneider Automation PLC programming software.

CPU

CPU is the abbreviation of Central Processing Unit: generic name used for Schneider Electric processors.

CRC

CRC is the abbreviation of Cyclic Redundancy Checksum: it indicates whether no character has been "deformed" during frame transmission.

D

DFB

DFB is the abbreviation of Derived Function Block. DFB types are function blocks that can be defined by the user in ST (Structured Text), IL (Instruction List), LD (Ladder Diagram) or FBD (Function Block Diagram) language. Using these DFB types in an application makes it possible to:

- simplify the design and entry of the program;
- make the program easier to read;
- make it easier to debug;
- reduce the amount of code generated.

DINT

DINT is the abbreviation of Double INTegeR (encoded in 32 bits). The upper/lower limits are as follows: -2 to the power of 31) to $(2$ to the power of 31) - 1. Example: -2147483648, 2147483647, 16#FFFFFFFF.

Discrete Module

Module with discrete inputs/outputs.

E**EBOOL**

EBOOL is the abbreviation of Extended BOOLean. An EBOOL type has a value (0 (FALSE) or 1 (TRUE), but also rising or falling edges and forcing functions. An EBOOL variable occupies one byte in memory. The byte contains the following information:

- one bit for the value;
- one bit for the history (whenever the object changes state, the value is copied to the history bit);
- one bit for forcing (equal to 0 if the object is not forced, or 1 if the bit is forced).

The default value of each bit is 0 (FALSE).

EF

EF is the abbreviation of Elementary Function. This is a block used in a program which performs a predefined logical function. A function does not have any information on the internal state. Several calls to the same function using the same input parameters always return the same output values. You will find information on the graphic form of the function call in the "[functional block (instance)]". Unlike a call to a function block, function calls include only an output which is not named and whose name is identical to that of the function. In FBD, each call is indicated by a unique [number] via the graphic block. This number is managed automatically and cannot be modified. You position and configure these functions in your program in order to execute your application. You can also develop other functions using the SDKC development kit.

F**FBD**

FBD is the abbreviation of Function Block Diagram. FBD is a graphical programming language that works like a flowchart. By adding simple logical blocks (AND, OR, etc.), each function or function block in the program is represented in this graphical format. For each block, the inputs are on the left and the outputs on the right. Block outputs can be linked to inputs of other blocks in order to create complex expressions.

Fipio

Field bus used to connect sensor or actuator type devices.

FLASH memory

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

Frame

A frame is a group of bits that form a discrete block of information. Frames contain network control information or data. The size and composition of a frame is determined by the network technology being used.

Full duplex

A method of data transmission capable of transmitting and receiving over the same channel simultaneously.

H

Half duplex

A method of data transmission capable of communication in either of two directions, but in only one direction at a time.

Hub

A hub device connects a series of flexible and centralized modules to create a network.

I

INT

INT is the abbreviation of single INTeger (encoded in 16 bits). The upper/lower limits are as follows: -2 to the power of 15 to $(2$ to the power of 15) - 1. Example: -32768, 32767, 2#1111110001001001, 16#9FA4.

IODDT

IODDT is the abbreviation of Input/Output Derived Data Type. The term IODDT indicates a structured data type representing a module or a channel of a PLC module. Each expert module has its own IODDTs.

L

LED

LED is the abbreviation of Light emitting diode. An indicator that lights up when electricity passes through it. It indicates the operation status of a communication module.

LRC

LRC is the abbreviation of Longitudinal redundancy check: it has been devised to address the low probability of error detection of parity checking.

M

Master task

Main program task. It is obligatory and is used to carry out sequential processing of the PLC.

Momentum

I/O modules using several open standard communication networks.

N**Network**

There are two meanings of the word "network".

- In LD (Ladder Diagram): a network is a set of interconnected graphic elements. The scope of a network is local, concerning the organizational unit (section) of the program containing the network.
- With expert communication modules: a network is a set of stations that intercommunicate. The term "network" is also used to define a group interconnected graphic elements. This group then makes up part of a program that may comprise a group of networks.

P**PLC**

PLC is the abbreviation of Programmable logic controller. The PLC is the brain of an industrial manufacturing process. It automates a process as opposed to relay control systems. PLCs are computers suited to survive the harsh conditions of the industrial environment.

Protocol

Describes message formats and a set of rules used by two or more devices to communicate using those formats.

R**RS232**

Serial communication standard which defines the voltage of the following service:

- a signal of +12 V indicates a logical 0,
- a signal of -12 V indicates a logical 1.

There is, however, in the case of any attenuation of the signal, detection provided up to the limits -3 V and +3 V. Between these two limits, the signal will be considered as invalid. RS232 connections are quite sensitive to interference. The standard specifies not to exceed a distance of 15 m or a maximum of 9600 bauds (bits/s).

RS485

Serial connection standard that operates in 10 V/+5 V differential. It uses two wires for send/receive. Their "3 states" outputs enable them to switch to listen mode when the transmission is terminated.

RTU

RTU is the abbreviation of Remote Terminal Unit. In RTU mode, data is sent as two four-bit, hexadecimal characters, providing for higher throughput than in ASCII mode for the same baudrate. Modbus RTU is a binary protocol and more time delay critical than the ASCII protocol.

S

Section

Program module belonging to a task which can be written in the language chosen by the programmer (FBD, LD, ST, IL, or SFC). A task can be composed of several sections, the order of execution of the sections corresponding to the order in which they are created. This order is modifiable.

SEPAM

Digital protection relay for protection, control and monitoring of power systems.

Socket

The association of a port with an IP address, serving as an identification of sender or recipient.

ST

ST is the abbreviation of Structured Text. The structured literal language is a developed language similar to computer programming languages. It can be used to organize a series of instructions.

STRING

A STRING variable is a series of ASCII characters. The maximum length of a string is 65,534 characters.

T

TAP

TAP is the abbreviation of Transmission Access Point: the bus connection unit.

Task

A group of sections and subroutines, executed cyclically or periodically for the MAST task, or periodically for the FAST task. A task possesses a level of priority and is linked to inputs and outputs of the PLC. These I/O are refreshed in consequence.

V

Variable

Memory entity of type BOOL, WORD, DWORD, etc., whose contents can be modified by the program currently running.

W

WORD

The type WORD is encoded in a 16 bit format and is used to perform processing on series of bits. This table shows the upper/lower limits of each of the bases that can be used:

Base	Lower limit	Upper limit
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binary	2#0	2#1111111111111111

Examples of representation:

Data	Representation in one of the bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

X

XBT

Graphical operator terminal.

XPS

Safety module used for processing of safety signals to monitor both the component and the wiring of a safety system, including devices for general monitoring as well as application specific models.



B

BMXNOM0200, *17*
BMXP341000, *17*
BMXP342000, *17*
BMXP342010, *17*
BMXP3420102, *17*
BMXP342020, *17*

C

Cabling, *37*
certifications, *24*
changing protocols, *117*
channel data structure for all modules
 T_GEN_MOD, *115, 115*
channel data structure for character mode
communication
 T_COM_CHAR_BMX, *110, 111*
channel data structure for communication
protocols
 T_COM_STS_GEN, *97, 98*
channel data structure for modbus communi-
cation
 T_COM_MB_BMX, *102, 103*
character mode, *69*
configuring character mode, *73*
configuring Modbus, *53*
connection devices, *27*

D

debugging character mode, *84*
debugging Modbus, *67*

I

INPUT_BYTE, *82*
INPUT_CHAR, *82*

M

Modbus bus, *45*

P

parameter settings, *87*
PRINT_CHAR, *82*
programming character mode, *82*
programming Modbus bus, *62*

Q

quick start, *119*

S

standards, *24*

T

T_COM_CHAR_BMX, *110, 111*
T_COM_MB_BMX, *102, 103*
T_COM_STS_GEN, *97, 98*
T_GEN_MOD, *115, 115*

W

wiring accessories, *37*

