Modicon M340 with Unity Pro

Serial Link

User Manual

12/2015

35012430.09

www.schneider-electric.com

Schneider Electric

Table of Contents

Safety Information 9

About the Book 11

Part I Introduction to Modbus Serial and Character Mode Communications 13

Chapter 1 Introduction to Modbus Serial and Character Mode Communications 15

Introduction to Modbus Serial and Character Mode Communications 15

Part II Hardware Installation for Modbus Serial and Character Mode Communications 17

Chapter 2 Introduction to Serial Communications 19

2.1 Serial Link on the BMX P34 1000/2000/2010/20102/2020 Processors 20

Presentation of the Serial Link on the BMX P34 1000/2000/2010/20102/2020 Processors. 20

2.2 2 RS-485/232 ports module BMX NOM 0200 25

Presentation of the BMX NOM 0200 2 RS-485/232 Ports Module . . . 26

Modicon M340H (Hardened) Equipment 32

Grounding of Installed Modules 33

Installation of the Module BMX NOM 0200 35

BMX NOM 0200 Wiring Considerations 37

Chapter 3 Serial Communication Architectures 39

3.1 Serial Communication Architectures for BMX P34 1000/2000/2010/20102/2020 processors 40

Modbus Line Termination and Polarization (RS485). 41

Connecting Modbus Devices (RS485) 43

Connecting Data Terminal Equipment (DTE) (RS232) 46

Connecting Data Circuit-terminating Equipment (DCE) (RS232) 48

3.2 Serial Communication Architectures for BMX NOM 0200 50

Modbus Line Termination and Polarization (RS485). 51

Connecting Modbus Devices (RS485) 53

Connecting Data Terminal Equipment (DTE) (RS232) 55

Connecting Data Circuit-terminating Equipment (DCE) (RS232) 57

3.3 Cabling 59

Cabling 59

Part III	**Software Implementation of Modbus Serial and Character Mode Communications**	**63**
Chapter 4	**Installation Methodology**	**65**
	Introduction to the Installation Phase	**65**
Chapter 5	**Modbus Serial Communication for BMX P34 1000/2000/2010/20102/2020 Processors**	**69**
5.1	Generalities	**70**
	About Modbus Serial	**71**
	Performance	**72**
	How to Access the Serial Link Parameters	**74**
5.2	Modbus Serial Communication Configuration	**77**
	Modbus Serial Communication Configuration Screen	**78**
	Accessible Modbus Functions	**81**
	Default Values for Modbus Serial Communication Parameters	**82**
	Application-linked Modbus Parameters	**83**
	Transmission-linked Modbus Parameters	**85**
	Signal and Physical Line Parameters in Modbus	**87**
5.3	Modbus Serial Communication Programming	**89**
	Services Supported by a Modbus Link Master Processor	**90**
	Services Supported by a Modbus Link Slave Processor	**98**
5.4	Debugging Modbus Serial Communication	**100**
	Modbus Serial Communication Debug Screen	**100**
Chapter 6	**Character Mode Communication for BMX P34 1000/2000/2010/20102/2020 Processors**	**103**
6.1	Generalities	**104**
	About Character Mode Communication	**105**
	Performance	**106**
6.2	Character Mode Communication Configuration	**108**
	Character Mode Communication Configuration Screen	**109**
	Accessible Functions in Character Mode	**112**
	Default Values for Character Mode Communication Parameters	**113**
	Message End Detection Parameters in Character Mode	**114**
	Transmission Parameters in Character Mode	**116**
	Signal and Physical Line Parameters in Character Mode	**118**
6.3	Character Mode Communication Programming	**121**
	Character Mode Communication Functions	**121**
6.4	Debugging Character Mode communication	**128**
	Character Mode Communication Debug Screen	**128**

Chapter 7 Modbus Serial Communication for BMX NOM 0200 . . 131

7.1 Generalities . . . 132

About Modbus Serial . . . 133

Performance . . . 134

How to Access the Serial Link Parameters . . . 136

7.2 Modbus Serial Communication Configuration . . . 139

Modbus Serial Communication Configuration Screen in a Modicon M340 Local Rack . . . 140

BMX NOM 0200 Modbus Serial Communication Configuration Screen in X80 Drop . . . 143

Accessible Modbus Functions . . . 146

Default Values for Modbus Serial Communication Parameters . . . 147

Application-linked Modbus Parameters . . . 148

Transmission-linked Modbus Parameters . . . 150

Signal and Physical Line Parameters in Modbus . . . 152

How to Set the BMX NOM0200 MODBUS Slave Address Without Unity Pro? . . . 154

7.3 Modbus Serial Communication Programming . . . 156

Services Supported by a Modbus Link Master Module . . . 157

Services Supported by a Modbus Link Slave Module . . . 165

Detail of Modbus Expert Mode . . . 167

7.4 Debugging Modbus Serial Communication . . . 174

Modbus Serial Communication Debug Screen . . . 174

Chapter 8 Character Mode Communication for BMX NOM 0200 . 177

8.1 Generalities . . . 178

About Character Mode Communication . . . 178

8.2 Character Mode Communication Configuration . . . 179

BMX NOM 0200 Character Mode Communication Configuration Screen in a Local Rack . . . 180

BMX NOM 0200 Character Mode Communication Configuration Screen in X80 Drop . . . 183

Accessible Functions in Character Mode . . . 186

Default Values for Character Mode Communication Parameters . . . 187

Message End Detection Parameters in Character Mode . . . 188

Transmission Parameters in Character Mode . . . 190

Signal and Physical Line Parameters in Character Mode . . . 192

8.3 Character Mode Communication Programming . . . 195

Character Mode Communication Functions . . . 196

Detail of Character Mode Expert Mode . . . 204

8.4 Debugging Character Mode communication 209
Character Mode Communication Debug Screen 209
Chapter 9 BMX NOM 0200 Module Diagnostics 211
9.1 BMX NOM 0200 Module Diagnostics. 212
Diagnostics of a BMX NOM 0200 Module 213
Detailed Diagnostics by Communication Channel 215
Chapter 10 Language Objects of Modbus and Character Mode Communications 217
10.1 Language Objects and IODDTs of Modbus and Character Mode Communications 218
Introduction to the Language Objects for Modbus and Character Mode Communications 219
Implicit Exchange Language Objects Associated with the Application-Specific Function 220
Explicit Exchange Language Objects Associated with the Application-Specific Function 221
Management of Exchanges and Reports with Explicit Objects 223
10.2 General Language Objects and IODDTs for Communication Protocols 226
Details of IODDT Implicit Exchange Objects of Type T_COM_STS_GEN 227
Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN 228
10.3 Language Objects and IODDTs Associated with Modbus Communication 230
Details concerning Explicit Exchange Language Objects for a Modbus Function 231
Details of the IODDTs Implicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT 232
Details of the IODDTs Explicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT 233
Details of language objects associated with configuration Modbus mode 236
10.4 Language Objects and IODDTs associated with Character Mode Communication 238
Details concerning Explicit Exchange Language Objects for Communication in Character Mode 239
Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX. 240
Details of IODDT Explicit Exchange Objects of Type T_COM_CHAR_BMX. 241
Details of language objects associated with configuration in Character mode 244

10.5 The IODDT Type T_GEN_MOD Applicable to All Modules 246
Details of the Language Objects of the IODDT of Type T_GEN_MOD 246
10.6 Language Objects and Device DDTs Associated with Modbus Communication 248
Communication Device DDT Names 248
Chapter 11 Dynamic Protocol Switching 251
Changing Protocol with BMX P34 1000/2000/2010/20102/2020 Processors 252
Changing Protocol with the BMX NOM 0200 Module 254
Part IV Quick Start: BMX NOM 0200.4 257
Chapter 12 Overview 259
Product Overview 260
Architecture Overview 261
Limitations 263
Chapter 13 Configuration in Unity Pro 265
Module Insertion 266
Module Configuration Screen 267
Part V Quick Start : Example of Serial Link Implementation 271
Chapter 14 Description of the Application 273
Overview of the Application 273
Chapter 15 Installing the Application Using Unity Pro 275
15.1 Presentation of the Solution Used 276
The Different Steps in the Process Using Unity Pro 276
15.2 Developing the Application 277
Creating the Project 278
Declaration of Variables 283
Using a Modem 287
Procedure for Programming 289
Programming Structure 291
Programming 294
Chapter 16 Starting the Application 303
Execution of the Application in Standard Mode 303
Glossary 307
Index 315

Safety Information

Important Information

NOTICE

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

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

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

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

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

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

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This manual describes the principle for hardware and software implementation of character mode and Modbus communication for BMX P34 1000/2000/2010/20102/2020 processors. This manual also describes the hardware and software installation of BMX NOM 0200 communication modules for Modicon M340 PLCs and X80 drops.

Validity Note

This documentation is valid for Unity Pro 10.0 or later.

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
Introduction to Modbus Serial and Character Mode Communications

Chapter 1
Introduction to Modbus Serial and Character Mode Communications

Introduction to Modbus Serial and Character Mode Communications

General

The serial links for BMX P34 1000/2000/2010/20102/2020 processors and the BMX NOM 0200 module 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.

Part II
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
2	Introduction to Serial Communications	19
3	Serial Communication Architectures	39

Chapter 2
Introduction to Serial Communications

Subject of this Chapter

This chapter introduces the serial communications on the BMX P34 1000/2000/2010/20102/2020 processors and on the BMX NOM 0200 module.

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

Using the integrated port of the CPU	Using the BMX NOM 0200 communcation module
- Limited transmission speed - Non isolated serial lines - Provision of power supply to terminal equipment	- 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 sections:

Section	Topic	Page
2.1	Serial Link on the BMX P34 1000/2000/2010/20102/2020 Processors	20
2.2	2 RS-485/232 ports module BMX NOM 0200	25

Section 2.1
Serial Link on the BMX P34 1000/2000/2010/20102/2020 Processors

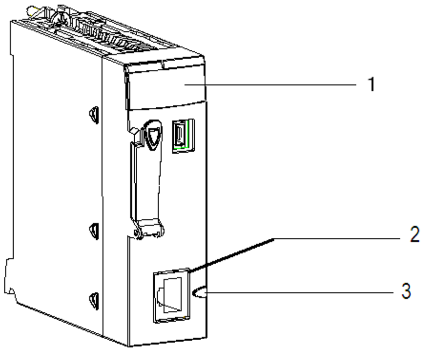
Presentation of the Serial Link on the BMX P34 1000/2000/2010/20102/2020 Processors.

General

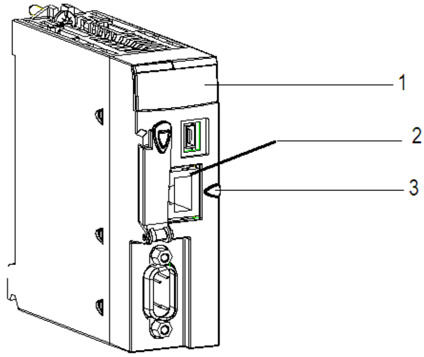
The following processors have an integrated communication channel dedicated to serial communications, and enable communication via serial link:
- BMX P34 1000/2000/2020,
- BMX P34 2010/20102.

Processors Introduction

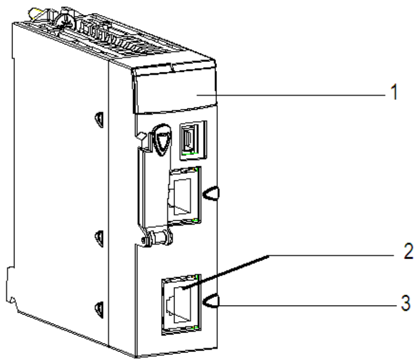
The illustration below shows the physical characteristics of the BMX P34 1000/2000/2010/20102/2020 processors:



BMX P34 1000/2000



BMX P34 2010/20102



BMX P34 2020

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)

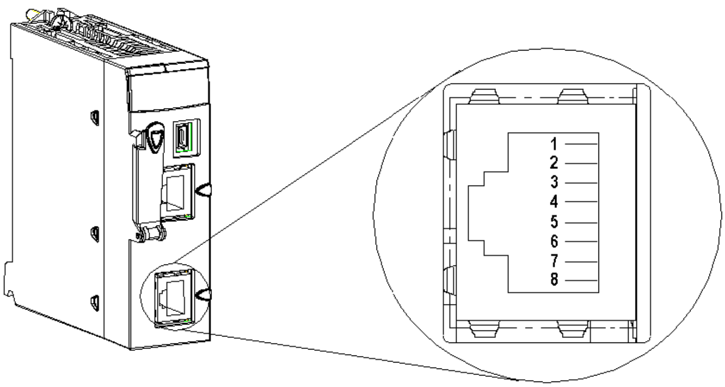
Visual Diagnostic of Serial Communication

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 Introduction

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 μF ceramic capacitor
  - 10 μF tantalum
- Pump charge startup: (on 5 V)
  - 4 x 1 μF ceramic capacitor
  - 2 x 10 μF 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 Unity Pro 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: ● RTU: 256 bytes (252 bytes of data) ● ASCII: 513 bytes (2x252 bytes of data)	Modbus Serial: ● 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.

Section 2.2
2 RS-485/232 ports module BMX NOM 0200

Subject of this Section

This section introduces the serial communications on the BMX NOM 0200 module.

What Is in This Section?

This section contains the following topics:

Topic	Page
Presentation of the BMX NOM 0200 2 RS-485/232 Ports Module	26
Modicon M340H (Hardened) Equipment	32
Grounding of Installed Modules	33
Installation of the Module BMX NOM 0200	35
BMX NOM 0200 Wiring Considerations	37

Presentation of the BMX NOM 0200 2 RS-485/232 Ports Module

General

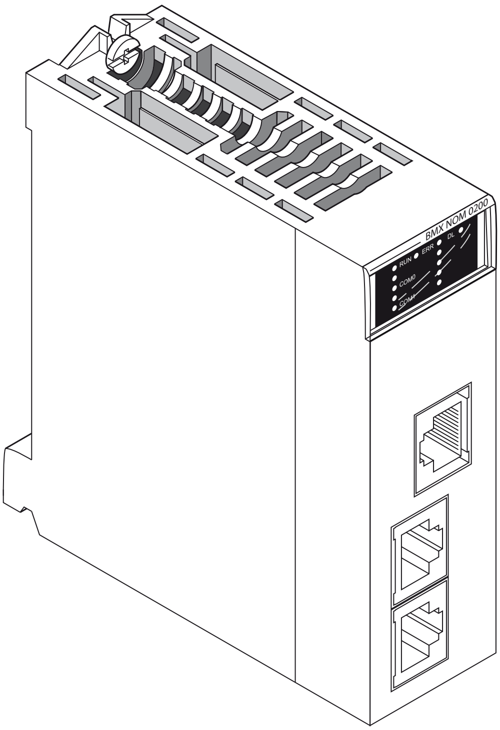
The BMX NOM 0200 and BMX NOM 0200H *(see page 32)* serial link modules are 2-way asynchronous serial line modules supporting Modbus Serial (master or slave) and Character Mode communications.

The BMX NOM 0200 is a simple-format, dedicated module, which can be installed on a Modicon M340 station rack.

NOTE: At the temperature extremes (-25... 0ºC and 60... 70ºC) (-13...32ºF) and (140...158ºF), the BMX NOM 0200H operating characteristics are the same as the BMX NOM 0200 characteristics within its (0…60ºC)(32...140ºF) temperature range.

Module introduction

The illustration below shows the physical characteristics of the BMX NOM 0200 module:



This BMX NOM 0200 module is composed of the elements in the following table:

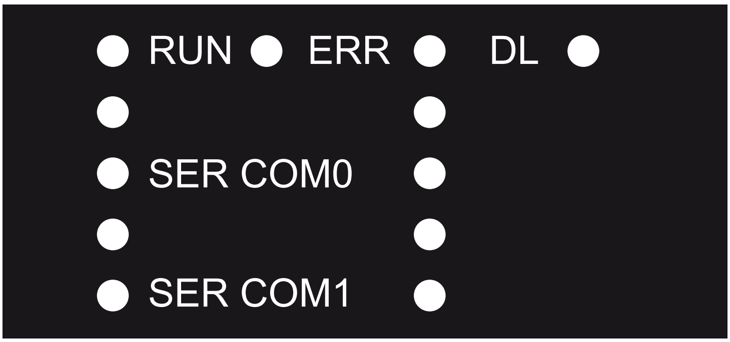
Key	Description
1	Five indicator LEDs on the front of the module: ● RUN and ERR show the module's status, ● SER COM0 displays the traffic status on the port 0 or 1 (channel 0), ● SER COM1 displays the traffic status on the port 2 (channel 1), ● DL shows the firmware download status.
2	Integrated channel (channel 0) dedicated to the serial link with 2 serial ports: RS232 (port 0) and RS485 (port 1). **Note:** Only one port can be active at a time.
3	Integrated channel (channel 1) dedicated to the serial link with 1 serial port: RS485 (port 2).

NOTE: In some operating modes, **LEDs can indicate more specific information** *(see page 27)*.

Visual Diagnostics

Five LEDs are located on the front panel of the BMX NOM 0200 module. They display information about the module operating status and about the communication status of the built-in serial link.

LED Display:



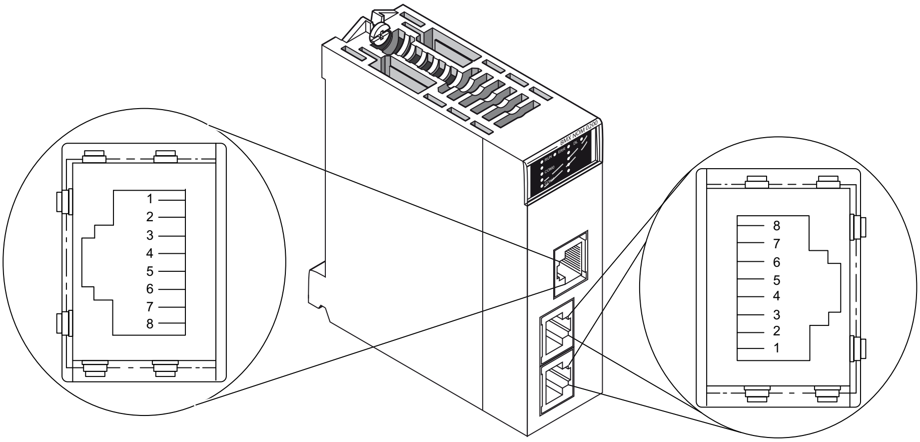
- RUN = The module is powered and well configured.
- ERR = The module has detected an error and cannot function correctly.
- DL = The firmware is being downloaded.
- SER COM0 = Communication detected on port 0 or 1 (channel 0).
- SER COM1 = Communication detected on port 2 (channel 1).

LED meaning:
- Each LED can be in one of these states:
  - 1 = On
  - 0 = Off
  - B = Blinking
- During module startup all LEDs are powered ON and OFF, this verifies that the LEDs are functioning correctly.

RUN	ERR	SER COM0	SER COM1	DL	Diagnose
0	–	–	–	–	The module is not powered or non-operational.
0	B	_	_	_	The module is not configured.
1	1	_	_	_	The module improperly operative.
1	1	1	0	_	The module has detected a problem on the channel 0.
1	1	1	B	_	The module has detected a problem on the channel 0, the channel 1 is exchanging data.
1	1	0	1	_	The module has detected a problem on the channel 1.
1	1	B	1	_	The module has detected a problem on the channel 1, the channel 0 is exchanging data.
1	0	B	_	_	The channel 0 is exchanging data.
1	0	_	B	_	The channel 1 is exchanging data.
B	B	_	_	0	The CPU is absent.
B	B	B	B	_	The module is performing self tests.
–	–	–	–	B	A module firmware is being downloaded.
–	–	–	–	1	The firmware is uploaded; the module must be reset.

Serial Ports Introduction

The illustration below shows the RJ45 serial ports on the BMX NOM 0200:



The table below shows the pin assignment for the serial port on the BMX NOM 0200:

Pin N°	RS485 channel 1 / port 1 or 2	RS232 channel 0 / port 0
1	_	RXD (Receive Data)
2	_	TXD (Transmit Data)
3	_	RTS (Request To Send)
4	D1 (B/B4)	DTR (Data Terminal Ready)
5	D0 (A/A4)	DSR (Data Set Ready)
6	_	CTS (Clear To Send)
7	_	DCD (Data Carrier Detect)
8	Potential serial link grounding (0 V)	Potential serial link grounding (0 V)

NOTE:

- The two RS485 lines are isolated. The isolation voltage between the two serial lines 500 V and between each isolated serial line and the backplane is up to 500V AC.
- The seven-wire RS232 and two-wire RS485 use the same female RJ45 connector. Only the signal cabling is different.

Channels Specifications

The channels of the BMX NOM 0200 module include:

- Two isolated RS485 Physical Interfaces,
- 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 57600 bauds.	19200 bauds. Parameters can be set from 300 bauds to 115200 bauds.	9600 bauds. Parameters can be set from 300 bauds to 57600 bauds.	9600 bauds. Parameters can be set from 300 bauds to 115200 bauds
Number of devices	32	32	–	–
Authorized slave addresses	1 to 247	1 to 247	–	–
Max. length of Bus without branching	Refer to the table below (15 m with Branching)	15 m	Refer to the table below (15 m with Branching	15 m
Message Size	Modbus Serial: ● RTU: 256 bytes (252 bytes of data) ● ASCII: 513 bytes (2x252 bytes of data)	Modbus Serial: ● 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.
Hardware Flow Control	–	Optionally via RTS/CTS signals.	–	Optionally via RTS/CTS signals.

The table below shows the maximum RS485 cable length that can be used, according to the baud rate chosen:

Baud Rate choice (bit/s)	Length (m)	Product reference
300	1000	(1)
600	1000	(1)
1200	1000	(1)
2400	1000	(1)
9600	1000	(1)
19200	600	(1)
38400	300	(1) or (2)
57600	200	(1) or (2)

- (1): Cable shielded twisted pair AWG24 gauge (TSX CSA 100, TSX CSA 200, TSX CSA 500)
- (2): Cable category 5 or higher

Consumption of the BMX NOM 0200 Module

This table shows the consumption of BMX NOM 0200 module:

Voltage	Typical Current	Maximum Current	Typical Power Dissipation	Maximum Power Dissipation
24 V DC	80 mA	130 mA	1.92 W	3.12 W

Modicon M340H (Hardened) Equipment

M340H

The Modicon M340H (hardened) equipment is a ruggedized version of M340 equipment. It can be used at extended temperatures (-25...70ºC) (-13...158ºF) and in harsh chemical environments.

This treatment increases the isolation capability of the circuit boards and their resistance to:

- condensation
- dusty atmospheres (conducting foreign particles)
- chemical corrosion, in particular during use in sulphurous atmospheres (oil, refinery, purification plant and so on) or atmospheres containing halogens (chlorine and so on)

The M340H equipment, when within the standard temperature range (0...60ºC) (32...140ºF), has the same performance characteristics as the standard M340 equipment.

At the temperature extremes (-25... 0ºC and 60... 70ºC) (-13...32ºF and 140...158ºF) the hardened versions can have reduced power ratings that impact power calculations for Unity Pro applications.

If this equipment is operated outside the -25...70ºC (-13...158ºF) temperature range, the equipment can operate abnormally.

⚠ CAUTION
UNINTENDED EQUIPMENT OPERATION Do not operate M340H equipment outside of its specified temperature range. **Failure to follow these instructions can result in injury or equipment damage.**

Hardened equipment has a conformal coating applied to its electronic boards. This protection, when associated with appropriate installation and maintenance, allows it to be more robust when operating in harsh chemical environments.

Grounding of Installed Modules

General

The grounding of Modicon M340 modules is crucial to avoid electric shock.

Grounding Processors and Power Supplies

DANGER

HAZARD OF ELECTRIC SHOCK, EXPLOSION OR ARC FLASH

Ensure ground connection contacts are present and not bent out of shape. If they are, do not use the module and contact your Schneider Electric representative.

Failure to follow these instructions will result in death or serious injury.

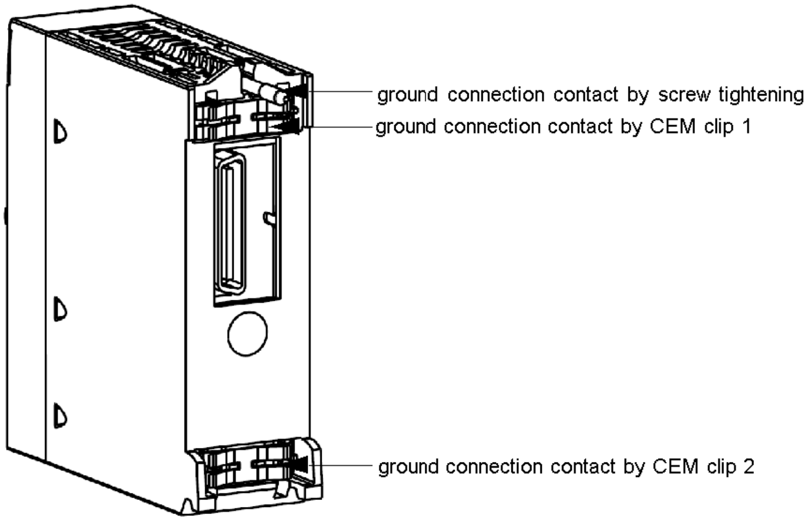
WARNING

UNINTENDED EQUIPMENT OPERATION

Tighten the clamping screws of the modules. A break in the circuit could lead to an unexpected behavior of the system.

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

All Modicon M340 modules are equipped with ground connection contacts at the rear for grounding purposes:



These contacts connect the grounding bus of the modules to the grounding bus of the rack.

Installation of the Module BMX NOM 0200

General

The BMX NOM 0200 module is installed in a Modicon M340 station rack and cannot use the slots required for the power supply and the processor. This installation must conform to the rack installation instructions.

The BMX NOM 0200 module requires the installation of a CPU with minimum OS version 02.10. This installation must conform to the CPU installation instructions.

An RJ45 connector can then be connected to the module according to the targeted network.

NOTE: The BMX NOM 0200 module can be installed in a rack while the application is running on the PLC.

⚠ 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 products. Follow all local and national safety codes and standards. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Number of Modules

Since the number of expert channels managed by a PLC station is related to the processor installed, the maximum number of BMX NOM 0200 modules in a station will therefore rely on:

- The number of channels configured on each BMX NOM 200 module (each channel counts as an expert channel),
- The type and version of processor installed *(see Modicon M340 Using Unity Pro, Processors, Racks, and Power Supply Modules, Setup Manual)*,
- The number of expert channels already used.

When the application is built, Unity Pro checks that the limitation is not exceeded.

Connection/ Disconnection

The BMX NOM 0200 module can be connected or disconnected while the power is on. When the module is disconnected from the rack, its internal memory is erased. The module goes through an initialization phase once it is reconnected to the backplane.

A NOM0200 (since V1.2) can be inserted into a rack at any free slot without have been configured. This is very usefull to connect a PUNIT while the CPU is not configured or as an extra point of connection. In this case the BMX NOM0200 is in default configuration.

The BMX NOM 0200 default configuration is MODBUS slave at address 248, RTU (delay between frames = 2ms), 8bits of data, 1 stop bit, even parity, RS232 at 115200bit/s on channel 0 and RS485 at 57600bit/s on channel 1.

The address 248 is the point-to-point address to which any BMX NOM 0200 slave module answers. This functionality aims at connecting directly to any slave module whose address is unknown.

Firmware Update

The firmware of the BMX NOM 0200 can be updated via the PLC backplane. Firmware update is defined in the *Unity Loader, a SoCollaborative software, User Manual.*

BMX NOM 0200 Wiring Considerations

Operational Consideration

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Although you can connect or disconnect the wires on the BMX NOM 0200 module and BMX P34 20x0 CPUs while the power to the BMX XBP 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 3
Serial Communication Architectures

Subject of this Chapter

This chapter provides an introduction to architectures that use serial communication on the BMX P34 1000/2000/2010/20102/2020 processors and on the BMX NOM 0200 module, as well as cabling requirements.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	Serial Communication Architectures for BMX P34 1000/2000/2010/20102/2020 processors	40
3.2	Serial Communication Architectures for BMX NOM 0200	50
3.3	Cabling	59

Section 3.1
Serial Communication Architectures for BMX P34 1000/2000/2010/20102/2020 processors

Subject of this Section

This section provides an introduction to architectures that use serial communication on the BMX P34 1000/2000/2010/20102/2020 processors, as well as cabling requirements.

What Is in This Section?

This section contains the following topics:

Topic	Page
Modbus Line Termination and Polarization (RS485)	41
Connecting Modbus Devices (RS485)	43
Connecting Data Terminal Equipment (DTE) (RS232)	46
Connecting Data Circuit-terminating Equipment (DCE) (RS232)	48

Modbus Line Termination and Polarization (RS485)

Overview

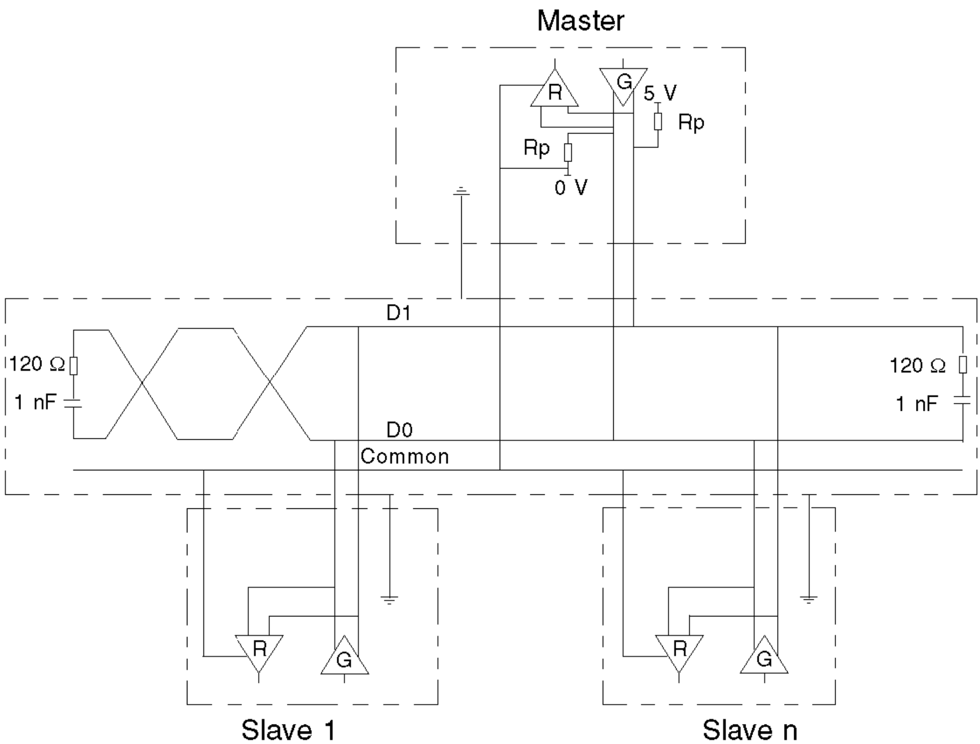
A multi-point Modbus network must have line termination and polarization.

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 45)* 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 23)* 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 (Rp) 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.

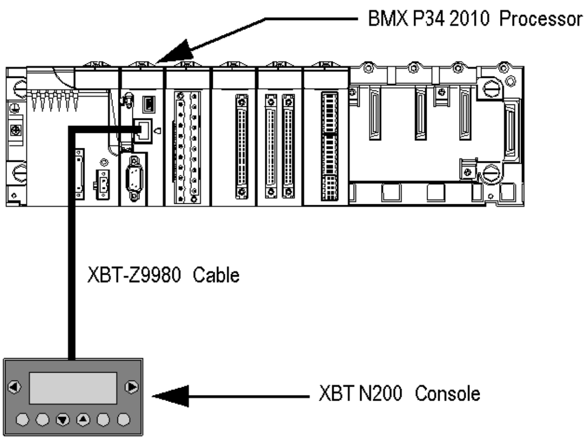
Connecting Modbus Devices (RS485)

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.

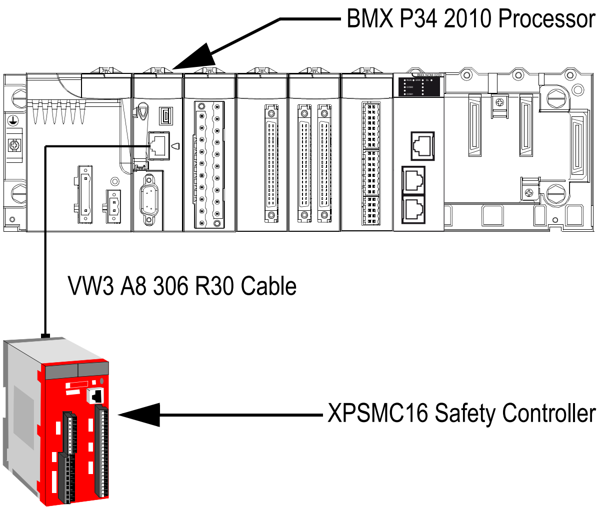
The XBT-Z9980 cable has the following properties:
- Connection: 2 male RJ45 connectors
- Wiring: 2 wires for the RS485 physical line and 2 for the serial link power supply

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:

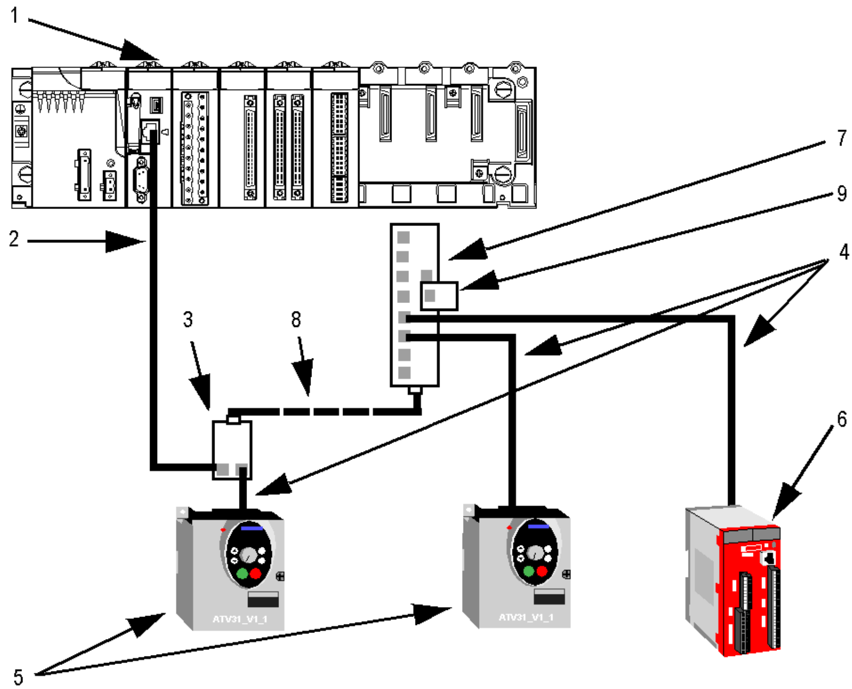
- Connection: 2 male RJ45 connectors
- Wiring: 2 wires for the RS485 physical line

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 R cable

Connecting Data Terminal Equipment (DTE) (RS232)

General

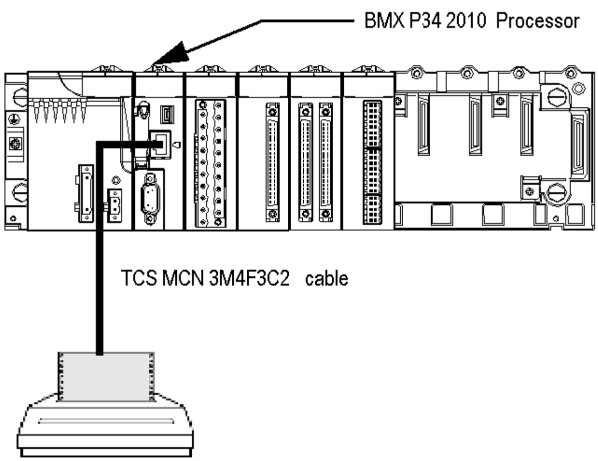
Data terminal equipment is the term used to describe devices such as:

- 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:



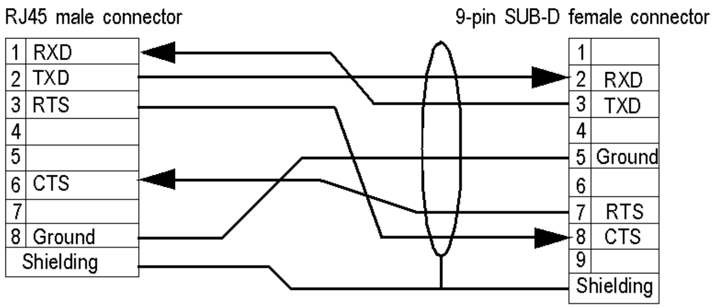
The communication protocol used is Character Mode.

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	• TCS MCN 3M4F3C2 cable • TSX CTC 07 adapter
25-pin SUB-D female connector	• 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.

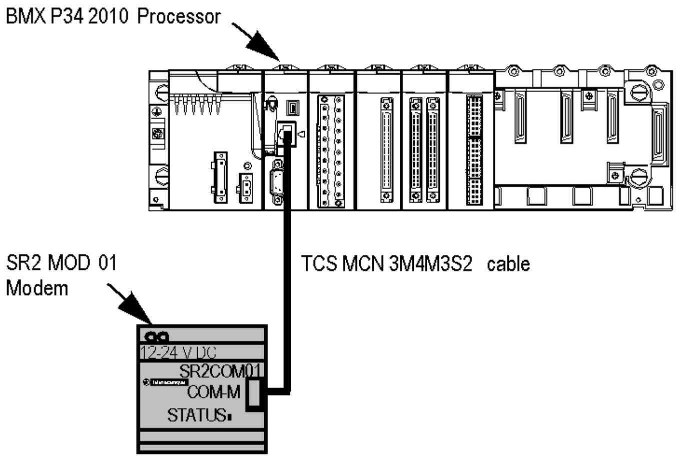
Modem Characteristics

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:

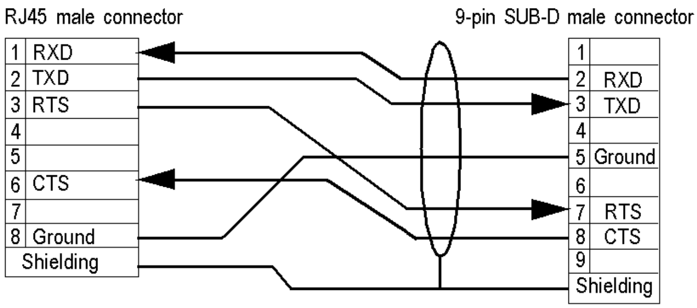


NOTE: In Modbus Serial, the waiting time must be between 100 and 250 ms.

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	• TCS MCN 3M4M3S2 cable • TSX CTC 09 adapter

Section 3.2
Serial Communication Architectures for BMX NOM 0200

Subject of this Section

This section provides an introduction to architectures that use serial communication on the BMX NOM 0200 module, as well as cabling requirements.

What Is in This Section?

This section contains the following topics:

Topic	Page
Modbus Line Termination and Polarization (RS485)	51
Connecting Modbus Devices (RS485)	53
Connecting Data Terminal Equipment (DTE) (RS232)	55
Connecting Data Circuit-terminating Equipment (DCE) (RS232)	57

Modbus Line Termination and Polarization (RS485)

Overview

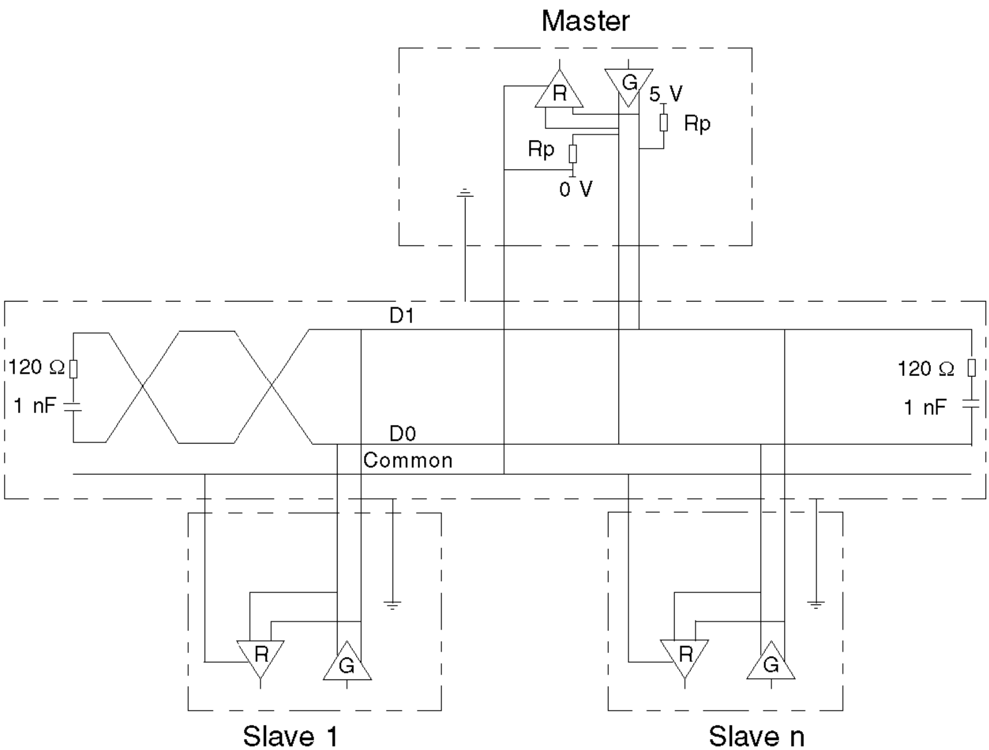
A multi-point Modbus network must have line termination and polarization.

Equipments 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 54)* including a BMX NOM 0200 module 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 306RC 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 BMX NOM 0200 module is used as a master, it is automatically driven by the system so there is no need of external polarization.
- If the BMX NOM 0200 module is used as a slave, the polarization must be implemented by two 450 to 650 Ω resistors (Rp) 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.

NOTE:
In character mode, the line polarization is configurable under Unity Pro. It is possible to choose betwen:

- low impedance polarization like in Modbus networks (the goal of this kind of polarization is to let the master maintain the default state),
- high polarization impedance (the goal of this kind of polarization is to let each device contribute to maintain the default state),
- no polarization (if an external polarization is used).

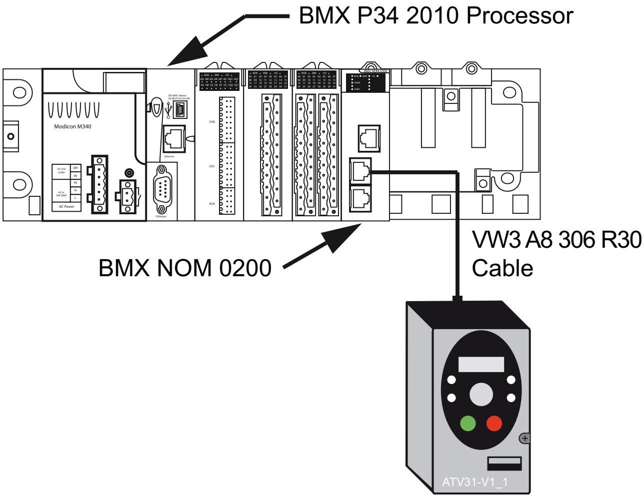
Connecting Modbus Devices (RS485)

General

The following pages present an example of Modbus device connection and a Modbus serial link architecture.

Connecting Modbus devices that are not powered via the Serial Link

The figure below shows a BMX NOM 0200 module connected to an ATV31drive:



The devices are configured as follows:
- A BMX P34 2010 processor,
- A BMX NOM 0200 module configured as master,
- An ATV31 drive configured as slave.

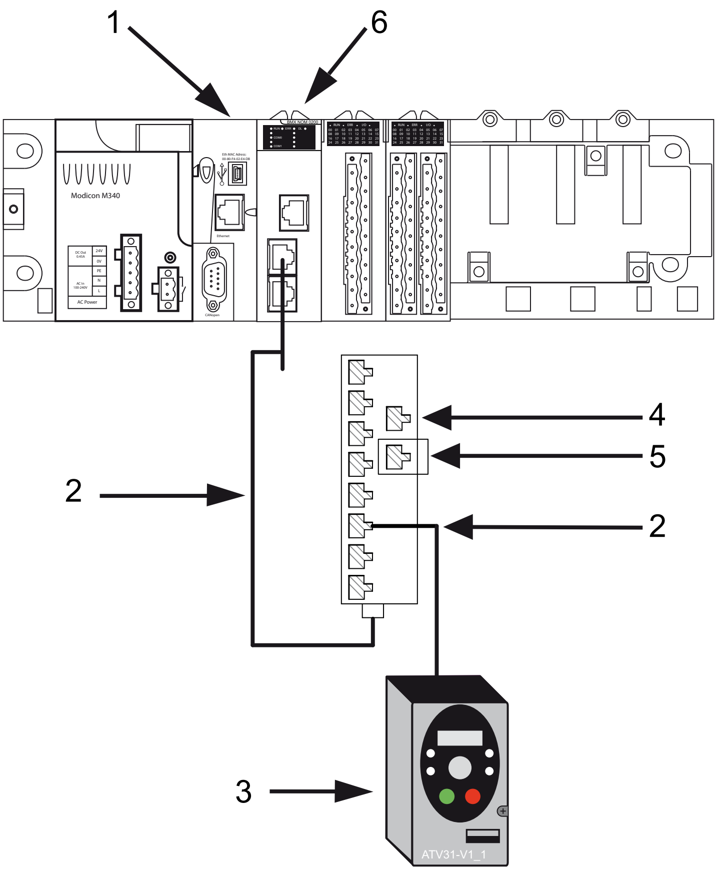
The VW3 A8 306 R30 cable has the following properties:
- Connection: 2 male RJ45 connectors
- Wiring: 2 wires for the RS485 physical line

Modbus Serial Link Architecture

The Modbus serial link architecture consists of the following elements:

- A BMX P34 2010 processor,
- A BMX NOM 0200 module configured as master,
- A TWDXCAISO isolated splitter block,
- An LU9 GC3 splitter block,
- Two ATV31 drives configured as slaves.

The illustration below represents the serial link architecture described above:



1 BMX P34 2010 processor
2 VW3 A8 306 R30 cable
3 ATV31 drive
4 LU9 GC3 splitter block
5 VW3 A8 306 R cable
6 BMX NOM 0200 module

Connecting Data Terminal Equipment (DTE) (RS232)

General

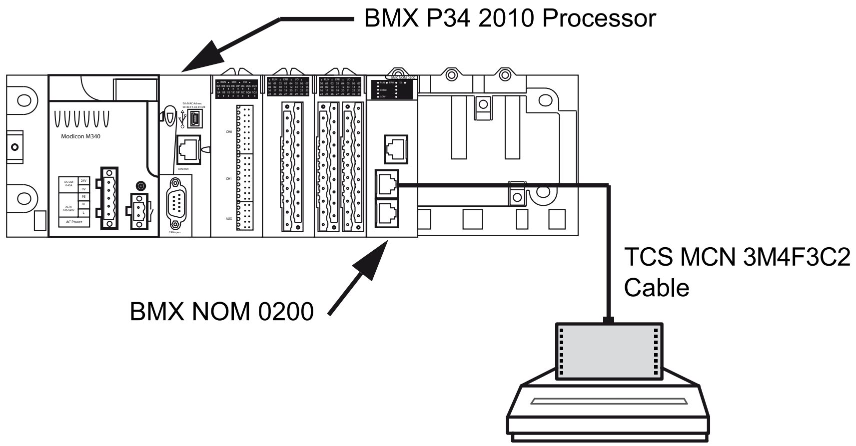
Data terminal equipment is the term used to describe devices such as:
- Common peripherals (printer, keyboard-screen, workshop terminal, etc.)
- Specialized peripherals (barcode readers, etc.)
- PCs

For a DTE type device, the RTS and CTS pins are crossed.

All data terminal equipment is connected to a BMX NOM 0200 module by a serial cross cable using the RS232 physical link.

Connecting Data Terminal Equipment

The figure below shows a printer connected to a BMX NOM 0200 module:



The communication protocol used is Character Mode.

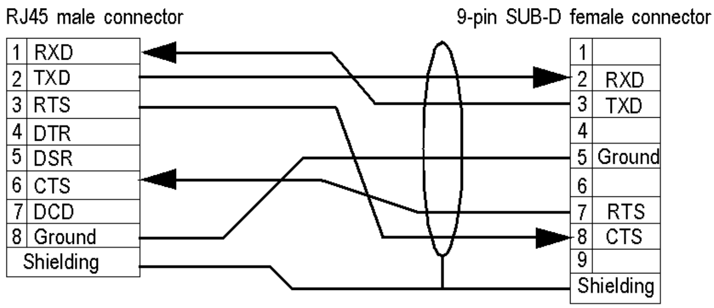
NOTE: Only one item of data terminal equipment may be connected to the BMX NOM 0200 module.

RS 232 Serial Cross Cable

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

- RJ45 male,
- 9-pin SUB-D female.

The figure 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
9-pin SUB-D male connector	TCS MCN 3M4F3C2 cable
25-pin SUB-D male connector	• TCS MCN 3M4F3C2 cable • TSX CTC 07 adapter
25-pin SUB-D female connector	• 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 NOM 0200 module 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

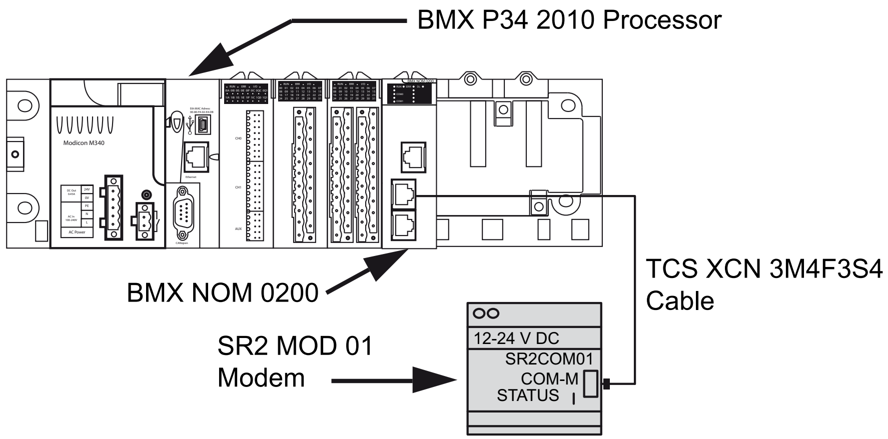
The BMX NOM 0200 module works with most modems on the market. To connect a modem to the serial port of a BMX NOM 0200 module, the modem must have the following characteristics:

- 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.

CTS, DTR, DSR and DCD signals can be managed by the application.

Connecting Data Circuit-terminating Equipment

The figure below shows a modem connected to a BMX NOM 0200 module:



The modem connection needs specific modem cable to work.

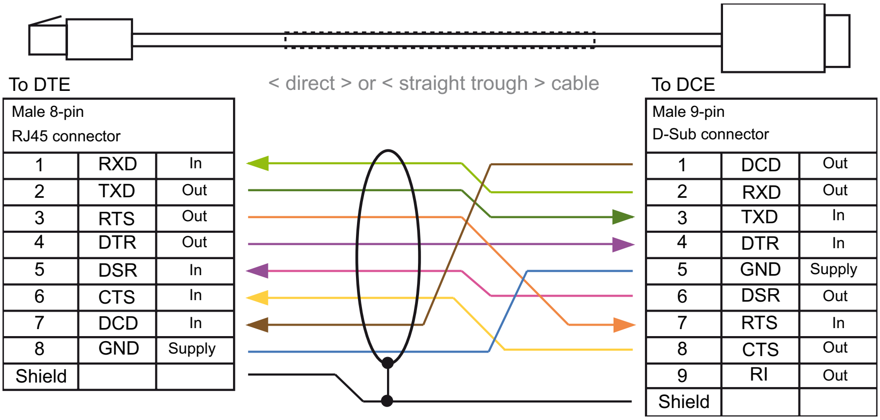
RS 232 Serial Direct Cable

Example of the TCS XCN 3M4F3S4 Cable:

The TCS XCN 3M4F3S4 serial direct cable is an 8 wires version and has two connectors:

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

The illustration below shows the pin assignment for a TCS XCN 3M4F3S4 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
9-pin SUB-D female connector	• TCS MCN 3M4M3S2 cable • TCS XCN 3M4F3S4 cable
25-pin SUB-D female connector	• TCS MCN 3M4M3S2 cable • TSX CTC 09 Adapter

Section 3.3
Cabling

Cabling

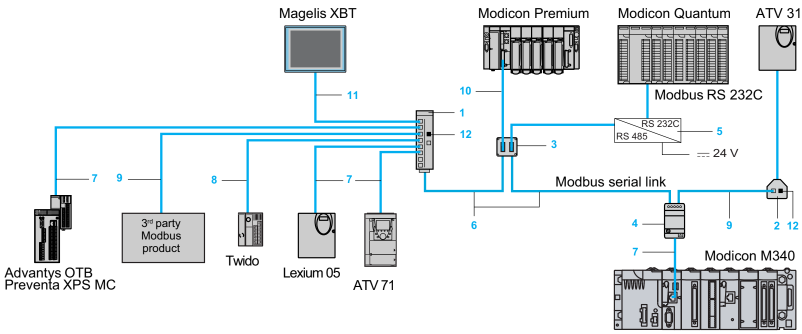
General

Several cables and accessories are required in order to set up a serial link on the following processors and module:

- BMX P34 1000,
- BMX P34 2000,
- BMX P34 2010/20102,
- BMX P34 2020, and
- BMX NOM 0200 module.

Cabling System

The figure below shows an example of Modicon M340 Modbus serial link and character mode cabling system. The **cables** *(see page 60)* and **connecting accessories** *(see page 61)* 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	Length	Characteristics	Product reference
6	RS485 double shielded twisted pair trunk cable	100 m	Two bare ends	TSX CSA 100
6	RS485 double shielded twisted pair trunk cable	200 m	Two bare ends	TSX CSA 200
6	RS485 double shielded twisted pair trunk cable	500 m	Two bare ends	TSX CSA 500
7	Modbus RS485 cable	0.3 m	Two RJ45 male connectors	VW3 A8 306 R03
7	Modbus RS485 cable	1 m	Two RJ45 male connectors	VW3 A8 306 R10
7	Modbus RS485 cable	3 m	Two RJ45 male connectors	VW3 A8 306 R30
-	Modbus RS485 cable	3 m	• One RJ45 male connector • One fifteen-pin SUB-D male connector	VW3 A8 306
4	Modbus RS485 cable	0.3 m	• One RJ45 male connector • One mini-DIN connector	TWD XCA RJ003
4	Modbus RS485 cable	1 m	• One RJ45 male connector • One mini-DIN connector	TWD XCA RJ010
4	Modbus RS485 cable	3 m	• One RJ45 male connector • One mini-DIN connector	TWD XCA RJ030
5	Modbus RS485 cable	3 m	• One RJ45 male connector • One bare end	VW3 A8 306 D30
9	Modbus RS485 cable	3 m	• One miniature connector • One 15-pin SUB-D connector	TSX SCP CM 4630
11	RS485 cable for Magelis XBT display and terminal	2.5 m	• One RJ45 male connector • One 25-pin SUB-D female connector **Note:** This cable is not compatible with BMX NOM 0200 module	XBT-Z938
-	RS485 cable for devices that are powered via the serial link	3 m	Two RJ45 male connectors **Note:** This cable is not compatible with BMX NOM 0200 module.	XBT-Z9980
-	Four-wire RS232 cable for Data Terminal Equipment (DTE)	3 m	• One RJ45 male connector • One nine-pin SUB-D female connector	TCS MCN 3M4F3C2

Figure Reference	Designation	Length	Characteristics	Product reference
-	Four-wire RS232 cable for Data Circuit-terminating Equipment (DCE)	3 m	• One RJ45 male connector • One nine-pin SUB-D male connector	TCS MCN 3M4M3S2
-	Seven-wire RS232 cable for Data Circuit-terminating Equipment (DCE)	3 m	• One RJ45 male connector • One 9-pin SUB-D male connector	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	• Ten RJ45 connectors • One screw terminal block	LU9 GC3
2	T-junction box	• Two RJ45 connectors • On-board 0.3 m cable with RJ45 connector at end	VW3 A8 306 TF03
2	T-junction box	• Two RJ45 connectors • On-board 1 m cable with RJ45 connector at end	VW3 A8 306 TF10
-	Passive T-junction box	• Three screw terminal blocks • RC line end adapter	TSX SCA 50
3	Passive 2-channel subscriber socket	• Two fifteen-pin SUB-D female connectors • Two screw terminal blocks • RC line end adapter	TSX SCA 62
4	Isolated RS485 T-junction box	• One RJ45 connectors • One screw terminal block	TWD XCA ISO
-	T-junction box	Three RJ45 connectors	TWD XCA T3RJ
-	Modbus / Bluetooth adapter	• 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 adpter for ATV speed drives	TWD XCA T3RJ

Figure Reference	Designation	Characteristics	Product reference
5	RS232C/RS485 line adapter without modem signals	19.2kbit/s	XGS Z24
12	Line terminator for RJ45 connector	• Resistance of 120 Ω • Capacity of 1 nF	VW3 A8 306 RC
-	Line terminator for screw terminal block	• Resistance of 120 Ω • Capacity of 1 nF	VW3 A8 306 DRC
-	Adapter for non-standard devices	• Two 25-pin SUB-D male connectors	XBT ZG999
-	Adapter for non-standard devices	• One 25-pin SUB-D male connector • One nine-pin SUB-D male connector	XBT ZG909
-	Adapter for data terminal equipment	• One nine-pin SUB-D male connector • One 25-pin SUB-D female connector	TSX CTC 07
-	Adapter for data terminal equipment	• One nine-pin SUB-D male connector • One 25-pin SUB-D male connector	TSX CTC 10
-	Adapter for Data Circuit-terminating Equipment (DCE)	• One nine-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 III
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 Unity Pro software.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	Installation Methodology	65
5	Modbus Serial Communication for BMX P34 1000/2000/2010/20102/2020 Processors	69
6	Character Mode Communication for BMX P34 1000/2000/2010/20102/2020 Processors	103
7	Modbus Serial Communication for BMX NOM 0200	131
8	Character Mode Communication for BMX NOM 0200	177
9	BMX NOM 0200 Module Diagnostics	211
10	Language Objects of Modbus and Character Mode Communications	217
11	Dynamic Protocol Switching	251

Chapter 4
Installation Methodology

Introduction to the Installation Phase

Introduction

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

- in offline mode
- in online mode

If you do not have a processor to which you can connect, Unity Pro 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

Phase	Description	Mode
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
Legend: (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	
Legend: (1) These phases may also be performed online.		

Configuration of Processor and Module

The configuration parameters may only be accessed from the Unity Pro software.

Technical Documentation Creation

Unity Pro allows to create a **project technical documentation** *(see Unity Pro, 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 5
Modbus Serial Communication for BMX P34 1000/2000/2010/20102/2020 Processors

Subject of this Chapter

This chapter presents the software implementation process for Modbus Serial communication for BMX P34 1000/2000/2010/20102/2020 processors.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
5.1	Generalities	70
5.2	Modbus Serial Communication Configuration	77
5.3	Modbus Serial Communication Programming	89
5.4	Debugging Modbus Serial Communication	100

Section 5.1
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	71
Performance	72
How to Access the Serial Link Parameters	74

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:

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

Exchange times are similar on the BMX P34 2020 and BMX P34 2000/2010/20102 processors, and for the BMX P34 1000, the exchange time is 10% lower than ones.

Exchange Time for 100 Words

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

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

Exchange times are similar on the BMX P34 2020 and BMX P34 2000/2010/20102 processors, and for the BMX P34 1000, the exchange time is 10% lower than ones.

Measurement Accuracy

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

How to Access the Serial Link Parameters

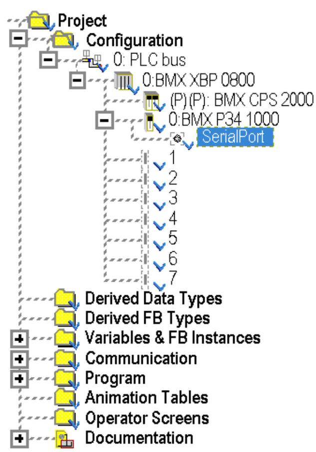
At a Glance

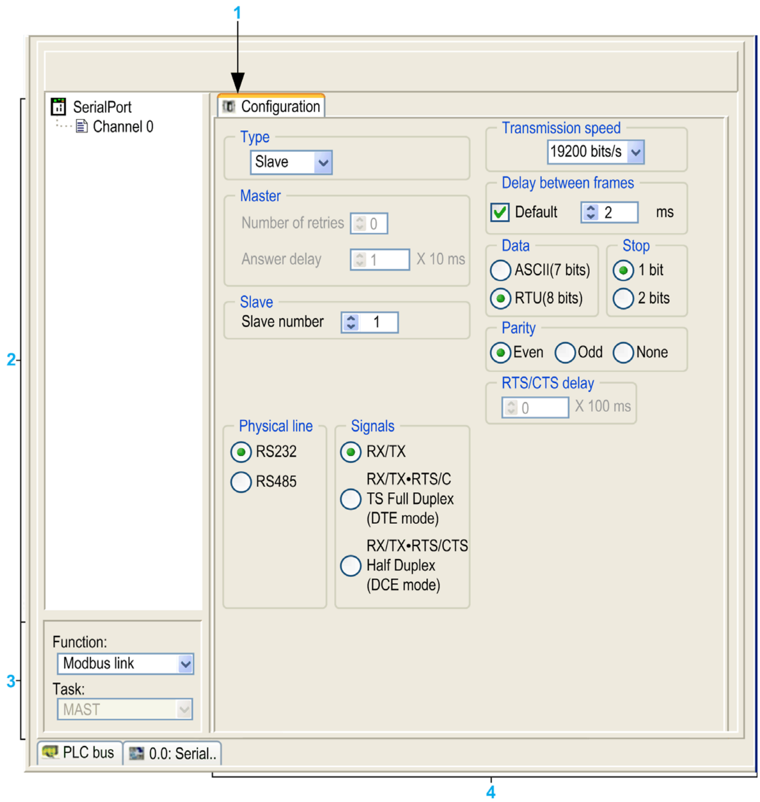
The following pages explain how to access the serial port configuration screen for the following processors as well as the general elements of Modbus and Character Mode link configuration and debug screens:

- 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	In the project browser, open the following directory:*Station\Configuration\0: PLC bus\0: rack reference\0: processor reference\SerialPort*. **Result**: The following screen appears: 

Step	Action
2	Double-click on the Serial Port sub-directory. **Result**: The confiuration screen appears: 

Description of Configuration Screen

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

Address	Element	Function
1	Tabs	The tab in the foreground indicates the current mode. Each mode can be selected using the corresponding tab. The available modes are: • Configuration • Debug screen (accessible in online mode only)

2	Channel zone	Enables you to:
		• 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:
		  • "Description", which gives the characteristics of the device.
		  • **"I/O Objects",** *(see Unity Pro, Operating Modes)* which is used to presymbolize the input/output objects.
		• Display the following tabs by clicking on the channel:
		  • Configuration
		  • Debugging
		• Display the channel name and symbol defined by the user using the variables editor.

3	General parameters zone	This enables you to choose the general parameters associated with the channel: • 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	In configuration mode, this zone is used to configure the channel parameters. In debug mode, it is used to debug the communication channel.

Section 5.2
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	78
Accessible Modbus Functions	81
Default Values for Modbus Serial Communication Parameters	82
Application-linked Modbus Parameters	83
Transmission-linked Modbus Parameters	85
Signal and Physical Line Parameters in Modbus	87

Modbus Serial Communication Configuration Screen

General

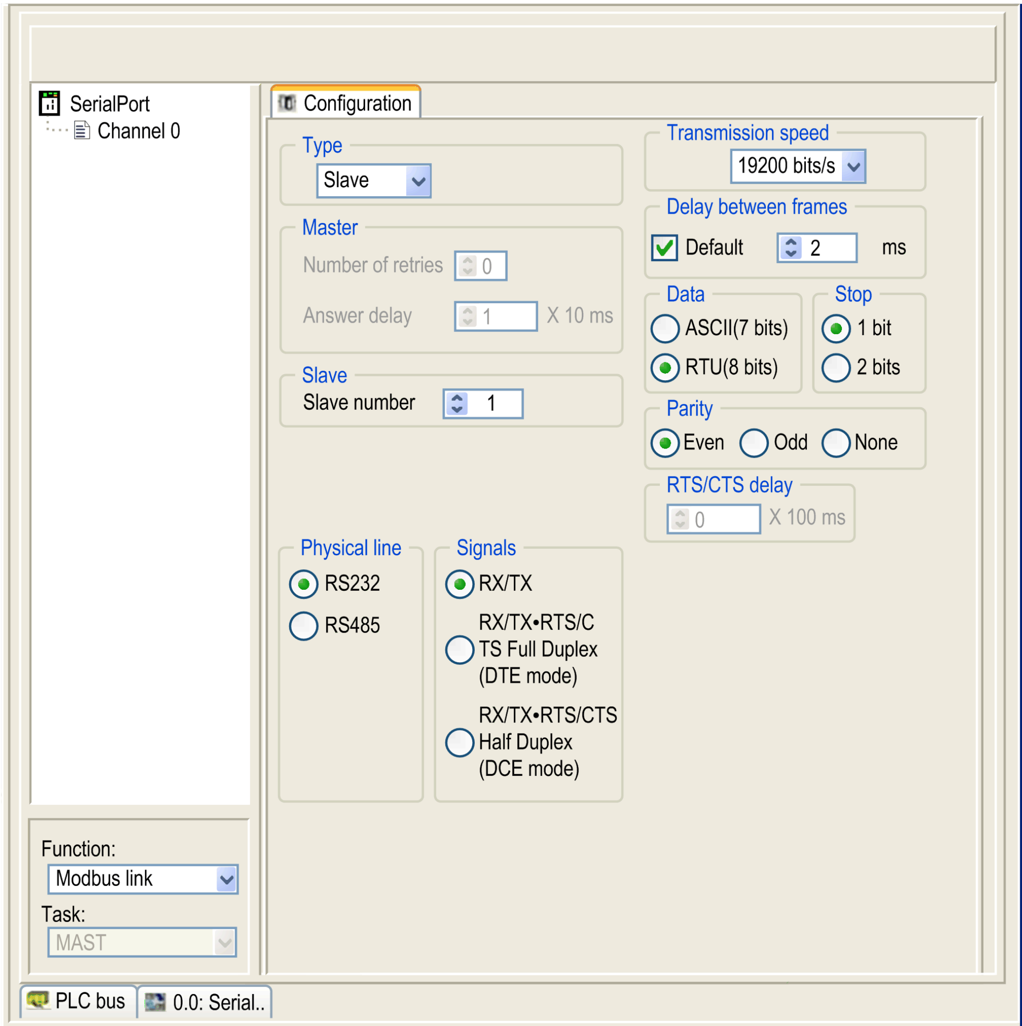
The following pages provide an introduction to the configuration screen for Modbus serial communication.

Access to the Configuration Screen

To access the Modbus serial communication configuration screen, open the Serial Port directory in the project browser *(see page 74)*.

Modbus Serial Communication Configuration Screen

The figure below shows the default configuration screen for Modbus serial communication:



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:

Key	Element	Comment
1	Application parameters (*see page 83*)	These parameters are accessible via three zones: ● Type, ● Master, ● Slave.
2	Transmission parameters (*see page 85*)	These parameters are accessible via five zones: ● Transmission speed, ● Delay between frames, ● Data, ● Stop bits, ● Parity.
3	Signal and physical line parameters (*see page 87*)	These parameters are accessible via three zones: ● Physical line, ● Signals, ● RTS/CTS delay.

NOTE: When configuring Modbus Serial communication in Master mode, the Slave zone is grayed out and cannot be modified and vice-versa.

Accessible Modbus Functions

At a Glance

Function accessibility for configuration of the serial link of the following processors using Modbus Serial, depends on the physical link being used:

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

Accessible Functions

The table below shows the different functions configurable according to the type of serial link used:

Function	RS 485 Link	RS 232 Link
Master number of retries	X	X
Master response time	X	X
Slave number	X	X
Transmission speed	X	X
Delay between frames	X	X
Data	• ASCII (7 bits) • RTU (8 bits)	• ASCII (7 bits) • RTU (8 bits)
Stop	• 1 bit • 2 bits	• 1 bit • 2 bits
Parity	• Odd • Even • None	• Odd • Even • None
RX/TX signals	X	X
RTS/CTS signals	-	X
RTS/CTS delay	-	X

X Accessible Function
- Inaccessible Function

Default Values for Modbus Serial Communication Parameters

At a Glance

All Modbus Serial communication parameters have default values.

Default Values

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

Configuration parameter	Value
Mode	Slave
Physical Line	RS232
Slave number	1
Delay between frames	2 ms
Transmission speed	19200 bits/s
Parity	Even
Data Bits	RTU (8 bits)
Stop bits	1 bit

Application-linked Modbus Parameters

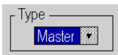
At a Glance

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

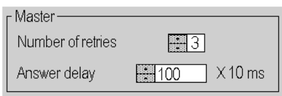
This configuration zone appears on the screen as shown below:



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:

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.

Transmission-linked Modbus Parameters

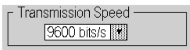
At a Glance

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

These parameters are accessible from five zones:
- The Transmission Speed zone,
- The Delay Between Characters zone,
- The Data zone,
- The Stop zone,
- The Parity zone.

The Transmission Speed Zone

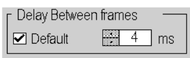
This configuration zone appears on the screen as shown below:



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:



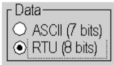
The Delay Between Frames is the minimum time separating two frames on reception. This delay is managed when the PLC (master or slave) is receiving messages.

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:

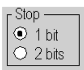


This zone allows you to enter the type of coding used to communicate using Modbus Serial. This field is set according to the other devices connected on the bus. There are two configurable modes:

- 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

Signal and Physical Line Parameters in Modbus

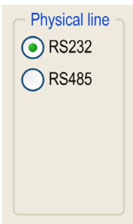
At a Glance

After configuring the communication channel, you need to enter the signal and physical line parameters.

These parameters are accessible via three zones:
- The Physical Line zone,
- The Signals zone,
- The RTS/CTS Delay zone.

The Physical Line Zone

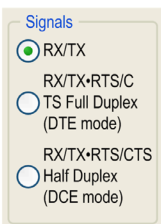
This configuration zone appears on the screen as shown below:



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 RS 232 line,
- The RS 485 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 RS 232 physical line:

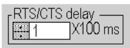
- RX/TX
- RX/TX + RTS/CTS DTEmode
- RX/TX + RTS/CTS DCEmode

If the RS 485 is configured, the entire zone will be grayed out and the default value is RX/TX.

NOTE: Only RX/TX and RX/TX + RTS/CTS signals are available when configuring the serial port for BMX P34 1000/2000/2010/20102/2020 processors.

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.

Section 5.3
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	90
Services Supported by a Modbus Link Slave Processor	98

Services Supported by a Modbus Link Master Processor

At a Glance

When used as the master processor in a Modbus link, the following processors support several services via the READ_VAR, WRITE_VAR and DATA_EXCH communication functions.

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

Data Exchanges

Reading or writing of variables are carried out by adressing 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	READ_VAR
Read words	16#03 or 16#04	READ_VAR
Write bits	16#0F	WRITE_VAR
Write words	16#10	WRITE_VAR
Other request	all	DATA_EXCH

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.

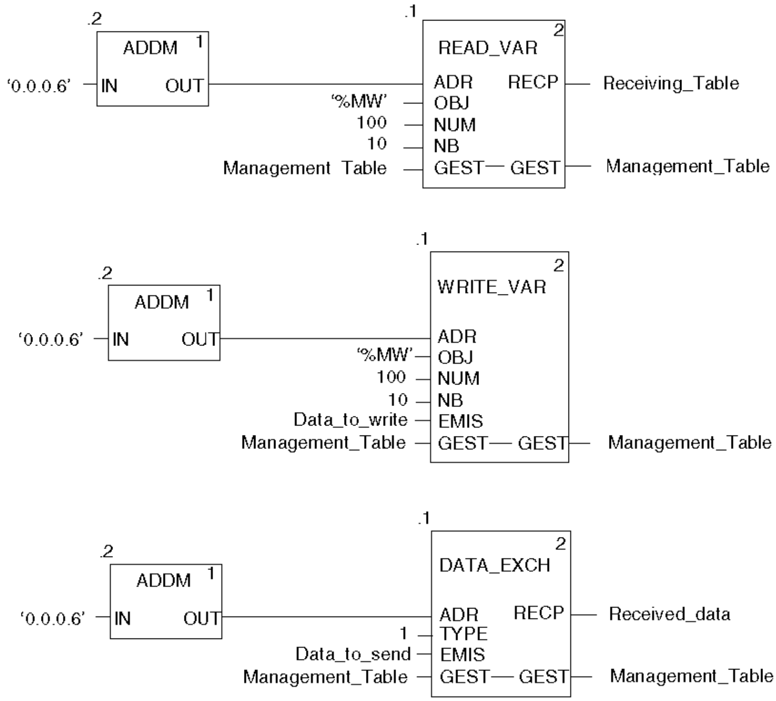
READ_VAR, WRITE_VAR and DATA_EXCH Communication Functions

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

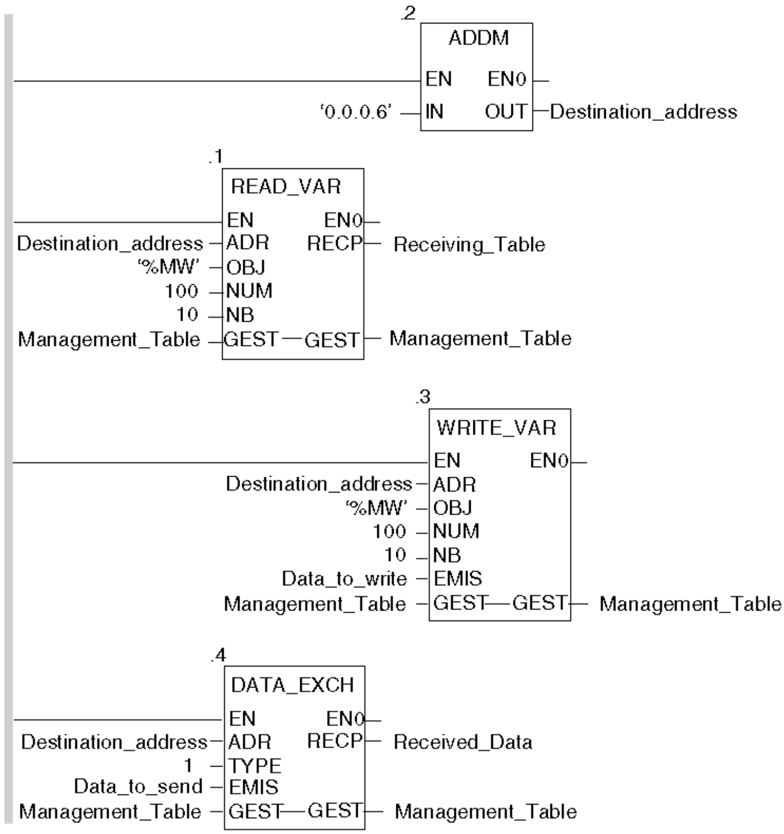
Programming Example in FBD

The diagram below represents an example of programming of the `READ_VAR`, `WRITE_VAR` and `DATA_EXCH` communication functions in the FBD language:



Programming Example in Ladder

The diagram below represents an example of programming of the `READ_VAR`, `WRITE_VAR` and `DATA_EXCH` communication functions in the Ladder language:



Programming Example in ST

The lines of code below represent an example of programming of the `READ_VAR`, `WRITE_VAR` and `DATA_EXCH` communication functions in the ST language:

```
READ_VAR(ADDM('0.0.0.6'), 'MW', 100, 10, Management_Table,
Receiving_Table);
```

```
WRITE_VAR(ADDM('0.0.0.6'), '%MW', 100, 10, Data_to_write,
Management_Table);
```

```
DATA_EXCH(ADDM('0.0.0.6'), 1, Data_to_send, Management_Table,
Received_data);
```

Cancelling an Exchange

An exchange executed by the READ_VAR, WRITE_VAR and DATA_EXCH functions can be cancelled with either ways of programming, which are both presented in ST language below:

- Using the CANCEL function:

```
IF (%MW40.0) THEN
 %MW200:=SHR(%MW40,8;)
 CANCEL(%MW200,%MW185);
END_IF;
```

  %MW40 is the GEST parameter (management table). %MW40.0 corresponds to the activity bit of the READ_VAR function and is set to 1 when the communication function is active. If this bit is set to 1, the program carries out the following instructions:
  - Moves the %MW40 bits one byte (8 bits) to the right and loads the byte corresponding to the communication's exchange number into the %MW200 word,
  - Cancels the exchange whose exchange number is contained within the %MW200 word using the CANCEL function.

- Using the communication function cancel bit:

```
IF (%MW40.0) THEN
 SET(%MW40.1);
 READ_VAR(ADDM('0.0.0.6'), '%MW', 100, 10, %MW40:4, %MW10:10);
END_IF;
```

  %MW40 is the GEST parameter (management table). %MW40.0 corresponds to the activity bit of the READ_VAR function and is set to 1 when the communication function is active. If this bit is set to 1, the program sets the %MW40.1 bit, the function cancel bit, to 1. This stops communication of the READ_VAR function.

NOTE: When using the communication function cancel bit contained in the function exchange management word (%MW40 in this example), the function (READ_VAR in this example) must be called in order to activate the cancellation of the exchange.

NOTE: When using the communication function cancel bit, it is possible to cancel a communication from an animation table. This can be done by simply setting the function cancel bit to 1 (%MW40.1 in this example) and then start again the communication function.

NOTE: This example of programming concerns the READ_VAR function, but is equally applicable to the WRITE_VAR as well as the DATA_EXCH functions.

NOTE: The CANCEL function uses a report word for the CANCEL function (%MW185 in this example).

Description of ADDM Function Parameters

The following table outlines the various parameters for the ADDM function:

Parameter	Type	Description
IN	STRING	Address of device on bus or serial link. The syntax of the address is of the 'r.m.c.node' type. The address is made up of the following parameters: ● **r**: Rack number of the processor, always = 0 ● **m**: Slot number of the processor within the rack, always = 0 ● **c**: Channel number, always = 0 as the serial link of a processor is always channel 0 ● **node**: Number of slave to which the request is being sent
OUT	ARRAY [0..7] OF INT	Array representing the address of a device. This parameter can be used as an input parameter for several communication functions.

Description of READ_VAR Function Parameters

The following table outlines the various parameters for the READ_VAR function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the destination entity given by the OUT parameter of the ADDM function.
OBJ	STRING	Type of object to be read. The available types are as follows: ● %M: internal bit ● %MW: internal word ● %I: external input bit ● %IW: external input word
NUM	DINT	Address of first object to be read.
NB	INT	Number of consecutive objects to be read.

Parameter	Type	Description

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		● Rank 1 word: A word managed by the system and consisting of two bytes:
		  ● Most significant byte: Exchange number,
		  ● Least significant byte: Activity bit (rank 0) and cancel bit (rank 1).
		● Rank 2 word: A word managed by the system and consisting of two bytes:
		  ● Most significant byte: Operation report,
		  ● Least significant byte: Communication report.
		● Rank 3 word: A word managed by the user which defines the maximum response time using a time base of 100 ms.
		● Rank 4 word: A word managed by the system which defines the length of the exchange.

RECP	ARRAY [n..m] OF INT	Word table containing the value of the objects read.

Description of `WRITE_VAR` Function Parameters

The following table outlines the various parameters of the `WRITE_VAR` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the destination entity given by the OUT parameter of the ADDM function.
OBJ	STRING	Type of object to be written. The available types are as follows: ● `%M`: internal bit ● `%MW`: internal word **Note:** `WRITE_VAR` cannot be used for `%I` and `%IW` variables.
NUM	DINT	Address of first object to be written.
NB	INT	Number of consecutive objects to be written.
EMIS	ARRAY [n..m] OF INT	Word table containing the value of the objects to be written.

Parameter	Type	Description

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		● Rank 1 word: A word managed by the system and consisting of two bytes:
		  ● Most significant byte: Exchange number,
		  ● Least significant byte: Activity bit (rank 0) and cancel bit (rank 1).
		● Rank 2 word: A word managed by the system and consisting of two bytes:
		  ● Most significant byte: Operation report,
		  ● Least significant byte: Communication report.
		● Rank 3 word: A word managed by the user which defines the maximum response time using a time base of 100 ms.
		● Rank 4 word: A word managed by the system which defines the length of the exchange.

Description of `DATA_EXCH` Function Parameters

The following table outlines the various parameters of the `DATA_EXCH` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the destination entity given by the OUT parameter of the ADDM function.
TYPE	INT	For Modicon M340 PLCs, the only possible value is: **1**: Transmission of an **EMIS** array, then the PLC waits for the reception of a **RECP** array.
EMIS	ARRAY [n..m] OF INT	Integers table to be sent to the destination device of the request. **Note:** It is imperative that the length of the data to be sent (in bytes) be assigned to the fourth word of the management table before launching the function, in order for this to be correctly executed.

Parameter	Type	Description

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		• Rank 1 word: A word managed by the system and consisting of two bytes:
		  • Most significant byte: Exchange number,
		  • Least significant byte: Activity bit (rank 0) and cancel bit (rank 1).
		• Rank 2 word: A word managed by the system and consisting of two bytes:
		  • Most significant byte: Operation report,
		  • Least significant byte: Communication report.
		• Rank 3 word: A word managed by the user which defines the maximum response time using a time base of 100 ms.
		• Rank 4 word: A word managed by the system which defines the length of the exchange.

RECP	ARRAY [n..m] OF INT	Integers table containing the data received. **Note:** The size of the data received (in bytes) is written automatically by the system in the fourth word of the management table.

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 5.4
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. *(see page 78)*
2	Select the "Debug" tab on the screen that appears.

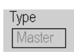
Description of the Debug Screen

The debug screen is divided into two zones:

- The Type zone,
- The Counters zone.

The Type Zone

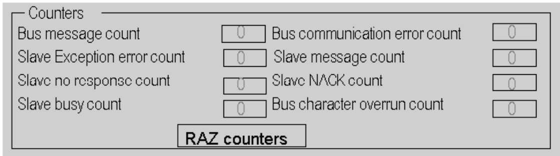
This zone looks like this:



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 6
Character Mode Communication for BMX P34 1000/2000/2010/20102/2020 Processors

Subject of this Section

This chapter presents the software implementation of communication using Character Mode for BMX P34 1000/2000/2010/20102/2020 processors.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Generalities	104
6.2	Character Mode Communication Configuration	108
6.3	Character Mode Communication Programming	121
6.4	Debugging Character Mode communication	128

Section 6.1
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	105
Performance	106

About Character Mode Communication

Introduction

Communication in Character Mode enables dialog and communication functions to be carried out between the PLCs and 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 Times for 80 Characters

The table below shows the exchange times for the transmission of 80 characters in Character Mode on a BMX P34 2020 processor:

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

19200	20	60
19200	50	95
19200	100	100
19200	255	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.

Measurement Accuracy

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

Section 6.2
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	109
Accessible Functions in Character Mode	112
Default Values for Character Mode Communication Parameters	113
Message End Detection Parameters in Character Mode	114
Transmission Parameters in Character Mode	116
Signal and Physical Line Parameters in Character Mode	118

Character Mode Communication Configuration Screen

General

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

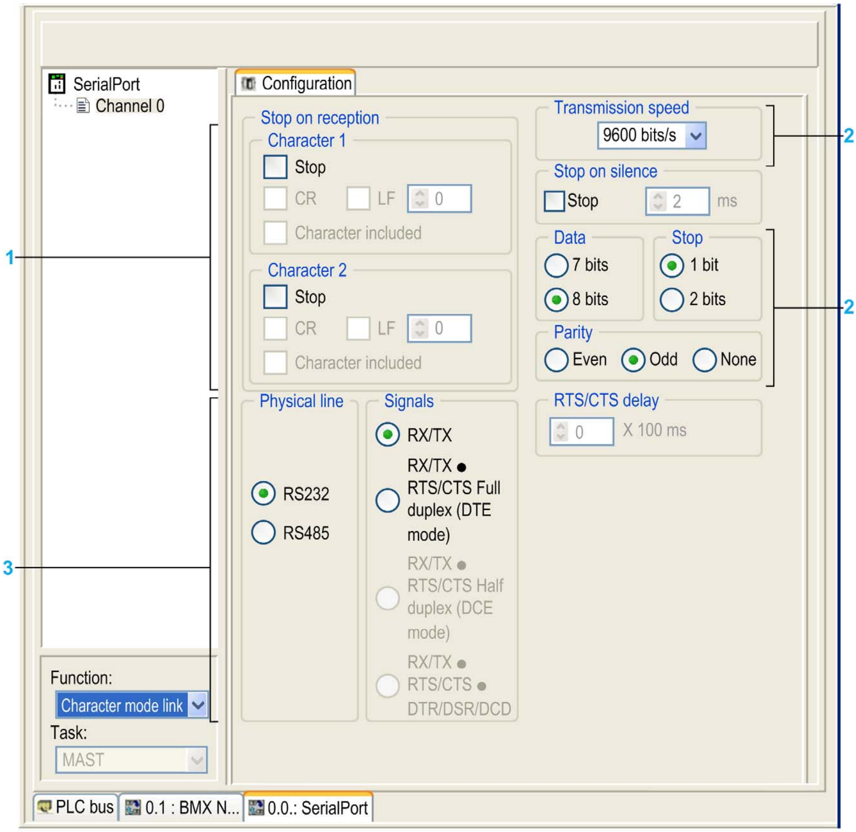
Accessing the Configuration Screen

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 *(see page 74)*.
2	Select the CHARACTER MODE LINK function on the screen that appears.

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:

Key	Element	Comment
1	Message end detection parameters *(see page 114)*	These parameters are accessible via two zones: ● Stop on reception, ● Stop on silence.
2	Transmission parameters *(see page 116)*	These parameters are accessible via four zones: ● Transmission speed, ● Data, ● Stop bits, ● Parity.
3	Signal and physical line parameters *(see page 118)*	These parameters are accessible via three zones: ● Physical line, ● Signals, ● RTS/CTS delay.

Accessible Functions in Character Mode

At a Glance

Function accessibility for configuration of the serial link for the following processors using Character Mode protocol depends on the physical link being used:

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

Accessible Functions

The table below shows the different functions configurable according to the type of serial link used:

Function	RS 485 Link	RS 232 Link
Transmission speed	X	X
Data	• 7 bits • 8 bits	• 7 bits • 8 bits
Stop	• 1 bit • 2 bits	• 1 bit • 2 bits
Parity	• Odd • Even • None	• Odd • Even • None
Stop on Reception	X	X
Stop on Silence	X	X
RX/TX Signals	X	X
RTS/CTS Signals	-	X
RTS/CTS delay	-	X

X Accessible Function
- Inaccessible Function

Default Values for Character Mode Communication Parameters

At a Glance

All Character Mode communication parameters have default values.

Default Values

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

Configuration parameter	Value
Physical Line	RS 232
Transmission speed	9600 bits/s
Parity	Odd
Data Bits	8 bits
Stop bits	1 bit

Message End Detection Parameters in Character Mode

At a Glance

After configuring the communication channel, you need to enter the message end detection parameters.

These parameters are accessible via two zones:

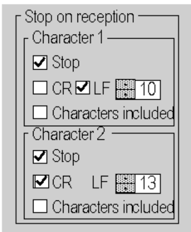
- The Stop on Reception Zone: stop on reception of a special character.
- The Stop on Silence Zone: stop on silence.

Conditions of Use

Selecting Stop on Silence means that Stop on Reception is deselected and vice versa.

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 CR or LF 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 window below, the end of reception of a message is detected by an LF or CR 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.

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

Transmission Parameters in Character Mode

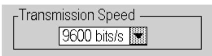
At a Glance

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

These parameters are accessible via four zones:
- The Transmission Speed zone,
- The Data zone,
- The Stop zone,
- The Parity zone.

The Transmission Speed Zone

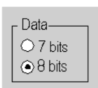
This configuration zone appears on the screen as shown below:



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:

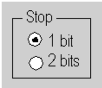


In this zone, you can specify the size of the data being exchanged on the link. The available values are:
- 7 bits
- 8 bits

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

Signal and Physical Line Parameters in Character Mode

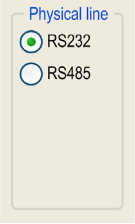
At a Glance

After configuring the communication channel, you need to enter the signal and physical line parameters.

These parameters are accessible via three zones:
- The Physical Line zone
- The Signals zone
- The RTS/CTS Delay zone

The Physical Line Zone

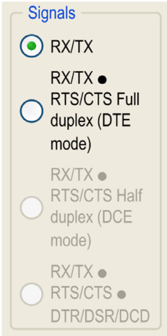
This configuration zone appears on the screen as shown below:



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 RS 232 line
- The RS 485 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 RS 232 physical line:

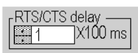
- RX/TX
- RX/TX + RTS/CTS Full Duplex (DTE mode)

If the RS 485 is configured, the entire zone is grayed out and the default value is RX/TX.

NOTE: Only RX/TX and RX/TX + RTS/CTS signals are available when configuring the serial port for BMX P34 1000/2000/2010/20102/2020 processors.

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: 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.

Section 6.3
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` *(see Unity Pro, Communication, Block Library)*: read a byte array of a maximum of 1,024 bytes.

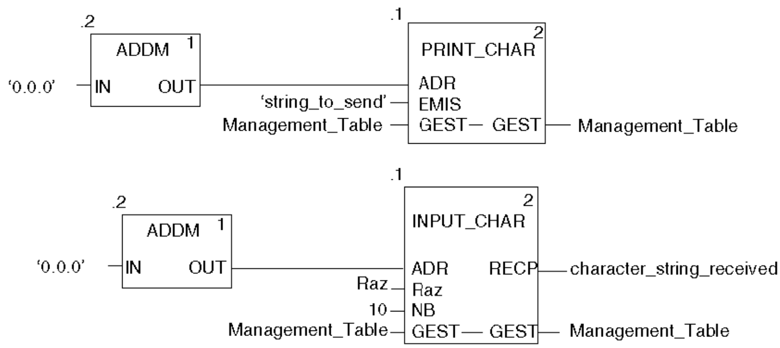
The Modicon M340 PLC serial port is full duplex, so a `PRINT_CHAR` function can be sent even when an `INPUT_CHAR` function has been sent and is still pending.

NOTE: For `INPUT_CHAR` function, a configured time-out 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 time-out.

NOTE: Contrary to the NOM0200 in RS485 Link ,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 make a `INPUT_CHAR` with the Reset buffer activated and cancel this EF before the Timeout.

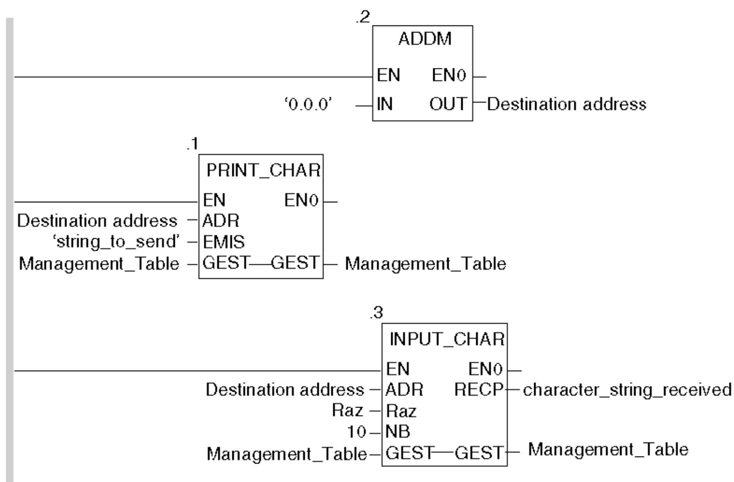
Example of Programming in FBD

The diagram below represents an example of programming of the `PRINT_CHAR` and `INPUT_CHAR` communication functions in FBD language:



Example of Programming in Ladder

The diagram below represents an example of programming of the `PRINT_CHAR` and `INPUT_CHAR` communication functions in Ladder language:



Example of Programming in ST

The lines of code below represent an example of programming of the `PRINT_CHAR` and `INPUT_CHAR` communication functions in ST language:

```
PRINT_CHAR(ADDM('0.0.0'), 'string_to_send', Management_Table);
INPUT_CHAR(ADDM('0.0.0'), reset_integer_to_0, 10, Management_Table,
character_string_received);
```

Feature of the INPUT_CHAR Function

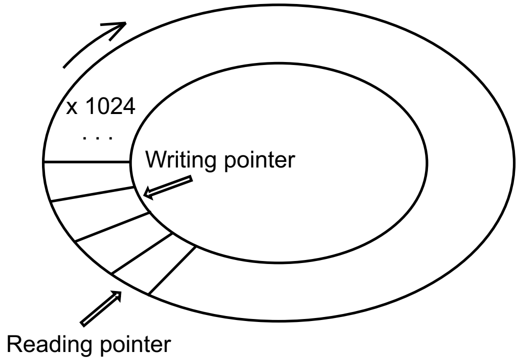
If the Reset input parameter is set to 1, the buffer is first reset then the processor is waiting for the reception of data. Using this feature is advised in order to start properly a reception by removing old data that can remain in the buffer.

Internal Mechanism of the CPU

The data received is 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.

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

The below figure represents this mechanism:



Cancelling an Exchange

There are two ways of programming that enable an exchange executed by the `PRINT_CHAR` and `INPUT_CHAR` functions to be cancelled. These are both presented in ST language below:

- Using the CANCEL function:
  ```
  IF (%MW40.0) THEN
   %MW200:=SHR(%MW40,8;)
   CANCEL(%MW200,%MW185);
  END_IF;
  ```
  `%MW40` is the `GEST` parameter (management table). `%MW40.0` corresponds to the activity bit of the `PRINT_CHAR` function and is set to 1 when the communication function is active. If this bit is set to 1, the program carries out the following instructions:
  - Moves the `%MW40` bits one byte (8 bits) to the right and loads the byte corresponding to the communication's exchange number into the `%MW200` word.
  - Cancels the exchange whose exchange number is contained within the `%MW200` word using the CANCEL function.

- Using the communication function's cancel bit:
  ```
  IF (%MW40.0) THEN
   SET(%MW40.1);
   PRINT_CHAR(ADDM('0.0.0'), 'string_to_send', %MW40:4);
  END_IF;
  ```
  `%MW40` is the `GEST` parameter (management table). `%MW40.0` corresponds to the activity bit of the `PRINT_CHAR` function and is set to 1 when the communication function is active. If this bit is set to 1, the program sets the `%MW40.1` bit, the function cancel bit, to 1. This stops communication of the `PRINT_CHAR` function.

NOTE: When using the communication function cancel bit, the function must be called in order to enable the cancel bit contained in the function exchange management word (`%MW40` in this example).

NOTE: When using the communication function cancel bit, it is possible to cancel a communication from an animation table. This can be done by simply setting the function cancel bit to 1 (`%MW40.1` in this example).

NOTE: This example of programming concerns the `PRINT_CHAR` function, but is equally applicable to the `INPUT_CHAR` function.

NOTE: The CANCEL function uses a report word for the CANCEL function (`%MW185` in this example).

Description of ADDM Function Parameters

The following table outlines the various parameters for the `ADDM` function:

Parameter	Type	Description
IN	STRING	Address of device on bus or serial link. The syntax of the address is of the 'r.m.c.node' type. The address is made up of the following parameters: ● **r**: rack number of the destination system, always = 0. ● **m**: slot number of the destination system within the rack, always = 0. ● **c**: channel number, always = 0 as the serial link of a remote system is always channel 0. ● **node**: optional field that may be `SYS` or empty.
OUT	ARRAY [0..7] OF INT	Table showing the address of a device. This parameter can be used as an input parameter for several communication functions.

Description of PRINT_CHAR Function Parameters

The following table outlines the various parameters of the `PRINT_CHAR` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the message receiving character mode channel given by the OUT parameter of the ADDM function.
EMIS	STRING	Character string to be sent.

Parameter	Type	Description

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		● Rank 1 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: exchange number
		  ● Least significant byte: activity bit (rank 0) and cancel bit (rank 1)
		● Rank 2 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: operation report
		  ● Least significant byte: communication report
		● Rank 3 word: a word managed by the user, which defines the maximum response time using a time base of 100 ms.
		● Rank 4 word: a word managed by the user which defines the length of the exchange.
		  ● If this parameter length is set to 0 then the system sends the string entirely.
		  ● If this parameter length is greater than the length of the string then the error 16#0A (Insufficient send buffer size) is returned into the 2nd management word and no character is sent.
		Note: In case of `PRINT_CHAR`, the fourth management word (length to send) must not be written while the activity bit is 1 (i.e. once EF is running). Otherwise `PRINT_CHAR` could be locked with error 0xB.

Description of INPUT_CHAR Function Parameters

The following table outlines the various parameters of the `INPUT_CHAR` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the message receiving character mode channel given by the OUT parameter of the ADDM function.
RAZ	INT	Reset. This parameter is used to reset the receive memory of the coupler: ● Value = 0: no memory reset ● Value = 1: memory reset
NB	INT	Length of character string to be received.

Parameter	Type	Description

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		● Rank 1 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: exchange number
		  ● Least significant byte: activity bit (rank 0) , cancel bit (rank 1) and immediate aknowledge bit (rank 2)
		● Rank 2 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: operation report
		  ● Least significant byte: communication report
		● Rank 3 word: a word managed by the user which defines the maximum response time using a time base of 100 ms.
		● Rank 4 word: a word managed by the system which defines the length of the exchange.

RECP	STRING	Character string received. This string is saved in a character string.

Section 6.4
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:

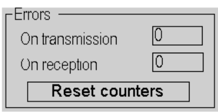
Step	Action
1	Access the configuration screen for Character Mode communication. *(see page 109)*
2	Select the "Debug" tab on the screen that appears.

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:

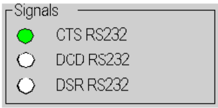


This zone indicates the number of communication interruptions counted by the processor:
- **On transmission**: corresponds to the number of interruptions on transmission (image of `%MW4` word).
- **On reception**: corresponds to the number of interruptions on reception (image of `%MW5` word).

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 7
Modbus Serial Communication for BMX NOM 0200

Subject of this Chapter

This chapter presents the software implementation process for Modbus serial communication for BMX NOM 0200.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	Generalities	132
7.2	Modbus Serial Communication Configuration	139
7.3	Modbus Serial Communication Programming	156
7.4	Debugging Modbus Serial Communication	174

Section 7.1
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	133
Performance	134
How to Access the Serial Link Parameters	136

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 communications port 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 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 NOM 0200 module:

Baud rate of communication in bits per second	Cycle time in ms	Exchange time in ms Modbus Slave is a BMX P34 1000 cyclic
4800	Cyclic	65
4800	10	68
4800	50	100
9600	Cyclic	38
9600	10	47
9600	50	50
19200	Cyclic	29
19200	10	38
19200	50	50
38400	Cyclic	24
38400	10	30
38400	50	50
57600	Cyclic	17
57600	10	20

Baud rate of communication in bits per second	Cycle time in ms	Exchange time in ms Modbus Slave is a BMX P34 1000 cyclic
57600	50	50
115200	Cyclic	17
115200	10	20
115200	50	50

Exchange Time for 100 Words

The table below shows exchange times for 100 words of Modbus communication on a BMX NOM 0200 processor:

Baud rate of communication in bits per second	Cycle time in ms	Exchange time in ms Modbus Slave is a BMX P34 1000 cyclic
4800	Cyclic	560
4800	10	560
4800	50	600
9600	Cyclic	286
9600	10	295
9600	50	300
19200	Cyclic	152
19200	10	160
19200	50	200
38400	Cyclic	86
38400	10	90
38400	50	100
57600	Cyclic	56
57600	10	60
57600	50	100
115200	Cyclic	36
115200	10	40
115200	50	50

Measurement Accuracy

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

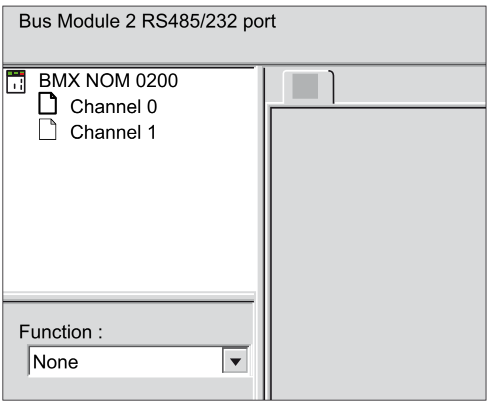
How to Access the Serial Link Parameters

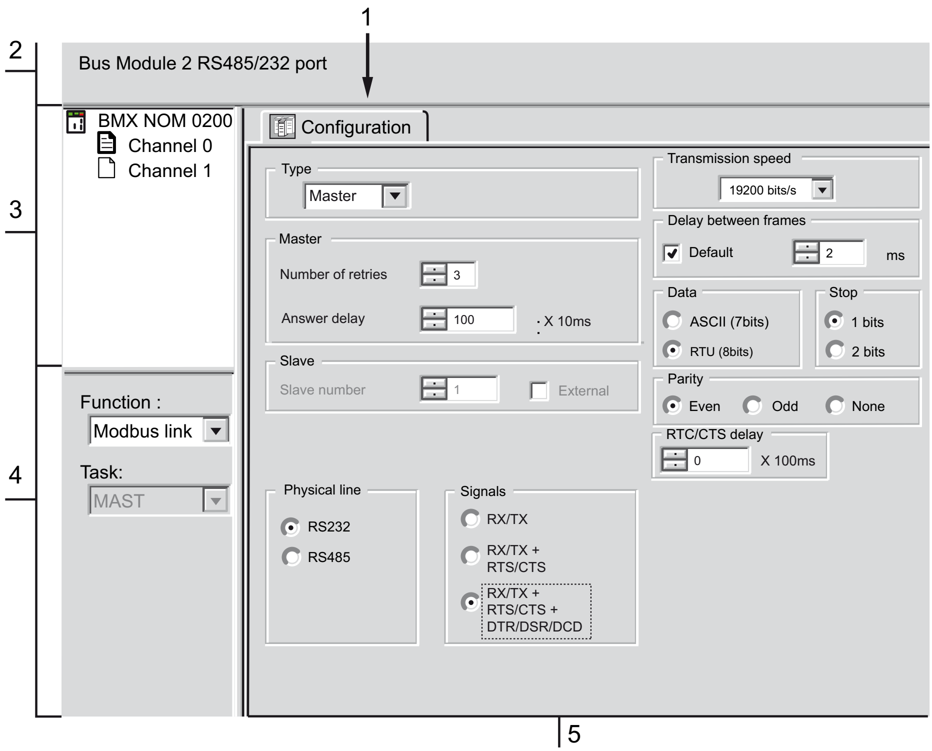
At a Glance

The following pages explain how to access the serial ports configuration screen for the BMX NOM 0200 module as well as the general elements of the Modbus and Character Mode link configuration and debug screens.

How to Access the Serial Link

The table below describes the procedure for accessing the serial link of a BMX NOM 0200 module:

Step	Action
1	Open the hardware configuration editor.
2	Double-click on the BMX NOM 0200 module.
3	Select the channel to configure (Channel 0 or Channel 1). **Result with Channel 0 selected:** 

Step	Action
4	Select the **Modbus link** function. **Result with Channel 0 selected:** 

Description of the Configuration Screen

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

Key	Element	Function
1	Tabs	The tab in the foreground indicates the mode currently in use (**Configuration** in this example). Each mode can be selected using the corresponding tab. The available modes are: ● **Configuration** ● **Debug** (accessible in online mode only) ● **Diagnostic** (accessible in online mode only)
2	Module Zone	Displays module reference and module LEDs status in online mode.

3	Channel zone	Enables you to:
		● Display the following tabs by clicking on **BMX NOM 0200**:
		  ● "Overview", which gives the characteristics of the device.
		  ● **"I/O Objects"** *(see Unity Pro, Operating Modes)*, which is used to presymbolize the input/output objects.
		  ● "Fault", which shows the detected device faults (in online mode).
		● Display the following tabs by clicking on **Channel 0** or **Channel 1**:
		  ● "Configuration"
		  ● "Debugging"
		  ● "Fault"
		● Display the channel name and symbol defined by the user (using the variables editor).

4	Genera parameters zone	This enables you to choose the general parameters associated with the channel: ● **Function**: The available functions are "None", "Modbus link" and “Character mode link". By default, the "None" function is configured. ● **Task**: Defines the master task in which the implicit exchange objects of the channel will be exchanged. This zone is grayed out and cannot be configured.
5	Configuration, debugging or fault zone	In configuration mode, this zone is used to configure the channel parameters. In debug mode, it is used to debug the communication channel. In diagnostic mode, it is used to display current detected errors either at module or at channel level.

Section 7.2
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 in a Modicon M340 Local Rack	140
BMX NOM 0200 Modbus Serial Communication Configuration Screen in X80 Drop	143
Accessible Modbus Functions	146
Default Values for Modbus Serial Communication Parameters	147
Application-linked Modbus Parameters	148
Transmission-linked Modbus Parameters	150
Signal and Physical Line Parameters in Modbus	152
How to Set the BMX NOM0200 MODBUS Slave Address Without Unity Pro?	154

Modbus Serial Communication Configuration Screen in a Modicon M340 Local Rack

General

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

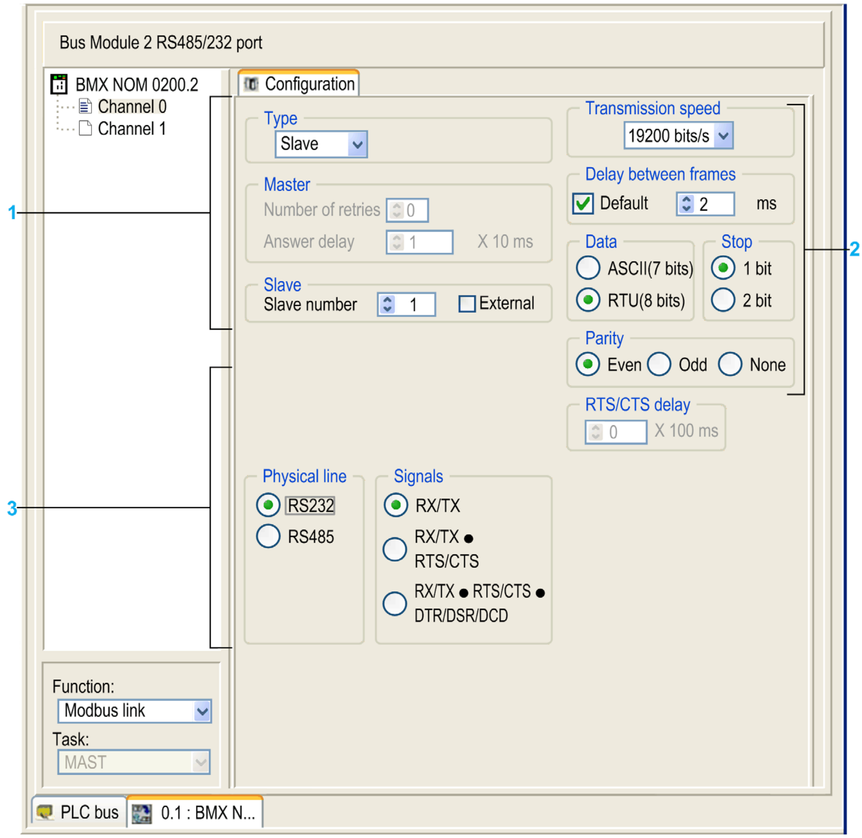
Access to the Configuration Screen

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

Step	Action
1	Open the BMX NOM 0200 sub-directory in the project browser *(see page 136)*.
2	Select the Channel to configure and "Modbus link" function on the screen that appears.

Illustration

The figure below shows the default configuration screen for Modbus serial communication on Channel 0:



Description

These zones are used to configure channel parameters. In the online mode, these zones are accessible. In the offline mode, these zones are 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:

Key	Element	Comment
1	Application parameters (*see page 148*)	These parameters are accessible via three zones: ● Type, ● Master, ● Slave.
2	Transmission parameters (*see page 150*)	These parameters are accessible via five zones: ● Transmission speed, ● Delay between frames, ● Data, ● Stop bits, ● Parity.
3	Signal and physical line parameters (*see page 152*)	These parameters are accessible via three zones: ● Physical line, ● Signals, ● RTS/CTS delay.

NOTE: When configuring Modbus serial communication in Master mode, the "Slave" zone is grayed out and cannot be modified and vice-versa.

BMX NOM 0200 Modbus Serial Communication Configuration Screen in X80 Drop

General

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

NOTE: When the BMX NOM 0200 is in a Quantum Ethernet I/O X80 drop, it must have:
- Product software = 04
- Software software ≥ 1.4

This information is visible on the label on the side of the module.

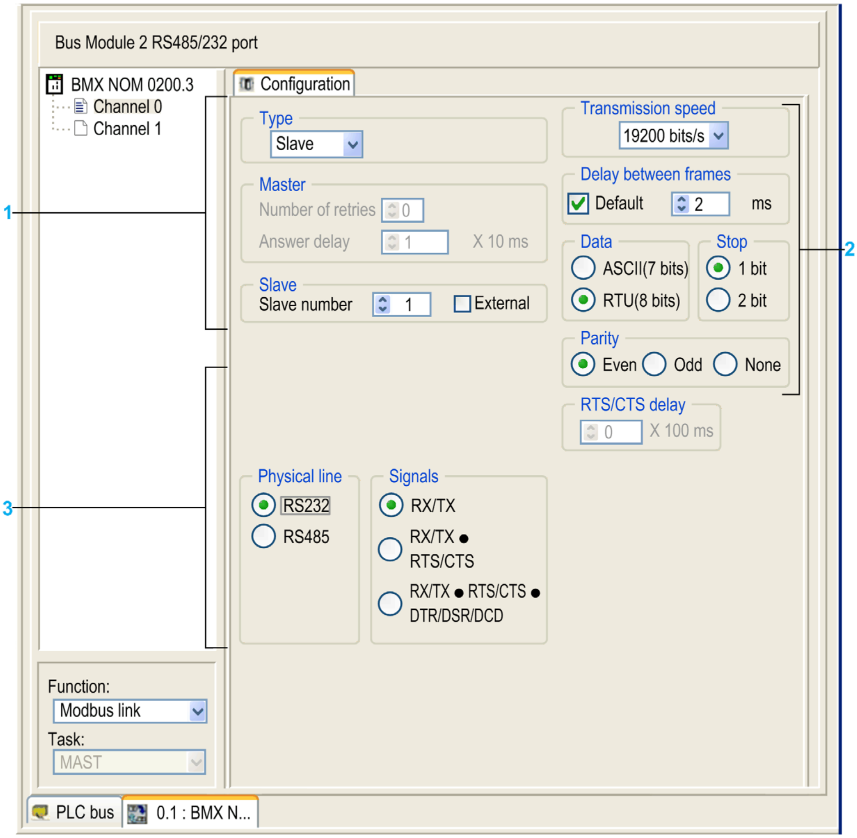
Access to the Configuration Screen

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

Step	Action
1	Open the BMX NOM 0200 sub-directory in the project browser *(see page 136)*.
2	Select the Channel to configure and "Modbus link" function on the screen that appears.

Illustration

The figure below shows the default configuration screen for Modbus serial communication on Channel 0:



Description

These zones are used to configure channel parameters. In the online mode, these zones are accessible. In the offline mode, these zones are 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:

Key	Element	Comment
1	Application parameters (*see page 148*)	These parameters are accessible via three zones: ● Type, ● Master, ● Slave.
2	Transmission parameters (*see page 150*)	These parameters are accessible via five zones: ● Transmission speed, ● Delay between frames, ● Data, ● Stop bits, ● Parity.
3	Signal and physical line parameters (*see page 152*)	These parameters are accessible via three zones: ● Physical line, ● Signals, ● RTS/CTS delay.

NOTE: When configuring Modbus serial communication in Master mode, the **Slave** parameters are grayed out and cannot be modified.

Accessible Modbus Functions

At a Glance

Function accessibility for configuration of the serial link of a BMX NOM 0200 module using Modbus serial depends on the physical link being used.

Accessible Functions

The table below shows the different functions configurable according to the type of serial link used:

Function	RS485 Link (on Channel 0 or Channel 1)	RS232 Link (on Channel 0)
Master number of retries	X	X
Master answer delay	X	X
Slave number	X	X
Transmission speed	X	X
Delay between frames	X	X
Data	• ASCII (7 bits) • RTU (8 bits)	• ASCII (7 bits) • RTU (8 bits)
Stop	• 1 bit • 2 bits	• 1 bit • 2 bits
Parity	• Odd • Even • None	• Odd • Even • None
RX/TX signals	X	X
RTS/CTS signals	-	X
RTS/CTS delay	-	X
DTR/DSR/DCD Signals	-	X
Polarization	-	-

X Accessible Function
- Unaccessible Function

Default Values for Modbus Serial Communication Parameters

At a Glance

All Modbus serial communication parameters have default values.

Default Values

The table below shows the default values for Modbus serial communication parameters on Channel 0 and Channel 1 of the BMX NOM 0200 module:

Configuration parameter	Value
Mode	Slave
Physical Line	RS232
Slave number	1
Delay between frames	2 ms
Transmission speed	19200 bits/s
Parity	Even
Data Bits	RTU (8 bits)
Stop bits	1 bit

Application-linked Modbus Parameters

At a Glance

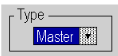
After configuring the communication channel, you need to enter the application parameters.

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:

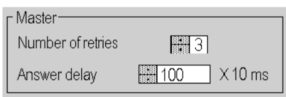


This zone enables you to select the role to be configured for the module in the Modbus serial communication:

- **Master**: When the module is the master.
- **Slave**: When the module 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:

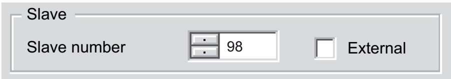
- **Number of retries**: number of connection attempts made by the master before defining the slave as absent.
  The default value is 3.
  Possible values range from 0 to 15.
  A value of 0 indicates no retries by the Master.
- **Answer delay**: the time between the Master’s initial request and a repeated 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 the 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.

NOTE: In broadcast mode, the value configured as Answer Delay is used as Broadcast delay: minimum time between two exchanges in broadcast mode.

The Slave Zone

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



This zone enables you to enter the processor’s slave number:

The default value is 1.

Possible values range from 1 to 247.

Selection of **External** grays the **Slave number** field and makes the module use the value of the slave address saved into its internal *(see page 154)* FLASH memory.

NOTE: If the address stored into the FLASH is not into the MODBUS range address, then the default slave address 248 will be used.
When the firmware of the module is updated, the default slave address stored into the FLASH is set to 248. A new command has to be used to re-initialize the FLASH address.

Transmission-linked Modbus Parameters

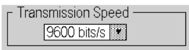
At a Glance

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

These parameters are accessible from five zones:
- The Transmission Speed zone,
- The Delay Between Characters zone,
- The Data zone,
- The Stop zone,
- The Parity zone.

The Transmission Speed Zone

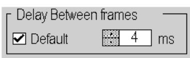
This configuration zone appears on the screen as shown below:



You can use it to select the transmission speed of the Modbus serial link. The selected speed has to be consistent with the other devices. The configurable values are 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200 (only on channel 0 in RS232 mode) bits per second.

The Delay Between Frames Zone

This configuration zone shown below is only accessible in RTU mode (it is grayed in ACSCII mode):



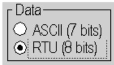
The Delay Between Frames is the minimum time separating two frames on reception. This delay is managed when the BMX NOM 0200 (master or slave) is receiving messages.

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:

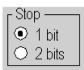


This zone allows you to enter the type of coding used to communicate using Modbus serial link. This field is set according to the other devices connected on the bus. There are two configurable modes:

- 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

Signal and Physical Line Parameters in Modbus

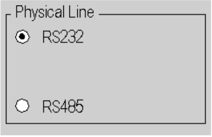
At a Glance

After configuring the communication channel, you need to enter the signal and physical line parameters.

These parameters are accessible via three zones:
- The Physical Line zone,
- The Signals zone,
- The RTS/CTS Delay zone.

The Physical Line Zone

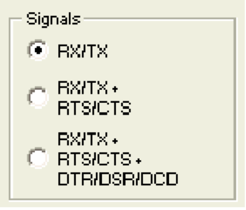
This configuration zone shown below is accessible only on Channel 0 (it is grayed out and configured to RS485 on Channel 1):



In this zone, you can choose between two types of physical line for the serial port on the BMX NOM 0200 module:
- The RS232 line,
- The RS485 line.

The Signals Zone

This configuration zone appears on the screen as shown below:

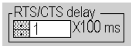


In this zone, you can select the signals supported by the RS232 physical line:
- RX/TX
- RX/TX + RTS/CTS (hardware flow management signals)
- RX/TX + RTS/CTS + DTR/DSR/DCD (Modem signals)

If the RS485 is configured, the entire zone is 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 or RX/TX+RTS/CTS+DTR/DSR/DCD check boxes are selected. An RTS/CTS hardware flow control is performed.

The RTS/CTS hardware flow control algorithm aims at preventing the overflow reception buffer (full duplex).

The RTS/CTS delay corresponds to the time out delay between the RTS rise up and the CTS rise up. A RTS/CTS delay value different from 0 also corresponds to the maximum waiting time between each character transmission after the rise of RTS and CTS signals. If the value is set to 0, UARTs can get stuck in a waiting state for an infinite time until the CTS rise up so the value 0 is used only in particular cases such as looping the RTS signal to the CTS signal in order to check that all connection are operating correctly.

NOTE: The default value is 0 ms.

How to Set the BMX NOM0200 MODBUS Slave Address Without Unity Pro?

Condition and Prerequisite

The FLASH address can be updated in any mode but it is taken into account only when an `operating mode` is performed.

The list below indicates the conditions and prerequisite to set the BMX NOM0200 MODBUS address without Unity Pro:

- To use the FLASH address, the module must be configured:
  - In MODBUS slave protocol with the **EXTERNAL** checkbox.
  - In MODBUS master protocol or in **CHAR** mode and then switched to MODBUS slave protocol.

Update the MODBUS Slave Address into the FLASH by Applicative Commands

The table below indicates the operations to update the MODBUS slave address into the FLASH by applicative commands:

Step	Action
1	Store the slave address into the `%MWr.m.c.25`.
2	Set the bit `%MWr.m.c.24.7`.
3	Send the `WRITE_CMD` to the module channel.
4	Check the command end (`%MWr.m.c.0.1` fall down) and the command is accepted (`%MWr.m.c.1.1` is at zero means no error) => the FLASH is updated.
5	Perform one of the following operating modes onto the channel to take the new address into account: ● Application Download ● Cold Start ● Warm Start ● Hot Swap ● Switch protocol (TO SLAVE)
6	Perform a READ_STS onto the channel to check the slave address in the `%MWr.m.c.3` most significant byte.

NOTE: Several orders can be embedded in the same command. If one of the orders cannot be executed, the whole command will be rejected and no order is executed.

Update the MODBUS Slave Address into the FLASH Over the Serial Line

The table below indicates the operations to update the MODBUS slave address into the FLASH over the serial line:

Step	Action
1	Configure the MASTER equipment with the same serial line parameter than a channel of the module.
2	Connect the MASTER to the module in point to point.
3	Send the request `0x11` to the point to point address: `0xF8 0x11 0x01` channelnumber(0 or 1) slaveID(`0..0xF8`)
4	Check the response is OK => the FLASH is updated.
5	Perform an operating mode onto the channel to take the modification in step 4 into account.
6	Send a request `0x11` to check the new slave address: slaveID `0x11 0x01`

NOTE: Do not modify the FLASH regularly to avoid to damage this component (100,000 writing cycles max).

Section 7.3
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 Module	157
Services Supported by a Modbus Link Slave Module	165
Detail of Modbus Expert Mode	167

Services Supported by a Modbus Link Master Module

At a Glance

When used as the master in a Modbus link, a BMX NOM 0200 module supports several services via the READ_VAR, WRITE_VAR and DATA_EXCH communication functions.

Data Exchanges

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

These requests use the READ_VAR and WRITE_VAR communication functions:

Modbus request	Function code	Communication function
Read bits	16#01 or 16#02	READ_VAR
Read words	16#03 or 16#04	READ_VAR
Write bits	16#0F	WRITE_VAR
Write words	16#10	WRITE_VAR

More generally, it is possible to send any Modbus requests to a slave device by using the DATA_EXCH communication function.

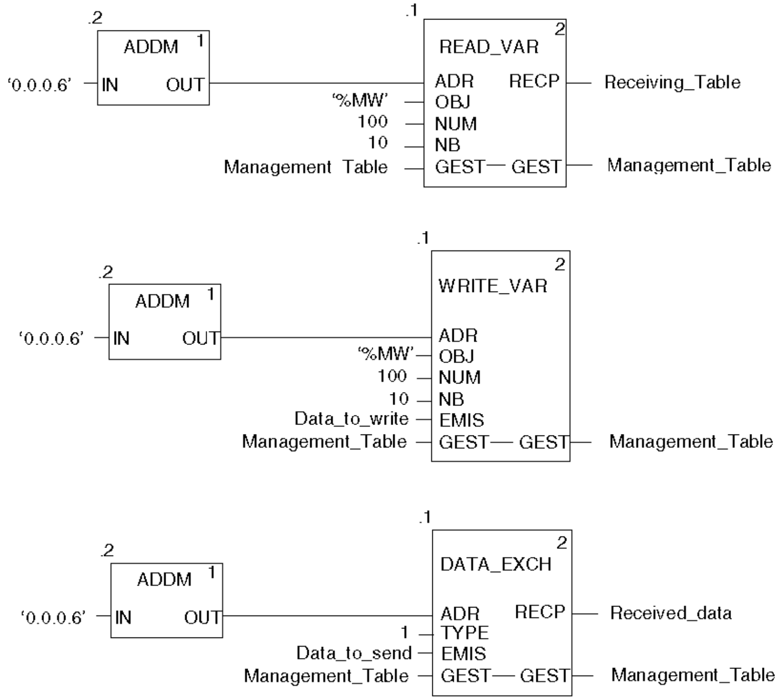
READ_VAR, WRITE_VAR and DATA_EXCH Communication Functions

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

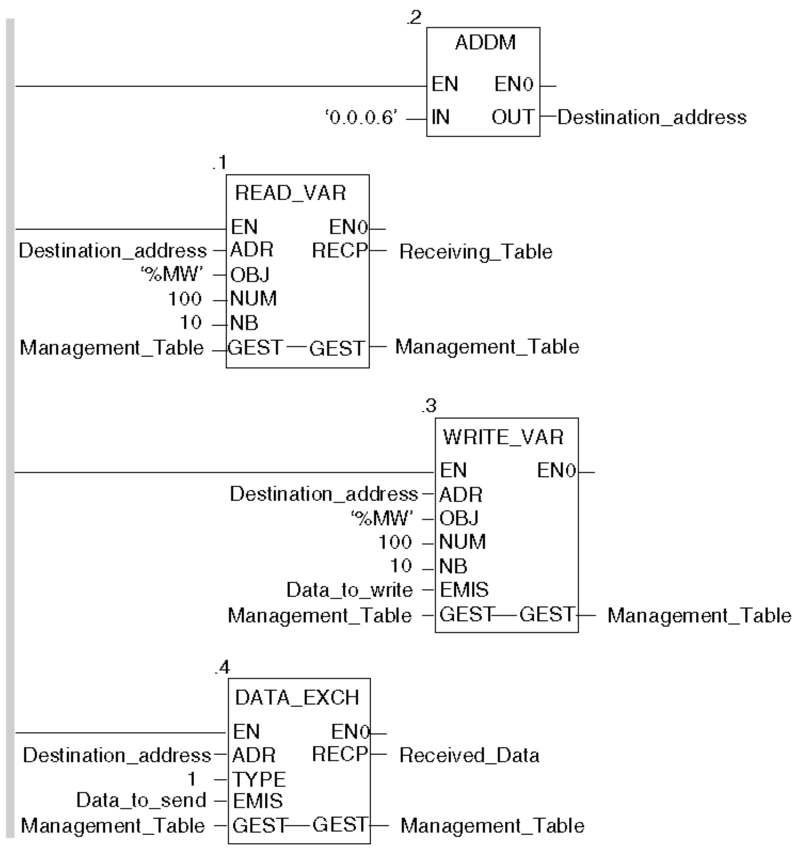
Programming Example in FBD

The diagram below represents an example of programming of the `READ_VAR`, `WRITE_VAR` and `DATA_EXCH` communication functions in the FBD language:



Programming Example in Ladder

The diagram below represents an example of programming of the `READ_VAR`, `WRITE_VAR` and `DATA_EXCH` communication functions in the Ladder language:



Programming Example in ST

The lines of code below represent an example of programming of the `READ_VAR`, `WRITE_VAR` and `DATA_EXCH` communication functions in the ST language:

```
READ_VAR(ADDM('0.0.0.6'), 'MW', 100, 10, Management_Table,
Receiving_Table);

WRITE_VAR(ADDM('0.0.0.6'), '%MW', 100, 10, Data_to_write,
Management_Table);

DATA_EXCH(ADDM('0.0.0.6'), 1, Data_to_send, Management_Table,
Received_data);
```

Cancelling an Exchange

An exchange executed by the `READ_VAR`, `WRITE_VAR` and `DATA_EXCH` functions can be cancelled with either ways of programming, which are both presented in ST language below:

- Using the CANCEL function:
  ```
  IF (%MW40.0) THEN
   %MW200:=SHR(%MW40,8;)
   CANCEL(%MW200,%MW185);
  END_IF;
  ```
  `%MW40` is the `GEST` parameter (management table). `%MW40.0` corresponds to the activity bit of the `READ_VAR` function and is set to 1 when the communication function is active. If this bit is set to 1, the program carries out the following instructions:
  - Moves the `%MW40` bits one byte (8 bits) to the right and loads the byte corresponding to the communication's exchange number into the `%MW200` word,
  - Cancels the exchange whose exchange number is contained within the `%MW200` word using the CANCEL function.
- Using the communication function cancel bit:
  ```
  IF (%MW40.0) THEN
   SET(%MW40.1);
   READ_VAR(ADDM('0.0.0.6'), '%MW', 100, 10, %MW40:4, %MW10:10);
  END_IF;
  ```
  `%MW40` is the `GEST` parameter (management table). `%MW40.0` corresponds to the activity bit of the `READ_VAR` function and is set to 1 when the communication function is active. If this bit is set to 1, the program sets the `%MW40.1` bit, the function cancel bit, to 1. This stops communication of the `READ_VAR` function.

NOTE: When using the communication function cancel bit contained in the function exchange management word (%MW40 in this example), the function (READ_VAR in this example) must be called in order to activate the cancellation of the exchange.

NOTE: When using the communication function cancel bit, it is possible to cancel a communication from an animation table. This can be done by simply setting the function cancel bit to 1 (%MW40.1 in this example) and then start again the communication function.

NOTE: This example of programming concerns the `READ_VAR` function, but is equally applicable to the `WRITE_VAR` as well as the `DATA_EXCH` functions.

NOTE: The CANCEL function uses a report word for the CANCEL function (`%MW185` in this example).

Description of ADDM Function Parameters

The following table outlines the various parameters for the ADDM function:

Parameter	Type	Description
IN	STRING	Address of device on bus or serial link. The syntax of the address is of the 'r.m.c.node' type. The address is made up of the following parameters: ● **r**: Rack number of the module ● **m**: Slot number of the module within the rack ● **c**: Channel number of the module ● **node**: Number of slave to which the request is being sent
OUT	ARRAY [0..7] OF INT	Array representing the address of a device. This parameter can be used as an input parameter for several communication functions.

Description of READ_VAR Function Parameters

The following table outlines the various parameters for the READ_VAR function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the destination entity given by the OUT parameter of the ADDM function.
OBJ	STRING	Type of object to be read. The available types are as follows: ● %M: internal bit ● %MW: internal word ● %I: external input bit ● %IW: external input word
NUM	DINT	Address of first object to be read.
NB	INT	Number of consecutive objects to be read.

Parameter	Type	Description

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		● Rank 1 word: A word managed by the system and consisting of two bytes:
		  ● Most significant byte: Exchange number,
		  ● Least significant byte: Activity bit (rank 0) and cancel bit (rank 1).
		● Rank 2 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: Operation report,
		  ● Least significant byte: Communication report.
		● Rank 3 word: A word managed by the user which defines the maximum response time using a time base of 100 ms.
		● Rank 4 word: A word managed by the system which defines the length of the exchange.

RECP	ARRAY [n..m] OF INT	Word table containing the value of the objects read.

Description of `WRITE_VAR` Function Parameters

The following table outlines the various parameters of the `WRITE_VAR` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the destination entity given by the OUT parameter of the ADDM function.
OBJ	STRING	Type of object to be written. The available types are as follows: ● `%M`: internal bit ● `%MW`: internal word **Note:** `WRITE_VAR` cannot be used for `%I` and `%IW` variables.
NUM	DINT	Address of first object to be written.
NB	INT	Number of consecutive objects to be written.
EMIS	ARRAY [n..m] OF INT	Word table containing the value of the objects to be written.

Parameter	Type	Description

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		• Rank 1 word: A word managed by the system and consisting of two bytes:
		  • Most significant byte: Exchange number,
		  • Least significant byte: Activity bit (rank 0) and cancel bit (rank 1).
		• Rank 2 word: A word managed by the system and consisting of two bytes:
		  • Most significant byte: Operation report,
		  • Least significant byte: Communication report.
		• Rank 3 word: A word managed by the user which defines the maximum response time using a time base of 100 ms.
		• Rank 4 word: A word managed by the system which defines the length of the exchange.

Description of `DATA_EXCH` Function Parameters

The following table outlines the various parameters of the `DATA_EXCH` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the destination entity given by the OUT parameter of the ADDM function.
TYPE	INT	For Modicon M340 PLCs, the only possible value is **1**: Transmission of an **EMIS** array, then the PLC waits for the reception of a **RECP** array.
EMIS	ARRAY [n..m] OF INT	Integers table to be sent to the destination device of the request. **Note:** It is imperative that the length of the data to be sent (in bytes) be assigned to the fourth word of the management table before launching the function, in order for this to be correctly executed.

<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>GEST</td><td>ARRAY [0..3] OF INT</td><td>Exchange management table consisting of the following words:
Rank 1 word: A word managed by the system and consisting of two bytes:
Most significant byte: Exchange number.
Least significant byte: Activity bit (rank 0) and cancel bit (rank 1).
Rank 2 word: A word managed by the system and consisting of two bytes,:
Most significant byte: Operation report,
Least significant byte: Communication report.
Rank 3 word: A word managed by the user which defines the maximum response time using a time base of 100 ms.
Rank 4 word: A word managed by the system which defines the length of the exchange.</td></tr>
<tr><td>RECP</td><td>ARRAY [n..m] OF INT</td><td>Integers table containing the data received.
Note: The size of the data received (in bytes) is written automatically by the system in the fourth word of the management table.</td></tr>
</table>

Services Supported by a Modbus Link Slave Module

At a Glance

When used as a slave in a Modbus link, a BMX NOM 0200 module supports several services.

Data Exchanges

A slave module manages the following requests:

Modbus request	Function code	PLC object
Read n output bits	16#01	%M
Read n output words	16#03	%MW
Write n output bits	16#0F	%M
Write n output words	16#10	%MW
Read/Write n output words	16#17	%MW

NOTE: Read/Write multiple %MW
The `WRITE` performs before the `READ` to be able to write and read the same registers in same time as `IOscanning`. If the exchange size of the `WRITE` or the `READ` is out of boundary then the return status will be "ILLEGAL DATA ADDRESS", but if only the `READ` fail then the `WRITE` will be done with the same status.

Diagnostics and Maintenance

The diagnostics and maintenance requests managed by a Modbus slave BMX NOM 0200 module are listed below:

Designation	Function code/sub-function code
Read exception status	16#07
Restart Communications Option	16#08 / 16#01
Return Diagnostic Register	16#08 / 16#02
Change ASCII Input Delimiter	16#08 / 16#03
Force Listen Only Mode	16#08 / 16#04
Clear Counters and Diagnostic Register	16#08 / 16#0A
Return Bus Message Count	16#08 / 16#0B
Return Bus Communication Error Count	16#08 / 16#0C
Return Bus Exception Error Count	16#08 / 16#0D
Return Slave Message Count	16#08 / 16#0E
Return Slave No Response Count	16#08 / 16#0F
Return Slave Negative Acknowledgements Count	16#08 / 16#10

Designation	Function code/sub-function code
Return Slave Busy Count	16#08 / 16#11
Return Bus Character Overrun Count	16#08 / 16#12
Get Communication event Counter	16#0B
Get Communication event Log	16#0C
Report Slave identification	16#11
Write Slave identification	16#11 / 16#01

Detail of Modbus Expert Mode

Expert Mode Communication

Expert mode is a set of commands that can be sent to the module to get extra features.

Address	Standard Symbol	Exchange Type	Type	Meaning
%MWr.m.c.24	CONTROL	Explicit	INT	Command signal, change protocol
%MWr.m.c.24.0		Explicit	BOOL	Erase local counters
%MWr.m.c.24.1		Explicit	BOOL	Change dynamically the retries count in MODBUS master mode (%MW26)
%MWr.m.c.24.2		Explicit	BOOL	Change the slave answer delay (%MW28) for a specific slave (%MW27) in master mode
%MWr.m.c.24.3		Explicit	BOOL	Modify the default slave blind time, the slave ignore received char after a frame reception forwarded to the CPU (%MW29)
%MWr.m.c.24.4		Explicit	BOOL	Modify the MODBUS RTU internal timings t1,5ch (%MW31), t3,5ch (%MW30), and inter exchange delay (%MW32). This value update may disturb the module if it's working

Address	Standard Symbol	Exchange Type	Type	Meaning
%MWr.m.c.24.6		Explicit	BOOL	Change HALF/FULL DUPLEX modem management mode ● If set simultaneously with RTS_ON (%MWr.m.c.24.10 works also with RTS_OFF %MWr.m.c.24.11 and use DTR if %MWr.m.r.24.8 or %MWr.m.r.24.9 is used) the half duplex modem mode is activated ● If this bit is set but none of the RTS/DTR (neither %MWr.m.c.24.8, %MWr.m.c.24.9, %MWr.m.c.24.10, %MWr.m.c.24.11) the full duplex mode is activated The %MW26 is used to set the StartDelay and %MW27 is used to set the EndDelay. So the bit %MW24.5 and %MW24.1 and %MW24.2 cannot be used simultaneously. **NOTE**: The user may have to restore the correct state of the RTS/DTR signals after the command has been accepted.
%MWr.m.c.24.7	SAVE_SLAVE_ADDR	Explicit	BOOL	Save the Modbus slave address into the FLASH (%MW25).
%MWr.m.c.24.8	DTR_ON	Explicit	BOOL	Set the DTR signal (positive voltage)
%MWr.m.c.24.9	DTR_OFF	Explicit	BOOL	Reset the DTR signal (negative voltage)
%MWr.m.c.24.10		Explicit	BOOL	Set the RTS signal (positive voltage)
%MWr.m.c.24.11		Explicit	BOOL	Reset the RTS signal (negative voltage)
%MWr.m.c.24.12	TO_MODBUS_MASTER	Explicit	BOOL	Switch to master mode
%MWr.m.c.24.13	TO_MODBUS_SLAVE	Explicit	BOOL	Switch to slave mode
%MWr.m.c.24.14	TO_CHAR_ MODE	Explicit	BOOL	Switch to character mode

Address	Standard Symbol	Exchange Type	Type	Meaning
%MWr.m.c.25	SLAVE_ADDR	Explicit	INT	Modbus slave address to store in FLASH
%MWr.m.c.26		Explicit	INT	LOW BYTE : MasterRetries count: Retry number in master mode [0..15] see %MW24.1 StartDelay if %MW26.6 is set. Time to wait after the CTS is OK before to start to send the frame. It is useful for modem that requires extra time after CTS or do not manage the CTS signal (in this case the RTS must be connected to the CTS). This time is in millisecond, the precision is about 3ms. Can be performed only in RS232 mode.
%MWr.m.c.27		Explicit	INT	LOW BYTE : Slave for which the master will adapt the answer delay [0..248, 255=ALL] see %MW24.2 and %MW28 EndDelay if %MW24.6 is set. Time to wait after having sent a frame, before to release the RTS signal to let enough time to the MODEM to completely send the frame before hand-up. This time is in millisecond, the precision is about 3ms. Can be performed only in RS232 mode.
%MWr.m.c.28		Explicit	INT	Specific answer delay for a slave in 10ms [1..1000] see %MW24.2 and %MW27
%MWr.m.c.29		Explicit	INT	Blind time in 10ms [1..10] see %MW24.3

Address	Standard Symbol	Exchange Type	Type	Meaning
%MWr.m.c.30		Explicit	INT	T3,5char: Inter frame delay in milliseconds [0..10000]. The value used depends of the speed. If the value is smaller or greater than possible values, the lower limit or upper limit is applied, and the command is accepted. A value 0 means no change in RTU. The answer delay is computed again.
%MWr.m.c.31		Explicit	INT	T1,5char : Delay between char in milliseconds [0..9999]. The value used depends of the speed. If the value is smaller or greater than possible values, the lower limit or upper limit is applied, and the command is accepted. A value 0 means compute T1,5 as T3,5ch – 2ch (default compute).
%MWr.m.c.32		Explicit	INT	Master inter exchange delay in RTU mode [0..256] in miliseconds. The value 0 means “no delay”, if the value is less than 10bits duration, the minimal value of 10 bits is used.

Sample of Code

```
(* master sideNOM is is rack 0 slot 9 *)
if HalfModemMaster then
 HalfModemMaster:=false;
 %MW0.9.0.24:=16#0450;(* switch to half duplex mode with RTS, and change MODBUS timings*)
 %MW0.9.0.26:=12;(* 12ms to wait before sending when CTS raise *)
 %MW0.9.0.27:=9; (* let RTS up 9ms after sending end *)
 %MW0.9.0.30:=0;
 %MW0.9.0.31:=0;(*use the value of the configuration screen equal 6ms *)
 %MW0.9.0.32:=50; (*50ms of delay before sending a new frame*)
 write_cmd(%ch0.9.0);(* send command and data to the NOM channel*)
```

```
end_if;
(* slave side the NOM is in rack 0 slot 3 *)
if HalfModemSlave then
 HalfModemSlave:=false;
 %MW0.3.0.24:=16#0448;(* switch to half duplex mode with RTS, and change the slave blind time*)
 %MW0.3.0.26:=12;(* 12ms to wait before sending when CTS raise *)
 %MW0.3.0.27:=9; (* let RTS up 9ms after sending end *)
 %MW0.3.0.29:=4; (* 4*10ms of blind time *)
 write_cmd(%ch0.3.0);(* send command and data to the NOM channel*)
end_if;
(* optional: sending the command automatically *)
if %S0 or %S1 or %S13 then
 memoSendCmd:=true;
end_if;
(* copy each cycle the module error to detect module disparition *)
memoSendCmd:=%I0.3.0.ERR;
(* if the module is OK send the command one time *)
if FE(memoSendCmd) then
 HalfModemSlave:=true;
end_if;
```

NOM Internal Register Readable

Nom internal registers can be accessed only in MODBUS mode by using the READ_VAR EF. Sample of code (the NOM module is in rack 0 slot 3):

```
if dataCh030GetChannelGlobalInfo then
  read_var(addm('0.3.0'), '%MW', 200, 3, dataCh030Mgt,  dataCh030Buff);
(* Internal_Reg@200 are copied into the buffer dataCh030Buff *)
 dataCh030GetChannelGlobalInfo := false;
end_if;
```

- Internal_Reg@0 : StartDelay in ms (precision about 3ms) (read or write access)
- Internal_Reg@1 : EndDelay in ms (precision about 3ms) (read or write access)
- Internal_Reg@200 : interface version number = 1
- Internal_Reg@201 : slave address stored in FLASH
- Internal_Reg@202 : 1=possible to change the FLASH, 0=forbidden to change it

- Internal_Reg@1000 : Modbus master RTU internal code ch0=1110, ch1=2110
- Internal_Reg@1002 : 0 = Full Duplex - Hardware flow control, or RS485 ; 1 = Half Duplex - Direction managed automatically by the module with RTS
- Internal_Reg@1010 : Internal sending inter char delay in bits (nbbits*1000/speed => duration in ms) [T1,5S].
- Internal_Reg@1012 : Internal reception inter char delay in bits [T1,5R].
- Internal_Reg@1014 : Internal sending inter frame delay in bits [T3,5S]
- Internal_Reg@1016 : Internal reception inter frame delay in bits [T3,5R]
- Internal_Reg@1018 : Delay to wait before sending the next frames in bits.
- Internal_Reg@1090 : MasterRetries count.
- Internal_Reg@1100 : Slave answer delay for broadcast in 10ms.
- Internal_Reg@1101 : Slave answer delay for slave 1 in 10ms.
- …
- Internal_Reg@1348 : Slave answer delay for point to point address (248).
- Internal_Reg@1500 : Modbus RTU slave internal code ch0=1120, ch1=2120
- Internal_Reg@1502 : 0 = Full Duplex - Hardware flow control, or RS485 ; 1 = Half Duplex - Direction managed automatically by the module with RTS
- Internal_Reg@1510 : Internal sending inter char delay in bits (nbbits*1000/speed => duration in ms) [T1,5S].
- Internal_Reg@1512 : Internal reception inter char delay in bits [T1,5R].
- Internal_Reg@1514 : Internal sending inter frame delay in bits [T3,5S].
- Internal_Reg@1516 : Internal reception inter frame delay in bits [T3,5R].
- Internal_Reg@1518 : Delay to wait before sending the next frames in bits.
- Internal_Reg@1602 : Blind time after reception in ms.
- Internal_Reg@1606 : Listen Only Mode active = 1, (not active = 0).
- Internal_Reg@2000 : Modbus master ASCII internal code ch0=1210, ch1=2210
- Internal_Reg@2002 : 0 = Full Duplex - Hardware flow control, or RS485 ; 1 = Half Duplex - Direction managed automatically by the module with RTS
- Internal_Reg@2010 : Internal sending inter char delay in bits (nbbits*1000/speed => duration in ms) [T1,5S].
- Internal_Reg@2012 : Internal reception inter char delay in bits [T1,5R].
- Internal_Reg@2014 : Internal sending inter frame delay in bits [T3,5S].
- Internal_Reg@2014 : Internal sending inter frame delay in bits [T3,5S].
- Internal_Reg@2014 : Internal sending inter frame delay in bits [T3,5S].
- Internal_Reg@2016 : Internal reception inter frame delay in bits [T3,5R].
- Internal_Reg@2018 : Delay to wait before sending the next frames in bits.
- Internal_Reg@2090 : MasterRetries count.
- Internal_Reg@2100 : Slave answer delay for broadcast in 10ms.
- Internal_Reg@2101 : Slave answer delay for slave 1 in 10ms.
- …
- Internal_Reg@2348 : Slave answer delay for point to point address (248).
- Internal_Reg@2500 : Modbus ASCII slave internal code ch0=1220, ch1=2220
- Internal_Reg@2502 : 0 = Full Duplex - Hardware flow control, or RS485 ; 1 = Half Duplex - Direction managed automatically by the module with RTS

- Internal_Reg@2510 : Internal sending inter char delay in bits (nbbits*1000/speed => duration in ms) [T1,5S].
- Internal_Reg@2512 : Internal reception inter char delay in bits [T1,5R].
- Internal_Reg@2514 : Internal sending inter frame delay in bits [T3,5S].
- Internal_Reg@2516 : Internal reception inter frame delay in bits [T3,5R].
- Internal_Reg@2518 : Delay to wait before sending the next frames in bits.
- Internal_Reg@2600 : Slave address in use.
- Internal_Reg@2602 : Blind time after reception in ms.
- Internal_Reg@2606 : Listen Only Mode active = 1, (not active = 0).
- Internal_Reg@3000 : Char mode internal code ch0=1000, ch1=2000
- Internal_Reg@3002 : 0 = Full Duplex - Hardware flow control, or RS485 ; 1 = Half Duplex - Direction managed automatically by the module with RTS
- Internal_Reg@3100 : 0=no stop criteria active, 1 stop on silence or stop on end of char
- Internal_Reg@3102 : Internal silence in bits (min is 2 bits, max is 65535 bits)
- Internal_Reg@3104 : First End of frame byte to use 16#0100 means no byte
- Internal_Reg@3106 : First EOF : 1=end of frame byte to let in the frame, 0=remove the end of frame byte
- Internal_Reg@3108 : Second end of frame byte
- Internal_Reg@3110 : Second EOF : 1=end of frame byte to let in the frame, 0=remove the end of frame byte

Section 7.4
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:

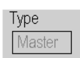
Step	Action
1	Access the configuration screen for Modbus serial communication. *(see page 140)*
2	Select the "Debug" tab on the screen that appears.

Description of the Debug Screen

The debug screen is divided into two or three zones:
- The Type and Slave number zone,
- The Counters zone,
- The Signals zone (if RS232).

The Type and Slave number Zone

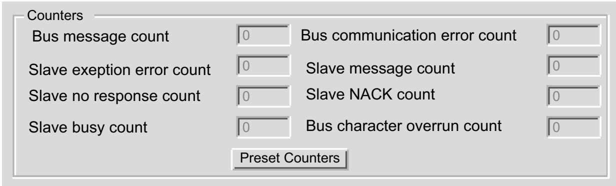
If the module has the function of Master in the Modbus link, this zone looks as following:



If the module has the function of Slave in the Modbus link, this zone looks as following:

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:

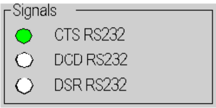
- **Bus message counter**: This counter indicates the number of messages that the module has detected on the serial link. Messages with a negative CRC check result are not counted.
- **Bus communication error counter**: This counter indicates the number of negative CRC check results counted by the module. 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 module.
- **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 module 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 event.

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

The Signals Zone

This zone displays only if RS232 is selected in configuration screen. If RS485 is selected in configuration screen, this window is not displayed at all.

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**: shows the activity of the DCD signal.
- **DSR RS232**: shows the activity of the DSR signal.

Chapter 8
Character Mode Communication for BMX NOM 0200

Subject of this Section

This chapter presents the software implementation of communication using Character Mode for BMX NOM 0200.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
8.1	Generalities	178
8.2	Character Mode Communication Configuration	179
8.3	Character Mode Communication Programming	195
8.4	Debugging Character Mode communication	209

Section 8.1
Generalities

About Character Mode Communication

Introduction

Communication in Character Mode enables dialog and communication functions to be carried out between the PLCs and 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.**

Section 8.2
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
BMX NOM 0200 Character Mode Communication Configuration Screen in a Local Rack	180
BMX NOM 0200 Character Mode Communication Configuration Screen in X80 Drop	183
Accessible Functions in Character Mode	186
Default Values for Character Mode Communication Parameters	187
Message End Detection Parameters in Character Mode	188
Transmission Parameters in Character Mode	190
Signal and Physical Line Parameters in Character Mode	192

BMX NOM 0200 Character Mode Communication Configuration Screen in a Local Rack

General

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

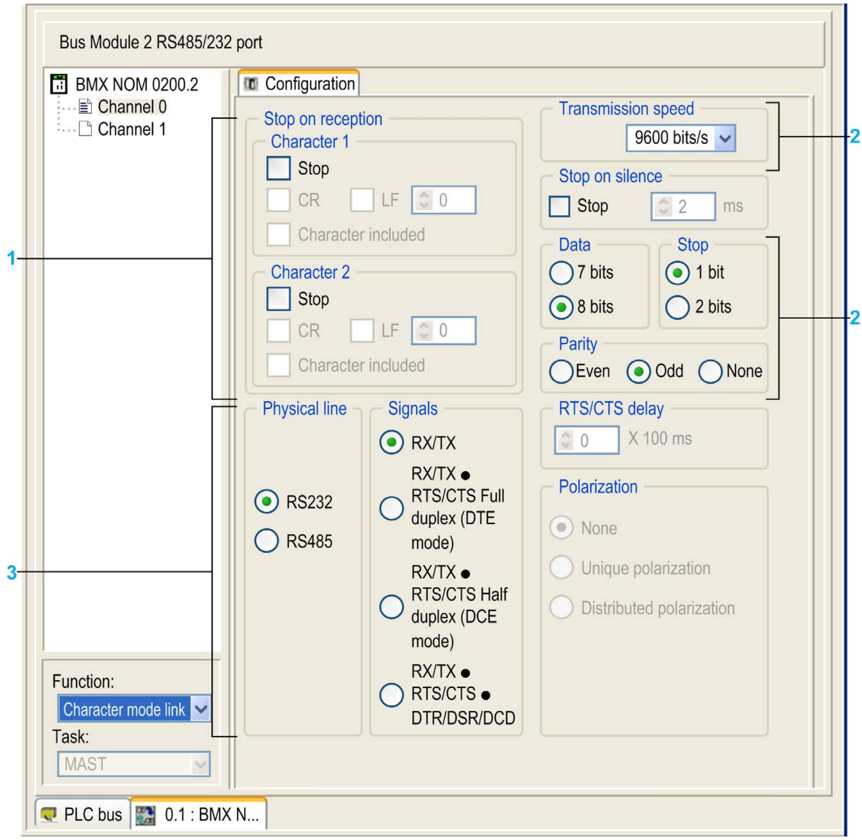
Accessing the Configuration Screen

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

Step	Action
1	Open the BMX NOM 0200 sub-directory in the project browser *(see page 136)*.
2	Select the Channel to configure and the Character mode link function on the screen that appears.

Character Mode Configuration Screen

The figure below shows the default configuration screen for Character Mode communication on Channel 0:



Description

These zones are used to configure channel parameters. In the online mode, these zones are accessible. In the offline mode, these zones 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:

Key	Element	Comment
1	Message end detection parameters *(see page 188)*	These parameters are accessible via two zones: ● Stop on reception, ● Stop on silence.
2	Transmission parameters *(see page 190)*	These parameters are accessible via four zones: ● Transmission speed, ● Data, ● Stop bits, ● Parity.
3	Signal and physical line parameters *(see page 192)*	These parameters are accessible via four zones: ● Physical line, ● Signals, ● RTS/CTS delay, ● Polarization.

NOTE: In this example, the "Polarization" and "RTS/CTS Delay" zones are grayed out respectively because an RS232 physical line and RX/TX signals have been chosen.

BMX NOM 0200 Character Mode Communication Configuration Screen in X80 Drop

General

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

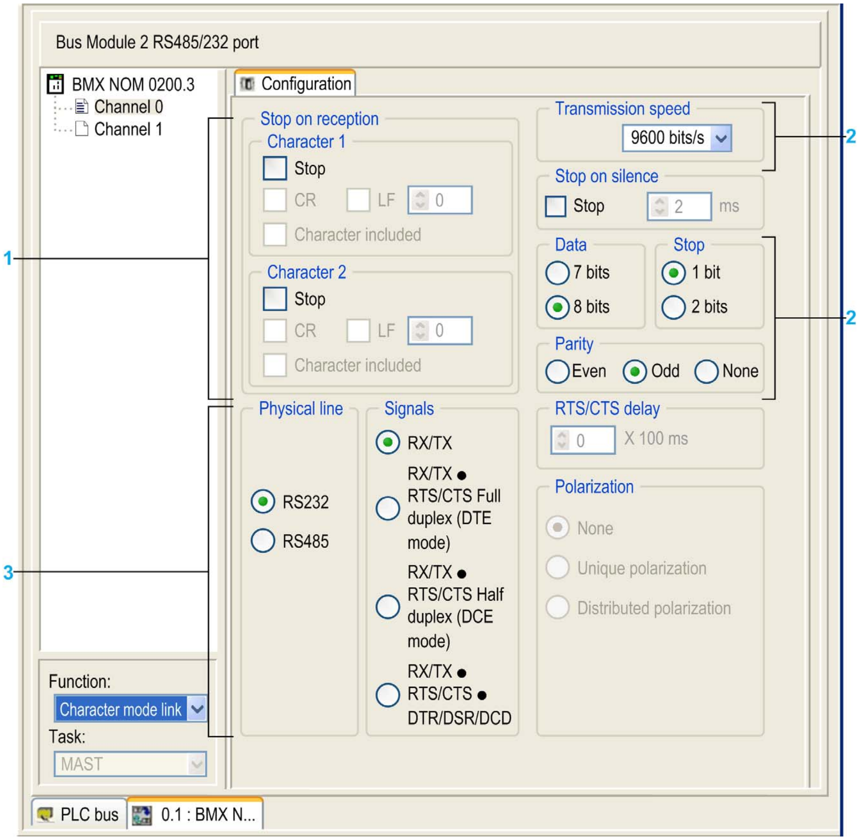
Accessing the Configuration Screen

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

Step	Action
1	Open the BMX NOM 0200 sub-directory in the project browser *(see page 136)*.
2	Select the Channel to configure and the Character mode link function on the screen that appears.

Character Mode Configuration Screen

The figure below shows the default configuration screen for Character Mode communication on Channel 0:



Description

These zones are used to configure channel parameters. In the online mode, these zones are accessible. In the offline mode, these zones 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:

Key	Element	Comment
1	Message end detection parameters *(see page 188)*	These parameters are accessible via two zones: ● Stop on reception, ● Stop on silence.
2	Transmission parameters *(see page 190)*	These parameters are accessible via four zones: ● Transmission speed, ● Data, ● Stop bits, ● Parity.
3	Signal and physical line parameters *(see page 192)*	These parameters are accessible via four zones: ● Physical line, ● Signals, ● RTS/CTS delay, ● Polarization.

NOTE: In this example, the "Polarization" and "RTS/CTS Delay" zones are grayed out respectively because an RS232 physical line and RX/TX signals have been chosen.

Accessible Functions in Character Mode

At a Glance

Function accessibility for configuration of the serial link of a BMX NOM 0200 using Character Mode protocol depends on the physical link being used.

Accessible Functions

The table below shows the different functions configurable according to the type of serial link used:

Function	RS 485 Link (Channel 0 or Channel 1)	RS 232 Link (Channel 0)
Transmission speed	X	X
Data	• 7 bits • 8 bits	• 7 bits • 8 bits
Stop	• 1 bit • 2 bits	• 1 bit • 2 bits
Parity	• Odd • Even • None	• Odd • Even • None
Stop on Reception	X	X
Stop on Silence	X	X
RX/TX Signals	X	X
RTS/CTS Signals	-	X
RTS/CTS delay	-	X
DTR/DSR/DCD Signals	-	X
Polarization	X	-

X Accessible Function
- Unaccessible Function

Default Values for Character Mode Communication Parameters

At a Glance

All Character Mode communication parameters have default values.

Default Values

The table below shows the default values for Character Mode communication parameters on Channel 0 and Channel 1 of BMX NOM 0200 module:

Configuration parameter	Value on Channel 0	Value on Channel 1
Physical Line	RS232	RS485
Signals	RX/TX	RX/TX (unique value)
Transmission speed	9600 bits/s	9600 bits/s
Parity	Odd	Odd
Data Bits	8 bits	8 bits
Stop bits	1 bit	1 bit
Polarization	None (unique value)	None

Message End Detection Parameters in Character Mode

At a Glance

After configuring the communication channel, you need to enter the message end detection parameters.

These parameters are accessible via two zones:

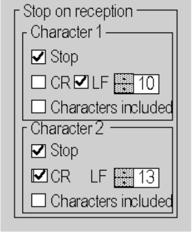
- The Stop on Reception Zone: stop on reception of a special character.
- The Stop on Silence Zone: stop on silence.

Conditions of Use

Selecting Stop on Silence means that Stop on Reception is deselected and vice versa.

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 CR or LF 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 window below, the end of reception of a message is detected by an LF or CR 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.

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

Transmission Parameters in Character Mode

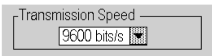
At a Glance

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

These parameters are accessible via four zones:
- The Transmission Speed zone,
- The Data zone,
- The Stop zone,
- The Parity zone.

The Transmission Speed Zone

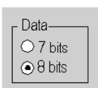
This configuration zone appears on the screen as shown below:



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, 57600 and 115200 (only on channel 0 in RS232 mode) bits per second.

The Data Zone

This configuration zone appears on the screen as shown below:



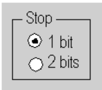
In this zone, you can specify the size of the data being exchanged on the link.

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

Signal and Physical Line Parameters in Character Mode

At a Glance

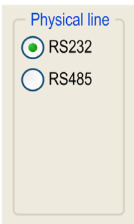
After configuring the communication channel, you need to enter the signal and physical line parameters.

These 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:

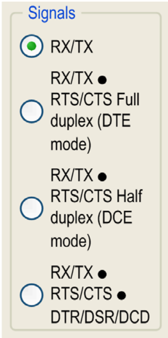


In this zone, you can choose between two types of physical line for the serial port on the BMX NOM 0200 module:

- The RS 232 line
- The RS 485 line

The Signals Zone

This configuration zone appears on the screen as shown below:



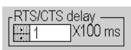
In this zone, you can select the signals supported by the RS 232 physical line:

- RX/TX
- RX/TX + RTS/CTS Full Duplex (DTE mode)
- RX/TX + RTS/CTS Half Duplex (DCE mode)
- RX/TX + RTS/CTS + DTR/DSR/DCD

If the RS 485 is configured, the entire zone is 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 or RX/TX+RTS/CTS+DTR/DSR/DCD check boxes are selected. An RTS/CTS hardware flow control is performed.

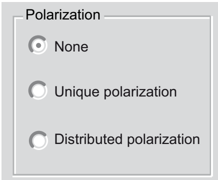
The RTS/CTS hardware flow control algorithm aims at preventing the overflow reception buffer (full duplex).

The RTS/CTS delay corresponds to the time out delay between the RTS rise up and the CTS rise up. A RTS/CTS delay value different from 0 also corresponds to the maximum waiting time between each character transmission after the rise of RTS and CTS signals. If the value is set to 0, UARTs can get stuck in a waiting state for an infinite time until the CTS rise up so the value 0 is used only in particular cases such as looping the RTS signal to the CTS signal in order to check that all connection cables are operating correctly.

NOTE: The default value is 0 ms.

The Polarization zone

This configuration zone shown below is accessible when "RS485" is selected in the "Physical Line" zone:



This zone gives the capability to choose between three types of configuration for the polarization on the channel:

- **None** to use no polarization in case you have your own termination.
- **Unique polarization** to use a low impedance polarization like in Modbus networks (the goal of this kind of polarization is to let the master maintain the default state).
- **Distributed polarization** to use a high polarization impedance (the goal of this kind of polarization is to let each device contribute to maintain the default state).

Section 8.3
Character Mode Communication Programming

Subject of this Section

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

What Is in This Section?

This section contains the following topics:

Topic	Page
Character Mode Communication Functions	196
Detail of Character Mode Expert Mode	204

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` sends a character string with a maximum of 16 x 1,024 bytes.
- `INPUT_CHAR` reads a character string with a maximum of 16 x 1,024 bytes.
- `INPUT_BYTE` reads byte arrays with a maximum of 16 x 1,024 bytes.

The BMX NOM 0200 can store a total of 16 frames in emission or reception. The frames in buffers are managed in FIFO order. Over RS-232 lines, they are managed in full duplex mode.

NOTE: For the `INPUT_CHAR` function, a configured time-out is necessary if the channel is configured without **Stop on silence**, to acknowledge the activity bit of the function.

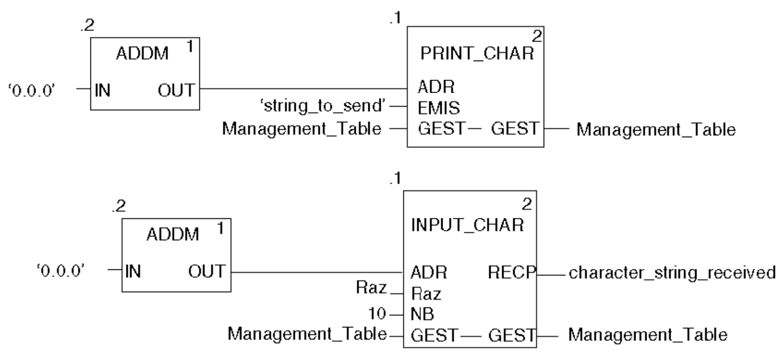
Number of Frames Received in Buffers

When the Modbus port si configured in the Character mode, %MWr.m.c.7 indicates the number of frames in the BMX NOM 0200 Receive buffer.

This WORD is incremented each time the BMX NOM 0200 received a frame over a RS-232 line.

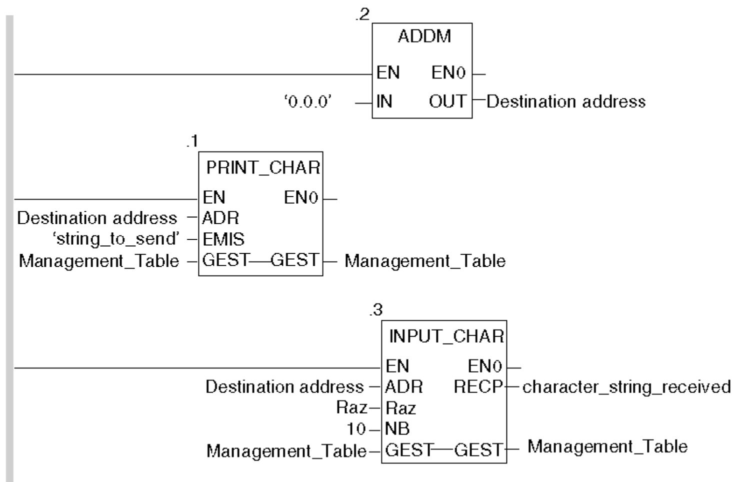
Example of Programming in FBD

The diagram below represents an example of programming of the `PRINT_CHAR` and `INPUT_CHAR` communication functions in FBD language:



Example of Programming in Ladder

The diagram below represents an example of programming of the `PRINT_CHAR` and `INPUT_CHAR` communication functions in the Ladder language:



Example of Programming in ST

The lines of code below represent an example of programming of the `PRINT_CHAR` and `INPUT_CHAR` communication functions in the ST language:

```
PRINT_CHAR(ADDM('0.1.0'), 'string_to_send', Management_Table);
INPUT_CHAR(ADDM('0.1.0'), reset_integer_to_0, 10, Management_Table,
character_string_received);
```

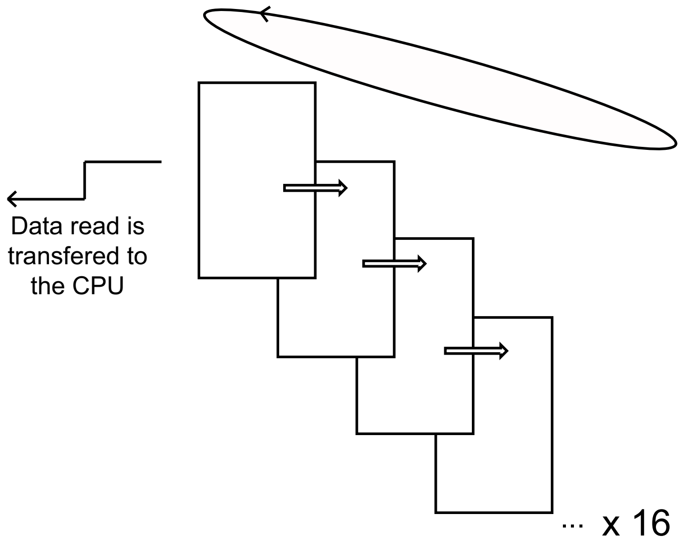
Feature of the INPUT_CHAR function

If the Reset input parameter is set to 1, all buffers are first reset then the module is waits for the reception of data. Using this feature is advised to start properly a reception by removing old data from the buffers.

Internal Mechanism of the BMX NOM 0200 Module

The data received is stored in 16 cycling buffers in series, each buffer containing 1024 bytes.

The figure below represents this mechanism:



Receiving data with `INPUT_CHAR`or `INPUT_BYTE`

Frames are retrieved by the application program using the `INPUT_CHAR` EF to receive a string, or the `INPUT_BYTE` EF to receive binary data.

The `INPUT_CHAR` or `INPUT_BYTE` EF may be executed before data is received by the module. In this case, the module waits for data from the line and then sends it to the CPU.

The EF can also be executed when the frame was already received (for example, after checking the %MWr.m.c.7 with `READ_STS`). In this case the module immediately sends the buffered frame to the CPU.

It is also possible to force the module to wait for data from the line by setting the Reset parameter of the EF to 1. In this case, the data previously buffered are flushed, and the BMX NOM 0200 waits for new data to be sent to the CPU.

Reception mode

The BMX NOM 0200 module can store a maximum of 16 frames.

It does not matter if the end of frame criteria has been defined or not, the BMX NOM 0200 can be set internally in the Message mode or Raw mode:

- The Message mode is set when an end of frame condition has been specified.
  In this mode, the NB parameter of `INPUT_CHAR` or `INPUT_BYTE` is the amount of bytes sent to the CPU.
  Special values NB = 0 and NB = 1024 specify the whole frame is sent to the CPU.
  Other values of NB specify the number of bytes to be sent to the CPU. If more bytes have been received before the call to `INPUT_CHAR` or `INPUT_BYTE`, the remaining bytes are discarded.
- Raw mode is set when no end of frame condition has been specified. Character mode is a "frame oriented" protocol. In reception, an end of frame is detected by the first of the following events:
  - a silence duration specified in the **Stop on silence** zone of the Character Mode Communication Configuration Screen *(see page 180)*
  - the character 1 or 2 has been received, as specified in the **Stop on reception** zones of the Character Mode Communication Configuration Screen *(see page 180)*
  - More than 1024 bytes are received

When an end of frame is detected the internal counter %MWr.m.c.7 is incremented.

Maximum Data Size

The maximum size of a frame sent by the BMX NOM 0200 to the CPU is 1024 byte. However internally the reception frame size has a maximum size of 1025 byte if an end of frame byte is configured and this byte is not to be included into the data sent to the CPU.

Zero-size frames

Zero-size frames are discarded. If an end of frame byte is configured, and not requested as part of the data, a zero size frame received by the BMX NOM 0200 is not sent to the CPU. In this case, if an end of frame byte is received without any data before it, this frame is discarded and no information is sent to the CPU.

Receiving several frames during a MAST task

Several frames can be forwarded by the BMX NOM 0200 to the CPU during one MAST task, and several INPUT_CHAR EF instances may be launched in parallel that address the same BMX NOM 0200 module. This can be required if a huge flow of data arrives over the serial line.

Cancel and Timeout

Cancel and Timeout are forwarded to the BMX NOM 0200 module. A Timeout condition and Cancel orders applied to an instance of `INPUT_CHAR` are forwarded to the BMX NOM 0200 module. The corresponding pending task is removed from the BMX NOM 0200 module task queue.

Internal Mechanism of the BMX NOM 0200 Module: Emission

Use the `PRINT_CHAR` EF to send data over the serial line of the BMX NOM 0200 module.

NOTE: If several frames have been sent (several `PRINT_CHAR` instances have been called) and a silence has been configured, the BMX NOM 0200 module inserts a silence time between each frame.

It is possible to launch up to 16 `PRINT_CHAR` requests: they are sent serially with a silence between each `PRINT_CHAR`.

Cancelling an Exchange

There are two ways of programming that enable an exchange executed by the `PRINT_CHAR` and `INPUT_CHAR` functions to be cancelled. These are both presented in ST language below:

- Using the `CANCEL` function:

  ```
  IF (%MW40.0) THEN
   %MW200:=SHR(%MW40,8;)
   CANCEL(%MW200,%MW185);
  END_IF;
  ```

  `%MW40` is the `GEST` parameter (management table). `%MW40.0` corresponds to the activity bit of the `PRINT_CHAR` function and is set to 1 when the communication function is active. If this bit is set to 1, the program carries out the following instructions:
  - Moves the `%MW40` bits one byte (8 bits) to the right and loads the byte corresponding to the communication's exchange number into the `%MW200` word.
  - Cancels the exchange whose exchange number is contained within the `%MW200` word using the `CANCEL` function.

- Using the communication function's cancel bit:

  ```
  IF (%MW40.0) THEN
   SET(%MW40.1);
   PRINT_CHAR(ADDM('0.1.0'), 'string_to_send', %MW40:4);
  END_IF;
  ```

  `%MW40` is the `GEST` parameter (management table). `%MW40.0` corresponds to the activity bit of the `PRINT_CHAR` function and is set to 1 when the communication function is active. If this bit is set to 1, the program sets the `%MW40.1` bit, the function cancel bit, to 1. This stops communication of the `PRINT_CHAR` function.

NOTE: When using the communication function cancel bit, the function must be called to enable the cancel bit contained in the function exchange management word (`%MW40` in this example).

NOTE: When using the communication function cancel bit, it is possible to cancel a communication from an animation table. This can be done by simply setting the function cancel bit to 1 (`%MW40.1` in this example).

NOTE: The CANCEL function uses a report word for the CANCEL function (`%MW185` in this example).

NOTE: This example of programming concerns the `PRINT_CHAR` function, but is equally applicable to the `INPUT_CHAR` function.

Description of ADDM Function Parameters

The following table outlines the various parameters for the `ADDM` function:

Parameter	Type	Description
IN	STRING	Address of device on bus or serial link. The syntax of the address is of the 'r.m.c.node' type. The address is made up of the following parameters: ● **r**: rack number of the destination system, always = 0. ● **m**: slot number of the destination system within the rack, always = 0. ● **c**: channel number, always = 0 as the serial link of a remote system is always channel 0. ● **node**: optional field that may be `SYS` or empty.
OUT	ARRAY [0..7] OF INT	Table showing the address of a device. This parameter can be used as an input parameter for several communication functions.

Description of PRINT_CHAR Function Parameters

The following table outlines the various parameters of the `PRINT_CHAR` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the message receiving character mode channel given by the OUT parameter of the ADDM function.
EMIS	STRING	Character string to be sent.

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		● Rank 1 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: exchange number
		  ● Least significant byte: activity bit (rank 0) and cancel bit (rank 1)
		● Rank 2 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: operation report
		  ● Least significant byte: communication report
		● Rank 3 word: a word managed by the user that defines the maximum response time using a time base of 100 ms.
		● Rank 4 word: a word managed by the user that defines the length of the exchange:
		  ● If this parameter length is set to 0 then the system sends the entire string.
		  ● If this parameter length is greater than the length of the string then the error 16#0A (Insufficient send buffer size) is returned into the 2nd management word and no character is sent.

Description of INPUT_CHAR Function Parameters

The following table outlines the various parameters of the `INPUT_CHAR` function:

Parameter	Type	Description
ADR	ARRAY [0..7] OF INT	Address of the message receiving character mode channel given by the OUT parameter of the ADDM function.
RAZ	INT	Reset. This parameter is used to reset the receive memory of the coupler: ● Value = 0: no memory reset ● Value = 1: memory reset **NOTE:** %MW.r.m.7 is reset to 0.
NB	INT	Length of character string to be received.

GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words:
		● Rank 1 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: exchange number
		  ● Least significant byte: activity bit (rank 0) and cancel bit (rank 1)
		● Rank 2 word: a word managed by the system and consisting of two bytes:
		  ● Most significant byte: operation report
		  ● Least significant byte: communication report
		● Rank 3 word: a word managed by the user which defines the maximum response time using a time base of 100 ms.
		● Rank 4 word: a word managed by the system which defines the length of the exchange.

RECP	STRING	Character string received. This string is saved in a character string.

Description of INPUT_BYTE Function Parameters

The following table outlines the various parameters of the `INPUT_BYTE` function:

Parameter	Type	Comment
`ADR`	`ARRAY [0.. 7] OF INT`	For Modicon M340 PLCs: ● Address of the message's receiving character mode channel is given by the `ADDM` function. ● The syntax of the address is of `ADDM ('r.m.c.node')`-type. Node is an optional field that may be `SYS` or empty (e.g. `ADDM('0.0.0.SYS')` equals `ADDM('0.0.0')`.

Parameter	Type	Comment
RAZ	INT	Reset. This parameter is used to reset the coupler's receive memory. ● value = 0: no memory reset ● value = 1: memory reset **NOTE:** %MW.r.m.7 is reset to 0. **NOTE**: On Modicon M340 PLCs, the INPUT_BYTE EF can be programmed with or without this parameter.
NB	INT	Length of the buffer or number of bytes to be received. ● Value = 0: Message read as soon as it is available on the channel. Here, a stop condition must be specified in the configuration screen. ● Value greater than 0: Specifies the number of bytes to be read. **NOTE**: The default end of message character is a carriage return (CR).

Minimal Silence Duration

When the "Stop" check box is selected, a silence on the input line is one of the conditions that determine the detection of a frame end.

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.

Detail of Character Mode Expert Mode

Expert Mode Communication

Expert mode is a set of commands that can be sent to the module to get extra features.

Address	Standard Symbol	Exchange Type	Type	Meaning
`%MWr.m.c.24`	CONTROL	Explicit	INT	Command signal, change protocol.
`%MWr.m.c.24.0`		Explicit	BOOL	Erase local counters.
`%MWr.m.c.24.4`		Explicit	BOOL	Modify the silence internal timings (`%MW30`). This value update may disturb the module if it's working.
`%MWr.m.c.24.5`		Explicit	BOOL	Modify the char mode end of frame byte 0 (`%MW26`) and byte 1 (`%MW27`)

Address	Standard Symbol	Exchange Type	Type	Meaning
%MWr.m.c.24.6		Explicit	BOOL	Change HALF/FULL DUPLEX modem management mode. ● If set simultaneously with RTS_ON (%MWr.m.c.24.10 works also with RTS_OFF %MWr.m.c.24.11 and use DTR if .8 or .9 is used) the half duplex modem mode is activated. ● If this bit is set but none of the RTS/DTR (neither %MWr.m.c.24.8, %MWr.m.c.24.9, %MWr.m.c.24.10, %MWr.m.c.24.11), the full duplex mode is activated. The %MW26 is used to set the StartDelay and %MW27 is used to set the EndDelay. So the bit %MW24.5 and %MW24.1 and %MW24.2 cannot be used simultaneously **NOTE**: The user may have to restore the correct state of the RTS/DTR signals after the command has been accepted.
%MWr.m.c.24.7		Explicit	BOOL	Save the Modbus slave address into the FLASH (%MW25).
%MWr.m.c.24.8	DTR_ON	Explicit	BOOL	Set the DTR signal (positive voltage)
%MWr.m.c.24.9	DTR_OFF	Explicit	BOOL	Reset the DTR signal (negative voltage)
%MWr.m.c.24.10		Explicit	BOOL	Set the RTS signal (positive voltage)
%MWr.m.c.24.11		Explicit	BOOL	Reset the RTS signal (negative voltage)

Address	Standard Symbol	Exchange Type	Type	Meaning
%MWr.m.c.24.12	TO_MODBUS_MASTER	Explicit	BOOL	switch to master mode
%MWr.m.c.24.13	TO_MODBUS_SLAVE	Explicit	BOOL	switch to slave mode
%MWr.m.c.24.14	TO_CHAR_MODE	Explicit	BOOL	Switch to character mode
%MWr.m.c.25		Explicit	INT	Modbus slave address to store in FLASH
%MWr.m.c.26		Explicit	INT	New EOF in char mode (eq %KW6) if %MW24.5 is set: • Bit 0: 1 byte 1 is set below, 0 no more byte 1 • Bit 1: 1 add the byte 1, 0 do not add the byte 1 • Bit2..7 : must be null.HIGH BYTE: the end of frame byte 1 StartDelay if %MW26.6 is set. Time to wait after the CTS is OK before to start to send the frame. It is useful for modem that requires extra time after CTS or do not manage the CTS signal (in this case the RTS must be connected to the CTS). This time is in millisecond, the precision is about 3ms. Can be performed only in RS232 mode.

Address	Standard Symbol	Exchange Type	Type	Meaning
%MWr.m.c.27		Explicit	INT	New EOF in char mode (eq %KW7) if %MW24.5 is set: • Bit 0: 1 byte 2 is set below, 0 no more byte 2 • Bit 1: 1 add the byte 2, 0 do not add the byte 2 • Bit2..7 : must be null.HIGH BYTE: the end of frame byte 2 StartDelay if %MW24.6 is set. Time to wait after having sent a frame, before to release the RTS signal to let enough time to the MODEM to completely send the frame before hand-up. This time is in millisecond, the precision is about 3ms. Can be performed only in RS232 mode.
%MWr.m.c.28		Explicit	INT	Reserved
%MWr.m.c.29		Explicit	INT	Reserved
%MWr.m.c.30		Explicit	INT	silence: Inter frame delay in milliseconds [0..10000]. The value used depends of the speed. If the value is smaller or greater than possible values, the lower limit or upper limit is applied, and the command is accepted. A value 0 means no silence.
%MWr.m.c.31		Explicit	INT	Reserved
%MWr.m.c.32		Explicit	INT	Reserved

Sample of Code

```
if HalfModemChar then
 HalfModemChar:=false;
  %MW0.9.0.24:=16#0440;(* switch to half duplex mode with RTS*)
  %MW0.9.0.26:=12;(* 12ms to wait before sending when CTS raise *)
 %MW0.9.0.27:=9; (* let RTS up 9ms after sending end *)
  write_cmd(%ch0.9.0);(* send command and data to the NOM channel*)
end_if;
```

Section 8.4
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:

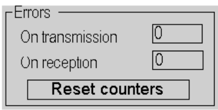
Step	Action
1	Access the configuration screen for Character Mode communication. (*see page 180*)
2	Select the "Debug" tab on the screen that appears.

Description of the Debug Screen

The debug screen consists of an Error zone and a Signals zone (if RS232).

The Error Zone

The Error zone looks like this:



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

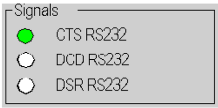
- **On transmission**: corresponds to the number of interruptions on transmission (image of `%MW4` word).
- **On reception**: corresponds to the number of interruptions on reception (image of `%MW5` word).

The Reset Counters button resets both counters to zero.

The Signals Zone

This zone is displayed only if RS232 is selected in configuration screen. If RS485 is selected in configuration screen, this window is not displayed at all.

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**: shows the activity of the DCD signal.
- **DSR RS232**: shows the activity of the DSR signal.

Chapter 9
BMX NOM 0200 Module Diagnostics

Section 9.1
BMX NOM 0200 Module Diagnostics

Subject of this Section

This section describes the diagnostics aspect in the implementation of a BMX NOM 0200 communication module.

What Is in This Section?

This section contains the following topics:

Topic	Page
Diagnostics of a BMX NOM 0200 Module	213
Detailed Diagnostics by Communication Channel	215

Diagnostics of a BMX NOM 0200 Module

At a Glance

The module diagnostics function displays anomalies when they occur, classified according to their category:

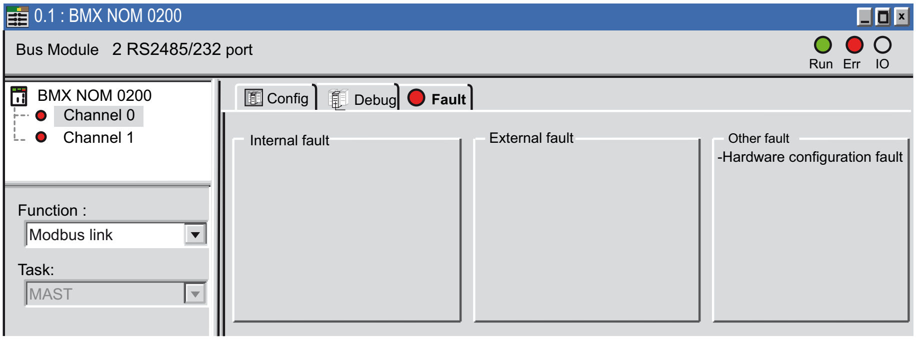
- **Internal detected error**:
  - module event
- **External event**:
  - Wiring control (broken-wire, overload or short-circuit)
- **Other anomalies**:
  - inoperative channel
  - configuration anomaly
  - module missing or off

A detected module error is indicated by a number of LEDs changing to red, such as:

- in the rack-level configuration editor:
  - the LED of the rack number
  - the LED of the slot number of the module on the rack
- in the module-level configuration editor:
  - the **Err** and **I/O** LEDs, depending on the type detected error
  - the **Channel** LED in the **Channel** field

Accessing the Module Diagnostic Screen

The table below shows the procedure for accessing the module diagnostic screen.

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Open the module debugging screen.</td></tr>
<tr><td>2</td><td>Click on the module reference in the channel zone and select the Fault tab.
Result: The list of module detected errors appears.


Note: It is not possible to access the module diagnostics screen if a configuration error, major breakdown error, or module missing error is detected. The following message then appears on the screen: " <code>The module is missing or different from that configured for this position.</code>"</td></tr>
</table>

Module Detected Errors List

The summary table below shows the various detected errors for a communication module:

Detected errors classification	Language objects
Internal fault: • Module detected failure	• %MWr.m.MOD.2.0
External fault: • Terminal block	• %MWr.m.MOD.2.2
Other fault: • Faulty channel(s) • Hardware configuration fault • Module missing or off	• %MWr.m.MOD.2.1 • %MWr.m.MOD.2.5 • %MWr.m.MOD.2.6

Detailed Diagnostics by Communication Channel

At a Glance

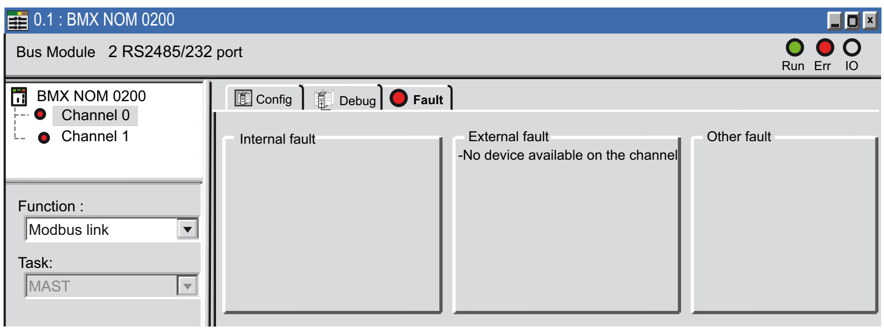
The channel Diagnostics function displays detected errors when they occur, classified according to their category:

- **Internal detected error**
  - self-tests in progress
- **External events**
  - device missing
  - device inoperative
  - serial-link communication time-out
- **Other detected errors**
  - line tool error
  - configuration error
  - communication loss
  - application error

A detected channel error is indicated in the **Debug** tab when the LED, located in the **Error** column, turns red.

Accessing the Channel Diagnostic Screen

The table below shows the procedure for accessing the channel diagnostic screen.

Step	Action
1	Open the module debugging screen.
2	For the inoperative channel, click on the button situated in the **Error** column. **Result**: The list of detected channel errors appears.  **Note:** Channel diagnostics information can also be accessed by program (instruction READ_STS).

Channel Detected Errors List

The summary table below shows the various detected errors for a configured serial link:

Detected errors classification	Language objects
Internal fault: • Self-tests in progress	• %MWr.m.c.2.4
External fault: • No device available on the channel • Device fault • Time-out error (CTS)	• %MWr.m.c.2.0 • %MWr.m.c.2.1 • %MWr.m.c.2.3
Other fault: • Line tool error • Hardware configuration fault • Problem communicating with the PLC • Application error	• %MWr.m.c.2.2 • %MWr.m.c.2.5 • %MWr.m.c.2.6 • %MWr.m.c.2.7

Chapter 10
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
10.1	Language Objects and IODDTs of Modbus and Character Mode Communications	218
10.2	General Language Objects and IODDTs for Communication Protocols	226
10.3	Language Objects and IODDTs Associated with Modbus Communication	230
10.4	Language Objects and IODDTs associated with Character Mode Communication	238
10.5	The IODDT Type T_GEN_MOD Applicable to All Modules	246
10.6	Language Objects and Device DDTs Associated with Modbus Communication	248

Section 10.1
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	219
Implicit Exchange Language Objects Associated with the Application-Specific Function	220
Explicit Exchange Language Objects Associated with the Application-Specific Function	221
Management of Exchanges and Reports with Explicit Objects	223

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 Unity Pro, Operating Modes)*.
- Using the Data Editor.

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.

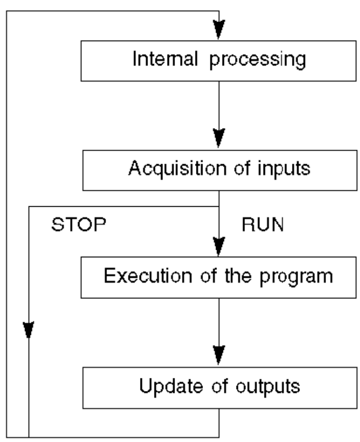
The outputs (`%Q` and `%QW`) are updated at the end of the task, only when the PLC is in RUN mode.

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 Unity Pro, I/O Management, Block Library)*: read status words
- WRITE_CMD *(see Unity Pro, I/O Management, Block Library)*: write command words

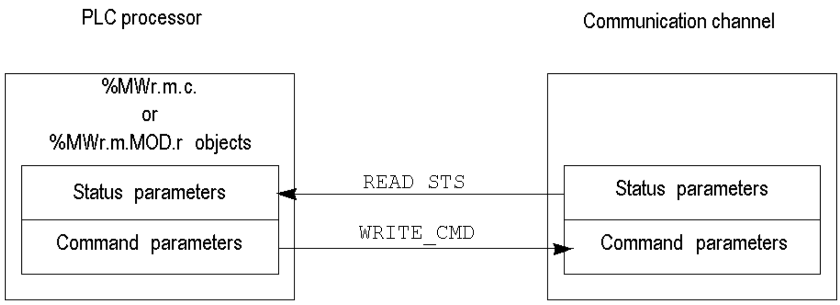
These exchanges apply to a set of `%MW` objects of the same type (status, commands or parameters) belonging to a channel.

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:



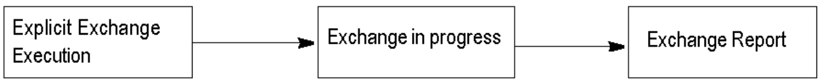
Managing Exchanges

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 Unity Pro, I/O Management, Block Library)*.
- The exchange report *(see Unity Pro, 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

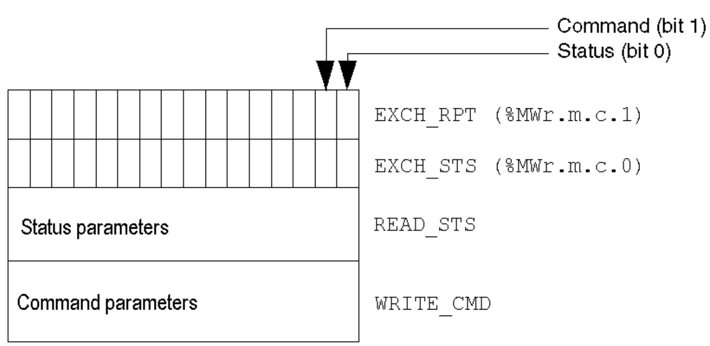
When data is exchanged between the PLC memory and the module, the module may require several task cycles to acknowledge this information.

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
STS_IN_PROGR	BOOL	R	Reading of channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2
RECONF_IN_PROGR	BOOL	R	Reconfiguration of the module in progress	%MWr.m.c.0.15

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 10.2
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	227
Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN	228

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 (%Ir.m.c.ERR):

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

Standard symbol	Type	Access	Meaning	Address
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 10.3
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	231
Details of the IODDTs Implicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT	232
Details of the IODDTs Explicit Exchange Objects of Types T_COM_MB_BMX and T_COM_MB_BMX_CONF_EXT	233
Details of language objects associated with configuration Modbus mode	236

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 235)*.

CH_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 detected error bit	`%Ir.m.c.ERR`

Word object in Modbus Master Mode

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

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

NOTE: When CTS is green in Punit, it means that `%IWr.m.c.0.0` is at 1 and that the voltage on this 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 (`%IWr.m.c.0`):

Standard symbol	Type	Access	Meaning	Address
LISTEN_ONLY	BOOL	R	Listen only mode	`%IWr.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 235)*. 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 function.	%MWr.m.c.3
PROTOCOL	INT	R	Byte 0 = 16#07 for Modbus slave function.	%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 254)*.

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
SAVE_CTRL_DATA	BOOL	R/W	Save the control data into the FLASH memory	`%MWr.m.c.24.7`
CONTROL_DATA	BOOL	R/W	Modbus slave address to store in the FLASH memory.	`%MWr.m.c.25`

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: ● 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: ● 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: ● 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.

Address	Type	Access	Meaning
%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.
%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 10.4
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	239
Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX	240
Details of IODDT Explicit Exchange Objects of Type T_COM_CHAR_BMX	241
Details of language objects associated with configuration in Character mode	244

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 (`%IWr.m.c.0`):

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

NOTE: When CTS is green in Punit, it means that `%IWr.m.c.0.0` is at 1 and that the voltage on this signal is positive. It is also appliable 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`
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 254)*.

%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	The byte 0 of this word corresponds to the transmission speed. This byte can take several values: ● 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) The byte 1 of this word corresponds to the format: ● 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	This word correcponds to the polarization type: ● 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	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.
%KWr.m.c.6	INT	R	Bit 0 of Byte 0 can have 2 values: ● 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 Bit 1 of Byte 0 can have 2 values: ● 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 Byte 1 of this word corresponds to the entered value of stop on reception of character 1 from 0 to 255.
%KWr.m.c.7	INT	R	Bit 0 of Byte 0 can have 2 values: ● 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 Bit 1 of Byte 0 can have 2 values: ● 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 Byte 1 of this word corresponds to the entered value of stop on reception of character 2 from 0 to 255.

Section 10.5
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 modules of Modicon M340 and X80 PLCs 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	%Ir.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

Section 10.6
Language Objects and Device DDTs Associated with Modbus Communication

Communication Device DDT Names

Introduction

This topic describes the Unity Pro **Communication Device DDT**.

The default device DDT name contains the following information:
- module Input and or output (***X*** symbol)
- module insertion number (**#** symbol)

Example: `MOD_COM_`***x***`_`**#**

The default device DDT type contains the following information:
- platform with:
  - M for Modicon M340 or Modicon M580
- device type (COM for discrete)
- function (NOM for BMX NOM 0200)
- direction:
  - IN
  - OUT

List of Implicit Device DDT

The table below shows the list of devices supported by Modicon M340 and Modicon M580, plus their corresponding device DDT name and type:

Device DDT Name	Device DDT Type	Modicon M340/M580 Devices
MOD_NOM_**#**	`T_M_COM_NOM`	BMX NOM 0200.3

Implicit Device DDT Description

The following table shows the `T_M_COM_NOM` status word bits:

Standard Symbol	Type	Meaning	Access
MOD_HEALTH	BOOL	0 = the module has a detected error	read
		1 = the module is operating correctly	

Standard Symbol	Type	Meaning	Access
MOD_FLT	BYTE	internal detected errors byte of the module	read
COM_CH	ARRAY [0...1] of `T_M_COM_NOM_CH`	array of structure	

The following table shows the `T_M_COM_NOM_CH`[0...1] status word bits:

Standard Symbol		Type	Bit	Meaning	Access
FCT_TYPE		WORD		0 = channel is not used	read
				1 = channel is used	
				3 = Character mode	
				7 = MODBUS slave	
CH_HEALTH		BOOL		0 = the channel has a detected error	read
				1 = the channel is operating correctly	
INPUT_SIGNALS [INT]	DCD	BOOL	0	Data Carrier Detect RS232 signal (applicable to BMX NOM 0200 module only)	read
	CTS	BOOL	2	clear to send RS232 signal	read
	DSR	BOOL	3	Data Set ready RS232 signal (applicable to BMX NOM 0200 module only)	read
COMMAND [INT]	STOP_EXCH	BOOL	0	rising edge at 1: All exchanges in progress are stopped.	read / write

Explicit Device DDT Instances Description

Explicit exchanges (Read Status) - only applicable to Modicon M340 and Modicon M580 I/O channels - are managed with `READ_STS_QX` EFB instance.

- Targeted channel address (`ADDR`) can be managed with ADDMX *(see Unity Pro, Communication, Block Library)* EF (connect `ADDMX OUT` to `ADDR`)
- READ_STS_QX *(see Unity Pro, I/O Management, Block Library)* output parameter (`STATUS`) can be connected to a "`T_M_xxx_yyy_CH_STS`" DDT instance (variable to be created manually), where:
  - `xxx` represents the device type
  - `yyy` represents the function

  Example: `T_M_COM_NOM_CH_STS`

The following table shows the `T_M_COM_NOM_CH_STS` status word bits:

Type	Type	Access
STRUCT	`T_M_COM_NOM_CH_STS`	

The following table shows the `T_M_COM_NOM_CH_STS` structure status word bits:

Standard Symbol		Type	Bit	Meaning	Access
CH_FLT [INT]	NO_DEVICE	BOOL	0	no device is working on the channel	read
	ONE_DEVICE_FLT	BOOL	1	inoperable device on the channel	read
	BLK	BOOL	2	terminal block fault detected (not connected)	read
	TO_ERR	BOOL	3	time out detected error (defective wiring)	read
	INTERNAL_FLT	BOOL	4	internal detected error or channel self-testing	read
	CONF_FLT	BOOL	5	configuration detected fault: different hardware and software configurations	read
	COM_FLT	BOOL	6	problem communicating with the PLC	read
	APPLI_FLT	BOOL	7	application detected error (adjustment or configuration detected error	read
PROTOCOL		BYTE		6 for Modbus Master, 3 for character mode	read
ADDRESS		BYTE		slave address	read

Chapter 11
Dynamic Protocol Switching

Subject of this Section

This chapter provides an introduction to dynamic switching between Modbus and Character Mode protocols.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Changing Protocol with BMX P34 1000/2000/2010/20102/2020 Processors	252
Changing Protocol with the BMX NOM 0200 Module	254

Changing Protocol with BMX P34 1000/2000/2010/20102/2020 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

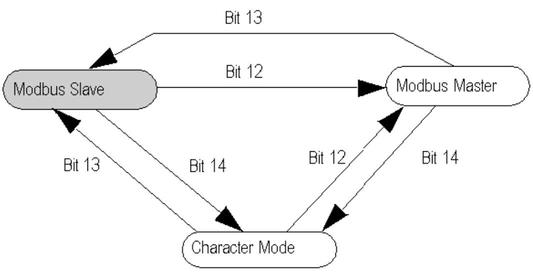
You must create first an IODDT variable linked to the processor's serial channel, then set to 1 the bit of word IODDT_VAR1.CONTROL (`%MWr.m.c.24`) that corresponds to the change of protocol desired:

- `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:



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

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.

Changing Protocol with the BMX NOM 0200 Module

General

This part describes how to change the protocol used by a BMX NOM 0200 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 either a T_COM_MB_BMX or a T_COM_MB_BMX CONF EXT type.

Changing Protocol: The Principle

You must create first an IODDT variable linked to the 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.

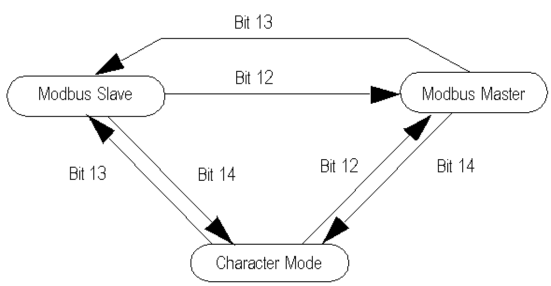
NOTE: A single bit can be set to 1 at a time: setting several bits to 1 will result in an error.

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 serial channel.

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).

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 from Modbus Slave to Modbus Master:
  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 3s
  - Number of retries set to 0
- Transfer from Modbus Slave/Master to Character Mode
  Switching to Character Mode is used to send AT commands to a modem. When a change is made from Modbus configuration to Character Mode configuration, transmission, signal and physical line parameters remain the same. Only the message end detection parameter specific to Character Mode is set to stop on reception of the `x0d` ending character.
- Transfer from Character Mode to Modbus Master and to Modbus Slave:
  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 (with the answer delay set to 3s and the number of retries set to 0) in order to send information about an event to another PLC. Once the message has been sent, the user returns to the Modbus Slave configuration: the slave number is set to the value stored in the FLASH memory or to 248 if none.

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.

NOTE: The default configuration of the module is the following: to be easily configurable by a computer like a PC, the channel 0 is configured in RS232 slave mode, and the channel 1 in RS485 mode. Other parameters are: 19200 bauds, RTU, even, 1 stop bit, no flow control, 1,75ms as default frame delay, slave number 248.

Part IV
Quick Start: BMX NOM 0200.4

Overview

This part describe how to configure the BMX NOM 0200.4 module as a Modbus RS-485 RTU slave in a modicon X80 drop over a Quantum PLC.

NOTE: For details on how to install and configure the BMX NOM 0200.4 in an M580 RIO drop, refer to *Configuring the BMX NOM 0200.4 Module in an X80 Drop (see Modicon M580, RIO Modules, Installation and Configuration Guide)*.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
12	Overview	259
13	Configuration in Unity Pro	265

Chapter 12
Overview

Prerequisites

To configure the BMX NOM 0200.4, you have to:

- Use the following firmware versions:
  - BMX CRA 312 10: SV ≥ V2.10
  - BMX NOM 0200: SV ≥ V1.4
- Interlink a 140 NOC 78• 00 to the Quantum 140 CRP 312 00

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Product Overview	260
Architecture Overview	261
Limitations	263

Product Overview

Presentation

The BMX NOM 0200.4 is a new generic module that you can find in the Unity Pro hardware catalog within communication family, for Quantum platforms only. First, you have to select a drop head BMX CRA 312 10.3.

Supported Protocols

For the BMX NOM 0200 modules:

- channel 0 is RS232 or RS485,
- and channel 1 is only RS485.

Declaring the BMX NOM 0200 module as a BMX NOM 0200.4 in Unity Pro allows you to configure the module as a:

- Modbus RTU slave on RS-485
- Modbus Serial RTU and ASCII Master on RS-232 and RS-485
- Character mode

Compatibility

This offer is compatible with the standard offer: BMX NOM 0200, BMX CRA 312 10, 140 CRP 312 00, and Quantum CPU.

Architecture Overview

Presentation

Modbus slave messages received by the BMX NOM 0200.4 are transferred to the head (BMX CRA 312 10.3) of the drop. Then, the head forwards the message on Ethernet I/O to the Quantum CPU.

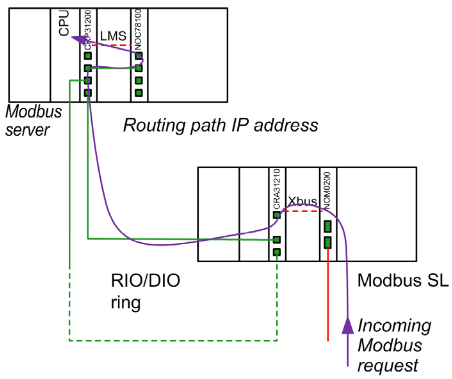
The Quantum 140 CRP 312 00 does not treat incoming Modbus messages. You have to plug an additional 140 NOC 78• 00 Ethernet module in the Quantum main rack and to interlink it with the CRP module.

After interlink, the drop head can send the Modbus messages to the 140 NOC 78• 00. The 140 NOC 78• 00 forwards the messages to the CPU.

For doing so, you must enter the IP address of the 140 NOC 78• 00 (Modbus server routing path *(see page 267)*) in the BMX CRA 312 10.3 configuration.

Illustration

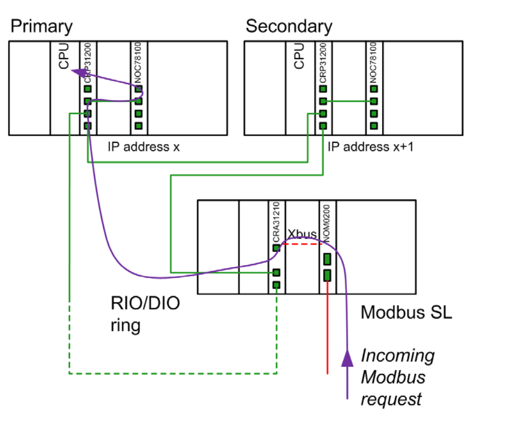
The Quantum CPU system treats the Modbus requests without any application program:

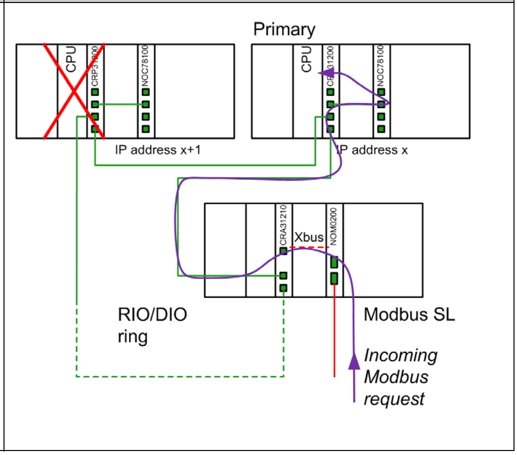


NOTE: The same path is used to route the Modbus response.

HSBY Specific Case

The 140 NOC 78• 00 IP address swaps in the case of PLC switch over. The Modbus requests are still forwarded to the operational CPU:

<table>
<tr><th>Nominal Mode</th><th>After Swich Over</th></tr>
<tr><td>


</td><td>


</td></tr>
</table>

NOTE: The Modbus client application manages the repetition of requests in case of a message loss that could occur during a PLC switch over.

Limitations

Maximum Configuration

This table shows the maximum configuration of the BMX NOM 0200.4:

Element	Maximum configuration
Master channel	4 per configured drop with a maximum of 36 expert channels per drop. **NOTE:** Each configured channel of the BMX NOM 0200.4 counts for an expert channel.
Drop	4 BMX NOM 0200.4 per drop.
Quantum system	16 BMX NOM 0200
Modbus frame length	256 bytes

IP Address

You must configure the IP address of the Modbus routing path for each BMX CRA 312 10.3 that supports a Modbus slave BMX NOM 0200.4 module.

Unity Pro provides no control on the consistency of those IP addresses.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Check that the IP address is really the one of the Quantum that supports the Modbus server.

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

Supported Protocols

Only Modbus RTU is supported as slave protocol.

Only RS-485 is supported when Modbus slave is selected.

Supported Modbus Function Codes

This table lists the Modbus function codes (FC) supported by the Quantum server:

Binding to -> Modbus FC:	Variable type	Code	Function
01	%M	0X	Read coil status (output bit)
02	%M	1X	Read input status (input bit)
03	%MW	4X	Read holding registers
05	%M	0X	Force single coil

Binding to -> Modbus FC:	Variable type	Code	Function
04	%MW	3X	Read input register
06	%MW	4X	Write single register
15	%M	0X	Write multiple coils
16	%MW	4X	Write multiple registers
23	%MW	4X	Read/write multiple registers

Chapter 13
Configuration in Unity Pro

Introduction

Most of the operating modes are identical to BMX NOM 0200 versions supported previously.

This chapter only details what is specific to the configuration of the BMX NOM 0200.4 module in Unity Pro.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Module Insertion	266
Module Configuration Screen	267

Module Insertion

Presentation

In a Quantum Ethernet I/O architecture, you can only insert the BMX NOM 0200.4 modules in a Modicon X80 remote drop, with BMX CRA 312 10.3 as adapter module.

Procedure

Follow this procedure to insert the BMX NOM 0200.4 module in a Modicon X80 remote drop:

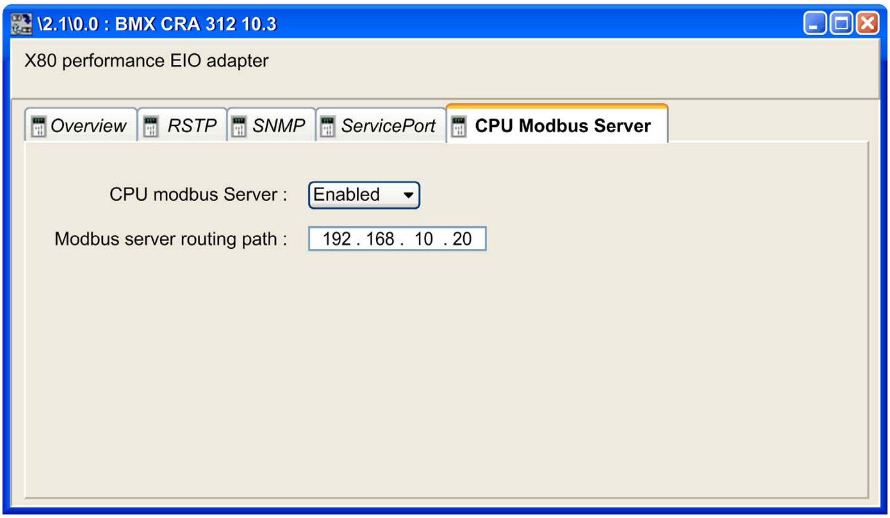
Step	Action
1	Insert the 140 CRP 312 00 module in the Quantum local rack.
2	Create on the **EIO Bus** an EIO Modicon X80 drop with a BMX CRA 312 10.3.
3	Insert the new BMX NOM 0200.4 module in the drop.
4	Insert the 140 NOC 78• 00 in the Quantum local rack.

Module Configuration Screen

Modbus Server Routing Path Configuration

This configuration is only possible in offline mode (PLC not connected).

Follow this procedure to set the Modbus server routing path:

Step	Action
1	Double-click the BMX CRA 312 10.3 module in the configurator editor.
2	Select the **Cpu Modbus Server** tab. 
3	Select **Enabled** in the **CPU modbus Server** field.
4	Enter the IP address of the 140 NOC 78• 00 in the **Modbus server routing path** field. The 140 NOC 78• 00 manages the routing of the Modbus frames between Ethernet I/O and the CPU.

Unity Pro provides no control on the consistency of those IP addresses.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION Check that the IP address is really the one of the Quantum that supports the Modbus server. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

NOTE: You must configure the IP address of the Modbus routing path for each BMX CRA 312 10.3 that supports a Modbus slave BMX NOM 0200.4 module.

Access to Channel Configuration Screens

Follow this procedure to access the channel configuration screens of the BMX NOM 0200.4 module:

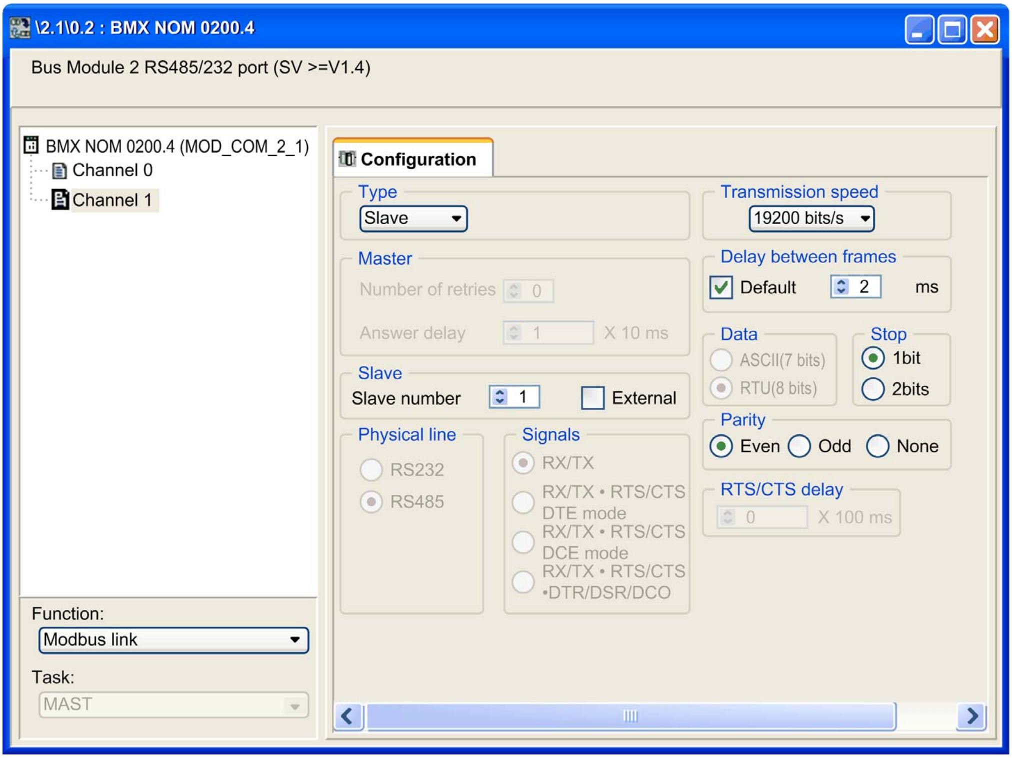
Step	Action
1	Open the BMX NOM 0200.4 subdirectory in the project browser.
2	Select the channel to configure. By default: ● **Channel 0** is configured with the **Character mode link** function. ● **Channel 1** is configured with the **Modbus link** function. **NOTE:** Some parameters are not accessible and are grayed out.

To configure Modbus serial communication in master mode, refer to chapter *Modbus Serial Communication for BMX NOM 0200 (see page 131)*.

To configure character mode communication, refer to chapter *Character Mode Communication for BMX NOM 0200 (see page 177)*.

Slave Modbus Link Configuration Screen

This figure shows the slave configuration screen of the BMX NOM 0200.4 module:



This table shows the default values of the parameters for Modbus slave configuration screen:

Configuration parameter	Default value
Type	Slave
Slave number	1
Physical line	RS-485 only
Signals	RX/TX only
Transmission speed	19200 bits/s
Delay between frames	2 ms
Data	8 bits only

Configuration parameter	Default value
Stop	1 bit
Parity	Even

NOTE: Modbus is a standard protocol. This module is based on a single mode of data exchange. When configuring Modbus serial communication in master mode, the slave parameters are grayed out and cannot be modified.

Part V
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
14	Description of the Application	273
15	Installing the Application Using Unity Pro	275
16	Starting the Application	303

Chapter 14
Description of the Application

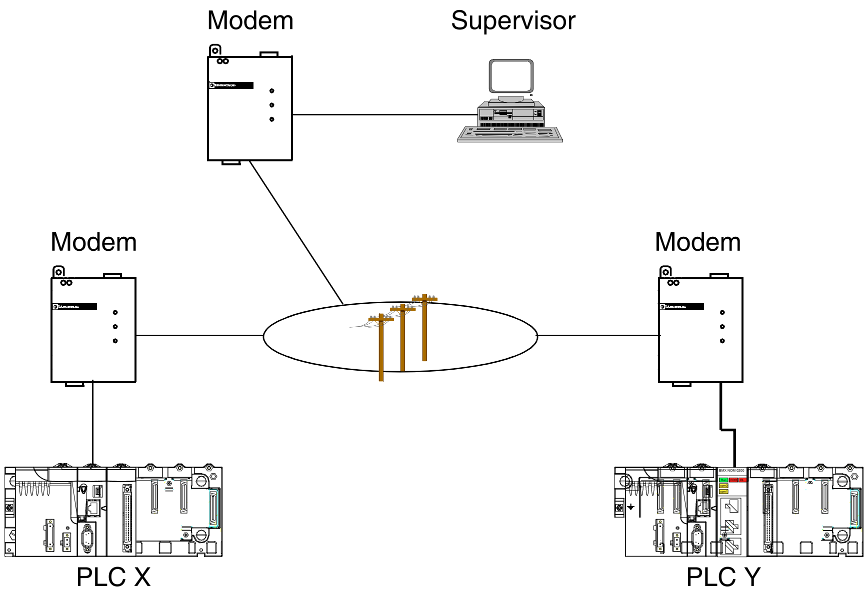
Overview of the Application

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 15
Installing the Application Using Unity Pro

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
15.1	Presentation of the Solution Used	276
15.2	Developing the Application	277

Section 15.1
Presentation of the Solution Used

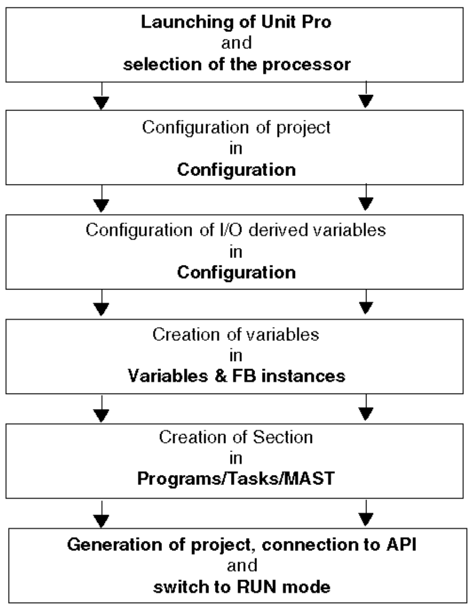
The Different Steps in the Process Using Unity Pro

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 15.2
Developing the Application

Subject of this Section

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

What Is in This Section?

This section contains the following topics:

Topic	Page
Creating the Project	278
Declaration of Variables	283
Using a Modem	287
Procedure for Programming	289
Programming Structure	291
Programming	294

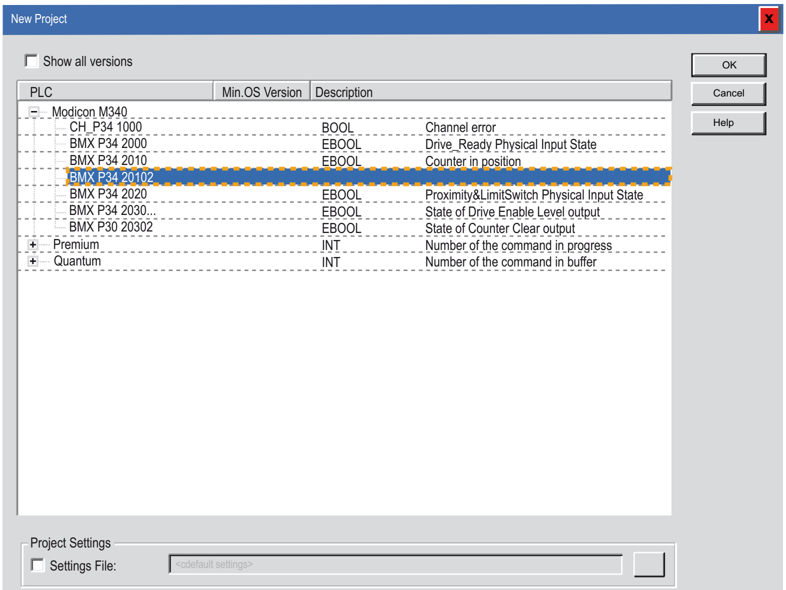
Creating the Project

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.

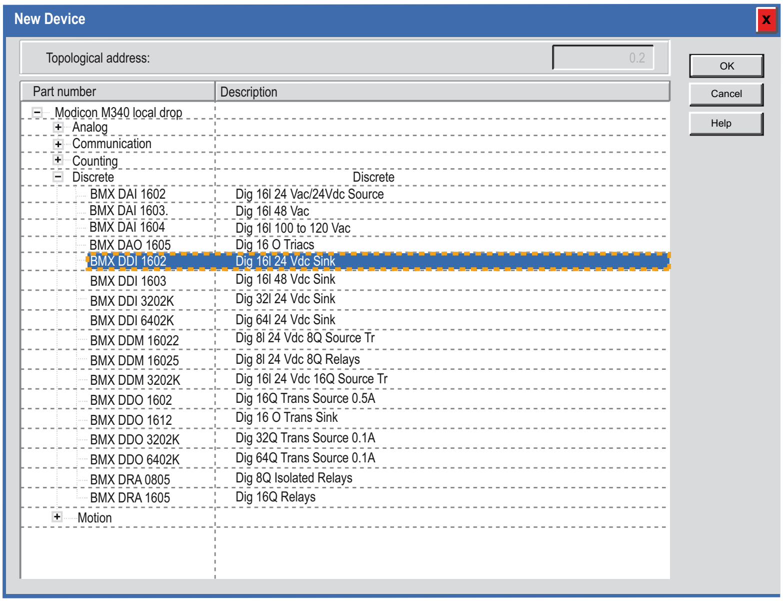
Procedure for Creating a Project

The table below shows the procedure for creating a project using Unity Pro.

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

Discrete Input Module Selection

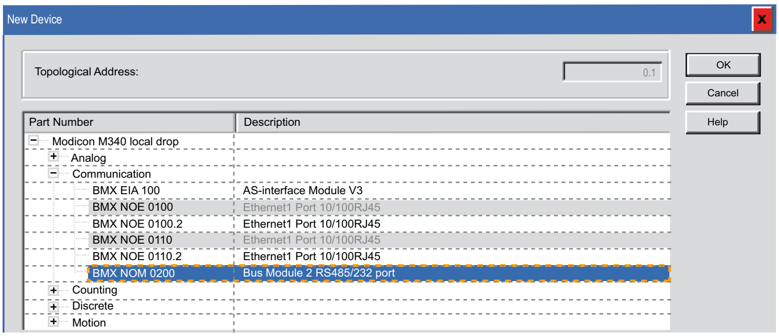
The table below shows the procedure for selecting the discrete module needed by the PLC X.

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>In the <code>Project Browser</code> double-click on <code>Configuration</code> then on <code>0:PLC Bus</code> and on <code>0:BMX XBP •••</code> (Where 0 is the rack number),</td></tr>
<tr><td>2</td><td>In the <code>PLC Bus</code> window, select a slot (for example slot 1) and double-click on it,</td></tr>
<tr><td>3</td><td>Choose the <code>BMX DDI 1602</code> discrete input module located in the <code>Discrete</code> modules list,

</td></tr>
<tr><td>4</td><td>Confirm with OK.</td></tr>
</table>

BMX NOM 0200 Module Selection

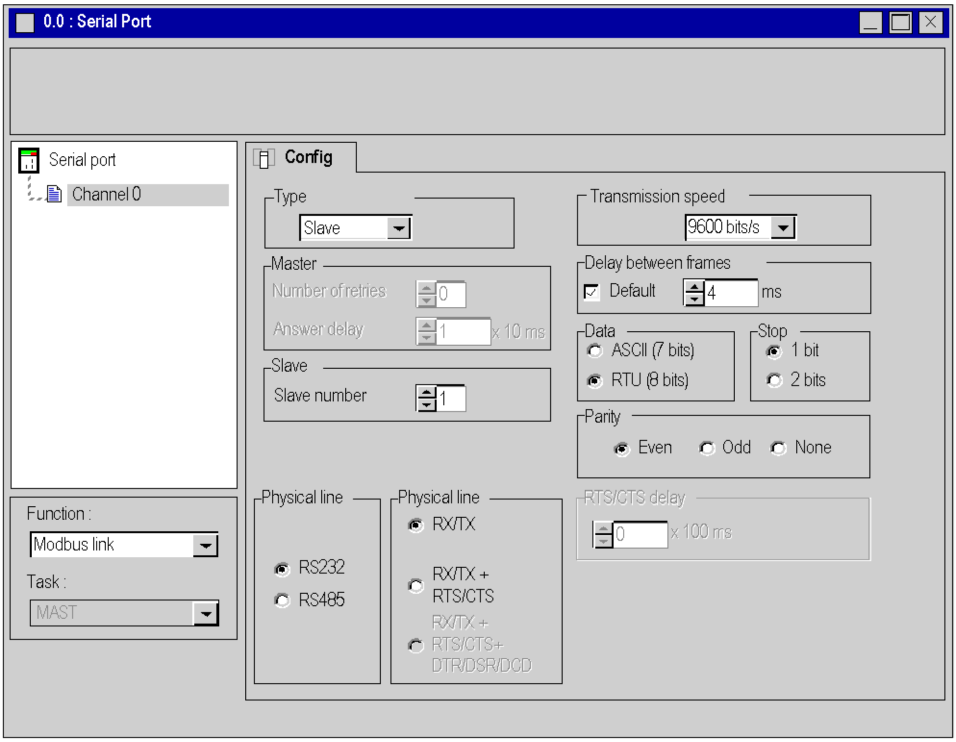
In this example, a BMX NOM 0200 module is used in the PLC Y for the serial link with the modem. Consequently it needs to be added to the project associated with the PLC Y.

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.

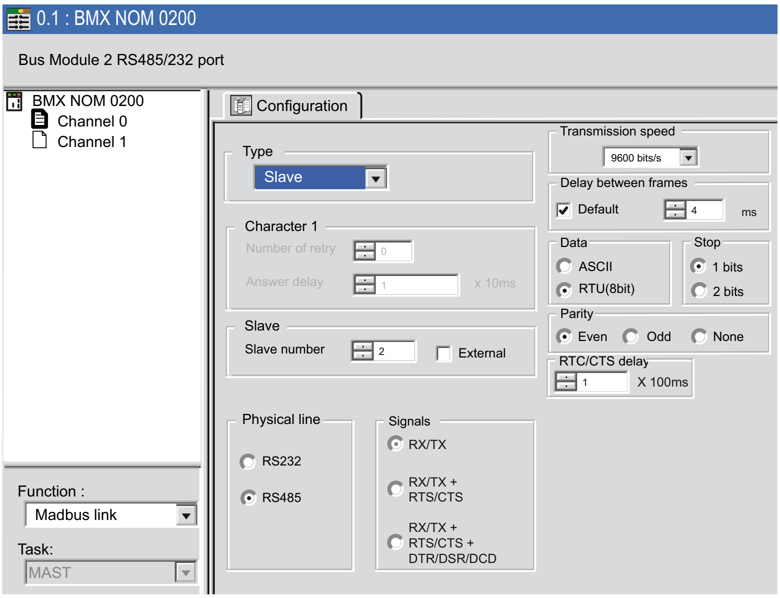
Processor Serial Port Configuration

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

Step	Action
1	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. 
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	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). 
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.
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, see Unity Pro online help (click on ?, then `Unity`, then `Unity Pro Software`, then `Operating Modes`, and `Data Editor`).

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.

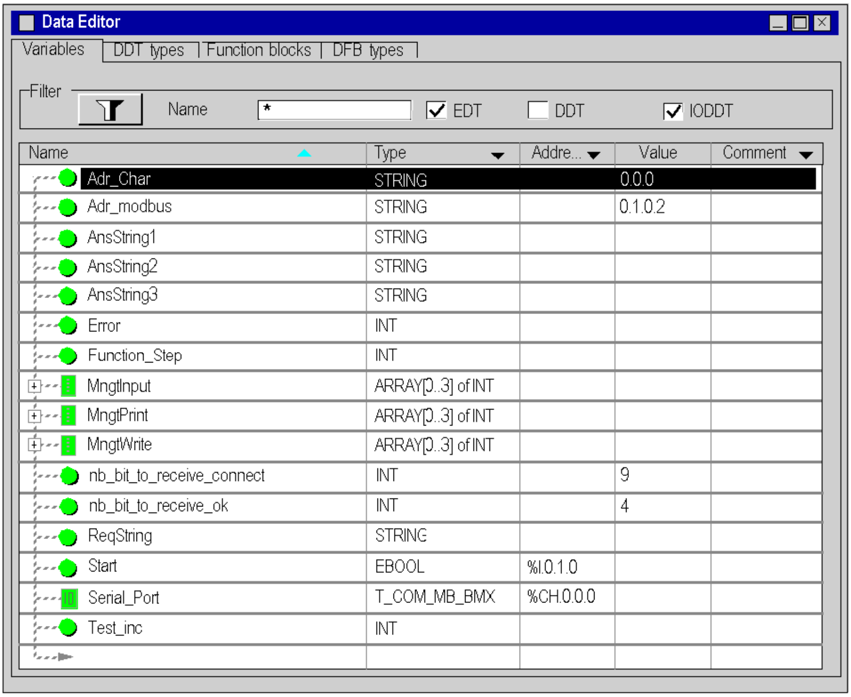
Variables Used for the Application

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.

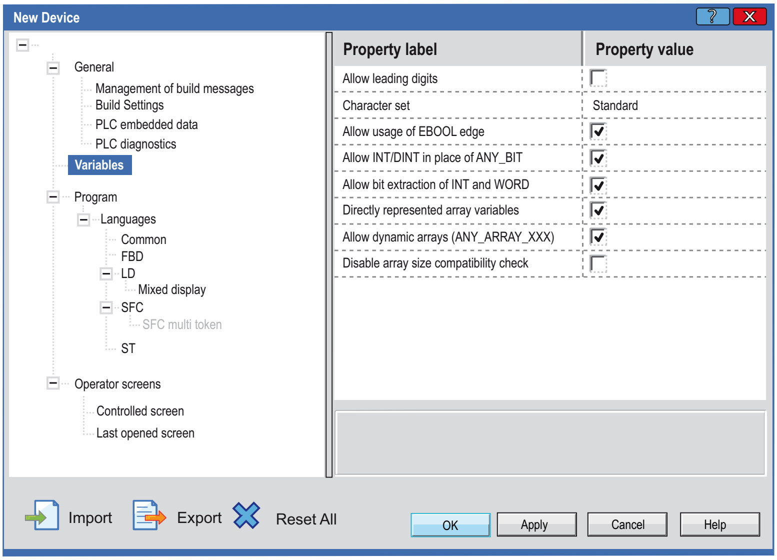
Variable	Type	Definition
nb_charac_to_receive_connect	INT	Number of character to receive: modem connexion
nb_charac_to_receive_ok	INT	Number of character to receive: modem confirmation message
ReqString	STRING	Modem answer.
Start	EBOOL	Starting mode (signal coming from channel 0 of the BMX DDI 1602 module).
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:



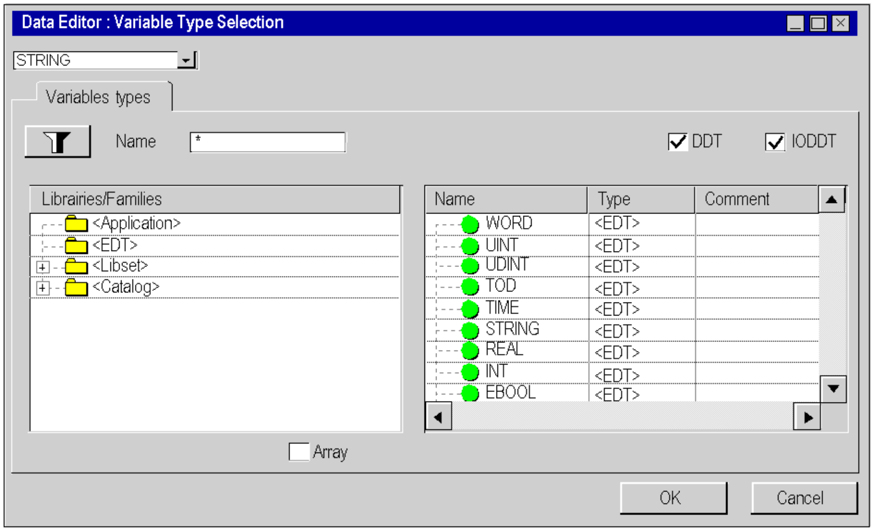
Declaring an Array Type

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 `ProjectBbrowser`, click on `Variables & FB instances`.
2	Click in the `Name` column and enter a name for the variable.

Step	Action
3	Double-click in the `Type` column and then click on the  button. The `Variable Type Selection` window opens: 
4	Choose the desired variable type (for example, click on `<EDT>` and select `INT`), then click into the Array checkbox. 
5	Modify the intervalle, then confirm with `OK`.

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: ATS2=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.	• Change the Slave Modbus mode to Character mode by sending a WRITE_CMD *(see page 290)* to the serial port. • For a modem transmission , send the HAYES command by using the PRINT_CHAR block to configure the modem *(see page 287)*. • 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 *(see page 287)*.
2	Master Modbus mode	• Switch to Modbus Master mode using the WRITE_CMD *(see page 290)* command. • Send data to write on the Slave PLC.
3	Reseting the communication port.	• Switch to Character mode using the WRITE_CMD *(see page 290)* command. • For a modem transmission, send the escape character, then send the disconnect command to send a disconnection message to the modem *(see page 288)* by using the PRINT_CHAR block. • Return to the starting mode of the serial port (Slave Modbus) using the WRITE_CMD *(see page 290)* 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.
	● If there is no error on the serial port
	    ● 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
	    ● 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.	

Step number	Step description	Element
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.
	● If there is no error on the serial port
	  ● 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
	  ● set Error to 45.
	  ● Go to step 130.	

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.
	● If there is no error on the serial port
	  ● 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
	  ● set Error to 65.
	  ● Go to step 130.	

Step number	Step description	Element
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.	
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.
	● If there is no error on the serial port
	  ● 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
	  ● 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 16
Starting the Application

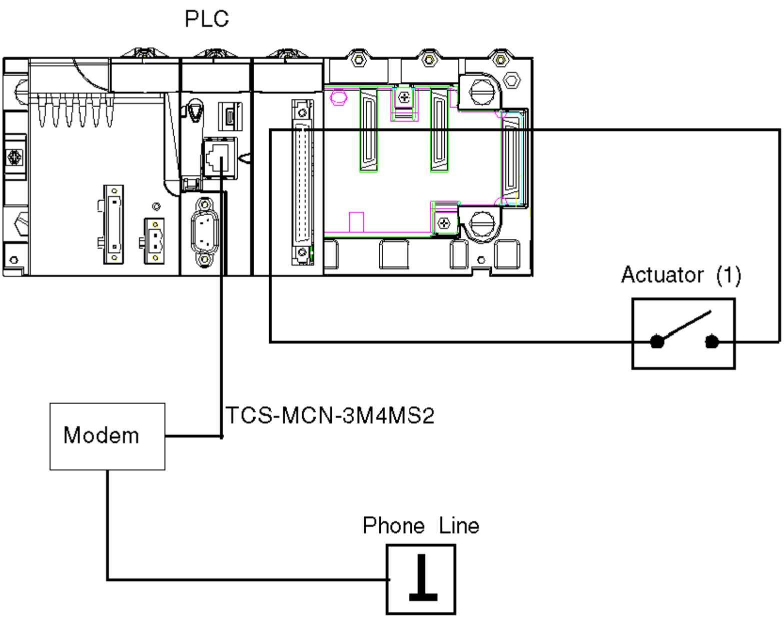
Execution of the Application in Standard Mode

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:



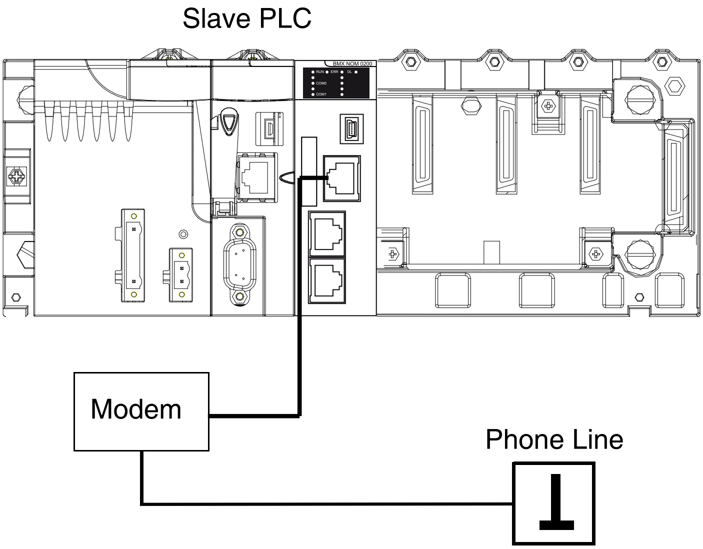
(1): the actuator is connected on the channel 0 of the discrete module.

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 transfering the project in standard mode:

Step	Action
1	In the `PLC` menu, click on `Standard Mode`,
2	In the `Build` menu, click on `Rebuild All Project`. Your project is generated and is ready to be transferred to the PLC.
3	In the `PLC` menu, click on `Connect`. You are now connected to the PLC.
4	In the `PLC` menu, click on `Transfer Project to PLC`. The `Transfer Project to PLC` window opens. Click on `Transfer`. 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 transfering the application in standard mode:

Step	Action
1	In the `PLC` menu, click on `Standard Mode`,
2	In the `Build` menu, click on `Rebuild All Project`. 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, Unity Pro indicates its location (click on the highlighted sequence).
3	In the `PLC` menu, click on `Connect`. You are now connected to the PLC.
4	In the `PLC` menu, click on `Transfer Project to PLC`. The `Transfer Project to PLC` window opens. Click on `Transfer`. 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 `PLC`, click on `Run`. The `Run` window opens. Click on `OK`. The application is now being executed on the PLC.
2	Disconnect the PC which is running Unity Pro software from the first slave PLC.
3	Connect the first slave PLC to a SR2MOD01 modem.

Glossary

!

%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.

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 interferance. 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.

U

Unity Pro

Schneider Automation PLC programming software.

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.

Index

B

BMXNOM0200, *19*
BMXNOM0200.4, *257*
BMXP341000, *19*
BMXP342000, *19*
BMXP342010, *19*
BMXP3420102, *19*
BMXP342020, *19*

C

Cabling, *59*
changing protocols, *252*, *254*
channel data structure for all modules
    T_GEN_MOD, *246*, *246*
channel data structure for character mode communication
    T_COM_CHAR_BMX, *240*, *241*
channel data structure for communication protocols
    T_COM_STS_GEN, *227*, *228*
channel data structure for modbus communication
    T_COM_MB_BMX, *232*, *233*
character mode, *103*
configuring character mode, *108*
configuring Modbus, *77*
connection devices, *39*

D

debugging character mode, *128*
debugging Modbus, *100*, *174*

G

grounding, *33*

I

INPUT_CHAR, *121*, *196*

M

M340
    hardened, *32*
    ruggedized, *32*
Modbus bus, *69*

P

parameter settings, *217*
PRINT_CHAR, *121*, *196*
programming character mode, *121*
programming Modbus bus, *89*

Q

quick start, *271*

T

T_COM_CHAR_BMX, *240*, *241*
T_COM_MB_BMX, *232*, *233*
T_COM_STS_GEN, *227*, *228*
T_GEN_MOD, *246*, *246*
T_M_COM_NOM, *248*

W

wiring accessories, *59*