

Modicon M340 with Unity Pro

Serial Link User Manual

07/2008 eng

Table of Contents



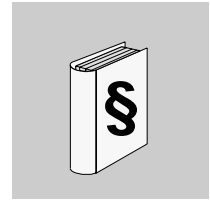
	Safety Information	7
	About the Book	9
Part I	Introduction to Modbus and Character Mode Communications	11
	At a Glance	11
Chapter 1	Introduction to Modbus and Character Mode Communications	13
	Introduction to Modbus and Character Mode Communications	13
Part II	Hardware Installation for Modbus and Character Mode Communications	15
	At a Glance	15
Chapter 2	Introduction to Serial Communications on the BMX P34 1000/2000/2010/2020 Processors	17
	Introduction to Serial Communications on the BMX P34 1000/2000/2010/2020 processors.	17
Chapter 3	Serial Communications Architectures	21
	At a Glance	21
	Modbus line adaptation and polarization.	22
	Connecting Modbus Devices.	24
	Connecting Data Terminal Equipment (D.T.E.).	27
	Connecting Data Circuit-Terminating Equipment (DCTE).	29
	Wiring Installation	31
Part III	Software Implementation of Modbus and Character Mode Communications	35
	At a Glance	35
Chapter 4	Installation Methodology	37

	Introduction to the Installation Phase	37
Chapter 5	Software Implementation of Modbus Communication	39
	At a Glance	39
5.1	General	40
	At a Glance	40
	About Modbus	41
	Performance	42
	How to Access the Serial Link Parameters for the BMX P34 1000/2000/2010/2020 Processors	44
5.2	Modbus Communication Configuration	48
	At a Glance	48
	Modbus Communication Configuration Screen	49
	Accessible Modbus Functions	51
	Default Values for Modbus Communication Parameters	52
	Configuration Screen for Modbus Communication	53
	Application-linked Modbus Parameters	55
	Transmission-linked Modbus Parameters	57
	Signal and Physical Line Parameters in Modbus	59
5.3	Modbus Communication Programming	61
	At a Glance	61
	Services Supported by a Modbus Link Slave Processor	62
	Services Supported by a Modbus Link Master Processor	63
5.4	Debugging Modbus Communication	69
	Modbus Communication Debug Screen	69
Chapter 6	Software Implementation of Communication Using Character Mode	71
	At a Glance	71
6.1	General	72
	At a Glance	72
	About Character Mode Communication	73
	Performance	74
6.2	Character Mode Communication Configuration	76
	At a Glance	76
	Character Mode Communication Configuration Screen	77
	Accessible Functions in Character Mode	79
	Default Values for Character Mode Communication Parameters	80
	Transmission Parameters in Character Mode	81
	Message End Parameters in Character Mode	83
	Signal and Physical Line Parameters in Character Mode	85
6.3	Character Mode Communication Programming	86
	Character Mode Communication Functions	86
6.4	Debugging Character Mode communication	92
	At a Glance	92
	Debug Screen for Character Mode communication	93

	Debugging Parameters in Character Mode	94
Chapter 7	Language Objects of Modbus and Character Mode Communications.	95
	At a Glance	95
7.1	Language Objects and IODDTs of Modbus and Character Mode Communications	96
	At a Glance	96
	Introduction to the Language Objects for Modbus and Character Mode Communications	97
	Implicit Exchange Language Objects Associated with the Application-Specific Function	98
	Explicit Exchange Language Objects Associated with the Application-Specific Function	99
	Management of Exchanges and Reports with Explicit Objects	101
7.2	General Language Objects and IODDTs for All Communication Protocols	104
	At a Glance	104
	Details of IODDT Implicit Exchange Objects of Type T_COM_STS_GEN	105
	Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN	106
7.3	Language Objects and IODDTs Associated with Modbus Communication	108
	At a Glance	108
	Details concerning Explicit Exchange Language Objects for a Modbus Function	109
	Details of the IODDT Implicit Exchange Objects of type T_COM_MB_BMX	110
	Details of the IODDT Explicit Exchange Objects of type T_COM_MB_BMX	111
	Details of language objects associated with configuration Modbus mode	113
7.4	Language Objects and IODDTs associated with Character Mode Communication	115
	At a Glance	115
	Details concerning Explicit Exchange Language Objects for Communication in Character Mode	116
	Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX	117
	Details of IODDT Explicit Exchange Objects of Type T_COM_CHAR_BMX	118
	Details of language objects associated with configuration in Character mode	120
7.5	The IODDT Type T_GEN_MOD Applicable to All Modules	122
	Details of the Language Objects of the IODDT of Type T_GEN_MOD	122
Chapter 8	Dynamic Protocol Switching	123

Changing Protocol	123
Part IV Quick start : example of Serial link implementation . .	127
At a glance	127
Chapter 9 Description of the application	129
Overview of the application	129
Chapter 10 Installing the application using Unity Pro	131
At a glance	131
10.1 Presentation of the solution used.	132
The different steps in the process using Unity Pro	132
10.2 Developping the application.	133
At a glance	133
Creating the project	134
Declaration of variables	138
Using a modem	143
Procedure for programming	145
Programming structure	147
Programming	150
Chapter 11 Starting the Application	157
Execution of Application in Standard Mode	157
Index	161

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, 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 an imminently hazardous situation, which, if not avoided, **will result** in death or serious injury.

WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

CAUTION

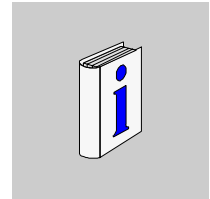
CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

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.

© 2008 Schneider Electric. All Rights Reserved.

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 1000/2000/2010/2020 processors.

Validity Note The data and illustrations found in this documentation are not binding. We reserve the right to modify our products in line with our policy of continuous product development.

The information in this document is subject to change without notice and should not be construed as a commitment by Schneider Electric.

Related Documents

Title of Documentation	Reference Number
Communication architectures and services	Included in the documentation CD-ROM

Product Related Warnings

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.

Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without the express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product.

For safety reasons and to ensure compliance with documented system data, only the manufacturer is authorized to perform repairs to components.

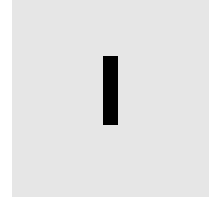
When controllers are used for applications with technical safety requirements, please follow the relevant instructions.

Failure to observe this warning about the product can result in injury or equipment damage.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

Introduction to Modbus and Character Mode Communications



At a Glance

In This Section

This section provides an introduction to Modbus and Character Mode communications.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Introduction to Modbus and Character Mode Communications	13

Introduction to Modbus and Character Mode Communications

1

Introduction to Modbus and Character Mode Communications

General

The serial link for BMX P34 1000/2000/2010/2020 processors supports two communication protocols:

- Modbus
- 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
 - ASCII

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 fails to respond within a given timescale.
- 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 the slave and awaits its response.
- The master sends a request to all the slaves without awaiting a reply (the general broadcast principle).

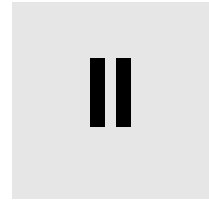
**Character Mode
Communication**

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.
-

Hardware Installation for Modbus and Character Mode Communications



At a Glance

In This Section

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

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
2	Introduction to Serial Communications on the BMX P34 1000/2000/2010/2020 Processors	17
3	Serial Communications Architectures	21

Introduction to Serial Communications on the BMX P34 1000/2000/2010/2020 Processors

2

Introduction to Serial Communications on the BMX P34 1000/2000/2010/2020 processors.

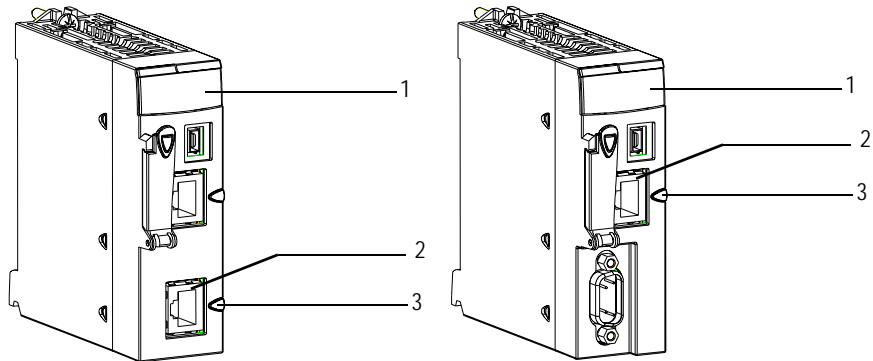
General

The BMX P34 1000/2000/2010/2020 processors enable communication via a serial link.

All these processors have an integrated communication channel dedicated to serial communications.

Introduction to the Processors

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



BMX P34 1000/2000/2020 Processors

BMX P34 2010 Processor

The BMX P34 1000/2000/2010/2020 processors are composed of the following elements:

Address	Description
1	Processor status LEDs on the front.
2	Integrated channel 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 the BMX P34 1000/2000/2010/2020 processors:

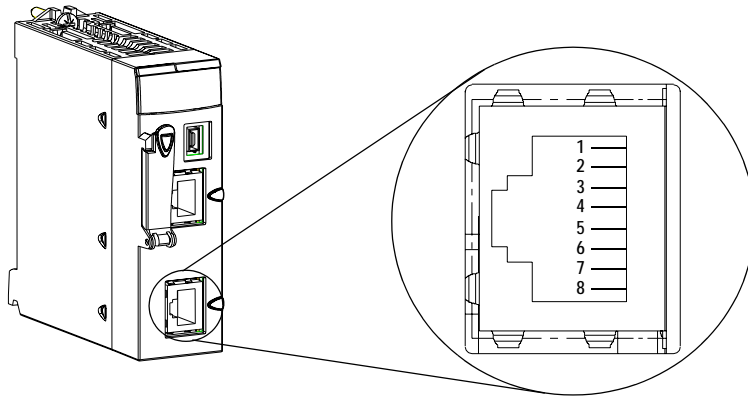
- LED flashing: serial communication is in progress.
- LED off: Serial communication is not in progress.

Introduction to the Serial Port

The properties of the serial communication channel for the BMX P34 1000/2000/2010/2020 processors are outlined in the table below:

Feature	Description
Channel number	Channel 0
Protocols supported	<ul style="list-style-type: none"> ● Modbus protocol (ASCII and RTU) ● Character Mode protocol
Connection	RJ45 female connector
Physical link	<ul style="list-style-type: none"> ● Non-isolated RS 485 2-wire serial link ● Non-isolated RS 232 serial link

The illustration below shows the RJ45 serial port on the BMX P34 1000/2000/2010/2020 processors:



The illustration below shows the pin assignment for the serial port on the BMX P34 1000/2000/2010/2020 processors:

1	RXD
2	TXD
3	RTS
4	D1
5	D0
6	CTS
7	Power Supply
8	Common
	Shielding

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

The pins used by the RS 232 serial link are as follows:

- 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 RS 485 serial link are as follows:

- 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: 5VDC/190mA

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 190mA: 5.25 V,
- Maximum voltage on CPU connector with no load: 5.5 V.

AC characteristics:

- capacitor charge: (on 5 V)
 - maximum 1 μ F ceramic capacitor
 - and 10 μ F tantalum ($Z=2.3u$)
- pump charge startup: (on 5 V)
 - 4 x 1 μ F ceramic capacitor
 - and 2 x 10 μ F tantalum

Note: The four-wire RS 232, two-wire RS 485 and two-wire RS 485 with power supply all use the same male connector, the RJ45. Only the signal cabling is different.

**Modbus Line
Electrical
Characteristics**

RS232 and 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.

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 a dynamic protocol switching. The polarization resistors value is 560 ohms.

In RS232 mode no polarization is required.

There is no built in line termination.

Serial Communications Architectures

3

At a Glance

Subject of this Chapter

This chapter provides an introduction to architectures that use serial communication on the BMX P34 1000/2000/2010/2020 processors, as well as the wiring to be installed.

What's in this Chapter?

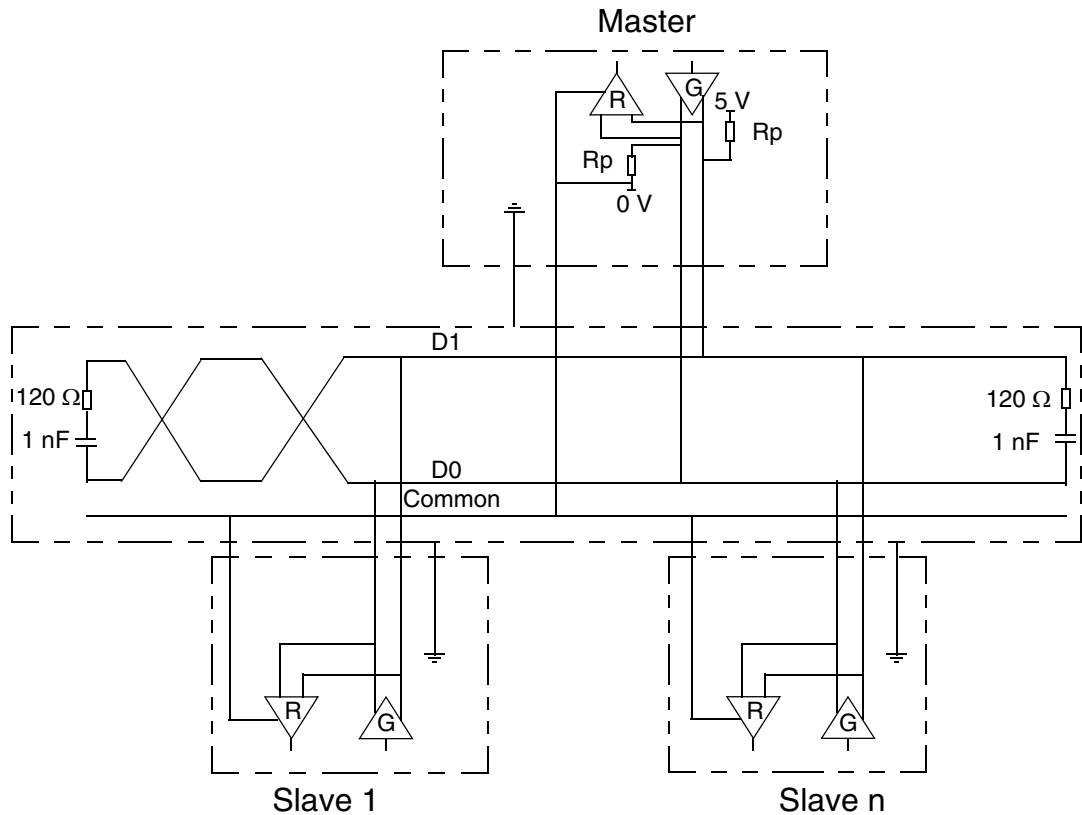
This chapter contains the following topics:

Topic	Page
Modbus line adaptation and polarization	22
Connecting Modbus Devices	24
Connecting Data Terminal Equipment (D.T.E.)	27
Connecting Data Circuit-Terminating Equipment (DCTE)	29
Wiring Installation	31

Modbus line adaptation and polarization

Overview

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



Line adaptation

line adaptation consist of two $120\ \Omega$ resistor and $1\ \text{nF}$ capacitor, placed at each end of the network (VW3 A8 306RC or VW3 A8 306 DRC). Don't place line adaptation at the end of a derivation cable.

Line polarization On Modbus line, polarization is needed for M340. It is automatically driven by M340 CPUs (see chapter above). If the M340 CPU is used as a slave on Modbus the polarization must be implemented by two 450 to 650 Ω resistors (R_p) connected on the RS485 balanced pair (if not done on master):

- a pull-up resistor to a 5 V voltage on the D1 circuit,
- a pull-down resistor to the common circuit on D0 circuit.

For an example, see the multipoint example **Connecting non-serial-Link-powered Modbus devices** (see *Connecting Non-Serial-Link-Powered Modbus Devices*, p. 25) below.

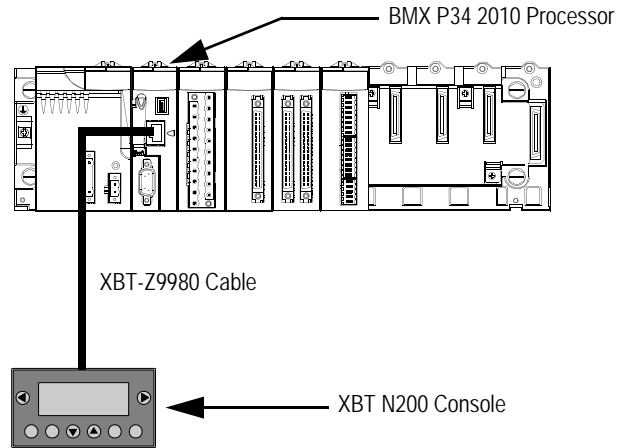
Connecting Modbus Devices

General

The pages that follow present two examples of Modbus device connection and one Modbus serial link architecture.

Connecting Serial-Link-Powered Modbus Devices

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 a slave,
- The XBT N200 human-machine interface is configured as a master.

The XBT-Z9980 cable has the following properties:

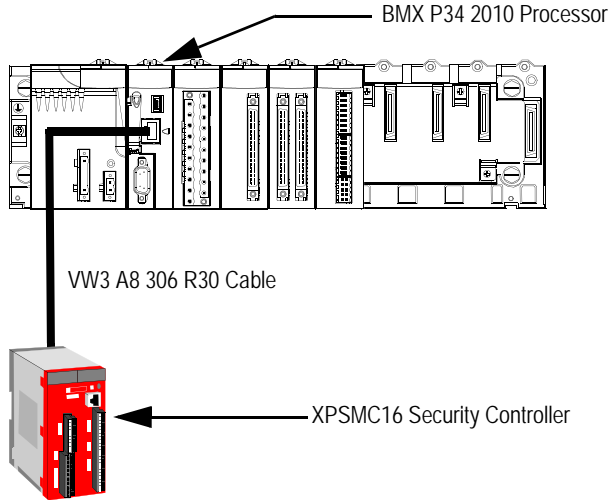
- Connection: 2 male RJ45 connectors
 - Wiring: 2 wires for the RS 485 physical line and 2 for the serial link power supply.
-

Connecting Non-Serial-Link-Powered Modbus Devices

This architecture consists of the following elements:

- A BMX P34 2010 processor configured as a master,
- An XPSMC16 security controller is configured as a slave.

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



The devices are configured as follows:

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

The VW3 A8 306 R30 cable has the following properties:

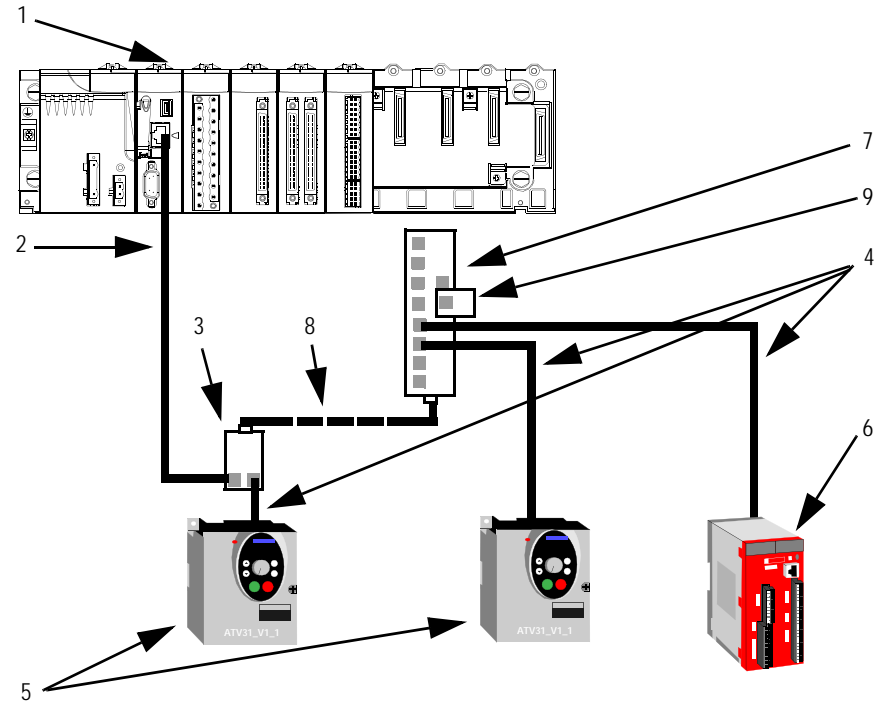
- Connection: 2 male RJ45 connectors
- Wiring: 2 wires for the RS 485 physical line

Modbus Serial Link Architecture

The Modbus serial link architecture consists of the following elements:

- A BMX P34 2010 processor, configured as a master.
- An XPSMC16 security controller, configured as a 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 security controller
- 7 LU9 GC3 splitter block
- 8 TSXCSAx00 Cable
- 9 VW3 A8 306 R Cable

Connecting Data Terminal Equipment (D.T.E.)

General

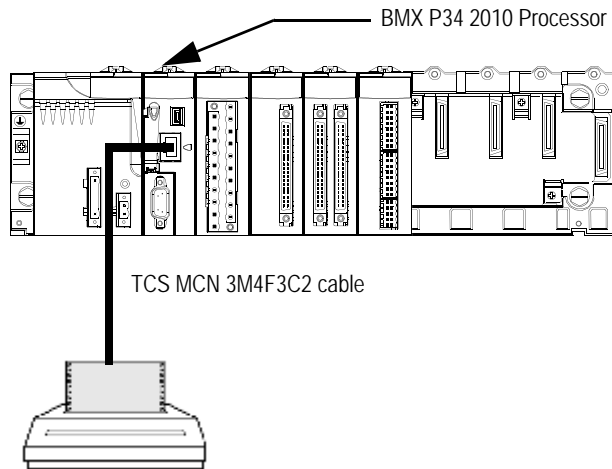
Data terminal equipment is the term used to describe devices such as:

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

All data terminal equipment is connected to a BMX P34 1000/2000/2010/2020 processor by a serial cross cable using the RS 232 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.

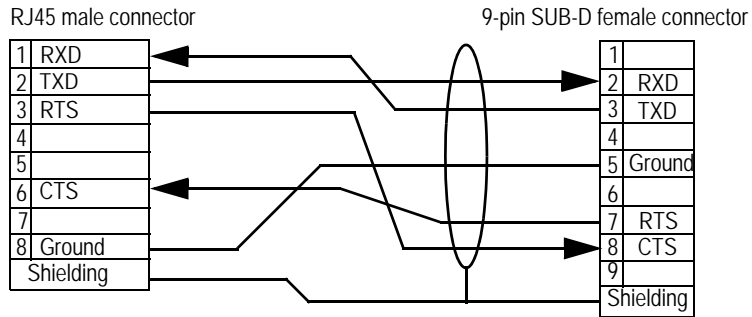
Note: Only one item of data terminal equipment may be connected to each BMX P34 1000/2000/2010/2020 processor.

RS 232 serial cross cable

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

- RJ45 male
- Nine-pin SUB-D female

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



Connecting Cables and Accessories

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

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

Connecting Data Circuit-Terminating Equipment (DCTE)

General

Data circuit-terminating equipment (DCTE) is the term used to describe devices such as modems.

All data circuit-terminating equipment is connected to a BMX P34 1000/2000/2010/2020 processor by serial direct cable using the RS 232 physical link.

Modem Characteristics

Should you wish to connect a modem to the serial port of a BMX P34 1000/2000/2010/2020 processor, the modem must have the following characteristics:

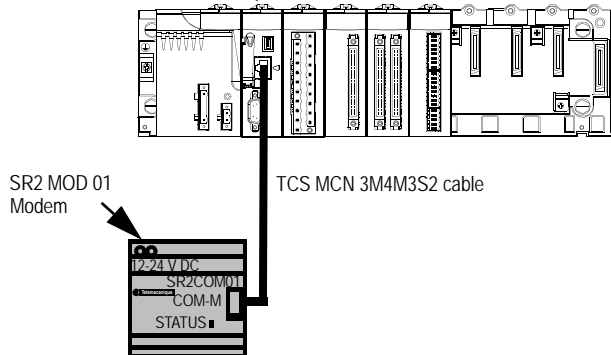
- Support 10 or 11 bits per character if the terminal port is used in Modbus protocol:
 - 7 or 8 data bits
 - 1 or 2 stop bits
 - Odd, even or no parity
- Operate without a data carrier check.
- Accept an incoming telephone call while characters arrive at its RS 232 serial port (if a modem/telephone network is used in response mode on a terminal port configured in Modbus Master mode).

Note: You are advised to check with your dealer that the modem you plan to use has the above-mentioned characteristics.

Connecting Data Circuit-Terminating Equipment

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

BMX P34 2010 Processor



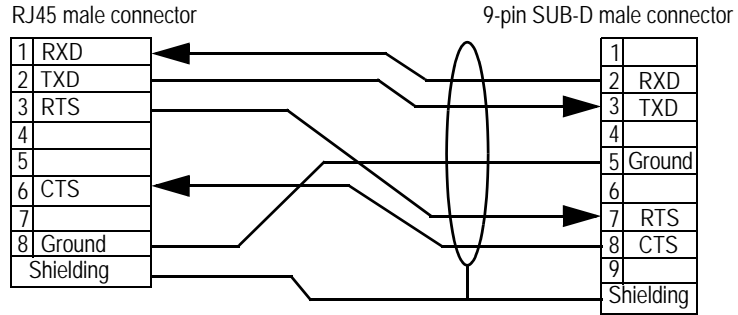
Note: In Modbus protocol, the waiting time must be between 100 and 250 ms.

RS 232 Serial Direct Cable

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

- RJ45 male
- Nine-pin SUB-D male

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



Connecting Cables and Accessories

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

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

Wiring Installation

General

In order to set up a serial link on a BMX P34 1000/2000/2010/2020 processor, several cables and accessories are required.

Cables

The table below shows the available cables that are compatible with serial communication on BMX P34 1000/2000/2010/2020 processors:

Designation	Length	Characteristics	Product reference
Two-wire RS 485 cable	1 m	<ul style="list-style-type: none"> ● One RJ45 male connector ● One nine-pin SUB-D male connector 	VW3 A58 306 R10
Two-wire RS 485 cable	3 m	<ul style="list-style-type: none"> ● One RJ45 male connector ● One nine-pin SUB-D male connector 	VW3 A58 306 R30
Two-wire RS 485 cable	3 m	<ul style="list-style-type: none"> ● One RJ45 male connector ● One bare end 	VW3 A8 306 D30
Two-wire RS 485 cable	0.3 m	Two RJ45 male connectors	VW3 A8 306 R03
Two-wire RS 485 cable	1 m	Two RJ45 male connectors	VW3 A8 306 R10
Two-wire RS 485 cable	3 m	Two RJ45 male connectors	VW3 A8 306 R30
Two-wire RS 485 cable	3 m	<ul style="list-style-type: none"> ● One RJ45 male connector ● One fifteen-pin SUB-D male connector 	VW3 A8 306
RS 485 cable for serial-link-powered devices	3 m	Two RJ45 male connectors	XBT-Z9980
RS 485 adapter for non-standard devices	3 m	<ul style="list-style-type: none"> ● One RJ45 male connector ● One 25-pin SUB-D female connector 	XBT-Z938
Two-wire RS 485 double shielded twisted pair cable	100 m	Two bare ends	TSX CSA 100
Two-wire RS 485 double shielded twisted pair cable	200 m	Two bare ends	TSX CSA 200
Two-wire RS 485 double shielded twisted pair cable	500 m	Two bare ends	TSX CSA 500
Four-wire RS 232 cable for data terminal equipment	3 m	<ul style="list-style-type: none"> ● One RJ45 male connector ● One nine-pin SUB-D female connector 	TCS MCN 3M4F3C2
Four-wire RS 232 cable for data circuit-terminating equipment (DCTE)	3 m	<ul style="list-style-type: none"> ● One RJ45 male connector ● One nine-pin SUB-D male connector 	TCS MCN 3M4M3S2

Connecting Accessories

The table below shows the available connecting accessories that are compatible with serial communication on BMX P34 1000/2000/2010/2020 processors:

Designation	Characteristics	Product reference
Modbus splitter block	<ul style="list-style-type: none"> • Ten RJ45 connectors • One screw terminal block 	LU9 GC3
Isolated Modbus splitter block	<ul style="list-style-type: none"> • Two RJ45 connectors • One screw terminal block 	TWDXCAISO
Modbus splitter block	Three RJ45 connectors	TWDXCAT3RJ
Modbus branch T-connector	<ul style="list-style-type: none"> • Two RJ45 connectors • On-board 0.3 m cable with RJ45 connector at end 	VW3 A8 306 TF03
Modbus tap T-connector	<ul style="list-style-type: none"> • Two RJ45 connectors • On-board 1 m cable with RJ45 connector at end 	VW3 A8 306 TF10
RC line-end adaptation for RJ45 connectors	<ul style="list-style-type: none"> • Resistance of 120 Ω • Capacity of 1 nF 	VW3 A8 306 RC
RC line-end adaptation for screw terminal block	<ul style="list-style-type: none"> • Resistance of 120 Ω • Capacity of 1 nF 	VW3 A8 306 DRC
Adapter for non-standard devices	<ul style="list-style-type: none"> • Two 25-pin SUB-D male connectors • For XBT G*** devices. 	XBT ZG999
Adapter for non-standard devices	<ul style="list-style-type: none"> • One 25-pin SUB-D male connector • One nine-pin SUB-D male connector • For XBT G*** devices 	XBT ZG909
Branching device	<ul style="list-style-type: none"> • Three screw terminal blocks • RC line end adaptation 	TSX SCA 50
Subscriber socket	<ul style="list-style-type: none"> • One fifteen-pin SUB-D male connector • Two screw terminal blocks • RC line end adaptation 	TSX SCA 62
Adapter for data terminal equipment	<ul style="list-style-type: none"> • One nine-pin SUB-D male connector • One 25-pin SUB-D female connector 	TSX CTC 07
Adapter for data terminal equipment	<ul style="list-style-type: none"> • One nine-pin SUB-D male connector • One 25-pin SUB-D male connector 	TSX CTC 10
Adapter for data circuit-terminating equipment (DCTE)	<ul style="list-style-type: none"> • 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.

**XBT Z998 and
XBT Z938 Cables**

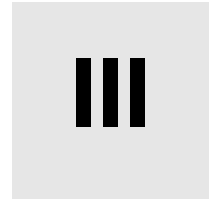
The XBT Z998 cable consists of a two-wire RS 485 link and a protected two-wire 5 VDC/190 mA power supply. This cable is used to link devices powered by the Modbus serial link. The devices that may be connected using this cable are the graphical user interface terminals with the following product references:

- XBT N200
- XBT N400
- XBT R400

The XBT Z938 cable consists of a two-wire RS 485 link. This cable can be used to connect the following graphical user interface terminals:

- XBT N410
 - XBT N401
 - XBT NU400
 - XBT R410
 - XBT R411
 - XBT G**** with an XBT ZG999 adapter
 - XBT GT**** with an XBT ZG909 adapter
-

Software Implementation of Modbus and Character Mode Communications



At a Glance

In This Section

This section provides an introduction to the software implementation of Modbus and Character Mode communications using Unity Pro software.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	Installation Methodology	37
5	Software Implementation of Modbus Communication	39
6	Software Implementation of Communication Using Character Mode	71
7	Language Objects of Modbus and Character Mode Communications	95
8	Dynamic Protocol Switching	123

Installation Methodology

4

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	Processor declaration	Offline
	Configuration of the processor's serial port	
Declaration of variables	Declaration of the IODDT-type variables specific to the processor 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	Transfer project to PLC	Online
Debug	Project debugging from debug screens and animation tables	Online
Documentation	Creating a documentation file and printing the miscellaneous information relating to the project.	Online
How it Works	Display 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	Processor declaration	Offline
	Configuration of the processor's serial port	
Declaration of variables	Declaration of the IODDT-type variables specific to the processor 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	Transfer project to simulator	Online
Simulation	Program simulation without inputs/outputs	Online
Adjustment/ Debugging	Project debugging from debug screens and animation tables	Online
	Modifying the program and adjustment parameters	
Legend: (1) These phases may also be performed online.		

Configuration of Processors

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

Software Implementation of Modbus Communication

5

At a Glance

Subject of this Chapter

This chapter presents the software implementation process for Modbus communication.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
5.1	General	40
5.2	Modbus Communication Configuration	48
5.3	Modbus Communication Programming	61
5.4	Debugging Modbus Communication	69

5.1 General

At a Glance

Subject of this Section

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

What's in this Section?

This section contains the following topics:

Topic	Page
About Modbus	41
Performance	42
How to Access the Serial Link Parameters for the BMX P34 1000/2000/2010/2020 Processors	44

About Modbus

Introduction

Communicating via Modbus enables data exchange between all devices connected to the bus. The Modbus protocol 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).

 WARNING
IMPROPER COMMUNICATION PORT USAGE
Communication ports should be used for non-critical data transfers only.
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.

Definition of "Exchange Time"

Exchange time is the time that passes between the creation of an exchange and the end of that exchange. It therefore includes 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 error occurs.
 - Time-out expires.
-

Exchange Times 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 times 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 processors.

Exchange times on the BMX P34 1000 processor are 10% lower than those on the BMX P34 2000/2010/2020 processors.

**Exchange Times
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/s	Cycle time in ms	Exchange times 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 2000/2010 and BMX P34 2020 processors.

Exchange times on the BMX P34 1000 processor are 10% lower than those on the BMX P34 2000/2010/2020 processors.

**Accuracy of
Measurements**

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

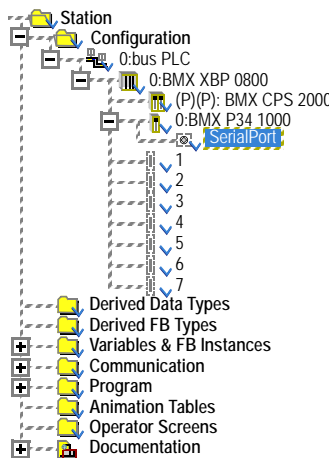
How to Access the Serial Link Parameters for the BMX P34 1000/2000/2010/2020 Processors

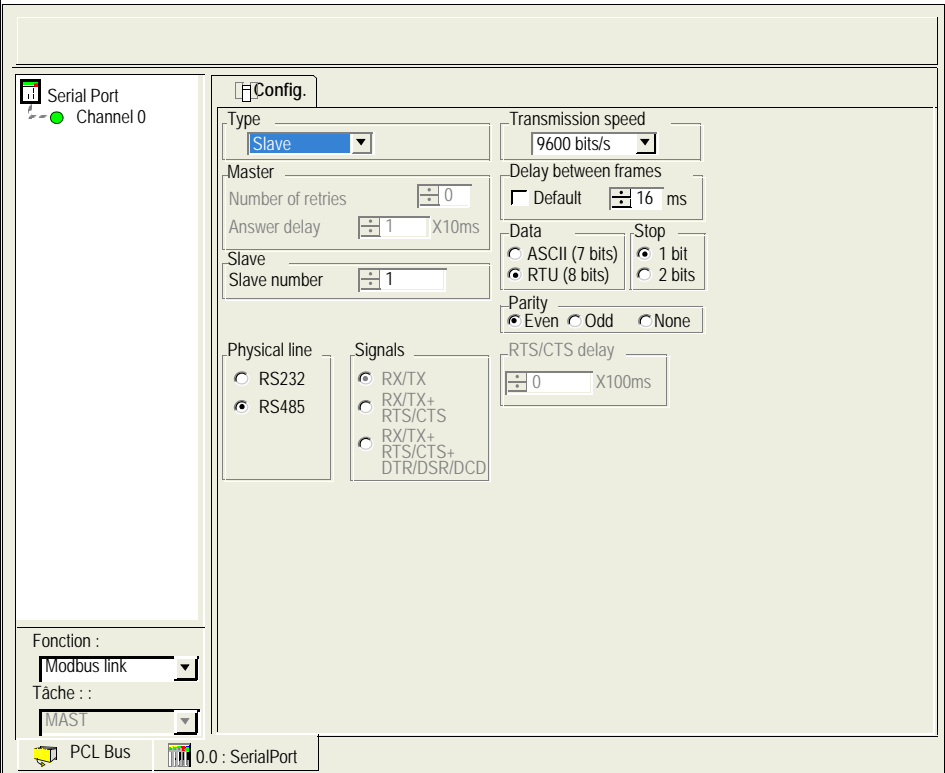
At a Glance

The pages that follow explain how to access the serial port configuration screen for the BMX P34 1000/2000/2010/2020 processors as well as the general elements of Modbus and Character Mode link configuration and debug screens.

How to Access the Serial Link

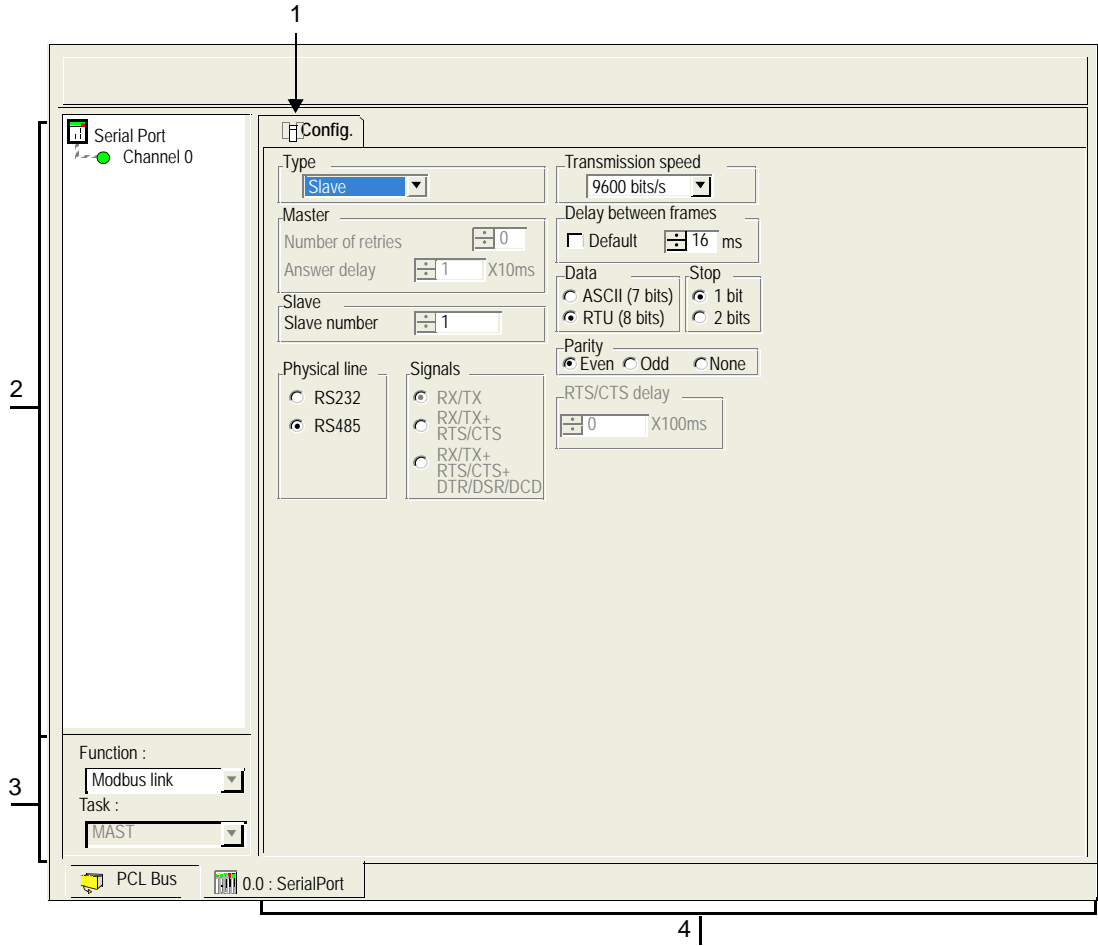
The table below describes the procedure for accessing the serial link of a BMX P34 1000/2000/2010/2020 processor:

Step	Action
1	<p>In the project browser, open the following directory: <i>Station\Configuration\0: PLC bus\0: rack reference\0: processor reference\SerialPort</i>.</p> <p>Result: the following screen appears:</p> 

Step	Action
2	<p>Double-click on the Serial Port sub-directory. Result: the following screen appears:</p> 

Description of the Configuration and Debug Screens

The figure below shows a configuration screen for Modbus communication:



Description The following table shows the different elements of the configuration and debug screens:

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: <ul style="list-style-type: none"> ● Configuration ● Debug screen accessible in online mode only.
2	Channel Zone	Enables you to: <ul style="list-style-type: none"> ● choose between the serial port and channel 0 by clicking on one or the other. ● display the following tabs by clicking on the serial port: <ul style="list-style-type: none"> ● "Description", which gives the characteristics of the device. ● "I/O Objects", (See Unity Pro 3.0: Operating Modes) which is used to presymbolize the input/output objects. ● display the following tabs by clicking on the channel: <ul style="list-style-type: none"> ● Configuration ● Debugging ● display the channel name and symbol defined by the user using the variables editor.
3	General Parameters Zone	This enables you to choose the general parameters associated with the channel: <ul style="list-style-type: none"> ● Function: the available functions are Modbus and Character Mode. The default configuration is with the Modbus function. ● Task: defines the MAST 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.

5.2 Modbus Communication Configuration

At a Glance

Subject of this Section This section describes the software configuration process for Modbus communication.

What's in this Section? This section contains the following topics:

Topic	Page
Modbus Communication Configuration Screen	49
Accessible Modbus Functions	51
Default Values for Modbus Communication Parameters	52
Configuration Screen for Modbus Communication	53
Application-linked Modbus Parameters	55
Transmission-linked Modbus Parameters	57
Signal and Physical Line Parameters in Modbus	59

Modbus Communication Configuration Screen

General

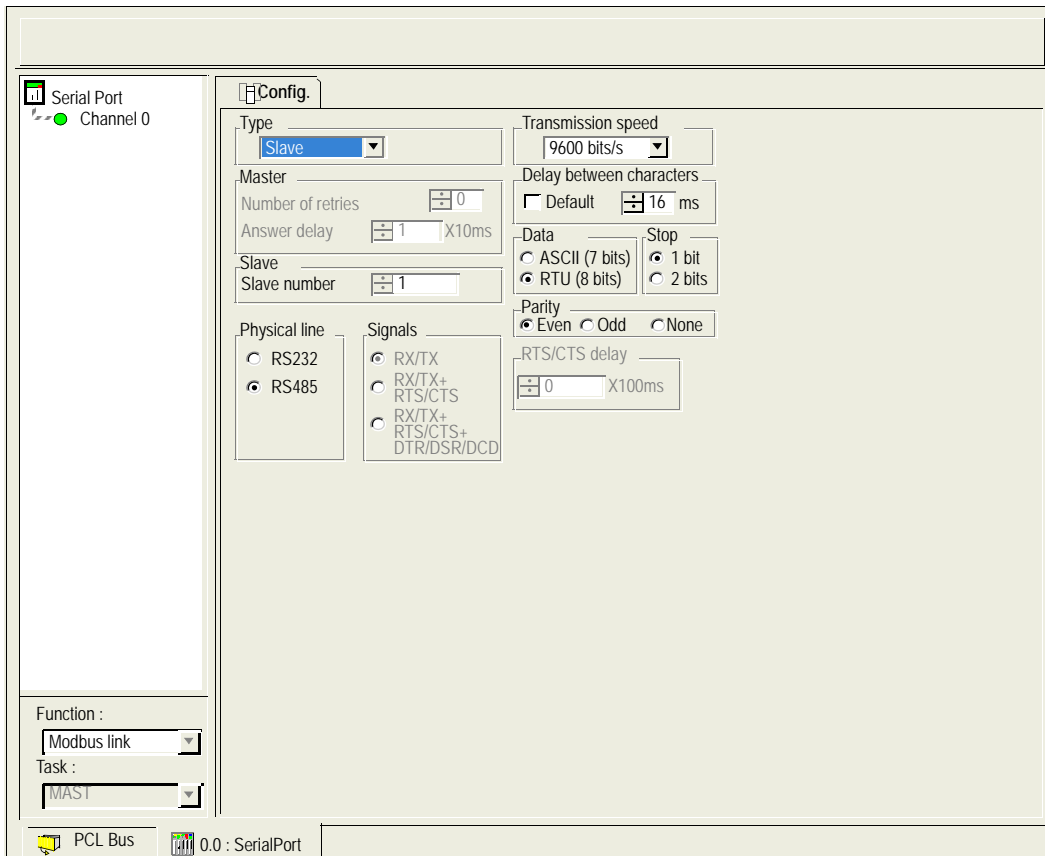
The pages that follow provide an introduction to the configuration screen for Modbus communication.

Access to the Configuration Screen

To access the Modbus communication configuration screen, double-click on the Serial Port sub-directory in the project browser (see *How to Access the Serial Link Parameters for the BMX P34 1000/2000/2010/2020 Processors*, p. 44).

Illustration

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



Description

This zone is used to configure channel parameters. In online mode, this zone is not accessible and will be grayed out. In offline mode, the zone is accessible but some parameters may not be accessible and will therefore be grayed out.

The configuration screen is composed of three types of parameters:

- Application parameters
 - Transmission parameters
 - Signal and physical line parameters
-

Accessible Modbus Functions

At a Glance

Function accessibility for configuration of the serial link of a BMX P34 1000/2000/2010/2020 processor using Modbus 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	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	<ul style="list-style-type: none"> ● ASCII (7 bits) ● RTU (8 bits) 	<ul style="list-style-type: none"> ● ASCII (7 bits) ● RTU (8 bits)
Stop	<ul style="list-style-type: none"> ● 1 bit ● 2 bits 	<ul style="list-style-type: none"> ● 1 bit ● 2 bits
Parity	<ul style="list-style-type: none"> ● Odd ● Even ● None 	<ul style="list-style-type: 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 Communication Parameters

At a Glance All Modbus communication parameters have default values.

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

Configuration parameter	Value
Mode	Slave
Physical Line	RS 485
Slave number	1
Delay between frames	2 ms
Transmission speed	19200 bits/s
Parity	Even
Data Bits	RTU (8 bits)
Stop bits	1 bit

Configuration Screen for Modbus Communication

General

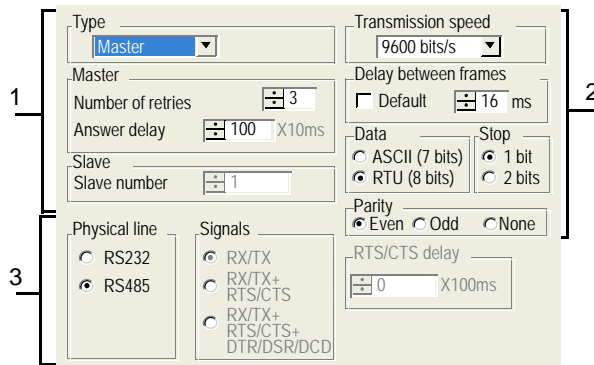
The configuration screens for Modbus Master and Modbus Slave communications are different in that the accessible parameters are not the same.

Accessing the Configuration Screen

To access the Modbus Master and Modbus Slave communication configuration screens, open the Serial Port directory in the project browser (see *How to Access the Serial Link*, p. 44).

Illustration

The figure below shows the configuration screen for Modbus communication:



Description

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

Address	Element	Comment
1	Application Parameters	<p>These parameters are accessible via three zones:</p> <ul style="list-style-type: none"> ● Type ● Master ● Slave <p>For further information about application parameters (see <i>Application-linked Modbus Parameters, p. 55</i>).</p>
2	Transmission Parameters	<p>These parameters are accessible via five zones:</p> <ul style="list-style-type: none"> ● Transmission speed ● Delay between frames ● Data ● Stop bits ● Parity <p>For further information about transmission parameters (see <i>Transmission-linked Modbus Parameters, p. 57</i>).</p>
3	Signal and Physical Line Parameters	<p>These parameters are accessible via three zones:</p> <ul style="list-style-type: none"> ● Physical line ● Signals ● RTS/CTS delay <p>For further information about signal and physical line parameters (see <i>Signal and Physical Line Parameters in Modbus, p. 59</i>).</p>

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

Note: In this example, the "Signals" and "RTS/CTS Delay" zones are grayed out because an RS 485 physical line has been chosen.

Application-linked Modbus Parameters

At a Glance

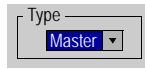
After configuring the communication channel, you need to enter the application parameters.

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:

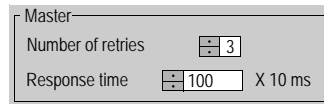


This zone enables you to select the type of Modbus Protocol to be used. The two types available are:

- **Master:** This is to be selected when the station concerned is the master.
 - **Slave:** This is to be selected 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.
 - Response time: the time that elapses between the Master's initial request and a repeat 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.
-

The Slave Zone

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



A screenshot of a configuration window titled "Slave". Inside the window, there is a label "Slave number" followed by a text input field containing the number "7". The input field has small up and down arrow icons on its left side, indicating it is a spin box.

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

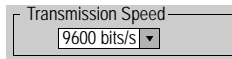
After configuring the communication channel, you need to enter the transmission parameters.

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:

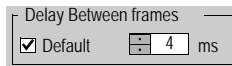


Transmission Speed
9600 bits/s ▾

You can use it to select the transmission speed of the Modbus 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 Delay Between Frames Zone

This configuration zone appears on the screen as shown below:



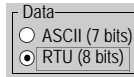
Delay Between frames
 Default 4 ms

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.

The Data Zone

This configuration zone appears on the screen as shown below:

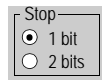


This zone allows you to enter the type of coding used to communicate using Modbus protocol. This field is set according to the other devices connected on the bus. There are two configurable modes:

- 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:

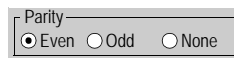


The Stop zone allows you to enter the number of stop bits used for communication. This field is set according to the other devices. The configurable values are:

- 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

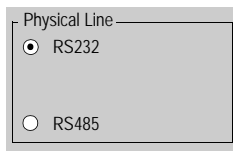
After configuring the communication channel, you need to enter the transmission parameters.

These parameters are accessible via three zones:

- The Physical Line Zone
 - The Signals Zone
 - The RTS/CTS Delay Zone
-

The Physical Line Zone

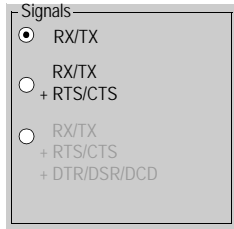
This configuration zone appears on the screen as shown below:



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

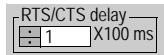
- RX/TX
- RX/TX + RTS/CTS

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

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

The RTS/CTS Delay Zone

This configuration zone appears on the screen as shown below:



In this zone, you can select the delay for waiting CTS signal when RS232 + RX/TX + RTS/CTS is selected. Available in Modbus protocol or Character mode (see *Signal and Physical Line Parameters in Character Mode, p. 85*).

If the RS485 is configured or RTS/CTS is not selected, the entire zone will be grayed out.

5.3 Modbus Communication Programming

At a Glance

Subject of this Section

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

What's in this Section?

This section contains the following topics:

Topic	Page
Services Supported by a Modbus Link Slave Processor	62
Services Supported by a Modbus Link Master Processor	63

Services Supported by a Modbus Link Slave Processor

At a Glance

When used as a slave processor in a Modbus link, a BMX P34 1000/2000/2010/2020 processor supports several services.

Data Exchanges

A slave processor 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

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 communication errors on the bus	16#08 / 16#0C
Read number of 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
Read event counter	16#0B
Read connection event	16#0C
Read identification	16#11
Read Device identification	16#2B / 16#0E

Services Supported by a Modbus Link Master Processor

At a Glance

When used as the master processor in a Modbus link, a BMX P34 1000/2000/2010/2020 processor supports several services via the READ_VAR and WRITE_VAR communication functions.

Data Exchanges

The following requests are addressed to the slave device with which you wish to carry out reading or writing of variables.

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

Note: write utilities can be sent in broadcast mode. In this case no response is returned to the transmitter. Unlike Premium, after the sending of a broadcast request the M340 resets the activity bit and the code 16#01 (Exchange stop on timeout) is returned into the EF 2nd management word.

Note: The objects read by M340 PLC can be of the type %I and %IW. In this case READ_VAR function generate a Modbus request FC 0x2 or 0x4. In a Quantum PLC it allows accessing to Input Status or Input Status Registers.

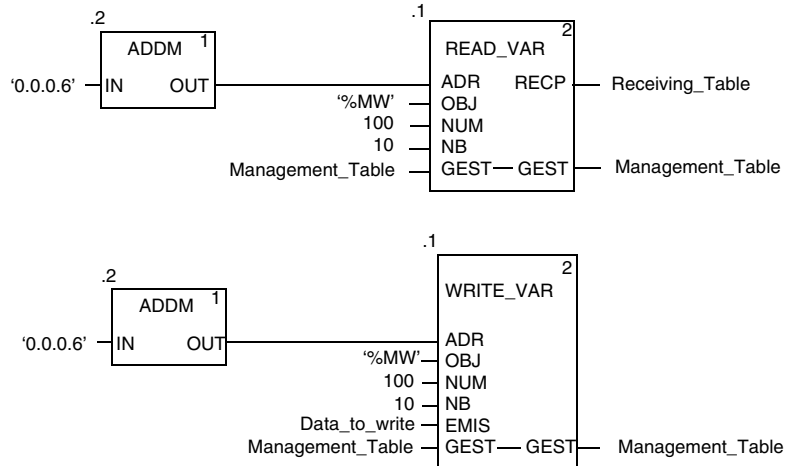
The READ_VAR and WRITE_VAR Communication Functions

Two 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.

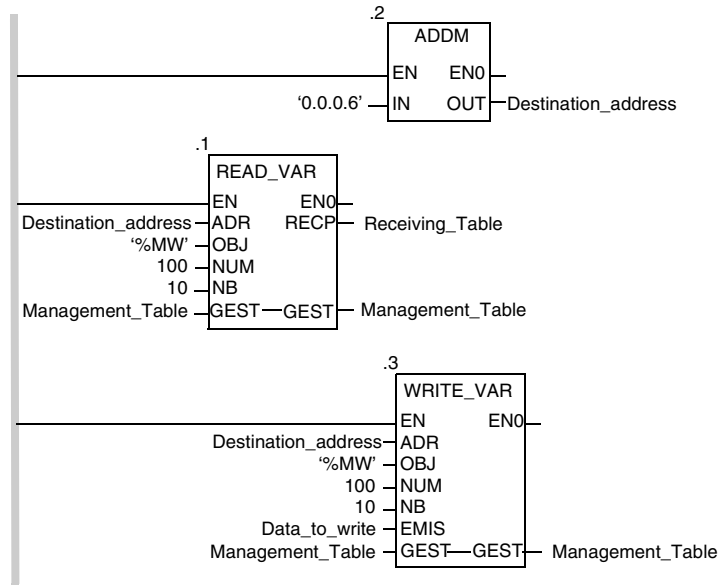
Example of Programming in DFB

The diagram below represents an example of programming of the READ_VAR and WRITE_VAR communication functions in DFB language:



Example of Programming in Ladder

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



Programming Example in ST

The lines of code below represent an example of programming of the `READ_VAR` and `WRITE_VAR` communication functions in 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);
```

Cancelling an Exchange

An exchange executed by the `READ_VAR` and `WRITE_VAR` functions can be cancelled with either of two 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 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: <ul style="list-style-type: none"> ● 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
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: <ul style="list-style-type: none"> ● %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.
GEST	ARRAY [0..4] OF INT	Exchange management table consisting of the following words: <ul style="list-style-type: none"> ● Rank 1 word: a word managed by the system and consisting of two bytes: <ul style="list-style-type: none"> ● 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: <ul style="list-style-type: none"> ● 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
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: <ul style="list-style-type: none"> ● %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.
GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words: <ul style="list-style-type: none"> ● Rank 1 word: a word managed by the system and consisting of two bytes: <ul style="list-style-type: none"> ● 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: <ul style="list-style-type: none"> ● 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.

5.4 Debugging Modbus Communication

Modbus Communication Debug Screen

General

The Modbus 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 communication:

Step	Action
1	Access the configuration screen for Modbus communication. (see <i>Access to the Configuration Screen, p. 49</i>)
2	Select the "Debug" tab on the screen that appears.

Description of the Debug Screen

The debug screen is divided into two zones:

- The Type zone
- The Counters zone

The Type Zone

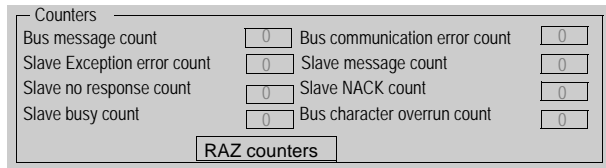
This zone looks like this:



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

The Counters Zone

This zone looks like this:



Here, you can view the various debugging counters.

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

How a Counter Operates

The Modbus communication debugging counters are as follows:

- 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 errors 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 fewer 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 malfunction.

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

Software Implementation of Communication Using Character Mode

6

At a Glance

Subject of this Section

This chapter presents the software implementation of communication using Character Mode.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	General	72
6.2	Character Mode Communication Configuration	76
6.3	Character Mode Communication Programming	86
6.4	Debugging Character Mode communication	92

6.1 General

At a Glance

Subject of this Section

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

What's in this Section?

This section contains the following topics:

Topic	Page
About Character Mode Communication	73
Performance	74

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
-

Performance

At a Glance

The following tables can be used to evaluate typical exchange times in Character Mode.

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

Definition of "Exchange Time"

Exchange time is the time that passes between the creation of an exchange and the end of that exchange. It therefore includes 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 error
 - Time-out expires.
-

Exchange Times for 80 characters

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

Baud rate of communication in bits/s	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

Exchange times are similar on the BMX P34 2000/2010 and BMX P34 2020 processors.

Exchange times on the BMX P34 1000 processor are 10% lower than those on the BMX P34 2000/2010/2020 processors.

**Accuracy of
Measurements**

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

6.2 Character Mode Communication Configuration

At a Glance

Subject of this Section

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

What's in this Section?

This section contains the following topics:

Topic	Page
Character Mode Communication Configuration Screen	77
Accessible Functions in Character Mode	79
Default Values for Character Mode Communication Parameters	80
Transmission Parameters in Character Mode	81
Message End Parameters in Character Mode	83
Signal and Physical Line Parameters in Character Mode	85

Character Mode Communication Configuration Screen

General

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

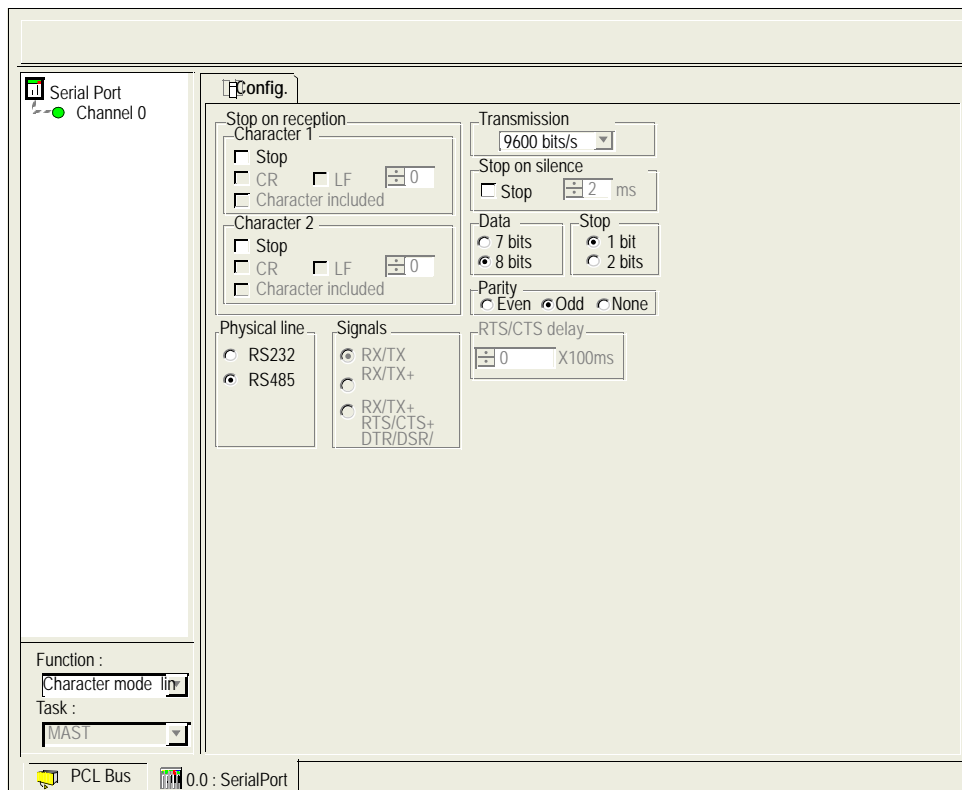
Accessing the Configuration Screen

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

Step	Action
1	Double-click on the Serial Port sub-directory in the project browser (see <i>How to Access the Serial Link Parameters for the BMX P34 1000/2000/2010/2020 Processors</i> , p. 44).
2	Select the CHARACTER MODE LINK function on the screen that appears.

Illustration

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



Description

The configuration screen is used to configure the channel parameters. The screen displays three types of parameters:

- Transmission parameters
 - Message end detection parameters
 - Signal and physical line parameters
-

Accessible Functions in Character Mode

At a Glance

Function accessibility for configuration of the serial link of a BMX P34 1000/2000/2010/2020 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	RS 232 Link
Transmission speed	X	X
Data	<ul style="list-style-type: none"> ● 7 bits ● 8 bits 	<ul style="list-style-type: none"> ● 7 bits ● 8 bits
Stop	<ul style="list-style-type: none"> ● 1 bit ● 2 bits 	<ul style="list-style-type: none"> ● 1 bit ● 2 bits
Parity	<ul style="list-style-type: none"> ● Odd ● Even ● None 	<ul style="list-style-type: 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 485
Transmission speed	9600 bits/s
Parity	Odd
Data Bits	8 bits
Stop bits	1 bit

Transmission Parameters in Character Mode

At a Glance

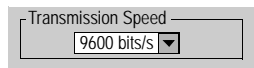
After configuring the communication channel, you need to enter the transmission parameters.

These parameters are accessible via four zones:

- The Transmission Speed Zone
- The Data Zone
- The Stop Zone
- The Parity Zone

The Transmission Speed Zone

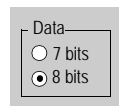
This configuration zone appears on the screen as shown below:



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 and 19200 bits per second.

The Data Zone

This configuration zone appears on the screen as shown below:



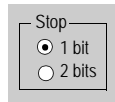
In this zone, you can specify the size of the data being exchanged on the link. The available values are:

- 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

Message End 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:

The screenshot shows a configuration window titled "Stop on reception". It is divided into two sections: "Character 1" and "Character 2".

- Character 1:**
 - Stop
 - CR LF
 - Value: 10
 - Characters included
- Character 2:**
 - Stop
 - CR LF
 - Value: 13
 - Characters included

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

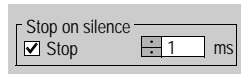
By checking the Stop option, it is possible to configure Stop on Reception to be activated by a specific end-of-message character:

- 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:



The image shows a configuration window titled "Stop on silence". Inside the window, there is a checked checkbox labeled "Stop". To the right of the checkbox is a numeric input field containing the value "1", followed by the unit "ms".

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.

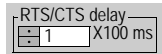
Signal and Physical Line Parameters in Character Mode

At a Glance

After configuring the communication channel, you need to enter the physical line and signal parameters. These parameters are identical to the signal and physical line parameters for Modbus communication (see *Signal and Physical Line Parameters in Modbus*, p. 59).

The RTS/CTS Delay Zone

This configuration zone appears on the screen as shown below:



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.

WARNING

Hazard of loss of characters

Put a value smaller than the time needed to transmit 24 characters added to MAST TASK period time, in order to not lose characters. Refer to INPUT_CHAR (see Unity Pro 3.1, Communication, Block Library: Description) function.

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

6.3 Character Mode Communication Programming

Character Mode Communication Functions

Available Functions

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

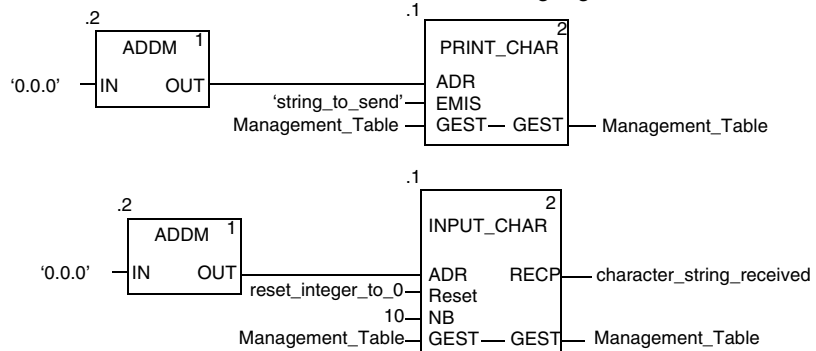
- `PRINT_CHAR`: send a character string of a maximum of 1,024 bytes.
- `INPUT_CHAR`: read a character string of a maximum of 1,024 bytes.

The Modicon M340 PLC's 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.

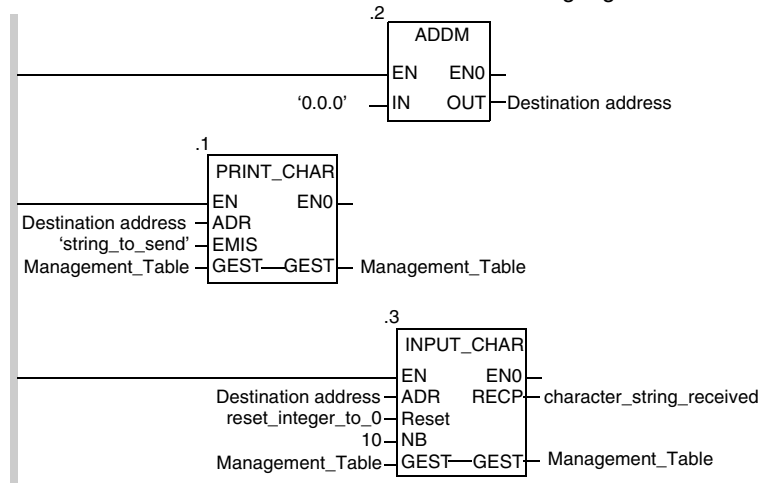
Example of Programming in FBD

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);
```

Other features of INPUT_CHAR function

List of features of INPUT_CHAR function.

- It is possible to launch the INPUT_CHAR function before ending the characters to the PLC.
- If the ending characters are used, if in the buffer there are many ending characters and the buffer hasn't been reset, each INPUT_CHAR function receives the beginning string of the buffer until it reaches the first ending character and then the buffer is removed from the read characters.
- It works in the same way for reading a number of characters.
- If ending characters are configured it could be possible to use the number of characters function.

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: <ul style="list-style-type: none"> ● 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.
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 destination entity 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: <ul style="list-style-type: none"> ● Rank 1 word: a word managed by the system and consisting of two bytes: <ul style="list-style-type: none"> ● 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: <ul style="list-style-type: none"> ● 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. <ul style="list-style-type: none"> ● 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.

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 destination entity given by the OUT parameter of the ADDM function.
Reset	INT	This parameter may take two values: <ul style="list-style-type: none"> ● Value 1: reset module reception memory to 0 ● Value 0: do not reset module reception memory to 0
NB	INT	Length of character string to be received.
RECP	STRING	Character string received. This string is saved in a character string.
GEST	ARRAY [0..3] OF INT	Exchange management table consisting of the following words: <ul style="list-style-type: none"> ● Rank 1 word: a word managed by the system and consisting of two bytes: <ul style="list-style-type: none"> ● 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: <ul style="list-style-type: none"> ● 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.

6.4 Debugging Character Mode communication

At a Glance

Subject of this Section This section describes the Debugging process during set-up of Character Mode communication.

What's in this Section? This section contains the following topics:

Topic	Page
Debug Screen for Character Mode communication	93
Debugging Parameters in Character Mode	94

Debug Screen for Character Mode communication

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:

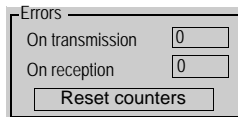
Step	Action
1	Access the configuration screen for Character Mode communication. (see <i>Accessing the Configuration Screen, p. 77</i>)
2	Select the "Debug" tab on the screen that appears.

Description of the Debug Screen

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

The Error Zone

The Error zone looks like this:



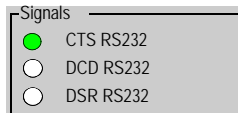
This zone indicates the number of communication errors counted by the processor:

- On transmission: corresponds to the number of errors on transmission (image of $\%MW4$ word).
- On reception: corresponds to the number of errors on reception (image of $\%MW5$ word).

The Reset Counters button resets both counters to zero.

The Signals Zone

The Signals zone looks like this:



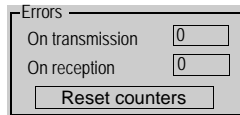
This zone indicates the activity of the signals:

- 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).

Debugging Parameters in Character Mode

At a Glance The debug zone contains the **Errors** window.

Errors Window This window looks like this:



This window indicates the number of communication errors counted by the processor:

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

The **Reset Counters** button resets both counters to zero.

Language Objects of Modbus and Character Mode Communications

7

At a Glance

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's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	Language Objects and IODDTs of Modbus and Character Mode Communications	96
7.2	General Language Objects and IODDTs for All Communication Protocols	104
7.3	Language Objects and IODDTs Associated with Modbus Communication	108
7.4	Language Objects and IODDTs associated with Character Mode Communication	115
7.5	The IODDT Type T_GEN_MOD Applicable to All Modules	122

7.1 Language Objects and IODDTs of Modbus and Character Mode Communications

At a Glance

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's in this Section?

This section contains the following topics:

Topic	Page
Introduction to the Language Objects for Modbus and Character Mode Communications	97
Implicit Exchange Language Objects Associated with the Application-Specific Function	98
Explicit Exchange Language Objects Associated with the Application-Specific Function	99
Management of Exchanges and Reports with Explicit Objects	101

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 all communication protocols.
- 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 3.0: Operating Modes).
 - Using the Data Editor (See Unity Pro 3.0: Operating Modes).
-

Types of Language Objects

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

There are two types of language objects:

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

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

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

Implicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

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

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

Reminders

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

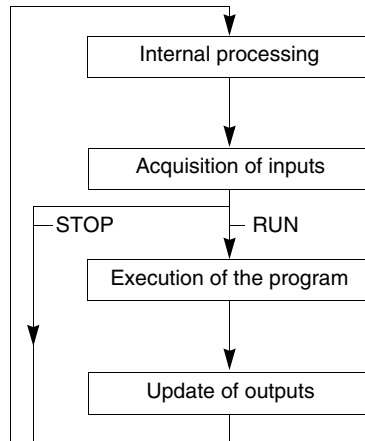
The outputs (%Q and %QW) are updated at the end of the task, only when the PLC is in RUN mode.

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

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

Illustration

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



Explicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

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

- READ_STS (see Unity 3.0 I/O Management Block Library, Description): read status words
- WRITE_CMD (see Unity 3.0 I/O Management Block Library, Description): write command words

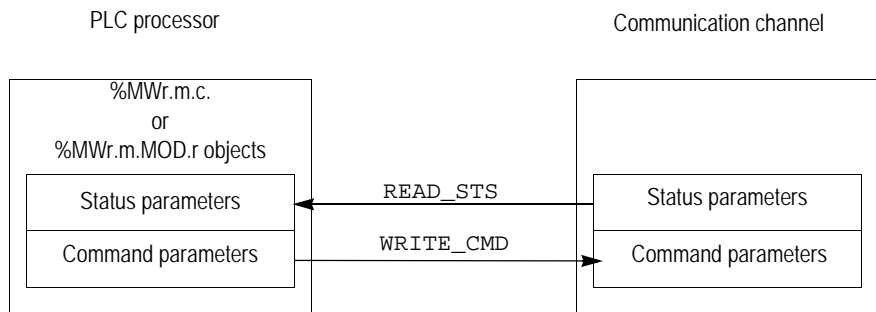
These exchanges apply to a set of %MW objects of the same type (status, commands or parameters) belonging to a channel.

Note: These objects provide information about the processor (e.g.: fault type for a channel, etc.), 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 without fail. 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:



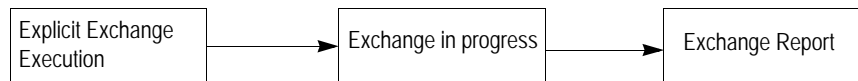
Managing Exchanges

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 Section Language Objects and IODDTs of Modbus and Character Mode Communications).
- The exchange report (see Section Language Objects and IODDTs of Modbus and Character Mode Communications).

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 (%MW_r.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:

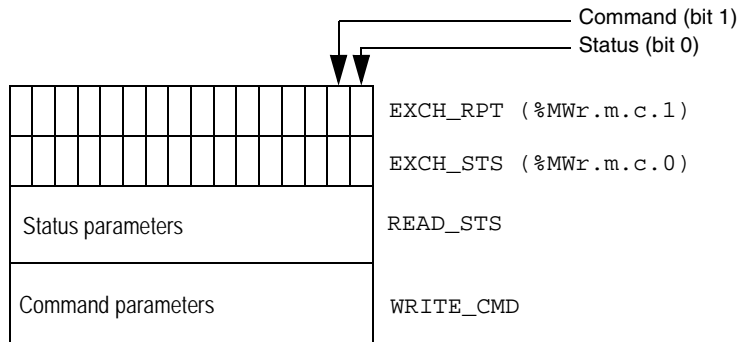
- EXCH_STS (%MWr.m.c.0) : exchange in progress.
- EXCH_RPT (%MWr.m.c.1) : report.

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`) and `EXCH_RPT` (`%MWr.m.MOD.1`) as per `T_GEN_MOD` type IODDTs.

Explicit Exchange Execution Flags: EXCH_STS

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

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

Note: If the module is not present or is disconnected, exchanges using explicit objects (`READ_STS`, for example) are not sent to the processor (`STS_IN_PROGR` (`%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	Error reading channel status words (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Error during a command parameter exchange (1 = failure)	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Error while exchanging adjustment parameters (1 = failure)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Error during reconfiguration of the channel (1 = failure)	%MWr.m.c.1.15

7.2 General Language Objects and IODDTs for All Communication Protocols

At a Glance

Subject of this Section

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

What's in this Section?

This section contains the following topics:

Topic	Page
Details of IODDT Implicit Exchange Objects of Type T_COM_STS_GEN	105
Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN	106

Details of IODDT Implicit Exchange Objects of Type T_COM_STS_GEN

At a Glance

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

Error bit

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

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

Details of IODDT Explicit Exchange Objects of Type T_COM_STS_GEN

At a Glance

This section presents the T_COM_STS_GEN type IODDT explicit exchange objects applicable to all communication protocols except Fipio. 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_P ROGR	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	Read error for channel status words.	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Error during command parameter exchange.	%MWr.m.c.1.1

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

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 faulty.	%MWr.m.c.2.1
BLK	BOOL	R	Terminal block fault (not connected).	%MWr.m.c.2.2
TO_ERR	BOOL	R	Time out error (defective wiring).	%MWr.m.c.2.3
INTERNAL_FLT	BOOL	R	Internal error or channel self-testing.	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Different hardware and software configurations.	%MWr.m.c.2.5
COM_FLT	BOOL	R	Problem communicating with the PLC.	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Application error (adjustment or configuration error).	%MWr.m.c.2.7

Reading is performed by the READ_STS (IODDT_VAR1) instruction .

7.3 Language Objects and IODDTs Associated with Modbus Communication

At a Glance

Subject of this Section

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

What's in this Section?

This section contains the following topics:

Topic	Page
Details concerning Explicit Exchange Language Objects for a Modbus Function	109
Details of the IODDT Implicit Exchange Objects of type T_COM_MB_BMX	110
Details of the IODDT Explicit Exchange Objects of type T_COM_MB_BMX	111
Details of language objects associated with configuration Modbus mode	113

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 without CRC error.
%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 character errors.
%MWr.m.c.24.0	BOOL	RW	Reset of error counters.

Details of the IODDT Implicit Exchange Objects of type T_COM_MB_BMX

At a Glance

The tables below show the implicit exchange objects of the IODDT of the T_COM_MB_BMX type that are applicable to Modbus serial.

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

Word object in Modbus master mode

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

Standard symbol	Type	Access	Meaning	Address
CTS	BOOL	R	Ready to send signal.	%IW.r.m.c.0.2

Word object in Modbus slave mode

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

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

Standard symbol	Type	Access	Meaning	Address
LISTEN_ONLY	BOOL	R	List mode only signal.	%IW.r.m.c.0.8

Details of the IODDT Explicit Exchange Objects of type T_COM_MB_BMX

At a Glance

This part presents the explicit exchange objects of the IODDT of the T_COM_MB_BMX type that are applicable to Modbus serial. 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.	%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	Read error for channel status words.	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Error during command parameter exchange.	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Error while exchanging adjustment parameters.	%MWr.m.c.1.2

Standard Channel 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 faulty.	%MWr.m.c.2.1

Standard symbol	Type	Access	Meaning	Address
BLK	BOOL	R	Terminal block fault (not connected).	%MWr.m.c.2.2
TO_ERR	BOOL	R	Time out error (defective wiring).	%MWr.m.c.2.3
INTERNAL_FLT	BOOL	R	Internal error or channel self-testing.	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Different hardware and software configurations.	%MWr.m.c.2.5
COM_FLT	BOOL	R	Problem communicating with the PLC.	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Application 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
RST_CPT	BOOL	R/W	Resets error counters when it is set to 1.	%MWr.m.c.24.0
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 *Changing Protocol*, p. 123).

Details of language objects associated with configuration Modbus mode

At a Glance

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

List of explicit exchange objects for Master mode

The table below shows the explicit exchange objects.

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

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

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, 247].
%KWr.m.c.4	INT	R	Used only in Modbus Master mode.

7.4 Language Objects and IODDTs associated with Character Mode Communication

At a Glance

Subject of this Section

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

What's in this Section?

This section contains the following topics:

Topic	Page
Details concerning Explicit Exchange Language Objects for Communication in Character Mode	116
Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX	117
Details of IODDT Explicit Exchange Objects of Type T_COM_CHAR_BMX	118
Details of language objects associated with configuration in Character mode	120

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
%MW \mathcal{L} .m.c.4	INT	R	Error in transmitted characters.
%MW \mathcal{L} .m.c.5	INT	R	Error in received characters.

Details of IODDT Implicit Exchange Objects of Type T_COM_CHAR_BMX

At a Glance

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

Error bit

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

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

Signal object on input

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

Standard symbol	Type	Access	Meaning	Address
CTS	BOOL	R	Ready to send signal.	%IWIr.m.c.0.2

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.	%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	Read error for channel status words.	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Error during command parameter exchange.	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Error during adjustment parameter exchange.	%MWr.m.c.1.2

Standard channel 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 faulty.	%MWr.m.c.2.1
BLK	BOOL	R	Terminal block fault (not connected).	%MWr.m.c.2.2
TO_ERR	BOOL	R	Time out error (defective wiring).	%MWr.m.c.2.3
INTERNAL_FLT	BOOL	R	Internal error or channel self-testing.	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Different hardware and software configurations.	%MWr.m.c.2.5
COM_FLT	BOOL	R	Problem communicating with the PLC.	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Application 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
RST_CPT	BOOL	R/W	Resets error counters when it is set to 1.	%MWr.m.c.24.0
TO_MODBUS_MASTE R	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 *Changing Protocol*, p. 123).

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
%KW.r.m.c.0	INT	R	The byte 0 of this word corresponds to the type. Value 3 corresponds to Character Mode.
%KW.r.m.c.1	INT	R	<p>The byte 0 of this word corresponds to the transmission speed. This byte can take several values:</p> <ul style="list-style-type: none"> ● Value -2 (0xFE) corresponds to 300 bits/s ● Value -1 (0xFF) corresponds to 600 bits/s ● Value 0 (0x00) corresponds to 1200 bits/s ● Value 1 (0x01) corresponds to 2400 bits/s ● Value 2 (0x02) corresponds to 4800 bits/s ● Value 3 (0x03) corresponds to 9600 bits/s (default value) ● Value 4 (0x04) corresponds to 19200 bits/s ● Value 5 (0x05) corresponds to 38400 bits/s <p>The byte 1 of this word corresponds to the format:</p> <ul style="list-style-type: none"> ● Bit 8: number of bits (1 = 8 bits (RTU), 0 = 7 bits (ASCII)) ● bit 9 = 1: parity management (1 = with, 0 = without) ● Bit 10: parity Type (1 = odd, 0 = even) ● Bit 11: number of stop bits (1 = 1 bit, 0 = 2 bits) ● Bit 13: physical line (1 = RS232, 0 = RS485) ● Bit 15: signals. If RS232 is selected this bit can take 2 different value, 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
%KW.r.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.
%KW.r.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.

Address	Type	Access	Meaning
%KWr.m.c.6	INT	R	<p>Bit 0 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the stop checkbox in the Stop on reception area for character 1 when checked ● value 0 corresponds to the stop checkbox in the Stop on reception area for character 1 when unchecked <p>Bit 1 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the Character Included checkbox in the Stop on reception area for character 1 when checked ● value 0 corresponds to the Character Included checkbox in the Stop on reception area for character 1 when unchecked <p>Byte 1 of this word corresponds to the entered value of stop on reception of character 1 from 0 to 255.</p>
%KWr.m.c.7	INT	R	<p>Bit 0 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the stop checkbox in the Stop on reception area for character 2 when checked ● value 0 corresponds to the stop checkbox in the Stop on reception area for character 2 when unchecked <p>Bit 1 of Byte 0 can have 2 values:</p> <ul style="list-style-type: none"> ● value 1 corresponds to the Character Included checkbox in the Stop on reception area for character 2 when checked ● value 0 corresponds to the Character Included checkbox in the Stop on reception area for character 2 when unchecked <p>Byte 1 of this word corresponds to the entered value of stop on reception of character 2 from 0 to 255.</p>

7.5 The IODDT Type T_GEN_MOD Applicable to All Modules

Details of the Language Objects of the IODDT of Type T_GEN_MOD

Introduction All the modules of Modicon M340 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 error bit	%I.r.m.MOD.ERR
EXCH_STS	INT	R	Module exchange control word	%MWr.m.MOD.0
STS_IN_PROGR	BOOL	R	Reading of status words of the module in progress	%MWr.m.MOD.0.0
EXCH_RPT	INT	R	Exchange report word	%MWr.m.MOD.1
STS_ERR	BOOL	R	Event when reading module status words	%MWr.m.MOD.1.0
MOD_FLT	INT	R	Internal error word of the module	%MWr.m.MOD.2
MOD_FAIL	BOOL	R	Internal error, 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 error	%MWr.m.MOD.2.5
NO_MOD	BOOL	R	Module missing or inoperative	%MWr.m.MOD.2.6
EXT_MOD_FLT	BOOL	R	Internal error word of the module (Fipio extension only)	%MWr.m.MOD.2.7
MOD_FAIL_EXT	BOOL	R	Internal detected fault, 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
CONF_FLT_EXT	BOOL	R	Hardware or software configuration error (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

Dynamic Protocol Switching



Changing Protocol

General

This part describes how to change the protocol used by a 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

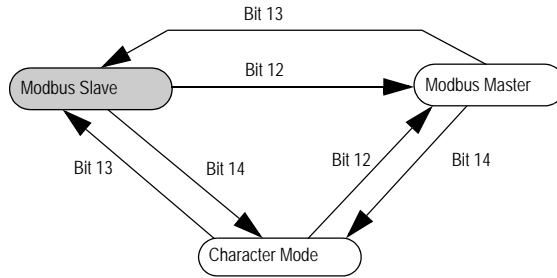
You must create first an IODDT variable linked to the processor's serial channel, then set to 1 the bit of word `IODDT_VAR1.CONTROL (%MWr.m.c.24)` that corresponds to the change of protocol desired:

- `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 (%MW_r.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.

Warm and Cold 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.

Quick start : example of Serial link implementation



At a glance

Overview

This section presents an example of Serial link implementation.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
9	Description of the application	129
10	Installing the application using Unity Pro	131
11	Starting the Application	157

Description of the application

9

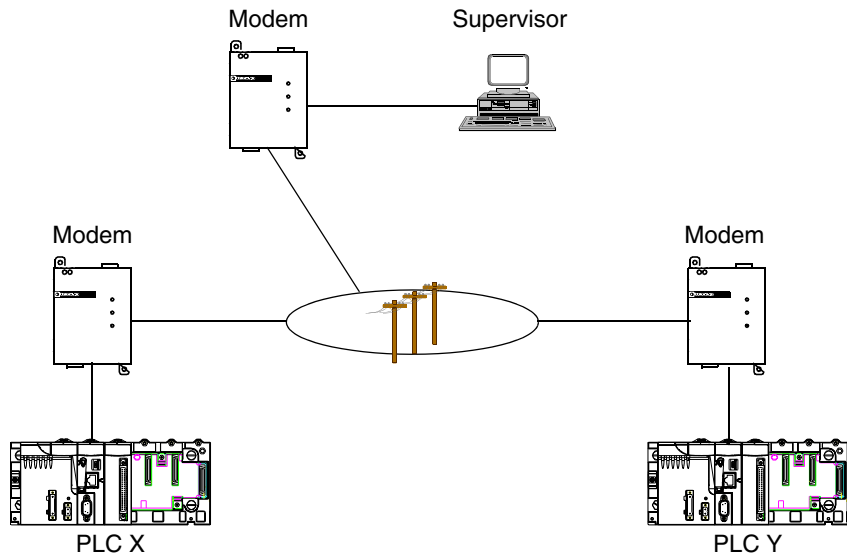
Overview of the application

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 devices communicate with each other using modems.

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 there's an alarm on 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.

Installing the application using Unity Pro

10

At a glance

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's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
10.1	Presentation of the solution used	132
10.2	Developping the application	133

10.1 Presentation of the solution used

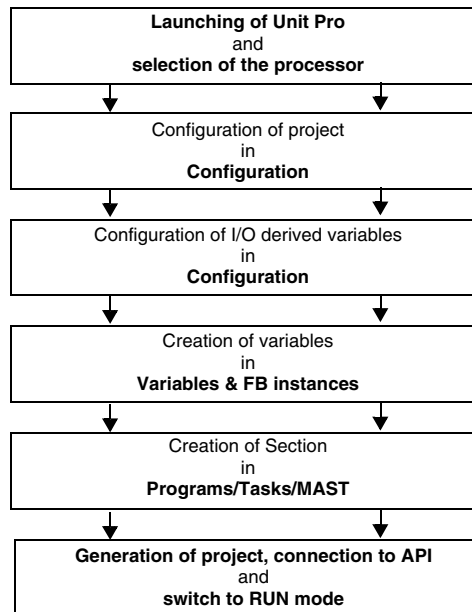
The different steps in the process using Unity Pro

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:



10.2 **Developping the application**

At a glance

Subject of this section

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

What's in this Section?

This section contains the following topics:

Topic	Page
Creating the project	134
Declaration of variables	138
Using a modem	143
Procedure for programming	145
Programming structure	147
Programming	150

Creating the project

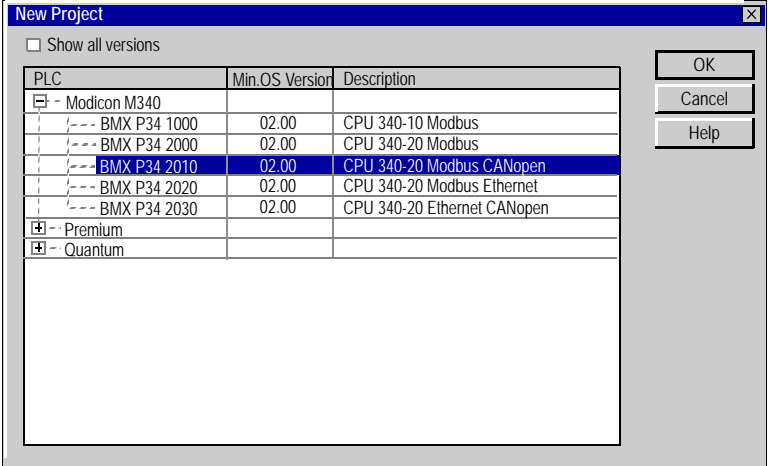
At a glance

For the development of the example, a project associated with the PLC X must be created.

Note: For more information, see Unity Pro online help (click on ?, then Unity, then Unity Pro, then Operate modes, and Project configuration).

Procedure for creating a project

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

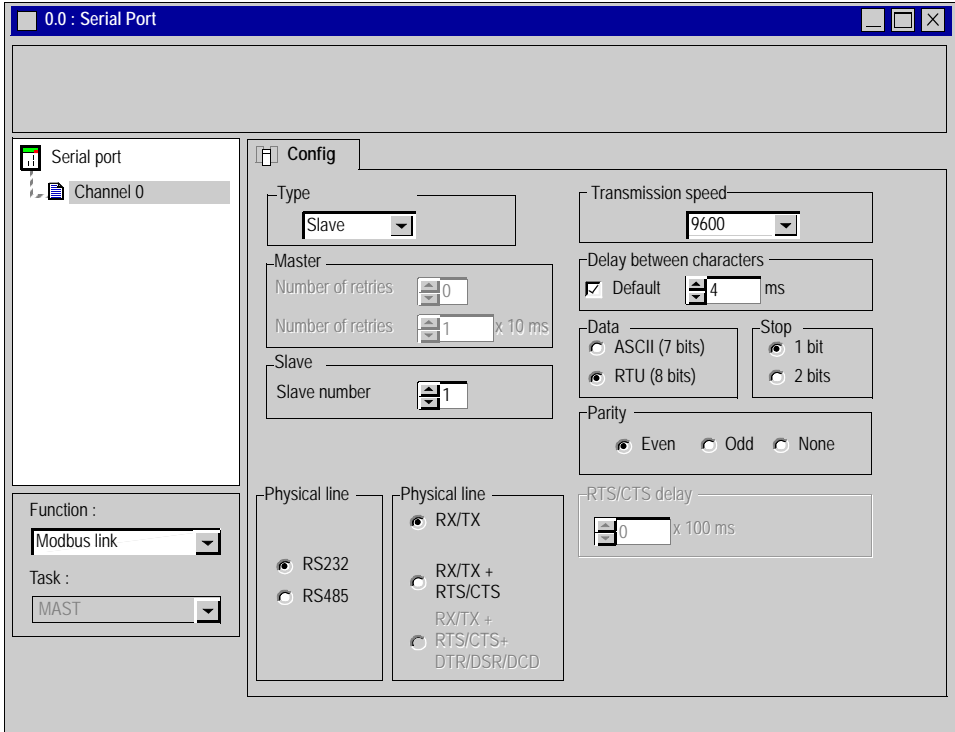
Etape	Action
1	Launch the Unity Pro software,
2	Click on File then New to select a BMX P34 2010 processor: 
3	Confirm with OK.

Module selection The table below shows the procedure for selecting discrete module.

Step	Action																														
1	In the Project browser double-click on Configuration then on 0:Bus X and on 0:BMX XBP ... (Where 0 is the rack number),																														
2	In the Bus X window, select a slot (for example slot 1) and double-click on it,																														
3	Choose the BMX DDI 1602 counting input module, <div data-bbox="450 375 1218 841" style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Part Number</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>[-] Basic Micro local drop</td> <td></td> </tr> <tr> <td>[+] Analog</td> <td></td> </tr> <tr> <td>[+] Communication</td> <td></td> </tr> <tr> <td>[+] Counting</td> <td></td> </tr> <tr> <td>[-] Discrete</td> <td></td> </tr> <tr> <td> -- BMX DAI 1604</td> <td>Dig 16 In 120 Vac</td> </tr> <tr> <td> -- BMX DDI 1602</td> <td>Dig 16 In 24 Vdc Sink</td> </tr> <tr> <td> -- BMX DDI 1603</td> <td>Dig 16 In 48 Vdc Sink</td> </tr> <tr> <td> -- BMX DDI 3202K</td> <td>Dig 32 In 24 Vdc Sink</td> </tr> <tr> <td> -- BMX DDI 6402K</td> <td>Dig 64 In 24 Vdc Sink</td> </tr> <tr> <td> -- BMX DDM 16022</td> <td>Dig 8 In 24 Vdc 8 Out Trans Positiv</td> </tr> <tr> <td> -- BMX DDM 16025</td> <td>Dig 8 In 24 Vdc 8 Out Relays</td> </tr> <tr> <td> -- BMX DDM3202K</td> <td>Dig 16 In 24 Vdc 16 Out Trans Positiv</td> </tr> <tr> <td> -- BMX DDO 1602</td> <td>Dig 16 Out Trans Source</td> </tr> </tbody> </table> </div>	Part Number	Description	[-] Basic Micro local drop		[+] Analog		[+] Communication		[+] Counting		[-] Discrete		-- BMX DAI 1604	Dig 16 In 120 Vac	-- BMX DDI 1602	Dig 16 In 24 Vdc Sink	-- BMX DDI 1603	Dig 16 In 48 Vdc Sink	-- BMX DDI 3202K	Dig 32 In 24 Vdc Sink	-- BMX DDI 6402K	Dig 64 In 24 Vdc Sink	-- BMX DDM 16022	Dig 8 In 24 Vdc 8 Out Trans Positiv	-- BMX DDM 16025	Dig 8 In 24 Vdc 8 Out Relays	-- BMX DDM3202K	Dig 16 In 24 Vdc 16 Out Trans Positiv	-- BMX DDO 1602	Dig 16 Out Trans Source
Part Number	Description																														
[-] Basic Micro local drop																															
[+] Analog																															
[+] Communication																															
[+] Counting																															
[-] Discrete																															
-- BMX DAI 1604	Dig 16 In 120 Vac																														
-- BMX DDI 1602	Dig 16 In 24 Vdc Sink																														
-- BMX DDI 1603	Dig 16 In 48 Vdc Sink																														
-- BMX DDI 3202K	Dig 32 In 24 Vdc Sink																														
-- BMX DDI 6402K	Dig 64 In 24 Vdc Sink																														
-- BMX DDM 16022	Dig 8 In 24 Vdc 8 Out Trans Positiv																														
-- BMX DDM 16025	Dig 8 In 24 Vdc 8 Out Relays																														
-- BMX DDM3202K	Dig 16 In 24 Vdc 16 Out Trans Positiv																														
-- BMX DDO 1602	Dig 16 Out Trans Source																														
4	Confirm with OK.																														

Serial port configuration

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

Step	Action
1	<p>In the Project browser double-click on Configuration then on 0: BMS XBP 0800 then on 0: BMX P34 2010. Then double click on Serial Port to access to the 0:0 Serial Port window.</p> 
2	Select the Slave type.
3	Select 9600 for transmission speed.
4	Select RS232 for physical line.
5	Select RTU (8bits) for data type.
6	Close the window and confirm with OK.

Step	Action
7	Do the same for the second processor: <ul style="list-style-type: none">● Type: Slave,● Slave number: 2,● Transmission type: 9600,● Data type: RTU (8 bits),● Stop bit: 1,● Parity: even.

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

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`, then `Operate modes`, and `Data editor`).

Procedure for declaring variables

The table below shows the procedure for declaring application variables:

Step	Action
1	In <code>Project browser / Variables & FB instances</code> , double-click on <code>Elementary variables</code>
2	In the <code>Data editor</code> window, select the box in the <code>Name</code> column and enter a name for your first variable.
3	Now select a <code>Type</code> 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:

Variable	Type	Definition
Adr_Char	STRING	Master PLC serial port address.
Adr_modbus	STRING	Modbus Slave PLC serial port address.
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.
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.
run	EBOOL	Running mode.
Serial_Port	T_COM_MB_BMX	Serial port I/O object
Test_inc	INT	Incrementation value

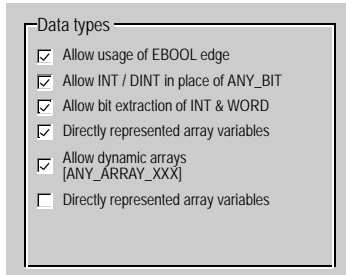
The following screen shows the application variables created using the data editor:

The screenshot shows the 'Data Editor' window with the 'Variables' tab selected. The window contains a filter section and a table of variables. The filter section includes a search icon, a 'Name' field with an asterisk, and checkboxes for 'EDT', 'DDT', and 'IODDT'. The table lists various variables with their names, types, addresses, values, and comments.

Name	Type	Address	Value	Comment
Adr_Char	STRING		0.0.0	
Adr_modbus	STRING		0.0.0.2	
AnsString1	STRING			
AnsString2	STRING			
AnsString3	STRING			
Error	INT			
Function_Step	INT			
MngtInput	ARRAY[0..3] of INT			
MngtPrint	ARRAY[0..3] of INT			
MngtWrite	ARRAY[0..3] of INT			
nb_bit_to_receive_connect	INT		9	
nb_bit_to_receive_ok	INT		4	
ReqString	STRING			
Run	BOOL	%I.0.1.0		
Serial_Port	T_COM_MB_BMX	%CH.0.0.0		
Test_inc	INT			

Declaring an Array type

Before declaring an Array type, click on **Tools/Project Settings/Language extension** 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 Project browser, click on Variables & FB instances.
2	Click in the Name column and enter a name for the variable.
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 (INT for example), then click into the Array checkbox.
5	Modify the intervalle, then confirm with OK.

Declaration of I/O object

The table below shows the procedure for declaring the I/O Derived Variables.

Step	Action
1	In the 0:0 Serial Port window, click on Serial Port and then on the I/O objects tab.
2	Click on the I/O object prefix address %CH then on the Update grid button, the channel address appears in the I/O object grid.
3	Click on the line %CH0.0.0 and then, in the I/O object creation windows, enter a channel name in the prefix for name zone ("Serial_Port" for example).
4	Now click on different Implicit I/O object prefix addresses then on update grid button to see the names and addresses of the implicit I/O objects.

Address	Name
1 %CH0.0.0	Serial_Port
2 %MW0.0.0	Serial_Port.EXCH
3 %MW0.0.1	Serial_Port.EXCH
4 %MW0.0.2	Serial_Port.CH.F
5 %MW0.0.3	Serial_Port.PROT
6 %MW0.0.4	
7 %MW0.0.5	
8 %MW0.0.6	
9 %MW0.0.7	
10 %MW0.0.8	
11 %MW0.0.9	
12 %MW0.0.10	
13 %MW0.0.11	
14 %MW0.0.12	
15 %MW0.0.13	
16 %MW0.0.14	
17 %MW0.0.15	
18 %MW0.0.16	
19 %MW0.0.17	
20 %MW0.0.18	
21 %MW0.0.19	
22 %MW0.0.20	
23 %MW0.0.21	
24 %MW0.0.22	
25 %MW0.0.23	
26 %MW0.0.24	Serial_Port.CONT

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: AT2=43).

Example of initializing command:

```
ATQ0&Q0E0&K0V1
```

with:

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

Dialing the modem

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

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

Example:

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

**Disconnecting
the modem**

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

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

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

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

Procedure for programming

Procedure to follow

The array below shows the procedure for programming the application.

Step	Action	Details
1	Preparing the communication port.	<ul style="list-style-type: none"> ● Change the Slave Modbus mode to Character mode by sending a WRITE_CMD (See <i>Writing the command words</i>, p. 146) to the serial port. ● For a modem transmission , send the HAYES command by using the PRINT_CHAR block to configure the modem (See <i>Using a modem</i>, p. 143). ● 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 <i>Using a modem</i>, p. 143).
2	Master Modbus mode	<ul style="list-style-type: none"> ● Switch to Modbus Master mode using the WRITE_CMD function. ● Send data to write on the Slave PLC.
3	Reseting the communication port.	<ul style="list-style-type: none"> ● Switch to Character mode using the WRITE_CMD command (See <i>Writing the command words</i>, p. 146). ● For a modem transmission, send the escape character, then send the disconnect command to send a disconnection message to the modem (See <i>Using a modem</i>, p. 143) by using the PRINT_CHAR block. ● Return to the starting mode of the serial port (Slave Modbus) using the WRITE_CMD command (See <i>Writing the command words</i>, p. 146).

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 Wait for change to 1 of run bit to go to step 5.	Modem
5	Switch to Character mode. Go to step 10.	
10	Read status of serial port. <ul style="list-style-type: none"> ● If there is an error on the serial port then <ul style="list-style-type: none"> ● Error is a 10 ● Go to step 130 ● If there is no error on the serial port <ul style="list-style-type: none"> ● and Character mode is active, then go to step 15 ● and no Character mode active, then test the status of change to Master Modbus mode on 1000 cycles, then error is at 10, and go to step 130. 	
15	Sending a dial command to the modem via the PRINT_CHAR. Go to step 20.	
20	If the result of PRINT_CHAR is conclusive then go to step 25 otherwise go to step 130 with Error at 20.	
25	Waiting for the response of the modem via the INPUT_CHAR.	
30	If the result of INPUT_CHAR is conclusive then go to step 35 otherwise go to step 130 with Error at 30.	
35	If the modem correctly responds then go to step 40 otherwise go to step 130 with Error at 35.	

Step number	Step description	Element
40	Switch to Master Modbus mode. Go to step 45.	Master Modbus Mode
45	Read status of serial port. <ul style="list-style-type: none"> ● If there is an error on the serial port then <ul style="list-style-type: none"> ● Error is a 45 ● Go to step 130 ● If there is no error on the serial port <ul style="list-style-type: none"> ● and Master Modbus mode is active, then go to step 50 ● and no Master Modbus mode active, then test the status of change to Master Modbus mode on 1000 cycles, then error is at 45, and 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 65 otherwise go to step 130 with Error at 55.	
60	Switch to Character mode. Go to step 65	Character mode
65	Read status of serial port. <ul style="list-style-type: none"> ● If there is an error on the serial port then <ul style="list-style-type: none"> ● Error is a 65 ● Go to step 130 ● If there is no error on the serial port <ul style="list-style-type: none"> ● and Character mode is active, then go to step 70 ● and no Character mode active, then test the status of change to Character mode on 1000 cycles, then error is at 65, and go to step 130. 	

Step number	Step description	Element
70	Sending an escape character 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 go to step 130 with Error at 75.	
80	Read response from the modem using an INPUT_CHAR	
85	If the result of INPUT_CHAR is conclusive then go to step 90 otherwise go to step 130 with Error at 85.	
90	If the modem correctly responds then go to step 95 otherwise go to step 130 with Error at 90.	
95	Sending 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 100 otherwise go to step 130 with Error at 100.	
105	Read response from the modem using an INPUT_CHAR	
110	If the result of INPUT_CHAR is conclusive then go to step 115 otherwise go to step 130 with Error at 110.	
115	If the modem correctly responds then go to step 120 otherwise go to step 130 with Error at 115.	
120	Switch to Slave Modbus mode. Go to step 130	
125	Read status of serial port. <ul style="list-style-type: none"> ● If there is an error on the serial port then <ul style="list-style-type: none"> ● Error is at 125 ● Go to step 130 ● If there is no error on the serial port <ul style="list-style-type: none"> ● and Slave Modbus mode is active, then go to step 130 ● and no Character mode active, then test the status of change to Character mode on 1000 cycles, then error is at 125, and go to step 130. 	
130	Return to step 0.	

Programming

Programming in ST language.

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

```

CASE Function_Step OF
0: (* Initialization *)
  IF (run) 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 *)
    (* 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;

```

```
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 *)
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 = '$NCONNEN') 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('0.0.0.2'),'%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 := 65; (* 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 activecommand *)
        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 *)
    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 *)
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;
END_IF;

130: (* End *)
Run := 0; (* allow new demand *)
Function_Step := 0; (* goto waiting state *)
END_CASE;
```

Starting the Application

11

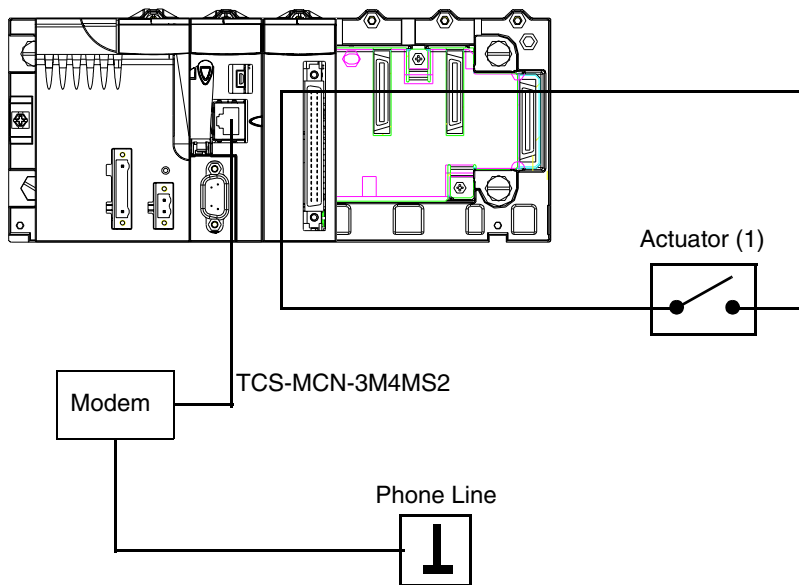
Execution of Application in Standard Mode

At a glance

Standard mode working requires the use of a PLC, a discrete input module and 2 SR1MOD01 modems.

First Slave PLC Wiring

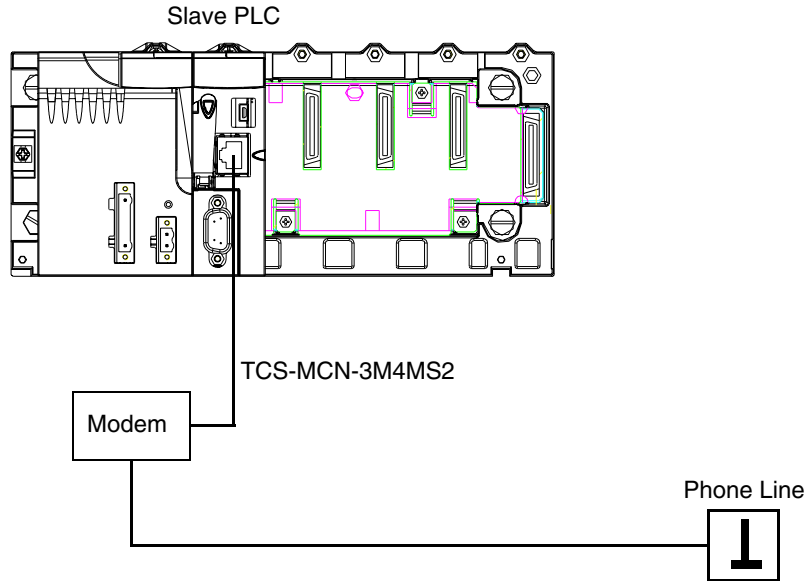
The first Slave PLC is connected as follow:
PLC



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

Second Slave PLC Wiring

The second Slave PLC is connected as follow:



Application transfer

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

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

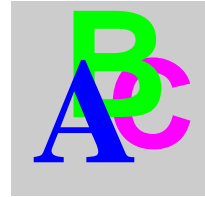
Step	Action
1	In the PLC menu, click on <code>Standard Mode</code> ,
2	In the <code>Build</code> menu, click on <code>Rebuild All Project</code> . Your project is generated and is ready to be transferred to the PLC. When you generate the project, you will see a results window. If there is an error in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

Application execution

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

Step	Action
1	In the PLC, click on Execute . The Execute window opens. Click on OK . The application is now being executed (in RUN mode) 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.

Index



B

BMXP341000, 17
BMXP342000, 17
BMXP342010, 17
BMXP342020, 17

C

changing protocols, 123
channel data structure for all modules
 T_GEN_MOD, 122
channel data structure for character mode
communication
 T_COM_CHAR_BMX, 117, 118
channel data structure for communication
protocols
 T_COM_STS_GEN, 105, 106
channel data structure for modbus
communication
 T_COM_MB_BMX, 110, 111
character mode, 71
configuring character mode, 76
configuring Modbus, 48
connection devices, 21

D

debugging character mode, 92
debugging Modbus, 69

I

INPUT_CHAR, 86

M

Modbus bus, 39

P

parameter settings, 95
PRINT_CHAR, 86
programming character mode, 86
programming Modbus bus, 61

Q

quick start, 127

T

T_COM_CHAR_BMX, 117, 118
T_COM_MB_BMX, 110, 111
T_COM_STS_GEN, 105, 106
T_GEN_MOD, 122

W

wiring accessories, 31

