Modbus Serial Line

Planning and Installation Guide

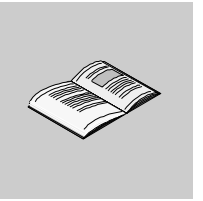
02/2009

33003925.01

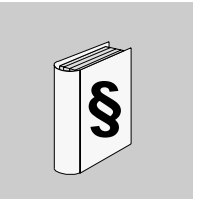
www.schneider-electric.com

Schneider Electric

Table of Contents



Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, 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 or serious injury.

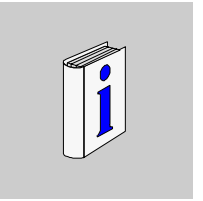
⚠ CAUTION
CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

CAUTION
CAUTION, used without the safety alert symbol, indicates a potentially hazardous situation which, if not avoided, **can result in** 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.

About the Book



At a Glance

Document Scope

This manual provides detailed information about Modbus Serial Line in order to enable you to easily install your individual applications.

The following descriptions are included in this book:

- Modbus physical layer
- Modbus data link layer
- Modbus protocol
- resolving error conditions (troubleshooting)

Validity Note

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.

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
Modbus Application Protocol Specification V1.1b (2006.12.28)	available at www.modbus.org
Modbus over Serial Line Specification and Implementation Guide V1.02 (2006.12.20)	available at www.modbus.org
IEC 62453-315 (2006)	IEC standard

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

Product Related Information

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

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

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

Failure to observe this product's safety-related warning can result in injury or equipment damage.

User Comments

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

Modbus Serial Line Overview

1

Overview

This chapter contains an overview of the Modbus Serial Line protocol and its presentation in this book.

What's in this Chapter?

This chapter contains the following topics:

Introduction

Definition Modbus Protocol

The Modbus protocol was developed by Modicon in 1978 and has been used by the industry as serial de facto standard since 1979. It is an application layer messaging protocol for client/server communication between devices that are connected on different types of buses or networks.

Definition Modbus SL

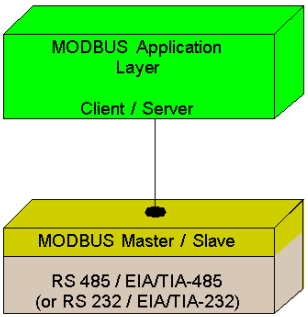
In combination with RS 485 / RS 232 lines the Modbus protocol provides asynchronous serial data transmission and is called Modbus SL, which is the abbreviation of Modbus over Serial Line. The most common interface is RS 485 2-wire, but RS 485 4-wire can also be implemented as an add-on option.

The Modbus SL protocol is operating according to the master/slave principle and is executed on level 2 of the OSI model as shown below.

Modbus Serial Line represented in the ISO/OSI model

Layer	ISO/OSI Model	
7	Application	Modbus Application Protocol
6	Presentation	Empty
5	Session	Empty
4	Transport	Empty
3	Network	Empty
2	Data Link	Modbus Serial Line Protocol
1	Physical	RS 485 / EIA/TIA-485 (or RS 232 EIA/TIA-232)

Modbus Serial Line represented in the ISO/OSI model (graphical overview)



Presentation in this Book

Representation of the ISO/OSI Layers in this Book

The structure of this book is based on the ISO/OSI model.

It therefore starts with layer 1 of the ISO/OSI model, i.e. with the description of any topics related to the physical layer, like the allowed physical interfaces with the respective connectors.

The following section is about the data link layer, i.e. the interaction between the individual devices.

Then follows the description of the Modbus protocol itself, i.e. the individual Modbus framing.

Last but not least this manual provides a troubleshooting section to assist you with problems arising during Modbus SL application setup.

Modbus Physical Layer

2

Overview

This chapter contains any information required to set up the physical layer of a Modbus Serial Line application.

What's in this Chapter?

This chapter contains the following sections:

2.1 Bus Topology

Topology

Topology Types

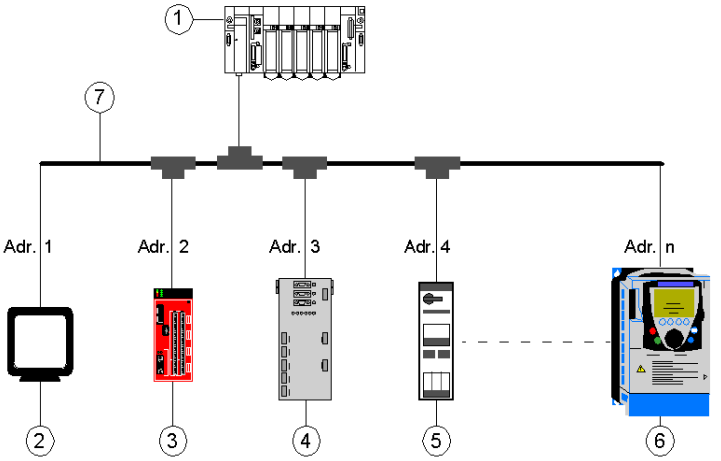
An RS 485 Modbus configuration without repeater mainly consists of 1 trunk cable, which is also named bus. It is possible to connect the individual devices to this bus in 2 different ways:

- Connect the individual devices directly to the trunk cable. (This topology is often referred to as daisy chaining.)
- Use short derivation cables to connect the individual devices to the trunk cable.

For both topologies consider the maximum cable length allowed which is indicated in the next section.

Example of Application with Derivation Cables

The figure below shows an application with derivation cables:



1 Master: Automation Platform Premium PLC
2 Magelis Graphic Terminal (e.g. XBTG)
3 XPSMF40 Safety PLC
4 XPSMF30 Safety PLC
5 TesysU
6 Altivar 71
7 Modbus SL bus

2.2 Network Topologies

Overview

This section includes a list of characteristics of RS 485 applications as well as examples of network topologies for 2-wire and 4-wire applications.

What's in this Section?

This section contains the following topics:

RS 485 Characteristics

Overview

All RS 485 applications provide the characteristics listed in the table below.

RS 485 Characteristics

Connection Type	• point-to-point connections • point-to-multipoint connections
Type of Trunk Cable	shielded cable with 1 twisted pair and at least a third conductor
Maximum Length of Bus	1,000 m (3,280 ft) at 19,200 bit/s with the Telemecanique TSX CSA• cable
Maximum Number of Devices (without repeater)	32 (1 UL) devices, i.e. 31 slaves
Maximum Length of Tap Links	• 20 m (65 ft) for one tap link • a total of 40 m (131 ft) for all tap links available on the bus

RS 485 2-Wire Systems

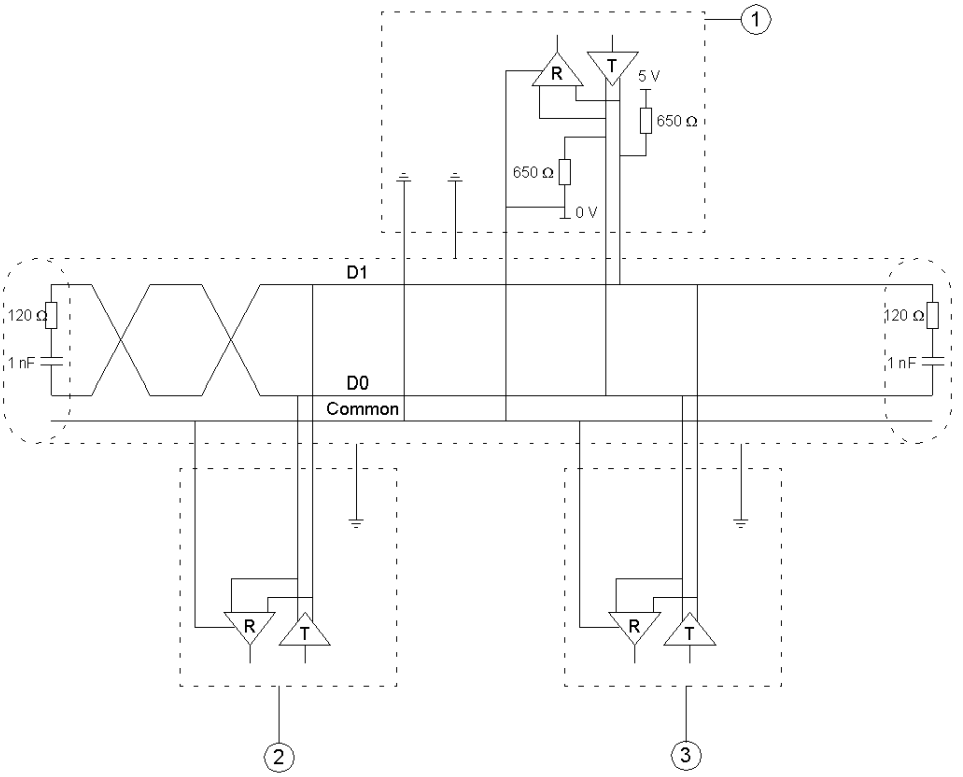
Overview

If there is a standard 2-wire RS 485 (EIA 485) serial line available for your application, make sure that a third conductor, the so-called common, is available to interconnect all devices on the bus.

Network Topology

Create a network topology as shown in the following figure.

Bus type topology for RS 485 2-wire applications:



Elements of the application

No.	Element
1	Master
2	Slave 1
3	Slave n

RS 485 4-Wire Systems

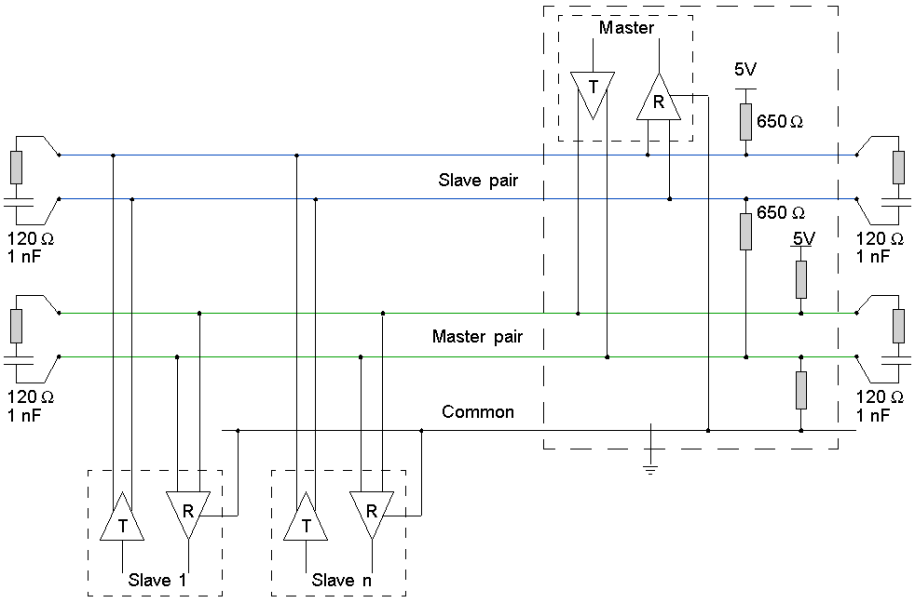
Overview

If there is a 4-wire RS 485 serial line available for your application, make sure that a fifth conductor, the so-called Common, is also available to interconnect all devices on the bus.

Network Topology

Create a network topology as shown in the following figure.

Bus type topology for RS 485 4-wire applications:



Signal Crossings

The 4-wire RS 485 communication can be integrated into an existing installation without modification. But the data on the master wire pair must only be received by the slaves and the data on the slave wire pair must only be received by the master.

In order to ensure that the signals are crossed, use either crossover cables or a tap which includes the crossing function. Since crossover cables may cause problems in 2-wire systems we recommend to use taps with crossing function.

2.3 RS 232 Applications

RS 232 Systems

Overview

For short point-to-point data transmission you can also use an RS 232 (EIA 232) interface between DTE and DCE.

RS 232 Characteristics

RS 232 applications provide the following characteristics:

Connection Type	point-to-point connections
Maximum Length of Bus	20 m (65.61 ft)
Maximum Length of Tap Links	• 20 m (65.61 ft) for one tap link • a total of 40 m (131.23 ft) for all tap links available on the bus
Interfaces	• RJ-45 connectors • SUB-D9 connectors

Wiring Requirements

For RS 232 wiring, proceed as follows:

Stage	Description
1	Determine which device is to be considered as DTE (data terminal equipment) and which as DCE (data communication equipment) device.
2	Connect a common signal to the DTE as well as to the DCE device.
3	Connect each TXD signal with the RXD signal of the other device.
4	You can furthermore connect the RTS signal with the CTS signal of the other device.
5	You can furthermore connect the DTR signal with the DSR signal of the other device.

2.4 Grounding and Polarization

Overview

For RS 485 communications a voltage range between +12 and -7 V is allowed on the transmission line. Any higher voltage (e.g. caused by electrostatic energy being discharged) may damage the network as well as the devices.

To guarantee that this voltage is not exceeded, proper grounding is required.

Furthermore it must be guaranteed that voltage is kept within this range even if there is no data activity on the line. For devices that cannot generate such polarization by themselves, it is required to implement bias resistors into the application.

What's in this Section?

This section contains the following topics:

Grounding

Overview

For RS 485 applications it is required to keep the voltage between drivers and receivers within the range of +12 to -7 V. Any higher voltage may damage the network and / or the devices. You should therefore pay specific attention to proper grounding.

Common Ground Wires

In order to keep the voltage between drivers and receivers within the allowed range, an additional third (in 2-wire systems) or fifth wire (in 4-wire systems) is required. This wire will be used as common circuit and must therefore be directly connected to protective ground, preferably at one point only for the entire bus.

As grounding point for the entire bus you should choose the master device or its tap.

However, the shield of the connector must be connected to protective ground at least at 1 point.

Biasing the Network

Overview

When there is no data activity on the Modbus bus, i.e. all nodes are in receive mode and there is no active driver available, the state of the line is unknown. In these cases the line is subjected to external noise or interference. In order to prevent the receivers from adopting improper states, the line needs to be biased, i.e. the constant state of the line must be maintained by an external pair of resistors connected to the RS 485 balanced pair.

Why Biasing Your Network

The following table shows the consequences for unbiased networks:

If your network is unbiased...	Then ...
the voltage level at the receiver's A and B input may become less than 200 mV	the logic level at the output of the receivers is undefined.
the bus will not constantly change states	synchronization will be lost and framing errors will occur.
noise on the idle bus can cause random switches	framing errors or even interrupts can occur.

Definition of Bias Resistors

Bias resistors maintain the proper idle voltage state on a transmission line. They consist of 2 parts

- 1 pull-up resistor on the data D1 line (typically to 5 V)
- 1 pull-down (to 0 V) resistor on the data D0 line

The value of bias resistors depends on the termination and the number of nodes in the network. They must be calculated to generate enough DC bias current in the network to maintain a minimum of 200 mV between the D1 and D0 data lines.

How to Bias Your Network

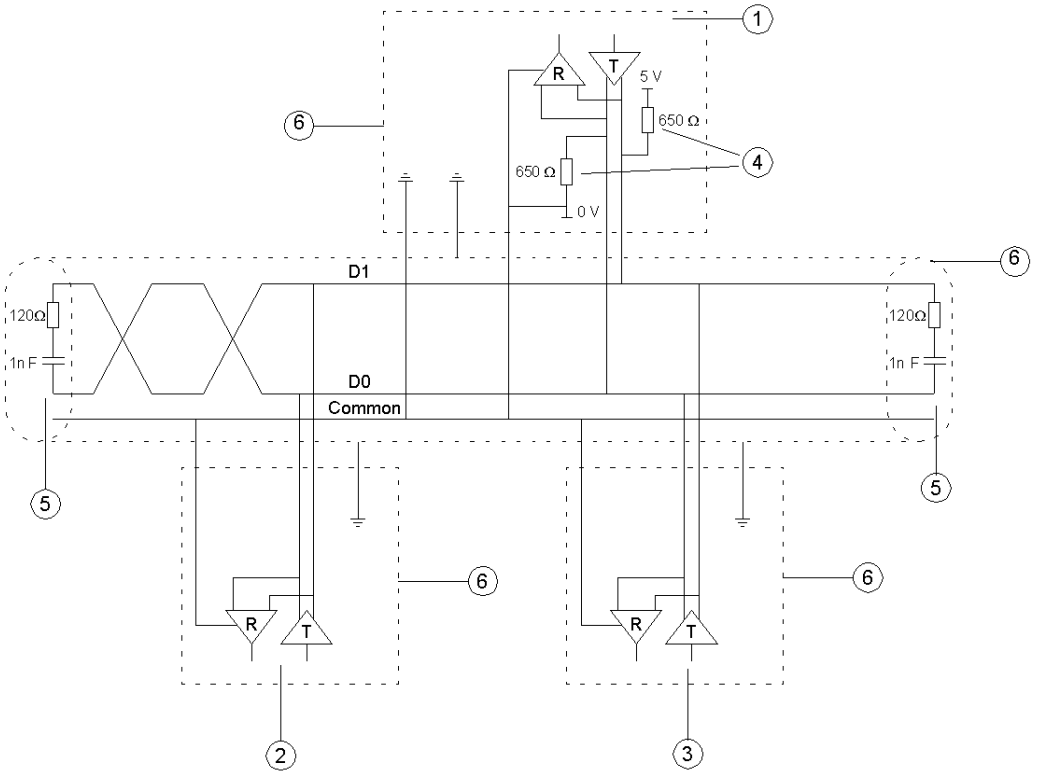
To guarantee constant line polarization, proceed as follows:

Step	Action
1	Check the devices you want to integrate into your Modbus SL application: Is there any device that needs external line polarization? If at least 1 of the devices needs external line polarization, proceed with step 2, otherwise no line polarization is required for your current application.
2	Integrate a pull-up resistor (650 Ω recommended) to a 5 V voltage into the D1 circuit.
3	Integrate a pull-down resistor (650 Ω recommended) to the common circuit into the D0 circuit.

Integrating Polarization Resistors into the Application

NOTE: This pair of polarization resistors must only be integrated at one location for the whole serial bus. You should integrate these resistors at the master device or its tap as shown in the figure below.

Schematic diagram



Elements of the application

No.	Element
1	master
2	slave 1
3	slave n
4	polarization resistors
5	line termination
6	shield

2.5 Termination

Overview

If your Modbus SL application requires high data rates and long cables, make sure to terminate the transmission lines properly in order to prevent unintended effects, like reflections.

There are different termination techniques available, but this document will only discuss RC termination.

What's in this Section?

This section contains the following topics:

Line Termination

Why Line Termination?

Proper termination eliminates reflections on the transmission line by providing the same impedance at the end of the cable as on the transmission line itself.

When Line Termination?

Since any reflections that occur on a line settle after about 3 round trip delays, the decision on whether line termination is required depends on the following 2 factors:

- cable length: Short cables have short round trip delays.
- transmission rate: Low data rates have long unit intervals.

This means that in applications with short cable lengths and low transmission rates the reflections settle before sampling.

Cable Length Determining Round Trip Time

The table below provides the round trip times for different cable lengths, assuming a given propagation velocity (value specified by the manufacturer for each cable) of 0.66xC (C = light speed).

Round trip times depending on cable lengths

Cable Length (m)	1	10	100	1000
Round Trip Time (µs)	0.01	0.10	1.01	10.10

Example: In an application using a 1000 m cable, one round trip covers 2000 m and is completed in approximately 10 µs.

Maximum Cable Length without Termination

The table below provides the maximum cable length without termination for different transmission rates, assuming the maximum reflection stabilization time does not exceed a quarter of a bit time.

Maximum cable lengths without termination

Transmission Rate	9600	19,200	57,600	115,200
Bit Time (µs)	104.17	52.08	17.36	8.68
Bit Time/4 (µs)	26.0	13.0	4.3	2.2
Max. Cable Length Without Termination (m)	859	430	143	72

Calculating the Maximum Cable Length without Termination

Calculating the maximum cable length without termination, assuming the maximum reflection stabilization time does not exceed a quarter of a bit time:

$$Lmax = \frac{0,66xC}{2} x \frac{1}{Nr} x \frac{Tb}{4}$$

Legend

Abbreviation	Meaning
Lmax	maximum line length without termination
C	light speed
Nr	number of round trips = 3
Tb	bit time

RC Termination

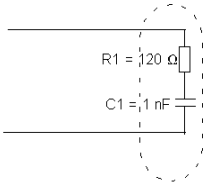
Overview

To prevent unintended effects, like reflections, from occurring in your Modbus SL application, make sure to terminate the transmission lines properly.

Use RC termination, as described in this section, to minimize the loop current and the line reflections. Furthermore RC termination increases the noise margin and supports open input receiver failsafe.

Terminating Your Network with RC Termination

To terminate your network with RC termination, proceed as follows:

Step	Action
1	Choose 2 serial capacitors of 1 nF, 10 V minimum and two 120 Ω (0.25 W) resistors as line termination. Example of a serial capacitor and a resistor as line termination: 
2	Integrate these components at both ends of your Modbus SL communication line as shown in pos. 5 of the schematic diagram in section *Integrating Polarization Resistors into the Application, page 25*.
3	Connect these line terminations between the 2 conductors of the balanced Modbus SL line.

2.6 Cable Length

Cable Length

Overview

The maximum length of cables used in Modbus SL applications is determined by different factors which will be listed in this section. Restrictions apply to the trunk cable (bus) as well as to the individual derivation cables.

Factors Influencing the Length of the Trunk Cable

The following factors influence the length of the trunk cable:

- transmission rate
- cable type (gauge, capacitance or characteristic impedance)
- number of loads that are directly connected (daisy chaining)
- network configuration (2-wire or 4-wire)

NOTE: If you are using a 4-wire cabling system for a 2-wire application, please note that the maximum cable length must be divided by two.

Cable Length Examples

The following table provides an example for determining the cable length according to the transmission rate and the cable type:

Transmission Rate:	19,200 bit/s
Cable Type:	0.125...0.161 mm^2 AWG26 (or wider)
Maximum Cable Length:	1000 m (3280 ft)

Expanding the Cable Length Using Repeaters

To expand the length of your Modbus SL trunk cable you can integrate repeaters in your system. With a maximum of 3 repeaters being allowed in 1 system, you can expand the allowed cable length by factor 4, i.e. to a maximum cable length of 4,000 m (13,123 ft).

Length of Derivation Cables

The length of each derivation cable must not exceed 20 m (65 ft).

If you are using a multi-port tap with n derivations, make sure that the maximum length of 40 m (313 ft) is not exceeded for all n derivations together.

2.7 Number of Devices to Connect

Number of Devices to Connect

Maximum Number of Devices without Repeater

An RS 485 network can principally have a maximum load of 32 devices.

Condition Allowing More Than 32 Devices

You can connect more than 32 devices to 1 RS 485 driver without repeater if you are using special RS 485 drivers in your application that support more than 32 devices.

Using a Repeater

If you want to connect more than 32 devices to a standard RS 485 driver, then integrate a repeater in your network.

2.8 Transmission Rates

Transmission Rates

Overview

The following transmission rates are available for RS 485 applications:

- 1,200 bit/s
- 2,400 bit/s
- 4,800 bit/s
- 9600 bit/s
- 19,200 bit/s (default)
- 38,400 bit/s
- 56 kbit/s
- 115 kbit/s
- 128 kbit/s
- 256 kbit/s

The actual transmission rate depends on the devices connected to the line.

2.9 Connectors

Overview

Different mechanical connectors are available for Modbus SL applications. The following connectors will be described in this section:

- screw terminals
- RJ-45 connectors
- SUB-D9 connectors
- M12 connectors

What's in this Section?

This section contains the following topics:

Screw Terminals

Overview

Screw terminals can be used in Modbus SL applications only for trunk interfaces, not for derivation interfaces.

Those screw terminals that can be removed are called open style connectors.

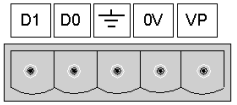
Two different types of screw terminals are available, depending on their ability to distribute direct voltage:

- 5-position screw terminals with VP distribution (standard version)
- 4-position screw terminals without VP distribution (reduced version)

Screw Terminals for Device Connectors

Screw terminals serving as connectors on devices must have male contacts.

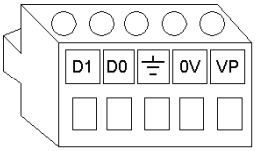
Schneider devices equipped with screw terminals have the following pinout:



Screw Terminals for Network Connectors

Screw terminals serving as network connectors on the cable side must have female contacts with the following pinout (with terminal 1 = D1 being on the left when looking at the wire entries).

Pinout for 5-position screw terminal (with VP distribution)



4-Position Screw Terminals

For 4-position connectors without direct voltage distribution use the same pinouts as described above omitting the VP pin.

RJ-45 Connector

Overview

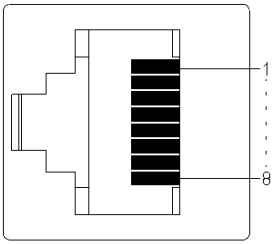
RJ-45 connectors can be used for 2-wire as well as for 4-wire Modbus SL applications. Make sure to use female connectors with the correct pin assignment for your application.

2-Wire RS 485 Applications

The following table shows the pin assignment of the RJ-45 socket for 2-wire RS 485 applications:

Pin	Signal	EIA / TIA 485 Name	Description
1	- - -	- - -	- - -
2	- - -	- - -	- - -
3	- - -	- - -	- - -
4	D1	B/'B	transceiver terminal 1, V1 voltage (V1 > V0 for binary 1 [OFF] state)
5	D0	A/A'	transceiver terminal 0, V0 voltage (V0 > V1 for binary 0 [ON] state)
6	- - -	- - -	- - -
7	VP	- - -	optional power supply
8	0 V	C/C'	common

Pin assignment of the Modbus RJ-45 socket

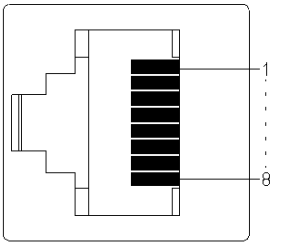


4-Wire RS 485 Applications

The following table shows the pin assignment of the RJ-45 socket for 4-wire RS 485 applications:

Pin	Signal	EIA / TIA 485 Name	Description
1	RXD0	A'	receiver terminal 0, Va' voltage (Va' > Vb' for binary 0 [ON] state)
2	RXD1	B'	receiver terminal 1, Vb' voltage (Vb' > Va' for binary 1 [OFF] state)
3	- - -	- - -	- - -
4	TXD1	B	generator terminal 1, Vb voltage (Vb >Va for binary 1 [OFF] state)
5	TXD0	A	generator terminal 0, Va voltage (Va >Vb for binary 0 [ON] state)
6	- - -	- - -	- - -
7	VP	- - -	optional power supply
8	0 V	C/C'	common

Pin assignment of the Modbus RJ-45 socket



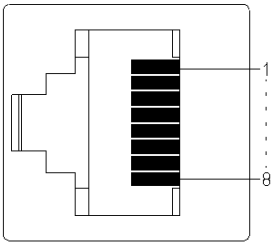
RS 232 Applications

The following table shows the pin assignment of the RJ-45 socket for RS 232 applications on a DCE device (e.g. a PLC):

Pin	Signal	Description
1	TXD	transmitted data
2	RXD	received data
3	CTS	clear to send
6	RTS	request to send
8	0 V	common

NOTE: On DTE devices (e.g. a PC) the pinouts are crossed.

Pin assignment of the Modbus RJ-45 socket



SUB-D9 Connectors

Overview

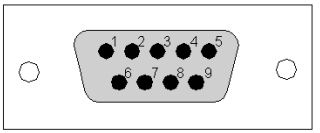
SUB-D9 connectors can be used for 2-wire as well as for 4-wire Modbus SL applications. Make sure to use male or female connectors with the correct pin assignment for your application.

2-Wire RS 485 Applications

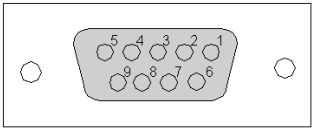
The following table shows the pin assignment of the SUB-D9 connector for 2-wire Modbus SL applications:

Pin	Signal	EIA / TIA 485 Name	Description
1	0 V	C/C'	common
2	VP	- - -	optional power supply
3	- - -	- - -	- - -
4	- - -	- - -	- - -
5	D1	B/B'	transceiver terminal 1, V1 voltage (V1 > V0 for binary 1 [OFF] state)
6	- - -	- - -	- - -
7	- - -	- - -	- - -
8	- - -	- - -	- - -
9	D0	A/A'	transceiver terminal 0, V0 voltage (V0 > V1 for binary 0 [ON] state)

Pin assignment of the male SUB-D9 connector



Pin assignment of the female SUB-D9 connector

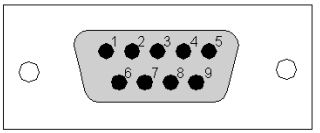


Pin Assignment for 4-Wire Applications

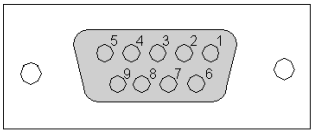
The following table shows the pin assignment of the SUB-D9 connector for 4-wire Modbus SL applications:

Pin	Signal	EIA / TIA 485 Name	Description
1	0 V	C/C'	common
2	VP	- - -	optional power supply
3	- - -	- - -	- - -
4	RXD1	B'	receiver terminal 1, Vb' voltage (Vb' > Va' for binary 1 [OFF] state)
5	TXD1	B	generator terminal 1, Vb voltage (Vb >Va for binary 1 [OFF] state)
6	- - -	- - -	- - -
7	- - -	- - -	- - -
8	RXD0	A'	receiver terminal 0, Va' voltage (Va' > Vb' for binary 0 [ON] state)
9	TXD0	A	generator terminal 0,Va voltage (Va >Vb for binary 0 [ON] state)

Pin assignment of the male SUB-D9 connector



Pin assignment of the female SUB-D9 connector



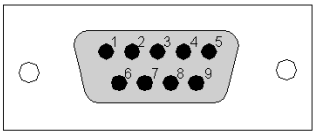
RS 232 Applications

The following table shows the pin assignment of the SUB-D9 connector for RS 232 applications on a DCE device (e.g. a PLC):

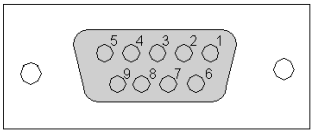
Pin	Signal	Description
2	TXD	transmitted data
3	RXD	received data
5	0 V	common
7	CTS	clear to send
8	RTS	request to send

NOTE: On DTE devices (e.g. a PC) the pinouts are crossed.

Pin assignment of the male SUB-D9 connector



Pin assignment of the female SUB-D9 connector



M12 Connector

Overview

M12 connectors have been standardized by Schneider Electric for 2-wire Modbus SL applications. Male as well as female M12 connectors are available.

Pin Assignment

The following table shows the pin assignment of the M12 socket for 2-wire Modbus SL applications:

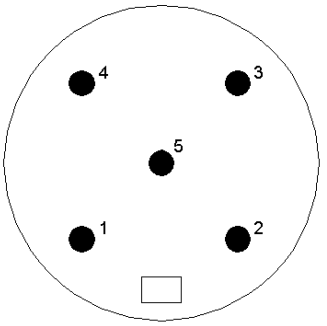
Pin	Signal	EIA / TIA 485 Name	Description
1	shield	- - -	optional shield connection
2	VP	- - -	optional power supply
3	0 V	C/C'	common
4	D0	A/A'	transceiver terminal 0, V0 voltage (V0 > V1 for binary 0 [ON] state)
5	D1	B/B'	transceiver terminal 1, V1 voltage (V1 > V0 for binary 1 [OFF] state)
Shell	- - -	- - -	- - -

Male Connectors

Select male M12 connectors for the following application devices:

- IP67 devices
- 1 trunk interface of each tap / multi tap device
- the power input connector of each tap / multi tap device
- 1 extremity of each IP67 Modbus SL cable

M12 male connector

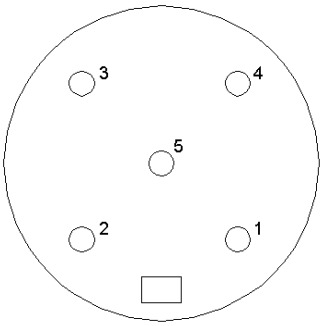


Female Connectors

Select female M12 connectors for the following application devices:

- 1 trunk interface of each tap / multi tap device
- the derivation interface of each tap / multi tap device
- 1 extremity of each IP67 Modbus SL cable

M12 female connector



2.10 Power Supply via Modbus SL

Power Supply via Modbus SL

Overview

Modbus RJ-45, M12 as well as screw connectors provide a specific pin for power supply from PLCs, drives etc. to small HMI or PC connection accessories like, for example, RS 232/RS 485 converter cables. One supplier can provide a voltage range of 5 V–24 V to one or more consumers, thus avoiding the need for an external power supply and an additional cable.

Voltages Provided by the Different Connector Types

The table below provides an overview of the individual connector types and the voltages they provide on the respective pins.

Connector Type	Voltage Range	Pins for Power Supply
screw terminals	24 V	VP and common (see *Screw Terminals, page 34*)
RJ-45	5–12 V	VP = pin 7 0 V = pin 8
SUB-D9	5 V	VP = pin 2 0 V = pin 1
M12	24 V	VP = pin 2 0 V = pin 3

⚠ CAUTION

EQUIPMENT DAMAGE

Before you connect devices using the power supply function make sure that the voltage range provided by the supplier is compatible with the voltage required by the consumer.

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

If you only want to transmit data over the Modbus line use the RJ-45 Schneider Electric cable that does not connect pin 7 to avoid accidentally connecting the power supply pins.

Modbus Data Link Layer

3

Overview

This chapter contains any information concerning the data link layer for Modbus applications.

What's in this Chapter?

This chapter contains the following sections:

3.1 Master / Slave Protocol

Overview

The following sections provide general information on the master / slave communication principle and the two different modes the master can forward information to the individual nodes, i.e. the unicast and the broadcast mode. The last section of this chapter is dedicated to the addresses that can be used in Modbus applications and how they are distributed.

What's in this Section?

This section contains the following topics:

Master / Slave Protocol Principle

Overview

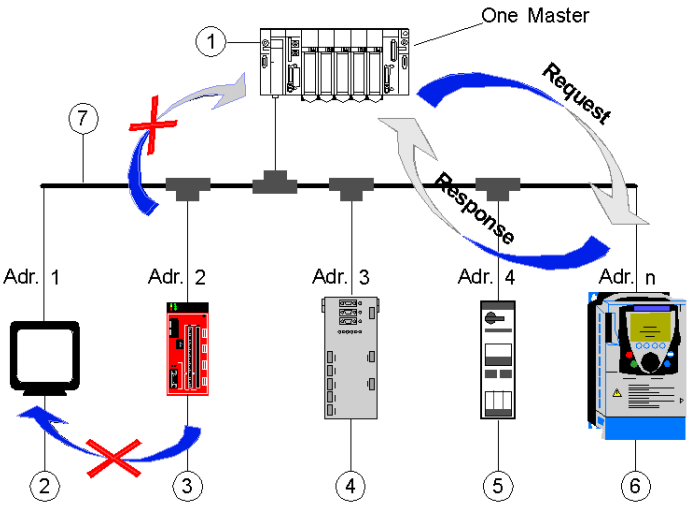
The Modbus SL protocol is working according to the master / slave principle on the application layer, i.e. the seventh layer of the OSI Model. The master of the Modbus serial line provides the client role whereas the slave nodes act as servers.

Characteristics of the Master / Slave Principle

The master / slave principle is characterized as follows:

- Only one master is connected to the bus at a time.
- One or several slave nodes can be connected to the same serial bus.
- Only the master is allowed to initiate communication, i.e. to send requests to the slave nodes.
- The master can only initiate one Modbus transaction at the same time.
- The master can address each slave node individually (unicast mode) or all slaves simultaneously (broadcast mode).
- The slave nodes can only answer requests from the master.
- The slave nodes are not allowed to initiate communication, neither to the master nor to any other slave nodes.

Master / slave communication



1 Automation Platform Premium PLC
2 Magelis Graphic Terminal (e.g. XBTG)
3 XPSMF40 Safety PLC
4 XPSMF30 Safety PLC
5 TesysU
6 Altivar 71
7 Modbus SL bus

Master / Slave Communication Modes

Overview

The master can address the individual slave nodes in the following 2 different communication modes:

- unicast mode
- broadcast mode

Unicast Mode

In unicast mode the master directly addresses a specific slave node individually.

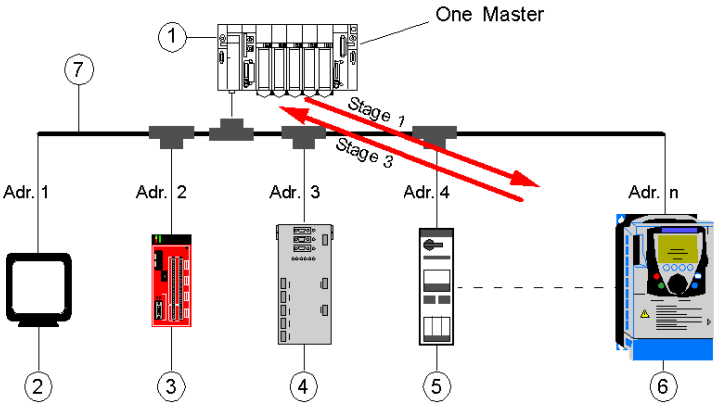
It is therefore required that each slave is assigned a unique address.

Stages

Communication in unicast mode consists of 3 stages:

Stage	Description
1	The master sends a request to an individual slave (stage 1 in figure below).
2	This slave processes the request from the master.
3	The slave sends a response message to the master (stage 3 in figure below).

The unicast mode principle



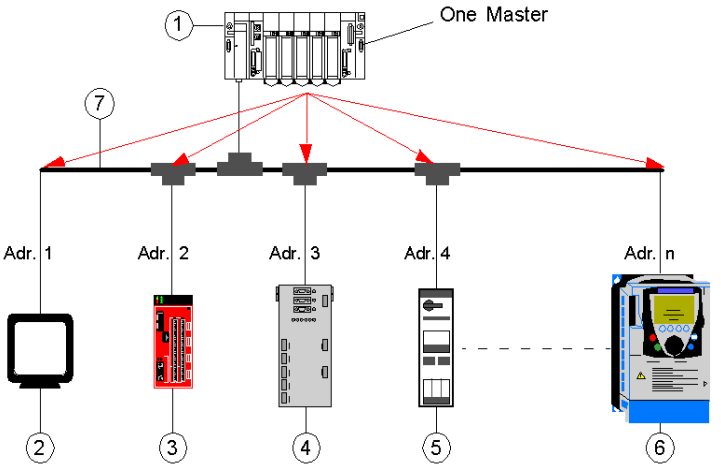
1 Master: Automation Platform Premium PLC
2 Address 1: Magelis Graphic Terminal (e.g. XBTG)
3 Address 2: XPSMF40 Safety PLC
4 Address 3: XPSMF30 Safety PLC
5 Address 4: TesysU
6 Address n: Altivar 71
7 Modbus SL bus

Broadcast Mode

In broadcast mode the master sends a request to all slave nodes. For broadcast messages the master uses the address 0.

All slave nodes in the network must accept broadcast messages that can clearly be identified by the address 0. The slave nodes only accept broadcast messages but do not reply to them.

The broadcast mode principle



1 Master: Automation Platform Premium PLC
2 Magelis Graphic Terminal (e.g. XBTG)
3 XPSMF40 Safety PLC
4 XPSMF30 Safety PLC
5 TesysU
6 Altivar 71
7 Modbus SL bus

Addressing Rules

Overview

Modbus SL supports up to 256 (0...255) different addresses. Each slave node must be assigned a slave address that is unique on the serial bus. The master node is not assigned a specific address.

Modbus addresses

Address	0	From 1...247	248	From 249...255
Function	Broadcast Address	Individual slave addresses	Supported on certain devices for point-to-point communications	Reserved

NOTE: Some devices use address 127 for point-to-point connections.

Broadcast Address

Address 0 is always reserved as broadcast address. Messages with this address must be recognized and accepted by all slave nodes.

3.2 Timeout Values

Overview

Two different timeout values can be configured in order to prevent the system from waiting for replies for too long time. The values response timeout for unicast and turnaround delay for broadcast mode will be described in the following sections.

What's in this Section?

This section contains the following topics:

Response Timeout for Unicast Mode

Definition

The response timeout is defined by a minimum of 3.5 characters (2 ms at 19,200 bit/s). It can be configured in Modbus devices for unicast mode. It should provide the slave nodes enough time to receive the request from the master, process this request and send a response to the master.

Turnaround Delay for Broadcast Mode

Definition

The turnaround delay can be configured in Modbus devices for broadcast mode. It should only provide the slave nodes enough time to process the request and to be able to receive a new one.

It can therefore be shorter than the response timeout for unicast mode. You should usually select a value between 100 and 200 ms.

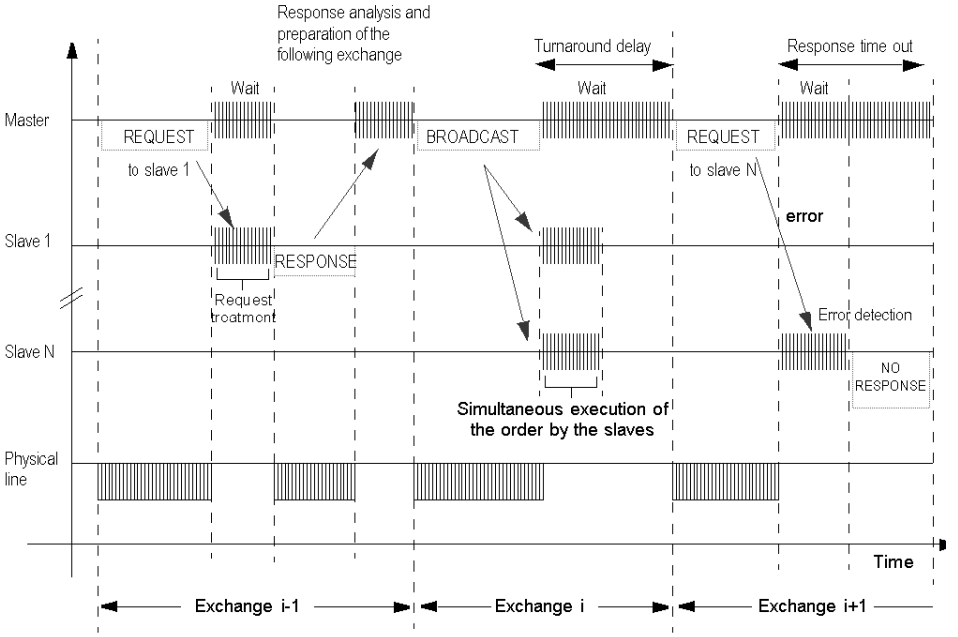
Communication Time Diagram

Overview

The figure below shows the Modbus communication over time. It provides an overview of the durations of REQUEST, RESPONSE and BROADCAST phases. The actual length of these phases in your network depends on the individual frame length as well as on the throughput.

The actual length of the WAIT and TREATMENT phases is determined by the request processing time required by your slave application.

Master/slave communication time diagram



3.3 Transmission Modes

Overview

Modbus SL serial data transmission can be performed in the following two different modes that individually define the contents and format of the message fields that are transmitted via the network as well as the detection of start and end points of messages:

- RTU mode
- ASCII mode

The RTU mode is by default integrated in all Modbus devices. The ASCII mode can additionally be implemented for specific applications.

NOTE: Configure the same transmission mode in all devices on one Modbus serial line.

What's in this Section?

This section contains the following topics:

RTU Transmission Mode

Overview

RTU is the standard transmission mode that must by default be implemented in every Modbus device.

In this transmission mode each 8-bit byte of a message contains 2 x 4-bit hexadecimal characters. This greater character density allows an improved data throughput compared with the ASCII transmission mode for the same transmission rate.

Byte Format

Each byte (11 bits) has the following format

Coding System	8-bit binary
Bits per Byte	1 start bit 8 data bits, least significant bit sent first 1 bit for parity completion 1 stop bit
Parity	even parity (default) odd parity no parity (requires 2 stop bits)

Serially Transmitting Characters

In serial data transmission each character or byte is sent as follows (left to right):

Least Significant Bit (LSB) ... Most Significant Bit (MSB)

Bit sequence in RTU mode with parity checking

NOTE: If you transmit data without parity checking please note that an additional stop bit is transmitted to form a complete 11-bit asynchronous character frame.

Bit sequence in RTU mode without parity checking

RTU Framing

Overview

A Modbus message is transmitted in a frame with a defined beginning and a defined end point.

RTU Frame

A Modbus message consists of a maximum of 256 characters, respectively bytes. Each message contains the following elements, as illustrated in the figure below:

- slave address
- function code
- data to be transmitted
- Cyclic Redundancy Checking (CRC) checksum

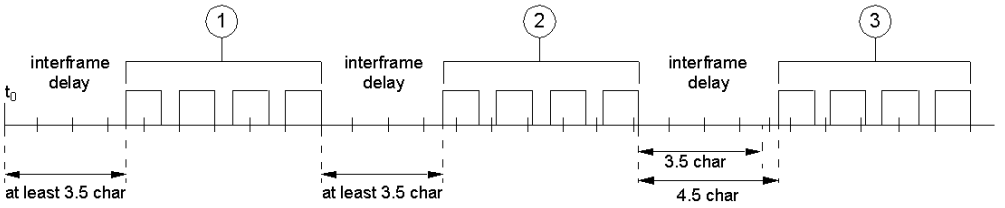
RTU message frame

Slave Address	Function Code	Data	CRC	
1 byte	1 byte	0...252 byte(s)	2 bytes	
			CRC Low	CRC Hi

Separating Message Frames by Silent Times

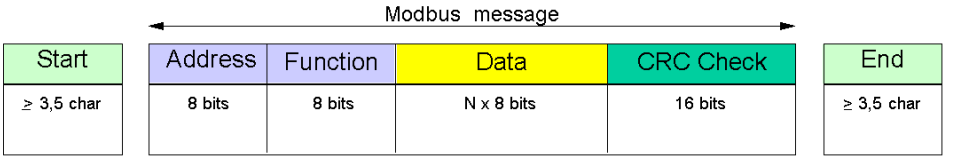
A Modbus message has a defined beginning and a defined end point. Individual frames are separated by a silent interval, also called interframe delay, of at least 3.5 character times. The following figure provides an overview of 3 frames being separated by an interframe delay of at least 3.5 character times.

Message frames separated by silent times



1 Frame 1
2 Frame 2
3 Frame 3

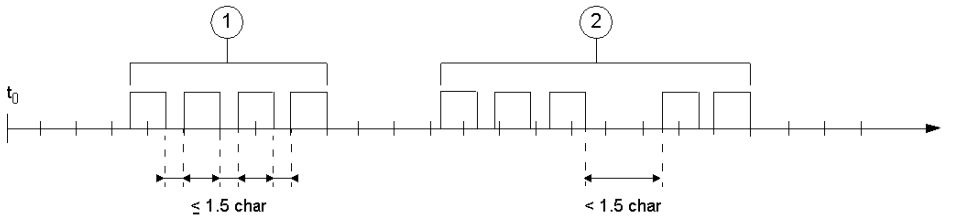
RTU message frame with start and end silent times



Detecting Incomplete Frames

In RTU mode it is required that the entire message frame is transmitted as a continuous stream of characters because silent times larger than 1.5 character times between 2 characters will be interpreted by the receiving device as incomplete frame. The receiver will discard this frame.

Detecting incomplete frames



1 Frame 1 OK
2 Frame 2 Not OK

RTU Framing

Overview

A Modbus message is transmitted in a frame with a defined beginning and a defined end point. This indicates to the receiving devices when a new message starts and when it is completed. The receiving devices can detect incomplete messages and inform the master by issuing errors.

RTU Frame

In addition to the user data, the RTU frame includes the following information:

- slave address (1 byte)
- function code (1 byte)
- Cyclic Redundancy Checking (CRC) field

The maximum size of an RTU frame is 256 bytes.

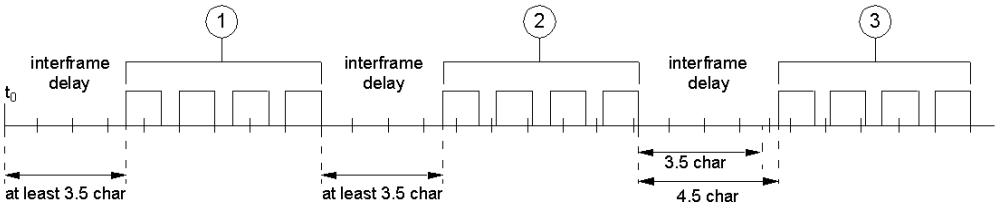
RTU message frame

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0...252 byte(s)	2 bytes
			CRC Low | CRC Hi

Separating Message Frames by Silent Times

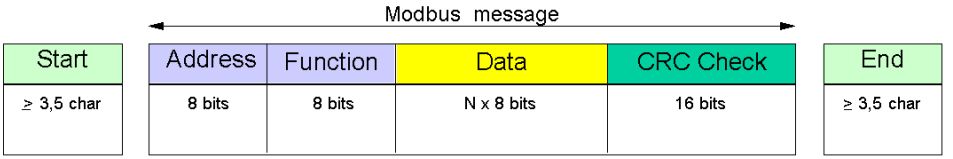
Individual frames are separated by a silent interval, also called interframe delay, of at least 3.5 character times. The following figure provides an overview of 3 frames being separated by an interframe delay of at least 3.5 character times.

Message frames separated by silent times



1 Frame 1
2 Frame 2
3 Frame 3

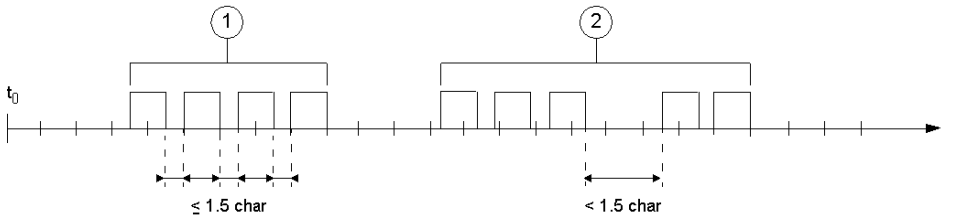
RTU message frame with start and end silent times



Detecting Incomplete Frames

In RTU mode it is required that the entire message frame is transmitted as a continuous stream of characters because silent times larger than 1.5 character times between 2 characters will be interpreted by the receiving device as incomplete frame. The receiver will discard this frame.

Detecting incomplete frames



1 Frame 1 OK
2 Frame 2 Not OK

CRC Error Checking In RTU Mode

Overview

An error checking field based on the Cyclic Redundancy Checking (CRC) method is appended as last field of each message by the master. The CRC field contains a 16-bit value implemented as two 8-bit bytes.

CRC Process

The CRC process is as follows:

1	The sending device calculates the CRC value.
2	The sending device appends the CRC as last field of the message with the low-order byte being the first and the high-order byte always being the last byte of the message to be sent.
3	The device receiving the message recalculates the CRC and compares the calculated value with the value integrated in the CRC field by the master.
4	If the two values are identical, the receiving device processes the message. If the value of the receiving device differs from the value in the CRC field, the message is regarded to be faulty. The receiving device will not process the message.

ASCII Transmission Mode

Overview

In case the physical communication link or the capabilities of the device do not allow the conformance with RTU mode requirements regarding timer management, devices can also be configured to communicate in ASCII (American Standard Code for Information Interchange) mode, even though this mode provides a less data throughput than the default RTU mode because each byte needs 2 characters.

Byte Format

Each byte (10 bits) has the following format

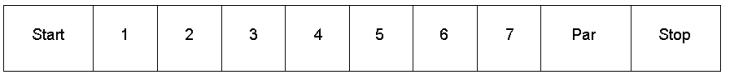
Coding System	hexadecimal ASCII characters 0-9, A-F
Bits per Byte	1 start bit 7 data bits, least significant bit sent first 1 bit for parity completion 1 stop bit
Parity	even parity (default) odd parity no parity (requires 2 stop bits)

Serially Transmitting Characters

In serial data transmission each character or byte is sent as follows (left to right):

Least Significant Bit (LSB) ... Most Significant Bit (MSB)

Bit sequence in ASCII mode with parity checking



NOTE: If you transmit data without parity checking, please note that an additional stop bit is transmitted to form a complete 10-bit asynchronous character frame.

Bit sequence in ASCII mode without parity checking

ASCII Framing

Overview

A Modbus message is transmitted in a frame with a defined beginning and a defined end point.

ASCII Frame

In addition to the user data, the ASCII frame includes the following information:

- start bit
- slave address (1 byte)
- function code (1 byte)
- Longitudinal Redundancy Checking (LRC) field

The maximum size of an ASCII frame is 513 bytes.

ASCII message frame

Start	Address	Function	Data	LRC	End
1 char . .	2 chars	2 chars	0...2x252 char(s)	2 chars	2 chars CR,LF

Separating Message Frames by Specific Characters

In contrast to RTU framing, where individual messages are detected via silent intervals, the ASCII mode includes special characters indicating the start and the end of a frame.

Frame Start / Stop	**ASCII Character**
Start of a Frame	*colon character (:)* (ASCII `3A` hex)
End of a Frame	*carriage return - line feed* (`CRLF`) character pair (ASCII `0D` and `0A` hex)

You can change the *line feed* character, that indicates the end of a frame, by using a specific Modbus application command (for details refer to the document *Modbus Application Protocol Specification* available at *www.modbus.org*).

The remaining data fields (apart from the start and stop frames) may contain the characters hexadecimal `0-9`, `A-F` (ASCII coded).

The receiving devices continuously monitor the data stream on the bus for the colon character. If this character is detected, all following characters will be decoded until the character pair indicating the end of the frame is detected.

LRC Error Checking In ASCII Mode

Overview

An error checking field based on the Longitudinal Redundancy Checking (LRC) method is appended as last field of each message by the master. The LRC field contains an 8-bit binary value implemented as 1 byte.

LRC Process

The LRC process is as follows:

1	The sending device calculates the LRC value.
2	The sending device appends the LRC as last field of the message.
3	The device receiving the message recalculates the LRC and compares the calculated value with the value integrated in the LRC field by the master.
4	If the two values are identical, the receiving device processes the message. If the value of the receiving device differs from the value in the LRC field, the message is regarded to be faulty. The receiving device will not process the message.

Modbus Protocol

4

Overview

This chapter provides information about the Modbus protocol itself.

What's in this Chapter?

This chapter contains the following sections:

4.1 Overview of the Modbus Protocol

Overview

This chapter provides information about the Modbus protocol. For more detailed information refer to the Modbus Application Protocol Specification V1.1b available at *www.modbus.org*.

What's in this Section?

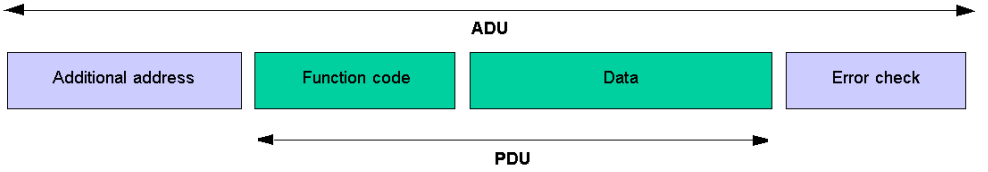
This section contains the following topics:

Modbus Frame Description

Overview

The basic Modbus frame consists of the protocol data unit (PDU) that may be extended by additional fields, depending on the specific application. This extended frame is called application data unit (ADU).

Modbus frame



Protocol Data Unit (PDU)

The protocol data unit includes the two basic parts of each Modbus frame:

- the function code
- the data field

Size of Modbus PDU

The size of the Modbus PDU depends on the serial line network that is used for the Modbus application.

The size of the Modbus PDU in RS 485 and RS 232 applications calculates as follows:

Total Frame Size (ADU)	**–**	**Server Address**	**–**	**Error Check**	**=**	**PDU Size**
256 bytes	–	1 byte	–	2 bytes	=	253 bytes

PDU Types

The following three different PDU types are available in Modbus applications:

- request PDU (mb_req_pdu)
- response PDU (mb_rsp_pdu)
- exception response PDU (mb_excep_rsp_pdu)

Request PDU

The request PDU consists of

- the function code
- the request data

These frame segments have the following size and content:

Frame Segment	Size	Description
function code	1 byte	This field contains the Modbus function code requested by the master.
request data	n bytes	The content of this field depends on the function code. It contains, for example, variable references, variable counts, data offsets, sub-function codes.

Response PDU

The response PDU consists of

- the function code
- the response data

These frame segments have the following size and content:

Frame Segment	Size	Description
function code	1 byte	This field contains the Modbus function code.
response data	n bytes	The content of this field depends on the function code. It contains, for example, variable references, variable counts, data offsets, sub-function codes.

Exception Response PDU

The exception response PDU consists of

- the error code
- the exception code

These frame segments have the following size and content:

Frame Segment	Size	Description
exception function code	1 byte	This field contains the Modbus function code + 0x80.
exception code	1 byte	This field contains the Modbus exception code that provides more detailed information about the kind of error occurred. For further details refer to *Exception Responses, page 93* or the *Modbus Application Protocol Specification V1.1b*.

Application Data Unit (ADU)

In RS 232 / RS 485 applications the PDU is extended by the following two frame segments:

- the additional address field
- the error check field

The PDU being extended by these fields is now called application data unit (ADU).

Size of Modbus ADU

The size of the Modbus ADU is always the entire size provided by the serial line network that is used for the Modbus application.

In RS 485 and RS 232 applications the Modbus ADU size is 256 bytes.

Function Code

Overview

The function code field is part of the PDU and thus always included in a Modbus frame.

Its size is always 1 byte.

It may contain valid codes between 1 and 127 decimal. Code 128 to 255 are reserved for exception responses. The code 0 is invalid.

Different function codes are available for requests sent by the master to the slave and for responses sent from the slave to the master. For details on the individual function codes see *Function Codes, page 76* or the *Modbus Application Protocol Specification V1.1b*.

Function Codes in Requests

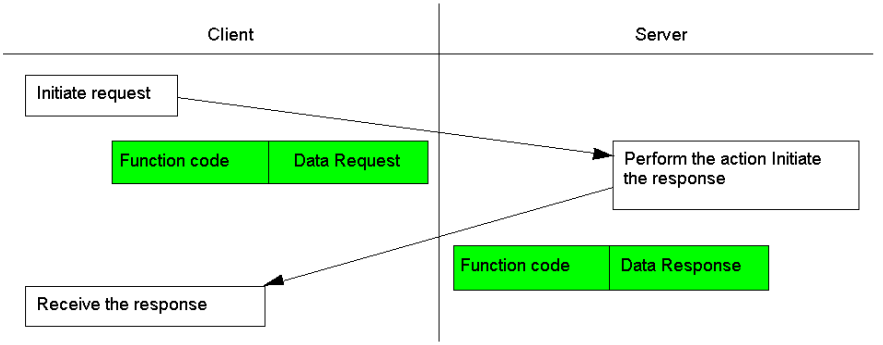
In requests from the master to the slave(s), the function code specifies the action the client should perform. It is located in the first byte of the PDU. It may also include sub-function codes for multiple actions.

Function Codes in Responses

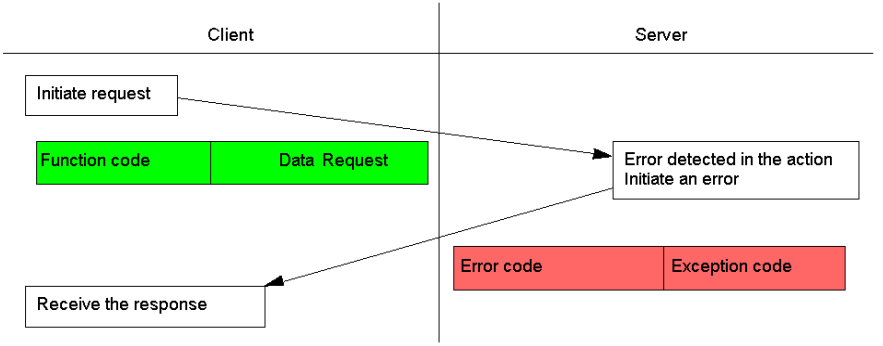
The content of function code fields in responses from slave to master depends on the fact whether an error occurred or not.

If...	Then ...
no error occurs within the request of the master	the slave returns a normal (error-free) response, which means that the original function code is returned to the master.
an error occurs within the request of the master	the slave returns an exception function code to the master. For further information on function codes refer to section *Function Codes, page 76*.

Modbus normal (error-free) transaction



Modbus exception (error) transaction



Data Field

Data Fields in Requests

In requests from the master to the slave, the data field can include information or it can be empty.

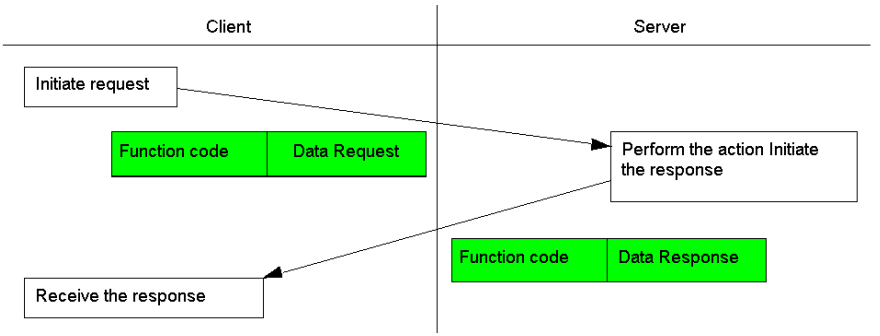
- The data field can provide the slave with additional information concerning the requested function code. This information may be, for example, discrete and register addresses, the quantity of items to be handled, the count of actual data bytes in the field.
- The data field may also be of zero length, i.e. it may not contain any information, in case there is no additional information to be provided to the slave.

Data Fields in Responses

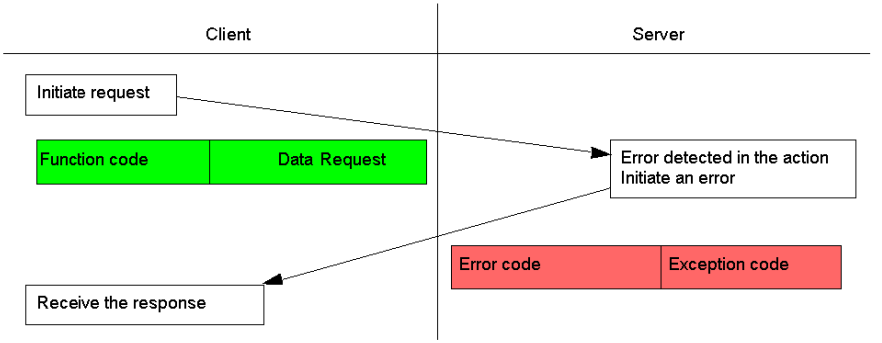
The content of data fields in responses from slave to master depends on the fact whether an error occurred or not.

If...	Then ...
no error occurs within the request of the master	the slave returns an error-free response and the data field contains the information requested by the master.
an error occurs within the request of the master	the slave returns an error response and the data field contains an exception code. For further information on exception codes refer to section *Exception Responses, page 93*.

Modbus normal (error-free) transaction



Modbus exception (error) transaction



4.2 Function Codes

Overview

The function code is the part of the Modbus frame that includes the information defining the action that should be performed by the slave device. This section lists those function codes that are supported by Schneider Modbus SL components and provides three detailed examples. It furthermore contains a list of exception codes included in exception responses to provide further information about the error that occurred while the slave was processing the request.

What's in this Section?

This section contains the following topics:

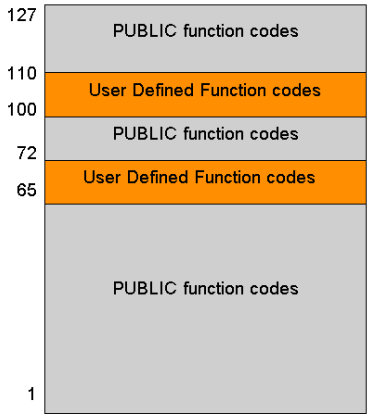
Function Code Categories

Overview

The 127 different function codes are divided into three different groups:

- public function codes
- user-defined function codes

Overview of function codes



Public Function Codes

Characteristics

Public function codes are defined by the following characteristics:

- well-defined
- guaranteed to be unique
- validated by the Modbus-IDA.org community
- publicly documented
- having conformance test

Both assigned function codes as well as unassigned function codes reserved for future use are available.

Overview of Public Function Codes

The following table lists the available public function codes.

			Function Codes		
			Code	Sub Code	(hex)
Bit Access	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**
	Internal Bits or Physical Coils	Read Coils	**01**		**01**
		Write Single Coil	**05**		**05**
		Write Multiple Coils	**15**		**0F**
16-Bits Access	Physical Input Registers	Read Input Register	**04**		**04**
	Internal Registers or Physical Output Registers	Read Holding Registers	**03**		**03**
		Write Single Register	**06**		**06**
		Write Multiple Registers	**16**		**10**
		Read/Write Multiple Registers	**23**		**17**
		Mask Write Register	**22**		**16**
		Read FIFO Queue	**24**		**18**
File Record Access		Read File Record	**20**		**14**
		Write File Record	**21**		**15**
Diagnostics		Read Exception Status	**07**		**07**
		Diagnostic	**08**	**00-18,20**	**08**
		Get Com Event Counter	**11**		**OB**
		Get Com Event Log	**12**		**0C**
		Report Slave ID	**17**		**11**
		Read Device Identification	**43**	**14**	**2B**
Other		Encapsulated Interface Transport	**43**	**13, 14**	**2B**

Examples

The following function codes will be described in the next sections as examples:

Name	Code	Description
Read Holding Registers	03	Reads several variables of any type from the slave's export area.
Diagnostics	08	Provides test functions for checking the communication between master and slave devices and the internal error conditions of slaves (Modbus SL, only).
Read Device Identification	43	Supplies the slave's identification data to the master.

For more detailed information refer to the Modbus Application Protocol Specification V1.1b available at *www.modbus.org.*

Function Code 03: Read Holding Registers

Overview

This function code is used by the master to read the contents of a contiguous block of holding registers from a slave device.

Request

In its request PDU, the master specifies the starting register address and the number of registers that should be read.

NOTE: In the PDU registers are treated as addresses starting at 0. This means that registers 1-16 are addressed as 0-15.

Request

Data	Size	Value
Function Code	1 byte	0x03
Starting Address	2 bytes	0x0000 ... 0xFFFF
Quantity of Registers	2 bytes	1 ... 125 (0x7D)

Response

In the response PDU, register data is packed as 2 bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second contains the low order bits.

Response

Data	Size	Value
Function Code	1 byte	0x03
Byte Count	1 byte	2 x N (N = quantity of registers)
Register Value	N x 2 bytes (N = quantity of registers)	–

Error Condition

In case the slave device cannot properly manage the request, it returns an exception response PDU to the master. The exception response PDU consists of the exception function code and the exception code as described in section *Exception Responses, page 93*.

Exception function code and exception code for FC03 Read Holding Registers looks as follows:

Data	Size	Value
Exception Function Code	1 byte	0x83
Exception Code	1 byte	01 or 02 or 03 or 04

Example

The following 2 tables provide an example of reading the values of registers 107...110.

Request

Field Name	Value (Hex)
Function	03
Starting Address Hi	00
Starting Address Lo	6B
No. of Registers Hi	00
No. of Registers Lo	03

Response

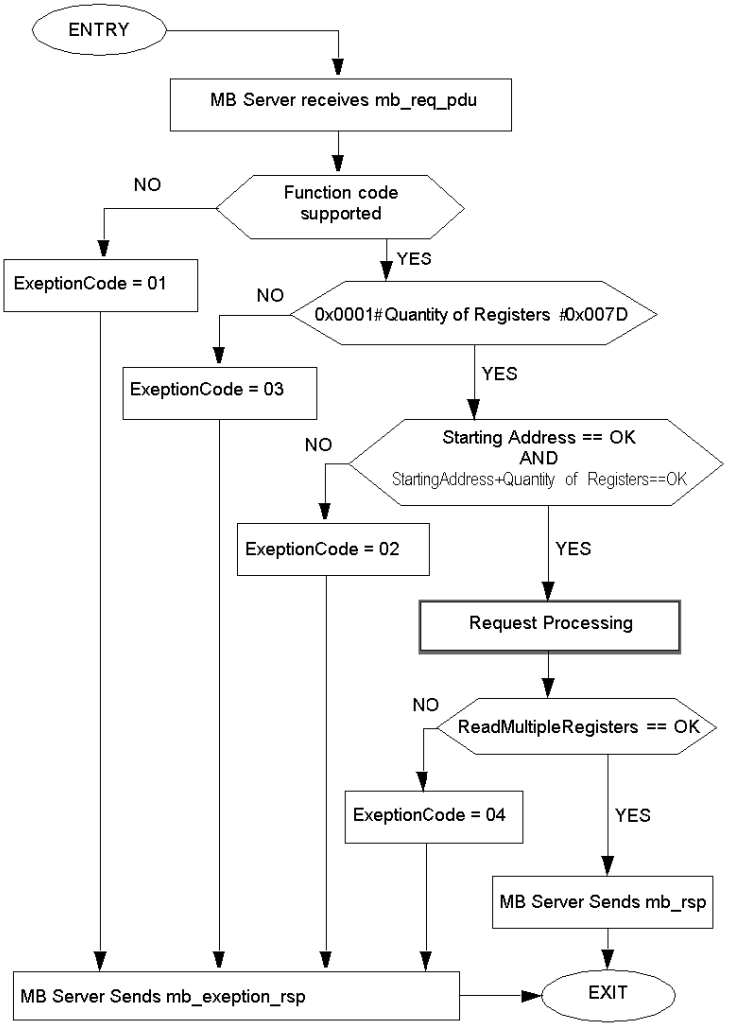
Field Name	Value (Hex)
Function	03
Byte Count	06
Register Value Hi (108)	02
Register Value Lo (108)	2B
Register Value Hi (109)	00
Register Value Lo (109)	00
Register Value Hi (110)	00
Register Value Lo (110)	64

The contents of register 107 are shown as the 2 byte values of 02 2B hex, or 555 decimal. The contents of registers 109...110 are 00 00 and 00 64 hex, or 0 and 100 decimal, respectively.

State Diagram

The following figure provides an overview of a slave device processing a request PDU with FC03 Read Holding Registers.

Read Holding Registers state diagram



Function Code 08: Diagnostics (Modbus SL, only)

Overview

This function code provides several tests for checking the communication between the master and the slave devices and for executing internal checks in the slave device. To define the test that should be executed, the function code includes several sub-function codes.

Normal responses from Modbus SL slave devices usually just loopback the data of the request, i.e. they contain the same function code and sub-function code as the request. Some responses additionally contain user data in the data field of the PDU.

Diagnostic requests usually not affect the user program running on the slave device. But diagnostic requests can

- reset the error counters of the slave devices and
- set slave devices into Listen Only Mode in order to prevent faulty devices from accessing the Modbus SL bus.

Overview of Sub-Function Codes

The following table provides an overview of the Diagnostics' sub-function codes defining the test that should be executed:

Hex Code	Dec Code	Name
00	00	Return Query Data
01	01	Restart Communications Option
02	02	Return Diagnostic Register
03	03	Change ASCII Input Delimiter
04	04	Force Listen Only Mode
–	05...09	Reserved
0A	10	Clear Counters and Diagnostic Register
0B	11	Return Bus Message Count
0C	12	Return Bus Communication Error Count
0D	13	Return Bus Exception Error Count
0E	14	Return Slave Message Count
0F	15	Return Slave No Response Count
10	16	Return Slave NAK Count
11	17	Return Slave Busy Count
12	18	Return Bus Character Overrun Count
13	19	Reserved
14	20	Clear Overrun Counter and Flag
–	21...65535	Reserved

The individual sub-function codes are described in the following sections.

00 Return Query Data

With this function code the slave device is requested to return the data passed in the request data field. The response data must be identical to the request data.

00 Return Query Data

Sub-Function	Data Field (Request)	Data Field (Response)
00 00	any	identical to request data

01 Restart Communications Option

With this function code the serial line port of the slave device is initialized and restarted. All counters of communications events in the slave device are cleared. The Restart Communications Option function code provides the only possibility to reactivate a slave device that is set to Listen Only Mode. This sub-function code provides the additional possibility to clear the communications event log file of this port. This is achieved by the entry `FF 00` hex in the request data field. If the request data field contains `00 00` the log file is not changed.

01 Restart Communications Option

Sub-Function	Data Field (Request)	Data Field (Response)
00 01	`00 00`	identical to request data
00 01	`FF 00`	identical to request data

Processing the Restart Communications Option in the Slave Device

The slave device processes the sub-function code `01` as follows:

Stage	Description
1	The slave device receives the request with the Restart Communications Option sub-function code `01`.
2	The slave device returns a normal response to the master if the slave is not in Listen Only Mode. (If the slave is in Listen Only Mode it skips this stage and does not send a response to the master.)
3	The slave device performs a restart and executes its power-up confidence tests. **Result**: After successful restart and testing the port of the slave device will be online.

02 Return Diagnostic Register

With this function code the slave device is requested to return the contents of its 16-bit diagnostic register.

02 Return Diagnostic Register

Sub-Function	Data Field (Request)	Data Field (Response)
00 02	`00 00`	contents of diagnostic register

03 Change ASCII Input Delimiter

This function code provides the possibility to replace the default end of message delimiter (i.e. the LF character) by a character CHAR.

03 Change ASCII Input Delimiter

Sub-Function	Data Field (Request)	Data Field (Response)
00 03	CHAR 00	identical to request data

04 Force Listen Only Mode

This function code sets a slave device into Listen Only Mode which means that it can no longer participate in Modbus SL communications.

In Listen Only Mode

- the active communication controls of the slave device are turned off.
- the Ready watchdog timer is allowed to expire and locks off the controls.
- the slave device can monitor any messages it receives and any broadcast messages but it cannot execute actions or send responses.
- the slave device can only process the Restart Communications Option function (function code 08, sub-function code 01). With the restart initiated by this command the Listen Only Mode will be disabled for the slave device.

04 Force Listen Only Mode

Sub-Function	Data Field (Request)	Data Field (Response)
00 04	00 00	no response returned

10 (0A Hex) Clear Counters and Diagnostic Register

This function code provides the possibility to clear all counters as well as the diagnostic register.

10 (0A hex) Clear Counters and Diagnostic Register

Sub-Function	Data Field (Request)	Data Field (Response)
00 0A	00 00	identical to request data

11 (0B Hex) Return Bus Message Count

This function code requests the slave device to return the number of messages it has detected on the communications link since its last restart, clear counters operation or power-up.

11 (0B hex) Return Bus Message Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 0B	00 00	total number of messages

12 (0C Hex) Return Bus Communication Error Count

This function code requests the slave device to return the number of CRC errors it has detected since its last restart, clear counters operation or power-up.

12 (0C hex) Return Bus Communication Error Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 0C	00 00	total number of CRC errors

13 (0D Hex) Return Bus Exception Error Count

This function code requests the slave device to return the total number of Modbus exception responses it has returned to the master since its last restart, clear counters operation or power-up (for details on exception responses see *Exception Responses, page 93*).

13 (0D hex) Return Bus Exception Error Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 0D	00 00	total number of exception responses

14 (0E Hex) Return Slave Message Count

This function code requests the slave device to return the total number of messages addressed to the slave device or broadcast messages it received and processed since its last restart, clear counters operation or power-up.

14 (0E hex) Return Slave Message Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 0E	00 00	total number of slave messages

15 (0F Hex) Return Slave No Response Count

This function code requests the slave device to return the total number of messages addressed to the slave device for which it has returned no response (neither a normal response nor an exception response) since its last restart, clear counters operation or power-up.

15 (0F hex) Return Slave No Response Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 0F	00 00	total number of slave messages without response

16 (10 Hex) Return Slave NAK Count

This function code requests the slave device to return the total number of messages for which it returned an NAK (Negative Acknowledge) exception response since its last restart, clear counters operation or power-up.

16 (10 hex) Return Slave NAK Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 10	00 00	total number of slave NAK exception responses

17 (11 Hex) Return Slave Busy Count

This function code requests the slave device to return the total number of messages for which it returned a Slave Device Busy exception response since its last restart, clear counters operation or power-up (for details on exception responses see *Exception Responses, page 93*).

17 (11 hex) Return Slave Busy Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 11	00 00	total number of Slave Device Busy exception responses

18 (12 Hex) Return Bus Character Overrun Count

This function code requests the slave device to return the total number of messages it received but it could not process due to a character overrun condition since its last restart, clear counters operation or power-up. A character overrun is caused by data characters arriving at the port faster than they can be stored, or by the loss of a character due to a hardware malfunction.

18 (12 hex) Return Bus Character Overrun Count

Sub-Function	Data Field (Request)	Data Field (Response)
00 12	00 00	total number of messages that could not be processed due to character overrun condition

20 (14 Hex) Clear Overrun Counter and Flag

This function code clears the overrun error counter and resets the error flag.

20 (14 hex) Clear Overrun Counter and Flag

Sub-Function	Data Field (Request)	Data Field (Response)
00 14	00 00	identical to request data

Example

The following table provides an example of requesting a slave device to Return Query Data (function code 08, sub-function code 00). The data to be returned is sent in the 2-byte data field (A5 37 hex).

Request

Field Name	Value (Hex)
Function	08
Sub-Function Hi	00
Sub-Function Lo	00
Data Hi	A5
Data Lo	37

Response

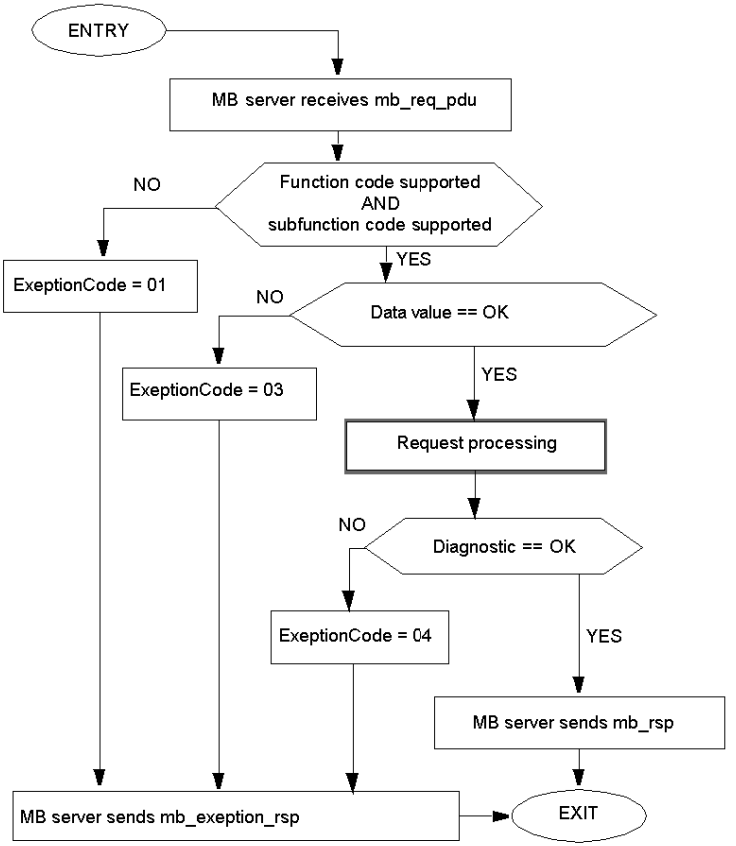
Field Name	Value (Hex)
Function	08
Sub-Function Hi	00
Sub-Function Lo	00
Data Hi	A5
Data Lo	37

Responses to other kinds of queries can contain the respective information requested by the sub-function code, i.e. the error count.

State Diagram

The following figure provides an overview of a slave device processing a request PDU with function code 08 Diagnostics.

Diagnostics state diagram



Function Code 43: Read Device Identification

Overview

This function code is used by the master to retrieve identification and additional information about the individual slave devices.

Read Device Identification

The XPSMF•• slave devices support the Modbus SL function code 43 Read Device Identification and provide the object-IDs listed in the tables below.

Basic:		
0x00	vendor name	Telemecanique
0x01	product code	XPSMF4020
0x02	major minor revision	<CPU Vx.y CRC / COM Vx.y CRC>

Regular:		
0x03	vendor URL	*http://www.schneider-electric.com*
0x04	product name	Preventa Safety PLC
0x05	model name	XPSMF40
0x06	user application name	<user application name>[S.R.S] from XPSMFWIN factory project

Extended:		
0x80	CPU BS version / CRC	<Vx.y / 0x234adcef>
0x81	CPU OSL version / CRC	<Vx.y / 0x234adcef>
0x82	CPU BL version / CRC	<Vx.y / 0x234adcef>
0x83	COM BS version / CRC	<Vx.y / 0x234adcef>
0x84	COM OSL version / CRC	<Vx.y / 0x234adcef>
0x85	COM BL version / CRC	<Vx.y / 0x234adcef>
0x86	configuration-CRC	<0x234adcef>

Exception Responses

Overview

Exception responses are returned by the slave device if it cannot handle a request from the master (e.g. the master requests reading a non-existent output or register). They provide information about the actual error to the master.

The exception response consists of

- the error code
- the exception code

These frame segments have the following size and content:

Frame Segment	Size	Description
Error Code	1 byte	This field contains the Modbus function code + 0x80. This means that the most significant bit (MSB) of the function code of exception responses is always 1 (their values are above 80 hexadecimal) whereas the MSB of the function code of normal responses is always 0 (below 80 hexadecimal). Thus, servers can easily detect exception responses and can evaluate the exception code to find out the reason for the error.
Exception codeC	1 byte	This field contains the Modbus exception code indicating the reason for the error state to the master. These exception codes are listed in the table below.

Exception Codes

The following Modbus exception codes are returned by the slave device to indicate different error states.

Code	Name	Description
01	ILLEGAL FUNCTION	The function code the slave device received in the query requests an action that is not allowed in the slave device. Possible reasons: • The function code is not supported by the slave device. • The slave device is in the wrong state, i.e. not correctly configured.
02	ILLEGAL DATA ADDRESS	The data address the slave device received in the query is not allowed for the slave device, i.e. the combination of reference number and transfer length is invalid. **Example** for a controller with 100 registers: A request with offset 96 and length 4 is processed by the slave. A request with offset 96 and length 5 will produce an exception response with exception code 02.
03	ILLEGAL DATA VALUE	A value of the query data field is not allowed for the slave device. This indicates a fault in the structure of the remainder of a complex request, e.g. the implied length may be incorrect.
04	SLAVE DEVICE FAILURE	An unrecoverable error occurred while the slave device was attempting to perform the requested action.
05	ACKNOWLEDGE	This exception code is frequently used with programming commands to indicate to the master that it will take the slave a while to process the request and to prevent timeout errors from occurring in the master. To find out whether the client has completed its task the master can next send a Poll Program Complete message to the slave.
06	SLAVE DEVICE BUSY	This exception code is frequently used with programming commands to indicate to the master that the slave is currently busy processing a long-duration program command. The master can send the request once again later when the slave has finished processing the command.

Code	Name	Description
08	MEMORY PARITY ERROR	This exception code is used especially with function codes 21 and 22 and reference type 6 to indicate that the extended file area failed to pass a consistency check. The slave device attempted to read a record file but detected a parity error in the memory. The master can try to send the request a second time but the slave device may require service.
0A	GATEWAY PATH UNAVAILABLE	This exception code is only used in conjunction with gateways to indicate that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. The gateway may not be correctly configured or may be overloaded.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	This exception code is only used in conjunction with gateways to indicate that the target device is not responding. The target device is probably not present in the network.

Transfer of 32-Bit Data (in Big-Endian/Little-Endian Format)

Order of Transfer

Big-endian and little-endian are terms that describe the order in which a sequence of words is transferred.

Big-endian is an order in which the "big end" (most significant value in the sequence) is transferred first (at the lowest storage address).

Little-endian is an order in which the "little end" (least significant value in the sequence) is transferred first.

32-Bit Data Transfer

XPSMF•• Safety PLCs transfer 32-bit data (`DWORD`, `DINT`, `REAL`) in big-endian format.

Other devices (e.g. Premium, Quantum, Magelis) transfer 32-bit data in little-endian format.

Compatibility Problems

⚠ CAUTION

UNINTENDED EQUIPMENT OPERATION

Be aware of compatibility problems while exchanging 32-bit data between a Safety Controller/PLC and other devices.

Different devices may use different transfer formats (big-endian, little-endian).

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

Diagnostics and Troubleshooting

5

Overview

This chapter is divided into two different sections, according to the two error types that can occur:

- communication errors (serial transmission errors)
- protocol errors (specific Modbus SL errors)

What's in this Chapter?

This chapter contains the following topics:

Communication Errors

Overview

In the case a communication error occurs, the slave device will not respond to a request and the master will generate a timeout error.

Communication Errors

The following errors are called communication errors:

- character timeouts (the allowed time was exceeded during the transmission of characters)
- parity errors (an error was detected during CRC checking)
- framing errors (an error was detected in the data frame)
- overrun errors (the receive register of the serial module is full)

Solutions

If a communication error occurs, check the following:

- Is the voltage supply switched on and the master for network operation started?
- Are the cable connections mechanically sound?
- Are the configured addresses correct?
- Are the same settings for baud rates and interface parameters (data bits, parity, stop bits) configured in the master as well as in the slave device?
- Has the network the right topology (termination, polarization, cable lengths)?

Protocol Errors

Overview

Protocol errors only occur while the slave device is responding a request from the master.

Master - Slave Procedure in Error Condition

In case a protocol error occurs, the master - slave procedure is as follows:

If...	Then ...
the slave device receives a request from the master	the slave device checks whether it can correctly execute the command.
the slave device can not correctly execute the requested operation	the slave device sends an exception code as a response to the master.

Exception Code

An exception code is always sent by the slave device as a response to the master when a protocol error occurs.

Responses with exception codes have the same function code as normal responses but differ in the following points:

- the Most Significant Bit (MSB) is set
- the function code is followed by a 1-byte exception code

The following table lists the values of function codes and exception codes with their meaning:

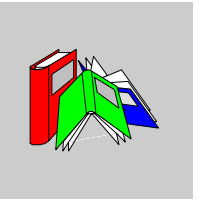
Field	Size (bytes)	Value	Meaning
Function Code	1	FC + 128 (80h) $03_h + 80_h = 83_h$ $08_h + 80_h = 88_h$ $10_h + 80_h = 90_h$ $17_h + 80_h = 97_h$ $2B_h + 80_h = AB_h$	Response code for errors FC3 FC8 FC16 FC23 FC43
MEI (only FC43)	1	14	Modbus encapsulated interface type (subfunction)
Exception Code	1	01_h...04_h	01_h = invalid function 02_h = invalid data addresses 03_h = invalid data 04_h = slave unit error

Causes for Protocol Errors

Protocol errors may have the following causes:

- unknown command, syntax error or incorrect data frame transmission
- parameter value outside the permissible value range
- action or control command not allowed during a running process
- error while executing an action or control command

Glossary



A

ADU

application data unit

ASCII

American standard code for information interchange = data transmission mode in Modbus communications

AWG

American wire gauge (wire diameter)

B

bps

bit/s

C

CRC

cyclic redundancy checking

CTS

clear to send (data transmission signal)

D

DCE

data communications equipment

DSR

data set ready (data transmission signal)

DTE

data terminal equipment

DTR

data terminal ready (data transmission signal)

E

EIA

Electronics Industry Alliance = trade association developing the standards RS 232 and RS 485

EN

European standards

ESD

electrostatic discharge

L

LRC

longitudinal redundancy checking

LSB

least significant bit

M

Modbus SL

Modbus serial line

MSB

most significant bit

N

NAK

negative acknowledge

O

OSI Model

open system interconnection model

P

PDU

protocol data unit

R

RJ-45

registered jack = standardized physical interface

RS 232

recommended standard for connecting serial devices = EIA/TIA 232

RS 485

recommended standard for connecting serial devices = EIA/TIA 485

RTS

request to send (data transmission signal)

RTU

remote terminal unit = data transmission mode in Modbus communications

RXD

receiving data (data transmission signal)

T

TXD

transmitting data (data transmission signal)

U

UL

unity load

Index

F

G

I

L

M

N

O

P

R

S

T

U

W