

SMAZ



Version RS-485

- Magnetic absolute linear encoder
- Measuring length up to 1250 mm / 49.212"
- Programmable resolution from 1.25 mm down to 0.1 mm
- MODBUS RTU (RS-485) interface
- Up to IP69K protection rate

Suitable for the following models:

- SMAZ-MB-...

Table of Contents

| | |
|-------------------------|----|
| Safety summary | 12 |
| Identification | 14 |
| Mechanical installation | 15 |
| Electrical connections | 19 |
| Quick reference | 25 |
| Modbus® interface | 44 |
| Programming parameters | 59 |
| Programming examples | 77 |

This publication was produced by Lika Electronic s.r.l. 2019. All rights reserved. Tutti i diritti riservati. Alle Rechte vorbehalten. Todos los derechos reservados. Tous droits réservés.

This document and information contained herein are the property of Lika Electronic s.r.l. and shall not be reproduced in whole or in part without prior written approval of Lika Electronic s.r.l. Translation, reproduction and total or partial modification (photostat copies, film and microfilm included and any other means) are forbidden without written authorisation of Lika Electronic s.r.l.

The information herein is subject to change without notice and should not be construed as a commitment by Lika Electronic s.r.l. Lika Electronic s.r.l. reserves the right to make all modifications at any moments and without forewarning.

This manual is periodically reviewed and revised. As required we suggest checking if a new or updated edition of this document is available at Lika Electronic s.r.l.'s website. Lika Electronic s.r.l. assumes no responsibility for any errors or omissions in this document. Critical evaluation of this manual by the user is welcomed. Your comments assist us in preparation of future documentation, in order to make it as clear and complete as possible. Please send an e-mail to the following address info@lika.it for submitting your comments, suggestions and criticisms.

The logo for Lika Electronic s.r.l. consists of the word "lika" in a bold, lowercase, sans-serif font. The letter "i" has a dot, and the letter "a" has a tail that curves slightly to the right.

General contents

| | |
|---|-----------|
| User's guide..... | 1 |
| General contents..... | 3 |
| Subject Index..... | 6 |
| Typographic and iconographic conventions..... | 7 |
| Preliminary information..... | 8 |
| Glossary of MODBUS terms..... | 9 |
| 1 Safety summary..... | 12 |
| 1.1 Safety..... | 12 |
| 1.2 Electrical safety..... | 12 |
| 1.3 Mechanical safety..... | 13 |
| 2 Identification..... | 14 |
| 3 Mechanical installation..... | 15 |
| 3.1 Overall dimensions..... | 15 |
| 3.2 Magnetic scale..... | 15 |
| 3.3 Mounting the sensor..... | 16 |
| 3.4 Measuring length..... | 18 |
| 3.5 Standard counting direction..... | 18 |
| 4 Electrical connections..... | 19 |
| 4.1 Connection scheme..... | 19 |
| 4.1.1 M8 cable specifications..... | 19 |
| 4.1.2 M12 8-pin connector..... | 20 |
| 4.2 Ground connection..... | 20 |
| 4.3 Node address..... | 21 |
| 4.4 Data transmission rate: baud rate and parity bit..... | 21 |
| 4.5 Bus termination resistor..... | 21 |
| 4.6 Diagnostic LED (Figure 3)..... | 22 |
| 5 Quick reference..... | 25 |
| 5.1 Getting started..... | 25 |
| 5.2 Configuring the encoder using the software tool by Lika Electronic..... | 25 |
| 5.3 Main page of the interface..... | 27 |
| 5.3.1 Configuring the serial port – Connection to the encoder..... | 28 |
| Read params..... | 30 |
| Write Holding..... | 31 |
| 5.3.2 Reading the Input Registers..... | 31 |
| Continuous reading..... | 32 |
| Current position..... | 32 |
| Speed..... | 32 |
| SW version..... | 32 |
| HW version..... | 32 |
| Status word..... | 33 |
| Alarm register..... | 33 |
| Machine data not valid..... | 33 |
| 5.3.3 Reading the exception responses – Exception error..... | 33 |
| Exception error..... | 33 |
| 5.3.4 Reading / writing the Holding Registers..... | 34 |

| | |
|---|-----------|
| Total resolution | 35 |
| Preset value | 35 |
| Offset value | 35 |
| Node address | 35 |
| Baud rate | 35 |
| Operating parameters | 36 |
| Enable scaling function..... | 36 |
| Change counting dir..... | 36 |
| Control word | 36 |
| Enable watchdog..... | 36 |
| Execute preset..... | 37 |
| Upload defaults..... | 37 |
| Auto save | 37 |
| Save parameters | 38 |
| 5.4 Update FW page - Firmware upgrade..... | 39 |
| 5.4.1 Information on firmware upgrade..... | 39 |
| 5.4.2 Preliminary operations and connections..... | 40 |
| 5.4.3 Launching the firmware upgrade process..... | 40 |
| 5.5 Manual frame page - Transmitting PDUs manually..... | 42 |
| 6 Modbus® interface | 44 |
| 6.1 Modbus Master / Slaves protocol principle..... | 44 |
| 6.2 Modbus frame description..... | 45 |
| 6.3 Transmission modes..... | 46 |
| 6.3.1 RTU transmission mode..... | 47 |
| 6.4 Function codes..... | 49 |
| 6.4.1 Implemented function codes..... | 49 |
| 03 Read Holding Registers | 49 |
| 04 Read Input Register | 51 |
| 06 Write Single Register | 53 |
| 16 Write Multiple Registers | 55 |
| 7 Programming parameters | 59 |
| 7.1 Parameters available..... | 59 |
| 7.1.1 Machine data parameters (Holding registers)..... | 59 |
| Resolution [0000 hex] | 59 |
| Preset value [0001 hex] | 61 |
| Offset value [0002 hex] | 63 |
| Operating parameters [0003 hex] | 63 |
| Scaling function | 64 |
| Code sequence | 64 |
| Node address [0004 hex] | 65 |
| Serial com baud rate [0005 hex] | 65 |
| Control Word [000A hex] | 66 |
| Watchdog enable | 66 |
| Save parameters | 67 |
| Load default parameters | 67 |
| Perform counting preset | 67 |
| 7.1.2 Input Register parameters..... | 69 |
| Alarms register [0000 hex] | 69 |
| Machine data not valid | 69 |
| Flash memory error | 69 |

| | |
|--|-----------|
| Hall sensors error..... | 69 |
| Mounting error..... | 69 |
| Watchdog..... | 70 |
| Current position [0001 hex]..... | 70 |
| Current velocity [0002 hex]..... | 70 |
| Wrong parameters list [0003 hex]..... | 71 |
| SW Version [0004 hex]..... | 71 |
| HW Version [0005 hex]..... | 72 |
| Status word [0006 hex]..... | 72 |
| Scaling..... | 72 |
| Counting direction..... | 72 |
| Alarm..... | 73 |
| 7.2 Exception response and exception codes..... | 74 |
| 8 Programming examples..... | 77 |
| 8.1 Using the 03 Read Holding Registers function code..... | 77 |
| 8.2 Using the 04 Read Input Register function code..... | 78 |
| 8.3 Using the 06 Write Single Register function code..... | 79 |
| 8.4 Using the 16 Write Multiple Registers function code..... | 80 |
| 9 Default parameters list..... | 81 |
| 9.1 List of the Holding Registers with default value..... | 81 |
| 9.2 List of the Input Registers..... | 81 |

Subject Index

A

| | |
|---------------------------------|----|
| Alarm..... | 73 |
| Alarm register..... | 33 |
| Alarms register [0000 hex]..... | 69 |
| Auto save..... | 37 |

B

| | |
|----------------|----|
| Baud rate..... | 35 |
|----------------|----|

C

| | |
|----------------------------------|----|
| Change counting dir..... | 36 |
| Code sequence..... | 64 |
| Continuous reading..... | 32 |
| Control word..... | 36 |
| Control Word [000A hex]..... | 66 |
| Counting direction..... | 72 |
| Current position..... | 32 |
| Current position [0001 hex]..... | 70 |
| Current velocity [0002 hex]..... | 70 |

E

| | |
|------------------------------|----|
| Enable scaling function..... | 36 |
| Enable watchdog..... | 36 |
| Exception error..... | 33 |
| Execute preset..... | 37 |

F

| | |
|-------------------------|----|
| Flash memory error..... | 69 |
|-------------------------|----|

H

| | |
|----------------------------|----|
| Hall sensors error..... | 69 |
| HW version..... | 32 |
| HW Version [0005 hex]..... | 72 |

L

| | |
|------------------------------|----|
| Load default parameters..... | 67 |
|------------------------------|----|

M

| | |
|-----------------------------|--------|
| Machine data not valid..... | 33, 69 |
| Mounting error..... | 69 |

N

| | |
|------------------------------|----|
| Node address..... | 35 |
| Node address [0004 hex]..... | 65 |

O

| | |
|--------------------------------------|----|
| Offset value..... | 35 |
| Offset value [0002 hex]..... | 63 |
| Operating parameters..... | 36 |
| Operating parameters [0003 hex]..... | 63 |

P

| | |
|------------------------------|----|
| Perform counting preset..... | 67 |
| Preset value..... | 35 |
| Preset value [0001 hex]..... | 61 |

R

| | |
|----------------------------|----|
| Read params..... | 30 |
| Resolution [0000 hex]..... | 59 |

S

| | |
|--------------------------------------|--------|
| Save parameters..... | 38, 67 |
| Scaling..... | 72 |
| Scaling function..... | 64 |
| Serial com baud rate [0005 hex]..... | 65 |
| Speed..... | 32 |
| Status word..... | 33 |
| Status word [0006 hex]..... | 72 |
| SW version..... | 32 |
| SW Version [0004 hex]..... | 71 |

T

| | |
|-----------------------|----|
| Total resolution..... | 35 |
|-----------------------|----|

U

| | |
|----------------------|----|
| Upload defaults..... | 37 |
|----------------------|----|

W




| | |
|---------------------------------------|----|
| Watchdog..... | 70 |
| Watchdog enable..... | 66 |
| Write Holding..... | 31 |
| Wrong parameters list [0003 hex]..... | 71 |

Typographic and iconographic conventions

In this guide, to make it easier to understand and read the text the following typographic and iconographic conventions are used:

- parameters and objects both of Lika device and interface are coloured in **GREEN**;
- alarms are coloured in **RED**;
- states are coloured in **FUCSIA**.

When scrolling through the text some icons can be found on the side of the page: they are expressly designed to highlight the parts of the text which are of great interest and significance for the user. Sometimes they are used to warn against dangers or potential sources of danger arising from the use of the device. You are advised to follow strictly the instructions given in this guide in order to guarantee the safety of the user and ensure the performance of the device. In this guide the following symbols are used:

| | |
|---|--|
|  | This icon, followed by the word WARNING , is meant to highlight the parts of the text where information of great significance for the user can be found: user must pay the greatest attention to them! Instructions must be followed strictly in order to guarantee the safety of the user and a correct use of the device. Failure to heed a warning or comply with instructions could lead to personal injury and/or damage to the unit or other equipment. |
|  | This icon, followed by the word NOTE , is meant to highlight the parts of the text where important notes needful for a correct and reliable use of the device can be found. User must pay attention to them! Failure to comply with instructions could cause the equipment to be set wrongly: hence a faulty and improper working of the device could be the consequence. |
|  | This icon is meant to highlight the parts of the text where suggestions useful for making it easier to set the device and optimize performance and reliability can be found. Sometimes this symbol is followed by the word EXAMPLE when instructions for setting parameters are accompanied by examples to clarify the explanation. |

Preliminary information

This guide is designed to provide the most complete information the operator needs to correctly and safely install and operate the **SMAZ absolute linear encoder fitted with MODBUS RTU interface**.

This sensor is designed to measure linear displacements in industrial machines and automation systems. The measurement system includes a magnetic scale and a magnetic sensor with conversion electronics. The scale is magnetized with a coded sequence of North-South poles generating a pseudo-random absolute pattern. As the sensor is moved along the magnetic scale, it detects the displacement and yields the absolute position information through the Modbus interface.

SMAZ is also available with SSI interface (SMAZ-BG..., SMAZ-GG...) or voltage (SMAZ-AV2...) / current (SMAZ-AI1...) analogue interface. SSI / analogue interface encoders are provided with their own technical documentation.

It is mandatory to pair the sensor with the **MTAZ type magnetic scale**. The measuring length is 1250 mm / 49.212", see the order code.

To make it easier to read the text, this guide can be divided into two main sections.

In the first section general information concerning the safety, the mechanical installation and the electrical connection as well as tips for setting up and running properly and efficiently the unit are provided.

While in the second section, entitled **MODBUS Interface**, both general and specific information is given on the Modbus interface. In this section the interface features and the registers implemented in the unit are fully described.

In the "Quick reference" section on page 25 the software tool designed by Lika Electronic to easily configure the encoder via RS-485 serial port is fully described.

Glossary of MODBUS terms

MODBUS, like many other networking systems, has a set of unique terminology. Table below contains a few of the technical terms used in this guide to describe the MODBUS interface. They are listed in alphabetical order.

| | |
|--------------------------------------|---|
| Address field | It contains the Slave address. |
| Application Process | The Application Process is the task on the Application Layer. |
| Application protocol | MODBUS is an application protocol or messaging structure that defines rules for organizing and interpreting data independent of the data transmission medium. |
| ASCII transmission mode | When devices are setup to communicate on a MODBUS serial line using ASCII (American Standard Code for Information Interchange) mode, each 8-bit byte in a message is sent as two ASCII characters. This mode is used when the physical communication link or the capabilities of the device does not allow the conformance with RTU mode requirements regarding timers management. |
| Bus | A bus is a communication medium connecting several nodes. Data can be transferred via serial or parallel circuits, that is, via electrical conductors or fibre optic. |
| Client | A Client is any network device that sends data requests to servers. MODBUS follows the Client/Server model. MODBUS Masters are referred to as Clients, while MODBUS Slaves are Servers. |
| Cyclic Redundancy Check (CRC) | Error-checking technique in which the frame recipient calculates a remainder by dividing frame contents by a prime binary divisor and compares the calculated remainder to a value stored in the frame by the sending node. |
| Data encoding | MODBUS uses a 'big-Endian' representation for addresses and data items. This means that when a numerical quantity larger than a single byte is transmitted, the most significant byte is sent first. |
| Exception code | Code to be returned by Slaves in the event of problems. All exceptions are signalled by adding 0x80 to the function code of the request. |
| Exception response | MODBUS operates according to the common client/server (Master/Slave) model: the Client (Master) sends a request telegram (service request) to the Server (Slave), and the Server replies with a response telegram. If the Server cannot process a request, it will instead return a error function code (exception response) that is the original function code plus 80H (i.e. with its most significant bit set to 1). |
| Function code | MODBUS is a request/reply protocol and offers services |

| | |
|----------------------------|--|
| | <p>specified by function codes. The function code is sent from a Client to the Server and indicates which kind of action the Server must perform. MODBUS function codes are elements of MODBUS request/reply PDUs.</p> <p>The function code field of a MODBUS data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (the range 128 – 255 is reserved and used for exception responses). Function code "0" is not valid. Like devices only implement public function codes.</p> |
| Holding register | In the MODBUS data model, a Holding register is the output data. A Holding register has a 16-bit quantity, is alterable by an application program, and allows either read-write or read-only access. |
| IEEE 1588 | This standard defines a protocol enabling synchronisation of clocks in distributed networked devices (e.g. connected via Ethernet). |
| Input register | In the MODBUS data model, an Input register is the input data. An Input register has a 16-bit quantity, is provided by an I/O system, and allows read-only access. |
| LRC Checking | In ASCII mode, messages include an error-checking field that is based on a Longitudinal Redundancy Checking (LRC) calculation that is performed on the message contents, exclusive of the beginning 'colon' and terminating CRLF pair characters. It is applied regardless of any parity checking method used for the individual characters of the message. |
| Master | A Master is any network device that sends data requests to Slaves. |
| Message | <p>The MODBUS messaging service provides a Client/Server communication between devices connected on the network. The Client / Server model is based on four types of messages:</p> <ul style="list-style-type: none"> • MODBUS Request • MODBUS Confirmation • MODBUS Indication • MODBUS Response <p>The MODBUS messaging services are used for information exchange.</p> |
| MODBUS Confirmation | A MODBUS Confirmation is the Response Message received on the Client side. |
| MODBUS Indication | A MODBUS Indication is the Request message received on the Server side. |
| MODBUS Request | A MODBUS Request is the message sent on the network by the Client to initiate a transaction. |
| MODBUS Response | A MODBUS Response is the Response message sent by the Server. |
| Network | Network is a group of computers on a single physical network segment. |

| | |
|---|--|
| PDU | <p>The Protocol Data Unit (PDU) is the MODBUS function code and data field. It is packed together with the Address Field and the CRC (or LRC) to form the Modbus Serial Line PDU.</p> <p>The MODBUS protocol defines three PDUs. They are:</p> <ul style="list-style-type: none"> • MODBUS Request PDU, mb_req_pdu • MODBUS Response PDU, mb_rsp_pdu • MODBUS Exception Response PDU, mb_except_rsp_pdu |
| Read Holding Registers (03, 0003hex) | This function code is used to READ the contents of a contiguous block of holding registers in a remote device; in other words, it allows to read the values set in a group of work parameters placed in order. |
| Read Input Register (04, 0004hex) | This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some result values and state / alarm messages in a remote device. |
| Register | MODBUS functions operate on memory registers to configure, monitor, and control device I/O. |
| RTU transmission mode | Remote Terminal Unit. When devices communicate on a MODBUS serial line using the RTU mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII mode for the same baud rate. Each message must be transmitted in a continuous stream of characters. |
| Server | <p>A Server is any program that awaits data requests to be sent to it. Servers do not initiate contacts with Clients, but only respond to them.</p> <p>MODBUS follows the Client/Server model. MODBUS Masters are referred to as clients, while MODBUS Slaves are servers.</p> |
| Service request | It is the MODBUS Request, i.e. the message sent on the network by the Client to initiate a transaction. |
| Slave | A Slave is any program that awaits data requests to be sent to it. Slaves do not initiate contacts with Masters, but only respond to them. |
| Transmission rate | Data transfer rate (in bps). |
| Write Multiple Registers (16, 0010hex) | This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device. |
| Write Single Register (06, 0006hex) | This function code is used to WRITE a single holding register in a remote device. |

1 Safety summary



1.1 Safety

- Always adhere to the professional safety and accident prevention regulations applicable to your country during device installation and operation;
- installation and maintenance operations have to be carried out by qualified personnel only, with power supply disconnected and stationary mechanical parts;
- device must be used only for the purpose appropriate to its design: use for purposes other than those for which it has been designed could result in serious personal and/or the environment damage;
- high current, voltage and moving mechanical parts can cause serious or fatal injury;
- warning ! Do not use in explosive or flammable areas;
- failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the equipment;
- Lika Electronic assumes no liability for the customer's failure to comply with these requirements.



1.2 Electrical safety

- Turn OFF power supply before connecting the device;
- connect according to explanation in the "Electrical connections" section on page 19;
- in compliance with 2014/30/EU norm on electromagnetic compatibility, following precautions must be taken:
 - before handling and installing the equipment, discharge electrical charge from your body and tools which may come in touch with the device;
 - power supply must be stabilized without noise; install EMC filters on device power supply if needed;
 - always use shielded cables (twisted pair cables whenever possible);
 - avoid cables runs longer than necessary;
 - avoid running the signal cable near high voltage power cables;
 - mount the device as far as possible from any capacitive or inductive noise source; shield the device from noise source if needed;
 - to guarantee a correct working of the device, avoid using strong magnets on or near by the unit;



- minimize noise by connecting the shield and/or the connector housing and/or the sensor to ground. Make sure that ground is not affected by noise. The connection point to ground can be situated both on the device side and on user's side. The best solution to minimize the interference must be carried out by the user;
- do not stretch the cable; do not pull or carry by cable; do not use the cable as a handle.



1.3 Mechanical safety

- Install the device following strictly the information in the "Mechanical installation" section on page 15;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the unit;
- do not tool the unit;
- delicate electronic equipment: handle with care; do not subject the device and the shaft to knocks or shocks;
- protect the unit against acid solutions or chemicals that may damage it;
- respect the environmental characteristics of the product;
- we suggest installing the unit providing protection means against waste, especially swarf as turnings, chips, or filings; should this not be possible, please make sure that adequate cleaning measures (as for instance brushes, scrapers, jets of compressed air, etc.) are in place in order to prevent the sensor and the magnetic scale from jamming.

2 Identification

Device can be identified through the **order code** and the **serial number** printed on the label applied to its body. Information is listed in the delivery document too. Please always quote the order code and the serial number when reaching Lika Electronic for purchasing spare parts or needing assistance. For any information on the technical characteristics of the product [refer to the technical catalogue](#).



Warning: devices having order code ending with "/Sxxx" may have mechanical and electrical characteristics different from standard and be supplied with additional documentation for special connections (Technical Info).

3 Mechanical installation



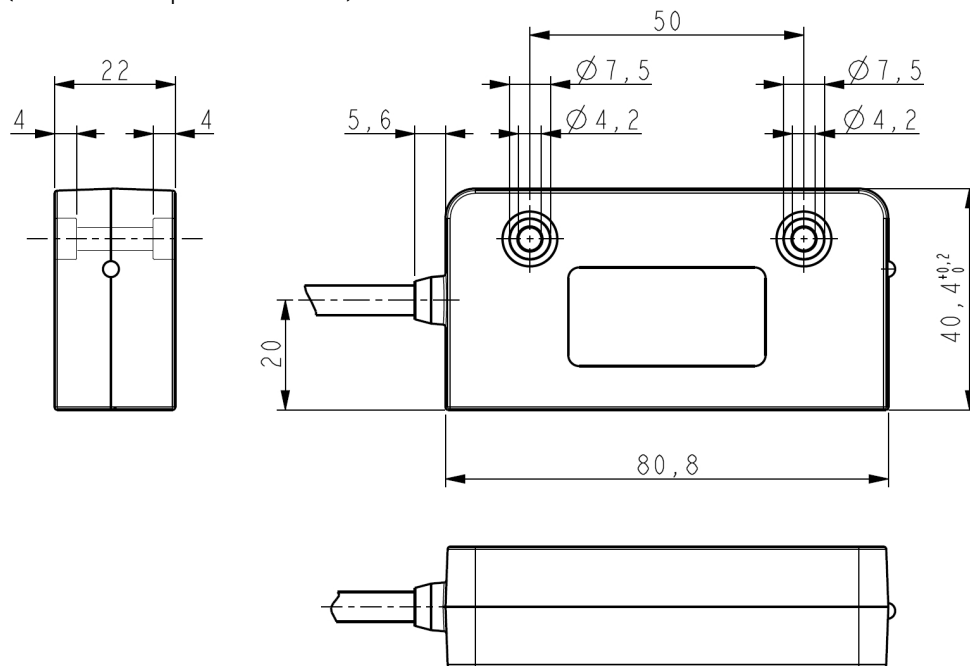
WARNING

Installation and maintenance operations have to be carried out by qualified personnel only, with power supply disconnected. Mechanical parts must be in stop.

For any information on the mechanical data and the electrical characteristics of the encoder please refer to the [technical catalogue](#).

3.1 Overall dimensions

(values are expressed in mm)



3.2 Magnetic scale

The sensor has to be paired with the **MTAZ type magnetic scale** only. For detailed information on the MTAZ type tape and how to mount it properly, please refer to the specific technical documentation.

Install the unit providing protection means against waste, especially swarf as turnings, chips or filings; should this not be possible, please make sure that

adequate cleaning measures (as for instance brushes, scrapers, jets of compressed air, etc.) are in place in order to prevent the sensor and the magnetic scale from jamming.

Make sure the mechanical installation meets the system's requirements concerning distance, planarity and parallelism between the sensor and the scale indicated in Figure 2 all along the whole measuring length.

MTAZ magnetic scale can be provided with a cover strip to protect its magnetic surface (see the order code).

Figure 1 shows how the sensor and the scale must be installed; the arrow indicates the **standard counting direction** (increasing count when the sensor moves in the direction indicated by the arrow; further information can be found in the "Code sequence" section on page 64).



WARNING

The system cannot operate if mounted otherwise than illustrated in Figure 1.

3.3 Mounting the sensor

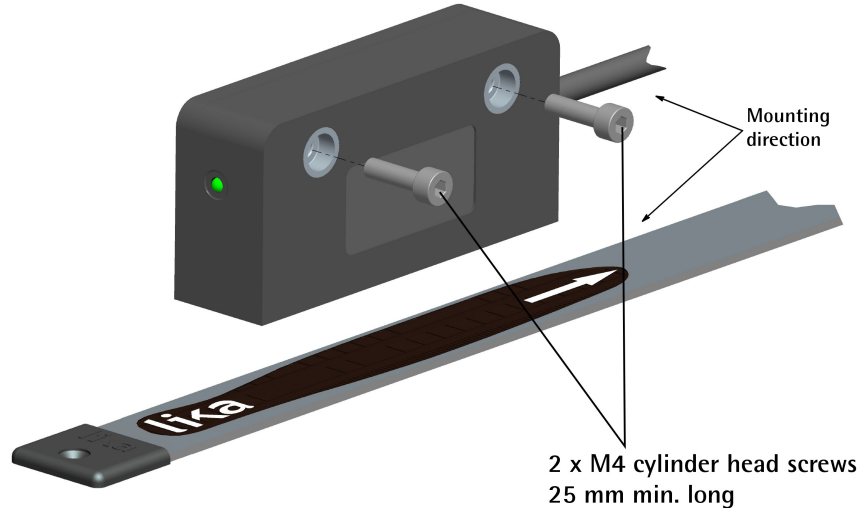


Figure 1

Make sure the mechanical installation complies with the system requirements concerning distance, planarity and parallelism between the sensor and the scale. Avoid contact between the parts. Sensor is fixed by means of **two M4 25 mm min. long cylinder head screws** inserted in the provided holes. Recommended **minimum bend radius** of the cable: **$R \geq 25 \text{ mm}$** . Install the sensor and the magnetic scale as shown in the Figure. The system does not operate if mounted

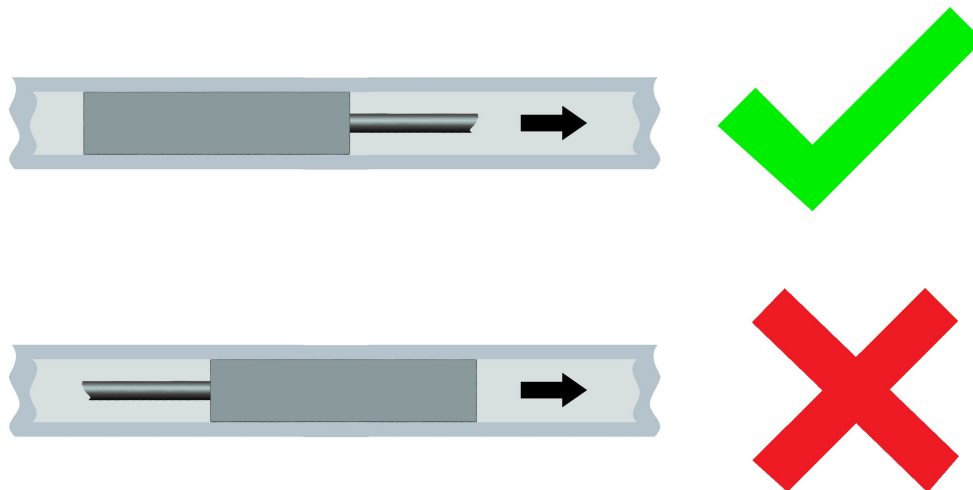
otherwise than illustrated in the Figure. The arrow is intended to indicate the standard counting direction (count up information).

Please note that the MTAZ magnetic scale can be provided with a cover strip to protect its magnetic surface (see the order code). Therefore the distance between the sensor and the magnetic scale is different whether the cover strip is applied.

The distance D (see Figure 2) between the centre of the screw fixing holes and the MTAZ magnetic scale has to be as follows:

| without cover strip | with cover strip |
|-------------------------------------|-------------------------------------|
| 31.7 mm ÷ 33.2 mm (1.248" ÷ 1.307") | 31.3 mm ÷ 32.8 mm (1.232" ÷ 1.291") |

For better operation the suggested distance D is 32.2 mm (1.267").



WARNING

Make sure the mechanical installation complies with the system requirements concerning distance, planarity and parallelism between the sensor and the scale as shown in Figure 2 all along the whole measuring length.

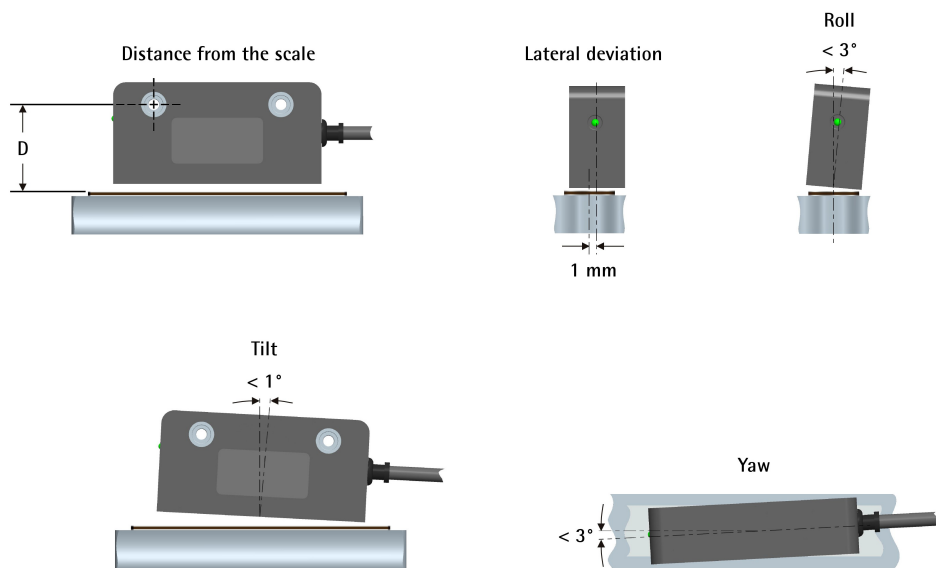


Figure 2



WARNING

After having installed the sensor on the magnetic scale a zero setting operation is compulsorily required. The zero setting operation is further required every time either the sensor or the scale is replaced. For any information on the zero setting operation please refer to the "Perform counting preset" section on page 67.

3.4 Measuring length

The **maximum length of the tape L** is 1330 mm / 52.362" (for further information refer the order code in the product datasheet). As the sensor area has always to be fully within the limits of the tape magnetic surface, then the **maximum measuring length ML** is the maximum length of the tape minus the length of the sensor head = $L - 80 \text{ mm} / 3.149"$ (1250 mm / 49.212").

3.5 Standard counting direction

The positive counting direction (count up information) is achieved when the sensor moves on the tape according to the white arrow shown in Figure 1. For further information see the "Code sequence" section on page 64.

4 Electrical connections



WARNING

Power supply must be turned off before performing any electrical connection!

For any information on the mechanical data and the electrical characteristics of the encoder please refer to the technical catalogue.

4.1 Connection scheme



WARNING

If wires of unused signals come in contact, irreparable damage could be caused to the device. Please insulate them singularly.

| Function | M8 cable | M12 8-pin |
|--------------------------------|----------|-----------|
| 0Vdc power supply ¹ | Black | 1 |
| +10Vdc +30Vdc power supply | Red | 2 |
| A_RS485 IN | Yellow | 3 |
| B_RS485 IN | Blue | 4 |
| A_RS485 OUT ² | Green | 5 |
| B_RS485 OUT ² | Orange | 6 |
| n.c. | White | 7 |
| n.c. | Grey | 8 |
| Shield | Shield | Case |

¹ 0Vdc of the RS-485 serial line too.

² In order to minimize cable reflections and ensure a defined noise level on the data lines a 120 Ω termination resistor must be provided between A_RS485 OUT and B_RS485 OUT if the encoder is the last slave in the line. For any information refer to the "4.5 Bus termination resistor" section on page 21.

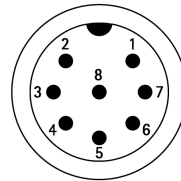
4.1.1 M8 cable specifications

| | |
|----------------|---|
| Model | : LIKA HI-FLEX sensor cable type M8 |
| Wires | : 2 x 0.22 mm ² + 6 x 0.14 mm ² (24/26 AWG) |
| Jacket | : Matt Polyurethane (TPU) halogen free, oil, hydrolysis, abrasion resistant |
| Shield | : tinned copper braid, coverage ≥ 85% |
| Outer diameter | : 5.3 mm ÷ 5.6 mm (0.209" ÷ 0.220") |

Min. bend radius : $\emptyset \times 7.5$
 Work temperature : $-40^{\circ}\text{C} + 90^{\circ}\text{C}$ ($-40^{\circ}\text{F} + 194^{\circ}\text{F}$) – dynamic installation
 : $-50^{\circ}\text{C} + 90^{\circ}\text{C}$ ($-58^{\circ}\text{F} + 194^{\circ}\text{F}$) – fixed installation
 Conductor resistance : $\leq 90 \Omega/\text{km}$ / $\leq 148 \Omega/\text{km}$

4.1.2 M12 8-pin connector

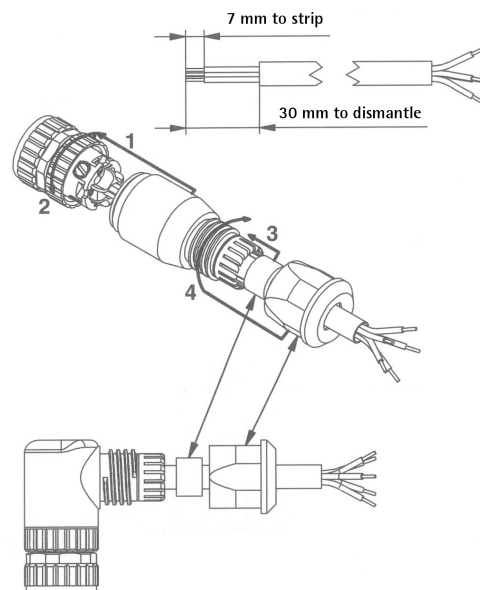
Male, frontal side



A coding

4.2 Ground connection

To minimize noise connect properly the shield and/or the connector housing and/or the frame to ground. Connect properly the cable shield to ground on user's side. Lika's EC- pre-assembled cables are fitted with shield connection to the connector ring nut in order to allow grounding through the body of the device. Lika's E- connectors have a plastic gland, thus grounding is not possible. If metal connectors are used, connect the cable shield properly as recommended by the manufacturer. Anyway make sure that ground is not affected by noise. It is recommended to provide the ground connection as close as possible to the device.



4.3 Node address

The node address of the device is set via software in the **Node address [0004 hex]** register (see on page 65).

The default address preset by Lika is 1. The address must have a value between 1 and 247.



NOTE

The default address is 1.

The address 0 is reserved to identify a "broadcast" exchange (Master sends a request to all Slaves connected to the Modbus network). See the "6.1 Modbus Master / Slaves protocol principle" section on page 44.

The Modbus Master node has no specific address, only the Slave nodes must have an address. Each Slave must have a unique address.

Addresses from 248 to 255 are reserved.

If you set the address 0, device will be set to 1 automatically.

Equally, if you set an address higher than 247, device will be set to 1 automatically.

4.4 Data transmission rate: baud rate and parity bit

The data transmission rate (baud rate and parity bit) is set via software in the **Serial com baud rate [0005 hex]** register (see on page 65). The default value is 04 hex = Baud rate 19,200 bit/s, Parity bit Even.

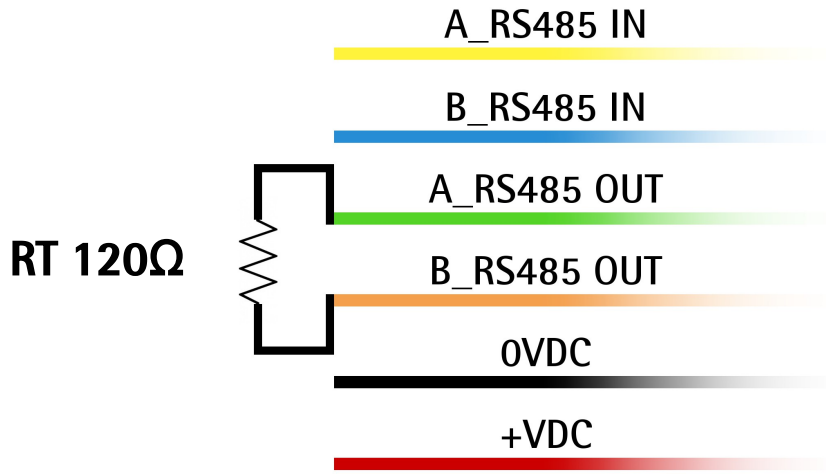
4.5 Bus termination resistor



WARNING

Power supply must be turned off before performing this operation!

As the RS-485 cables have an impedance of typically 120 Ω it is necessary to always add a termination resistor (RT) in order to minimize cable reflections and ensure a defined noise level on the data lines. The termination resistor must be installed at both physical ends of the line: at the beginning of the RS-485 communication bus (typically in the PLC) and at the end of the bus, in the last slave of the line. We recommend a 120 Ω termination resistor to be used. The termination resistor has to be provided between the A_RS485 OUT and the B_RS485 OUT signal wires in the last slave of the line, as shown in the Figure below.



4.6 Diagnostic LED (Figure 3)

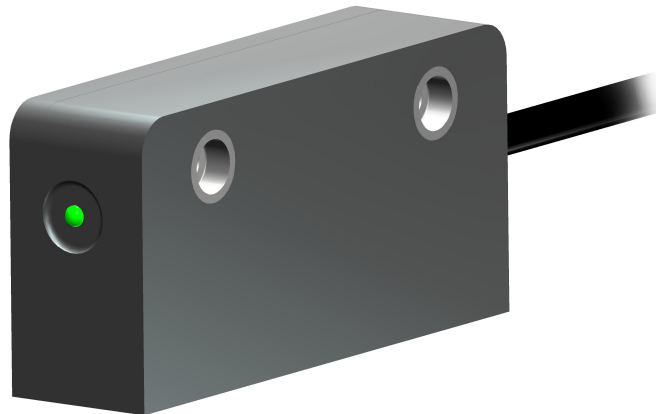


Figure 3: Diagnostic LED

A green LED located in the front of the encoder (see the Figure above) is meant to show visually the operating or fault status of the Modbus interface and the device as well. The LED operation is explained in the following table. In case of error, to know in detail which alarm has been triggered, see the [Alarms register \[0000 hex\]](#) variable on page 69.

| GREEN LED | Description |
|---|--|
| ON (Solid GREEN) | The encoder is operating properly, there are no active errors. |
| Blinking at high frequency (100 ms ON / 100 ms OFF) | Machine data parameters error. To see in detail which parameter is wrong please enter the Wrong parameters list [0003 hex] register on page 71. |
| Blinking slowly (500 ms ON / 500 ms OFF) | Flash memory error, it cannot be restored (bad checksum error, etc.). |
| Blinking very slowly (1 s ON / 1 s OFF) | An error has occurred in the Hall sensors while reading the magnetic scale, the read value does not exist. |
| Single flash (200 ms ON / 1 s OFF) | The encoder is installed too far from the magnetic scale, the installation does not comply with the mounting tolerances between the sensor and the scale (see Figure 2) Refer to the "Mechanical installation" section on page 15. |
| Double flash (200 ms ON/OFF twice / 1 s OFF) | Several errors are active at the same time. To know in detail which alarm has been triggered, see the Alarms register [0000 hex] variable on page 69. |

While performing the firmware upgrade operation (bootloading, refer to the "5.4 Update FW page - Firmware upgrade" section on page 39), the green LED operates in a specific way, as explained in the following table.

| GREEN LED | Description |
|--|---|
| Blinking at 2 Hz with duty cycle = 50% | The operator has pressed the BOOT STATE button in the Firmware Upgrade page, the encoder is waiting for the firmware upgrade operation to start. For any information please refer to the "5.4 Update FW page - Firmware upgrade" section on page 39. |
| Blinking at 5 Hz with duty cycle = 50% | The operator has pressed the DOWNLOAD button in the Firmware Upgrade page, the |

| | |
|---|--|
| | firmware upgrade operation is in progress. For any information please refer to the "5.4 Update FW page - Firmware upgrade" section on page 39. |
| Blinking at 10 Hz with duty cycle = 50% | An error occurred while performing the firmware upgrade operation. You must turn the power off and on again to reset the device and restart the operation. For any information please refer to the "5.4 Update FW page - Firmware upgrade" section on page 39. |
| ON (Solid GREEN) | The firmware upgrade operation has been carried out successfully, the encoder is operating properly and no error is active. For any information please refer to the "5.4 Update FW page - Firmware upgrade" section on page 39. |

5 Quick reference

5.1 Getting started

The following instructions are provided to allow the operator to set up the device for standard operation in a quick and safe mode.

- Install the device mechanically;
- perform the electrical connections;
- switch +10Vdc ÷ +30 Vdc power supply on;
- if needed, set the data transmission rate (baud rate and parity bit; see the **Serial com baud rate [0005 hex]** register on page 65); the default value set by Lika Electronic at factory set-up is "4" = baud rate 19,200 bit/s, parity bit Even;
- if needed, set the node address (node ID; see the **Node address [0004 hex]** register on page 65); the default value set by Lika Electronic at factory set-up is "1";
- if you want to use the default resolution of the unit (10 = 0.1 mm resolution), please check that the **Scaling function** item is disabled (bit 0 in the **Operating parameters [0003 hex]** register = 0; see on page 64);
- otherwise if you need a specific resolution, please enable the **Scaling function** item (bit 0 in the **Operating parameters [0003 hex]** register = 1; see on page 64) and then set the resolution you need for your application next to the **Resolution [0000 hex]** item (register 1; see on page 59);
- now, if you need you can set the Preset next to the **Preset value [0001 hex]** register and then execute the **Perform counting preset** command in **Control Word [000A hex]**; see on page 61;
- finally save the new setting values (**Save parameters** command bit 9 in the **Control Word [000A hex]** register; see on page 67).

5.2 Configuring the encoder using the software tool by Lika Electronic

SMAZ linear encoder with MODBUS interface is supplied with a software expressly developed by Lika Electronic in order to easily programme and configure the device. It allows the operator to set the working parameters of the device and monitor whether the device is running properly. The program is supplied for free and can be installed in any PC fitted with a Windows operating system (Windows XP or later). The name of the program executable file is **MODBUS-RTU.EXE**, it is available for download through the SOFTWARE link in

the page of the website dedicated to the device. The program is designed to be installed simply by copying the executable file to the desired location and **no installation process** is required. To launch the program just double-click the file icon. To close the program press the **CLOSE** button at the top right of the window.



NOTE

Before starting the program and establishing a communication with the device, it is necessary to connect it to the personal computer. The interface of the SMAZ encoder is a serial RS-485 type, the standard of the serial port in the personal computer (when the port is available) is the RS-232 type. Therefore you must install an RS-232 to RS-485 converter, easily available in the market. Should the personal computer not be equipped with a serial port (RS-232 or RS-485), you must install a USB to RS-485 converter, easily available in the market too. For complete information on the connection scheme and the cable pinout refer to the instruction sheet provided with the converter.

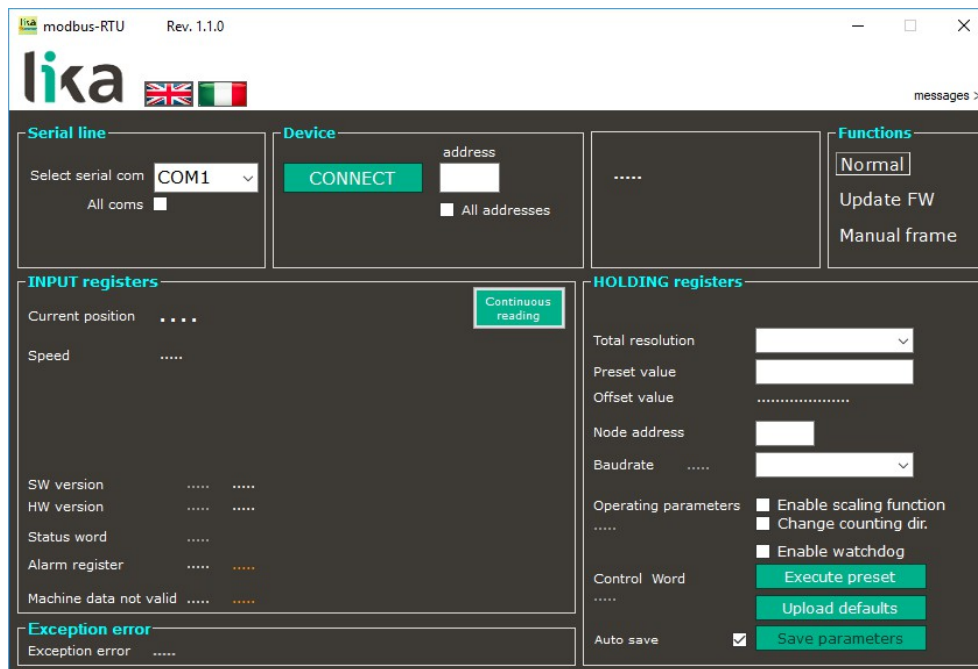
On the ENCODER side the cable must be connected as described in the "Electrical connections" section on page 19. Always be sure that RX wire in the MODBUS ENCODER is connected up to TX wire in the PC and RX wire in the PC is connected up to TX wire in the MODBUS ENCODER.

5.3 Main page of the interface

To launch the program and configure the MODBUS encoder double-click the **MODBUS-RTU.EXE** executable file.

The interface consists of a main page and two subpages.

When the program starts, the main page will appear on the screen.





The main page of the interface can be divided into seven parts.

1. In the **SERIAL LINE** group box on the top left of the page the items used to select the serial port for connecting to the device are available.
2. In the **DEVICE** group box on the top centre of the page the commands needful for setting the node address, for starting the connection process and, once the connection is established, for reading and writing the parameters are available.
3. In the group box on the right information about the device will be displayed as soon as the connection is established.
4. In the **FUNCTIONS** group box on the top right of the page you can find the buttons used to enter the main page, the page for firmware upgrade and the page that allows to send Request PDUs manually.
5. In the **INPUT REGISTERS** group box on the bottom left of the page the Input registers are available, they provide result values and status / alarm information on the device. These items are described in the "7.1.2 Input Register parameters" section on page 69.
6. In the **EXCEPTION ERROR** group box just beneath, the exception response messages that the Server transmits to the Client when an error

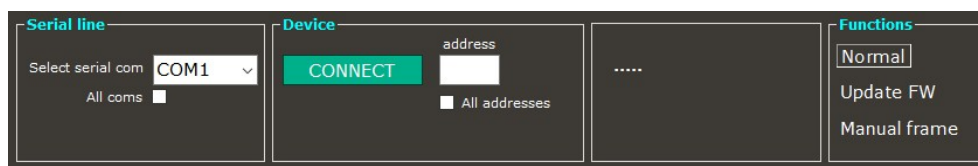
occurs because the Server is not able to handle the request from the Client are shown. For more information on the exception responses and the MODBUS exception codes please refer to the "7.2 Exception response and exception codes" section on page 74.

7. In the **HOLDING REGISTERS** group box on the bottom right of the page the Holding Registers are available; the items in this area allow to read in or write into the working parameters of the connected device.

The main page allows the operator to choose the language used to display texts and items in the user interface. Click the **Italian flag**  icon at the top right of the page to choose the Italian language; click the **UK flag**  icon to choose the English language.

On the top right of the page over the **FUNCTIONS** group box the **MESSAGES >** button is available. By pressing the button the main page widens and an additional section appears on the right. The communication frames transmitted from the software tool to the device and vice versa are shown. To close the message window and go back to the standard visualization of the main page press the **< CLOSE** button.

5.3.1 Configuring the serial port – Connection to the encoder



The four group boxes at the top of the main page group the items that are necessary to configure the serial port and connect to the device. In particular they allow (in order):

- to select the serial port of the pc the encoder is connected to (in the **SERIAL LINE** group box);
- to set the node address of the connected device and start the scanning operation for finding the connected device (in the **DEVICE** group box);
- to show some information about the device, once the connection is established;
- to choose the page to display.

When the page of the interface opens, by means of the drop-down box in the **Select serial com** field you can choose the serial port the device is connected to. The port that is currently selected is indicated in the drop-down box. If you do not know the number of the COM port the device is connected to select the **All coms** check box.

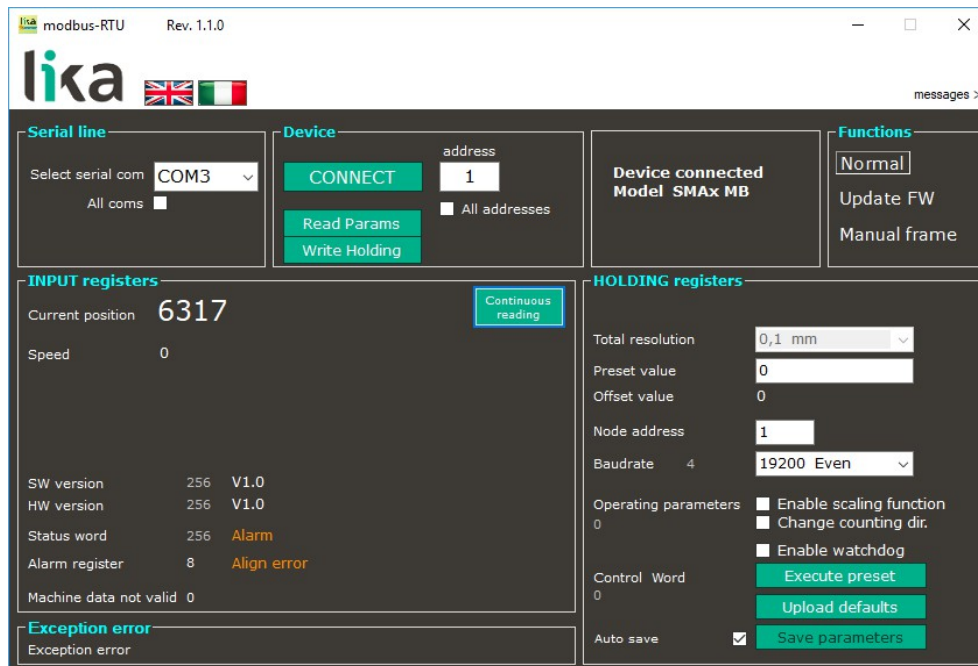
In the **Address** field in the **DEVICE** group box, the MODBUS address of the connected device must be entered. By default the address of all Lika devices is "1". If you do not know the MODBUS address of the networked device select the **All addresses** check box.

If you select the **All coms** and **All addresses** check boxes the tool will scan all the available serial ports (COM1 first, then COM2, COM3, etc., if installed) and will test for each one all the baud rate and parity bit options (9600 No parity, 9600 Even, 9600 Odd, etc.) and all the possible addresses (from 1 to 247, according to the MODBUS protocol).

It is clear that this searching operation may take a while.

To start searching and connect to the device press the **CONNECT** button in the **DEVICE** group box. While the program is attempting to connect to the device, a green progress bar appears over the button to indicate that the searching operation is in progress; a further progress bar and the label **Busy** over the drop-down box of the **Select serial com** field indicate that the port is currently open and being checked. As long as the communication is active the bar indicates that the port is open. When there is no communication the port becomes available for other applications after a timeout of about 1 second.

If problems occur while trying to establish the connection (you have selected a wrong or not available serial port, or the port is currently busy; or you have set a wrong node address, etc.), the interface will go on trying to establish the connection uninterruptedly, until the **CONNECT** button is pressed once again. The message **No device** and the name of the last address and serial port that have been checked will appear under the button.



If the connection succeeds, the **Device connected** message as well as the model of the found device are displayed in the group box between **DEVICE** and **FUNCTIONS** group boxes. The software in fact is able to recognize automatically the model of the connected device and changes the displayed page and the list of the available registers consequently.

The values of the registers that are currently set in the device are shown in the **INPUT REGISTERS** and **HOLDING REGISTERS** group boxes. Furthermore the **Read params** and **Write Holding** buttons appear in the **DEVICE** group box.

Read params

When you press the **Read params** button you send a single command to read the Input Registers and the Holding Registers. The Input Registers and the Holding Registers listed in the page are updated according to the values of the device in the moment when the request is transmitted (instantaneous reading). To read the registers uninterruptedly you must press the **Continuous reading** button, see on page 32.

Write Holding

When you press the **Write Holding** button you send a command to write all Holding Registers at the same time. It is also possible to press the **ENTER** key in the keyboard of your personal computer: it sends a command to write the only register where the cursor is placed. After having set a new value next to any Holding Register, press the **Write Holding** button to transmit to the encoder all Holding Registers data; or just press the **ENTER** key to send only the datum of the Holding Register where the cursor is placed.

Inside the **FUNCTIONS** group box two further buttons are available: **UPDATE FW** and **MANUAL FRAME**. The first button allows the operator to enter the page for the firmware upgrade; the second button allows to enter the page where PDUs can be transmitted manually. For complete information on the firmware upgrade procedure please refer to the "5.4 Update FW page - Firmware upgrade" section on page 39; for complete information on the manual transmission of the PDUs please refer to the "5.5 Manual frame page - Transmitting PDUs manually" section on page 42. By pressing the **NORMAL** button you move back to the main page.

5.3.2 Reading the Input Registers



In the largest group box on the left of the programming interface the Input Registers are available, they provide result values and status / alarm information

on the device. These items are described in the "7.1.2 Input Register parameters" section on page 69 of this manual.

In this group box the items listed hereafter are available.

Continuous reading

When you press the **Continuous reading** button you enable the transmission of continuous commands to read the Input Registers uninterruptedly. After pressing the button, its background is coloured orange and a green progress bar appears beneath the button to indicate that a reading operation is being executing. A further green progress bar and the label **Busy** over the drop-down box of the **Select serial com** field indicate that the serial port is open. The items in the **FUNCTIONS** and **HOLDING REGISTERS** group boxes are made unavailable. The values of the Input Registers listed in the page are updated without cease. To stop the continuous reading press the **Continuous reading** button once again. To send a single command to read the registers (instantaneous reading) press the **Read params** button in the **DEVICE** group box, see on page 30.

Current position

It shows the current position of the device expressed in counts. The value does not appear if an error is active in the device. See the **Current position [0001 hex]** register on page 70.

Speed

This register is not used currently and reserved for future use. See the **Current velocity [0002 hex]** register on page 70.

SW version

It shows the version of the software that is installed currently (both as a decimal value -256 in the Figure- and as a string -V1.0 in the Figure). See the **SW Version [0004 hex]** register on page 71.

HW version

It shows the version of the hardware (PCB version) that is installed currently (both as a decimal value -256 in the Figure- and as a string -V1.0 in the Figure). See the **HW Version [0005 hex]** register on page 72.

Status word

It shows the value expressed in decimal notation (256 in the Figure) that can be read currently next to the **Status word [0006 hex]** register, refer to page 72. If there are active alarms, the **Alarm** message appears on the right while the position value above disappears. A further message shows whether the scaling function and the counting direction function are enabled.

Alarm register

It shows the value expressed in decimal notation (8 in the Figure) that can be read currently next to the **Alarms register [0000 hex]** register, refer to page 69. If there are active alarms, the specific alarm message appears on the right (e.g. **Align error** in the Figure) while the position value above disappears.

Machine data not valid

It shows the value expressed in decimal notation (0 in the Figure) that can be read currently next to the **Wrong parameters list [0003 hex]** register, refer to page 71. If a wrong parameter has been set, it is stated on the right while the position value above disappears. **Status word** and **Alarm register** registers activate too. The input field of the wrong parameter in the **HOLDING REGISTERS** group box of the interface is highlighted in red.

5.3.3 Reading the exception responses – Exception error

In the **EXCEPTION ERROR** group box just beneath the **INPUT REGISTERS** group box the exception response messages that the Server transmits to the Client when an error occurs are shown.

Exception error

It shows the exception response messages that the Server transmits to the Client when an error occurs because the Server is not able to handle the request from the Client (for example, if you confirm values that are not allowed or because of a request to read a non-existent output or register). For more information on the exception responses and the MODBUS exception codes please refer to the "7.2 Exception response and exception codes" section on page 74.

5.3.4 Reading / writing the Holding Registers

HOLDING registers

| | | |
|----------------------|-------------------------------------|--|
| Total resolution | | 0,1 mm ▼ |
| Preset value | | 0 |
| Offset value | | 0 |
| Node address | | 1 |
| Baudrate | 4 | 19200 Even ▼ |
| Operating parameters | | <input type="checkbox"/> Enable scaling function <input type="checkbox"/> Change counting dir. <input type="checkbox"/> Enable watchdog |
| Control Word | | <div style="background-color: #008080; color: white; padding: 5px; margin-bottom: 5px;">Execute preset</div> <div style="background-color: #008080; color: white; padding: 5px; margin-bottom: 5px;">Upload defaults</div> <div style="background-color: #008080; color: white; padding: 5px;">Save parameters</div> |
| Auto save | <input checked="" type="checkbox"/> | |

In the largest group box on the right of the programming interface, the Holding Registers are available. The items in this group box allow to read in or write into the working parameters of the device, by pressing the **Read params** and **Write Holding** buttons respectively, they are available in the **DEVICE** group box. The Holding Registers are fully described in the "7.1.1 Machine data parameters (Holding registers)" section on page 59 in this manual.



WARNING

If the **Auto save** check box at the bottom of the page is selected (see on page 37), all parameters of the Holding registers are stored automatically and instantaneously as soon as they are set: check box settings are saved as soon as the check box is selected / deselected; the write registers are saved as soon as you press the **ENTER** key in the keyboard or place the cursor anywhere outside the field after setting the value.

If the **Auto save** check box is not selected instead, you must press the **Save parameters** button to store the parameters permanently on the EEPROM after setting (see on page 38).

In this section the items listed hereafter are available.

Total resolution

It allows both to set a custom resolution of the encoder (measuring step tailored for the specific application) and to show the value that is set currently. You can select one of the resolution options available in the drop-down menu or set a custom value. The custom value must be expressed in hundredths of a millimetre. See the **Resolution [0000 hex]** register on page 59. This register can be modified only if the **Enable scaling function** option next to the **Operating parameters** item is enabled (=1).

Preset value

It allows both to set the preset value and to show the value that is set currently. To execute the preset operation you must then press the **Execute preset** button next to the **Control word** item at the bottom of the page: it executes the whole sequence of preset commands (activation of the **Perform counting preset** bit and registers setting; deactivation of the **Perform counting preset** bit and registers setting; save of parameters). Refer also to page 37. For more information refer to the **Preset value [0001 hex]** register on page 61.

Offset value

It shows the offset value which results from the setting of the **Preset value**. For more information refer to the **Offset value [0002 hex]** register on page 63.

Node address

It allows both to set the node address of the device and to show the value that is set currently. For more information refer to the **Node address [0004 hex]** register on page 65.

Baud rate

It allows both to set the baud rate of the encoder's serial communication and to show the value that is set currently. The current setting is shown in the drop-down box. The decimal value that results from the current setting (4 in the

Figure above) is also shown. For more information refer to the [Serial com baud rate \[0005 hex\]](#) register on page 65.

Operating parameters

It groups the functions available in the [Operating parameters \[0003 hex\]](#) register (see on page 63) and shows their current enabling 1/disabling 0 state. The decimal value which results from the binary sequence of the sixteen bits in the register (consider that 0 = DISABLED, 1 = ENABLED) will appear in the field under the label (0 in the Figure above).

Enable scaling function

It allows both to enable 1/disable 0 the scaling function and to show its current enabling/disabling state. Select/deselect the check box to enable 1/disable 0 the function, the decimal value which results from the binary sequence of the sixteen bits in the [Operating parameters \[0003 hex\]](#) register will appear in the field on the left side. When the function is enabled, the text of the label is coloured yellow. For more information refer to the [Scaling function](#) parameter on page 64.

Change counting dir.

It allows both to change the counting direction function and to show its current setting. Select/deselect the check box to switch between the options, the decimal value which results from the binary sequence of the sixteen bits in the [Operating parameters \[0003 hex\]](#) register will appear in the field on the left side. When the function is enabled, the text of the label is coloured yellow. For more information refer to the [Code sequence](#) parameter on page 64.

Control word

It groups the functions available in the [Control Word \[000A hex\]](#) register (see on page 66) and to show their current enabling-activation 1/disabling-deactivation 0 state. Use the check boxes / buttons on the right side to set the functions, the decimal value which results from the binary sequence of the sixteen bits in the register will appear in the field under the label (0 in the Figure above). For more information refer to the [Control Word \[000A hex\]](#) register on page 66.

Enable watchdog

It allows both to enable 1/disable 0 the watchdog function provided by the MODBUS protocol and to show the enabling/disabling state. Select/deselect the check box to enable 1/disable 0 the function, the decimal value which results from the binary sequence of the sixteen bits in the [Control Word \[000A hex\]](#) register will appear in the field on the left side. When the function is enabled,

the text of the label is coloured yellow. For more information refer to the **Watchdog enable** parameter on page 66.

Execute preset

This button allows to activate the preset function in order to set the output value to the value entered next to the **Preset value** parameter (see on page 35). The **Execute preset** button executes the whole sequence of preset setting commands: activation of the **Perform counting preset** bit and registers setting; deactivation of the **Perform counting preset** bit and registers setting; save of parameters. While the commands are being executing, the background of the button is coloured orange and the decimal value which results from the binary sequence of the sixteen bits in the **Control Word [000A hex]** register is updated in real time in the field under the label. As soon as the operation is carried out, the value in the **Current position** field is the same as the value entered in the **Preset value** parameter (you are not required to press the **Read params** button in order to refresh the current position value). For more information refer to the **Perform counting preset** parameter on page 67.

Upload defaults

This button allows to activate the function meant to upload the default parameters. It executes the whole sequence of default parameters upload commands: activation of the **Load default parameters** bit and registers setting; deactivation of the **Load default parameters** bit and registers setting; save of parameters). While the commands are being executing, the background of the button is coloured orange and the decimal value which results from the binary sequence of the sixteen bits in the **Control Word [000A hex]** register is updated in real time in the field under the label. As soon as the operation is carried out, the value in the parameter is updated automatically (you are not required to press the **Read params** button in order to refresh the values that are currently set). For more information refer to the **Load default parameters** parameter on page 67.



WARNING

The execution of this command causes all parameters which have been previously set to be overwritten!

Auto save

It enables / disables the autosave function.

If the **Auto save** check box is selected, all parameters of the Holding registers are stored automatically and instantaneously as soon as they are set: check box settings are saved as soon as the check box is selected / deselected; the write

registers are saved as soon as you press the **ENTER** key in the keyboard or place the cursor anywhere outside the field after setting the value.

If the **Auto save** check box is not selected instead, you must press the **Save parameters** button to store the parameters permanently on the EEPROM after setting.

Save parameters

This button is active only if the **Auto save** check box is not selected. It allows to activate the function meant to store the parameters permanently on the EEPROM. It executes the whole sequence of data store commands in order to store the parameters permanently on the EEPROM: activation of the **Save parameters** bit and registers setting; deactivation of the **Save parameters** bit and registers setting). While the commands are being executing, the background of the button is coloured orange and the decimal value which results from the binary sequence of the sixteen bits in the **Control Word [000A hex]** register is updated in real time in the field under the label. For more information refer to the **Save parameters** parameter on page 67.

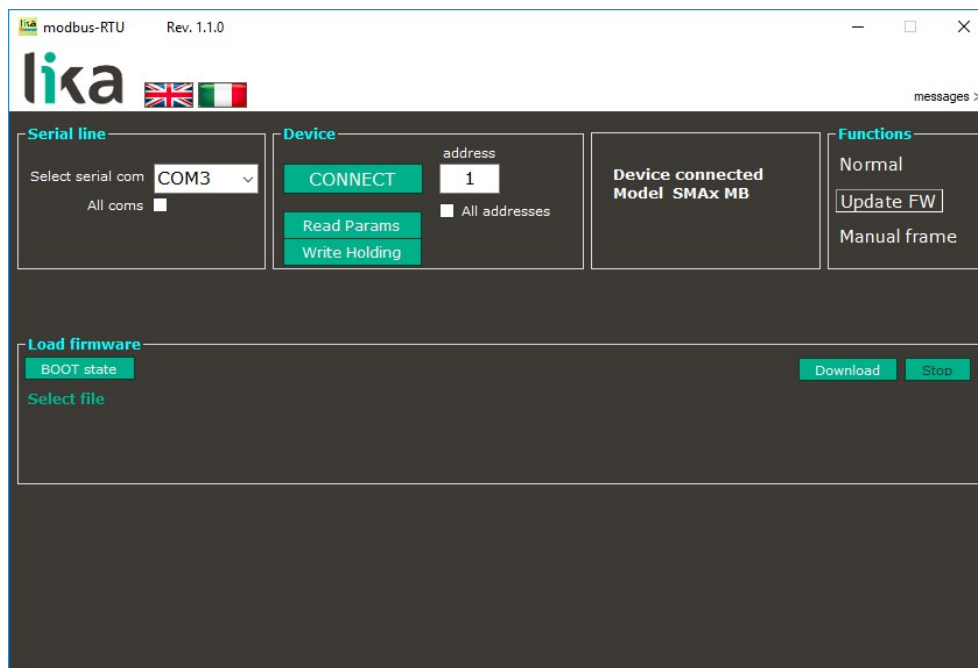


WARNING

Select the **Auto save** check box to activate the autosave function. If the check box is selected, all parameters of the Holding registers are stored automatically and instantaneously as soon as they are set and you are never required to press the **Save parameters** button: check box settings are saved as soon as the check box is selected / deselected; the write registers are saved as soon as you press the **ENTER** key in the keyboard or place the cursor anywhere outside the field after setting the value. For more information please refer to the item in the previous page.

5.4 Update FW page - Firmware upgrade

When you press the **UPDATE FW** button in the **FUNCTIONS** group box on the top right of the main page of the interface, you enter the page that allows to upgrade the firmware of the device.



WARNING

The firmware upgrading operation must be accomplished by skilled and competent personnel. If a wrong or incompatible firmware program is installed, then the unit may not be updated correctly, in some cases preventing the unit from working. It is mandatory to perform the upgrade according to the instructions provided in this section.

Before installation always ascertain that the firmware program is compatible with the hardware and software of the device. Furthermore never turn the power off during flash upgrade.

5.4.1 Information on firmware upgrade

This operation allows to upgrade the unit firmware by downloading upgrading data to the flash memory.

Firmware is a software program which controls the function and operation of a device; the firmware program, sometimes referred to as "user program", is stored in the flash memory integrated inside the unit. These encoders are designed so that the firmware can be easily updated by the user himself. This allows Lika

Electronic to make new improved firmware programs available during the lifetime of the product.

Typical reasons for the release of new firmware programs are the necessity to make corrections, improve and even add new functions to the device.

The firmware upgrading program consists of a single file having .BIN extension to be downloaded to the unit using the tools available in this page. Files are released by Lika Electronic Technical Assistance & After Sale Service.

If the latest firmware version is already installed in the unit, you do not need to proceed with any new firmware installation. The current firmware version can be checked in the **SW version** item of the interface (see on page 32) or in the **SW Version [0004 hex]** register after having connected properly to the unit (see the page 71).

**NOTE**

If you are not confident that you can perform the update successfully please contact Lika Electronic Technical Assistance & After Sale Service.

5.4.2 Preliminary operations and connections

Before proceeding with the firmware upgrade please ascertain that the following requirements are fully met:

- the encoder is properly connected to a PC through an RS-485 serial COM port;
- **Modbus-RTU.exe** interface is open and the connection is active;
- you have the .BIN file to hand for firmware upgrade.

5.4.3 Launching the firmware upgrade process

To upgrade the firmware program please proceed as follows:

1. open the **Load firmware** page by pressing the **UPDATE FW** button in the **FUNCTIONS** group box;
2. press the **SELECT FILE** button; once you press the button, the **Open** dialogue box appears on the screen: open the folder where the firmware upgrading .BIN file released by Lika Electronic is located, select the file, and confirm by pressing the **OPEN** button;

3. press the **BOOT STATE** button; if the encoder is connected properly and the system is able to enter the boot mode successfully, the **BOOT OK** message appears on the right side of the button and the LED fitted in the encoder's enclosure starts blinking at 2 Hz with duty cycle = 50%; the encoder is now ready and waits for the firmware upgrade operation to start;



WARNING

As long as the encoder is in the boot mode, you can restore the normal communication mode simply by switching the device off and then on again. On the contrary, once you start the download process, any event that may happen such as the push of the **STOP** button or an unexpected occurrence will require that you switch the device off and then on again and also that you restart the firmware download operation once more.



WARNING

Before installation always ascertain that the firmware program is compatible with the hardware and software of the device. Never turn the power off during flash upgrade.

4. Press the **DOWNLOAD** button to start the firmware upgrading process; the LED fitted in the encoder's enclosure starts blinking at 5 Hz with duty cycle = 50% and the **DOWNLOADING** message appears on the screen;
5. as soon as the operation is carried out successfully, the **DOWNLOADED** message appears on the screen;
6. turn the encoder power off and then on again to complete the operation.

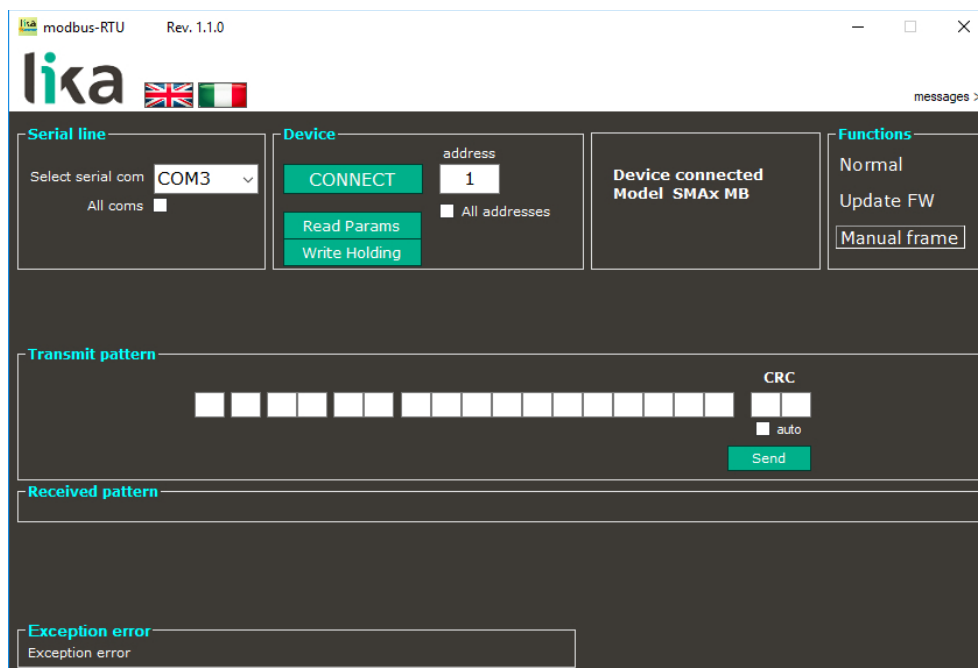


NOTE

While downloading the firmware upgrading program, you may be required to press the **STOP** button; or unexpected conditions may arise which could lead to a failure of the installation process. When such a matter occurs, the download process cannot be carried out successfully and thus the operation is aborted; the LED fitted in the encoder's enclosure starts blinking at 10 Hz with duty cycle = 50%; when it happens, you must turn the power off and then on again to reset the device and restart the operation.

5.5 Manual frame page – Transmitting PDUs manually

When you press the **MANUAL FRAME** button in the **FUNCTIONS** group box on the top right of the main page of the interface, you enter the page that allows you to enter and transmit Request PDUs manually. In the "Programming examples" section on page 77 you can find some examples of Request PDU messages and the relevant Response PDU messages.



If you need to enter and transmit a Request PDU manually proceed as follows:

1. enter the PDU message expressed in hexadecimal notation in the fields under the **Transmit pattern** item; you can use the TAB key in the keyboard to move through the fields under **Transmit pattern**;
2. enter the Cyclical Redundancy Check (CRC) value in the last two fields on the right side;
3. if you select the **CRC auto** check box you will not be required to enter the CRC value manually: CRC will be calculated automatically by the program when the message is transmitted;
4. press the **SEND** button to transmit the Request PDU message.

In the field under the **Received pattern** item the Response PDU transmitted back by the Server will appear in hexadecimal format.

If the system is unable to receive the Response PDU, the **No pattern** error message appears next to the **Received pattern** field.

When an error occurs because the Server is not able to handle the request from the Client (for example, if you confirm values that are not allowed or because of a request to read a non-existent output or register), the exception response messages that the Server transmits to the Client will be displayed next to the **Exception error** field at the bottom of the page. For more information on the exception responses and the MODBUS exception codes please refer to the "7.2 Exception response and exception codes" section on page 74.

6 Modbus® interface

Lika SMAZ Modbus series linear encoders are Slave devices and implement the Modbus application protocol (level 7 of OSI model) and the "Modbus over Serial Line" protocol (levels 1 & 2 of OSI model).

For any further information or omitted specifications please refer to "Modbus Application Protocol Specification V1.1b" and "Modbus over Serial Line. Specification and Implementation Guide V1.02" available at www.modbus.org.

6.1 Modbus Master / Slaves protocol principle

The Modbus Serial Line protocol is a Master – Slaves protocol. One only Master (at the same time) is connected to the bus and one or several (247 maximum number) Slave nodes are also connected to the same serial bus. A Modbus communication is always initiated by the Master. The Slave nodes will never transmit data without receiving a request from the Master node. The Slave nodes will never communicate with each other. The Master node initiates only one Modbus transaction at the same time.

The Master node issues a Modbus request to the Slave nodes in two modes:

- **UNICAST mode:** in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must have a unique address (from 1 to 247) so that it can be addressed independently from other nodes. Lika devices only implement commands in "unicast" mode.
- **BROADCAST mode:** in that mode the Master can send a request to all Slaves at the same time. No response is returned to "broadcast" requests sent by the Master. The "broadcast" requests are necessarily writing commands. The address 0 is reserved to identify a "broadcast" exchange.

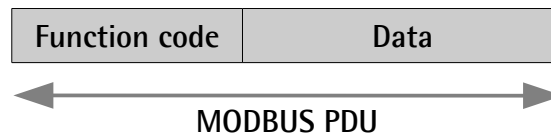


NOTE

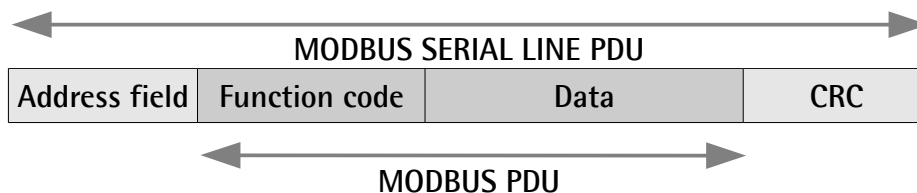
Lika devices do not implement commands in "broadcast" mode.

6.2 Modbus frame description

The Modbus application protocol defines a simple Protocol Data Unit (PDU) independent of the underlying communication layers:



The mapping of Modbus protocol on a specific bus or network introduces some additional fields on the Protocol Data Unit. The client that initiates a Modbus transaction builds the Modbus PDU, and then adds fields in order to build the appropriate communication PDU.



- **ADDRESS FIELD:** on Modbus Serial Line the address field only contains the Slave address. The valid Slave node addresses are in the range of 0 – 247 decimal (see the [Node address \[0004 hex\]](#) register on page 65). The individual Slave devices are assigned addresses in the range of 1 – 247. A Master addresses a Slave by placing the Slave address in the **ADDRESS FIELD** of the message. When the Slave returns its response, it places its own address in the response **ADDRESS FIELD** to let the Master know which Slave is responding.
- **FUNCTION CODE:** the function code indicates to the Server what kind of action to perform. The function code can be followed by a **DATA** field that contains request and response parameters. For any further information on the implemented function codes refer to the "6.4 Function codes" section on page 49.
- **DATA:** the **DATA** field of messages contains the bytes for additional information and transmission specifications that the server uses to take the action defined by the **FUNCTION CODE**. This can include items such as discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field. The structure of the **DATA** field depends on each **FUNCTION CODE** (refer to the "6.4 Function codes" section on page 49).
- **CRC (Cyclic Redundancy Check):** error checking field is the result of a "Redundancy Checking" calculation that is performed on the message contents. This is intended to check whether transmission has been performed properly. The CRC field is two bytes, containing 16-bit binary

value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The device that receives recalculates a CRC during receipt of the message and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The Modbus protocol defines three PDUs. They are:

- **Modbus Request PDU;**
- **Modbus Response PDU;**
- **Modbus Exception Response PDU.**

The **Modbus Request PDU** is defined as {function_code, request_data}, where:
function_code = Modbus function code [1 byte];
request_data = this field is function code dependent and usually contains information such as variable references, variable counts, data offsets, sub-function, etc. [n bytes].

The **Modbus Response PDU** is defined as {function_code, response_data}, where:
function_code = Modbus function code [1 byte];
response_data = this field is function code dependent and usually contains information such as variable references, variable counts, data offsets, sub-function, etc. [n bytes].

The **Modbus Exception Response PDU** is defined as {exception-function_code, exception_code}, where:
exception-function_code = Modbus function code + 80 hex [1 byte];
exception_code = Modbus Exception code, refer to the table "Modbus Exception Codes" in the section 7 of the document "Modbus Application Protocol Specification V1.1b".

6.3 Transmission modes

Two different serial transmission modes are defined in the Modbus serial protocol: the **RTU (Remote Terminal Unit) mode** and the **ASCII mode**. The transmission mode defines the bit contents of message fields transmitted serially on the line. It determines how information is packed into the message fields and decoded. The transmission mode and the serial port parameters must be the same for all devices on a Modbus Serial Line. All devices must implement the RTU mode, while the ASCII mode is an option. Lika devices only implement RTU transmission mode, as described in the following section.

6.3.1 RTU transmission mode

When devices communicate on a Modbus serial line using the RTU (Remote Terminal Unit) mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. Each message must be transmitted in a continuous stream of characters. Synchronization between the messages exchanged by the transmitting device and the receiving device is achieved by placing an interval of at least 3.5 character times between successive messages, this is called "silent interval". In this way a Modbus message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows devices that receive a new frame to begin at the start of the message and to know when the message is completed. So when the receiving device does not receive a message for an interval of 4 character times, it considers the previous message as completed and the next byte will be the first of a new message, i.e. an address.

When baud rate = 9600 bit/s the "silent interval" is 4 ms.

When baud rate = 19200 bit/s the "silent interval" is 2 ms.

When baud rate = 115200 bit/s the "silent interval" is 3.5 ms.

The format (11 bits) for each byte in RTU mode is as follows:

Coding system: 8-bit binary

Bits per Byte: 1 start bit;

8 data bits, least significant bit (lsb) sent first;

1 bit for parity completion (= Even);

1 stop bit.

Modbus protocol uses a "big-Endian" representation for addresses and data items. This means that when a numerical quantity greater than a single byte is transmitted, the most significant byte (MSB) is sent first.

Each character or byte is sent in this order (left to right):

lsb (Least Significant Bit) ... msb (Most Significant Bit)

| | | | | | | | | | | |
|-------|---|---|---|---|---|---|---|---|---------|------|
| Start | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Parity* | Stop |
|-------|---|---|---|---|---|---|---|---|---------|------|

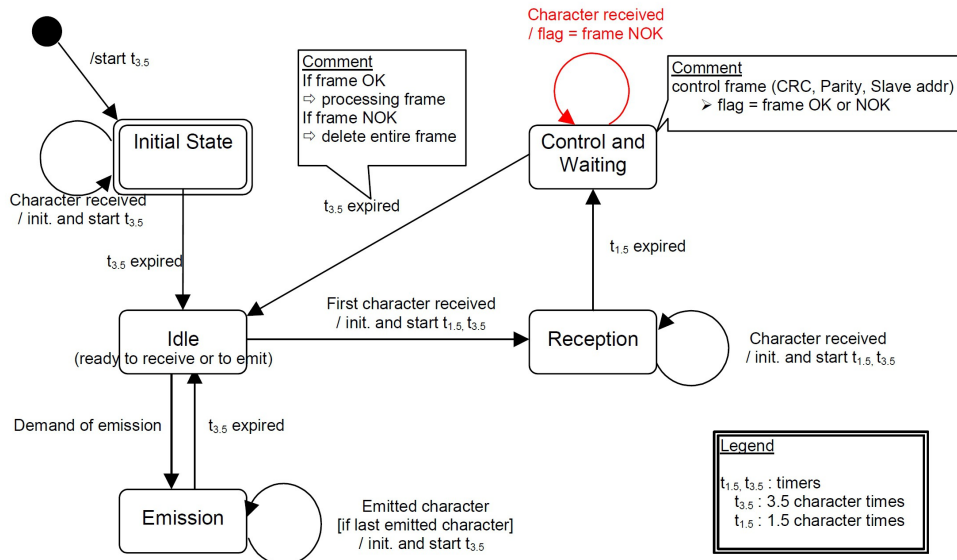
* When "No parity" is activated, the parity bit is replaced by a stop bit.

The default parity mode must be Even parity.

The maximum size of the Modbus RTU frame is 256 bytes, its structure is as follows:

| Slave Address | Function code | Data | CRC | |
|---------------|---------------|---------------------|---------|--------|
| 1 byte | 1 byte | 0 up to 252 byte(s) | CRC Low | CRC Hi |

The following drawing provides a description of the RTU transmission mode state diagram. Both "Master" and "Slave" points of view are expressed in the same drawing.



- Transition from **Initial State** to **Idle** state needs an interval of at least 3.5 character times (time-out expiration = $t_{3.5}$).
- **Idle** state is the normal state when neither emission nor reception is active. In RTU mode, the communication link is declared in **Idle** state when there is no transmission activity after a time interval equal to at least 3.5 characters ($t_{3.5}$).
- A request can only be sent in **Idle** state. After sending a request, the Master leaves the **Idle** state and cannot send a second request at the same time.
- When the link is in **Idle** state, each transmitted character detected on the link is identified as the start of the frame. The link goes to **Active** state. Then the end of the frame is identified when no more character is transmitted on the link after the time interval of at least $t_{3.5}$.
- After detection of the end of frame, the CRC calculation and checking is completed. Afterwards the address field is analysed to determine if the frame is addressed to the device. If not, the frame is discarded. In order to reduce the reception processing time the address field can be analysed as soon as it is received without waiting the end of frame. In this case the CRC will be calculated and checked only if the frame is actually addressed to the Slave.

6.4 Function codes

As previously stated, the function code indicates to the Server what kind of action to perform. The function code field of a Modbus data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (the range 128 ... 255 is reserved and used for Exception Responses). When a message is sent from a Client to a Server device the function code field tells the Server what kind of action to perform. Function code "0" is not valid.

There are three categories of Modbus function codes, they are: **public function codes**, **user-defined function codes** and **reserved function codes**.

Public function codes are in the range 1 ... 64, 73 ... 99 and 111 ... 127; they are well defined function codes, validated by the MODBUS-IDA.org community and publicly documented; furthermore they are guaranteed to be unique. Ranges of function codes from 65 to 72 and from 100 to 110 are **user-defined function codes**: user can select and implement a function code that is not supported by the specification, it is clear that there is no guarantee that the use of the selected function code will be unique. **Reserved function codes** are not available for public use.

6.4.1 Implemented function codes

Lika SMAZ Modbus series linear encoders only implement public function codes, they are described hereafter.

03 Read Holding Registers

FC = 03 (03 hex) r0

This function code is used to READ the contents of a contiguous block of holding registers in a remote device; in other words, it allows to read the values set in a group of work parameters placed in order. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore registers numbered 1-16 are addressed as 0-15.

The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).

For the complete list of holding registers accessible using **03 Read Holding Registers** function code please refer to the "7.1.1 Machine data parameters (Holding registers)" section on page 59.

Request PDU

| | | |
|-----------------------|---------|----------------------|
| Function code | 1 byte | 03 hex |
| Starting address | 2 bytes | 0000 hex to FFFF hex |
| Quantity of registers | 2 bytes | 1 to 125 (007D hex) |

Response PDU

| | | |
|----------------|---------------------|---------------|
| Function code | 1 byte | 03 hex |
| Byte count | 1 byte | 2 x N* |
| Register value | N* x 2 bytes | |

*N = Quantity of registers

Exception Response PDU

| | | |
|----------------|--------|----------------------------------|
| Error code | 1 byte | 83 hex (=03 hex + 80 hex) |
| Exception code | 1 byte | 01 or 02 or 03 or 04 |



Here is an example of a request to read the **Preset value [0001 hex]** parameter (register 2).

| Request | | Response | |
|---------------------|-----------|---------------------|-----------|
| Field name | (Hex) | Field name | (Hex) |
| Function | 03 | Function | 03 |
| Starting address Hi | 00 | Byte count | 02 |
| Starting address Lo | 01 | Register 2 value Hi | 05 |
| No. of registers Hi | 00 | Register 2 value Lo | DC |
| No. of registers Lo | 01 | | |

As you can see in the table, **Preset value [0001 hex]** parameter (register 2) contains the value 05 DC hex, i.e. 1500 in decimal notation.

The full frame needed for the request to read the **Preset value [0001 hex]** parameter (register 2) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)

[01][03][00][01][00][01][D5][CA]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

[00][01] = starting address (**Preset value [0001 hex]** parameter, register 2)

[00][01] = number of requested registers

[D5][CA] = CRC

The full frame needed to send back the value of the **Preset value [0001 hex]** parameter (register 2) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)

[01][03][02][05][DC][BA][8D]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

[02] = number of bytes (2 bytes for each register)

[05][DC] = value of register 2, 05 DC hex = 1500 dec

[BA][8D] = CRC

04 Read Input Register

FC = 04 (04 hex)

This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some results values and state / alarm messages in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15. The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).

For the complete list of input registers accessible using **04 Read Input Register** function code please refer to the "7.1.2 Input Register parameters" section on page 69.

Request PDU

| | | |
|-----------------------------|---------|----------------------|
| Function code | 1 byte | 04 hex |
| Starting address | 2 bytes | 0000 hex to FFFF hex |
| Quantity of Input Registers | 2 bytes | 0000 hex to 007D hex |

Response PDU

| | | |
|----------------------|--------------|---------------|
| Function code | 1 byte | 04 hex |
| Byte count | 1 byte | 2 x N* |
| Input register value | N* x 2 bytes | |

*N = Quantity of registers

Exception Response PDU

| | | |
|----------------|--------|----------------------------------|
| Error code | 1 byte | 84 hex (=04 hex + 80 hex) |
| Exception code | 1 byte | 01 or 02 or 03 or 04 |



Here is an example of a request to read the **Current position [0001 hex]** parameter (input register 2).

| Request | | Response | |
|---------------------------|-----------|---------------------|-----------|
| Field name | (Hex) | Field name | (Hex) |
| Function | 04 | Function | 04 |
| Starting address Hi | 00 | Byte count | 02 |
| Starting address Lo | 01 | Register 2 value Hi | 13 |
| Quantity of Input Reg. Hi | 00 | Register 2 value Lo | C5 |
| Quantity of Input Reg. Lo | 01 | | |

As you can see in the table, **Current position [0001 hex]** parameter (input register 2) contains the value 13 C5 hex, i.e. 5061 in decimal notation.

The full frame needed for the request to read the **Current position [0001 hex]** parameter (input register 2) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)

[01][04][00][01][00][01][60][0A]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[00][01] = starting address (**Current position [0001 hex]** parameter, register 2)

[00][01] = number of requested registers

[60][0A] = CRC

The full frame needed to send back the value of the **Current position [0001 hex]** parameter (register 2) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)

[01][04][02][13][C5][74][53]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[02] = number of bytes (2 bytes for each register)

[13][C5] = value of register 2 **Current position [0001 hex]**, 13 C5 hex = 5061 dec

[74][53] = CRC

06 Write Single Register

FC = 06 (06 hex)

This function code is used to WRITE a single holding register in a remote device. The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0.

The normal response is an echo of the request, returned after the register contents have been written.

For the complete list of registers accessible using **06 Write Single Register** function code please refer to the "7.1.1 Machine data parameters (Holding registers)" section on page 59.

Request PDU

| | | |
|------------------|---------|----------------------|
| Function code | 1 byte | 06 hex |
| Register address | 2 bytes | 0000 hex to FFFF hex |
| Register value | 2 bytes | 0000 hex to FFFF hex |

Response PDU

| | | |
|------------------|---------|----------------------|
| Function code | 1 byte | 06 hex |
| Register address | 2 bytes | 0000 hex to FFFF hex |
| Register value | 2 bytes | 0000 hex to FFFF hex |

Exception Response PDU

| | | |
|----------------|--------|----------------------------------|
| Error code | 1 byte | 86 hex (=06 hex + 80 hex) |
| Exception code | 1 byte | 01 or 02 or 03 or 04 |



Here is an example of a request to write in the **Operating parameters [0003 hex]** item (register 4): we need to set the scaling function (**Scaling function = 1**) and the increasing counting when the sensor moves in the direction shown by the arrow in Figure 1 (**Code sequence = 0**).

| Request | | Response | |
|---------------------|-----------|---------------------|-----------|
| Field name | (Hex) | Field name | (Hex) |
| Function | 06 | Function | 06 |
| Register address Hi | 00 | Register address Hi | 00 |
| Register address Lo | 03 | Register address Lo | 03 |
| Register value Hi | 00 | Register value Hi | 00 |
| Register value Lo | 01 | Register value Lo | 01 |

As you can see in the table, the value 00 01 hex, i.e. 0000 0000 0000 0001 in binary notation, is set in the **Operating parameters [0003 hex]** item (register

4): bit 0 **Scaling function** = 1; bit 1 **Code sequence** = 0; the remaining bits are not used, therefore their value is 0.

The full frame needed for the request to write the value 00 01 hex in the **Operating parameters [0003 hex]** item (register 4) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)

[01][06][00][03][00][01][B8][0A]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][03] = address of the register (**Operating parameters [0003 hex]** item, register 4)

[00][01] = value to be set in the register

[B8][0A] = CRC

The full frame needed to send back a response following the request to write in the **Operating parameters [0003 hex]** item (register 4) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)

[01][06][00][03][00][01][B8][0A]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][03] = address of the register (**Operating parameters [0003 hex]** item, register 4)

[00][01] = value set in the register

[B8][0A] = CRC

16 Write Multiple Registers

FC = 16 (10 hex)

This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.

The values to be written are specified in the request data field. Data is packed as two bytes per register.

The normal response returns the function code, starting address and quantity of written registers.

For the complete list of registers accessible using **16 Write Multiple Registers** function code please refer to the "7.1.1 Machine data parameters (Holding registers)" section on page 59.

Request PDU

| | | |
|-----------------------|---------------------|----------------------|
| Function code | 1 byte | 10 hex |
| Starting address | 2 bytes | 0000 hex to FFFF hex |
| Quantity of registers | 2 bytes | 0001 hex to 007B hex |
| Byte count | 1 byte | 2 x N* |
| Registers value | N* x 2 bytes | value |

*N = Quantity of registers

Response PDU

| | | |
|-----------------------|---------|----------------------|
| Function code | 1 byte | 10 hex |
| Starting address | 2 bytes | 0000 hex to FFFF hex |
| Quantity of registers | 2 bytes | 1 to 123 (007B hex) |

Exception Response PDU

| | | |
|----------------|--------|-----------------------------------|
| Error code | 1 byte | 90 hex (= 10 hex + 80 hex) |
| Exception code | 1 byte | 01 or 02 or 03 or 04 |



Here is an example of a request to write the value 00 0A hex (=10) next to the **Node address [0004 hex]** parameter (register 5) and the value 00 01 hex (= 1 = baud rate 9600 bit/s, parity bit Even) next to the **Serial com baud rate [0005 hex]** parameter (register 6).

| Request | | Response | |
|------------|-----------|------------|-----------|
| Field name | (Hex) | Field name | (Hex) |
| Function | 10 | Function | 10 |

| | | | |
|--------------------------|----|--------------------------|----|
| Starting address Hi | 00 | Starting address Hi | 00 |
| Starting address Lo | 04 | Starting address Lo | 04 |
| Quantity of registers Hi | 00 | Quantity of registers Hi | 00 |
| Quantity of registers Lo | 02 | Quantity of registers Lo | 02 |
| Byte count | 04 | | |
| Register 1 value Hi | 00 | | |
| Register 1 value Lo | 0A | | |
| Register 2 value Hi | 00 | | |
| Register 2 value Lo | 01 | | |

As you can see in the table, the value 00 0A hex, i.e. 10 in decimal notation, is set in the register 5 **Node address [0004 hex]** parameter; while the value 00 01 hex, i.e. 1 in decimal notation = baud rate 9600 bit/s, parity bit Even, is set in the register 6 **Serial com baud rate [0005 hex]** parameter. Thus the encoder will be programmed to have node address 10 and data transmission rate = 1 = baud rate 9600 bit/s, parity bit Even.

The full frame needed for the request to write the value 10 dec next to the **Node address [0004 hex]** parameter (register 5) and the value 1 dec next to the **Serial com baud rate [0005 hex]** parameter (register 6) to the Slave having currently the node address 1 is as follows:

Request PDU (in hexadecimal format)

[01][10][00][04][00][02][04][00][0A][00][01][13][9E]

where:

[01] = Slave address

[10] = **16 Write Multiple Registers** function code

[00][04] = starting address (**Node address [0004 hex]** parameter, register 5)

[00][02] = number of requested registers

[04] = number of bytes (2 bytes for each register)

[00][0A] = value to be set in the register 5, 00 0A hex = 10 dec

[00][01] = value to be set in the register 6, 00 01 hex = 1 dec

[13][9E] = CRC

The full frame needed to send back a response following the request to write the value 10 next to the **Node address [0004 hex]** parameter (register 5) and the

value 1 next to the **Serial com baud rate [0005 hex]** parameter (register 6) from the Slave having the node address 1 previously and the node address 10 currently is as follows:

Response PDU (in hexadecimal notation)

[0A][10][00][04][00][02][01][72]

where:

[0A] = new Slave address

[10] = **16 Write Multiple Registers** function code

[00][04] = starting address (**Node address [0004 hex]** parameter, register 5)

[00][02] = number of written registers

[01][72] = CRC



WARNING

For safety reasons, when the encoder is on, a continuous data exchange between the Master and the Slave has to be planned in order to be sure that the communication is always active; this is intended to prevent danger situations from arising in case of failures in the communication network.

For this purpose the Watchdog function is implemented and can be activated as optional. Watchdog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running Watchdog safety system immediately takes action and commands an alarm to be triggered. To enable the Watchdog function, set to "1" the **Watchdog enable** bit 0 in the **Control Word [000A hex]** variable. If "0" is set the Watchdog is disabled; if "1" is set the Watchdog is enabled. When the Watchdog function is enabled, if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watchdog** alarm message is invoked to appear as soon as the Modbus network communication is restored).

7 Programming parameters

7.1 Parameters available

Hereafter the parameters available for the Modbus encoders are listed and described as follows:

Parameter name [Register address]

[Register number, data types, attribute]

- The register address is expressed in hexadecimal notation.
- The register number is expressed in decimal notation.
- Attribute:
 - ro = read only access
 - rw = read and write access

7.1.1 Machine data parameters (Holding registers)

Machine data parameters are accessible for both writing and reading; to read the value set in a parameter use the **03 Read Holding Registers** function code (reading of multiple registers); to write a value in a parameter use the **06 Write Single Register** function code (writing of a single register) or the **16 Write Multiple Registers** (writing of multiple registers); for any further information on the implemented function codes refer to the "6.4.1 Implemented function codes" section on page 49.

Resolution [0000 hex]

[Register 1, Unsigned16, rw]

This parameter is used to set a custom resolution (measuring step).

This register can be programmed only if the bit 0 **Scaling function** in the **Operating parameters [0003 hex]** item is set to "=1"; otherwise the system will use the standard resolution "=10" = 0.1 mm (the encoder will provide 12,500 information for the whole travel of the MTAZ-1330 scale, 14 bits).

The resolution can be defined as the smallest change in the underlying quantity that produces a response in the measurement, the response being the information that is provided to output.

By default the resolution is 0.1 mm, thus the encoder provides 12,500 information (14 bits) for the whole travel, the measuring length of the MTAZ-1330 scale being 1,250 mm.

You are allowed to set whatever value between 0.1 mm (10 hundredths of a millimetre) and 1.25 mm (125 hundredths of a millimetre). The entered value has to be expressed in hundredths of a millimetre.

If you set a value greater than the maximum resolution allowed, after sending the Request PDU the **Machine data not valid** error message will be sent back while the relevant bit in the **Wrong parameters list [0003 hex]** item will be set to 1.

Default = 10 (min. = 10, max. = 125)



NOTE

If you have set the preset, when you change the value next to the **Resolution [0000 hex]** parameter, then you must check the value in the **Preset value [0001 hex]** parameter and perform the homing operation (bit 11 **Perform counting preset** in **Control Word [000A hex]** = 1).



EXAMPLE

The main and default features of the SMAZ linear encoder are as follows:

- **Default resolution** = 0.1 mm
- **MTAZ-1330 max. measuring length** = 1,250 mm
- **Max. number of information** = 12,500 (14 bits)

As stated, the number of information provided to output is calculated as follows:

$$\text{number of information} = \frac{\text{measuring length}}{\text{resolution}}$$

Thus, in a default configuration the number of information is:

$$\text{number of information} = \frac{\text{measuring length}}{\text{resolution}} = \frac{1,250}{0.1} = 12,500$$

Let's assume that you need 5,000 information to be provided for the max. measuring length. It follows that you need to calculate and then set a custom resolution.

The resolution value results from the following calculation:

$$\text{resolution} = \frac{\text{measuring length}}{\text{number of information}}$$

Thus, in the example the resolution will be:

$$\text{resolution} = \frac{\text{measuring length}}{\text{number of information}} = \frac{1,250}{5,000} = 0.25$$

As the value next to the **Resolution [0000 hex]** parameter has to be expressed in hundredths of a millimetre, then you have to enter the value **25**.

The complete programming sequence will be:

- Enable the **Scaling function: Operating parameters [0003 hex]**, bit 0 = 1
- Set the resolution: **Resolution [0000 hex]** = 25 (0000 0019 hex)
- Save the set parameters (**Save parameters** register; see on page 67)



NOTE

Please note that when the count is decreasing (count down information, see **Code sequence** in the **Operating parameters [0003 hex]** register) and you cross the zero, the value immediately after 0 will be 2^{N-1} , where N is the overall information expressed in bits. In the above example the overall information is 5,000, i.e. 13 bits (13 bits are necessary to represent 5,000 information). $2^{13} = 8,192$, thus the value after 0 will be 8,191.

| | | | | | | | | | | |
|-----|------|------|------|------|---|---|---|---|---|-----|
| ... | 8188 | 8189 | 8190 | 8191 | 0 | 1 | 2 | 3 | 4 | ... |
|-----|------|------|------|------|---|---|---|---|---|-----|

Preset value [0001 hex]

[Register 2, Unsigned16, rw]

This register is intended to set the Preset value. Preset function is meant to assign a desired value to a physical position of the encoder. The chosen physical position will get the value set next to this item and all the previous and following positions will get a value according to it. For instance, this can be

useful for getting the zero point of the encoder and the zero point of the application to match. The preset value will be set for the position of the encoder in the moment when the **Perform counting preset** command (bit 11) in the **Control Word [000A hex]** register is sent.

Default = 0 (min. = 0, max. = 16383)



Example

Let's take a look at the following example to better understand the preset function and the meaning and use of the related registers and commands: **Preset value [0001 hex]**, **Offset value [0002 hex]** and **Perform counting preset**.

The encoder position which is transmitted results from the following calculation:

Transmitted value = **read position** (it does not matter whether the position is physical or scaled) + **Preset value [0001 hex]** - **Offset value [0002 hex]**.

If you never set the **Preset value [0001 hex]** and the homing command has been executed never before (**Perform counting preset** command), the transmitted value and the read position are necessarily the same as **Preset value [0001 hex]** = 0 and **Offset value [0002 hex]** = 0.

When you set the **Preset value [0001 hex]** and then execute the **Perform counting preset** command in the **Control Word [000A hex]**, system saves the current encoder position in the **Offset value [0002 hex]** register. It follows that the transmitted value and the **Preset value [0001 hex]** are the same as **read position** - **Offset value [0002 hex]** = 0; in other words, the value set next to the **Preset value [0001 hex]** item is paired with the current position of the encoder as you wish.

For example, let's assume that the value "50" is set next to the **Preset value [0001 hex]** item and you execute the **Perform counting preset** command when the encoder position is "1000". In other words, you want to receive the value "50" when the encoder reaches the position "1000".

We will obtain the following information sequence:

Transmitted value = **read position** (= "1000") + **Preset value [0001 hex]** (= "50") - **Offset value [0002 hex]** (= "1000") = 50.

The following transmitted value will be:

Transmitted value = **read position** (= "1001") + **Preset value [0001 hex]** (= "50") - **Offset value [0002 hex]** (= "1000") = 51.

And so on.



NOTE

- If the **Scaling function** is disabled (bit 0 in the **Operating parameters [0003 hex]** register = 0), **Preset value [0001 hex]** must be lower than or

equal to the maximum number of information for the default resolution - 1 ($2^{14} - 1 = 16383$).

- If the **Scaling function** is enabled (bit 0 in the **Operating parameters [0003 hex]** register = 1), **Preset value [0001 hex]** must be lower than or equal to the maximum number of information for the custom resolution - 1 (for instance -see the example on page 60: 5000 information, number of bits = 13, the max. value you are allowed to enter for the Preset is: $2^{13} - 1 = 8191$).



WARNING

After having entered a new value in the **Resolution [0000 hex]** register it is compulsory to check the **Preset value [0001 hex]** and then perform a homing operation (bit 11 **Perform counting preset** in **Control Word [000A hex]** = 1).

Offset value [0002 hex]

[Register 3, Unsigned16, ro]

As soon as you send the **Perform counting preset** command (see bit 11 in **Control Word [000A hex]**), the current position of the encoder is saved in this register. The offset value is then used in the preset function in order to calculate the encoder position value to be transmitted. To zero set the value in this register you must upload the factory default values (see bit 10, **Load default parameters** command, in **Control Word [000A hex]** on page 67).

For any further information on the preset function and the meaning and use of the related registers and commands **Preset value [0001 hex]**, **Offset value [0002 hex]** and **Perform counting preset** refer to page 61.

Default = 0 (min. = 0, max. = 16383)

Operating parameters [0003 hex]

[Register 4, Unsigned16, rw]

Byte structure of the **Operating parameters [0003 hex]** register:

| byte | MSB | | | LSB | | |
|------|-----|-----|-----|-----|-----|-----|
| bit | 15 | ... | 8 | 7 | ... | 0 |
| | msb | | lsb | msb | | lsb |

Byte 0

Scaling function

bit 0

This is meant to enable / disable the scaling parameter **Resolution [0000 hex]**. When the scaling function is disabled (bit 0 = 0), the encoder uses its own default resolution; otherwise, when the scaling function is enabled (bit 0 = 1), the encoder uses the resolution set next to the **Resolution [0000 hex]** register. Complete information at the **Resolution [0000 hex]** register on page 59.

The default features of the SMAZ linear encoder are:

- **Default resolution** = 0.1 mm
- **MTAZ-1330 max. measuring length** = 1,250 mm
- **Max. number of information** = 12,500 (14 bits)

To know whether the **Scaling function** is currently enabled, you can read the **Scaling** bit 0 of the **Status word [0006 hex]**, see on page 72.

Code sequence

bit 1

This is intended to set if the count is increasing (count up information) either when the sensor moves in the direction indicated by the arrow in Figure 1 or when the sensor moves in reverse of the standard direction, i.e. in the opposite direction to the one shown by the arrow in Figure 1. Setting 0 (bit 1 = 0) causes the encoder counting to increment when the sensor moves as indicated by the arrow in Figure 1; setting 1 (bit 1 = 1) causes the encoder counting to increment when the sensor moves in reverse of the standard direction, i.e. in the opposite direction to the one shown by the arrow in Figure 1.

To know whether the **Code sequence** is currently enabled, you can read the **Counting direction** bit 1 of the **Status word [0006 hex]**, see on page 72.



NOTE

Please note that when the count is decreasing (count down information) and you cross the zero, the value immediately after 0 will be 2^{N-1} , where N is the overall information expressed in bits. Let's suppose the overall information is 5,000, i.e. 13 bits (13 bits are necessary to represent 5,000 information).

$2^{13} = 8,192$, thus the value after 0 will be 8,191.



bit 2 ... 7 Not used.

Byte 1 Not used.

Node address [0004 hex]

[Register 5, Unsigned16, rw]

This register allows to set the node address of the device.

The default address is 1. The individual Slave devices are assigned addresses in the range 1 – 247.

The address 0 is reserved to identify a "broadcast" exchange (Master sends a request to all Slaves connected to the Modbus network). See the "6.1 Modbus Master / Slaves protocol principle" section on page 44.

The Modbus Master node has no specific address, only the Slave nodes must have an address. Each Slave must have a unique address.

Addresses from 248 to 255 are reserved.

If you set the address 0, device will be set to 1 automatically.

Equally, if you set an address higher than 247, device will be set to 1 automatically.

Default = 1 (min. = 1, max. = 247)

Serial com baud rate [0005 hex]

[Register 6, Unsigned16, rw]

This is meant to set the data transmission rate (baud rate and parity bit) of the serial port. The default value is 04 hex = Baud rate 19200 bit/s, Parity bit Even.

Default = 4 (min. = 0, max. = 8)

| Value | Baud rate | Parity bit |
|-------------------------|--------------------|-------------|
| 00 hex | 9600 bit/s | No parity |
| 01 hex | 9600 bit/s | Even |
| 02 hex | 9600 bit/s | Odd |
| 03 hex | 19200 bit/s | No parity |
| 04 hex (default) | 19200 bit/s | Even |
| 05 hex | 19200 bit/s | Odd |
| 06 hex | 115200 bit/s | No parity |
| 07 hex | 115200 bit/s | Even |
| 08 hex | 115200 bit/s | Odd |

Control Word [000A hex]

[Register 11, Unsigned16, rw]

This variable contains the commands to be sent in real time to the Slave in order to manage it.

Byte structure of the **Control Word [000A hex]** register:

| byte | MSB | | | LSB | | |
|------|-----|-----|-----|-----|-----|-----|
| bit | 15 | ... | 8 | 7 | ... | 0 |
| | msb | | lsb | msb | | lsb |

Byte 0 Not used.

Byte 1

Watchdog enable

bit 8

Setting the **Watchdog enable** bit to "1" causes the Watchdog function to be enabled; setting the **Watchdog enable** bit to "0" causes the Watchdog function to be disabled. When the Watchdog function is enabled, if the device does not receive any message from the Server within 1 second, the system forces an alarm condition (the **Watchdog** alarm is invoked to appear as soon as the Modbus network communication is restored). Watchdog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running Watchdog safety system immediately takes action and commands an alarm to be triggered.

Save parameters

bit 9 Data is saved on non-volatile memory at each rising edge of this bit; in other words, data save is performed each time this bit is switched from logic level low ("0") to logic level high ("1").

Load default parameters

bit 10 Default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) are restored at each rising edge of this bit; in other words, the default parameters uploading operation is performed each time this bit is switched from logic level low ("0") to logic level high ("1"). The complete list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 81.



WARNING

The execution of this command causes all parameters which have been previously set to be overwritten!

Perform counting preset

bit 11 It allows to perform a homing operation of the encoder. As soon as the command is sent, the position value which will be transmitted for the current position of the encoder is the one set next to the **Preset value [0001 hex]** register and all the previous and following positions will get a value according to it. Operation is performed at each rising edge of this bit, i.e. each time this bit is switched from logic level low ("0") to logic level high ("1"). When this command is sent, the current encoder position is temporarily saved in the **Offset value [0002 hex]** register. For any further information on the preset function and the meaning and use of the related registers and commands **Preset value [0001 hex]**, **Offset value [0002 hex]** and **Perform counting preset** refer to page 61.



WARNING

To save the current encoder position in the **Offset value [0002 hex]** register permanently, please execute the **Save parameters** command. Should the power be turned off without saving data, the **Offset value [0002 hex]** will be lost!

bit 12 ... 15 Not used.



NOTE

Save the set values using **Save parameters** function.
Should the power be turned off all data not saved will be lost!

7.1.2 Input Register parameters

Input Register parameters are accessible for reading only; to read the value set in an input register parameter use the **04 Read Input Register** function code (reading of multiple input registers); for any further information on the implemented function codes refer to the "6.4.1 Implemented function codes" section on page 49.

Alarms register [0000 hex]

[Register 1, Unsigned16, ro]

This variable is meant to show the alarms currently active in the device. When an alarm is active, also the LED shows visually the fault condition (see the "4.6 Diagnostic LED (Figure 3)" section on page 22).

Structure of the alarms byte:

| byte | MSB | | | LSB | | |
|------|-----|-----|-----|-----|-----|-----|
| bit | 15 | ... | 8 | 7 | ... | 0 |
| | msb | | lsb | msb | | lsb |

The available alarm error codes are listed hereafter:

Byte 0

Machine data not valid

bit 0 One or more parameters are not valid, set proper values to restore normal work condition. To see in detail which parameter is wrong please enter the **Wrong parameters list [0003 hex]** register.

Flash memory error

bit 1 Flash memory internal error, it cannot be restored (bad checksum error, etc.).

Hall sensors error

bit 2 An error has occurred in the Hall sensors while reading the magnetic scale, the read value does not exist.

Mounting error

bit 3 The encoder is installed too far from the magnetic scale, the installation does not comply with the mounting tolerances between the sensor and the scale (see Figure 2). Refer to the "Mechanical installation" section on page 15.

bit 4 ... 7 Not used.

Byte 1

bit 8 ... 10 Not used.

Watchdog

bit 11 When the Watchdog function is enabled (**Watchdog enable** in **Control Word [000A hex]** is set to "1"), if the device does not receive any message from the Server within 1 second, the system forces an alarm condition (the **Watchdog** alarm bit is activated). The alarm is invoked to appear as soon as the Modbus network communication is restored. Watchdog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running, Watchdog safety system immediately takes action and commands an alarm to be triggered.

bits 12 ... 15 Not used.



NOTE

Please note that should the alarm be caused by wrong parameter values (see **Machine data not valid** and **Wrong parameters list [0003 hex]** register), normal work status can be restored only after having set proper values. The **Watchdog** alarm is deleted automatically as soon as the communication is restored. The **Flash memory error** alarm cannot be reset.

Current position [0001 hex]

[Register 2, Integer16, ro]

This register is meant to show the current position of the device at the moment when the request is sent. The output value is scaled according to the set scaling parameters, see **Scaling function** on page 64. Value is expressed in counts.

Current velocity [0002 hex]

[Register 3, Integer16, ro]

This register is not used and reserved for future use.

Wrong parameters list [0003 hex]

[Register 4, Unsigned16, ro]

The operator has entered invalid data and the **Machine data not valid** alarm has been triggered. This variable is meant to show in detail (bit value = HIGH) which parameter is wrong, according to the following table.

Please note that the normal work status can be restored only after having set proper values.

| Bit | Parameter |
|----------|---------------------------------|
| 0 | Not used |
| 1 | Resolution [0000 hex] |
| 2 | Preset value [0001 hex] |
| 3 | Offset value [0002 hex] |
| 4 | Operating parameters [0003 hex] |
| 5 | Node address [0004 hex] |
| 6 | Serial com baud rate [0005 hex] |
| 7 ... 15 | Not used |

SW Version [0004 hex]

[Register 5, Unsigned16, ro]

This is meant to show the software version of the encoder.

The major number shows the firmware edition, while the minor number shows the firmware revision.

The meaning of the 16 bits in the register is as follows:

| | | | | | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| Ms bit | | | | | | | | Ls bit | | | | | | | |
| Major number | | | | | | | | Minor number | | | | | | | |

Value 01 00 hex in hexadecimal notation corresponds to the binary representation 00000001 00000000 and has to be interpreted as: firmware edition 01, firmware revision 00.

HW Version [0005 hex]

[Register 6, Unsigned16, ro]

This is meant to show the hardware (PCB) version of the encoder.

The major number shows the hardware edition, while the minor number shows the hardware revision.

The meaning of the 16 bits in the register is as follows:

| | | | | | | | | | | | | | | | |
|--------------|----|----|----|----|----|----|----|--------------|----|----|----|----|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 09 | 08 | 07 | 06 | 05 | 04 | 03 | 02 | 01 | 00 |
| Ms bit | | | | | | | | Ls bit | | | | | | | |
| Major number | | | | | | | | Minor number | | | | | | | |

Value 01 01 hex in hexadecimal notation corresponds to the binary representation 00000001 00000001 and has to be interpreted as: hardware edition 01, hardware revision 01.

Status word [0006 hex]

[Register 7, Unsigned16, ro]

This register contains information about the current state of the device. The eight bits of Byte 0 (LSB) shows the currently set values of the **Operating parameters [0003 hex]** register Byte 0 (LSB); while bit 8 of MSB is used to signal active alarms.

Byte structure of the **Status word [0006 hex]** register:

| byte | MSB | | | LSB | | |
|------|-----|-----|-----|-----|-----|-----|
| bit | 15 | ... | 8 | 7 | ... | 0 |
| | msb | | lsb | msb | | lsb |

Byte 0

Scaling

bit 0

It shows the value which is currently set next to the **Scaling function** parameter. In other words, it is intended to show whether the scaling function is enabled or disabled. If the value is "=0" the scaling function is disabled; if the value is "=1" instead the scaling function is enabled. For any further information on setting and using the scaling function refer to the **Scaling function** parameter on page 64.

Counting direction

bit 1

It shows the value which is currently set next to the **Code sequence** parameter. If the bit is "=0" the

output encoder position value has been set to increment when the sensor moves in the direction shown by the arrow in Figure 1; if the bit is "1" instead the output encoder position value has been set to increment when the sensor moves in reverse of the standard direction, i.e. in the opposite direction to the one shown by the arrow in Figure 1. For any further information on setting and using the counting direction function refer to the **Code sequence** parameter on page 64.

bit 2 ... 7

Not used.

Byte 1 Alarm

bit 8

If value is "1" one or more alarms are active; to know in detail which alarm has been triggered, see the **Alarms register [0000 hex]** variable on page 69.

bit 9 ... 15

Not used.

7.2 Exception response and exception codes

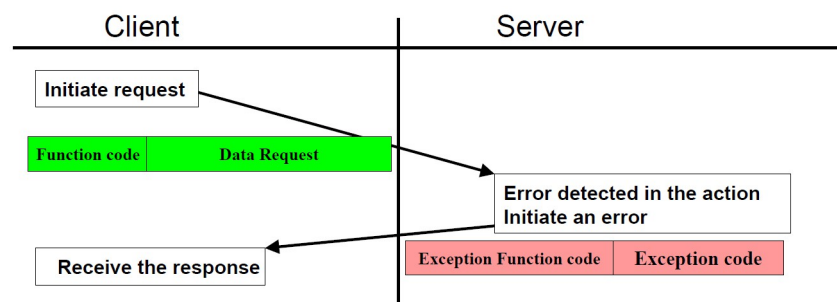
When a Client device sends a request to a Server device it expects a normal response. One of four possible events can occur from the Master's query:

- If the Server device receives the request without a communication error and can handle the query normally, it returns a normal response.
- If the Server does not receive the request due to a communication error, no response is returned. The client program will eventually process a timeout condition for the request.
- If the Server receives the request, but detects a communication error (parity, CRC, ...), no response is returned. The client program will eventually process a timeout condition for the request.
- If the Server receives the request without a communication error, but cannot handle it (for example, if the request is to read a non-existent output or register), the Server will return an exception response informing the Client about the nature of the error.

The exception response message has two fields that differentiate it from a normal response:

FUNCTION CODE FIELD: in a normal response, the Server echoes the function code of the original request in the function code field of the response. All function codes have a most significant bit (msb) of 0 (their values are all below 80 hexadecimal). In an exception response, the Server sets the msb of the function code to 1. This makes the function code value in an exception response exactly 80 hexadecimal higher than the value would be for a normal response. With the function code's msb set, the client's application program can recognize the exception response and can examine the data field for the exception code.

DATA FIELD: in a normal response, the Server may return data or statistics in the data field (any information that was requested in the request). In an exception code, the Server returns an exception code in the data field. This defines the Server condition that caused the exception.





NOTE

Please note that here follows the list the exception codes indicated by MODBUS but not necessarily supported by the manufacturer.

| MODBUS Exception codes | | |
|------------------------|-----------------------|---|
| Code | Name | Meaning |
| 01 | ILLEGAL FUNCTION | The function code received in the query is not an allowable action for the server. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server is in the wrong state to process a request of this type, for example because it is not configured and is being asked to return register values. |
| 02 | ILLEGAL DATA ADDRESS | The data address received in the query is not an allowable address for the server. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, the PDU addresses the first register as 0, and the last one as 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers 96, 97, 98, 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with Exception Code 0x02 "Illegal Data Address" since it attempts to operate on registers 96, 97, 98, 99 and 100, and there is no register with address 100. |
| 03 | ILLEGAL DATA VALUE | A value contained in the query data field is not an allowable value for server. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register. |
| 04 | SERVER DEVICE FAILURE | An unrecoverable error occurred while the server was attempting to perform the requested action. |

| | | |
|-----------|---|--|
| 05 | ACKNOWLEDGE | Specialized use in conjunction with programming commands. The server has accepted the request and is processing it, but a long duration of time will be required to do so. This response is returned to prevent a timeout error from occurring in the client. The client can next issue a Poll Program Complete message to determine if processing is completed. |
| 06 | SERVER DEVICE BUSY | Specialized use in conjunction with programming commands. The server is engaged in processing a long-duration program command. The client should retransmit the message later when the server is free. |
| 08 | MEMORY PARITY ERROR | Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server attempted to read record file, but detected a parity error in the memory. The client can retry the request, but service may be required on the server device. |
| 0A | GATEWAY PATH UNAVAILABLE | Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded. |
| 0B | GATEWAY TARGET DEVICE FAILED TO RESPOND | Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network. |

For any information on the available exception codes and their meaning refer to the "MODBUS Exception Responses" section on page 48 of the "MODBUS Application Protocol Specification V1.1b" document.

8 Programming examples

Hereafter are some examples of both reading and writing parameters. Unless otherwise stated, all values are expressed in hexadecimal notation.

8.1 Using the 03 Read Holding Registers function code



EXAMPLE 1

Request to read the **Preset value [0001 hex]** parameter (register 2) to the Slave having the node address 1.

Request PDU (in hexadecimal notation)

[01][03][00][01][00][01][D5][CA]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

[00][01] = starting address (**Preset value [0001 hex]** parameter, register 2)

[00][01] = number of requested registers

[D5][CA] = CRC

Response PDU (in hexadecimal notation)

[01][03][02][05][DC][BA][8D]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

[02] = number of bytes (2 bytes for each register)

[05][DC] = value of register 2, 05 DC hex = 1500 dec

[BA][8D] = CRC

Preset value [0001 hex] parameter (register 2) contains the value 05 DC hex, i.e. 1500 in decimal notation; in other words the value set in the **Preset value [0001 hex]** parameter is 1500 dec.

8.2 Using the 04 Read Input Register function code



EXAMPLE 1

Request to read the **Current position [0001 hex]** parameter (register 2) to the Slave having the node address 1.

Request PDU (in hexadecimal notation)

[01][04][00][01][00][01][60][0A]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[00][01] = starting address (**Current position [0001 hex]** parameter, register 2)

[00][01] = number of requested registers

[60][0A] = CRC

Response PDU (in hexadecimal notation)

[01][04][02][13][C5][74][53]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[02] = number of bytes (2 bytes for each register)

[13][C5] = value of register 2 **Current position [0001 hex]**, 13 C5 hex = 5061 dec

[74][53] = CRC

Current position [0001 hex] parameter (register 2) contains the value 13 C5 hex, i.e. 5061 in decimal notation.

8.3 Using the 06 Write Single Register function code



EXAMPLE 1

Request to write in the **Operating parameters [0003 hex]** register (register 4) to the Slave having the node address 1: we need to set the scaling function (**Scaling function** = 1) and the increasing counting when the sensor moves in the direction shown by the arrow in Figure 1 (**Code sequence** = 0). The value to set is 00 01 hex (= 0000 0000 0000 0001 in binary notation: bit 0 **Scaling function** = 1; bit 1 **Code sequence** = 0; the remaining bits are not used, therefore their value is 0).

Request PDU (in hexadecimal notation)

[01][06][00][03][00][01][B8][0A]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][03] = address of the register (**Operating parameters [0003 hex]** item, register 4)

[00][01] = value to be set in the register

[B8][0A] = CRC

Response PDU (in hexadecimal notation)

[01][06][00][03][00][01][B8][0A]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][03] = address of the register (**Operating parameters [0003 hex]** item, register 4)

[00][01] = value set in the register

[B8][0A] = CRC

The value 00 01 hex is set, i.e. 0000 0000 0000 0001 in binary notation: bit 0 **Scaling function** = 1; bit 1 **Code sequence** = 0; the remaining bits are not used, therefore their value is 0.

8.4 Using the 16 Write Multiple Registers function code



EXAMPLE 1

Request to write the value 00 0A hex (= 10 dec) next to the **Node address [0004 hex]** parameter (register 5) and the value 00 01 hex (= 1 dec = baud rate 9600 bit/s, parity bit Even) next to the **Serial com baud rate [0005 hex]** parameter (register 6) of the Slave having currently the node address 1.

Request PDU (in hexadecimal notation)

[01][10][00][04][00][02][04][00][0A][00][01][13][9E]

where:

[01] = Slave address

[10] = **16 Write Multiple Registers** function code

[00][04] = starting address (**Node address [0004 hex]** parameter, register 5)

[00][02] = number of requested registers

[04] = number of bytes (2 bytes for each register)

[00][0A] = value to be set in the register 5, 00 0A hex = 10 dec

[00][01] = value to be set in the register 6, 00 01 hex = 1 dec

[13][9E] = CRC

Response PDU (in hexadecimal notation)

[0A][10][00][04][00][02][01][72]

where:

[0A] = new Slave address

[10] = **16 Write Multiple Registers** function code

[00][04] = starting address (**Node address [0004 hex]** parameter, register 5)

[00][02] = number of written registers

[01][72] = CRC

The values 00 0A hex, i.e. 10 in decimal notation, is set in the register 5 **Node address [0004 hex]** parameter; while the value 00 01 hex, i.e. 1 in decimal notation = baud rate 9600 bit/s, parity bit Even, is set in the register 6 **Serial com baud rate [0005 hex]** parameter. Thus the encoder is programmed to have node address 10 and data transmission rate = 1 = baud rate 9600 bit/s, parity bit Even.

9 Default parameters list

9.1 List of the Holding Registers with default value

| Registers list and address | Default value | | |
|---|---------------|--|--|
| Resolution [0000 hex] hundredths of a mm | 10 | | |
| Preset value [0001 hex] | 0 | | |
| Offset value [0002 hex] | 0 | | |
| Scaling function in Operating parameters [0003 hex] | 0 | | |
| Code sequence in Operating parameters [0003 hex] | 0 | | |
| Node address [0004 hex] | 1 | | |
| Serial com baud rate [0005 hex] | 4 | | |
| Watchdog enable in Control Word [000A hex] | 0 | | |
| Save parameters in Control Word [000A hex] | - | | |
| Load default parameters in Control Word [000A hex] | - | | |
| Perform counting preset in Control Word [000A hex] | - | | |

9.2 List of the Input Registers

| Registers list and address | Description of the bits |
|----------------------------------|--|
| Alarms register [0000 hex] | 0 Machine data not valid 1 Flash memory error 2 Hall sensors error 3 Mounting error 11 Watchdog |
| Current position [0001 hex] | - |
| Current velocity [0002 hex] | - |
| Wrong parameters list [0003 hex] | 1 Resolution [0000 hex] 2 Preset value [0001 hex] 3 Offset value [0002 hex] 4 Operating parameters [0003 hex] 5 Node address [0004 hex] 6 Serial com baud rate [0005 hex] |
| SW Version [0004 hex] | - |
| HW Version [0005 hex] | - |
| Status word [0006 hex] | 0 Scaling 1 Counting direction 8 Alarm |

This page intentionally left blank

This page intentionally left blank

| Document release | Release date | Description | HW | SW | Version of the interface |
|------------------|--------------|---|-----|-----|--------------------------|
| 1.0 | 04.09.2014 | 1 st issue | 1.0 | 1.0 | - |
| 1.1 | 17.11.2015 | Hexadecimal value characters updating, general review, mounting tolerances correction | 1.0 | 1.0 | - |
| 1.2 | 13.02.2019 | General review, "Mechanical installation" section updated, description of the new programming interface | 1.0 | 1.0 | 1.1.0.0 |



Dispose separately

lika

Lika Electronic

Via S. Lorenzo, 25 • 36010 Carrè (VI) • Italy

Tel. +39 0445 806600

Fax +39 0445 806699



info@lika.biz • www.lika.biz