Smart encoders & actuators

User's guide

AM58S-1314-MTP9-M12
AM59H-1314-MT15-M12

- **Singleturn up to 13-bit, multiturn up to 14-bit**
- **Complies with the "Modbus over TCP/IP" protocol**
- **With Energy Harvesting Technology**
- **IP65 protection**
- **M12 output connectors**

General Contents

Preliminary information	8
1 - Safety summary	17
2 - Mounting instructions	19
3 - Electrical connections	23
4 - Quick reference	33
5 - MODBUS®/TCP interface	36
6 - Programming parameters	53
7 - Integrated web server	77

lika

General contents

User's guide 1
General contents 3
Subject index 6
Typographic and iconographic conventions 7
Preliminary information 8
Glossary of MODBUS/TCP terms 9
List of abbreviations 15
References 16
1 Safety summary 17
1.1 Safety 17
1.2 Electrical safety 17
1.3 Mechanical safety 18
2 Mounting instructions 19
2.1 Solid shaft encoders 19
2.1.1 Customary installation 19
2.1.2 Installation using fixing clamps (code LKM386) 20
2.1.3 Installation using a mounting bell (code PF4256) 20
2.2 Hollow shaft encoders 21
3 Electrical connections 22
3.1 M12 connectors 22
3.1.1 PWR Power supply connector (Figure 1) 22
3.1.2 P1 Port 1 and P2 Port 2 connectors (Figure 1) 23
3.2 Network configuration: topologies, cables, hubs, switches - Recommendations 23
3.3 Line Termination 24
3.4 Ground connection 24
3.5 Connection of the shield 24
3.6 MAC address and IP address 25
3.7 Setting the IP address and the network configuration parameters 25
3.7.1 Setting the node ID via software 26
3.7.2 DIP A DIP switch: Resetting the network configuration parameters to the factory values 26
3.8 Diagnostic LEDs (Figure 1) 27
3.9 LED state definition 30
3.10 Tactile switch (Figure 2) 31
4 Quick reference 32
4.1 Getting started 32
4.1.1 Setting the scaling function and custom resolution 33
4.1.2 Reading the absolute position 33
4.1.3 Reading the velocity value 34
4.1.4 Setting and executing the preset 34
4.1.5 Saving data 34
4.1.6 Restoring defaults 34
5 MODBUS®/TCP interface 35
5.1 MODBUS protocol principles 35
5.2 General MODBUS frame description 36
5.3 MODBUS on TCP/IP Application Data Unit 36

5.4 MODBUS PDUs ... 38
5.5 Function codes ... 39
5.5.1 Implemented function codes ... 40
03 Read Holding Registers ... 40
04 Read Input Registers ... 42
06 Write Single Register ... 44
16 Write Multiple Registers ... 46
5.6 Encoder states ... 51
WAIT_PROCESS ... 51
ERROR ... 51
PROCESS_ACTIVE ... 51
EXCEPTION ... 51
6 Programming parameters ... 52
6.1 Parameters available ... 52
6.1.1 Holding Register parameters ... 52
Watchdog timeout [82] ... 53
Current position [95-96] ... 53
Speed value [97-98] ... 53
Status word [99-100] ... 53
Counts per revolution [101-102] ... 53
Total Resolution [103-104] ... 55
Preset value [105-106] ... 57
Speed format [107-108] ... 59
Operating parameters [109-110] ... 59
Scaling function ... 59
Code sequence ... 60
Control Word [111-112] ... 61
Save parameters ... 61
Restore default parameters ... 61
Perform counting preset ... 62
Singleturn resolution [113-114] ... 63
Number of revolutions [115-116] ... 63
Supported alarms [117-118] ... 64
Supported warnings [119-120] ... 65
Alarm registers [121-122] ... 65
Position error ... 66
Machine data not valid ... 66
Setting data not valid ... 66
Flash memory error ... 66
Warning registers [123-124] ... 67
Position Warning ... 67
Wrong parameters list [125-126] ... 67
Counts per revolution error ... 68
Total resolution error ... 68
Preset value error ... 68
Offset value error ... 68
Offset value [127-128] ... 68
Software revision [129-130] ... 69
Hardware revision [131-132] ... 69

6.1.2 Input Register parameters ... 70
Current position [1-2] ... 70
Speed value [3-4] ... 70
Status word [5-6] ... 70
Scaling function ... 71
Code sequence ... 71
Alarm ... 71
6.2 Exception response and codes ... 73
7 Integrated web server ... 76
7.1 Integrated web server – Preliminary information ... 76
7.2 Web server Home page ... 77
7.3 Encoder position and speed ... 79
7.3.1 Specific notes on using Internet Explorer ... 80
7.4 Setting the Preset value ... 81
7.5 Encoder information (MODBUS registers) ... 83
7.6 Setting the registers ... 85
7.7 Network parameters ... 87
7.8 Firmware update ... 89
8 Programming examples ... 94
8.1 Using the 03 Read Holding Registers function code ... 94
8.2 Using the 04 Read Input Registers function code ... 95
8.3 Using the 06 Write Single Register function code ... 96
8.4 Using the 16 Write Multiple Registers function code ... 97
9 Default parameters list ... 99

Subject index

A

Alarm ... 71
Alarm registers [121-122] ... 65

C

Code sequence ... 60, 71
Control Word [111-112] ... 61
Counts per revolution [101-102] ... 53
Counts per revolution error ... 68
Current position [1-2] ... 70
Current position [95-96] ... 53

E

ERROR ... 51
EXCEPTION ... 51

F

Flash memory error ... 66

H

Hardware revision [131-132] ... 69

M

Machine data not valid ... 66

N

Number of revolutions [115-116] ... 63

O

Offset value [127-128] ... 68
Offset value error ... 68
Operating parameters [109-110] ... 59

P

Perform counting preset ... 62
Position error ... 66
Position Warning ... 65, 67
Preset value [105-106] ... 57
Preset value error ... 68
PROCESS_ACTIVE ... 51

R

Restore default parameters ... 61

S

Save parameters ... 61
Scaling function ... 59, 71
Setting data not valid ... 66
Singleturn resolution [113-114] ... 63
Software revision [129-130] ... 69
Speed format [107-108] ... 59
Speed value [3-4] ... 70
Speed value [97-98] ... 53
Status word [5-6] ... 70
Status word [99-100] ... 53
Supported alarms [117-118] ... 64
Supported warnings [119-120] ... 65

T

Total Resolution [103-104] ... 55
Total resolution error ... 68

W

WAIT_PROCESS ... 51
Warning registers [123-124] ... 67
Watchdog timeout [82] ... 53
Wrong parameters list [125-126] ... 67

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 the device and the 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 following rotary encoders **equipped with MODBUS/TCP interface**:

- AM58S-1314-MTP9-M12
- AM59H-1314-MT15-M12

For technical specifications please refer to the product datasheet.

Modbus over TCP/IP is an extension of the popular Modbus RTU protocol to replace the serial connection with Ethernet technology. According to a recent research report (2024), serial Modbus is estimated at 4% of the overall market share, and Modbus/TCP is currently at 4% as well. Both they are among the market-leading protocols; furthermore they are commonly and homogeneously used worldwide, from USA to Europe and Asia as well.
Among the reasons behind such popularity are simplicity and ease of interface, low-cost development, minimum hardware requirement and high reliability. Modbus/TCP is user-friendly and basic and can be implemented fast and uncomplicated.
These encoders are equipped with robust **magnetic sensing technology**. They are designed in a complete selection of mechanical versions: solid shaft versions and servo or clamp flange mounting; and blind hollow shaft versions. The industrial 58 mm flange enclosure enables IP65 protection. The range of the working temperature is comprised between -25°C and +85°C to cover most industrial applications.
The encoders are light and compact and integrate the **Energy Harvesting Technology** circuit. It enables the multiturn counter to be battery-free and gearless and the risk of mechanical failures to be significantly reduced at the same time.

To make it easier to read the text, this guide can be divided into three 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.
In the second section, entitled **MODBUS/TCP Interface** and **Programming parameters**, 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 third section, entitled **Integrated web server**, the integrated web server is described.

Glossary of MODBUS/TCP terms

MODBUS/TCP, 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/TCP interface. They are listed in alphabetical order.

ADU	Application Data Unit, it is the data frame of the MODBUS protocol. It takes the form of a 7 byte header (MBAP Header: transaction identifier + protocol identifier + length field + unit identifier), and the Protocol Data Unit (PDU: function code + data). The MODBUS/TCP ADU is inserted into the data field of a standard TCP frame and sent via TCP on registered port 502, which is specifically reserved for MODBUS applications. Thus, this packet is encapsulated by the data frames imposed by the TCP/IP stack of protocols (TCP/IP/MAC) before being transmitted onto the network. Refer to page 36.
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. TCP/IP only guarantees that application messages are transferred between the devices over the Ethernet Local-Area Network (LAN), it does not guaranty that the devices actually understand or interoperate with one another. For MODBUS/TCP, this capability is provided by the application layer protocol MODBUS.
Broadcast address	An IP address with a host portion that is all ones.
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 fiber optic.
Client	A Client is any network device that sends data requests to servers. MODBUS/TCP follows the Client/Server model. MODBUS Masters are referred to as Clients, while MODBUS Slaves are Servers.
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. Refer to page 36.
Determinism	It is the ability of the communication protocol to guaranty that a message is sent or received in a finite and predictable amount of time.
Deterministic Communication	It describes a communication process whose timing behavior can be predicted exactly. I.e. the time when a message reaches

	the recipient is predictable.
DHCP	DHCP (Dynamic Host Configuration Protocol) is a standardized network protocol used on Internet Protocol (IP) networks for dynamically distributing network configuration parameters, such as IP addresses for interfaces and services. A DHCP server assigns dynamic IP addresses at startup, and the addresses might change over time. DHCP servers on the network acknowledge the request by offering the client an IP address. The client acknowledges the first offer it receives, and the DHCP server in turn tells the client that it has succeeded in leasing the IP address for a specified amount of time.
DNS	DNS (Domain Name System) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. DNS is a host name resolution service that you can use to determine the IP address of a computer from its host name. This lets users work with host names, such as www.example.com, rather than an IP address, such as 192.168.5.102 or 192.168.12.68.
Encapsulation	The term "encapsulation" refers to the action of packing (embedding) the MODBUS message into the TCP container, the IP container, and the MAC container.
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. Refer to page 73.
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). Refer to pages 40 and 75.
Function code	MODBUS is a request/reply protocol and offers services 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. 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. Lika encoders only implement public function codes. Refer to page 39.
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. Refer to page 52.
Host	A computer or other device on a TCP/IP network.

IEEE 1588	This standard defines a protocol enabling Synchronization 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. Refer to page 70.
Internet	The global collection of networks that are connected together and share a common range of IP addresses.
InterNIC	The organization responsible for administration of IP addresses on the Internet.
IP	The network protocol used for sending network packets over a TCP/IP network or the Internet.
IP Address	The IP Address is a 32-bit number that uniquely identifies a host (computer or other device, such as a printer or router) on a TCP/IP network. IP addresses are normally expressed in dotted-decimal format, with four numbers separated by periods, such as 192.168.123.132. An IP address has two parts. The first part of an IP address is used as a network address, the last part as a host address. If you take the example 192.168.123.132 and divide it into these two parts you get the following: 192.168.123. = Network; .132 = Host. Or: 192.168.123.0 = network address; 0.0.0.132 = host address. Refer to page 25.
Isochronous	Pertains to processes that require timing coordination to be successful. Isochronous data transfer ensures that data flows continuously and at a steady rate in close timing with the ability of connected devices.
Legacy Ethernet	Ethernet as standardised in IEEE 802.3 (non-deterministic operation in non-time-critical environments).
MAC address	The MAC address is an identifier unique worldwide consisting of two parts: the first 3 bytes are the manufacturer ID and are provided by IEE standard authority; the last three bytes represent a consecutive number of the manufacturer. Refer to page 25.
Master	A Master is any network device that sends data requests to Slaves.
MBAP Header	The MBAP header (MODBUS Application Header) is a 7-byte header added to the start of the message and is used on TCP/IP to identify the MODBUS Application Data Unit. It has the following data: • Transaction Identifier: 2 bytes set by the Client to uniquely identify each request. These bytes are echoed by the Server since its responses may not be received in the same order as the requests. • Protocol Identifier: 2 bytes set by the Client, always = 00 00 • Length: 2 bytes identifying the number of bytes in the

	message to follow. • Unit Identifier: 1 byte set by the Client and echoed by the Server for identification of a remote slave connected on a serial line or on other buses. Refer to page 36.
Media Access Control (MAC)	One of the sub-layers of the Data Link Layer that controls who gets access to the medium to send a message.
Message	The MODBUS messaging service provides a Client/Server communication between devices connected on the Ethernet TCP/IP network. The Client / Server model is based on four types of messages: • MODBUS Request • MODBUS Confirmation • MODBUS Indication • MODBUS Response The MODBUS messaging services are used for information exchange.
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. Refer to page 38.
MODBUS Response	A MODBUS Response is the Response message sent by the Server. Refer to page 38.
Network	Network is a group of computers on a single physical network segment; otherwise it is an IP network address range that is allocated by a system administrator.
Network address	An IP address with a host portion that is all zeros.
Octet	An 8-bit number, 4 of which comprise a 32-bit IP address. They have a range of 00000000-11111111 that correspond to the decimal values 0- 255.
Packet	A unit of data passed over a TCP/IP network or wide area network.
PDU	The Protocol Data Unit (PDU) is the MODBUS function code and data field in their original form. It is packed together with the MBAP Header to form the Application Data Unit (ADU). The MODBUS protocol defines three PDUs. They are: • MODBUS Request PDU, mb_req_pdu • MODBUS Response PDU, mb_rsp_pdu • MODBUS Exception Response PDU, mb_excep_rsp_pdu Refer to page 38.
Port	It is an address that is used locally at the transport layer (on one node) and identifies the source and destination of the

	packet inside the same node. Port numbers are divided between well-known port numbers (0-1023), registered user port numbers (1024-49151) and private-dynamic port numbers (49152-65535). For TCP, port number 0 is reserved and cannot be used. Ports allow TCP/IP to multiplex and demultiplex a sequence of IP datagrams that need to go to many different (simultaneous) application processes. MODBUS/TCP uses well-known port 502 to listen and receive MODBUS messages over Ethernet.
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. Refer to page 40.
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. Refer to page 42.
Real-time	Real-time means that a system processes external events within a defined time. If the reaction of a system is predictable, one speaks of a deterministic system. The general requirements for real-time are therefore: deterministic response and defined response time.
Register	MODBUS functions operate on memory registers to configure, monitor, and control device I/O. Refer to page 52.
Router	A device that passes network traffic between different IP networks.
Server	A Server is any program that awaits data requests to be sent to it. Servers do no initiate contacts with Clients, but only respond to them. MODBUS/TCP follows the Client/Server model. MODBUS Masters are referred to as clients, while MODBUS Slaves are servers.
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 no initiate contacts with Masters, but only respond to them.
Subnet Mask	A 32-bit number used to distinguish the network and host portions of an IP address. In other terms, it is used by the TCP/IP protocol to determine whether a host is on the local subnet or on a remote network.
Subnet or Subnetwork	A smaller network created by dividing a larger network into equal parts.
TCP/IP	Used broadly, the set of protocols, standards and utilities commonly used on the Internet and large networks. The Ethernet system is designed solely to carry data. It is comparable to a highway as a system for transporting goods

	and passengers. The data is actually transported by protocols. This is comparable to cars and commercial vehicles transporting passengers and goods on the highway. Tasks handled by the basic Transmission Control Protocol (TCP) and Internet Protocol (IP) (abbreviated to TCP/IP): 1. The sender splits the data into a sequence of packets. 2. The packets are transported over the Ethernet to the correct recipient. 3. The recipient reassembles the data packets in the correct order. 4. Faulty packets are sent again until the recipient acknowledges that they have been transferred successfully.
Topology	Network structure. Commonly used structures: • Line topology; • Ring topology; • Star topology; • Tree topology. Refer to page 23.
Transmission rate	Data transfer rate (in bps). Refer to page 23.
Wide area network (WAN)	A large network that is a collection of smaller networks separated by routers. The Internet is an example of a very large WAN.
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. Refer to page 46.
Write Single Register (06, 0006hex)	This function code is used to WRITE a single holding register in a remote device. Refer to page 44.

List of abbreviations

Table below contains a list of abbreviations (in alphabetical order) which may be used in this guide to describe the MODBUS/TCP interface.

ADU	Application Data Unit
AP (task)	Application (task)
BOOTP	Bootstrap Protocol
CMD	Command Table
DHCP	Dynamic Host Configuration Protocol
DPM	Dual-port memory
HDLC	High level Data Link Control
HMI	Human Machine Interface
I/O	Input/Output
IETF	Internet Engineering Task Force
IP	Internet Protocol
MAC	Media Access Control
MB	MODBUS Protocol
MBAP	MODBUS Application Protocol
MBAP header	MODBUS Application Header
OMB	Open Modbus/TCP
OMBV3	Open Modbus/TCP version 3
OMBV5	Open Modbus/TCP version 5
PDU	Protocol Data Unit
PLC	Programmable Logic Controller
RTE	Real Time Ethernet
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

References

[1] MODBUS Application Protocol Specification, V1.1a, June 4

[2] MODBUS Application Protocol Specification, Version V1.1b3

[3] MODBUS messaging on TCP/IP implementation guide, V1.0a, June 4, 2004

[4] MODBUS messaging on TCP/IP implementation guide, Version V1.0b

[5] RFC 791, Internet Protocol, Sep81 DARPA

[6] RFC 1122 Requirements for Internet Hosts -- Communication Layers

[7] IEC 61918 Industrial communication networks – Installation of communication networks in industrial premises

[8] IEC 61784-5-13 Industrial communication networks – Profiles – Part 5-13: Installation of fieldbuses – Installation profiles for CPF 13

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 the power supply before connecting the device;
- connect according to the explanation in the "3 - Electrical connections" section on page 22;
- connect +Vdc and 0Vdc and check the power supply is correct first before connecting the communication ports;
- in compliance with the 2014/30/EU norm on electromagnetic compatibility, the following precautions must be taken:
  - before handling and installing, 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 frame 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. Provide the ground connection as close as possible to the

encoder. We suggest using the ground point provided in the housing, use 1 TCEI UNI M3 x 6 cylindrical head screw with two tooth lock washers.

1.3 Mechanical safety

- Install the device following strictly the information in the "2 - Mounting instructions" section on page 19;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the encoder;
- do not tool the encoder or its shaft;
- delicate electronic equipment: handle with care; do not subject the device and the shaft to knocks or shocks;
- respect the environmental characteristics declared by manufacturer
- unit with solid shaft: in order to guarantee maximum reliability over time of mechanical parts, we recommend a flexible coupling to be installed to connect the encoder and user's shaft; make sure the misalignment tolerances of the flexible coupling are respected;
- unit with hollow shaft: the encoder can be mounted directly on a shaft whose diameter has to respect the technical characteristics specified in the purchase order and clamped by means of the collar and, when requested, the anti-rotation pin.

2 Mounting instructions

WARNING

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

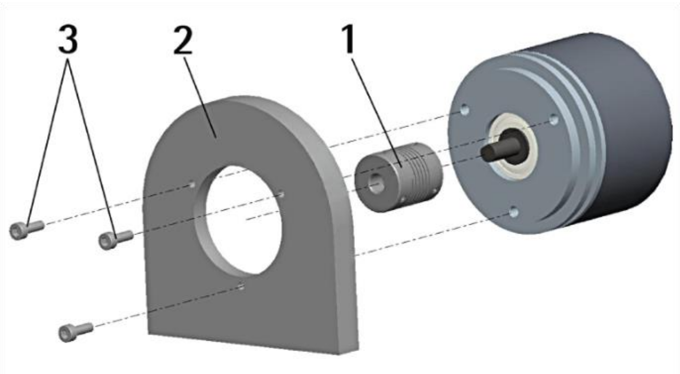
For any information on the mechanical data and the electrical characteristics of the encoder please refer to the encoder datasheet.

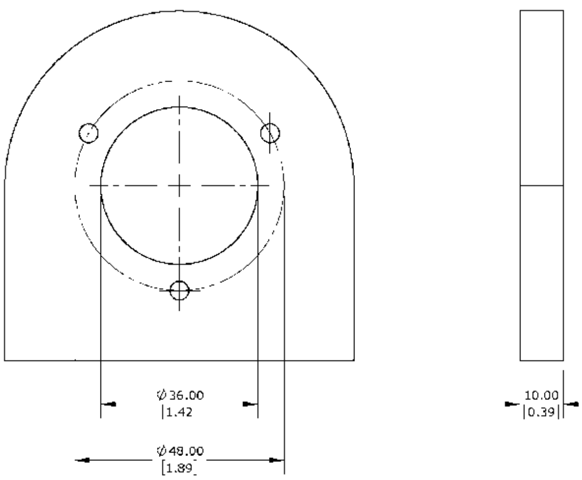
Values are expressed in millimeters (mm).

2.1 Solid shaft encoders

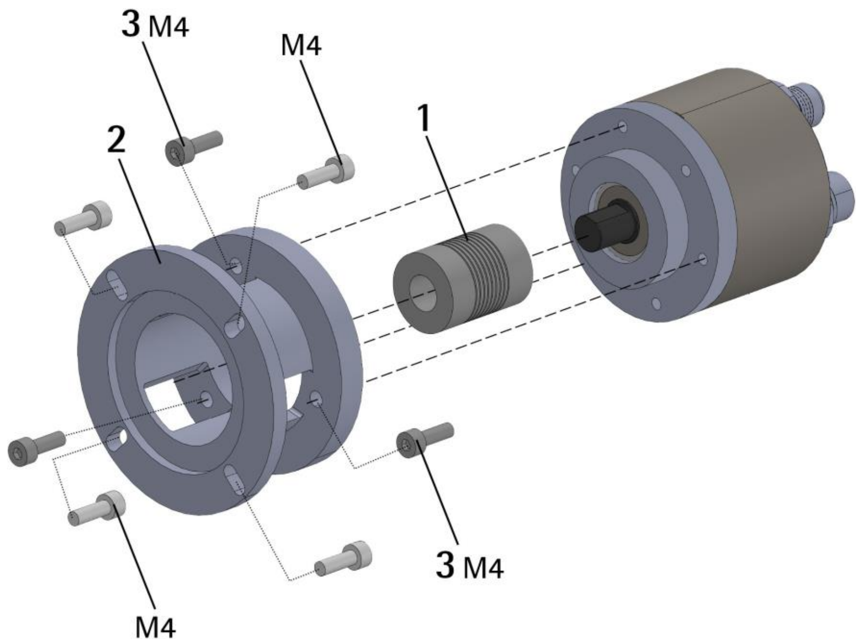
- Mount the flexible coupling **1** on the encoder shaft;
- fix the encoder to the flange **2** (or to the mounting bell) by means of the M4 screws **3**;
- mount the flexible coupling **1** on the motor shaft;
- secure the flange **2** to the support (or the mounting bell to the motor);
- make sure the misalignment tolerances of the flexible coupling **1** are met.

2.1.1 Standard solid shaft installation





2.1.2 Solid shaft installation using PF4256 mounting bell

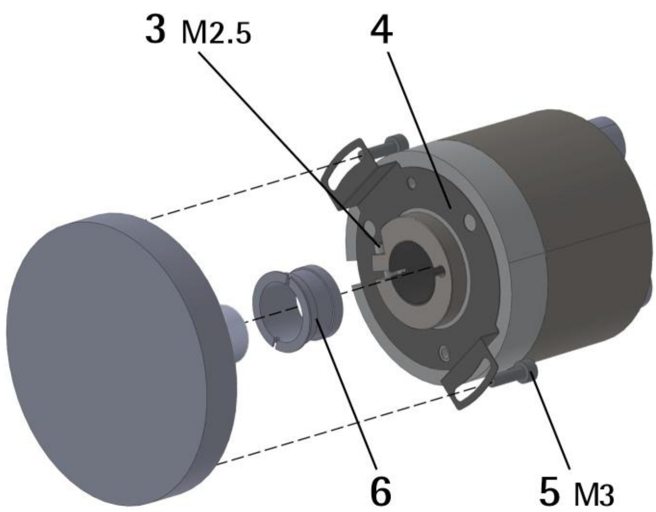


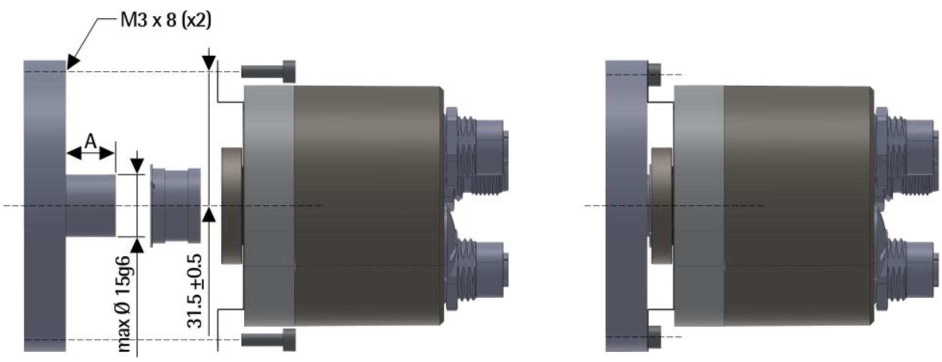
NOTE

In order to guarantee reliability over time of the encoder mechanical parts, we recommend a flexible coupling to be installed between the encoder and the motor shaft. Make sure the misalignment tolerances of the flexible coupling are met.

2.2 Standard hollow shaft installation

- Mount the encoder on the motor shaft using the BR1-xx reducing sleeve **6** (if supplied). Avoid forcing the encoder shaft;
- fasten the fixing plate **4** to the rear of the motor using two M3 cylindrical head screws **5**;
- fix the collar to the encoder shaft by means of the M2.5 screw **3**).





A = min. 8 mm, max. 18 mm

NOTE
You are strongly advised not to carry out any mechanical operations (drilling, milling, etc.) on the encoder shaft. This could cause serious damages to the internal parts and an immediate warranty loss. Please contact our technical personnel for the complete availability of "custom made" shafts.

3 Electrical connections

WARNING
Power supply must be turned off before performing any electrical connection! Installation, electrical connection and maintenance operations must be carried out by qualified personnel only, with power supply disconnected. Mechanical components must be in stop.
Do not remove the plug **A** on the rear of the encoder unless otherwise indicated. Damage may be caused to internal components.

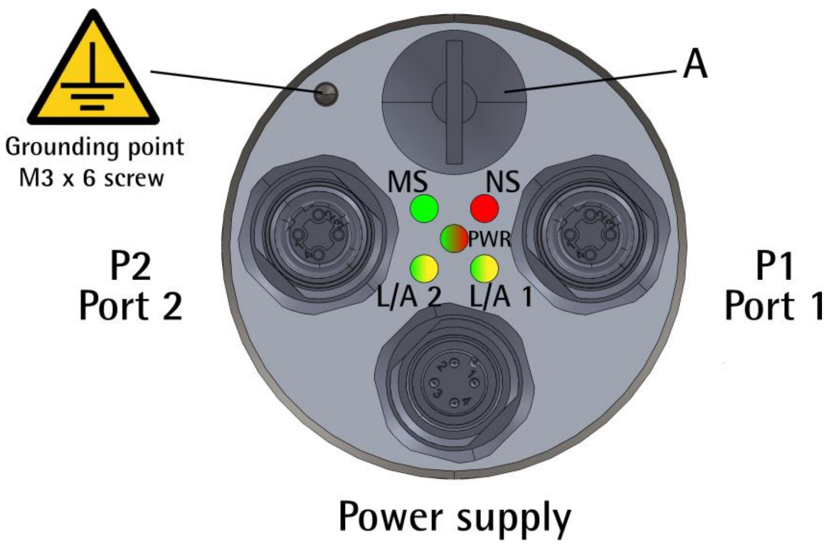


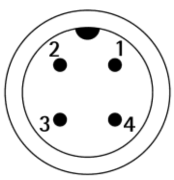
Figure 1 - Connectors and diagnostic LEDs

3.1 M12 connectors

Three M12 connectors with pin-out in compliance with the Ethernet standard are mounted on the encoder housing. Therefore you can use standard Ethernet cables commercially available. PORT 1 and PORT 2 are interchangeable.

3.1.1 PWR Power supply connector (Figure 1)

M12 4-pin male connector with A coding is used for power supply.



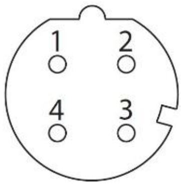
Description	Pin
+5Vdc +30Vdc	1
n.c.	2
0Vdc	3
n.c.	4

n.c. = not connected

WARNING
Connect +Vdc and 0Vdc and check the power supply is correct first before connecting the communication ports.

3.1.2 P1 Port 1 and P2 Port 2 connectors (Figure 1)

Two M12 4-pin female connectors with D coding are used for Ethernet connection through port 1 and port 2.



Description	Pin
Tx Data +	1
Rx Data +	2
Tx Data -	3
Rx Data -	4

The Ethernet interface supports 10/100 Mbit/s, full/half duplex operation.
P1 PORT 1 and P2 PORT 2 M12 connectors have pin-out in compliance with the Ethernet standard. Therefore you can use standard Ethernet cables commercially available, for more information see further on.
The ports are equal and interchangeable - if only one connection is required, either port can be used.

WARNING
Connect +Vdc and 0Vdc and check the power supply is correct first before connecting the communication ports.

3.2 Network configuration: topologies, cables, hubs, switches - Recommendations

Using Ethernet several topologies of connection are supported by MODBUS/TCP networks: line, tree, daisy-chain, star, … . Furthermore MODBUS/TCP networks can be configured in almost any topology in the same structure.
The connection of the encoder can be made directly with a network card or indirectly with a switch, hub, or company network.
Cables and connectors comply with the IEEE 802.3 Ethernet specifications.
If you use a direct connection to a computer/controller without network components in between, you need to use a standard, "straight" network cable (not a crossover cable).
You need at least a CAT-5 cable (category 5) to get a data transfer rate up to 100 Mbit/s. If there is a network component in the network which does not provide fast Ethernet, the encoder will automatically switch down to 10 Mbit/s.
Standard Ethernet cables commercially available can be used.
For complete information please refer to IEC 61918, IEC 61784-5-13 and IEC 61076-2-101.
To increase noise immunity only S/FTP or SF/FTP cables must be used (CAT-5).
The maximum cable length (100 meters) predefined by Ethernet 100Base-TX must be compulsorily fulfilled.

Regarding wiring and EMC measures, the IEC 61918 and IEC 61784-5-13 must be considered.

3.3 Line Termination

MODBUS/TCP network needs no line termination because the line is terminated automatically; in fact every Slave is able to detect the presence of the downstream Slaves.

3.4 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.

3.5 MAC address and IP address

The unit can be identified in the network through the **MAC address** and the **IP address**.

The MAC address is an identifier unique worldwide and has to be intended as a permanent and globally unique identifier assigned to the unit for communication on the physical layer; while the IP address is the name of the unit in a network using the Internet protocol. The MAC address is 6-byte long and cannot be modified. It consists of two parts, numbers are expressed in hexadecimal notation: the first three bytes are used to identify the manufacturer (OUI, namely Organizationally Unique Identifier) and are provided by IEE standard authority; while the last three bytes represent a consecutive number of the manufacturer and are the specific identifier of the unit. The MAC address can be found for commissioning purposes on the label applied to the encoder and is displayed in the **Encoder Information** page of the web server.

The MAC address has the following structure:

Bit value 47 ... 24			**Bit value 23 ... 0**		
10	B9	FE	X	X	X
Company code (OUI)			**Consecutive number**		

The IP address must be assigned by the user to each interface of the unit to be connected in the network, the default IP address assigned by Lika Electronic is 192.168.1.10, while the subnet mask is always 255.255.255.0 as in a class C net. To set the network configuration parameters refer to the next section.

3.6 Setting the IP address and the network configuration parameters

WARNING
Only competent technicians, who are properly trained, have adequate experience and are familiar with computer architecture, network design and operating systems should configure the network communication parameters.
The inappropriate setting of the network parameters results in an incorrect operation of the system.

WARNING
The MODBUS TCP/IP address and communication parameters can be set only via software by connecting to the encoder via the Web server.

The following table summarises the default IP address and the network configuration parameters.

IP Parameter	Value
IP address	192.168.1.10
Subnet mask	255.255.255.0
Default Gateway	0.0.0.0

To configure the network and set specific communication parameters, the operator must enter the **Network parameters** page of the Web server.
Any change is valid in the range: 0.0.0.0 ... 255.255.255.255 in compliance with the Internet Protocol rules.
For any information on the **Network parameters** page refer to the "7.7 Network parameters" section on page 87.

NOTE
If for any reason you must restore the factory values (default values) of the network configuration parameters you must access the **DIP A** DIP switch located inside the encoder after removing the **A** plug (see Figure 1). For complete information please refer to the "3.7.2 DIP A DIP switch: Resetting the network configuration parameters to the factory values" section on page 26.

3.6.1 Setting the node ID via software

The software values can be changed by using the web server and setting a proper value next to the **IP Address**, **Subnet Mask**, and **Gateway** items, refer to the "7.7 Network parameters" section on page 87.
Any Net ID value and Host ID value can be set.

3.6.2 DIP A DIP switch: Resetting the network configuration parameters to the factory values

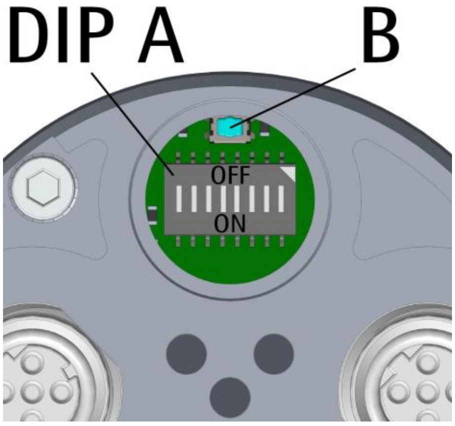
WARNING
Power supply must be turned off before setting the **DIP A** DIP switch!
Be careful not to press the **B** tactile switch.

WARNING
To access the **DIP A** DIP switch meant to reset the network configuration parameters to the factory values (default values) you must remove the A plug (see Figure 1).

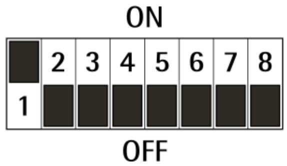
If for any reason you must restore the factory values (default values) of the network configuration parameters (IP address, Subnet mask, etc.), access the **DIP A** DIP switch proceeding as follows.

WARNING

Please pay the utmost attention to the internal wirings and connections while the **A** plug is removed.



- Turn the power supply off;
- loosen and remove the **A** plug (see Figure 1);
- set the hardware switch 1 to ON;



- turn the power supply on and wait for the initialization process to be completed;
- turn the power supply off;
- set the hardware switch 1 to OFF again;
- replace and tighten firmly the **A** plug;
- turn the power supply on to restore the normal encoder operation.

The following table summarises the IP address and the network configuration parameters after reset.

IP Parameter	Value
IP address	192.168.1.10
Subnet mask	255.255.255.0
Gateway	0.0.0.0

3.7 Diagnostic LEDs (Figure 1)

Five LEDs located in the rear of the encoder (see the Figure 1) are meant to show visually the operating or fault status of the encoder and the MODBUS interface. The meaning of each LED is explained in the following tables.

NOTE

The **OMB task** is the Open Modbus/TCP stack implementation. It is responsible for the protocol handling, the communication from and to the TCP/IP stack and it is the counterpart of the AP task.

The **AP task** / GenAP task provides the interface to the user application and the control of the stack. It also completely handles the dual-port memory interface of the communication channel. In detail, it is responsible for the following:

- Handling the communication channels dual-port memory interface
- Configuration management
- Mailbox packet handling and routing
- Control of LEDs

MS Module Status LED (green)

It shows the state of the MODBUS/TCP device. It is also referred to as RUN LED.

MS LED	Description
OFF	**Not ready**: the OMB task is not ready.
ON green	**Connected**: the OMB task has communication. At least one TCP connection is established.
FLASHING (1 Hz) green	**Ready, not configured yet**: the OMB task is ready and not configured yet.
FLASHING (5 Hz) green	**Waiting for communication**: the OMB task is configured.

NS Network Status LED (red)

It shows the current state of the network. It is also referred to as ERR LED.

LED	Description
OFF	No communication error is active. For more information refer to page 73.
ON red	A communication error is active. For more information refer to page 73.
FLASHING red (2 Hz, 25% ON)	A system error is active. For more information refer to page 73.

PWR Power LED (green / red)

It shows the power supply and system state. It is also referred to as SYS (System) LED.

PWR LED	Description	Meaning
OFF	Power OFF	The encoder power supply is switched OFF. No supply voltage for the device or hardware fault.
ON green	Power ON	The encoder power supply is switched ON. The firmware is running.
BLINKING red	No firmware program installed, firmware update mode	At power ON the LED blinks red at 1 Hz. The firmware program is not installed, the encoder enters the firmware update mode and waits for the firmware file to be installed.

L/A Link/Activity LED for port 2 P2 (green / yellow)

It shows the state and the activity of the physical link (port 2 P2).

L/A LED	Description	Meaning
OFF	No link No activity	The device has no link to the Ethernet, the link through port 2 P2 is not active. There is no activity on port 2 P2, the device does not send/receive Ethernet frames through port 2 P2.
ON green	Link active No activity	Port 2 P2 link active, the device is linked to the Ethernet, there is no activity on port 2 P2.
FLICKERING yellow	Activity	Port 2 P2 link is active, there is activity on port 2 P2, the device sends/receives Ethernet frames through port 2 P2.

L/A Link/Activity LED for port 1 P1 (green / yellow)

It shows the state and the activity of the physical link (port 1 P1).

LED	Description	Meaning
OFF	No link No activity	The device has no link to the Ethernet, the link through port 1 P1 is not active. There is no activity on port 1 P1, the device does not send/receive Ethernet frames through port 1 P1.
ON green	Link active No activity	Port 1 P1 link active, the device is linked to the Ethernet, there is no activity on port 1 P1.
FLICKERING yellow	Activity	Port 1 P1 link is active, there is activity on port 1 P1, the device sends/receives Ethernet frames through port 1 P1.

3.8 LED state definition

LED state	Description
Flashing (1 Hz)	The LED turns ON and OFF with a frequency of 1 Hz: "ON" for 500 ms, followed by "OFF" for 500 ms.
Flashing (2 Hz, 25% ON)	The LED turns ON and OFF with a frequency of 2 Hz: "ON" for 125 ms followed by "OFF" for 375 ms.
Blinking (5 Hz)	The LED turns ON and OFF with a frequency of 5 Hz: "ON" for 100 ms followed by "OFF" for 100 ms.
Flickering (load dependent)	The LED turns ON and OFF with a frequency of approximately 10 Hz to indicate high Ethernet activity: "ON" for approximately 50 ms, followed by "OFF" for 50 ms. The LED turns ON and OFF in irregular intervals to indicate low Ethernet activity.

3.9 Tactile switch (Figure 2)

WARNING
Be careful not to press the **B** tactile switch unless specifically requested.

The B tactile switch is located inside the enclosure. You must remove the A plug (see Figure 1) to access it. It has no useful function to the operator under the normal usage conditions, so never press it unless specifically requested by Lika Electronic's technicians.

4 Quick reference

4.1 Getting started

The following instructions allow the operator to quickly and safely set up the encoder in a standard operational mode and to execute its main functions.
Sometimes a function or a procedure can be accomplished by using alternative ways, i.e. by means of the Integrated Web Server (see the "7 Integrated web server" section on page 76).
They are all mentioned whenever available.
For complete and detailed information please read the mentioned pages thoroughly.

- Mechanically install the device (see on page 19);
- execute the power supply connections and switch on the +5Vdc +30Vdc power supply, see on page 24 ff; check the soundness of the connection;
- switch off the power supply and execute the network connections, then switch on the power supply again, see on page 24 ff; check the soundness of the connection;
- if required, set the communication parameters to allow the unit to access the MODBUS/TCP network, see the "3.7 Setting the IP address and the network configuration parameters" section on page 25; the default network configuration parameters are as follows:

IP Parameter	Value
IP address	192.168.1.10
Subnet mask	255.255.255.0
Default Gateway	0.0.0.0

- you are not required to set any line termination (see on page 24);
- you are not required to install any configuration file;
- if you need to use the physical resolution of the unit, please check that the **Scaling function** parameter is disabled (the bit 0 in the **Operating parameters [109-110]** registers = 0; see on page 59); the encoder will use the **Singleturn resolution [113-114]** and the **Number of revolutions [115-116]** register values to calculate the absolute position information;
- otherwise if you need a custom resolution, please enable the **Scaling function** parameter (the bit 0 in the **Operating parameters [109-110]** registers = 1; see on page 59);
- then set the custom value you need for the singleturn resolution next to the **Counts per revolution [101-102]** registers, see on page 53;
- set the custom value you need for the overall resolution next to the **Total Resolution [103-104]** registers, see on page 55;
- now, if you need you can set a Preset value next to the **Preset value [105-106]** registers and then activate it by executing the **Perform**

counting preset command available in the **Control Word [111-112]** registers, see on page 57;
- save the new setting values (use the **Save parameters** command available in the **Control Word [111-112]** registers, see on page 61).

NOTE
MODBUS/TCP protocol does not require any configuration file.

4.1.1 Setting the scaling function and custom resolution

- If you want to use the physical resolution of the encoder, please check that the **Scaling function** parameter is disabled (the bit 0 in the **Operating parameters [109-110]** registers is ="0", see on page 59); in this case, the device uses the physical resolution (see the **Singleturn resolution [113-114]** and **Number of revolutions [115-116]** registers) to calculate the absolute position value.
  You can also use the Integrated Web Server, see the "7.6 Setting the registers" section on page 85.
- On the contrary, if you need a custom resolution, you must enable the scaling function by setting the **Scaling function** parameter (the bit 0 in the **Operating parameters [109-110]** registers) to ="1" first and then set the required resolution registers:
  - set the custom singleturn resolution next to the **Counts per revolution [101-102]** registers, see on page 53;
  - set the custom total resolution next to the **Total Resolution [103-104]** registers, see on page 55.

  You can also use the Integrated Web Server, see the "7.6 Setting the registers" section on page 85.

4.1.2 Reading the absolute position

To read the position value you can choose among the following methods.

- To read the absolute position of the encoder see the **Current position [95-96]** registers in the Holding registers, see on page 53; or see the **Current position [1-2]** registers in the Input registers, see on page 70.
- Open the Integrated Web Server, see the "7.3 Encoder position and speed" section on page 79.

4.1.3 Reading the velocity value

To read the velocity value you can choose among the following methods.

- To read the velocity value of the encoder see the **Speed value [97-98]** registers in the Holding registers, see on page 53; or see the **Speed value [3-4]** registers in the Input registers, see on page 70.
- Open the Integrated Web Server, see the "7.3 Encoder position and speed" section on page 79.

4.1.4 Setting and executing the preset

To set and execute the preset you can choose among the following methods.

- Enter a suitable value next to the **Preset value [105-106]** registers and then activate it by executing the bit 11 **Perform counting preset** command available in the **Control Word [111-112]** registers, see on page 57.
- Open the **Preset** page in the Integrated Web Server, see the "7.4 Setting the Preset value" section on page 81.

4.1.5 Saving data

To save the parameters permanently you can choose among the following methods.

- Set properly the bit 9 **Save parameters** command in the **Control Word [111-112]** registers, see on page 61.
- Press the **SAVE PARAMETERS** button in the **Set Encoder Registers** page of the Integrated Web Server, see the "7.6 Setting the registers" section on page 85.

4.1.6 Restoring defaults

To restore the default parameters you can choose among the following methods.

- Set properly the bit 10 **Restore default parameters** command in the **Control Word [111-112]** registers, see on page 61.
- Press the **LOAD DEFAULT PARAM.** button in the **Set Encoder Registers** page of the Integrated Web Server, see the "7.6 Setting the registers" section on page 85.

5 MODBUS®/TCP interface

Lika MODBUS/TCP encoders are Slave (Server) devices and implement the MODBUS application protocol (level 7 of OSI model) and the "MODBUS messaging on TCP/IP" protocol (Ethernet: levels 1 & 2 of OSI model; TCP/IP: levels 3 & 4 of OSI model).
For any further information or omitted specifications please refer to "MODBUS Application Protocol Specification, Version V1.1b3" and "MODBUS messaging on TCP/IP implementation guide, Version V1.0b" available at www.modbus.org.

5.1 MODBUS protocol principles

MODBUS is an application layer messaging protocol, positioned at level 7 of the OSI model, which provides Client/Server communication between devices connected on different types of buses or networks. In particular, the MODBUS/TCP messaging service provides a Client/Server communication between devices connected on an Ethernet TCP/IP network.
The Modbus protocol was developed in 1979 by Modicon, for industrial automation systems and Modicon programmable controllers. It has since become an industry standard method for the transfer of discrete/analogue I/O information and register data between industrial control and monitoring devices.

MODBUS devices communicate using a Master-Slave (Client-Server) technique in which only one device (the Master/Client) can initiate transactions (called queries). The other devices (Slaves/Servers) respond by supplying the requested data to the Master, or by taking the action requested in the query. A Slave is any peripheral device (I/O transducer, valve, network drive, or other measuring device) which processes information and sends its output to the Master using MODBUS.
Masters can address individual Slaves, or can initiate a broadcast message to all Slaves. Slaves return a response to all queries addressed to them individually, but do not respond to broadcast queries. Slaves do not initiate messages on their own, they only respond to queries from the Master.

MODBUS/TCP (also MODBUS-TCP or MODBUS TCP/IP) is simply the MODBUS RTU protocol with a TCP interface that runs on Ethernet.
The MODBUS messaging structure is the application protocol that defines the rules for organizing and interpreting the data independent of the data transmission medium.
TCP/IP refers to the Transmission Control Protocol and Internet Protocol, which provides the transmission medium for MODBUS/TCP messaging.

Among the significant advantages of MODBUS/TCP are:

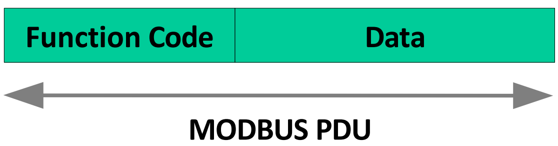
- MODBUS/TCP simply takes the MODBUS instruction set and wraps TCP/IP around it;

- it supports standard Ethernet and does not require dedicated Masters or chipsets; standard PC Ethernet cards and PCs can be used to communicate in any Ethernet network;
- it does not require any configuration file;
- it does not need any specific software thanks to the possibility of integrating a web server: it is designed to offer helpful functions and deliver complete information on the device that can be accessed through the Internet.
  In particular it allows:
  - to display and check the currently set parameters;
  - to set the network communication parameters;
  - to set the device work parameters;
  - to upgrade the firmware;
  - to monitor the device and access some advanced maintenance functions.

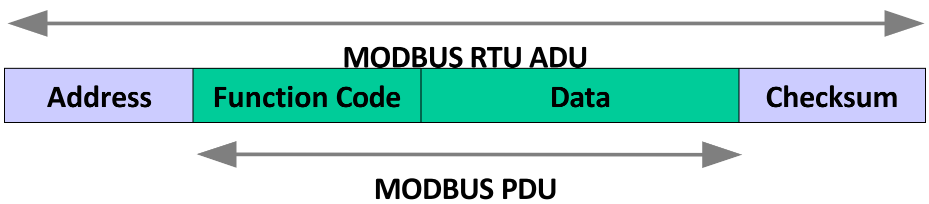
  The web server can be accessed from any PC running a web browser.

5.2 General MODBUS frame description

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



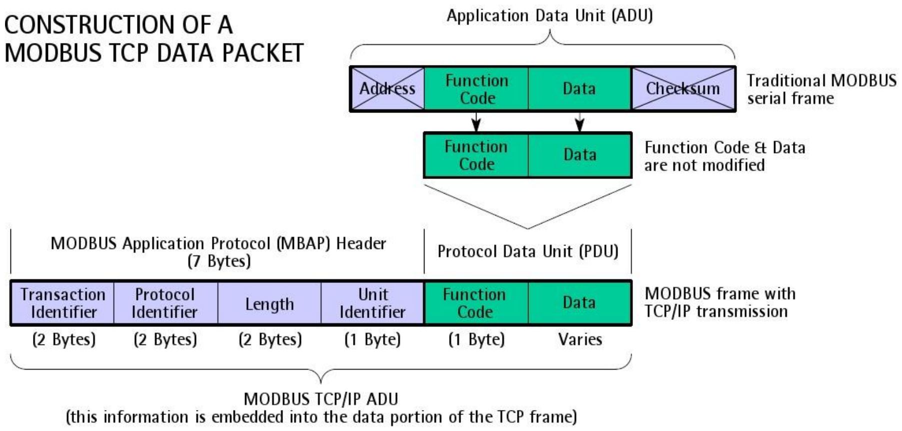
The mapping of MODBUS protocol on a specific bus or network introduces some additional fields on the **Application Data Unit (ADU)**. The Client that initiates a MODBUS transaction builds the MODBUS Application Data Unit, and then adds fields in order to build the appropriate communication ADU. The function code indicates to the Server which kind of action to perform.



5.3 MODBUS on TCP/IP Application Data Unit

MODBUS/TCP uses TCP/IP and Ethernet to carry the data of the MODBUS message structure between compatible devices. That is, MODBUS/TCP combines a physical network (Ethernet) with a networking standard (TCP/IP), and a standard method of representing data (MODBUS as the application protocol).

Essentially, the MODBUS/TCP message is simply a MODBUS communication encapsulated in an Ethernet TCP/IP wrapper.
In practice, MODBUS TCP embeds a standard MODBUS data frame into a TCP frame, without the MODBUS RTU address and checksum, as shown in the following diagram.



As you can see, the Protocol Data Unit is integrated in its original form.
The MODBUS/TCP Application Data Unit (ADU) takes the form of a 7-byte header (MBAP Header -MODBUS Application Protocol Header: Transaction Identifier + Protocol Identifier + Length field + Unit Identifier), and the protocol data unit (MODBUS PDU: Function Code + Data).

- **MBAP HEADER** (MODBUS Application Protocol Header). A dedicated header is used on TCP/IP to identify the MODBUS Application Data Unit. The MBAP Header contains the following fields:
  - **Transaction Identifier**: it is 2-byte long and is used for transaction pairing, i.e. for identification of a MODBUS Request / Response transaction. It is initialized by the Client, the Server copies in the response the Transaction Identifier received with the request.
  - **Protocol Identifier**: it is 2-byte long and is used for intra-system multiplexing. The MODBUS protocol is identified by the value 0. It is initialized by the Client, the Server copies in the response the Protocol Identifier received with the request.
  - **Length**: it is 2-byte long and is a byte count of the following fields, including the Unit Identifier, the Function Code and the Data field. It is initialized by the Client in the request, and it is initialized by the Server in the response.
  - **Unit Identifier**: it is 1-byte long and is used for intra-system routing purpose. It is typically used to communicate to a MODBUS+ or a MODBUS serial line Slave through a gateway between an Ethernet TCP/IP network and a MODBUS serial line. This field is set by the MODBUS Client in the request and must be returned with the same value in the response by the Server. In a typical MODBUS/TCP Server application, the Unit Identifier is set to 00 hex or FF hex.
- **FUNCTION CODE**: the function code indicates to the Server what kind of action to perform. The function code is followed by a **DATA** field that

contains request and response parameters. All MODBUS request and responses are designed in such a way that the recipient can verify that a message is finished. For function codes where the MODBUS PDU has a fixed length, the function code alone is enough. For function codes carrying a variable amount of data in the request or in the response, the data field includes a byte count. For any further information on the implemented function codes refer to the "5.5 Function codes" section on page 39.

- **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 "5.5 Function codes" section on page 39).

The complete MODBUS/TCP Application Data Unit is embedded into the data field of a standard TCP frame and sent via TCP to **registered port 502**, which is specifically reserved for MODBUS applications. MODBUS/TCP Clients and Servers listen and receive MODBUS data via port 502.

NOTE
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 (MSB) is sent first. So for example:

Register size	value	
16–bit	1234hex	the first byte sent is 12hex, then 34hex

5.4 MODBUS PDUs

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 + 0080 hex [1 byte];
exception_code = MODBUS Exception code, refer to the table "MODBUS Exception Codes" in the section 6 of the document "MODBUS Application Protocol Specification V1.1b", [1 byte].

The size of the MODBUS PDU is limited by the size constraint inherited from the first MODBUS implementation on Serial Line network (max. RS-485 ADU = 256 bytes).
Therefore:
MODBUS **PDU for serial line communication** = 256 - Server address (1 byte) - CRC (2 bytes) = 253 bytes.
Consequently:
RS-232 / RS-485 **ADU** = 253 bytes + Server address (1 byte) + CRC (2 bytes) = **256 bytes**.
TCP MODBUS **ADU** = 253 bytes + MBAP (7 bytes) = **260 bytes**.

5.5 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 Protocol 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**;
- **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.

5.5.1 Implemented function codes

Lika MODBUS/TCP encoders implement public function codes, they are described hereafter.

03 Read Holding Registers

FC = 03 (0003 hex) ro

This function code is used to READ the contents of a contiguous block of Holding Registers (4X Reference Addresses) 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 is 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 the holding registers accessible using the **03 Read Holding Registers** function code please refer to the "6.1.1 Holding Register parameters" section on page 52.

Request PDU

Function code	1 byte	**0003 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	**0003 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	**0083 hex (=0003 hex + 0080 hex)**
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to read the **Preset value [105-106]** registers (address 104-105).

Request	
Field name	(Hex)
Function	**03**
Starting address Hi	**00**
Starting address Lo	**68**
No. of registers Hi	**00**
No. of registers Lo	**02**

Response	
Field name	(Hex)
Function	**03**
Byte count	**04**
Register 105 value Hi	**05**
Register 105 value Lo	**DC**
Register 106 value Hi	**00**
Register 106 value Lo	**00**

As you can see in the table, the **Preset value [105-106]** registers (address 104-105) contain the value 00 00 hex and 05 DC hex, i.e. 1,500 in decimal notation.

The MODBUS/TCP ADU needed for the request to read the **Preset value [105-106]** registers (address 104-105) is as follows:

MBAP Header + Request PDU (in hexadecimal notation)

[00][01][00][00][00][06][00][03][00][68][00][02]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[03] = **03 Read Holding Registers** function code
[00][68] = starting address (**Preset value [105-106]** registers, address 104-105)
[00][02] = number of requested registers

The MODBUS/TCP ADU needed to send back the values of the **Preset value [105-106]** registers (address 104-105) is as follows:

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][07][00][03][04][05][DC][00][00]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][07] = Length
[00] = Unit Identifier
[03] = **03 Read Holding Registers** function code

[04] = number of bytes (2 bytes for each register)
[05][DC] = value of register 105, 05 DC hex = 1,500
dec [00][00] = value of register 106, 00 00 hex = 0 dec

04 Read Input Registers

FC = 04 (0004 hex)

This function code is used to READ from 1 to 125 contiguous Input Registers (3X Reference Addresses) 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 the input registers accessible using the **04 Read Input Registers** function code please refer to the "6.1.2 Input Register parameters" section on page 70.

Request PDU

Function code	1 byte	**0004 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	**0004 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	**0084 hex (=0004 hex + 0080 hex)**
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to read the **Current position [1-2]** registers (address 0-1).

Request	
Field name	(Hex)
Function	**04**
Starting address Hi	**00**
Starting address Lo	**00**
Quantity of Input Reg. Hi	**00**
Quantity of Input Reg. Lo	**02**

Response	
Field name	(Hex)
Function	**04**
Byte count	**04**
Register 1 value Hi	**2F**
Register 1 value Lo	**F0**
Register 2 value Hi	**00**
Register 2 value Lo	**00**

As you can see in the table, the **Current position [1-2]** registers (address 0-1) contain the values 00 00 hex and 2F F0 hex, i.e. 12,272 in decimal notation.

The MODBUS/TCP ADU needed for the request to read the **Current position [1-2]** registers (address 0-1) is as follows:

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][04][00][00][00][02]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[04] = **04 Read Input Registers** function code
[00][00] = starting address (**Current position [1-2]** registers, address 0-1)
[00][02] = number of requested registers

The MODBUS/TCP ADU needed to send back the value of the **Current position [1-2]** registers (address 0-1) is as follows:

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][07][00][04][04][2F][F0][00][00]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][07] = Length
[00] = Unit Identifier
[04] = **04 Read Input Registers** function code
[04] = number of bytes (2 bytes for each register)
[2F][F0] = value of register 1, 2F F0 hex = 12,272 dec

[00][00] = value of register 2, 00 00 hex = 0 dec

06 Write Single Register

FC = 06 (0006 hex)

This function code is used to WRITE a single Holding Register (4X Reference Addresses) 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 the holding registers accessible using the **06 Write Single Register** function code please refer to the “6.1.1 Holding Register parameters” section on page 52.

Request PDU

Function code	1 byte	**0006 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	**0006 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	**0086 hex (=0006 hex + 0080 hex)**
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to write in the **Watchdog timeout [82]** register (address 81): you want to enable the Watchdog function and set the timeout to 10 ms. Please note that the **Watchdog timeout [82]** register is implemented but not used in this encoder. It is mentioned only as an example.

Request		**Response**	
Field name	(Hex)	Field name	(Hex)
Function	**06**	Function	**06**
Register address Hi	**00**	Register address Hi	**00**
Register address Lo	**51**	Register address Lo	**51**
Register value Hi	**00**	Register value Hi	**00**
Register value Lo	**0A**	Register value Lo	**0A**

As you can see in the table, the value 00 0A hex (10 dec) is set in the **Watchdog timeout [82]** register (address 81): the Watchdog function is enabled and the timeout is set to 10 ms.

The MODBUS/TCP ADU needed for the request to write the value 00 0A hex in the **Watchdog timeout [82]** register (address 81) is as follows:

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][06][00][51][00][0A]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[06] = **06 Write Single Register** function code
[00][51] = address of the **Watchdog timeout [82]** register, 51 hex = 81 dec
[00][0A] = value to be set in the register

The MODBUS/TCP ADU needed to send back a response following the request to write the value 00 0A hex in the **Watchdog timeout [82]** register (address 81) is as follows:

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][06][00][51][00][0A]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[06] = **06 Write Single Register** function code

[00][51] = address of the Watchdog timeout [82] register, 51 hex = 81 dec
[00][0A] = value set in the register

16 Write Multiple Registers

FC = 16 (0010 hex)

This function code is used to WRITE a block of contiguous Holding Registers (4X Reference Addresses, 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 the holding registers accessible using the **16 Write Multiple Registers** function code please refer to the "6.1.1 Holding Register parameters" section on page 52.

Request PDU

Function code	1 byte	**0010 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	**0010 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	**0090 hex (= 0010 hex + 0080 hex)**
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to write the value 00 00 08 00 hex (=2,048 dec) next to the **Counts per revolution [101-102]** registers (address 100-101) and the value 00 80 00 00 hex (=8,388,608 dec) next to the **Total Resolution [103-104]** registers (address 102-103).

Request	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**64**
Quantity of registers Hi	**00**
Quantity of registers Lo	**04**
Byte count	**08**
Register 101 value Hi	**08**
Register 101 value Lo	**00**
Register 102 value Hi	**08**
Register 102 value Lo	**00**
Register 103 value Hi	**00**
Register 103 value Lo	**00**
Register 104 value Hi	**00**
Register 104 value Lo	**80**

Response	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**64**
Quantity of registers Hi	**00**
Quantity of registers Lo	**04**

As you can see in the table, the values 00 00 hex and 08 00 hex, i.e. 2,048 in decimal notation, are set in the **Counts per revolution [101-102]** registers at address 100-101; while the values 00 80 hex and 00 00 hex, i.e. 8,388,608 in decimal notation, are set in the **Total Resolution [103-104]** registers at the address 102-103. Thus the encoder will be programmed to have a 2,048-count-per-revolution singleturn resolution and 4,096 revolutions (8,388,608/2,048).

The MODBUS/TCP ADU needed for the request to write the value 2,048 dec next to the **Counts per revolution [101-102]** registers (address 100-101) and the value 8,388,608 dec next to the **Total Resolution [103-104]** registers (address 102-103) is as follows:

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][0F][00][10][00][64][00][04][08][08][00][00][00][00][00][00][80]
where:
[00][01] = Transaction Identifier

[00][00] = Protocol Identifier
[00][0F] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][64] = starting address (**Counts per revolution [101-102]** registers, address 100-101)
[00][04] = number of requested registers
[08] = number of bytes (2 bytes for each register)
[08][00] = value to be set in the register 101, 08 00 hex
[00][00] = value to be set in the register 102, 00 00 hex (00 00 08 00 hex = 2,048 dec)
[00][00] = value to be set in the register 103, 00 00 hex
[00][80] = value to be set in the register 104, 00 80 hex (00 80 00 00 hex = 8,388,608 dec)

The MODBUS/TCP ADU needed to send back a response following the request to write the value 2,048 next to the **Counts per revolution [101-102]** registers (address 100-101) and the value 8,388,608 next to the **Total Resolution [103-104]** registers (address 102-103) is as follows:

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][10][00][64][00][04]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][64] = starting address (**Counts per revolution [101-102]** registers, address 100-101)
[00][04] = number of written registers

Here is an example of a request to write in the **Operating parameters [109-110]** registers (address 108-109): we need to set the scaling function (bit 0 **Scaling function** = 1) and the count up information with clockwise rotation of the encoder shaft (bit 1 **Code sequence** = 0).

Request	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**6C**
Quantity of registers Hi	**00**
Quantity of registers Lo	**02**
Byte count	**04**
Register 109 value Hi	**00**
Register 109 value Lo	**01**
Register 110 value Hi	**00**
Register 110 value Lo	**00**

Response	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**6C**
Quantity of registers Hi	**00**
Quantity of registers Lo	**02**

As you can see in the table, the value 00 00 00 01 hex, i.e. 0000 0000 0000 0000 0000 0000 0000 0001 in binary notation, is set in the **Operating parameters [109-110]** registers (address 108-109): the bit 0 **Scaling function** = 1; the bit 1 **Code sequence** = 0; the remaining bits are not used, therefore their value is 0.

The MODBUS/TCP ADU needed for the request to set the scaling function (bit 0 **Scaling function** = 1) and the count up information with clockwise rotation of the encoder shaft (bit 1 **Code sequence** = 0) in the **Operating parameters [109-110]** registers (address 108-109) is as follows:

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][0B][00][10][00][6C][00][02][04][00][01][00][00]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][0B] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][6C] = starting address (**Operating parameters [109-110]** registers, address 108-109)
[00][02] = number of requested registers

[04] = number of bytes (2 bytes for each register)
[00][01] = value to be set in the register 109, 00 01 hex
[00][00] = value to be set in the register 110, 00 00 hex

The MODBUS/TCP ADU needed to send back a response following the request to set the scaling function (bit 0 **Scaling function** = 1) and the count up information with clockwise rotation of the encoder shaft (bit 1 **Code sequence** = 0) in the **Operating parameters [109-110]** registers (address 108-109) is as follows:

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][10][00][6C][00][02]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][6C] = starting address (**Operating parameters [109-110]** registers, address 108-109)
[00][02] = number of written registers

5.6 Encoder states

The table below describes the states the encoder can enter during operation in the MODBUS/TCP network.

Encoder state	Description
WAIT_PROCESS	Waiting for MODBUS requests. The encoder shifts to the PROCESS_ACTIVE state as soon as a MODBUS request is received.
ERROR	An IP address conflict has been detected in the MODBUS network. The NS Network Status LED starts flashing red (see on page 27).
PROCESS_ACTIVE	The encoder shifts to the WAIT_PROCESS state if no requests are received within the preset time.
EXCEPTION	A Watchdog timeout has occurred, any MODBUS requests will be ignored. The NS Network Status LED starts flashing red (see on page 27).

6 Programming parameters

6.1 Parameters available

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

Parameter name [Register number]
[register address, data type, attribute]

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

The MODBUS registers are 16-bit long; while all the encoder parameters -except the **Watchdog timeout [82]** parameter- are 2-register long, i.e. 32-bit long (independently of their size, whether they are 32-bit long or 16-bit long). Please note that the **Watchdog timeout [82]** register is implemented but not used in this encoder.

word	MSW			LSW		
bit	31	...	16	15	...	0
	msb		lsb	msb		lsb

6.1.1 Holding Register parameters

Holding registers (**Machine data** parameters) are 4X Reference Registers and accessible for both writing and reading; to read the value set in the parameter use the **03 Read Holding Registers** function code (reading of multiple registers); to write a value in the 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 “5.5.1 Implemented function codes“ section on page 40.

NOTE
Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the bit 9 **Save parameters** command available in the **Control Word [111-112]** registers, see on page 61.
Should the power supply be turned off all data that has not been saved previously will be lost!

Watchdog timeout [82]

[81, Unsigned16, rw]

This register is implemented but not used in this encoder.

Current position [95-96]

[94-95, Unsigned32, ro]

The **Current position [1-2]** input registers are also available as holding registers at the address 94-95 and accessible by using the **03 Read Holding Registers** function code. For any information refer to page 70.

Speed value [97-98]

[96-97, Signed32, ro]

The **Speed value [3-4]** input registers are also available as holding registers at the address 96-97 and accessible by using the **03 Read Holding Registers** function code. For any information refer to page 70.

Status word [99-100]

[98-99, Unsigned32, ro]

The **Status word [5-6]** input registers are also available as holding registers at the address 98-99 and accessible by using the **03 Read Holding Registers** function code. For any information refer to page 70.

Counts per revolution [101-102]

[100-101, Unsigned32, rw]

WARNING

These registers are active only if the bit 0 **Scaling function** in the **Operating parameters [109-110]** registers is set to "=1"; otherwise they are ignored and the system uses the physical values (see the **Singleturn resolution [113-114]** and **Number of revolutions [115-116]** registers) to calculate the position information.

These registers set the custom number of distinguishable steps per revolution that are output for the absolute singleturn position value.

To avoid counting errors, please check that:

$$\frac{\text{Singleturn resolution [113-114]}}{\text{Counts per revolution [101-102]}} = \text{integer value.}$$

You are allowed to set whatever integer value less than or equal to the **maximum number of physical steps per revolution** (see the **Singleturn resolution [113-114]** registers).
If you enter an out-of-range value (i.e. greater than the maximum number of physical steps per revolution), the number of measuring units per revolution is forced to the physical singleturn resolution and the **Alarm registers [121-122]** registers (see the bit 12 **Machine data not valid**) as well as the **Wrong parameters list [125-126]** registers (see the bit 0 **Counts per revolution error**) signal the error.

WARNING
To avoid counting errors please always make sure that the following condition is met:

$$\frac{\text{Total Resolution [103-104]}}{\text{Counts per revolution [101-102]}} = \text{a power of 2.}$$

Furthermore, after having set a new value next to the **Counts per revolution [101-102]** registers, make sure that also the following condition is met:

$$\frac{\text{Total Resolution [103-104]}}{\text{Counts per revolution [101-102]}} \leq \text{Number of physical revolutions (see Number of revolutions [115-116])}$$

Let's suppose that the encoder is programmed as follows:
Counts per revolution [101-102]: 8,192 cpr
Total Resolution [103-104] = 33,554,432 = 8,192 (cpr) * 4,096 (rev.)
Let's set a new singleturn resolution, for instance: **Counts per revolution [101-102]** = 360.

If we do not change the **Total Resolution [103-104]** value at the same time, we will get the following result:

$$\text{Number of revolutions} = \frac{\text{33,554,432 (Total Resolution [103-104])}}{\text{360 (Counts per revolution [101-102])}} = 93{,}206.755...$$

As you can see, the encoder is required to carry out more than 93,000 revolutions, this cannot be because the hardware number of revolutions can be max. 16,384 (see the **Singleturn resolution [113-114]** registers). When this happens, the encoder falls into an error signalling the fault condition (see the

Alarm registers [121-122] and the Wrong parameters list [125-126] registers).

WARNING
When you enable the scaling function (bit 0 **Scaling function** in the Operating parameters [109-110] registers = 1), please enter scaled values next to the Counts per revolution [101-102] and Total Resolution [103-104] registers that are consistent with the physical values. In the case of inconsistent values, the system will warn about the wrong parametrization and fault condition by means of the dedicated registers.

WARNING
If you have set the preset, every time you change the value next to the Counts per revolution [101-102] registers, then you are required to activate a new preset (see the bit 11 **Perform counting preset** in the Control Word [111-112] registers = 1).

Total Resolution [103-104]

[102-103, Unsigned32, rw]

WARNING
These registers are active only if the bit 0 **Scaling function** in the Operating parameters [109-110] registers is set to "=1"; otherwise they are ignored and the system uses the physical values (see the Singleturn resolution [113-114] and Number of revolutions [115-116] registers) to calculate the position information.

These registers are intended to set a custom number of distinguishable steps over the total measuring range (overall resolution of the encoder). The total resolution of the encoder results from the product of Counts per revolution [101-102] by the **required number of revolutions**.

You are allowed to set whatever integer value less than or equal to the **overall hardware resolution** (see the encoder identification label as well as the Singleturn resolution [113-114] and Number of revolutions [115-116] registers). The overall hardware resolution results from:
Singleturn resolution [113-114] * Number of revolutions [115-116]
If you set an out-of-range value (i.e. greater than the overall hardware resolution), the total resolution is forced to the physical total resolution and the Alarm registers [121-122] registers (see the bit 12 **Machine data not valid**) as well as the Wrong parameters list [125-126] registers (see the bit 1 **Total resolution error**) signal the error.

Default = 134,217,728 (min. = 1, max. = 134,217,728)

WARNING

To avoid counting errors please always make sure that the following condition is met:

$$\frac{\text{Total Resolution [103-104]}}{\text{Counts per revolution [101-102]}} = \text{a power of 2.}$$

Setting the **Number of revolutions** to a value which is a power of 2 is meant to avoid problems when using the device in endless operations requiring the physical zero to be overstepped. If you set the **Number of revolutions** which is not a power of 2, a counting error is generated before the physical zero.

Furthermore, after having set a new value next to the Total Resolution [103-104] registers, always check also the Counts per revolution [101-102] registers and make sure that the following condition is met:

$$\frac{\text{Total Resolution [103-104]}}{\text{Counts per revolution [101-102]}} \leq \text{Number of physical revolutions (see Number of revolutions [115-116])}$$

Let's suppose that the encoder is programmed as follows:
Counts per revolution [101-102]: 8,192 cpr
Total Resolution [103-104] = 33,554,432 = 8,192 (cpr) * 4,096 (rev.)
Let's set a new total resolution, for instance: Total Resolution [103-104] = 360.
As the Total Resolution [103-104] must be greater than or equal to the Counts per revolution [101-102], the above setting is not allowed.

WARNING

If you have set the preset, every time you change the value next to the Total Resolution [103-104] registers, then you are required to activate a new preset (bit 11 **Perform counting preset** in the Control Word [111-112] registers = 1).

EXAMPLE

We install the multiturn encoder. Its physical resolution is

as follows (see the ordering code):

- **Hardware counts per revolution**: **Singleturn resolution [113-114]** = 8192 cpr (2^{13})
- **Hardware number of revolutions**: **Number of revolutions [115-116]** = 16,384 turns (2^{14})
- **Total hardware resolution**: **Singleturn resolution [113-114]** ∗ **Number of revolutions [115-116]** = 134,217,728 ($2^{13+14} = 2^{27}$)

In the specific installation **2,048 counts/rev.** * **1,024 turns** are required:

- Enable the scaling function: **Operating parameters [109-110]**, bit 0 **Scaling function** = "1"
- Counts per revolution: **Counts per revolution [101-102]** = 2,048 (0000 0800 hex)
- Total resolution: **Total Resolution [103-104]** = 2,048 ∗ 1,024 = 2,097,152 (0020 0000 hex)

NOTE

We suggest setting values which are a power of 2 (2^n: 2, 4, …, 2048, 4096, 8192, …) to be set in the **Counts per revolution [101-102]** and **Total Resolution [103-104]** registers to avoid counting errors.

WARNING

If **Counts per revolution [101-102]** and/or **Total Resolution [103-104]** values change, the **Preset value [105-106]** must be updated in accordance with the new resolution. A new preset operation is required.

Preset value [105-106]

[104-105, Unsigned32, rw]

These registers allow to set the encoder position to a Preset value. The Preset function is meant to assign a desired value to a physical position of the encoder shaft. The chosen physical position will get the value set next to these registers and all the previous and following positions will get a value according to it. This function is useful, for example, when the zero position of the encoder and the zero position of the axis need to match. The preset value will be set for the position of the encoder in the moment when the **Perform counting preset** command available in the **Control Word [111-112]** registers is sent. We suggest activating the preset value when the encoder is in stop.

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 [105-106]**, **Offset value [127-128]** 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 [105-106]** - **Offset value [127-128]**.

If you never set the **Preset value [105-106]** and you never performed the preset setting (**Perform counting preset** command in the **Control Word [111-112]**), then the transmitted value and the read position are necessarily the same as **Preset value [105-106]** = 0 and **Offset value [127-128]** = 0.

When you set the **Preset value [105-106]** and then execute the **Perform counting preset** command, the system saves the current encoder position in the **Offset value [127-128]** registers. It follows that the transmitted value and the **Preset value [105-106]** are the same as **read position** - **Offset value [127-128]** = 0; in other words, the value set next to the **Preset value [105-106]** registers 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 [105-106]** registers 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:

Transmitted value = **read position** (="1000") + **Preset value [105-106]** (="50") - **Offset value [127-128]** (="1000") = **50**.

The following transmitted value will be:

Transmitted value = **read position** (="1001") + **Preset value [105-106]** (="50") - **Offset value [127-128]** (="1000") = **51**.

And so on.

NOTE

- If the **Scaling function** is disabled (the bit 0 in the **Operating parameters [109-110]** registers = 0), then the **Preset value [105-106]** must be less than or equal to the "Total hardware resolution" - 1, i.e. (**Singleturn resolution [113-114]** * **Number of revolutions [115-116]**) - 1.

- If the **Scaling function** is enabled (the bit 0 in the **Operating parameters [109-110]** registers = 1), then the **Preset value [105-106]** must be less than or equal to **Total Resolution [103-104]** - 1.

WARNING

Check the value in the **Preset value [105-106]** registers and perform the preset operation if necessary (the bit 11 **Perform counting preset** in the **Control Word [111-112]** registers = 1) every time you set a new **Code sequence** or enable the **Scaling function** or change the scaled values (**Counts per revolution [101-102]** and / or **Total Resolution [103-104]** registers).

Speed format [107-108]

[106-107, Unsigned32, rw]

These registers set the engineering unit for the velocity value (see the **Speed value [3-4]** registers on page 70).

0 = steps/s: number of steps per second;
1 = rpm: revolutions per minute.
Default = 0 (min. = 0, max. = 1)

Operating parameters [109-110]

[108-109, Unsigned32, rw]

Bit	Function	bit = 0	bit = 1
0	**Scaling function**	**disabled**	enabled
1	**Code sequence**	**Count up with clockwise (CW) rotation**	Count up with counter clockwise (CCW) rotation
2 … 31	not used		

Default values are highlighted in bold

Default = 0000 0000 hex (min. = 0000 0000 hex, max. = 0000 0003 hex)

Byte 0

Scaling function

bit 0 If this function is disabled (the bit 0 **Scaling function** = 0), the devices uses the physical resolution to calculate the absolute position value (see the **Singleturn resolution [113-114]** and **Number of revolutions [115-116]** registers); if this function is enabled (the bit 0 **Scaling function** = 1), the device uses the custom resolution set next to the **Counts per revolution [101-102]** and **Total Resolution [103-104]** registers in compliance with the following relation:

$$\text{Transmitted position} = \frac{\text{Counts per revolution [101-102]}}{\text{Singleturn resolution [113-114]}} * \text{Real position} \leq \text{Total Resolution [103-104]}$$

NOTE

To know whether the bit 0 **Scaling function** is currently enabled, you can read the bit 0 **Scaling function** of the **Status word [5-6]** input registers, see on page 71.

WARNING

Every time you enable/disable the scaling function and/or change the scaling values (see the **Counts per revolution [101-102]** and **Total Resolution [103-104]** registers), then you are required to activate a new preset value (see the **Preset value [105-106]** registers) and finally save the new parameters (see the bit 9 **Save parameters** command in the **Control Word [111-112]** registers).

Code sequence

bit 1

It sets whether the position value output by the encoder increases (count up information) when the encoder shaft rotates clockwise (0 = CW) or counter-clockwise (1 = CCW). If the bit 1 **Code sequence** = 0, the absolute position value **increases** when the encoder shaft rotates **clockwise**; on the contrary, if the bit 1 **Code sequence** = 1, the absolute position value **increases** when the encoder shaft rotates **counter-clockwise**. CW and CCW rotations are viewed from the shaft end.

WARNING

Changing this value causes also the position calculated by the controller to be necessarily affected. Therefore it is mandatory to execute a new preset (see the **Preset value [105-106]** registers) and save the parameters after setting this item.

NOTE

To know whether the bit 1 **Code sequence** is currently enabled, you can read the bit 1 **Code sequence** of the **Status word [5-6]** input registers, see on page 71.

bits 2 … 7 — Not used.

Bytes 1 … 3 — Not used.

Control Word [111-112]

[110-111, Unsigned32, rw]
This variable contains the commands to be sent in real time to the Slave in order to manage it.

Bit	Function	bit = 0	bit = 1
0 … 8	not used		
9	**Save parameters**		
10	**Restore default parameters**		
11	**Perform counting preset**		
12 … 31	not used		

Byte 0 Not used.

Byte 1

bit 8 Not used.

Save parameters
bit 9 This function allows to save all parameters on non-volatile memory. Data is saved on non-volatile memory at each rising edge of the bit; in other words, data save is performed each time this bit is switched from logic level low ("0") to logic level high ("1"). Then the bit must be switched back to logic level low ("0") to make the function available again.

NOTE
Always save the new values after setting in order to store them in the non-volatile memory permanently.
Should the power supply be turned off all data that has not been saved previously will be lost!

Restore default parameters
bit 10 This function allows the operator to restore all parameters to default values (default values are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode). This function can be useful, for instance, to restore the factory values in case the encoder is set incorrectly and you are not able to resume the proper operation.
Default parameters are restored at each rising edge of the 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"). Then the bit

must be switched back to logic level low ("0") to make the function available again. The complete list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 99.

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

Perform counting preset

bit 11 — This command is used to activate a preset value in the encoder. As soon as the command is sent, the position value which is transmitted for the current encoder position is the one set next to the **Preset value [105-106]** registers and all the previous and following positions will get a value according to it. The operation is performed at each rising edge of the bit, i.e. each time this bit is switched from logic level low ("0") to logic level high ("1"). Then the bit must be switched back to logic level low ("0") to make the function available again. When the command is sent, the current encoder position is saved temporarily in the **Offset value [127-128]** registers. For any further information on the preset function and the meaning and use of the related registers and commands **Preset value [105-106]**, **Offset value [127-128]** and **Perform counting preset** refer to page 57.

WARNING
To save permanently the current encoder position in the **Offset value [127-128]** registers, please execute the bit 9 **Save parameters** command. Should the power supply be turned off without saving data, the **Offset value [127-128]** that has not been saved will be lost!

bits 12 … 15 — Not used.

Bytes 2 and 3 — Not used.

NOTE
Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the bit 9 **Save parameters** function, see on page 61.
Should the power supply be turned off all data that has not been saved previously will be lost!

Singleturn resolution [113-114]

[112-113, Unsigned32, ro]

WARNING

These registers are active only if the bit 0 **Scaling function** in the **Operating parameters [109-110]** registers is set to "=0"; otherwise they are ignored and the system uses the custom resolution values (**Counts per revolution [101-102]** and **Total Resolution [103-104]**) to calculate the position information.

These registers are intended to show the number of physical distinguishable steps provided per each turn by the hardware (physical singleturn resolution, see the hardware counts per revolution in the encoder identification label).

If you want to set a custom singleturn resolution see the **Counts per revolution [101-102]** registers.

Number of revolutions [115-116]

[114-115, Unsigned32, ro]

WARNING

These registers are active only if the bit 0 **Scaling function** in the **Operating parameters [109-110]** registers is set to "=0"; otherwise they are ignored and the system uses the custom resolution values (**Counts per revolution [101-102]** and **Total Resolution [103-104]**) to calculate the position information.

These registers are intended to show the number of physical distinguishable turns provided by the hardware (number of physical revolutions, see the hardware revolutions in the encoder identification label).

The **Total hardware resolution** results from **Singleturn resolution [113-114]** * **Number of revolutions [115-116]**.

If you want to set a custom number of turns see the **Counts per revolution [101-102]** and **Total Resolution [103-104]** registers.

Supported alarms [117-118]

[116-117, Unsigned32, ro]

Bit	Function	bit = 0	bit = 1
0 … 10	not used		
11	**Position error**	Alarm not supported	**Alarm supported**
12	**Machine data not valid**	Alarm not supported	**Alarm supported**
13	**Setting data not valid**	Alarm not supported	**Alarm supported**
14	**Flash memory error**	Alarm not supported	**Alarm supported**
15 … 31	not used		

These registers contain the information on the alarms supported by the encoder. The available alarm messages are described in the **Alarm registers [121-122]** item.

The supported alarms are listed here afterwards:

Byte 0 Not used.

Byte 1

bits 8 … 10 Not used.

Position error
bit 11

Machine data not valid
bit 12

Setting data not valid
bit 13

Flash memory error
bit 14

bit 15 Not used.

Bytes 2 and 3 Not used.

Default = 0000 7800h (= 0000 0000 0000 0000 0111 1000 0000 0000 = alarms at bits 11, 12, 13, and 14 of the **Alarm registers [121-122]** item are supported).

Supported warnings [119-120]

[118-119, Unsigned32, ro]

Bit	Function	bit = 0	bit = 1
0	**Position Warning**	Warning not supported	**Warning supported**
1 … 31	not used		

These registers contain the information on the warnings supported by the encoder. The available warning messages are described in the **Warning registers [123-124]** item.

The supported warnings are listed here afterwards:

Byte 0

Position Warning
bit 0

bits 1 … 7 Not used.

Bytes 1 … 3 Not used.

Default = 0000 0001h (= 0000 0000 0000 0000 0000 0000 0000 0001 = warning at bit 0 of the **Warning registers [123-124]** item is supported).

Alarm registers [121-122]

[120-121, Unsigned32, ro]

Bit	Function	bit = 0	bit = 1
0 … 10	not used		
11	**Position error**	Alarm not active	Alarm active
12	**Machine data not valid**	Alarm not active	Alarm active
13	**Setting data not valid**	Alarm not active	Alarm active
14	**Flash memory error**	Alarm not active	Alarm active
15 … 31	not used		

These registers are meant to show the alarms currently active in the device. An alarm will be set if a malfunction in the encoder could lead to incorrect position value. If an alarm occurs, the according bit is set to logical high (1) until the alarm is cleared and the encoder is able to provide an accurate position value.
The available alarm messages are described here afterwards.
Refer also to the bit 8 **Alarm** in the **Status word [5-6]** registers, see on page 71.

Byte 0 Not used.

Byte 1

bits 8 … 10 Not used.

Position error

bit 11 Fault and malfunction of the encoder position measurement system or the measured value processing unit. This error causes invalid position and speed actual values, it may be due to the hardware or the signal quality.

Machine data not valid

bit 12 One or more parameters are not valid, an out-of-tolerance parameter has been set. Set proper values to restore the normal work condition. For more details about the specific out-of-tolerance parameter see the list of the wrong parameters in the **Wrong parameters list [125-126]** registers.

Setting data not valid

bit 13 This alarm message is currently disabled in this firmware version.

Flash memory error

bit 14 Flash memory internal error, it cannot be restored (bad checksum error, etc.). The flash memory contains corrupted data; or maybe the flash memory is damaged.

bit 15 Not used.

Bytes 2 and 3 Not used.

NOTE

Please note that should the alarm be caused by wrong parameter values (see **Machine data not valid** and **Wrong parameters list [125-126]** registers), the normal work status can be restored only after having set proper values. The **Flash memory error** alarm cannot be reset.

Warning registers [123-124]

[122-123, Unsigned32, ro]

Bit	Function	bit = 0	bit = 1
0	Position Warning	Warning not active	Warning active
1 … 31	not used		

These registers are meant to show the warnings currently active in the device. A warning will be set if the malfunction in the encoder does not lead to incorrect position value. If a warning occurs, the according bit is set to logical high (1) until the warning is cleared and the normal operation of the encoder is resumed.
The available warning messages are described here afterwards.

Byte 0

Position Warning
bit 0
Fault and malfunction of the encoder position measurement system or the measured value processing unit. This warning does not cause an invalid position and speed actual value, it may be due to the hardware or the signal quality.

bits 1 … 7 Not used.

Bytes 1 … 3 Not used.

Wrong parameters list [125-126]

[124-125, Unsigned32, ro]
The operator has entered invalid data and the **Machine data not valid** alarm in the **Alarm registers [121-122]** has been triggered. This variable is meant to show (bit value = HIGH) the list of the wrong parameters, according to the following table.
Please note that the normal work status can be restored only after having set proper values.

Bit	Function	bit = 0	bit = 1
0	Counts per revolution error	Alarm not active	Alarm active
1	Total resolution error	Alarm not active	Alarm active
2	Preset value error	Alarm not active	Alarm active
3	Offset value error	Alarm not active	Alarm active
4 … 31	not used		

Byte 0

Counts per revolution error

bit 0 Wrong data has been set next to the **Counts per revolution [101-102]** registers. The tolerances for the parameter have been exceeded. Set proper values to restore the normal work condition. The alarm is cleared if the tolerances are within normal parameters again.

Total resolution error

bit 1 Wrong data has been set next to the **Total Resolution [103-104]** registers. The tolerances for the parameter have been exceeded. Set proper values to restore the normal work condition. The alarm is cleared if the tolerances are within normal parameters again.

Preset value error

bit 2 Wrong data has been set next to the **Preset value [105-106]** registers. The tolerances for the parameter have been exceeded. Set proper values to restore the normal work condition. The alarm is cleared if the tolerances are within normal parameters again.

Offset value error

bit 3 Wrong data has been set next to the **Preset value [105-106]** registers and the calculated **Offset value [127-128]** is out-of-tolerance. The tolerances for the parameter have been exceeded. Save proper values to restore the normal work condition. The alarm is cleared if the tolerances are within normal parameters again.

bits 4 … 7 Not used.

Bytes 1 ... 3 **Not used.**

Offset value [127-128]

[126-127, Signed32, ro]

These registers contain the Offset value. As soon as you activate the preset by sending the **Perform counting preset** command (see the bit 11 in the **Control Word [111-112]** registers), the current position of the encoder is saved in these registers. 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 these registers you must upload the factory default values (see the bit 10, **Restore default parameters** command, in the **Control Word [111-112]** registers on page 61).

For any further information on the preset function and the meaning and use of the related registers and commands **Preset value [105-106]**, **Offset value [127-128]** and **Perform counting preset** refer to page 57.
Default = 0

Software revision [129-130]

[128-129, Unsigned32, ro]
These registers are meant to show the current version of the firmware installed in the encoder.
The meaning of the 32 bits in the registers is as follows:

Word	**MS Word**			**LS Word**		
bit	31	…	16	15	…	0
	msb		lsb	msb		Lsb
	Major version			Minor version		

For example, the value 0001 0001 hex in hexadecimal notation corresponds to the binary representation 0000 0000 0000 0001 0000 0000 0000 0001 and has to be interpreted as: firmware version 1.1 .

Hardware revision [131-132]

[130-131, Unsigned32, ro]
These registers are meant to show the current version of the hardware installed in the encoder.
The meaning of the 32 bits in the registers is as follows:

Word	**MS Word**			**LS Word**		
bit	31	…	16	15	…	0
	msb		lsb	msb		Lsb
	Major version			Minor version		

For example, the value 0001 0000 hex in hexadecimal notation corresponds to the binary representation 0000 0000 0000 0001 0000 0000 0000 0000 and has to be interpreted as: hardware version 1.0 .

6.1.2 Input Register parameters

Input Registers are 3X Reference Registers and accessible for reading only; to read the value set in an input register parameter use the **04 Read Input Registers** function code (reading of multiple input registers); for any further information on the implemented function codes refer to the "5.5.1 Implemented function codes" section on page 40.

Current position [1-2]

[000-001, Unsigned32, ro]

These registers are meant to show the current position of the device at the time the request is sent. The output value is scaled according to the set scaling parameters, see the bit 0 **Scaling function** in the **Operating parameters [109-110]** registers, page 61. The value is expressed in counts.

The **Current position [1-2]** input registers are also available as holding registers at the address 94-95 and accessible by using the **03 Read Holding Registers** function code. For any information refer to page 53.

Speed value [3-4]

[002-003, Signed32, ro]

These registers show the current output speed value detected by the position encoder and calculated every 100 ms.

The value can be expressed in either steps per second or revolutions per minute according to the setting next the **Speed format [107-108]** registers on page 59.

The **Speed value [3-4]** input registers are also available as holding registers at the address 96-97 and accessible by using the **03 Read Holding Registers** function code. For any information refer to page 53.

Status word [5-6]

[004-005, Unsigned32, ro]

These registers contain the information about the current state of the device. The eight bits of the Byte 0 show the values currently set in the **Operating parameters [109-110]** registers; while the eight bits of the Byte 1 are used to signal if any alarm is active. Bytes 2 and 3 are not used.

Structure of the **Status word [5-6]** registers:

Word	**MS Word**			**LS Word**		
bit	31	…	16	15	…	0
	msb		lsb	msb		Lsb

Byte 0

Scaling function

bit 0

It shows whether the scaling function (see the bit 0 **Scaling function** of the **Operating parameters [109-110]** registers) is currently disabled or enabled. If the value is "=0" the scaling function is disabled (i.e. the system uses the physical values -**Singleturn resolution [113-114]** and **Number of revolutions [115-116]**- to calculate the position information); if the value is "=1" the scaling function is enabled (i.e. the system uses the custom resolution values – **Counts per revolution [101-102]** and **Total Resolution [103-104]**- to calculate the position information). To disable / enable the scaling function you must set the bit 0 **Scaling function** of the **Operating parameters [109-110]** registers to 0 / 1. For any further information on setting and using the scaling function refer to the **Scaling function** parameter on page 59.

Code sequence

bit 1

It shows whether the code sequence (see the bit 1 **Code sequence** of the **Operating parameters [109-110]** registers) is currently set to clockwise (CW) or counter-clockwise (CCW). If the bit is "=0" the output encoder position value has been set to increase (count up information) when the encoder shaft rotates clockwise; if the bit is "=1" the output encoder position value has been set to increase when the encoder shaft rotates counter-clockwise. CW and CCW rotations are viewed from the shaft end. To set the code sequence to either CW or CCW you must set the bit 1 **Code sequence** of the **Operating parameters [109-110]** registers to 0 / 1. For any further information on setting and using the counting direction function refer to the **Code sequence** parameter on page 60.

bits 2 … 7

Not used.

Byte 1

Alarm

bit 8

If the value is "=1" an alarm has occurred, see details in the **Alarm registers [121-122]** variable on page 65.

bits 9 … 15

Not used.

Bytes 2 and 3 Not used.

NOTE

The **Status word [5-6]** input registers are also available as holding registers at the address 98-99 and accessible by using the **03 Read Holding Registers** function code. For any information refer to page 53.

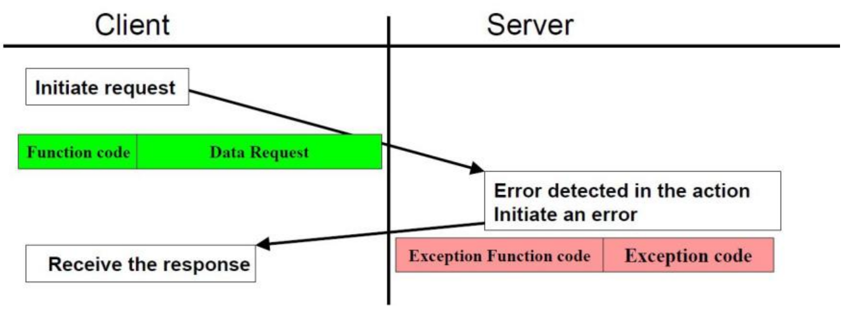
6.2 Exception response and codes

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, 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 45 of the “MODBUS Application Protocol Specification V1.1b3” document.

7 Integrated web server

7.1 Integrated web server – Preliminary information

MODBUS/TCP encoders from Lika Electronic integrate a web server. This web-based user interface is designed to offer helpful functions and deliver complete information on the device that can be accessed through the Internet.
In particular it allows:

- to display the current position and speed values;
- to set some parameters such as the preset and the code sequence;
- to display and check the parameters set currently;
- to monitor the encoder;
- to set the network communication parameters;
- to upgrade the firmware.

The web server can be accessed from any PC running a web browser. Since its only requirement is a HTTP connection between the web browser and the web server running on the device, it is perfectly fitted also for remote access scenarios.

Before opening the MODBUS/TCP encoder web server please ascertain that the following requirements are fully satisfied:

- the encoder is connected to the network;
- the encoder has valid IP address;
- the PC is connected to the network;
- a web browser (Internet Explorer, Mozilla Firefox, Google Chrome, Opera, …) is installed in the PC or in the device used for connection.

NOTE
This web server has been tested and verified using the following web browsers:

- Internet Explorer IE11 version 11.1593.14393.0
- Mozilla Firefox version 116.0.1
- Google Chrome version 115.0.5790.111
- Opera version 68.0.3618.165

NOTE
Please note that the appearance of the snapshots may vary depending on the web browser used. The following snapshots were taken from Google Chrome.

7.2 Web server Home page

To open the MODBUS/TCP encoder web server proceed as follows:

1. type the IP address of the encoder you want to connect to (in the example: 192.168.1.10, this is the default IP address set at Lika, see on page 25) in the address bar of your web browser and confirm by pressing **ENTER**;

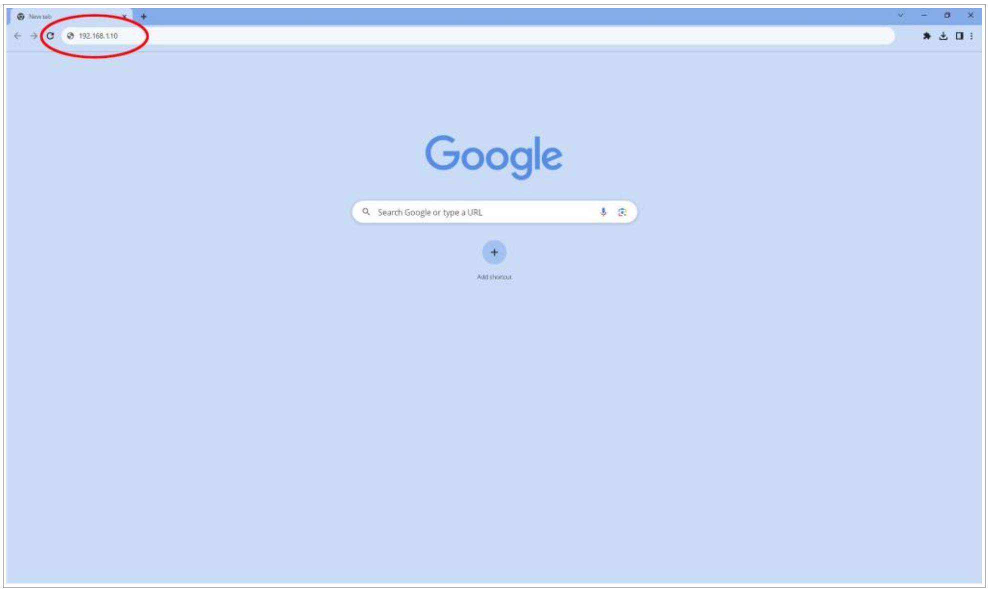


Figure 3 - Opening the web server

2. as soon as the connection is established, the web server **Home** page will appear on the screen;

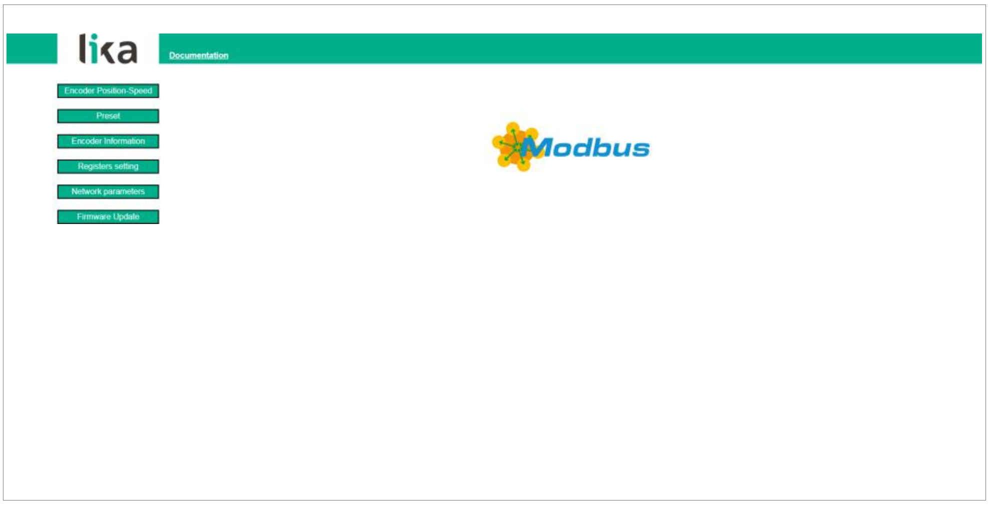


Figure 4 - Web server Home page

Some commands are available in the menu bar of the **Home** page.

Press on the **Lika logo** to enter Lika's web site (www.lika.biz).

Press the **DOCUMENTATION** button to enter the MODBUS/TCP encoder technical documentation page available on Lika's web site (https://www.lika.it/eng/products/rotary-encoders/absolute/ethernet/) where specific technical information and documentation concerning the MODBUS/TCP encoder can be found.

Furthermore some buttons are available in the left navigation bar. All the pages except the **Firmware Update** page are freely accessible through the buttons in the bar. The **Firmware Update** page is protected and requires a password.
These buttons allow to enter specific pages where information and diagnostics on the connected encoder as well as useful functions can be achieved.
They are described in the following sections.

7.3 Encoder position and speed

Press the **ENCODER POSITION-SPEED** button in the left navigation bar of the Web server **Home** page to enter the page where the current encoder position and the current encoder speed are displayed.

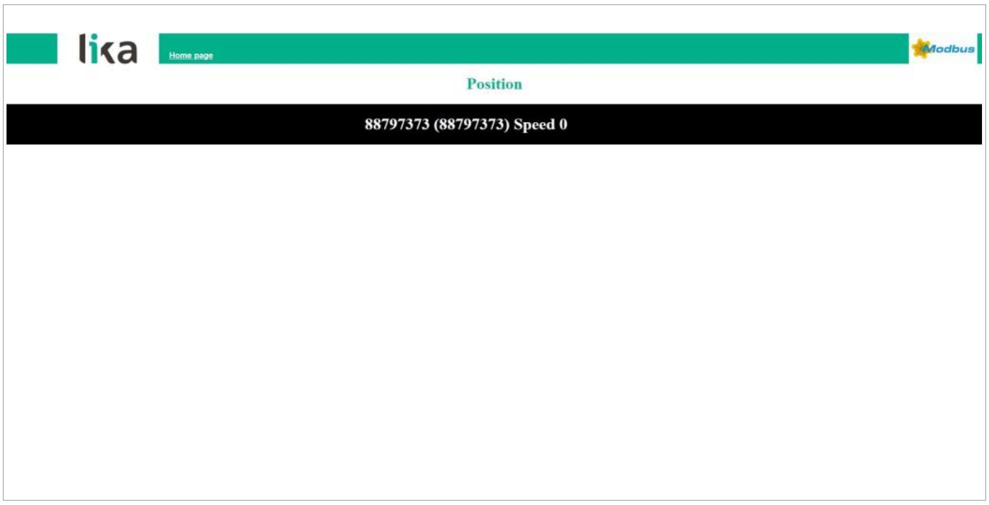


Figure 5 - Encoder position and speed page

The first value (under the Position item) is the absolute position calculated considering scaling and preset functions, if activated; the value in brackets is the raw value (physical absolute position). Both encoder positions are expressed in counts. For any information refer to the **Current position [1-2]** registers on page 68.

The current encoder speed (next to the Speed item) is expressed according to the setting next the **Speed format [107-108]** registers on page 57 (by default it is expressed in counts per second). For any information refer to the **Speed value [3-4]** registers on page 68.

NOTE
The current encoder position and speed values are real-time processed and updated continuously (every 200 msec. on the screen).

Press the **HOMEPAGE** button to move back to the Web server **Home** page.

7.3.1 Specific notes on using Internet Explorer

The following options must be set properly on Internet Explorer in order to get the **Encoder position and speed** page to be updated continuously.

- Open the **Settings** menu;
- open the **Internet Options** property sheet;
- in the **General** tabbed page, press the **Setting** button available in the **History Browsing** section;
- under **Check for newer versions of stored pages**, click **Every time I visit the webpage**;
- press the **OK** button to confirm whenever requested.

7.4 Setting the Preset value

Press the **PRESET** button in the left navigation bar of the Web server **Home** page to enter the **Set Encoder Preset** page and set/activate a Preset value. For complete information on the preset function please refer to the **Preset value [105-106]** registers on page 57.

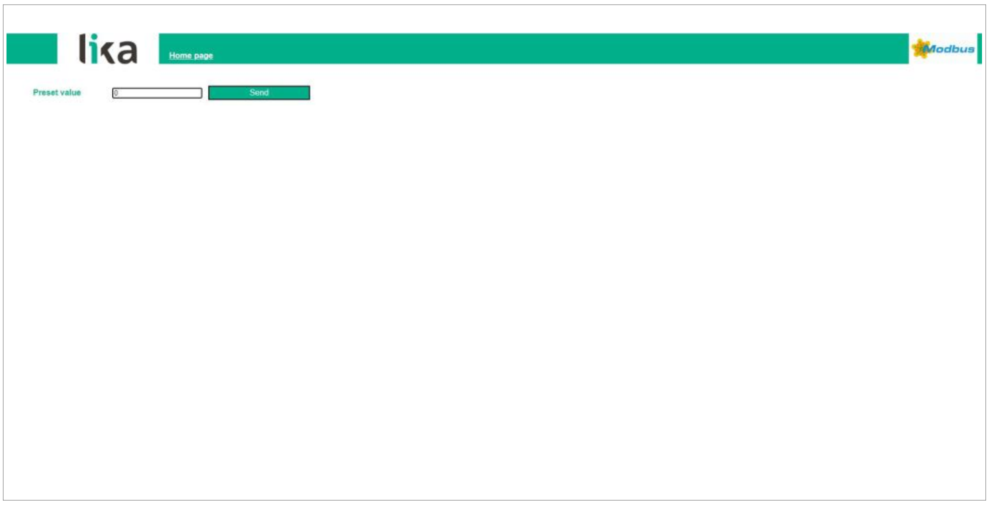


Figure 6 - Set Encoder Preset page

The Preset value that is currently set in the encoder (see the **Preset value [105-106]** registers on page 57) will be displayed in the **PRESET VALUE** box.

To change the Preset enter a suitable value in the same box and then press the **SEND** button to confirm. The value has to be set in decimal notation. The Preset value is set and activated automatically for the position of the encoder in the moment when the **SEND** button is pressed. We suggest activating the preset value when the encoder is in stop. For more information refer also to the bit 11 **Perform counting preset** command in the **Control Word [111-112]** registers on page 62.

NOTE

Please note that the Preset value is now saved <u>temporarily</u> in the **Preset value [105-106]** registers. To save <u>permanently</u> the set Preset value in the **Preset value [105-106]** registers, please press the **SAVE PARAMETERS** button in the **Set Encoder Registers** page. Should the power supply be turned off without saving data, the Preset value that has not been saved on the Flash EEProm will be lost! For more information refer to the bit 9 **Save parameters** command in the **Control Word [111-112]** registers on page 61.

NOTE

At each confirmation of the Preset setting and activation, a message will appears under the box. It informs whether the operation has been accomplished properly or an error occurred (for example, the message **Preset executed correctly!** will appear if everything went well; or **An error occurred executing the preset. Retry.** if something went wrong).

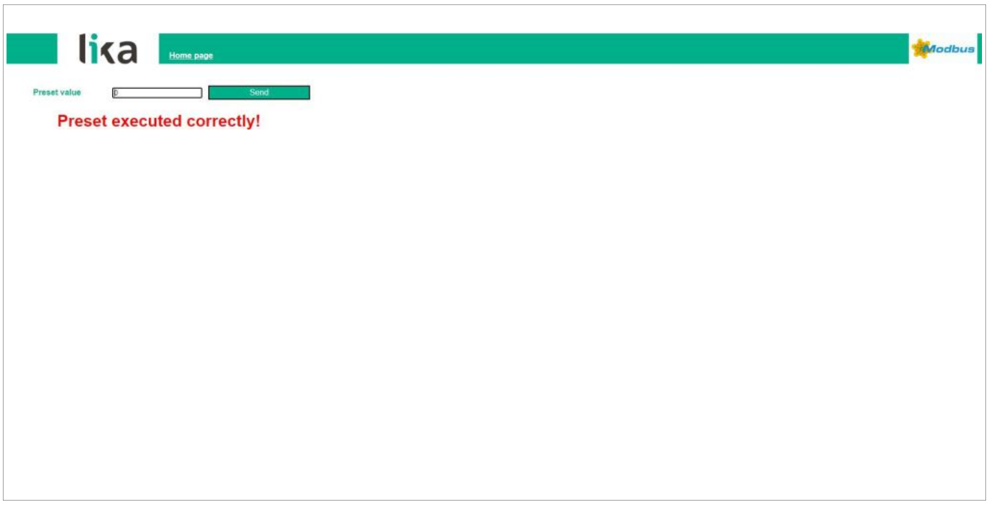


Figure 7 - Preset executed

Press the **HOMEPAGE** button to move back to the Web server **Home** page.

7.5 Encoder information (MODBUS registers)

Press the **ENCODER INFORMATION** button in the left navigation bar of the Web server **Home** page to enter the **Encoder Information** page. In this page the list of the available MODBUS registers is displayed.

Figure 8 - Encoder Information page

The registers listed in this page are **Holding registers**, i.e. they are the encoder configuration parameters; they can be either read-write or read-only access parameters. For a complete description of the Holding registers please refer to the "6.1.1 Holding Register parameters" section on page 52.
Furthermore this page offers some information useful to identify the encoder such as the MAC Address, the serial number, etc.

NOTE
The parameters are made up of two 16-bit registers. For such reason only the first register appears under the **INDEX** column. To read -for instance- the **115 Number of Revolutions** item (see **Number of revolutions [115-116]** on page 63), you must read the register at the address 114 (MSWord) and the register at the address 115 (LSWord).

NOTE
Please note that the values shown in the **Encoder Information** page are "frozen" in the moment when the page is displayed. To update the values you must refresh the web page.

NOTE
The registers in the **Encoder Information** page cannot be changed even though they are read-write access registers. To change the set values please enter the **Set Registers** page (see on page 85).

Press the **HOMEPAGE** button to move back to the Web server **Home** page.

7.6 Setting the registers

Press the **REGISTERS SETTING** button in the left navigation bar of the Web server **Home** page to enter the **Set Encoder Registers** page. In this page the read-write access MODBUS registers are displayed and their value can be changed.
For complete information on the available holding registers please refer to the "6.1.1 Holding Register parameters" section on page 52.

Figure 9 - Set Encoder Registers page

The values that are currently set in the encoder are displayed in green next to each item.

To change any value enter a suitable value in the box next to the desired parameter and then press the **SET** button on the right to confirm. The values have to be set in decimal notation.
For complete information on the available registers please refer to the "6.1.1 Holding Register parameters" section on page 52.

EXAMPLE
The **Counts per revolution [101-102]** registers are currently set to "**8192**" (see the green value in the first line of the Figure above). To change the set value enter a suitable value in the corresponding box of the same line and then press the **SET** button to confirm.

NOTE
Please note that, after pressing the confirmation button, the set value is saved temporarily in the registers. To save it permanently, please press the **SAVE PARAMETERS** button. Should the power supply be turned off without saving data, the values that have not been saved on the Flash EEProm will be lost! For

more information refer to the bit 9 **Save parameters** command in the **Control Word [111-112]** registers on page 61.

Press the **LOAD DEFAULT PARAM.** button to restore all parameters to default values. Default values are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode. This function can be useful, for instance, to restore the factory values in case the encoder is set incorrectly and you are not able to resume the proper operation. For more information refer to the bit 10 **Restore default parameters** command in the **Control Word [111-112]** registers on page 61.

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

Press the **SAVE PARAMETERS** button to save the parameters permanently. Should the power supply be turned off without saving data, the values that have not been saved on the Flash EEProm will be lost! For more information refer to the **Save parameters** command in the **Control Word [111-112]** registers on page 61.

NOTE
At each confirmation of the set registers, a message will appear. It informs whether the operation has been accomplished properly or an error occurred (for example **Register correctly set** if everything went well; or **An error occurred setting the register. Retry.** if something went wrong).

Press the **HOMEPAGE** button to move back to the Web server **Home** page.

7.7 Network parameters

Press the **NETWORK PARAMETERS** button in the left navigation bar of the Web server **Home** page to enter the **Network Parameters** page. This page allows the operator to configure the TCP/IP properties, that is how the encoder communicates with other devices in the network.
For further information on the network communication parameters please refer to the "3.7 Setting the IP address and the network configuration parameters" section on page 25.

WARNING
The network configuration has to be accomplished by skilled and competent personnel.

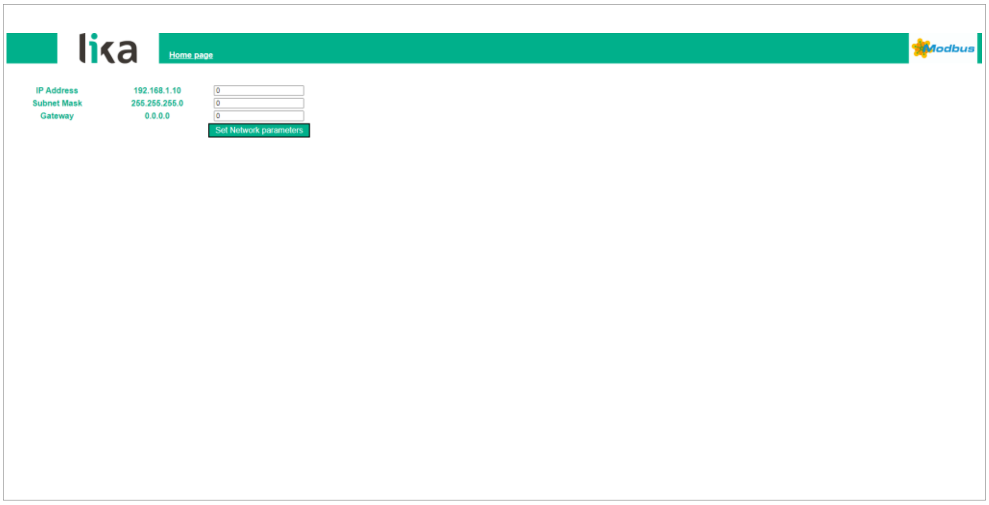


Figure 10 - Network Parameters page

WARNING
Only competent technicians, who are properly trained, have adequate experience and are familiar with computer architecture, network design and operating systems should configure the network communication parameters.
The inappropriate setting of the network parameters results in an incorrect operation of the system.

In this page it is possible to set the parameters that affect the proper communication of the encoder in the TCP/IP network: IP address, Subnet mask, etc.

The following table summarises the default IP address and the network configuration parameters.

IP Parameter	Value
IP address	192.168.1.10
Subnet mask	255.255.255.0
Gateway Address	0.0.0.0

To save the set values permanently, please press the **SAVE PARAMETERS** button in the **Set Encoder Registers** page. Should the power supply be turned off without saving data, the values that have not been saved on the Flash EEProm will be lost!

WARNING

After any setting please note down the configuration values to have access to the encoder and the Web server pages in the future. If for any reason you are not able to communicate with the encoder and enter the Web server pages you must restore the factory values (factory default values) of the network configuration parameters. To do this you must access the DIP A DIP switch located inside the connection cap. For complete information please refer to the "3.7.2 DIP A DIP switch: Resetting the network configuration parameters to the factory values" section on page 26.

NOTE

If for any reason you must restore the factory values (default values) of the network configuration parameters you must access the DIP A DIP switch located inside the connection cap. For complete information please refer to the "3.7.2 DIP A DIP switch: Resetting the network configuration parameters to the factory values" section on page 26.

Press the **HOMEPAGE** button to move back to the Web server **Home** page.

7.8 Firmware update

Press the **FIRMWARE UPDATE** button in the left navigation bar of the Web server **Home** page to enter the **Firmware Update** page. Please note that this is a password protected page, thus a password is requested to access the page.

Password: **LiKa** (“L” and “K” in uppercase letters; “i” and “a” in lowercase letters)

WARNING

Firmware updating process has to be accomplished by skilled and competent personnel. It is mandatory to perform the update 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 off the power supply during the flash update.

This operation allows to update the unit firmware by downloading updating data to the flash memory.
Firmware is a software program which controls the functions 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 functionalities to the device.
The firmware upgrading program consists of a single file having .ZIP extension. It is 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 firmware version currently installed can be read next to the **Software revision** item in the **Encoder Information** page after connection to the web server (see on page 83; see also the **Software revision [129-130]** registers on page 69).

Before proceeding with the firmware update please ascertain that the following requirements are fully satisfied:

- the encoder is connected to the Ethernet network;
- the encoder has valid IP address;
- the PC is connected both to the network and to the IO controller;

- a web browser (Internet Explorer, Mozilla Firefox, Google Chrome, Opera, …) is installed in the PC or device used for connection;
- you have the .ZIP file for firmware update.

To update the firmware program please proceed as follows:

1. press the **FIRMWARE UPDATE** button in the left navigation bar of the Web server **Home** page to enter the **Firmware Update** page;
2. the operator is requested to submit a password before starting the firmware update procedure;

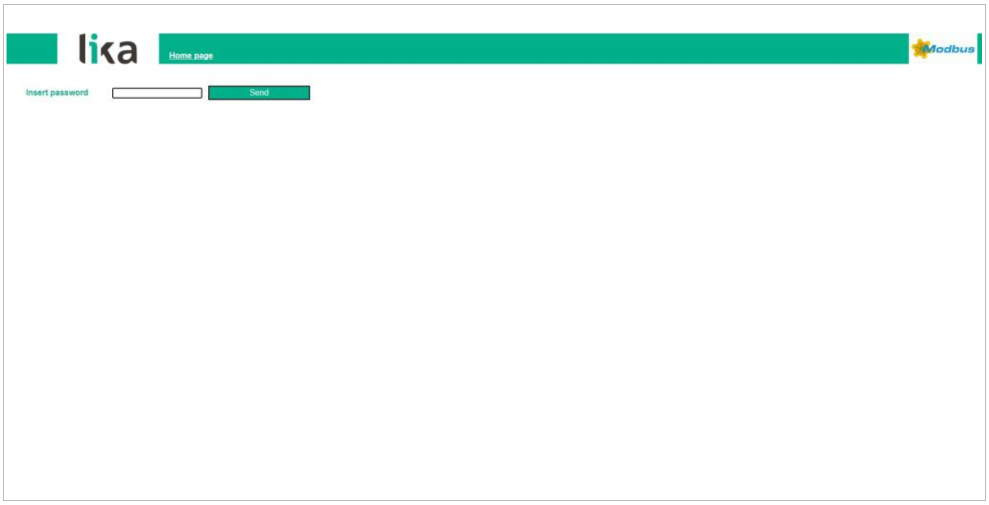


Figure 11 - Firmware Update page

3. in the **Insert password** text box type the password **LiKa** ("L" and "K" in uppercase letters; "i" and "a" in lowercase letters) and then press the **SEND** button;
4. if the password you typed is wrong, the following warning message will appear on the screen: **WRONG PASSWORD INSERTED. RETRY**. Please retype the password and confirm;

5. if the password you typed is correct, the **Firmware Update** page will appear on the screen;

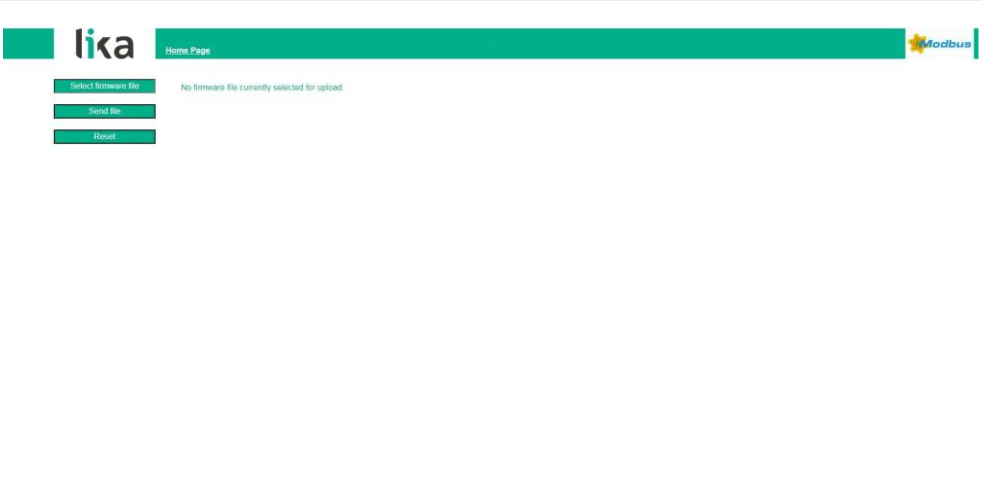


Figure 12 - Firmware Update page

6. press the **SELECT FIRMWARE FILE** button; once you press the **SELECT FIRMWARE FILE** button an **OPEN** dialog box appears on the screen: open the folder where the firmware updating .ZIP file released by Lika Electronic is located, select the file and confirm. Please check the file properties and ascertain that you are installing the correct update file;

Figure 13 - Selecting the firmware update .zip file

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

7. press the **SEND FILE** button to start the upload of the firmware program;
8. during the operation and as soon as the operation is carried out successfully, some messages will appear on the screen;

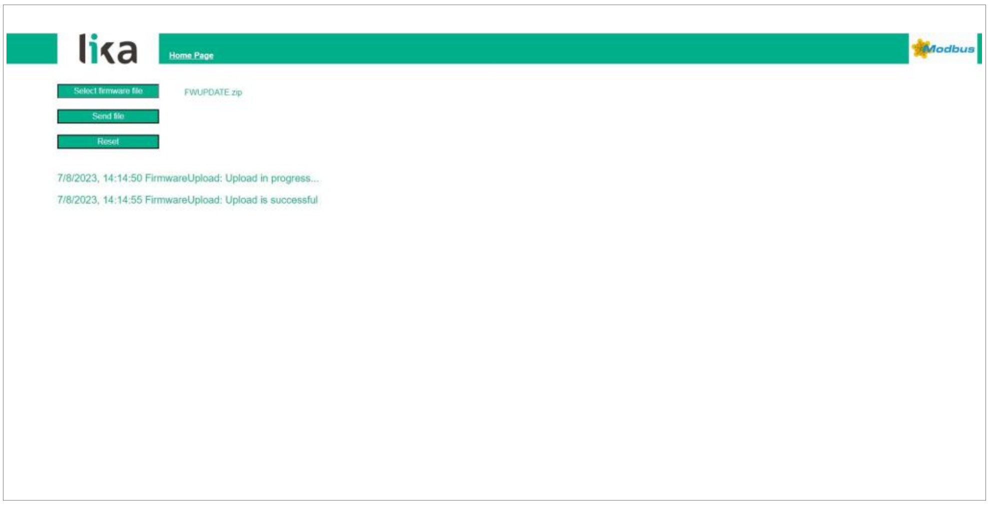


Figure 14 - Messages during firmware upload

9. finally press the **RESET** button to automatically reset and restart the encoder and complete the operation.

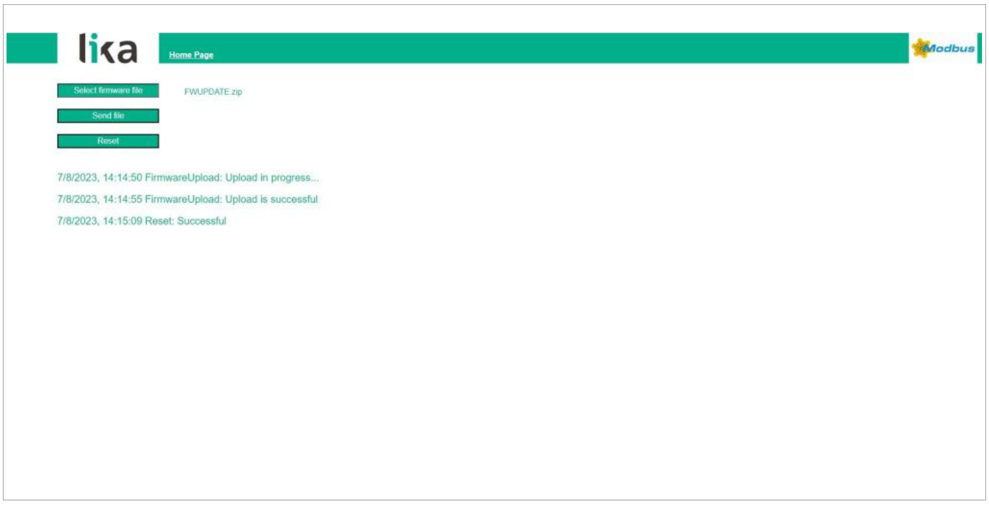


Figure 15 - Firmware update process accomplished

NOTE

While downloading the firmware updating program, 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. In case of flash update error, please switch the encoder off and then on again and retry the operation.

Press the **HOMEPAGE** button to move back to the Web server **Home** page.

8 Programming examples

Hereafter are some examples of both reading and writing parameters. All values are expressed in hexadecimal notation. For any information on the MODBUS/TCP ADU (MBAP Header + PDU) refer to the "5.3 MODBUS on TCP/IP Application Data Unit" section on page 36.

8.1 Using the 03 Read Holding Registers function code

EXAMPLE 1
Request to read the **Preset value [105-106]** registers (address 104-105).

MBAP Header + Request PDU (in hexadecimal notation)

[00][01][00][00][00][06][00][03][00][68][00][02]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[03] = **03 Read Holding Registers** function code
[00][68] = starting address (**Preset value [105-106]** registers, address 104-105)
[00][02] = number of requested registers

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][07][00][03][04][00][68][00][02]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][07] = Length
[00] = Unit Identifier
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[05][DC] = value of register 105, 05 DC hex = 1,500 dec
[00][00] = value of register 106, 00 00 hex = 0 dec

The **Preset value [105-106]** registers (address 104-105) contain the value 00 00 hex and 05 DC hex, i.e. 1,500 in decimal notation; in other words the value set in the **Preset value [105-106]** registers is 1,500 dec.

8.2 Using the 04 Read Input Registers function code

EXAMPLE 1
Request to read the **Current position [1-2]** registers (address 0-1).

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][04][00][00][00][02]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[04] = **04 Read Input Registers** function code
[00][00] = starting address (**Current position [1-2]** registers, address 0-1)
[00][02] = number of requested registers

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][07][00][04][04][2F][F0][00][00]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][07] = Length
[00] = Unit Identifier
[04] = **04 Read Input Registers** function code
[04] = number of bytes (2 bytes for each register)
[2F][F0] = value of register 1, 2F F0 hex = 12,272 dec
[00][00] = value of register 2, 00 00 hex = 0 dec

The **Current position [1-2]** registers (address 0-1) contain the value 00 00 2F F0 hex, i.e. 12,272 in decimal notation.

8.3 Using the 06 Write Single Register function code

EXAMPLE 1

Request to write in the **Watchdog timeout [82]** register (address 81): you want to enable the Watchdog function and set the timeout to 10 ms. Please note that the **Watchdog timeout [82]** register is implemented but not used in this encoder. It is mentioned only as an example.

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][06][00][51][00][0A]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[06] = **06 Write Single Register** function code
[00][51] = address of the **Watchdog timeout [82]** register, 51 hex = 81 dec
[00][0A] = value to be set in the register

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][06][00][51][00][0A]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[06] = **06 Write Single Register** function code
[00][51] = address of the **Watchdog timeout [82]** register, 51 hex = 81 dec
[00][0A] = value set in the register

The value 00 0A hex (10 dec) is set is set in the **Watchdog timeout [82]** register (address 81): the Watchdog function is enabled and the timeout is set to 10 ms.

8.4 Using the 16 Write Multiple Registers function code

EXAMPLE 1
Request to write the value 00 00 08 00 hex (=2,048 dec) next to the **Counts per revolution [101-102]** registers (address 100-101) and the value 00 80 00 00 hex (= 8,388,608 dec) next to the **Total Resolution [103-104]** registers (address 102-103).

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][0F][00][10][00][64][00][04][08][08][00][00][00][00][00][00][80]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][0F] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][64] = starting address (**Counts per revolution [101-102]** registers, address 100-101)
[00][04] = number of requested registers
[08] = number of bytes (2 bytes for each register)
[08][00] = value to be set in the register 101, 08 00 hex
[00][00] = value to be set in the register 102, 00 00 hex (00 00 08 00 hex = 2,048 dec)
[00][00] = value to be set in the register 103, 00 00 hex
[00][80] = value to be set in the register 104, 00 80 hex (00 80 00 00 hex = 8,388,608 dec)

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][10][00][64][00][04]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][64] = starting address (**Counts per revolution [101-102]** registers, address 100-101)
[00][04] = number of written registers

The values 00 00 hex and 08 00 hex, i.e. 2,048 in decimal notation, are set in the **Counts per revolution [101-102]** registers at address 100-101; while the values 00 80 hex and 00 00 hex, i.e. 8,388,608 in decimal notation, are set in the **Total Resolution [103-104]** registers at address 102-103. Thus the encoder will be programmed to have a 2,048-count-per-revolution single-turn resolution and 4,096 revolutions (8,388,608 / 2,048).

EXAMPLE 2

Request to write in the Operating parameters [109-110] registers (address 108-109): we need to set the scaling function (bit 0 **Scaling function** = 1) and the count up information with clockwise rotation of the encoder shaft (bit 1 **Code sequence** = 0). The value to set is 00 00 00 01 hex (= 0000 0000 0000 0000 0000 0000 0000 0001 in binary notation: the bit 0 **Scaling function** = 1; the bit 1 **Code sequence** = 0; the remaining bits are not used, therefore their value is 0).

MBAP Header + Request PDU (in hexadecimal notation)
[00][01][00][00][00][0B][00][10][00][6C][00][02][04][00][01][00][00]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][0B] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][6C] = starting address (Operating parameters [109-110] registers, address 108-109)
[00][02] = number of requested registers
[04] = number of bytes (2 bytes for each register)
[00][01] = value to be set in the register 109, 00 01 hex
[00][00] = value to be set in the register 110, 00 00 hex

MBAP Header + Response PDU (in hexadecimal notation)
[00][01][00][00][00][06][00][10][00][6C][00][02]
where:
[00][01] = Transaction Identifier
[00][00] = Protocol Identifier
[00][06] = Length
[00] = Unit Identifier
[10] = **16 Write Multiple Registers** function code
[00][6C] = starting address (Operating parameters [109-110] registers, address 108-109)
[00][02] = number of written registers

The value 00 00 00 01 hex, i.e. 0000 0000 0000 0000 0000 0000 0000 0001 in binary notation is set in the Operating parameters [109-110] registers (address 108-109): the bit 0 **Scaling function** = 1; the bit 1 **Code sequence** = 0; the remaining bits are not used, therefore their value is 0.

9 Default parameters list

Default values are expressed in decimal notation, unless otherwise indicated.

Parameters list	Default values		
Watchdog timeout [82]	0 (not used)		
Counts per revolution [101-102]	8,192		
Total Resolution [103-104]	134,217,728		
Preset value [105-106]	0		
Speed format [107-108]	0 = steps/s		
Operating parameters [109-110]	0000 0000 hex		
	bit 0 Scaling function = 0		
	bit 1 Code sequence = 0		
Singleturn resolution [113-114]	8,192		
Number of revolutions [115-116]	16,384		
Supported alarms [117-118]	0000 7800 hex		
Supported warnings [119-120]	0000 0001 hex		

This device is to be supplied by a Class 2 Circuit or Low-Voltage Limited Energy or Energy Source not exceeding 30 Vdc. Refer to the ordering code for supply voltage rate.
Ce dispositif doit être alimenté par un circuit de Classe 2 ou à très basse tension ou bien en appliquant une tension maxi de 30Vcc. Voir le code de commande pour la tension d'alimentation.

Dispose separately