lika
### Smart encoders & actuators

User's guide

SFA-5000-MT
SFA-10000-MT

Modbus
MODBUS TCP/IP

- 5000 mm & 10000 mm draw-wire encoder
- Integrated 27 bit absolute multiturn encoder
- Programmable resolution down to 24 μm
- M12 connectors
- Complies with the "Modbus over TCP/IP" protocol

Suitable for the following models:
- SFA-5000-MT
- SFA-10000-MT

General Contents

Lika Electronic • Tel. +39 0445 806600 • info@lika.biz • www.lika.biz

lika

General contents

Subject index

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 and exhaustive information the operator needs to correctly and safely install and operate the **SFA-5000 and SFA-10000 absolute draw-wire encoders with MODBUS TCP/IP interface**.
The cable pulling mechanism integrates a 13 x 14 bit absolute multiturn encoder (13 bits = singleturn resolution = 8,192 cpr; 14 bits = 16,384 revolutions).

For technical specifications please refer to the product datasheet.

SFA-5000 / SFA-10000 cable-pulling encoder is aimed at speed and position measurements and controls in a variety of industrial applications through the movement of a **5,000 mm (196.85") or 10,000 mm (393.7")** stainless steel wire. The typical back and forth travel of the moving equipment causes the wire to reel and unreel and thus the linear movement to be converted into a rotary motion detected by the encoder which coupled to a drum.
The stroke per turn is always 200 mm (7.874"), the maximum number of turns is 25 for SFA-5000 and 50 for SFA-10000.

To make it easier to read and understand the text, this guide can be divided into two main sections.
In the first section general information concerning the safety, the mechanical installation and the electrical connection as well as tips for setting up and running properly and efficiently the SFA-5000 / SFA-10000 cable-actuated encoder are provided.
In the second section, entitled **MODBUS TCP/IP Interface**, you can find detailed information on the MODBUS interface. In this section the interface features and the registers implemented in the unit are fully described.

Glossary of MODBUS TCP/IP terms

MODBUS TCP/IP, 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/IP 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/IP 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 34.
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/IP, 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/IP 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 34.
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 behaviour 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 72.
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 36 and 72.
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 37.
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 50.
Host	A computer or other device on a TCP/IP network.

IEEE 1588	This standard defines a protocol enabling synchronisation of clocks in distributed networked devices (e.g. connected via Ethernet).
Input register	In the MODBUS data model, an Input register is the input data. An Input register has a 16-bit quantity, is provided by an I/O system, and allows read-only access. Refer to page 68.
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 26.
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 26.
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 34.
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 36.
MODBUS Response	A MODBUS Response is the Response message sent by the Server. Refer to page 36.
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 36.
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 if IP datagrams that need to go to many different (simultaneous) application processes. MODBUS TCP/IP 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 ina group of work parameters placed in order. Refer to page 38.
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 40.
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 50.
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/IP 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 25.
Transmission rate	Data transfer rate (in bps). Refer to page 25.
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 44.
Write Single Register (06, 0006hex)	This function code is used to WRITE a single holding register in a remote device. Refer to page 42.

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/IP interface.

ADU	Application Data Unit
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
PDU	Protocol Data Unit
PLC	Programmable Logic Controller
TCP	Transmission Control Protocol

References

[1]   MODBUS Application Protocol Specification, Version V1.1b3
[2]   MODBUS messaging on TCP/IP implementation guide, Version V1.0b
[3]   RFC 791, Internet Protocol, Sep81 DARPA
[4]   RFC 1122 Requirements for Internet Hosts -- Communication Layers
[5]   IEC 61918 Industrial communication networks – Installation of communication networks in industrial premises
[6]   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 explanation in the "4 - Electrical connections" section on page 24;
- in compliance with the 2014/30/EU norm on electromagnetic compatibility, 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 cap, use one TCEI M3 x 6 cylindrical head screw with two tooth lock washers.

1.3 Mechanical safety

- Install the device following strictly the information in the "3 - Mounting instructions" section on page 20;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the encoder;
- do not tool the encoder;
- delicate electronic equipment: handle with care; do not subject the device to knocks or shocks;
- respect the environmental characteristics of the product;
- we suggest installing the unit providing protection means against waste, especially swarf as turnings, chips, or filings; should this not be possible, please make sure that adequate cleaning measures are in place in order to prevent the wire from jamming;
- to avoid failures, never exceed the maximum measuring length and prevent the wire from tangling up;
- never release the wire freely, always help the wire wind properly: risk of personal injury and/or damage to the equipment;
- always keep the wire aligned not to damage the equipment;
- the stroke per turn of the draw-wire unit is 200 mm (7.874").

2   Identification

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

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

3 Mounting instructions

WARNING
Installation, electrical connection, and maintenance operations must be carried out by qualified personnel only, with power supply disconnected. Mechanical components must be in stop.

3.1 Overall dimensions

SFA-5000   SFA-10000

SFA-5000   SFA-10000

A = 45 SFA-10000
    39 SFA-5000

B = 10 SFA-10000
    7 SFA-5000
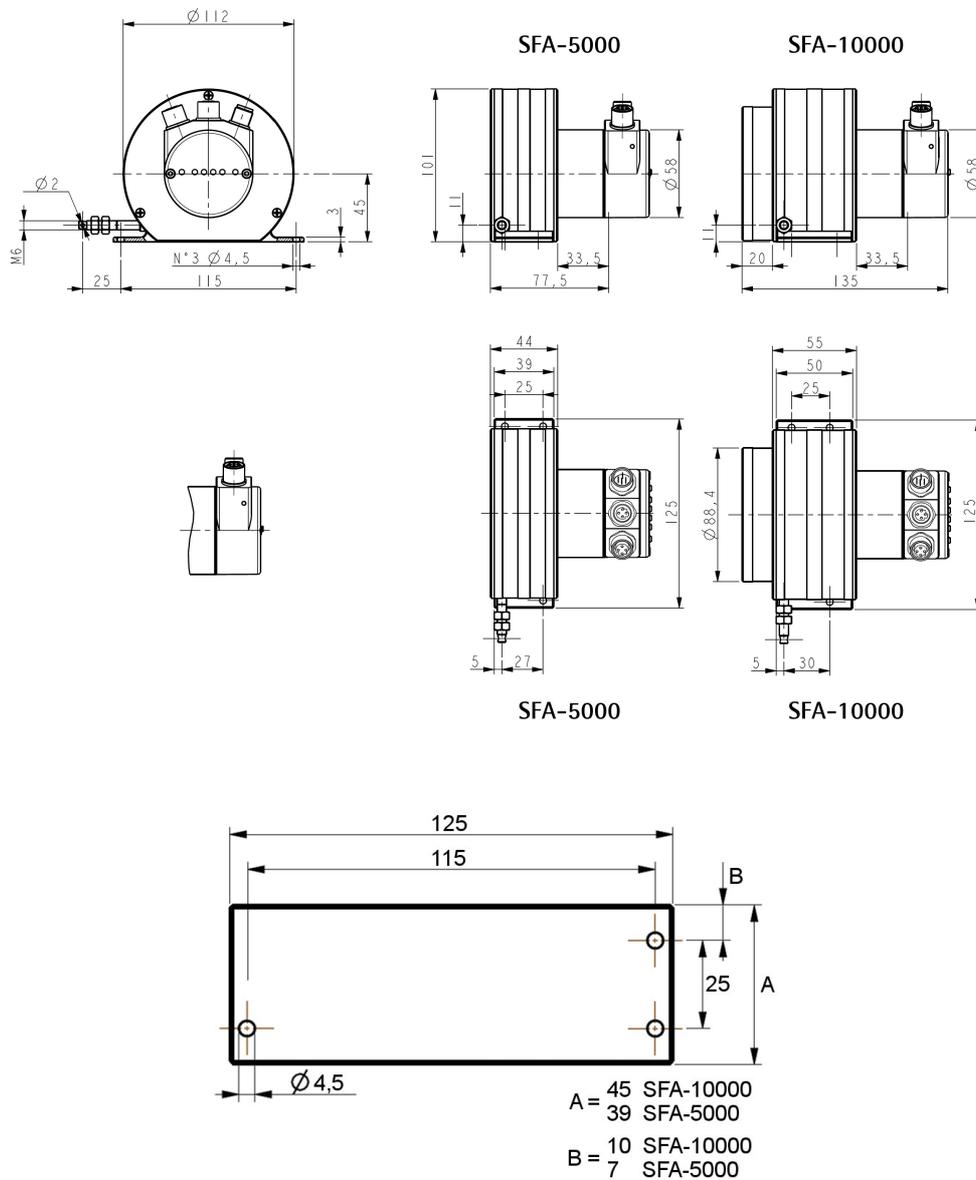
Values are expressed in mm

3.2 Mounting instructions
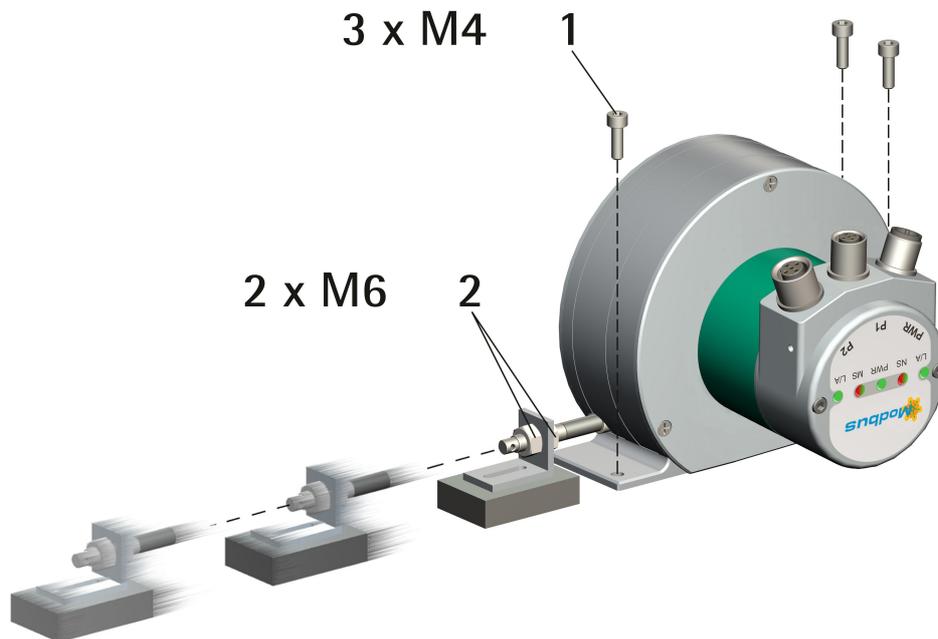
- Fasten the draw-wire unit onto a fixed support using three M4 screws **1**;
- remove the transport safety wire that pins the end of the measuring wire;
- fix the end of the measuring wire to the moving unit using the provided M6 nuts **2**.

WARNING
We suggest installing the unit providing protection means against waste, especially swarf as turnings, chips, or filings; should this not be possible, please make sure that adequate cleaning measures are in place in order to prevent the wire from jamming.
To avoid irreparable failures, never exceed the maximum measuring length and prevent the wire from tangling up.

Never release the wire freely, always help the wire wind properly: risk of personal injury and/or equipment damage.
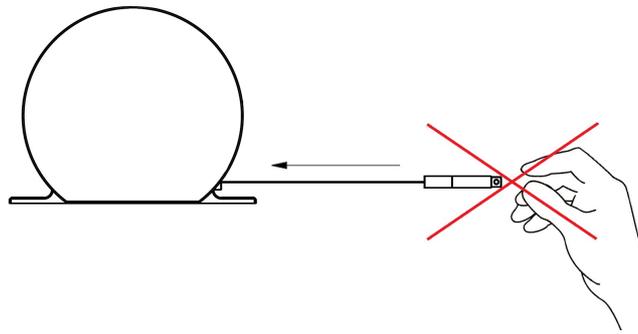
Always keep the wire aligned not to damage the equipment (maximum deviation: 3°).
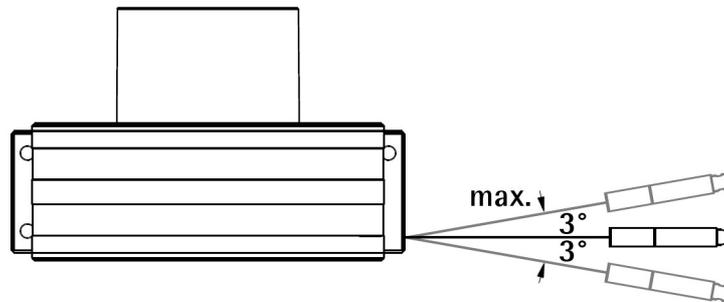
3.3 Useful information

If you want to know the **maximum measuring length** and the **physical linear resolution** of the draw-wire encoder please refer to the order code. The stroke per turn is always 200 mm (7.874"), the maximum number of turns is 25 for SFA-5000 and 50 for SFA-10000.

EXAMPLE 1
SFA-**5000**-**MT**-**8192**-M12 using the physical resolution (**Scaling function** in **Operating parameters [109-110]** = 0)
Stroke per turn of the drum = 200 mm (7.874")
Physical resolution per turn = 13 bits = 8,192 cpr
Max. number of physical revolutions = 14 bits = 16,384 revolutions
Total physical resolution = 27 bits = 134,217,728 information
Physical linear resolution = 0.024 mm = 24 µm
Max. number of turns of the drum = 25
Max. measuring length = 5,000 mm (196.85")
Number of information = 204,800

EXAMPLE 2

SFA-**10000**-**EC**-**8192**-M12 using a custom resolution (**Scaling function** in **Operating parameters [109-110]** = 1)

Stroke per turn of the drum = 200 mm (7.874")
Physical resolution per turn = 13 bits = 8,192 cpr
Max. number of physical revolutions = 14 bits = 16,384 revolutions
Custom resolution per turn = **Counts per revolution [101-102]** = 2,000 cpr (example)
Total Resolution [103-104] = 8,192,000 information (example)

$$\text{Custom number of encoder revolutions} = \frac{\textbf{Total Resolution [103-104]}}{\textbf{Counts per revolution [101-102]}} = 4{,}096$$

Linear resolution = 0.1 mm = 100 µm
Max. number of turns of the drum = 50
Max. measuring length = 10,000 mm (393.7")
Number of information = 100,000

3.4 Maintenance

The measuring system does not need any particular maintenance; anyway it has to be handled with the utmost care as any delicate electronic equipment. From time to time we recommend the following operations:

- the unit and the wire have to be cleaned regularly using a soft and clean cloth to remove dust, chips, moisture etc.; do not use oil to clean the wire.

4   Electrical connections

WARNING
Installation, electrical connection, and maintenance operations must be carried out by qualified personnel only, with power supply disconnected. Mechanical components must be in stop.

For any information on the mechanical and electrical characteristics of the encoder please refer to the technical catalogue.
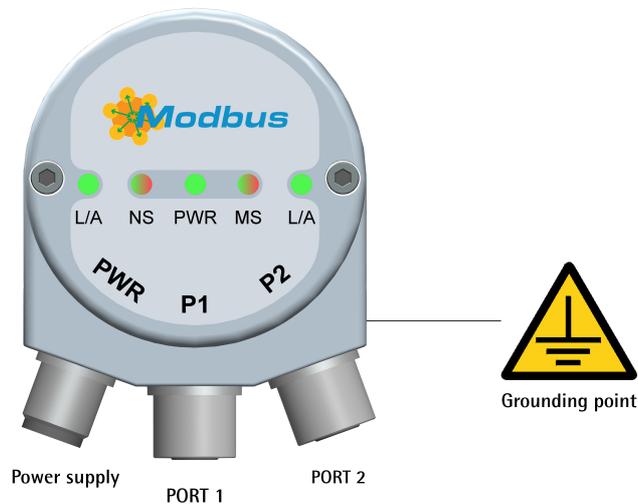
Figure 1 – Connectors and diagnostic LEDs

4.1 M12 connectors
The connection cap is fitted with three M12 connectors with pin-out in compliance with the Ethernet standard. Therefore you can use standard Ethernet cables commercially available. PORT 1 and PORT 2 are interchangeable.

4.1.1 PWR Power supply connector (Figure 1)
M12 4-pin male connector with A coding is used for power supply.
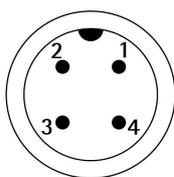
Description	Pin
+10Vdc +30Vdc	1
n.c.	2
0Vdc	3
n.c.	4

n.c. = not connected

4.1.2 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.
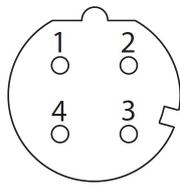
Description	Pin
Tx Data +	1
Rx Data +	2
Tx Data –	3
Rx Data –	4

The ports are equal and interchangeable - if only one connection is required, either port can be used. The Ethernet interface supports 10/100 Mbit/s, full/half duplex operation.

4.1.3 Network configuration: topologies, cables, hubs, switches – Recommendations

Using Ethernet several topologies of connection are supported by MODBUS TCP/IP networks: line, tree, daisy-chain, star, ... Furthermore MODBUS TCP/IP 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.

For a complete list of the available cordsets and connection kits please refer to the product datasheet ("Accessories" list).

4.2 Ground connection

To minimize noise connect properly the shield and/or the connector housing and/or the frame to ground. Connect properly the cable shield to ground on user's side. Lika's EC- pre-assembled cables are fitted with shield connection to the connector ring nut in order to allow grounding through the body of the

device. Lika's E- connectors have a plastic gland, thus grounding is not possible. If metal connectors are used, connect the cable shield properly as recommended by the manufacturer. Anyway make sure that ground is not affected by noise. It is recommended to provide the ground connection as close as possible to the device. We suggest using the ground point provided in the cap (see Figure 1, use 1 TCEI M3 x 6 cylindrical head screw with two tooth lock washers).

4.3 Connection of the shield

Disentangle and shorten the shielding **1** and then bend it over the part **2**; finally place the ring nut **3** of the connector. Be sure that the shielding **1** is in tight contact with the ring nut **3**.
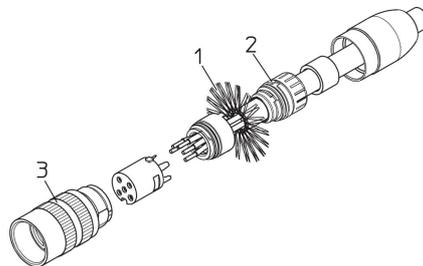
4.4 MAC address and IP address
The unit can be identified in the network through the **MAC address** and the **IP address**.
The MAC address is an identifier unique worldwide and has to be intended as a permanent and globally unique identifier assigned to the unit for communication on the physical layer; 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 default subnet mask is 255.255.255.0.
To set the network configuration parameters refer to the next section.

4.5 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.

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 Configuration** 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 Configuration** page refer to the "8.8 Network configuration" section on page 92.

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 "4.8 DIP A: Resetting the network configuration parameters to the factory values" section on page 29.

4.6 Line Termination

MODBUS TCP/IP 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.

4.7 Diagnostic LEDs (Figure 1)

Five LEDs located in the cap of the encoder (see the Figure 1) are designed 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.

LED	Description
L/A Link/Activity LED for port 1 P1 (green)	It shows the state and the activity of the physical link (port 1 P1).
OFF	Neither link active nor activity
ON	Port 1 P1 link active, no activity.
BLINKING	Activity on port 1 P1.

LED	Description
NS Network State Error LED (green / red)	It shows the current state of the network.
OFF	No IP address detected or fault status.
ON green	One MODBUS message at least has been received.
FLASHING green	The encoder is waiting for the first MODBUS message to be received.
ON red	IP address conflict detected or FATAL ERROR.
FLASHING red	Connection timeout, no MODBUS message has been received within the set timeout time (see the Watchdog timeout [82] register on page 51).

LED	Description
PWR Power LED (green)	It shows the power supply state.

OFF	The encoder power supply is switched OFF.
ON	The encoder power supply is switched ON.

LED	Description
MS Module Status LED (green / red)	It shows the state of the MODBUS TCP/IP device.
OFF	The device power supply is switched OFF.
ON green	The device is in normal operational state.
ON red	Major fault, FATAL ERROR.
FLASHING red	Minor fault.

LED	Description
L/A Link/Activity LED for port 2 P2 (green)	It shows the state and the activity of the physical link (port 2 P2).
BLINKING	Activity on port 2 P2.
ON	Port 2 P2 link active, no activity.

NOTE
If both the NS Network Status LED and the MS Module Status LED are red, a fatal error has occurred.

NOTE
During startup, the device carries out a hardware test to check the LEDs operation.
Both NS Network Status and MS Module Status LEDs light up.

4.8 DIP A: Resetting the network configuration parameters to the factory values

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

connection cap. For any information on removing the connection cap please refer to the following "4.9 Connection cap (Figure 2)" section.

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

WARNING
Please pay the utmost attention to the internal wirings and connections while the cap is open.
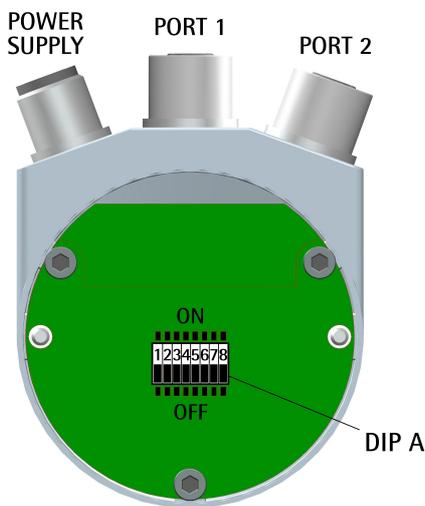
- Turn the power supply off;
- remove the connection cap as explained in the following "4.9 Connection cap (Figure 2)" section;
- set the hardware switch 1 to ON;
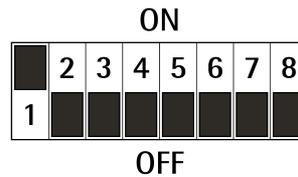
- reconnect the cap, then turn the power supply on and wait for the initialization process to be completed;
- turn the power supply off;
- remove the connection cap again and set the hardware switch 1 to OFF again;
- replace the connection cap as explained in the following "4.9 Connection cap (Figure 2)" section;
- 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
DHCP	OFF

4.9 Connection cap (Figure 2)

WARNING
Do not remove or mount the connection cap with power supply switched ON. Damage may be caused to internal components.

The DIP A DIP switch meant to reset the network configuration parameters to the factory values (default values) is located inside the connection cap (see on page 29). Thus you must remove the connection cap to access it.

NOTE
Be careful not to damage the internal components when you perform this operation.

To remove the connection cap loosen the two screws **1**. Please be careful with the internal connector.
Always replace the connection cap at the end of the operation. Take care in re-connecting the internal connector. Tighten the screws **1** using a tightening torque of approx. 2.5 Nm.

WARNING
You are required to check that the encoder body and the connection cap are at the same potential before replacing the connection cap!

Grounding point
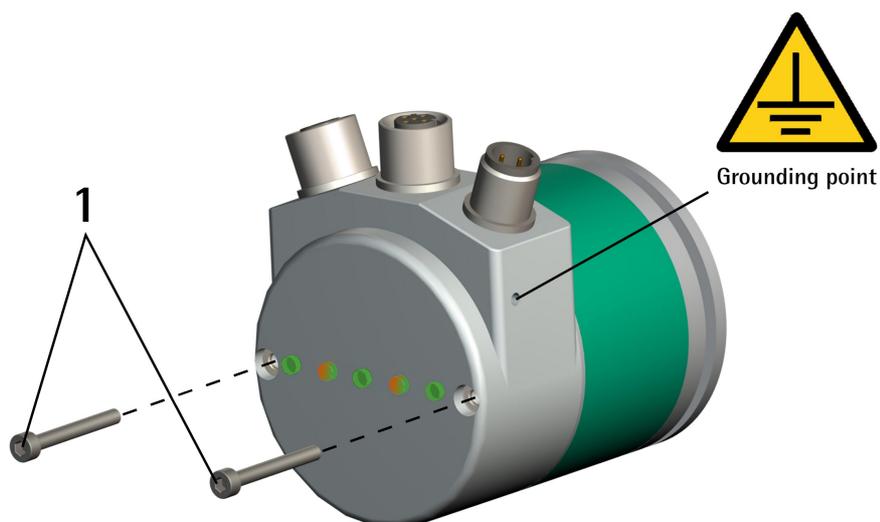
Figure 2 – Removing the connection cap

5   Quick reference

5.1 Getting started

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

- Mechanically install the device;
- execute the electrical connections;
- if required, set the communication parameters to allow the unit to access the MODBUS TCP/IP network, see the "4.5 Setting the IP address and the network configuration parameters" section on page 27; 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

- switch on the +10Vdc +30Vdc power supply;
- if you need to use the physical resolution of the unit, please check that the **Scaling function** item is disabled (the bit 0 in the Operating parameters [109-110] registers = 0; see on page 58); the encoder will use the Singleturn resolution [113-114] and the Number of revolutions [115-116] register values to arrange the absolute position information;
- otherwise if you need a specific resolution, please enable the **Scaling function** item (the bit 0 in the Operating parameters [109-110] registers = 1; see on page 58);
- then set the value you need for the singleturn resolution next to the Counts per revolution [101-102] registers, see on page 52;
- set the value you need for the overall resolution next to the Total Resolution [103-104] registers, see on page 54;
- 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 60).

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

6 MODBUS® TCP/IP interface

Lika MODBUS TCP/IP draw-wire 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.

6.1 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/IP 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/IP (also MODBUS-TCP) 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/IP messaging.

Among the significant advantages of MODBUS TCP/IP are:
- MODBUS TCP/IP 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.

6.2 General MODBUS frame description

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

Function Code	Data
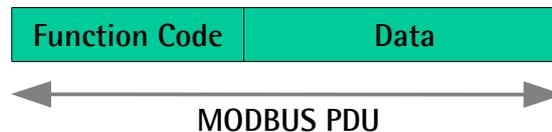
MODBUS PDU

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.

MODBUS RTU ADU

Address	Function Code	Data	Checksum

MODBUS PDU
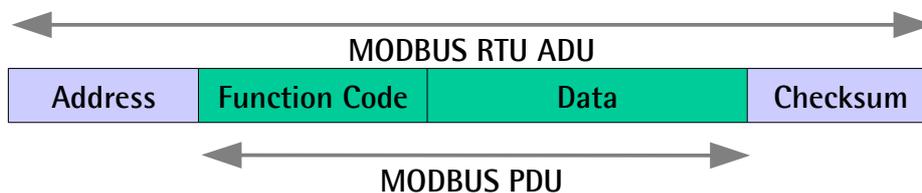
6.3 MODBUS on TCP/IP Application Data Unit

MODBUS TCP/IP uses TCP/IP and Ethernet to carry the data of the MODBUS message structure between compatible devices. That is, MODBUS TCP/IP 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/IP 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.

CONSTRUCTION OF A MODBUS TCP DATA PACKET
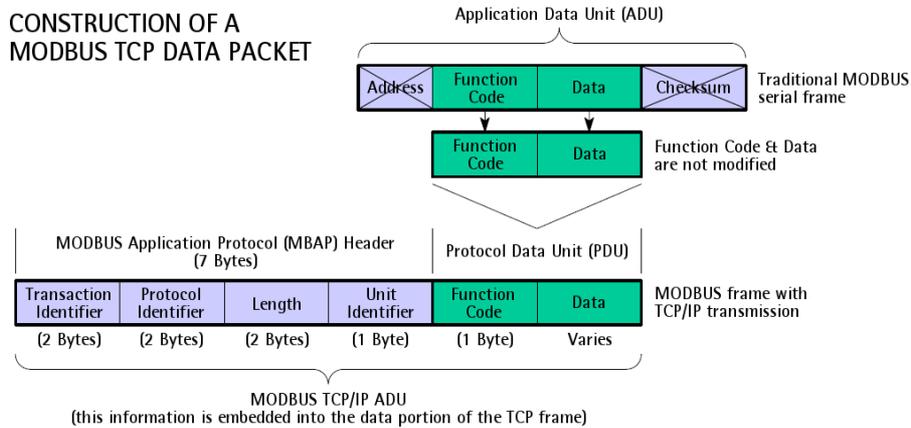
As you can see, the Protocol Data Unit is integrated in its original form.

The MODBUS TCP/IP 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/IP 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 "6.5 Function codes" section on page 37.
- **DATA**: the **DATA** field of messages contains the bytes for additional information and transmission specifications that the Server uses to take the action defined by the **FUNCTION CODE**. This can include items such as discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field. The structure of the **DATA** field depends on each **FUNCTION CODE** (refer to the "6.5 Function codes" section on page 37).

The complete MODBUS TCP/IP 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/IP 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

6.4 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 7 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**.

6.5 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.

6.5.1 Implemented function codes

Lika MODBUS TCP/IP draw-wire encoders only 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 "7.1.1 Holding Register parameters" section on page 50.

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			Response		
Field name		(Hex)	Field name		(Hex)
Function		**03**	Function		**03**
Starting address Hi		**00**	Byte count		**04**
Starting address Lo		**68**	Register 105 value Hi		**00**
No. of registers Hi		**00**	Register 105 value Lo		**00**
No. of registers Lo		**02**	Register 106 value Hi		**05**
			Register 106 value Lo		**DC**

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. 1500 in decimal notation.

The MODBUS TCP/IP 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/IP 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][00][00][05][DC]
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)
[00][00] = value of register 105, 00 00 hex = 0 dec
[05][DC] = value of register 106, 05 DC hex = 1500 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 "7.1.2 Input Register parameters" section on page 68.

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			Response	
Field name	(Hex)		Field name	(Hex)
Function	04		Function	04
Starting address Hi	00		Byte count	04
Starting address Lo	00		Register 1 value Hi	00
Quantity of Input Reg. Hi	00		Register 1 value Lo	00
Quantity of Input Reg. Lo	02		Register 2 value Hi	2F
			Register 2 value Lo	F0

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. 12272 in decimal notation.

The MODBUS TCP/IP 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/IP 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][00][00][2F][F0]
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)
[00][00] = value of register 1, 00 00 hex = 0 dec

[2F][F0] = value of register 2, 2F F0 hex = 12272 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 "7.1.1 Holding Register parameters" section on page 50.

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.

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/IP 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/IP 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 "7.1.1 Holding Register parameters" section on page 50.

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 (=2048 dec) next to the Counts per revolution [101-102] registers (address 100-101) and the value 00 80 00 00 hex (=8388608 dec) next to the Total Resolution [103-104] registers (address 102-103).

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

As you can see in the table, the values 00 00 hex and 08 00 hex, i.e. 2048 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. 8388608 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 2048-count-per-revolution singleturn resolution and 4096 revolutions (8388608/2048).

The MODBUS TCP/IP ADU needed for the request to write the value 2048 dec next to the Counts per revolution [101-102] registers (address 100-101) and the value 8388608 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][00][00][08][00][00][80][00][00]
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)
[00][00] = value to be set in the register 101, 00 00 hex
[08][00] = value to be set in the register 102, 08 00 hex (00 00 08 00 hex = 2048 dec)
[00][80] = value to be set in the register 103, 00 80 hex
[00][00] = value to be set in the register 104, 00 00 hex (00 80 00 00 hex = 8388608 dec)

The MODBUS TCP/IP ADU needed to send back a response following the request to write the value 2048 next to the **Counts per revolution [101–102]** registers (address 100-101) and the value 8388608 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 when you rewind the wire (bit 1 **Code sequence** = 0).

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

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/IP ADU needed for the request to set the scaling function (bit 0 **Scaling function** = 1) and the count up information when you rewind the wire (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][00][00][01]
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][00] = value to be set in the register 109, 00 00 hex
[00][01] = value to be set in the register 110, 00 01 hex

The MODBUS TCP/IP 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 when you rewind the wire (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

WARNING
For safety reasons, while the encoder is on, a continuous data exchange between the Master and the Slave has to be planned in order to be sure that the communication is always active; this is intended to prevent danger situations from arising in case of failures in the communication network.
For this purpose the Watchdog function is implemented and can be enabled. The Watchdog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the communication be cut off while a command is still active and running, the Watchdog safety system immediately takes action and commands an alarm to be triggered.
The **Watchdog timeout [82]** register is used to disable/enable the Watchdog function and further to set the timeout value expressed in milliseconds. Setting the register to 0 causes the Watchdog function to be disabled. Any value greater than 0 enables the Watchdog function and sets the Watchdog timeout. When the Watchdog function is enabled, if the device does not receive a message from the Server within the set time, the system forces the encoder to exit the network participation and shift to the **EXCEPTION** state. Furthermore the NS Network State Error LED starts flashing red.

6.6 Encoder states

The table below describes the states the encoder can enter during operation in the MODBUS TCP/IP 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 State Error LED lights up red (see on page 28).
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 State Error LED starts flashing red (see on page 28).

7 Programming parameters

7.1 Parameters available

Hereafter the parameters available for the MODBUS draw-wire 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).

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

7.1.1 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 "6.5.1 Implemented function codes" section on page 38.

NOTE

Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the **Save parameters** function available in the **Control Word [111-112]** registers, see on page 60.
Should the power supply be turned off all data that has not been saved previously will be lost!

Watchdog timeout [82]

[81, Unsigned16, rw]

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

For this purpose the Watchdog function is implemented and can be enabled. The Watchdog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the communication be cut off while a command is still active and running, the Watchdog safety system immediately takes action and commands an alarm to be triggered.

This register is used to disable/enable the Watchdog function and further to set the timeout value expressed in milliseconds. Setting the register to 0 causes the Watchdog function to be disabled. Any value greater than 0 enables the Watchdog function and sets the Watchdog timeout. When the Watchdog function is enabled, if the device does not receive a message from the Server within the set time, the system forces the encoder to exit the network participation and shift to the **EXCEPTION** state. Furthermore the NS Network State Error LED starts flashing red.

Default = 0 (min. = 0, max. = 65535)

NOTE

As soon as the Watchdog function is enabled (**Watchdog timeout [82]** > 0), one MODBUS command at least must be sent in order to activate the timeout.

NOTE

The **Watchdog timeout [82]** value is immediately saved in the memory as soon as it is set; thus you are not required to save it by means of the the **Save parameters** function.

Current position [95-96]

[94-95, Unsigned16, 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 68.

Speed value [97-98]

[96-97, Signed16, 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 69.

Status word [99-100]

[98-99, Unsigned16, 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 (custom singleturn resolution).

You are allowed to set whatever integer value that is 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 value is not accepted.

To avoid counting errors, please check that:

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

Default = 8,192 (min. = 1, max. = 8,192)

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 SFA-5000-MT-8192-M12 draw-wire 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 cpr.

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

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

As you can see, the encoder is required to carry out more than 93,000 revolutions, this cannot be as the hardware number of revolutions is, as stated, 16,384 (see the **Singleturn resolution [113-114]** registers). When this happens, the value is not accepted.

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, they are not accepted.

WARNING
If you have set the preset, every time you change the value next to the **Counts per revolution [101-102]** registers, then you must check the value in the **Preset value [105-106]** registers; if necessary you are required to set a new preset and perform the preset operation (bit 11 **Perform counting preset** in **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 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 that is less than or equal to the **overall hardware resolution** (Singleturn resolution [113-114] * Number of revolutions [115-116]). See the encoder identification label.
If you set an out-of-range value (i.e. greater than the overall hardware resolution), the value is not accepted.

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:

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

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:

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

Let's suppose that the SFA-5000-MT-8192-M12 draw-wire encoder is programmed as follows:
Counts per revolution [101-102]: 8,192
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. When this happens, the value is not accepted.

WARNING
If you have set the preset, every time you change the value next to the **Total Resolution [103-104]** registers, then you must check the value in the **Preset value [105-106]** registers; if necessary you are required to set a new preset and perform the preset operation (bit 11 **Perform counting preset** in the **Control Word [111-112]** registers = 1).

EXAMPLE
We install the following draw-wire encoder: **SFA-5000-MT-8192-M12**.

The physical values are:
Stroke per turn of the drum = 200 mm (7.874")
Physical resolution per turn = 13 bits = 8,192 cpr
Max. number of physical revolutions = 16,384 revolutions
Total physical resolution = 27 bits = 134 217 728 information
Physical linear resolution = 0.024 mm = 24 μm
Max. number of turns of the drum = 25
Max. measuring length = 5,000 mm (196.85")
Number of information = 204,800

Let's suppose that we need a tenth of a millimetre linear resolution in the specific installation.
- Enable the scaling function: bit 0 **Scaling function** in the **Operating parameters [109-110]** registers = 1
- Custom resolution per turn = **Counts per revolution [101-102]** = 2,000 cpr
- Linear resolution = 0.1 mm = 100 μm

$$\text{Linear resolution} = \frac{\text{Stroke per turn}}{\textbf{Counts per revolution [101-102]}} = \frac{200 \text{ mm}}{2,000} = 0.1 \text{ mm}$$

The custom number of revolutions can be as the physical number of revolutions:

$$\text{Custom number of encoder revolutions} = \frac{\textbf{Total Resolution [103-104]}}{\textbf{Counts per revolution [101-102]}} = 16,384$$

- **Total Resolution [103-104]** = 32 768 000

NOTE
Please note that if you set a preset along the path, when the encoder moves back and cross the zero, the value immediately after 0 will be 32 768 000 - 1, i.e. 32 767 999.

...	32,767,997	32,767,998	32,767,999	0	1	2	...
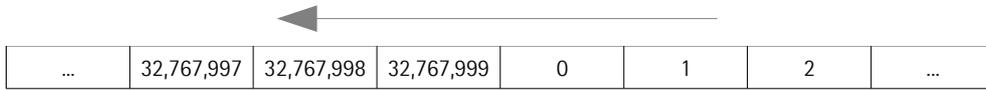
EXAMPLE
Using the values in the previous example let's suppose that the travel in the application is 2 m long. As the stroke per turn is 200 mm you need 10 revolutions to cover the travel length.

- **Total Resolution [103-104]** = **Counts per revolution [101-102]** * custom number of revolutions = 2,000 * 10 = 20,000

In fact:

$$\text{Custom number of encoder revolutions} = \frac{\text{Total Resolution [103-104]}}{\text{Counts per revolution [101-102]}} = 10$$

In this case you will obtain several 20,000 information sections following each other all along the whole measuring length. The position information will be from 0 to 19,999; then again from 0 to 19,999 and so on.

...	19,997	19,998	19,999	0	1	2	...	19,997	19,998	19,999	0	1	2	...

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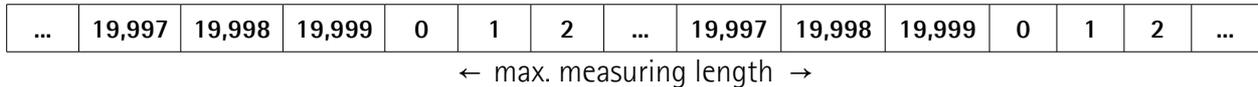
NOTE
We suggest setting values which are a power of 2 (2^n: 2, 4, ..., 2048, 4096, 8192, ...) to be set in the **Counts per revolution [101-102]** and **Total Resolution [103-104]** registers to avoid counting errors.

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 also 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 (i.e. a position in the travel of the wire). 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.

Default = 0 (min. = 0, max. = 134217727 *)

* See the note below.

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 <u>disabled</u> (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 <u>enabled</u> (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, Unsigned16, rw]
These registers define the engineering unit for the velocity value (see the **Speed value [3-4]** registers on page 69).
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, Unsigned16, rw]

Bit	Function	bit = 0	bit = 1
0	**Scaling function**	**disabled**	enabled
1	**Code sequence**	Count up rewinding the wire	**Count up pulling the wire out**
2 ... 31	not used		

Default values are highlighted in bold

Default = 0000 0002 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 device uses the physical resolution to arrange 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:

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

The value in the Counts per revolution [101-102] registers must be less than or equal to the value in the Singleturn resolution [113-114] registers; the total custom resolution in the Total Resolution [103-104] registers must be less than or equal to the maximum physical value Singleturn resolution [113-114] * Number of revolutions [115-116].

NOTE

To know whether the **Scaling function** is currently enabled, you can read the bit 0 **Scaling function** of the Status word [5-6] input registers, see on page 70.

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 set a new preset value (see the Preset value [105-106] registers) and finally save the new parameters (see the **Save parameters** function).

Code sequence
bit 1

This is intended to set whether the count increases (count up information) when you rewind the wire or when you pull the wire out.

If the bit 1 **Code sequence** = 0, the absolute position value **increases** when you rewind the wire; on the contrary, if the bit 1 **Code sequence** = 1, the absolute position value **increases** when you pull the wire out.

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 **Code sequence** is currently enabled, you can read the bit 1 **Code sequence** of the **Status word [5-6]** input registers, see on page 70.

bits 2 ... 7   Not used.

Bytes 1 ... 3   Not used.

Control Word [111-112]

[110-111, Unsigned16, 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 101.

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 <u>temporarily</u> 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 <u>permanently</u> the current encoder position in the **Offset value [127–128]** registers, please execute the **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 **Save parameters** function, see on page 60.

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 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 <u>physical</u> distinguishable steps per each turn provided by the hardware (physical singleturn resolution).
If you want to set a custom resolution see the **Counts per revolution [101-102]** registers.

Default = 8192

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 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 <u>physical</u> distinguishable revolutions provided by the hardware (number of physical revolutions).
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.

Default = 16384

Supported alarms [117-118]

[116-117, Unsigned16, ro]

Bit	Function	bit = 0	bit = 1
0 … 11	not used		
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 ... 11   Not used.

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 7000h (= 00000 0000 0000 0000 0111 0000 0000 0000 = alarms at bits 12, 13 and 14 of the **Alarm registers [121-122]** item are supported).

Supported warnings [119-120]
[118-119, Unsigned16, ro]
These registers contain the information on the warnings supported by the encoder. No warnings are supported in this encoder.
Default = 0

Alarm registers [121-122]
[120-121, Unsigned16, ro]

Bit	Function	bit = 0	bit = 1
0 ... 11	not used		
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.

Byte 0   Not used.

Byte 1

bits 8 ... 11   Not used.

Machine data not valid
bit 12   One or more parameters are not valid, set proper values to restore normal work condition. 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.).

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), normal work status can be restored only after having set proper values. The **Flash memory error** alarm cannot be reset.

Warnings register [123-124]
[122-123, Unsigned16, ro]
This variable is meant to show the warnings currently active in the device. No warnings are supported in this encoder.

Wrong parameters list [125-126]

[124-125, Unsigned16, ro]

The operator has entered invalid data and the **Machine data not valid** alarm 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. Set proper values to restore the normal work condition.

Total resolution error

bit 1   Wrong data has been set next to the **Total Resolution [103-104]** registers. Set proper values to restore the normal work condition.

Preset value error

bit 2   Wrong data has been set next to the **Preset value [105-106]** registers. Set proper values to restore the normal work condition.

Offset value error

bit 3   Wrong data has been saved on the **Offset value [127-128]** registers. Save proper values to restore the normal work condition.

bits 4 … 7   Not used.

Bytes 1 … 3   **Not used.**

Offset value [127-128]

[126-127, Signed32, ro]

As soon as you send 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

DSC Firmware Version [129-130]

[128-129, Unsigned32, ro]

These registers are meant to show the firmware version of the DSC (Digital Signal Controller).

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.

PCB Hardware Version [131-132]

[130-131, Unsigned32, ro]

These registers are meant to show the hardware version of the PCB (Printed Circuit Board).

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: hardware version 1.1.

7.1.2 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 "6.5.1 Implemented function codes" section on page 38.

Current position [1-2]

[000-001, Unsigned32, ro]
These registers are meant to show the current position of the device in the moment in which the request is sent. The output value is scaled according to the set scaling parameters, see **Scaling function** on page 58. 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 51.

WARNING
Please note that the position value issued by the encoder is expressed in counts; thus you have then to convert the number of counts into a linear measuring unit.
To convert the position value into millimetres (mm) or micrometres (μm) you have to multiply the number of information by the linear resolution of the encoder expressed in millimetres or micrometres.
To know the linear resolution of the encoder please consider that **the stroke per turn of the drum is 200 mm**.

The linear resolution results from the following calculation:

$$\textbf{Linear resolution} = \frac{\text{Stroke per turn of the drum}}{\text{Singleturn resolution cpr}}$$

If you want to know the linear position value you will need to multiply the transmitted position value by the linear resolution.

Linear position value = transmitted position * linear resolution

NOTE
Please note that the linear physical resolution of the encoder can be read also in the order code next to the rotary resolution. Refer to the product datasheet.

EXAMPLE 1

Let's suppose that we are using the physical resolution of the SFA-5000-MT-8192-M12 draw-wire encoder (the bit 0 **Scaling function** in the **Operating parameters [109-110]** registers = 0).

The physical singleturn resolution of the measuring device is 8,192 cpr (= 0.024 mm, see the order code in the product datasheet).

As stated, the linear resolution results from the following calculation:

$$\text{Linear resolution} = \frac{\text{Stroke per turn of the drum}}{\text{Resolution cpr}}$$

$$\text{Linear resolution} = \frac{200}{8192} = 0.024 \text{ mm} = 24 \text{ µm}$$

Let's say that the transmitted position value is 123.

Thus the linear position value will be as follows:
Linear position value = transmitted position * linear resolution

Linear position value = 123 * 0.024 = 2.952 mm = 2,952 µm

EXAMPLE 2

Let's suppose that we are using the SFA-5000-MT-8192-M12 draw-wire encoder. The singleturn resolution is set to the custom value of 4,000 cpr (**Counts per revolution [101-102]** = 4000). The transmitted position value is 1,569.

The linear resolution can be easily calculated as follows:

$$\text{Linear resolution} = \frac{200}{4000} = 0.05 \text{ mm} = 50 \text{ µm}$$

Thus the linear position value will be as follows:

Linear position value = 1,569 * 0.05 = 78.45 mm = 78 450 µm

Speed value [3-4]

[002-003, Signed32, ro]

This attribute shows 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 58.

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 51.

Status word [5-6]

[004-005, Unsigned16, 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 58.

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 0 or 1. If the bit is "=0" the output encoder position value has been set to increase (count up information) when you rewind the wire; if the bit is "=1" the output encoder position value has been set to increase when you pull the wire

out. To configure the code sequence 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 59.

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 64.

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 52.

7.2 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.
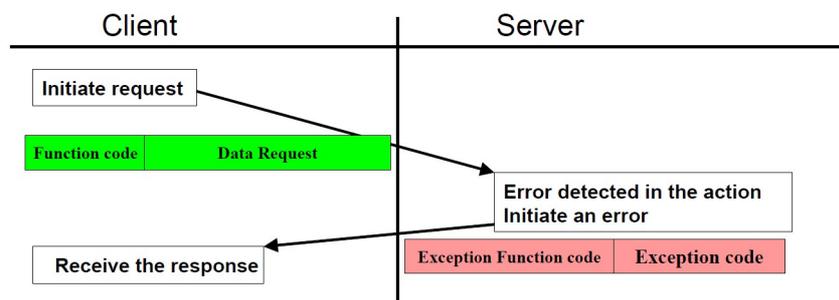
NOTE
Please note that here follows the list the exception codes indicated by MODBUS but not necessarily supported by the manufacturer.

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 47 of the "MODBUS Application Protocol Specification V1.1b3" document.

8 Integrated web server

8.1 Integrated web server – Preliminary information

MODBUS TCP/IP 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 and check the currently set parameters;
- to set the network communication parameters;
- to set some parameters such as the preset and the code sequence;
- to upgrade the firmware;
- to monitor the encoder and access some advanced maintenance functions.

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/IP 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 11.829.17134.0
- Mozilla Firefox version 68.0.1
- Google Chrome version 75.0.3770.142
- Opera version 62.0.3331.72

NOTE
Please note that the snapshot look may vary depending on the used web browser. The following snapshots have been taken from Mozilla Firefox.

8.2 Web server Home page

To open the MODBUS TCP/IP 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 27) in the address bar of your web browser and confirm by pressing **ENTER**;
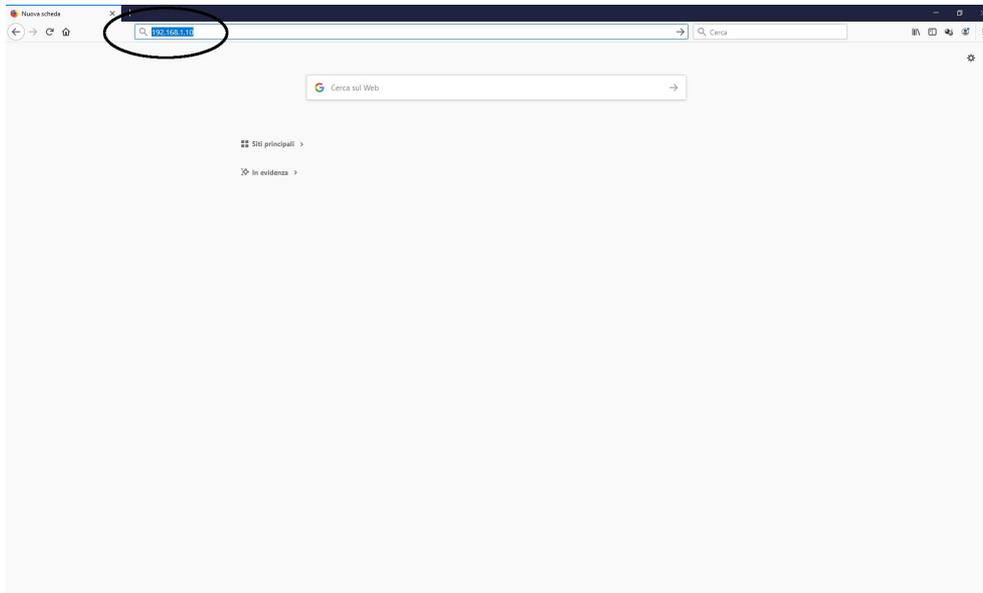
Figure 3 – Opening the web server

2. as soon as the connection is established, the web server Home page will appear on the screen;
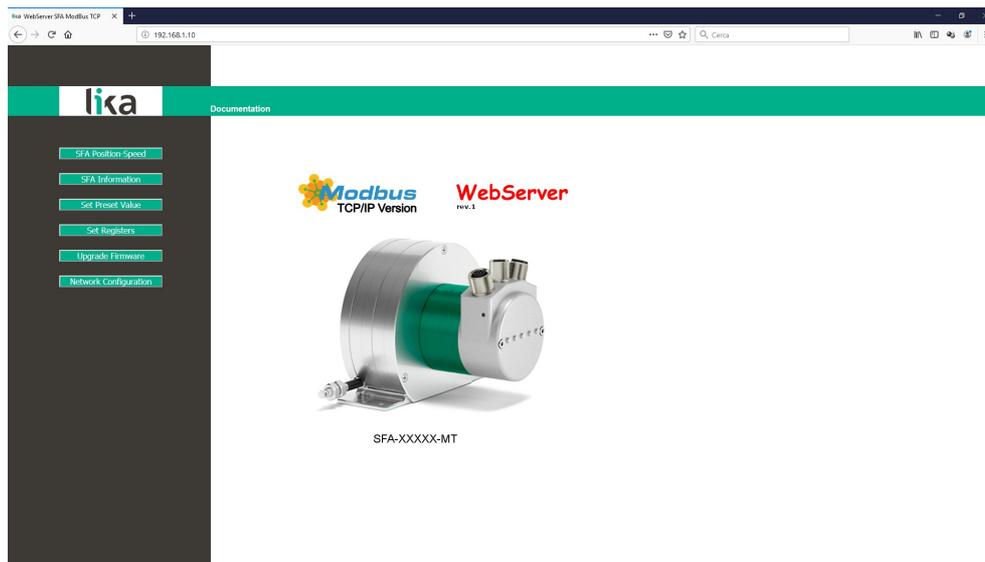
Figure 4 – Web server Home page

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** command to enter the MODBUS TCP/IP encoder technical documentation page available on Lika's web site (http://www.lika.it/eng/prodotti.php?id_cat=267&id_fam=268&id_sfam=294) where specific technical information and documentation concerning the MODBUS TCP/IP encoder can be found.

Furthermore some commands are available in the left navigation bar. All the pages that can be entered through the commands in the bar are freely accessible except the **Upgrade firmware** page that is protected and requires a password to allow access.
These commands 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.

8.3 Draw-wire encoder position and speed

Press the **SFA Position-Speed** command in the left navigation bar of the Web server Home page to enter the page where the <u>current encoder position</u> and the <u>current encoder speed</u> are displayed.
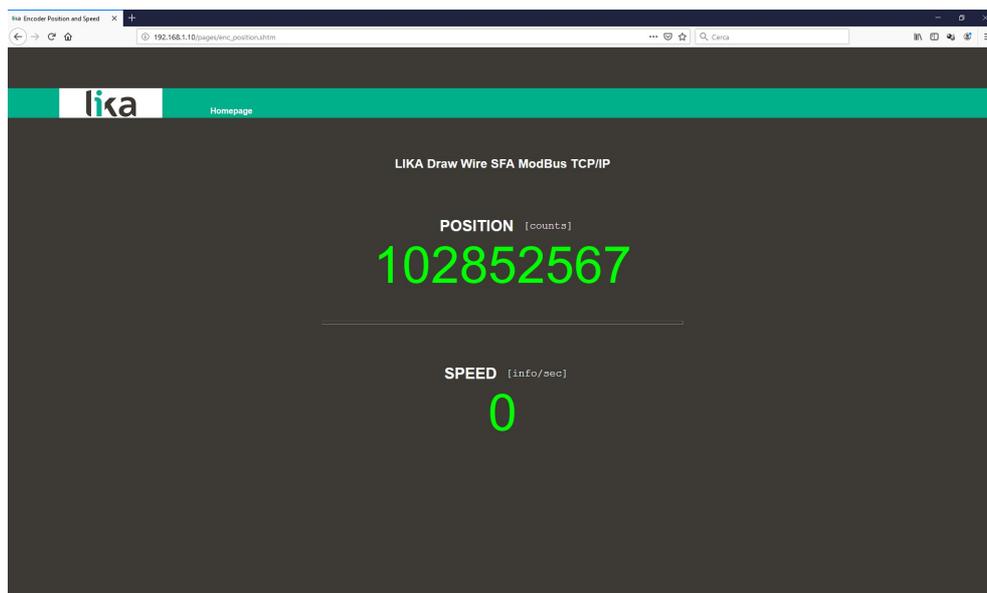
Figure 5 – Draw-wire encoder position and speed page

The current encoder position is expressed in counts. For any information refer to the **Current position [1-2]** registers on page 68.

The current encoder speed is expressed according to the setting next the **Speed format [107-108]** registers on page 58 (steps per second or revolutions per minute). For any information refer to the **Speed value [3-4]** registers on page 69.

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

Press the **Homepage** command to move back to the Web server Home page.

8.3.1 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 continuously updated.

- 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.

8.4 Encoder information (MODBUS registers)

Press the **SFA information** command in the left navigation bar of the Web server Home page to enter the **Draw Wire SFA Information** page. In this page the complete list of the available MODBUS registers is displayed.
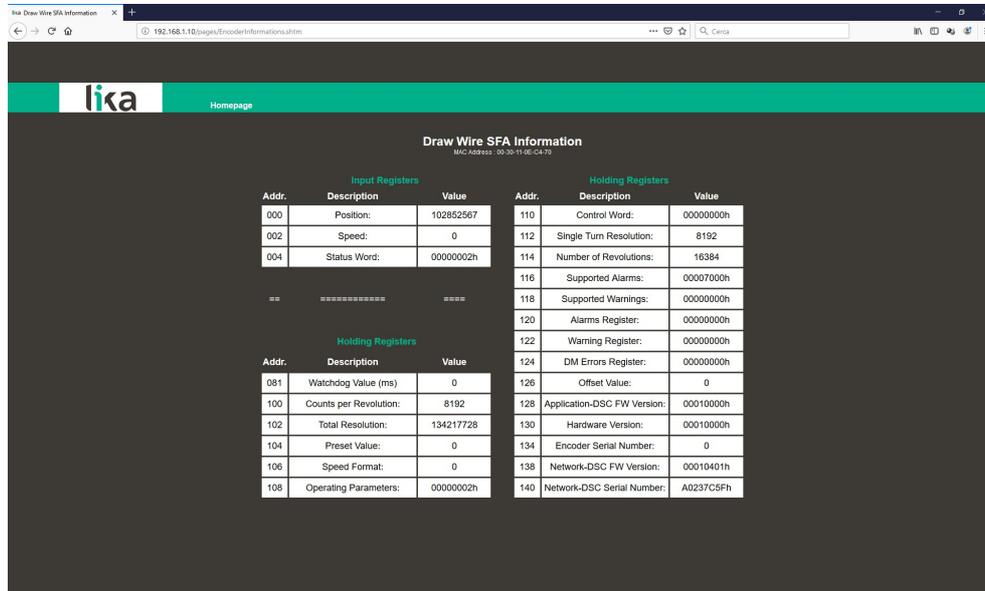
Figure 6 – Encoder Information page

The registers listed under the **Input registers** section are process data and read-only access values. For a complete description of the Input registers please refer to the "7.1.2 Input Register parameters" section on page 68.
The registers listed under the **Holding registers** section 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 "7.1.1 Holding Register parameters" section on page 50.

NOTE
The parameters are made up of two 16-bit registers (except the **Watchdog Value (ms)** which is a single 16-bit register, see the **Watchdog timeout [82]** register on page 51). For such reason only the start address appears under the **ADDR.** column. To read -for instance- the **114 Number of Revolutions** item, 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 **Draw Wire SFA 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 **Draw Wire SFA Information** page cannot be changed even though they are read-write access registers. To change the set values please enter the **Set Draw Wire SFA Registers** page (see on page 82).

Press the **Homepage** command to move back to the Web server Home page.

8.5 Setting the Preset value

Press the **Set Preset Value** command in the left navigation bar of the Web server Home page to enter the **Set Draw Wire SFA 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.

A soon as you press the **Set Preset Value** command a warning message (**Are you sure you want to change Preset Value?**) appears on the screen: it warns the operator about the awkwardness of the operation, thus he is required to confirm the procedure before continuing.
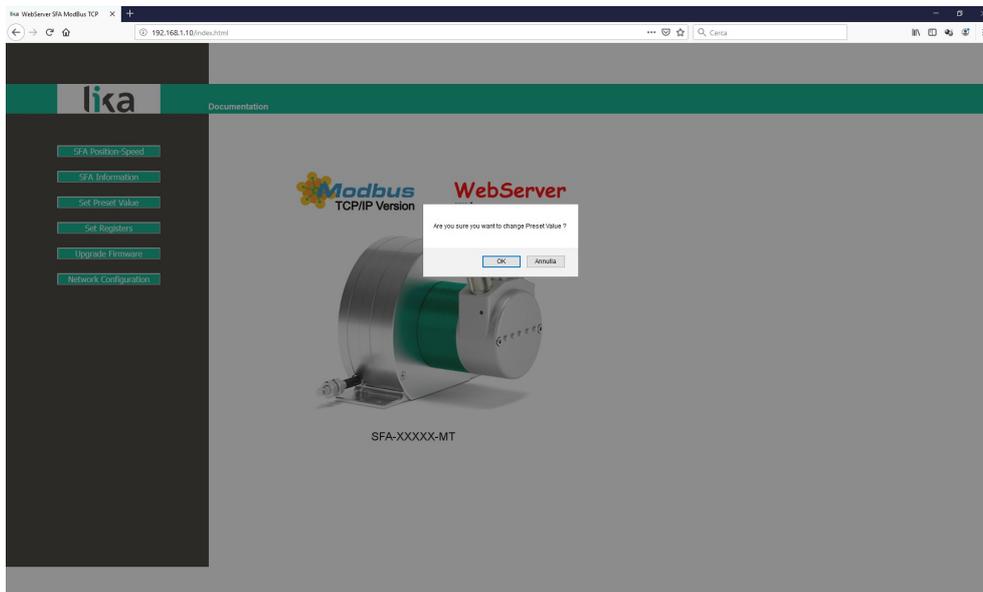
Figure 7 – Entering the Set Draw Wire SFA Preset page

Press the **OK** button to proceed, otherwise press the **EXIT** button to exit the procedure. The **Set Preset cancelled!** message will appear on the screen. Press the **OK** button to move back to the Web server Home page.
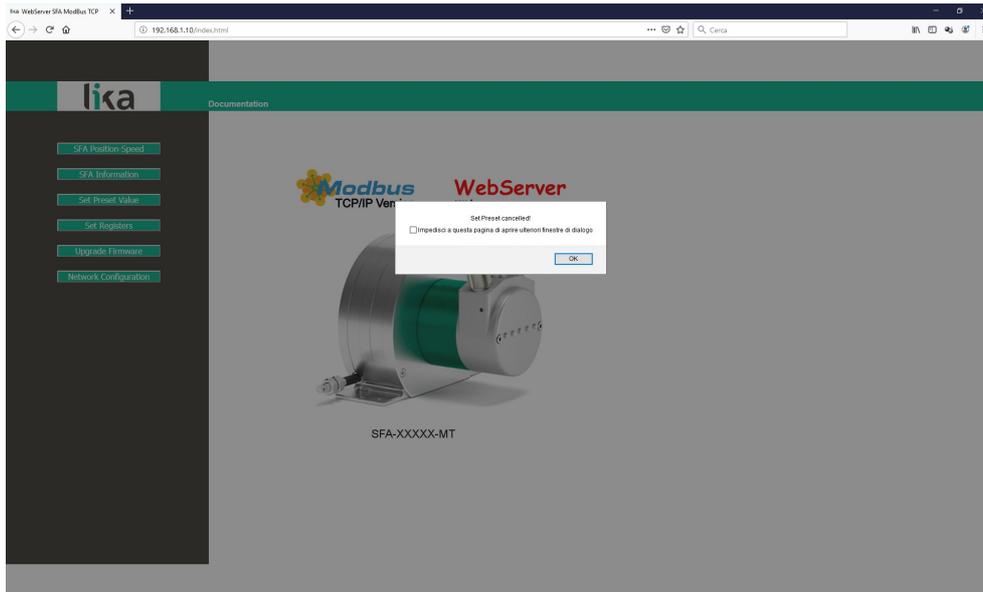
Figure 8 – Preset operation aborted

If you confirm the procedure, the **Set Draw Wire SFA Preset** page will appear on the screen:
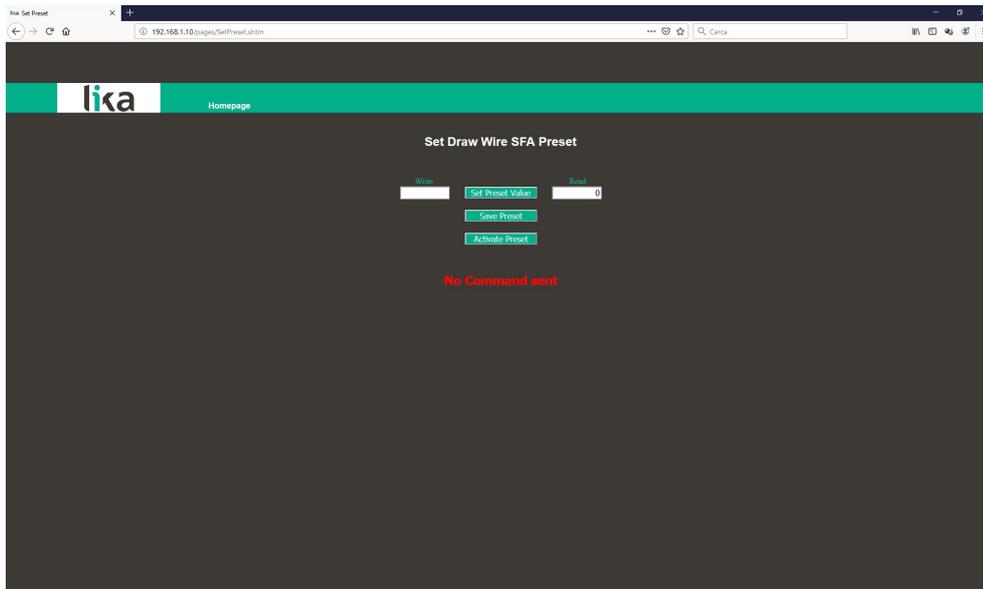
Figure 9 – Set Draw Wire SFA Preset page

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

To change the Preset enter a suitable value in the **WRITE** box and then press the **Set Preset Value** button to confirm. The value has to be set in decimal notation.

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 Preset** button. 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 **Save parameters** command in the **Control Word [111-112]** registers on page 60.

After saving the Preset value, you must activate it (see the **Perform counting preset** command in the **Control Word [111-112]** registers on page 61).
Press the **Activate Preset** button to activate the preset value. The Preset value will be set for the position of the encoder in the moment when the **Activate Preset** button is pressed. We suggest activating the preset value when the encoder is in stop.

NOTE

At each confirmation of the Preset setting and activation, a message will appears in the **No Command sent** line. It informs whether the operation has been accomplished properly or an error occurred.

Press the **Homepage** command to move back to the Web server Home page.

8.6 Setting the registers

Press the **Set Registers** command in the left navigation bar of the Web server Home page to enter the **Set Draw Wire SFA 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 "7.1.1 Holding Register parameters" section on page 50.

A soon as you press the **Set Registers** command a warning message (**Are you sure you want to change Registers Values?**) appears on the screen: it warns the operator about the awkwardness of the operation, thus he is required to confirm the procedure before continuing.
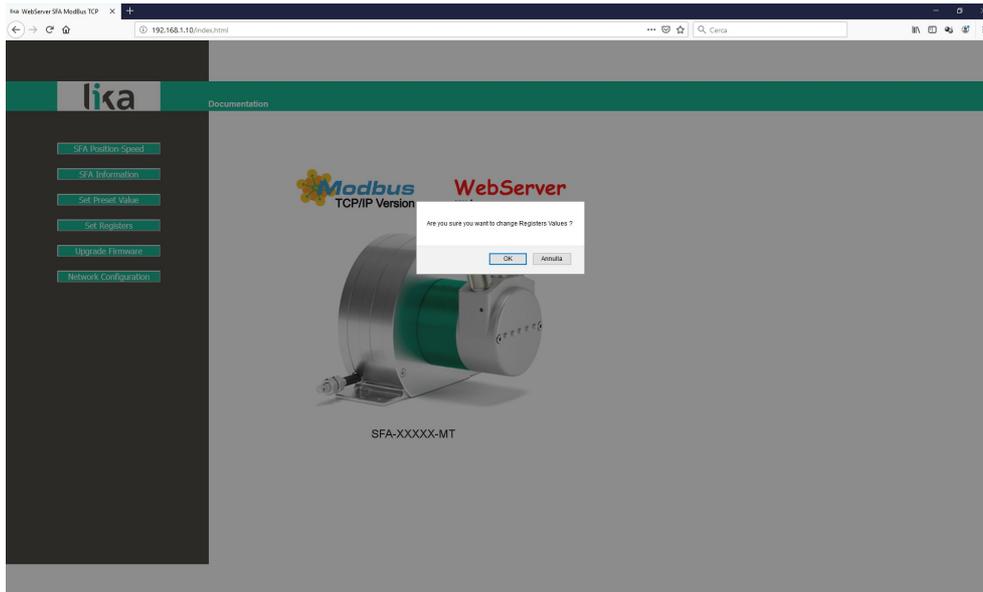
Figure 10 – Entering the Set Draw Wire SFA Registers page

Press the **OK** button to proceed, otherwise press the **EXIT** button to exit the procedure. The **Set Registers cancelled!** message will appear on the screen. Press the **OK** button to move back to the Web server Home page.
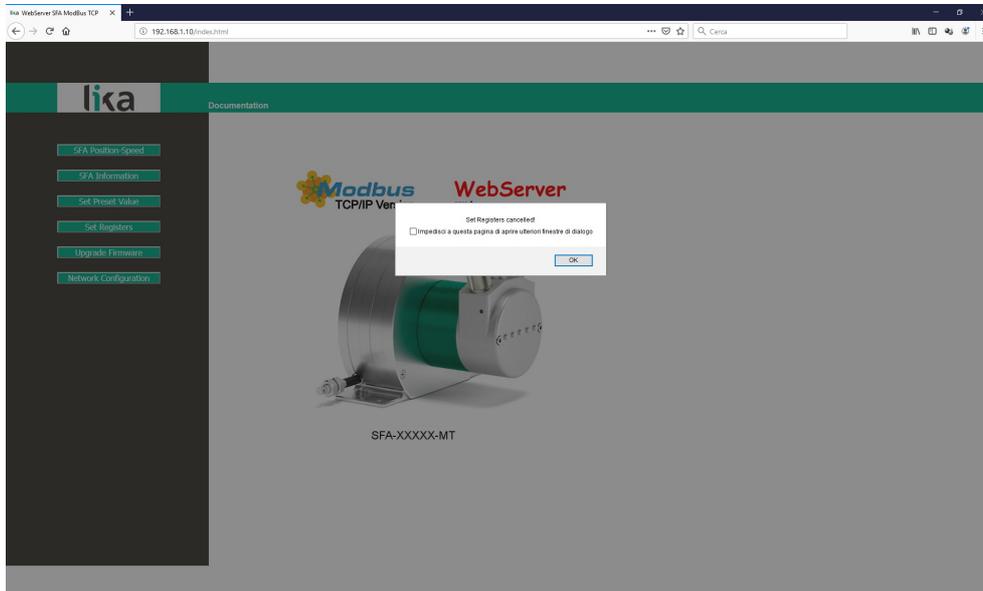

Figure 11 – Register setting operation aborted

If you confirm the procedure, the **Set Encoder Registers** page will appear on the screen:
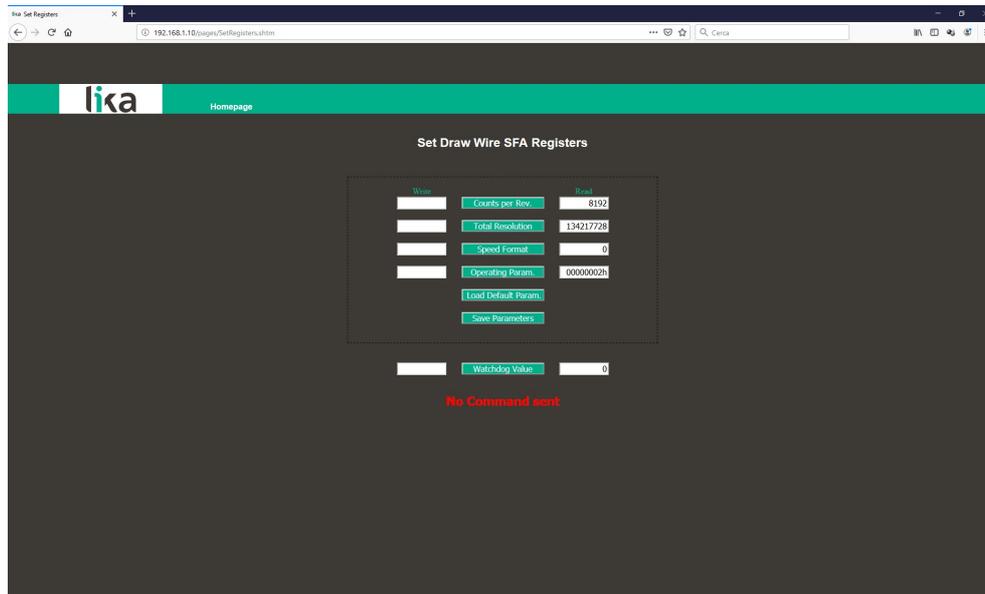
Figure 12 – Set Draw Wire SFA Registers page

The values that are currently set in the encoder are displayed in the **READ** box.

To change any value enter a suitable value in the **WRITE** box next to the desired parameter and then press the button between the boxes to confirm. The values have to be set in decimal notation.
For complete information on the available registers please refer to the "7.1.1 Holding Register parameters" section on page 50.

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

NOTE
Please note that, after pressing the button between the boxes, 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 **Save parameters** command in the Control Word [111-112] registers on page 60.

NOTE

The Watchdog Value parameter only is saved automatically after setting.

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 **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!

NOTE

At each confirmation of the set registers, a message will appears in the No Command sent line. It informs whether the operation has been accomplished properly or an error occurred.

Press the **Homepage** command to move back to the Web server Home page.

8.7 Firmware upgrade

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

WARNING
Firmware upgrading process has to be accomplished by skilled and competent personnel. It is mandatory to perform the upgrade according to the instructions provided in this section.
Before installation always ascertain that the firmware program is compatible with the hardware and software of the device. Furthermore never turn off the power supply during the flash upgrade operation.

This operation allows to upgrade the unit firmware by downloading upgrading data to the flash memory.
The firmware is a software program which controls the functions and the 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 .BIN 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 Firmware Version parameter in the **Draw Wire SFA Information** page after connection to the web server (see on page 79; see also the DSC Firmware Version [129-130] registers on page 67).

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

Before proceeding with the firmware upgrade 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 both to the network and the IO controller;

3. Press the **OK** button to proceed, otherwise press the **EXIT** button to exit the procedure. The **Firmware upgrade cancelled!** message will appear on the screen. Press the **OK** button to move back to the Web server Home page.


Figure 14 – Firmware upgrade operation aborted

4. If you confirm the procedure, the **Firmware Upgrade** page will appear on the screen: the operator is requested to submit a password before starting the firmware upgrade procedure.
5. In the **Password** text box type the password **LIKA** (all uppercase letters) and then press the **Send Request** button.
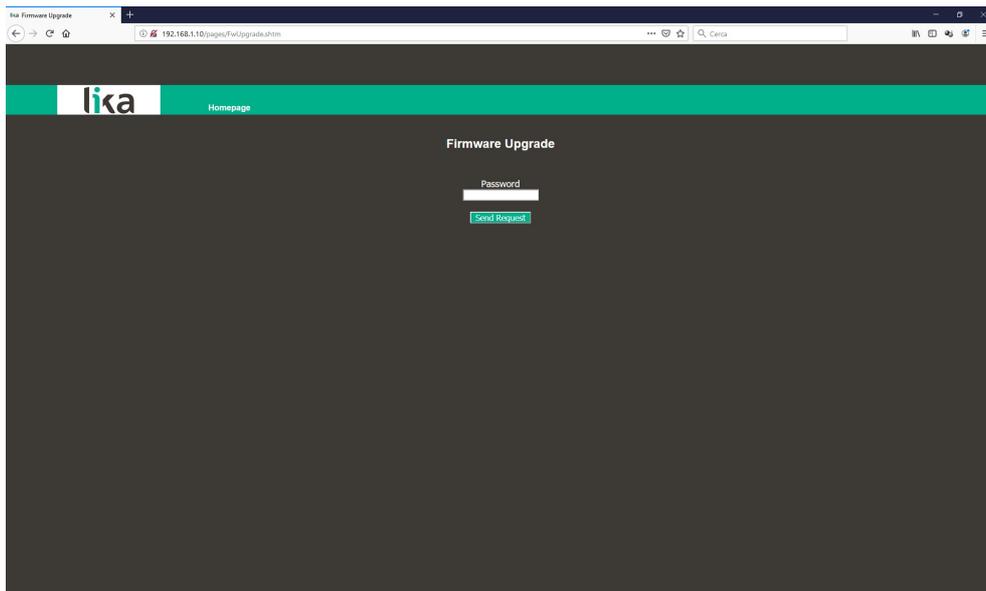

Figure 15 – Firmware Upgrade page

6. If the password you typed is wrong, the following warning message will appear on the screen: **THE PASSWORD INSERTED IS INCORRECT. PLEASE RETRY!**. Please retype the password and confirm.

7. If the password you typed is correct, the following message will appear on the screen: **THE PASSWORD INSERTED IS CORRECT. THE WEB SERVER OF THE ENCODER IS STOPPED. NOW LAUNCH THE PROGRAM SW_ETH_REVX_Y.EXE**.

8. The encoder is now ready to accept the firmware program: the web server is stopped and the communication with the encoder through the web browser is interrupted; if you need to exit the procedure and restore the communication you must switch the encoder off and then on again.

9. Now you must launch the SW_ETH_REVX_Y.EXE executable file provided by Lika Electronic to continue with the procedure; X and Y indicate the version of the firmware upgrading program: REV1_0 is the version 1.0.

10. Launch the SW_ETH_REVX_Y.EXE executable file; the following page will appear:
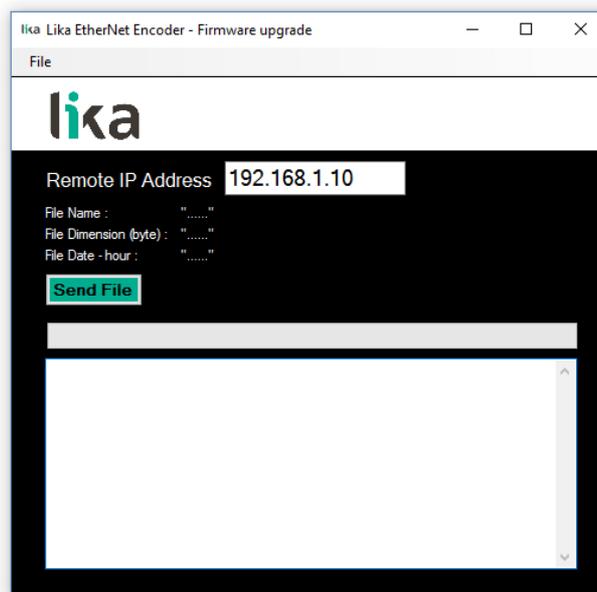
Figure 16 – Firmware upgrade executable file

11. Type the encoder IP address in the **Remote IP Address** box. The default IP address set by Lika Electronic is 192.168.1.10.

12. Press the **FILE** command and then the **OPEN** command in the menu bar; once you press the **OPEN** command the **OPEN** dialog box appears on the screen: open the folder where the firmware upgrading .BIN file released by Lika Electronic is located, select the file and confirm. Hx in the file name shows the hardware version of the PCB; Sx shows the software version of the firmware upgrading file.

WARNING

Please pay attention to install the BIN file that perfectly matches the series of the encoder to be updated.

SFA_XXXXX_MT_Hx_Sx.bin   for SFA MODBUS/TCP series
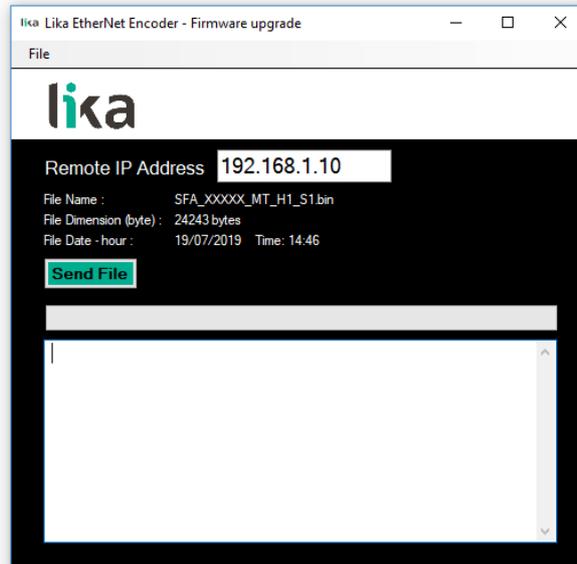
Figure 17 – Selecting the firmware upgrade .BIN file

13. Some properties of the selected file are shown next to the relevant labels in the page: **File Name**, **File Dimension (byte)**, **File Date – hour**. Please check the file properties and ascertain that you are installing the correct upgrade 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 upgrade operation.

14. Press the **Send File** button to start the firmware upgrade process.

15. A download progress bar as well as additional information are shown in the page while upgrading the firmware.
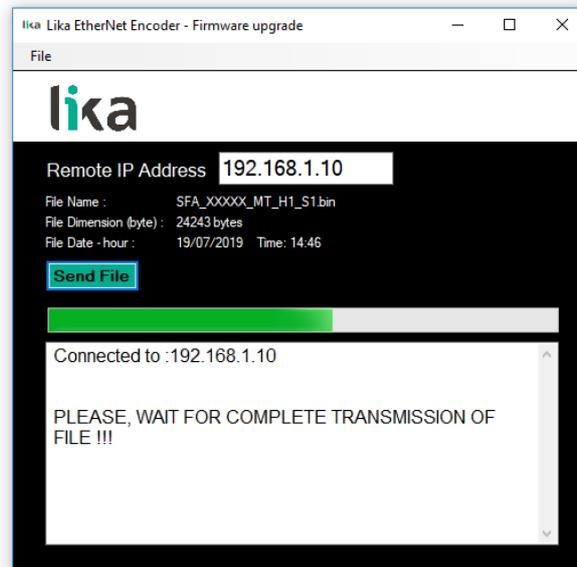

Figure 18 – Updating the firmware

16. As soon as the operation is carried out successfully, the **FILE SENT CORRECTLY** message appears on the screen.
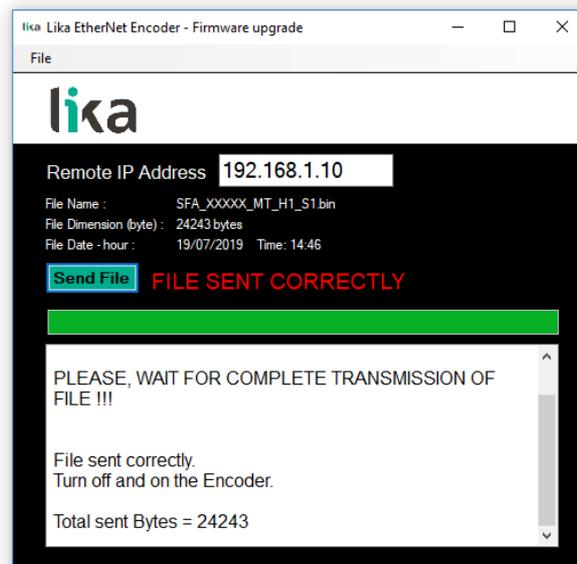

Figure 19 – Firmware upgrade process accomplished

17. Now you are required to turn the encoder power supply off and then on. Close the program.
18. Turn the encoder power supply off and then on to complete the operation.

NOTE
While downloading the firmware upgrading 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; error messages are displayed. In case of flash upgrade error, please switch the encoder off and then on again and retry the operation.

Press the **Homepage** command to move back to the Web server Home page.

8.8 Network configuration

Press the **Network Configuration** command in the left navigation bar of the Web server Home page to enter the **Network IP Configuration** 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 "4.5 Setting the IP address and the network configuration parameters" section on page 27.

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

A soon as you press the **Network Configuration** command a warning message (**Are you sure you want to change Network Parameters?**) appears on the screen: it warns the operator about the awkwardness of the operation, thus he is required to confirm the procedure before continuing.
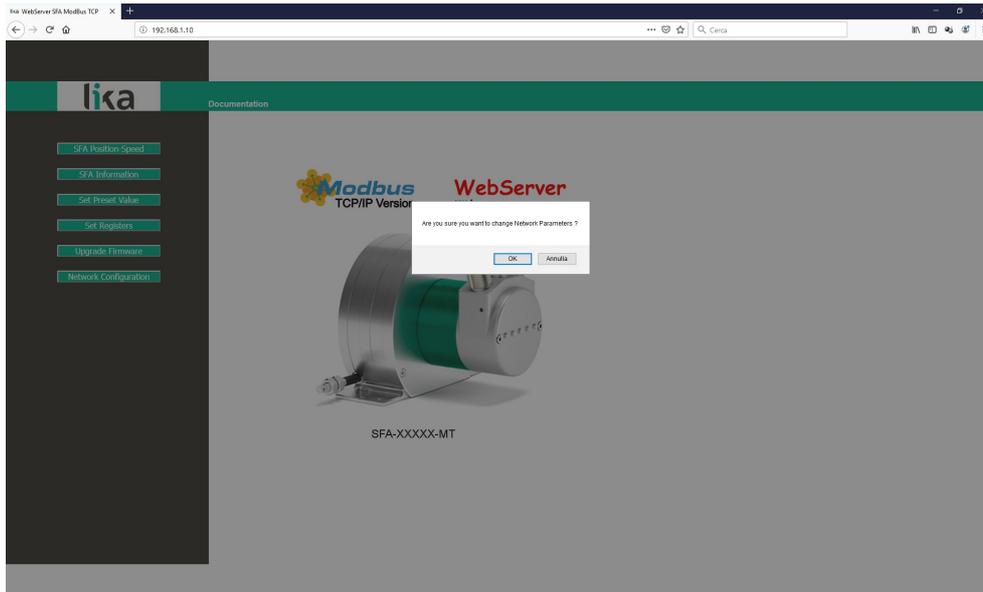
Figure 20 – Entering the Network Configuration page

Press the **OK** button to proceed: the **Network Configuration** page will appear on the screen.
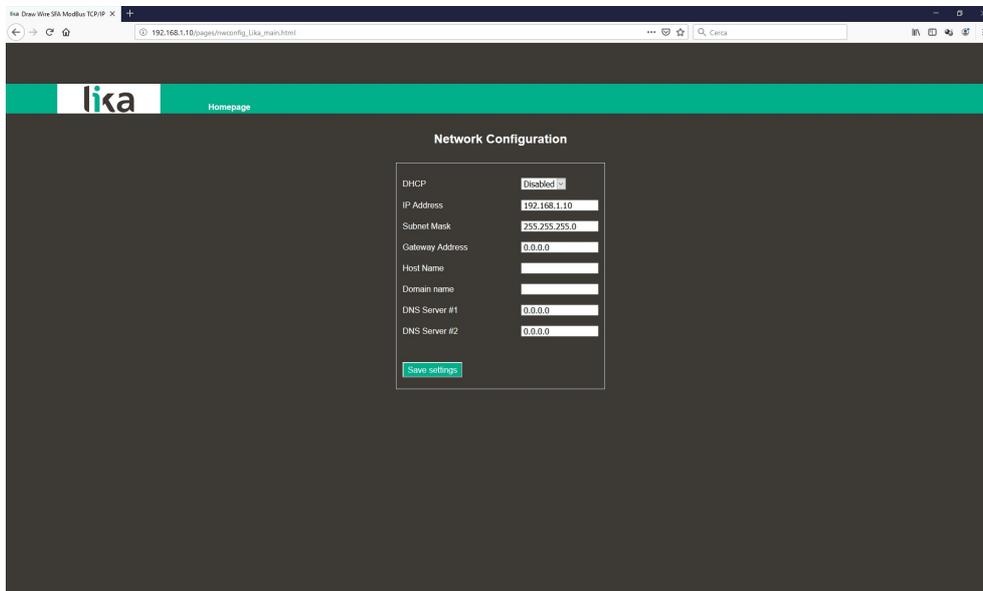

Figure 21 – Network Configuration page

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, DHCP, DNS, etc.

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

IP Parameter	Value
DHCP	Disabled
IP address	192.168.1.10
Subnet mask	255.255.255.0
Gateway Address	0.0.0.0
Host name	
Domain name	
DNS Server #1	0.0.0.0
DNS Server #2	0.0.0.0

To save the set values <u>permanently</u>, please press the **Save Settings** button below in the 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 (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 "4.8 DIP A: Resetting the network configuration parameters to the factory values" section on page 29.

WARNING
If you enable the DHCP network protocol (DHCP = ENABLED), then the following default parameters are set for the encoder:
IP ADDRESS = 0.0.0.0
SUBNET MASK = 0.0.0.0
Please check that these settings are allowed by the DHCP server and they are valid address values.

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 "4.8 DIP A: Resetting the network configuration parameters to the factory values" section on page 29.

Press the **Homepage** command to move back to the Web server Home page.

9 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/IP ADU (MBAP Header + PDU) refer to the "6.3 MODBUS on TCP/IP Application Data Unit" section on page 34.

9.1 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][00][05][DC]
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)
[00][00] = value of register 105, 00 00 hex = 0 dec
[05][DC] = value of register 106, 05 DC hex = 1500 dec

The **Preset value [105-106]** registers (address 104-105) contain the value 00 00 hex and 05 DC hex, i.e. 1500 in decimal notation; in other words the value set in the **Preset value [105-106]** registers is 1500 dec.

9.2 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][00][00][2F][F0]
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)
[00][00] = value of register 1, 00 00 hex = 0 dec
[2F][F0] = value of register 2, 2F F0 hex = 12272 dec

The **Current position [1-2]** registers (address 0-1) contain the value 00 00 2F F0 hex, i.e. 12272 in decimal notation.

9.3 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.

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.

9.4 Using the 16 Write Multiple Registers function code

EXAMPLE 1

Request to write the value 00 00 08 00 hex (=2048 dec) next to the **Counts per revolution [101-102]** registers (address 100-101) and the value 00 80 00 00 hex (= 8388608 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][00][00][08][00][00][80][00][00]
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)
[00][00] = value to be set in the register 101, 00 00 hex
[08][00] = value to be set in the register 102, 08 00 hex (00 00 08 00 hex = 2048 dec)
[00][80] = value to be set in the register 103, 00 80 hex
[00][00] = value to be set in the register 104, 00 00 hex (00 80 00 00 hex = 8388608 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. 2048 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. 8388608 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 2048-count-per-revolution single-turn resolution and 4096 revolutions (8388608/2048).

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 when you rewind the wire (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][00][00][01]
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][00] = value to be set in the register 109, 00 00 hex
[00][01] = value to be set in the register 110, 00 01 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.

10 Default parameters list

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

Parameters list	Default values		
Watchdog timeout [82]	0		
Counts per revolution [101-102]	8192		
Total Resolution [103-104]	134217728		
Preset value [105-106]	0		
Speed format [107-108]	0 = steps/s		
Operating parameters [109-110]	0000 0002 hex		
	bit 0 Scaling function = 0		
	bit 1 Code sequence = 1		
Singleturn resolution [113-114]	8192		
Number of revolutions [115-116]	16384		
Supported alarms [117-118]	0000 7000 hex		
Supported warnings [119-120]	0		

This page intentionally left blank

This page intentionally left blank

Document release	Release date	Description	HW	SW	Interface
1.0	22.07.2019	First issue	1.0	1.0	-