User's guide

EM58x EMC58

Modbus

RS-485 version

lika

Smart encoders & actuators

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 Lika device and interface are coloured in ORANGE;
- 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.
ℹ	This icon is meant to highlight the parts of the text where suggestions useful for making it easier to set the device and optimize performance and reliability can be found. Sometimes this symbol is followed by the word **EXAMPLE** when instructions for setting parameters are accompanied by examples to clarify the explanation.

Preliminary information

This guide is designed to provide the most complete information the operator needs to correctly and safely install and operate the **EMx58 series multiturn absolute encoders fitted with Modbus interface**.

To make it easier to read the text, this guide can be divided into two main sections.
In the first section general information concerning the safety, the mechanical installation and the electrical connection as well as tips for setting up and running properly and efficiently the unit are provided.
While in the second section, entitled **Modbus Interface**, both general and specific information is given on the Modbus interface. In this section the interface features and the registers implemented in the unit are fully described.

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

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 s.r.l. assumes no liability for the customer's failure to comply with these requirements.

1.2 Electrical safety

- Turn OFF power supply before connecting the device;
- connect according to explanation in section "Electrical connections";
- in compliance with 2004/108/EC norm on electromagnetic compatibility, following precautions must be taken:
  - before handling and installing the equipment, discharge electrical charge from your body and tools which may come in touch with the device;
  - power supply must be stabilized without noise; install EMC filters on device power supply if needed;
  - always use shielded cables (twisted pair cables whenever possible);
  - avoid cables runs longer than necessary;
  - avoid running the signal cable near high voltage power cables;
  - mount the device as far as possible from any capacitive or inductive noise source; shield the device from noise source if needed;
  - to guarantee a correct working of the device, avoid using strong magnets on or near by the unit;
  - minimize noise by connecting the shield and/or the connector housing and/or the 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.

1.3 Mechanical safety

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

2 Identification

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

3 Mechanical installation

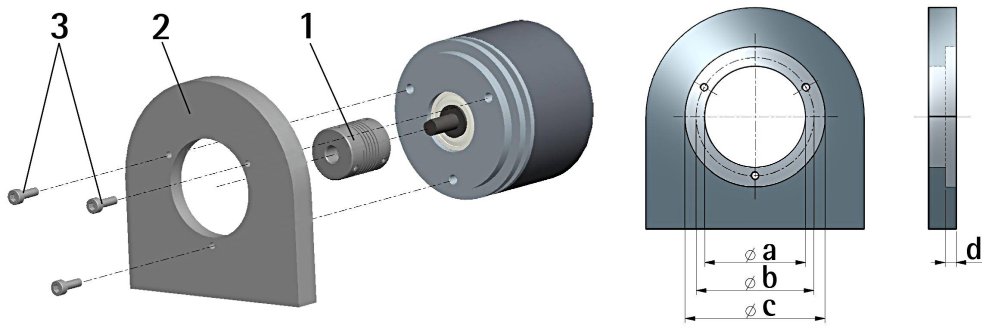
WARNING
Installation and maintenance operations have to be carried out by qualified personnel only, with power supply disconnected. Shaft and mechanical components must be in stop.

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

3.1 Solid shaft encoders

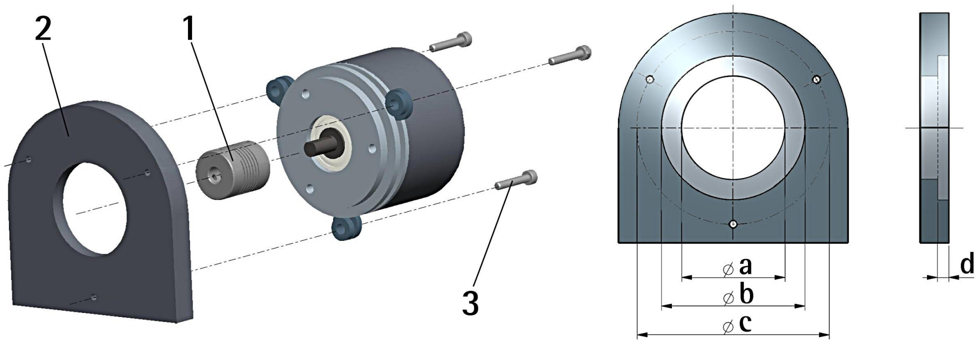
- Mount the flexible coupling **1** on the encoder shaft;
- fix the encoder to the flange **2** (or to the mounting bell) by means of the screws **3**;
- secure the flange **2** to the support (or the mounting bell to the motor);
- mount the flexible coupling **1** on the motor shaft;
- make sure the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

3.1.1 Customary installation



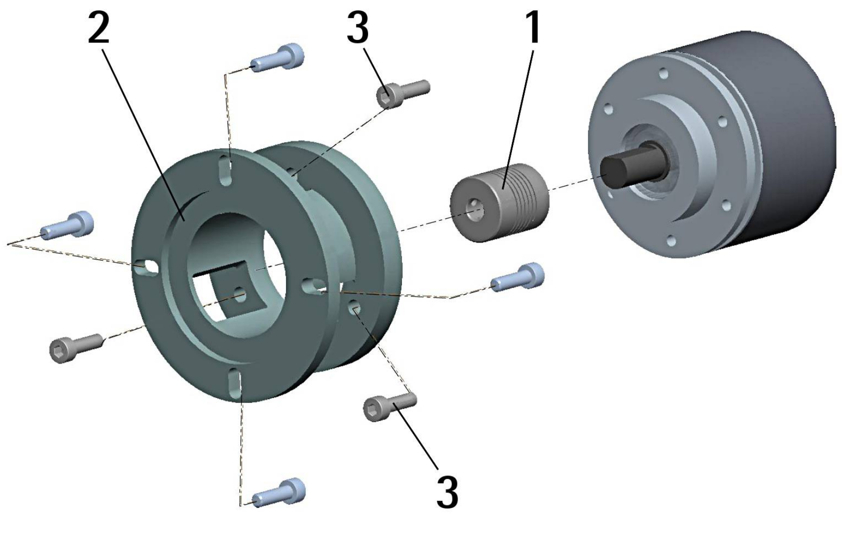
	a [mm]	b [mm]	c [mm]	d [mm]
EM58	-	42	50 F7	4
EM58S	36 H7	48	-	-

3.1.2 Installation using fixing clamps (code LKM-386)



	a [mm]	b [mm]	c [mm]	d [mm]
EM58	-	50 F7	67	4
EM58S	36 H7	-	67	-

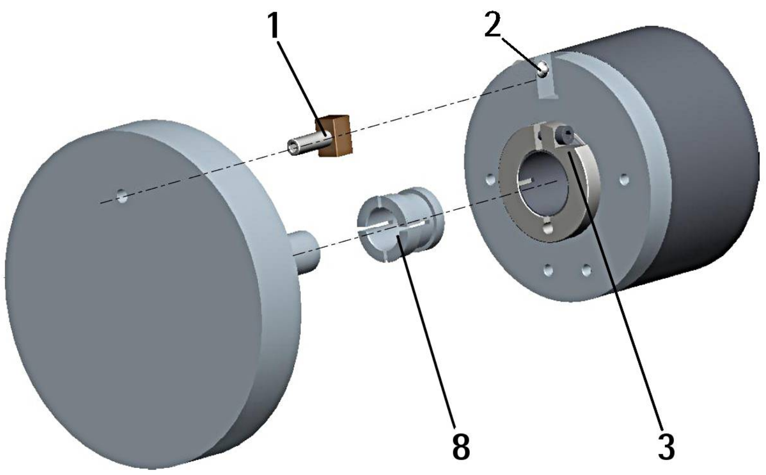
3.1.3 Installation using a mounting bell (code PF4256)

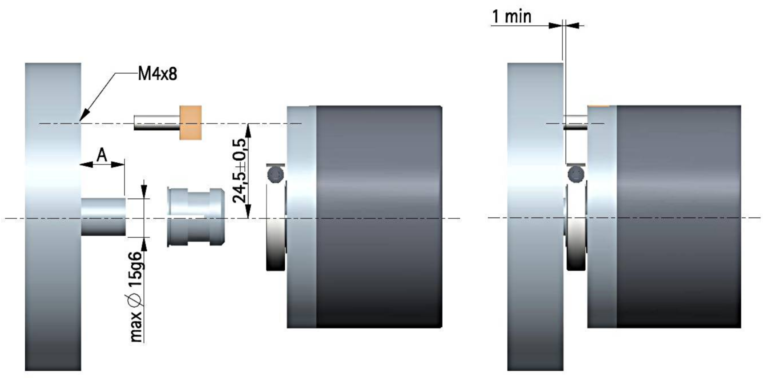


3.2 Hollow shaft encoders

3.2.1 EMC58

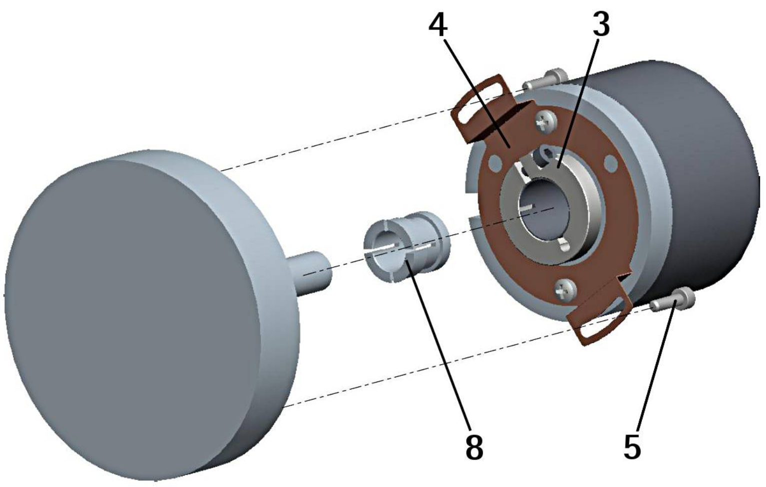
- Fasten the anti-rotation pin **1** to the rear of the motor (secure it using a locknut);
- mount the encoder on the motor shaft using the reducing sleeve **8** (if supplied). Avoid forcing the encoder shaft;
- insert the anti-rotation pin **1** into the slot on the flange of the encoder; this secures it in place by grub screw **2**, preset at Lika;
- fix the collar **3** to the encoder shaft (apply threadlocker to screw **3**).

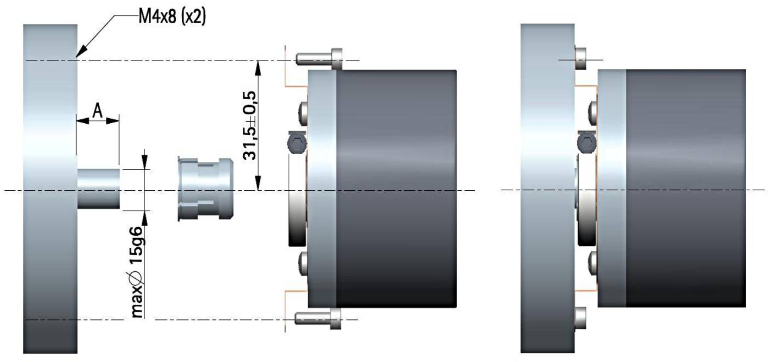




3.2.2 EMC59

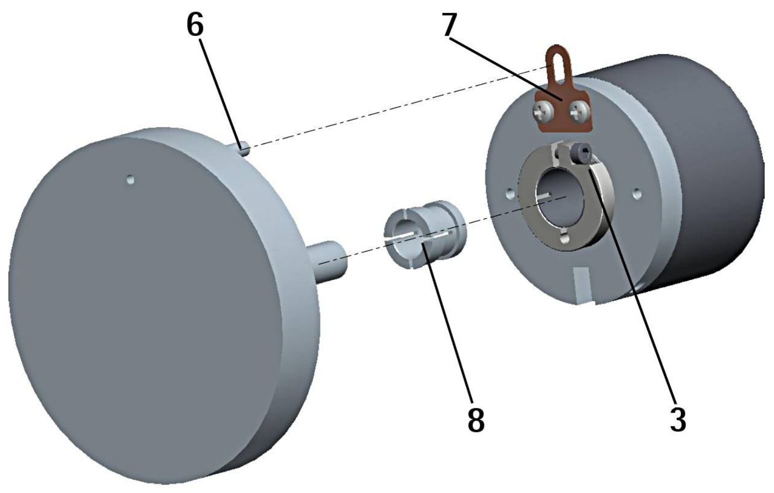
- Mount the encoder on the motor shaft using the reducing sleeve **8** (if supplied). Avoid forcing the encoder shaft;
- fasten the fixing plate **4** to the rear of the motor using two M3 cylindrical head screws **5**;
- fix the collar **3** to the encoder shaft (apply threadlocker to screw **3**).

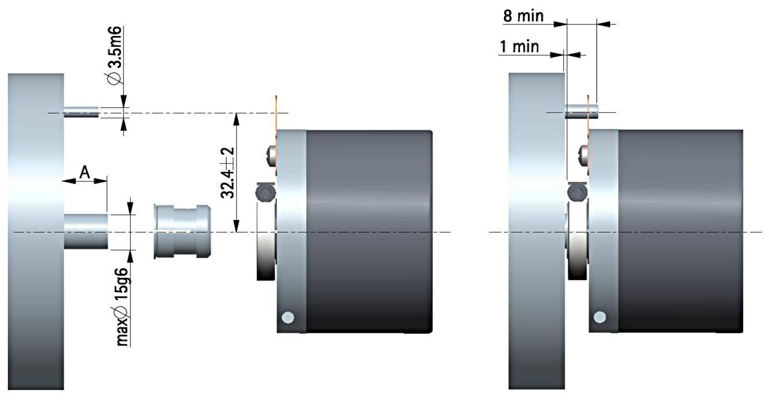




3.2.3 EMC60

- Fix the tempered pin **6** to the rear of the motor;
- mount the encoder on the motor shaft using the reducing sleeve **8** (if supplied). Avoid forcing the encoder shaft;
- make sure the anti-rotation pin **6** is inserted properly into the fixing plate **7**;
- fix the collar **3** to the encoder shaft (apply threadlocker to screw **3**).





4 Electrical connections

WARNING
Power supply must be turned off before performing any electrical connection! Never force manually the rotation of the shaft not to cause permanent damages!

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

4.1 Cable

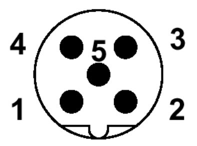
Colour	Description
Red	+10VDC +30VDC supply voltage
Black[1]	0VDC supply voltage
White	Modbus A (RS-485)
Blue	Modbus B (RS-485)

1 0VDC of the RS-485 serial port too.

Cable specifications

Model	: LIKA CB type cable
Wires	: 2 x 0,24 mm^2 + 2 x 0,38 mm^2 twisted pair cable
Shield	: copper braid
External diameter	: Ø 6,1 – 7,1 mm ±5% (0.24" - 0.28" ±5%)
Impedance at 1 MHz	: 120 Ω ±10%

4.2 M12 5-pin connector



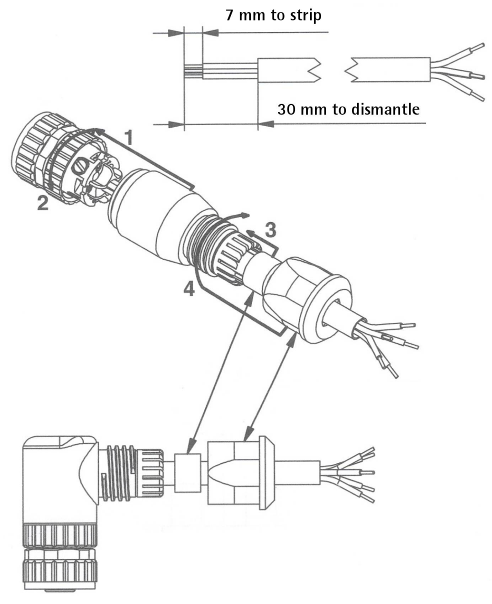
M12 5-pin
male connector
A coding
(frontal view)

Pin	Description
1[1]	Shielding
2	+10VDC +30VDC supply voltage
3[2]	0VDC supply voltage
4	Modbus A (RS-485)
5	Modbus B (RS-485)
Case[3]	Shielding

1 Pin 1 is intended to allow the connection of the shield to ground even if the plug connector has a plastic case.
2 0VDC of the RS-485 serial port too.
3 Lika's EC- pre-assembled cables only

4.3 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. In the specific case pin 1 of M12 connector is specifically intended to allow the connection of the shield to ground. 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.


7 mm to strip
30 mm to dismantle
1
2
3
4

4.4 Diagnostic LEDs (Figure 1)

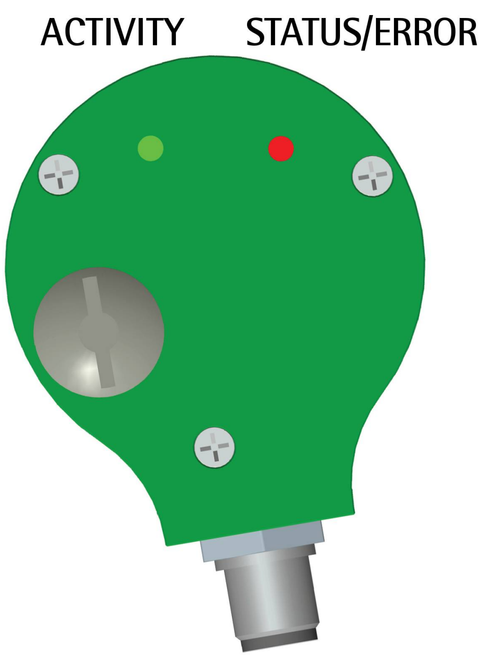


Figure 1: Diagnostic LEDs

Two bi coloured LEDs located in the rear of the encoder (see Figure here above) are meant to show visually the operating or fault status of the Modbus interface and the device as well. The meaning of each LED is explained in the following table:

ACTIVITY LED	Description
Blinking GREEN	Indicates the device is sending or receiving a message.
OFF	Indicates there is no send - receive activity.

STATUS/ERROR LED	Description
Blinking RED	Indicates there are either active alarms or an internal error. For further information refer to Alarms register [0x00] on page 50.
ON RED	An hardware error has occurred which prevents the unit from continuing to run. Please turn the device off and then on again. If error continues, please contact Lika Electronic's After Sales Service.
ON GREEN	No active alarm.

ACTIVITY LED	STATUS/ERROR LED	Description
Blinking GREEN	Blinking GREEN	While downloading data to the flash memory for upgrading the firmware of the unit (see section "Upgrade Firmware"), both LEDs blink green at 5 Hz.
Blinking RED	Blinking RED	While downloading data to the flash memory for upgrading the firmware of the unit (see section "Upgrade Firmware"), if an error occurs which stops the upgrading process (for instance: a voltage drop and/or the switching off of the unit), as soon as the power is turned on again both LEDs start blinking red at 5 Hz as the user program is not installed in the flash memory (it has been deleted previously). For any information on restoring the unit please refer to the section "Upgrade Firmware".
ON RED	ON RED	While downloading data to the flash memory for upgrading the firmware of the unit (see section "Upgrade Firmware"), if data transmission is cut off (for instance, because of the disconnection of the serial cable), after 5 seconds both LEDs come on solidly red. For any information on restoring the unit please refer to the section "Upgrade Firmware".

During initialisation, system checks the diagnostic LEDs for proper operation; therefore they blink for a while.

4.5 Dip-Switches (Figure 2 and Figure 3)

WARNING
Power supply must be turned off before performing this operation!

NOTE
When performing this operation be careful not to damage the internal components and the connection wires.

To access DIP-Switches loosen and remove the M12 metal screw plug in the rear of the encoder. Be careful to replace the screw plug at the end of the operation.

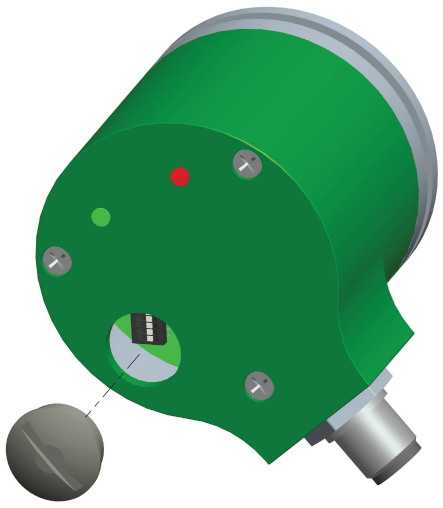


Figure 2: Reaching the DIP-Switches

DIP-Switches are located just beneath.

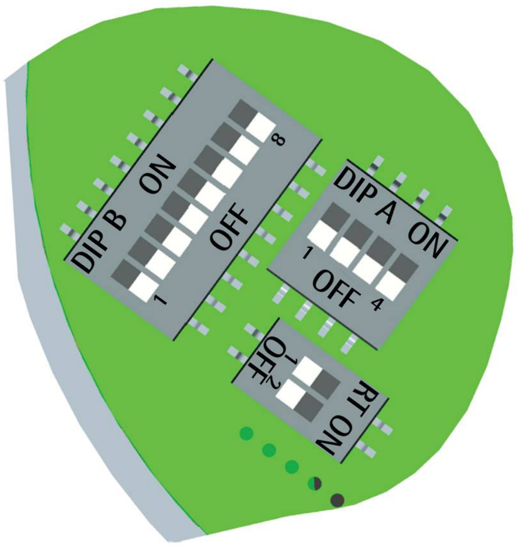


Figure 3: DIP-Switches

4.5.1 Setting data transmission rate: Baud rate and Parity bit (Figure 3)

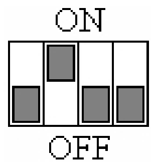
WARNING
Power supply must be turned off before performing this operation!

Use the DIP-Switch A to set the data transmission rate (baud rate and parity bit). Set the binary value of the baud rate and the parity bit according to the following table, considering that: ON = 1; OFF = 0.

Switch	Baud rate	Parity bit
0000	9600 bit/s	No parity
1000	9600 bit/s	Even
0100	9600 bit/s	Odd
1100	19200 bit/s	No parity
0010 (default)	19200 bit/s	Even
1010	19200 bit/s	Odd
0110	115200 bit/s	No parity
1110	115200 bit/s	Even
0001	115200 bit/s	Odd

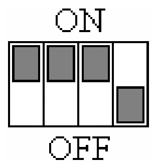
Example
Set the baud rate to 9600 bits per second and Odd parity bit:

Switches	**1**	**2**	**3**	**4**
Position	OFF	ON	OFF	OFF
Value	0	1	0	0



Set the baud rate to 115200 bits per second and Even parity bit:

Switches	**1**	**2**	**3**	**4**
Position	ON	ON	ON	OFF
Value	1	1	1	0



The data transmission rate which is currently set for the unit can be read next to the **Dip-switch baud rate [0x06]** register, see on page 52.

4.5.2 Setting the node address (Figure 3)

WARNING
Power supply must be turned off before performing this operation!

Use the DIP-Switch B to set the node address.
Set the binary value of the node address, considering that: ON = 1; OFF = 0.

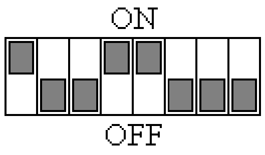
bit	1 LSB	2	3	4	5	6	7	8 MSB
	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7

The range of node addresses is between 1 and 247. The default address is 1.

Example

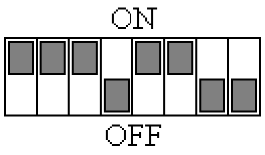
Set the node address to 25:
25_{10} = 0001 1001_2 (binary value)

Switches	1	2	3	4	5	6	7	8
Position	ON	OFF	OFF	ON	ON	OFF	OFF	OFF
Value	1	0	0	1	1	0	0	0



Set the node address to 55:
55_{10} = 0011 0111_2 (binary value)

Switches	1	2	3	4	5	6	7	8
Position	ON	ON	ON	OFF	ON	ON	OFF	OFF
Value	1	1	1	0	1	1	0	0



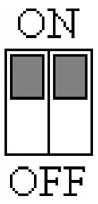
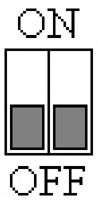
NOTE
The default address is 1.
The address 0 is reserved to identify a "broadcast" exchange (Master sends a request to all Slaves connected to the Modbus network). See section "6.1 Modbus Master / Slaves protocol principle" on page 27.
The Modbus Master node has no specific address, only the Slave nodes must have an address. Each Slave must have a unique address.
Addresses from 248 to 255 are reserved.

If you set the address 0, device will be set to 1 automatically.
If you set an address higher than 247, device will be set to 247 automatically.
The node address which is currently set for the unit can be read next to the **Dip-switch node ID [0x07]** register, see on page 53.

4.5.3 Bus termination (Figure 3)

WARNING
Power supply must be turned off before performing this operation!

Use the RT DIP-Switch to activate or deactivate the bus termination. The bus termination resistor must be activated as line termination in the last device of the transmission line (both at the beginning and the end of the communication bus).

RT	Description
1 = 2 = ON 	Activated: when the device is at the end of the transmission line
1 = 2 = OFF 	Deactivated: when the device is not at the end of the transmission line

5 Quick reference

5.1 Getting started

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

- Mechanically install the device;
- execute electrical connections;
- set the data transmission rate (baud rate and parity bit; see section "4.5.1 Setting data transmission rate: Baud rate and Parity bit (Figure 3)" on page 22); the default value set by Lika Electronic at factory set-up is "baud rate = 19200 bit/s, parity = Even";
- set the node address (node ID; see section "4.5.2 Setting the node address (Figure 3)" on page 24); the default value set by Lika Electronic at factory set-up is "1";
- switch +10VDC ÷ +30 VDC power supply on;
- if you want to use the physical resolution of the unit, please check the **Scaling function** item is disabled (bit 0 in the register **Operating parameters [0x08]** = 0; see on page 46);
- otherwise if you need a specific resolution, please enable the **Scaling function** item (bit 0 in the register **Operating parameters [0x08]** = 1; see on page 46);
- then set the value you need for the single-turn resolution next to the **Counts per revolution [0x00-0x01]** item (registers 1 and 2; see on page 42);
- set the value you need for the overall resolution next to the **Total Resolution [0x02-0x03]** item (registers 3 and 4; see on page 43);
- now, if you need you can set the Preset next to the register **Preset value [0x04-0x05]** and then execute the **Perform counting preset** command in **Control Word [0x09]**; see on page 44);
- save new setting values (**Save parameters** register; see on page 48).

6 Modbus® interface

Lika EMx58 Modbus series encoders are Slave devices and implement the Modbus application protocol (level 7 of OSI model) and the "Modbus over Serial Line" protocol (levels 1 & 2 of OSI model).
For any further information or omitted specifications please refer to "Modbus Application Protocol Specification V1.1b" and "Modbus over Serial Line. Specification and Implementation Guide V1.02" available at www.modbus.org.

6.1 Modbus Master / Slaves protocol principle

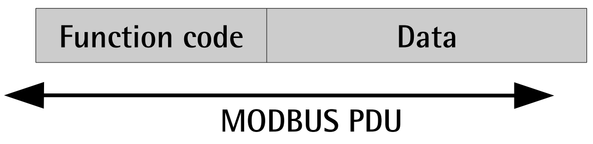
The Modbus Serial Line protocol is a Master – Slaves protocol. One only Master (at the same time) is connected to the bus and one or several (247 maximum number) Slave nodes are also connected to the same serial bus. A Modbus communication is always initiated by the Master. The Slave nodes will never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

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

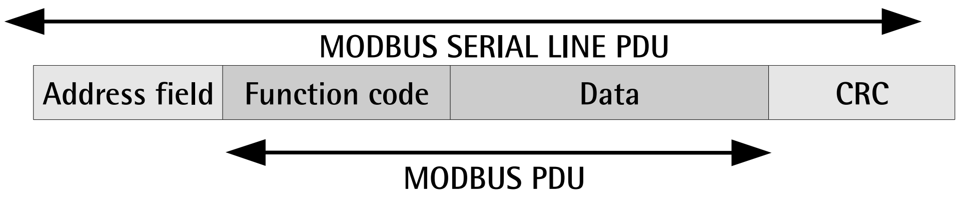
- **UNICAST mode**: in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must have a unique address (from 1 to 247) so that it can be addressed independently from other nodes. Lika devices only implement commands in "unicast" mode.
- **BROADCAST mode**: in that mode the Master can send a request to all Slaves at the same time. No response is returned to "broadcast" requests sent by the Master. The "broadcast" requests are necessarily writing commands. The address 0 is reserved to identify a "broadcast" exchange. Lika devices do not implement commands in "broadcast" mode.

6.2 Modbus frame description

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



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



- **ADDRESS FIELD**: on Modbus Serial Line the address field only contains the Slave address. As previously stated (see section "4.5.2 Setting the node address (Figure 3)" on page 24), the valid Slave node addresses are in the range of 0 – 247 decimal. The individual Slave devices are assigned addresses in the range of 1 – 247. A Master addresses a Slave by placing the Slave address in the **ADDRESS FIELD** of the message. When the Slave returns its response, it places its own address in the response **ADDRESS FIELD** to let the Master know which Slave is responding.
- **FUNCTION CODE**: the function code indicates to the Server what kind of action to perform. The function code can be followed by a **DATA** field that contains request and response parameters. For any further information on the implemented function codes refer to the section "6.4 Function codes" on page 32.
- **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 section "6.4 Function codes" on page 32).
- **CRC (Cyclical Redundancy Checking)**: error checking field is the result of a "Redundancy Checking" calculation that is performed on the message contents. This is intended to check whether transmission has

been performed properly. The CRC field is two bytes, containing 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The device that receives recalculates a CRC during receipt of the message and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The Modbus protocol defines three PDUs. They are:

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

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

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

The **Modbus Exception Response PDU** is defined as {exception-function_code, exception_code}, where:
exception-function_code = Modbus function code + 0x80 [1 byte];
exception_code = Modbus Exception code, refer to the table "Modbus Exception Codes" in the section 7 of the document "Modbus Application Protocol Specification V1.1b".

6.3 Transmission modes

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

6.3.1 RTU transmission mode

When devices communicate on a Modbus serial line using the RTU (Remote Terminal Unit) mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. Each message must be transmitted in a continuous stream of characters. Synchronization between the messages exchanged by the transmitting device and the receiving device is achieved by placing an interval of at least 3,5 character times between successive messages, this is called "silent interval". In this way a Modbus message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows devices that receive a new frame to begin at the start of the message and to know when the message is completed. So when the receiving device does not receive a message for an interval of 4 character times, it considers the previous message as completed and the next byte will be the first of a new message, i.e. an address.
When baud rate = 9600 bit/s the "silent interval" is 4 ms.
When baud rate = 19200 bit/s the "silent interval" is 2 ms.
When baud rate = 115200 bit/s the "silent interval" is 3,5 ms.

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

Coding system:	8-bit binary
Bits per Byte:	1 start bit; 8 data bits, least significant bit (lsb) sent first; 1 bit for parity completion (= Even); 1 stop bit.

Modbus protocol uses a "big-Endian" representation for addresses and data items. This means that when a numerical quantity greater than a single byte is transmitted, the most significant byte (MSB) is sent first.
Each character or byte is sent in this order (left to right):

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

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

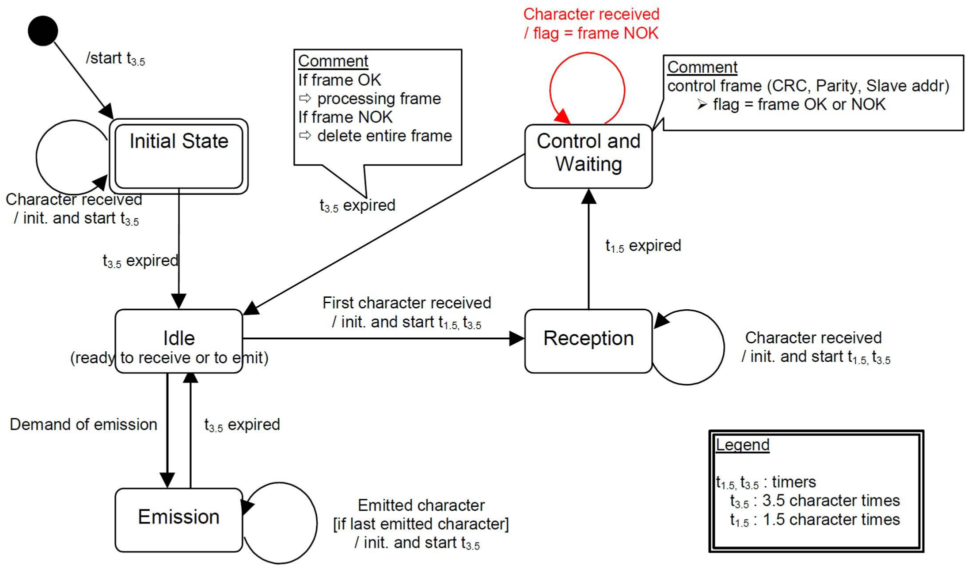
* When "No parity" is activated, the parity bit is replaced by a stop bit.

The default parity mode must be even parity.

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

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

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


/start $t_{3.5}$
Initial State
Character received / init. and start $t_{3.5}$
$t_{3.5}$ expired
Comment
If frame OK
⇨ processing frame
If frame NOK
⇨ delete entire frame
$t_{3.5}$ expired
Character received / flag = frame NOK
Control and Waiting
Comment
control frame (CRC, Parity, Slave addr)
➢ flag = frame OK or NOK
$t_{1.5}$ expired
Idle (ready to receive or to emit)
First character received / init. and start $t_{1.5}$, $t_{3.5}$
Reception
Character received / init. and start $t_{1.5}$, $t_{3.5}$
Demand of emission
$t_{3.5}$ expired
Emission
Emitted character [if last emitted character] / init. and start $t_{3.5}$
Legend
$t_{1.5}$, $t_{3.5}$: timers
$t_{3.5}$: 3.5 character times
$t_{1.5}$: 1.5 character times

- Transition from **Initial State** to **Idle** state needs an interval of at least 3,5 character times (time-out expiration = $t_{3,5}$).
- **Idle** state is the normal state when neither emission nor reception is active. In RTU mode, the communication link is declared in **Idle** state when there is no transmission activity after a time interval equal to at least 3,5 characters ($t_{3,5}$).
- A request can only be sent in **Idle** state. After sending a request, the Master leaves the **Idle** state and cannot send a second request at the same time.
- When the link is in **Idle** state, each transmitted character detected on the link is identified as the start of the frame. The link goes to **Active** state. Then the end of the frame is identified when no more character is transmitted on the link after the time interval of at least $t_{3,5}$.
- After detection of the end of frame, the CRC calculation and checking is completed. Afterwards the address field is analysed to determine if the frame is addressed to the device. If not, the frame is discarded. In order to reduce the reception processing time the address field can be analysed as soon as it is received without waiting the end of frame. In this case the CRC will be calculated and checked only if the frame is actually addressed to the Slave.

6.4 Function codes

As previously stated, the function code indicates to the Server what kind of action to perform. The function code field of a Modbus data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (the range 128 ... 255 is reserved and used for Exception Responses). When a message is sent from a Client to a Server device the function code field tells the Server what king of action to perform. Function code "0" is not valid.
There are three categories of Modbus function codes, they are: **public function codes**, **user-defined function codes** and **reserved function codes**.
Public function codes are in the range 1 ... 64, 73 ... 99 and 111 ... 127; they are well defined function codes, validated by the MODBUS-IDA.org community and publicly documented; furthermore they are guaranteed to be unique. Ranges of function codes from 65 to 72 and from 100 to 110 are **user-defined function codes**: user can select and implement a function code that is not supported by the specification, it is clear that there is no guarantee that the use of the selected function code will be unique. **Reserved function codes** are not available for public use.

6.4.1 Implemented function codes

Lika EMx58 Modbus series encoders only implement public function codes, they are described hereafter.

03 Read Holding Registers

FC = 03 (Hex = 0x03) ro
This function code is used to READ the contents of a contiguous block of holding registers in a remote device; in other words, it allows to read the values set in a group of work parameters placed in order. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore registers numbered 1-16 are addressed as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).
For the complete list of holding registers accessible using 03 Read Holding Registers function code please refer to section "7.1.1 Machine data parameters" on page 42.

Request PDU

Function code	1 byte	**0x03**
Starting address	2 bytes	0x0000 to 0xFFFF
Quantity of registers	2 bytes	1 to 125 (0x7D)

Response PDU

Function code	1 byte	**0x03**
Byte count	1 byte	2 x **N***
Register value	**N*** x 2 bytes	

*N = Quantity of registers

Exception Response PDU

Error code	1 byte	**0x83 (=0x03 + 0x80)**
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to read the parameter Preset value [0x04-0x05] (registers 5 and 6).

Request	
Field name	(Hex)
Function	**03**
Starting address Hi	**00**
Starting address Lo	**04**
No. of registers Hi	**00**
No. of registers Lo	**02**

Response	
Field name	(Hex)
Function	**03**
Byte count	**04**
Register 5 value Hi	**00**
Register 5 value Lo	**00**
Register 6 value Hi	**05**
Register 6 value Lo	**DC**

As you can see in the table, Preset value [0x04-0x05] parameter (registers 5 and 6) contains the value 00 00 hex and 05 DC hex, i.e. 1500 in decimal notation.

The full frame needed for the request to read the parameter **Preset value [0x04-0x05]** (registers 5 and 6) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][03][00][04][00][02][85][CA]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[00][04] = starting address (**Preset value [0x04-0x05]** parameter, register 5)
[00][02] = number of requested registers
[85][CA] = CRC

The full frame needed to send back the values of the parameter **Preset value [0x04-0x05]** (registers 5 and 6) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][03][04][00][00][05][DC][F8][FA]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[00][00] = value of register 5, 00 00 hex = 0 dec
[05][DC] = value of register 6, 05 DC hex = 1500 dec
[F8][FA] = CRC

04 Read Input Register

FC = 04 (Hex = 0x04)

This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some results values and state / alarm messages in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).
For the complete list of input registers accessible using **04 Read Input Register** function code please refer to section "7.1.2 Input Register parameters" on page 50.

Request PDU

Function code	1 byte	**0x04**
Starting address	2 bytes	0x0000 to 0xFFFF
Quantity of Input Registers	2 bytes	0x0000 to 0x007D

Response PDU

Function code	1 byte	**0x04**
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	**0x84 (=0x04 + 0x80)**
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to read the Current position [0x01-0x02] parameter (input registers 2 and 3).

Request	
Field name	(Hex)
Function	**04**
Starting address Hi	**00**
Starting address Lo	**01**
Quantity of Input Reg. Hi	**00**
Quantity of Input Reg. Lo	**02**

Response	
Field name	(Hex)
Function	**04**
Byte count	**04**
Register 3 value Hi	**00**
Register 3 value Lo	**00**
Register 4 value Hi	**2F**
Register 4 value Lo	**F0**

As you can see in the table, Current position [0x01-0x02] parameter (input registers 2 and 3) contains the values 00 00 hex and 2F F0 hex, i.e. 12272 in decimal notation.

The full frame needed for the request to read the Current position [0x01-0x02] parameter (input registers 2 and 3) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][04][00][01][00][02][20][0B]
where:
[01] = Slave address
[04] = 04 Read Input Register function code
[00][01] = starting address (Current position [0x01-0x02] parameter, register 2)
[00][02] = number of requested registers
[20][0B] = CRC

The full frame needed to send back the value of the Current position [0x01-0x02] parameter (registers 2 and 3) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][04][04][00][00][2F][F0][E7][F0]
where:
[01] = Slave address
[04] = 04 Read Input Register function code
[04] = number of bytes (2 bytes for each register)
[00][00] = value of register 2 Current position [0x01-0x02], 00 00 hex = 0 dec
[2F][F0] = value of register 3 Current position [0x01-0x02], 2F F0 hex = 12272 dec
[E7][F0] = CRC

06 Write Single Register

FC = 06 (Hex = 0x06)

This function code is used to WRITE a single holding register in a remote device. The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0.
The normal response is an echo of the request, returned after the register contents have been written.
For the complete list of registers accessible using 06 Write Single Register function code please refer to section "7.1.1 Machine data parameters" on page 42.

Request PDU

Function code	1 byte	**0x06**
Register address	2 bytes	0x0000 to 0xFFFF
Register value	2 bytes	0x0000 to 0xFFFF

Response PDU

Function code	1 byte	**0x06**
Register address	2 bytes	0x0000 to 0xFFFF
Register value	2 bytes	0x0000 to 0xFFFF

Exception Response PDU

Error code	1 byte	**0x86 (=0x06 + 0x80)**
Exception code	1 byte	01 or 02 or 03 or 04

Here is an example of a request to write in the **Operating parameters [0x08]** item (register 9): we need to set the scaling function (**Scaling function** = 1) and the increasing counting with clockwise rotation of the encoder shaft (**Code sequence** = 0).

Request		**Response**	
Field name	(Hex)	Field name	(Hex)
Function	**06**	Function	**06**
Register address Hi	**00**	Register address Hi	**00**
Register address Lo	**08**	Register address Lo	**08**
Register value Hi	**00**	Register value Hi	**00**
Register value Lo	**01**	Register value Lo	**01**

As you can see in the table, the value 00 01 hex, i.e. 0000 0000 0000 0001 in binary notation, is set in the **Operating parameters [0x08]** item (register 9):

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

The full frame needed for the request to write the value 00 01 hex in the Operating parameters [0x08] item (register 9) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][06][00][08][00][01][C9][C8]
where:
[01] = Slave address
[06] = 06 Write Single Register function code
[00][08] = address of the register (Operating parameters [0x08] item, register 9)
[00][01] = value to be set in the register
[C9][C8] = CRC

The full frame needed to send back a response following the request to write in the Operating parameters [0x08] item (register 9) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][06][00][08][00][01][C9][C8]
where:
[01] = Slave address
[06] = 06 Write Single Register function code
[00][08] = address of the register (Operating parameters [0x08] item, register 9)
[00][01] = value set in the register
[C9][C8] = CRC

16 Write Multiple Registers

FC = 16 (Hex = 0x10)

This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
The values to be written are specified in the request data field. Data is packed as two bytes per register.
The normal response returns the function code, starting address and quantity of written registers.

For the complete list of registers accessible using **16 Write Multiple Registers** function code please refer to section "7.1.1 Machine data parameters" on page 42.

Request PDU

Function code	1 byte	**0x10**
Starting address	2 bytes	0x0000 to 0xFFFF
Quantity of registers	2 bytes	0x0001 to 0x007B
Byte count	1 byte	2 x **N***
Registers value	**N*** x 2 bytes	value

*N = Quantity of registers

Response PDU

Function code	1 byte	**0x10**
Starting address	2 bytes	0x0000 to 0xFFFF
Quantity of registers	2 bytes	1 to 123 (0x7B)

Exception Response PDU

Error code	1 byte	**0x90 (= 0x10 + 0x80)**
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 parameter **Counts per revolution [0x00-0x01]** (registers 1 and 2) and the value 00 80 00 00 hex (=8388608 dec) next to the parameter **Total Resolution [0x02-0x03]** (registers 3 and 4).

Request		**Response**	
Field name	(Hex)	Field name	(Hex)
Function	**10**	Function	**10**
Starting address Hi	**00**	Starting address Hi	**00**

Starting address Lo	**00**
Quantity of registers Hi	**00**
Quantity of registers Lo	**04**
Byte count	**08**
Register 1 value Hi	**00**
Register 1 value Lo	**00**
Register 2 value Hi	**08**
Register 2 value Lo	**00**
Register 3 value Hi	**00**
Register 3 value Lo	**80**
Register 4 value Hi	**00**
Register 4 value Lo	**00**

Starting address Lo	**00**
Quantity of registers Hi	**00**
Quantity of registers Lo	**04**

As you can see in the table, the values 00 00 hex and 08 00 hex, i.e. 2048 in decimal notation, are set respectively in the registers 1 and 2 of the Counts per revolution [0x00-0x01] parameter; while the values 00 80 hex and 00 00 hex, i.e. 8388608 in decimal notation, are set respectively in the registers 3 and 4 of the Total Resolution [0x02-0x03] parameter. Thus the encoder will be programmed to have a 2048-count-per-revolution single-turn resolution and a 4096-turn multi-turn resolution (8388608/2048).

The full frame needed for the request to write the value 2048 dec next to the parameter Counts per revolution [0x00-0x01] (registers 1 and 2) and the value 8388608 dec next to the parameter Total Resolution [0x02-0x03] (registers 3 and 4) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][10][00][00][00][04][08][00][00][08][00][00][80][00][00][B6][DA]
where:
[01] = Slave address
[10] = 16 Write Multiple Registers function code
[00][00] = starting address (Counts per revolution [0x00-0x01] parameter, register 1)
[00][04] = number of requested registers
[08] = number of bytes (2 bytes for each register)
[00][00] = value to be set in the register 1

[08][00] = value to be set in the register 2, 00 00 08 00 hex = 2048 dec
[00][80] = value to be set in the register 3
[00][00] = value to be set in the register 4, 00 80 00 00 hex = 8388608 dec
[B6][DA] = CRC

The full frame needed to send back a response following the request to write the value 2048 next to the parameter **Counts per revolution [0x00-0x01]** (registers 1 and 2) and the value 8388608 next to the parameter **Total Resolution [0x02-0x03]** (registers 3 and 4) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][10][00][00][00][04][C1][CA]
where:
[01] = Slave address
[10] = **16 Write Multiple Registers** function code
[00][00] = starting address (**Counts per revolution [0x00-0x01]** parameter, register 1)
[00][04] = number of written registers
[C1][CA] = CRC

WARNING
For safety reasons, when the encoder is on, a continuous data exchange between the Master and the Slave has to be planned in order to be sure that the communication is always active; this is intended to prevent danger situations from arising in case of failures in the communication network.
For this purpose the Watch dog function is implemented and can be activated as optional. Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running Watch dog safety system immediately takes action and commands an alarm to be triggered. To enable the Watch dog function, set to "=1" the **Watch dog enable** bit 0 in the **Control Word [0x09]** variable. If "=0" is set the Watch dog is disabled; if "=1" is set the Watch dog is enabled. When the Watch dog function is enabled, if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm message is invoked to appear as soon as the Modbus network communication is restored).

7 Programming parameters

7.1 Parameters available

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

Parameter name [Register address]
[Register number, data types, attribute]

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

7.1.1 Machine data parameters

Machine data parameters are accessible for both writing and reading; to read the value set in a parameter use the **03 Read Holding Registers** function code (reading of multiple registers); to write a value in a parameter use the **06 Write Single Register** function code (writing of a single register) or the **16 Write Multiple Registers** (writing of multiple registers); for any further information on the implemented function codes refer to the section "6.4.1 Implemented function codes" on page 32.

Counts per revolution [0x00-0x01]

[Registers 1-2, Unsigned32, rw]
This parameter sets the number of distinguishable steps per revolution.
This value is considered only if the bit 0 in the **Operating parameters [0x08]** item is set to "=1".
You are allowed to set whatever integer value lower than or equal to the **maximum number of physical steps per revolution** (see the hardware counts per revolution in the encoder identification label). If you set a value greater than the maximum number of physical steps per revolution, after sending the Request PDU the **Machine data not valid** error message will be sent back while the relevant bit in the **Wrong parameters list [0x04-0x05]** item will be set to 1.
For any further information on the maximum number of physical steps per revolution and the maximum number of physical revolutions refer to the identification label of the specific device.
Default = 4096 (min. = 1, max. = 4096)

NOTE
To avoid counting errors please always make sure to meet the following condition:

$$\frac{\text{Total Resolution [0x02-0x03]}}{\text{Counts per revolution [0x00-0x01]}} = \text{power of 2.}$$

Furthermore, after having set a new value next to the Counts per revolution [0x00-0x01] parameter, make sure to meet also the following condition:

$$\frac{\text{Total Resolution [0x02-0x03]}}{\text{Counts per revolution [0x00-0x01]}} \leq \text{Number of physical revolutions}$$

If you have set the preset, when you change the value next to the Counts per revolution [0x00-0x01] parameter, then you must check the value in the Preset value [0x04-0x05] parameter and perform the homing operation (bit 11 Perform counting preset in Control Word [0x09] = 1).

Total Resolution [0x02-0x03]

[Registers 3-4, Unsigned32, rw]
This is intended to set the number of distinguishable steps over the whole measuring range (overall resolution of the encoder).
This value is considered only if the bit 0 in the Operating parameters [0x08] item is set to "=1".
You are allowed to set whatever integer value lower than or equal to the **overall hardware resolution** (see the encoder identification label). If you set a value greater than the overall hardware resolution, after sending the Request PDU the Machine data not valid error message will be sent back while the relevant bit in the Wrong parameters list [0x04-0x05] item will be set to 1.
For any further information on the maximum number of physical steps per revolution and the maximum number of physical revolutions refer to the identification label of the specific device.
Default = 67108864 (min. = 1, max. = 67108864)

NOTE
To avoid counting errors please always make sure to meet the following condition:

$$\frac{\text{Total Resolution [0x02-0x03]}}{\text{Counts per revolution [0x00-0x01]}} = \textbf{power of 2.}$$

If you have set the preset, when you change the value next to the Total Resolution [0x02-0x03] parameter, then you must check the value in the Preset value [0x04-0x05] parameter and perform the homing operation (bit 11 Perform counting preset in Control Word [0x09] = 1).

Example
Multiturn encoder EM58 **12/16384**.

The physical resolution is as follows:

• **Hardware counts per revolution**	= 4096 PPR (2^{12})
• **Hardware number of revolutions**	= 16384 turns (2^{14})
• **Overall hardware resolution**	= 67108864 P (2^{26})

You need to set a single-turn resolution of **2048 counts per revolution** while **1024 revolutions** are required:

- Enable the Scaling function: Operating parameters [0x08], bit 0 = 1
- Set the number of distinguishable steps per revolution: Counts per revolution [0x00-0x01] = 2048 (0000 0800 hex)
- Set the overall resolution: Total Resolution [0x02-0x03] = 2048 * 1024 = 2097152 (0020 0000 hex)
- Save the set parameters (Save parameters register; see on page 48)

Preset value [0x04-0x05]

[Registers 5-6, Unsigned32, rw]
This register is intended to set the Preset value. Preset function is meant to assign a certain value to a desired physical position of the encoder. The chosen physical position will get the value set next to this item and all the previous and following positions will get a value according to it. For instance, this can be useful for getting the zero point of the encoder and the zero point of the application to match. The preset value will be set for the position of the encoder in the moment when the Perform counting preset command in Control Word [0x09] is sent.
Default = 0 (min. = 0, max. = 67108864)

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 [0x04-0x05], Offset value [0x06-0x07] 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 [0x04-0x05] - Offset value [0x06-0x07].
If you never set the Preset value [0x04-0x05] and the homing command has not been executed before anyway (Perform counting preset command), the transmitted value and the read position are necessarily the same as Preset value [0x04-0x05] = 0 and Offset value [0x06-0x07] = 0.
When you set the Preset value [0x04-0x05] and then execute the Perform counting preset command in the Control Word [0x09], system saves the current encoder position in the register Offset value [0x06-0x07]. It follows that the transmitted value and the Preset value [0x04-0x05] are the same as read position - Offset value [0x06-0x07] = 0; in other words, the value set next to the Preset value [0x04-0x05] item is paired with the current position of the encoder as you wish.
For example, let's assume that the value "50" is set next to the Preset value [0x04-0x05] item and you execute the Perform counting preset command when the encoder position is "1000". In other words, you want to receive the value "50" when the encoder reaches the position "1000".
We will obtain the following:
Transmitted value = **read position** (="1000") + Preset value [0x04-0x05] (="50") - Offset value [0x06-0x07] (="1000") = **50**.
The following transmitted value will be:
Transmitted value = **read position** (="1001") + Preset value [0x04-0x05] (="50") - Offset value [0x06-0x07] (="1000") = **51**.
And so on.

NOTE

- If the Scaling function is disabled (bit 0 in the register Operating parameters [0x08] = 0), Preset value [0x04-0x05] must be lower than or equal to the total hardware resolution (i.e. hardware counts per revolution * number of hardware revolutions).
- If the Scaling function is enabled (bit 0 in the register Operating parameters [0x08] = 1), Preset value [0x04-0x05] must be lower than or equal to Total Resolution [0x02-0x03].

WARNING
After having entered a new value in the registers Counts per revolution [0x00-0x01] and / or Total Resolution [0x02-0x03] it is compulsory to check the Preset value [0x04-0x05] and then perform a homing operation (bit 11 Perform counting preset in Control Word [0x09] = 1).

Offset value [0x06-0x07]

[Registers 7-8, Unsigned32, ro]
As soon as you send the Perform counting preset command (see bit 11 in Control Word [0x09]), the current position of the encoder is saved in this register. The offset value is then used in the preset function in order to calculate the encoder position value to be transmitted. To zero set the value in this register you must upload the factory default values (see bit 10, Load default parameters command, in Control Word [0x09] on page 48).
For any further information on the preset function and the meaning and use of the related registers and commands Preset value [0x04-0x05], Offset value [0x06-0x07] and Perform counting preset refer to page 44.
Default = 0 (min. = 0, max. = 67108864)

Operating parameters [0x08]

[Register 9, Unsigned16, rw]
Byte structure of the Operating parameters [0x08] register:

byte	MSB			LSB		
bit	15	…	8	7	…	0
	msb		lsb	msb		lsb

Byte 0

Scaling function

bit 0 — This is meant to enable / disable the scaling parameters Counts per revolution [0x00-0x01] and Total Resolution [0x02-0x03]. When the scaling function is disabled (bit 0 = 0), the encoder uses its own physical resolution (i.e. hardware counts per revolution and number of hardware revolutions, see the encoder identification label); otherwise, when the scaling function is enabled (bit 0 = 1), the encoder uses the resolution set next to the registers Counts per revolution [0x00-0x01] and Total Resolution [0x02-0x03] in accordance with the following relation:

$$\text{Transmitted position value} = \frac{\text{Counts per revolution [0x00-0x01]}}{\text{Hardware counts per revolution}} \times \text{Real position} \leq \text{Total Resolution [0x02-0x03]}$$

The currently set value can be read in the Scaling bit 0 of the Status word [0x0A], see on page 54.

Code sequence

bit 1	This is intended to set if the encoder counting increments either when the shaft is rotating clockwise (CW) or when the shaft is rotating counter-clockwise (CCW). CW and CCW rotations are viewed from shaft end. Setting 0 (bit 1 = 0) causes the encoder counting to increment when the shaft is rotating clockwise; setting 1 (bit 1 = 1) causes the encoder counting to increment when the shaft is rotating counter-clockwise. The currently set value can be read in the Counting direction bit 1 of the Status word [0x0A], see on page 54.
bit 2 … 7	Not used.
Byte 1	Not used.

Control Word [0x09]

[Register 10, Unsigned16, rw]
This variable contains the commands to be sent in real time to the Slave in order to manage it.
Byte structure of the Control Word [0x09] register:

byte	MSB			LSB		
bit	15	…	8	7	…	0
	msb		lsb	msb		lsb

Byte 0 Not used.

Byte 1

Watch dog enable

bit 8 Setting the **Watch dog enable** bit to "=1" causes the Watch dog function to be enabled; setting the **Watch dog enable** bit to "=0" causes the Watch dog function to be disabled. When the Watch dog function is enabled, if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm is invoked to appear as soon as the Modbus network communication is restored). Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running Watch dog safety system immediately takes action and commands an alarm to be triggered.

Save parameters

bit 9 Data is saved on non-volatile memory at each rising edge of 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").

Load default parameters

bit 10 Default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) are restored at each rising edge of 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"). The complete list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 60.

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

Perform counting preset

bit 11 It allows to perform a homing operation of the encoder. As soon as the command is sent, the position value which will be transmitted for the current position of the encoder is the one set next to the register **Preset value [0x04-0x05]** and all the previous and following positions will get a value according to it. 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"). When this command is sent, the current encoder position is temporarily saved in the register **Offset value [0x06-0x07]**. For any further information on the preset function and the meaning and use of the related registers and commands **Preset value [0x04-0x05]**, **Offset value [0x06-0x07]** and **Perform counting preset** refer to page 44.

WARNING
To save the current encoder position in the register **Offset value [0x06-0x07]** permanently, please execute the **Save parameters** command. Should the power be turned off without saving data, the **Offset value [0x06-0x07]** will be lost!

bit 12 ... 15 Not used.

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

7.1.2 Input Register parameters

Input Register parameters are accessible for reading only; to read the value set in an input register parameter use the **04 Read Input Register** function code (reading of multiple input registers); for any further information on the implemented function codes refer to the section "6.4.1 Implemented function codes" on page 32.

Alarms register [0x00]

[Register 1, Unsigned16, ro]
This variable is meant to show the alarms currently active in the device.
Structure of the alarms byte:

byte	**MSB**			**LSB**		
bit	15	...	8	7	...	0
	msb		lsb	msb		lsb

The available alarm error codes are listed hereafter:

Byte 0

Machine data not valid

bit 0	One or more parameters are not valid, set proper values to restore normal work condition. See the list of the wrong parameters in the register **Wrong parameters list [0x04-0x05]**.

Flash memory error

bit 1	Flash memory internal error, it cannot be restored (bad checksum error, etc.).
bit 2 ... 7	Not used.

Byte 1

bit 8 ... 10	Not used.

Watch dog

bit 11	When the Watch dog function is enabled (**Watch dog enable** in **Control Word [0x09]** is set to "=1"), if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm bit is activated). The alarm is invoked to appear as soon as

	the Modbus network communication is restored. Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running Watch dog safety system immediately takes action and commands an alarm to be triggered.
bits 12 ... 15	Not used.

NOTE
Please note that should the alarm be caused by wrong parameter values (see **Machine data not valid** and **Wrong parameters list [0x04-0x05]** register), normal work status can be restored only after having set proper values. The **Watch dog** alarm is deleted automatically as soon as the communication is restored. The **Flash memory error** alarm cannot be reset.

Current position [0x01-0x02]

[Registers 2-3, Integer32, 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 46. Value is expressed in pulses.

Register 4 [0x03]

[Register 4, Integer16, ro]
This register is currently not used and reserved for future use.

Wrong parameters list [0x04-0x05]

[Registers 5-6, Unsigned32, 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	Parameter
0	Not used
1	Counts per revolution [0x00-0x01]
2	Total Resolution [0x02-0x03]
3	Preset value [0x04-0x05]
4	Offset value [0x06-0x07]
5	Operating parameters [0x08]
6	Dip-switch node ID [0x07]
7	Dip-switch baud rate [0x06]
8 ... 15	Not used

Dip-switch baud rate [0x06]

[Register 7, Unsigned16, ro]
This is meant to show the data transmission rate (baud rate and parity bit) of the serial port fitted in the unit; data transmission rate has to be set through the provided dip-switch. For any further information on setting the baud rate and the parity bit refer to the section "4.5.1 Setting data transmission rate: Baud rate and Parity bit (Figure 3)" on page 22.

Value	Baud rate	Parity bit
0x00	9600 bit/s	No parity
0x01	9600 bit/s	Even
0x02	9600 bit/s	Odd
0x03	19200 bit/s	No parity
0x04	19200 bit/s	Even
0x05	19200 bit/s	Odd
0x06	115200 bit/s	No parity
0x07	115200 bit/s	Even
0x08	115200 bit/s	Odd

Dip-switch node ID [0x07]

[Register 8, Unsigned16, ro]
This is meant to show the node address set in the unit; node address has to be set through the provided dip-switch. For any further information on setting the node ID refer to the section "4.5.2 Setting the node address (Figure 3)" on page 24.

SW Version [0x08]

[Register 9, Unsigned16, ro]
This is meant to show the software version of the encoder.
The meaning of the 16 bits in the register is as follows:

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

Value 01 02 hex in hexadecimal notation corresponds to the binary representation 00000001 00000010 and has to be interpreted as: version 1.2.

HW Version [0x09]

[Register 10, Unsigned16, ro]
This is meant to show the hardware version of the encoder.
The meaning of the 16 bits in the register is as follows:

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

Value 01 00 hex in hexadecimal notation corresponds to the binary representation 00000001 00000000 and has to be interpreted as: version 1.0.

Status word [0x0A]

[Register 11, Unsigned16, ro]

This register contains information about the current state of the device. The eight bits of Byte 0 (LSB) shows the currently set values of the **Operating parameters [0x08]** register Byte 0 (LSB); while bit 8 of MSB is used to signal active alarms.

Byte structure of the **Status word [0x0A]** register:

byte	MSB			LSB		
bit	15	…	8	7	…	0
	msb		lsb	msb		lsb

Byte 0

Scaling

bit 0	It shows the currently set value that is entered in the parameter **Scaling function**. In other words, it is intended to show whether the scaling function is enabled or disabled. If the value is "=0" the scaling function is disabled; if the value is "=1" instead the scaling function is enabled. For any further information on setting and using the scaling function refer to the **Scaling function** parameter on page 46.

Counting direction

bit 1	It shows the currently set value that is entered in the parameter **Code sequence**. If the bit is "=0" the output encoder position value has been set to increment when the shaft rotates clockwise; if the bit is "=1" instead the output encoder position value has been set to increment when the shaft rotates counter-clockwise. For any further information on setting and using the counting direction function refer to the **Code sequence** parameter on page 47.
bit 2 … 7	Not used.

Byte 1

Alarm

bit 8	If value is "=1" there is an active alarm, see details in the **Alarms register [0x00]** variable on page 50.
bit 9 … 15	Not used.

7.2 Exception codes

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

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

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

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

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

8 Programming examples

Hereafter are some examples of both reading and writing parameters. All values are expressed in hexadecimal notation.

8.1 Using the 03 Read Holding Registers function code

Example 1
Request to read the parameter Preset value [0x04-0x05] (registers 5 and 6) to the Slave having the node address 1.

Request PDU (in hexadecimal notation)
[01][03][00][04][00][02][85][CA]
where:
[01] = Slave address
[03] = 03 Read Holding Registers function code
[00][04] = starting address (Preset value [0x04-0x05] parameter, register 5)
[00][02] = number of requested registers
[85][CA] = CRC

Response PDU (in hexadecimal notation)
[01][03][04][00][00][05][DC][F8][FA]
where:
[01] = Slave address
[03] = 03 Read Holding Registers function code
[04] = number of bytes (2 bytes for each register)
[00][00] = value of register 5, 00 00 hex = 0 dec
[05][DC] = value of register 6, 05 DC hex = 1500 dec
[F8][FA] = CRC

Preset value [0x04-0x05] parameter (registers 5 and 6) contains 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 [0x04-0x05] parameter is 1500 dec.

8.2 Using the 04 Read Input Register function code

Example 1
Request to read the Current position [0x01-0x02] parameter (registers 2 and 3) to the Slave having the node address 1.

Request PDU (in hexadecimal notation)
[01][04][00][01][00][02][20][0B]
where:
[01] = Slave address
[04] = 04 Read Input Register function code
[00][01] = starting address (Current position [0x01-0x02] parameter, register 2)
[00][02] = number of requested registers
[20][0B] = CRC

Response PDU (in hexadecimal notation)
[01][04][04][00][00][2F][F0][E7][F0]
where:
[01] = Slave address
[04] = 04 Read Input Register function code
[04] = number of bytes (2 bytes for each register)
[00][00] = value of register 2 Current position [0x01-0x02], 00 00 hex = 0 dec
[2F][F0] = value of register 3 Current position [0x01-0x02], 2F F0 hex = 12272 dec
[E7][F0] = CRC

Current position [0x01-0x02] parameter (registers 3 and 4) contains the value 00 00 2F F0 hex, i.e. 12272 in decimal notation.

8.3 Using the 06 Write Single Register function code

Example 1
Request to write in the Operating parameters [0x08] register (register 9) to the Slave having the node address 1: we need to set the scaling function (Scaling function = 1) and the increasing counting with clockwise rotation of the encoder shaft (Code sequence = 0). The value to set is 00 01 hex (= 0000 0000 0000 0001 in binary notation: bit 0 Scaling function = 1; bit 1 Code sequence = 0; the remaining bits are not used, therefore their value is 0).

Request PDU (in hexadecimal notation)
[01][06][00][08][00][01][C9][C8]
where:
[01] = Slave address
[06] = 06 Write Single Register function code
[00][08] = address of the register (Operating parameters [0x08] item, register 9)
[00][01] = value to be set in the register
[C9][C8] = CRC

Response PDU (in hexadecimal notation)
[01][06][00][08][00][01][C9][C8]
where:
[01] = Slave address
[06] = 06 Write Single Register function code
[00][08] = address of the register (Operating parameters [0x08] item, register 9)
[00][01] = value set in the register
[C9][C8] = CRC

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

8.4 Using the 16 Write Multiple Registers function code

Example 1
Request to write the value 00 00 08 00 hex (=2048 dec) next to the parameter Counts per revolution [0x00-0x01] (registers 1 and 2) and the value 00 80 00 00 hex (= 8388608 dec) next to the parameter Total Resolution [0x02-0x03] (registers 3 and 4) of the Slave having the node address 1.

Request PDU (in hexadecimal notation)
[01][10][00][00][00][04][08][00][00][08][00][00][80][00][00][B6][DA]
where:
[01] = Slave address
[10] = 16 Write Multiple Registers function code
[00][00] = starting address (Counts per revolution [0x00-0x01] parameter, register 1)
[00][04] = number of requested registers
[08] = number of bytes (2 bytes for each register)
[00][00] = value to be set in the register 1
[08][00] = value to be set in the register 2, 00 00 08 00 hex = 2048 dec
[00][80] = value to be set in the register 3
[00][00] = value to be set in the register 4, 00 80 00 00 hex = 8388608 dec
[B6][DA] = CRC

Response PDU (in hexadecimal notation)
[01][10][00][00][00][04][C1][CA]
where:
[01] = Slave address
[10] = 16 Write Multiple Registers function code
[00][00] = starting address (Counts per revolution [0x00-0x01] parameter, register 1)
[00][04] = number of written registers
[C1][CA] = CRC

The values 00 00 hex and 08 00 hex, i.e. 2048 in decimal notation, are set respectively in the registers 1 and 2 of the Counts per revolution [0x00-0x01] parameter; while the values 00 80 hex and 00 00 hex, i.e. 8388608 in decimal notation, are set respectively in the registers 3 and 4 of the Total Resolution [0x02-0x03] parameter. Thus the encoder will be programmed to have a 2048-count-per-revolution single-turn resolution and a 4096-turn multi-turn resolution (8388608/2048).

9 Default parameters list

9.1 List of the Holding Registers with default value

Registers list and address	Default value		
Counts per revolution [0x00-0x01] PPR	4096		
Total Resolution [0x02-0x03] P	67108864		
Preset value [0x04-0x05] ms	0		
Offset value [0x06-0x07] P	0		
Scaling function in Operating parameters [0x08]	0		
Code sequence in Operating parameters [0x08]	0		
Watch dog enable in Control Word [0x09]	0		
Save parameters in Control Word [0x09]	-		
Load default parameters in Control Word [0x09]	-		
Perform counting preset in Control Word [0x09]	-		

9.2 List of the Input Registers

Registers list and address	Description of the bits
Alarms register [0x00]	0 Machine data not valid 1 Flash memory error 11 Watch dog
Current position [0x01-0x02]	-
Register 4 [0x03]	-
Wrong parameters list [0x04-0x05]	1 Counts per revolution [0x00-0x01] 2 Total Resolution [0x02-0x03] 3 Preset value [0x04-0x05] 4 Offset value [0x06-0x07] 5 Operating parameters [0x08] 6 Dip-switch node ID [0x07] 7 Dip-switch baud rate [0x06]
Dip-switch baud rate [0x06]	-
Dip-switch node ID [0x07]	-
SW Version [0x08]	-
HW Version [0x09]	-
Status word [0x0A]	0 Scaling 1 Counting direction 8 Alarm

This page intentionally left blank

HW-SW release	Document release	Description
1-1	1.0	First issue
1-1	1.1	Preliminary information and Quick reference sections updated

LIKA Electronic
Via S. Lorenzo, 25
36010 Carré (VI) • Italy
Tel. +39 0445 806600
Fax +39 0445 806699

Italy: eMail info@lika.it - www.lika.it
World: eMail info@lika.biz - www.lika.biz