Smart encoders & actuators

User's guide

RD6

RS-485 version

- Rotary actuator with MODBUS RTU interface
- RS-485 communication interface in compliance with “Modbus over Serial Line” protocol
- BLDC motor sizes: P_N 157 W, M_N 0.5 Nm, n_N 3000 rpm
  P_N 250 W, M_N 0.8 Nm, n_N 3000 rpm
- Real multiturn absolute encoder 4096 cpr x 65536 rev.
- For change-over operations and automated positioning systems

Suitable for the following models:

- RD6-P8-157-MB-...
- RD6-P8-250-MB-...

General Contents

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 Lika device and interface are coloured in GREEN;
- alarms are coloured in RED;
- states are coloured in FUCSIA.

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

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

Preliminary information

This guide is designed to provide the most complete information the operator needs to correctly and safely install and operate the **DRIVECOD rotary actuators RD6 model with MODBUS RTU interface**.

RD6 units are positioning devices which integrate into one system a brushless motor, a drive, a multiturn absolute encoder and a position controller. They are not equipped with gears. The 70 mm (2.76") size square flange and the 14 mm (0.55") shaft are designed to be coupled with planetary gearboxes available in the market. Thus the units can be easily integrated into custom applications to meet specific torque and speed requirements. RD6 are designed to drive positioning systems, change-over applications and linear guides. Typical uses are packaging lines, food processing and pharmaceutical industries, wood & metalworking machinery, paper machinery, material handling equipment, bending machines, filling and bottling plants, printing machines, mold changers, mobile stops, tool changers, spindle positioning devices, among others.

RD6 rotary actuators can be equipped with the following interfaces:

- RD6-x-xxx-**CB**-... = CANopen DS301 interface;
- RD6-x-xxx-**EC**-... = EtherCAT interface;
- RD6-x-xxx-**EP**-... = EtherNet/IP interface;
- RD6-x-xxx-**MB**-... = Modbus RTU (RS-485) interface;
- RD6-x-xxx-**PB**-... = Profibus-DP interface;
- RD6-x-xxx-**PL**-... = POWERLINK interface.

The present manual is specifically designed to describe the MODBUS interface model. For information on the actuators designed for the integration into other fieldbus/Ethernet networks, please refer to the specific documentation.

In the MODBUS RTU version the configuration of the DRIVECOD unit can be done through a software expressly developed and released by Lika Electronic in order to allow an easy set up of the device. The program is supplied for free and can be installed in any PC fitted with a Windows operating system (Windows XP or later). It allows the operator to set the working parameters of the device; control manually some movements and functions; and monitor whether the device is running properly. MODBUS uses RS-485 serial communcation.

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.

Glossary of MODBUS terms

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

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

	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 actuators only implement public function codes.
Holding register	In the MODBUS data model, a Holding register is the output data. A Holding register has a 16-bit quantity, is alterable by an application program, and allows either read-write or read-only access.
IEEE 1588	This standard defines a protocol enabling synchronisation of clocks in distributed networked devices (e.g. connected via Ethernet).
Input register	In the MODBUS data model, an Input register is the input data. An Input register has a 16-bit quantity, is provided by an I/O system, and allows read-only access.
LRC Checking	In ASCII mode, messages include an error-checking field that is based on a Longitudinal Redundancy Checking (LRC) calculation that is performed on the message contents, exclusive of the beginning 'colon' and terminating CRLF pair characters. It is applied regardless of any parity checking method used for the individual characters of the message.
Master	A Master is any network device that sends data requests to Slaves.
Message	The MODBUS messaging service provides a Client/Server communication between devices connected on the network. The Client / Server model is based on four types of messages: • 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.
MODBUS Response	A MODBUS Response is the Response message sent by the Server.
Network	Network is a group of computers on a single physical network segment.

PDU	The Protocol Data Unit (PDU) is the MODBUS function code and data field. It is packed together with the Address Field and the CRC (or LRC) to form the Modbus Serial Line PDU. 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
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.
Read Input Register (04, 0004hex)	This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some result values and state / alarm messages in a remote device.
Register	MODBUS functions operate on memory registers to configure, monitor, and control device I/O.
RTU transmission mode	Remote Terminal Unit. When devices communicate on a MODBUS serial line using the RTU mode, each 8–bit byte in a message contains two 4–bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII mode for the same baud rate. Each message must be transmitted in a continuous stream of characters.
Server	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 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.
Transmission rate	Data transfer rate (in bps).
Write Multiple Registers (16, 0010hex)	This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
Write Single Register (06, 0006hex)	This function code is used to WRITE a single holding register in a remote device.

1 Safety summary

1.1 Safety

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

1.2 Electrical safety

- Turn OFF power supply before connecting the device;
- connect according to explanation in the "Electrical connections" section on page 18;
- a safety push-button for emergency power off must be installed to shut off the motor power supply in case of emergency situations;
- in compliance with 2014/30/EU norm on electromagnetic compatibility, following precautions must be taken:
  - before handling and installing the equipment, discharge electrical charge from your body and tools which may come in touch with the device;
  - power supply must be stabilized without noise; install EMC filters on device power supply if needed;
  - always use shielded cables (twisted pair cables whenever possible);
  - avoid cables runs longer than necessary;
  - avoid running the signal cable near high voltage power cables;
  - mount the device as far as possible from any capacitive or inductive noise source; shield the device from noise source if needed;
  - to guarantee a correct working of the device, avoid using strong magnets on or near by the unit;

- minimize noise by connecting the shield and/or the connector housing and/or the 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 "Mechanical installation" section on page 15;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the unit;
- do not tool the unit 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;
- the actuator can be mounted directly on the drive shaft or coupled with planetary gearboxes. An adapting flange can be further interposed.

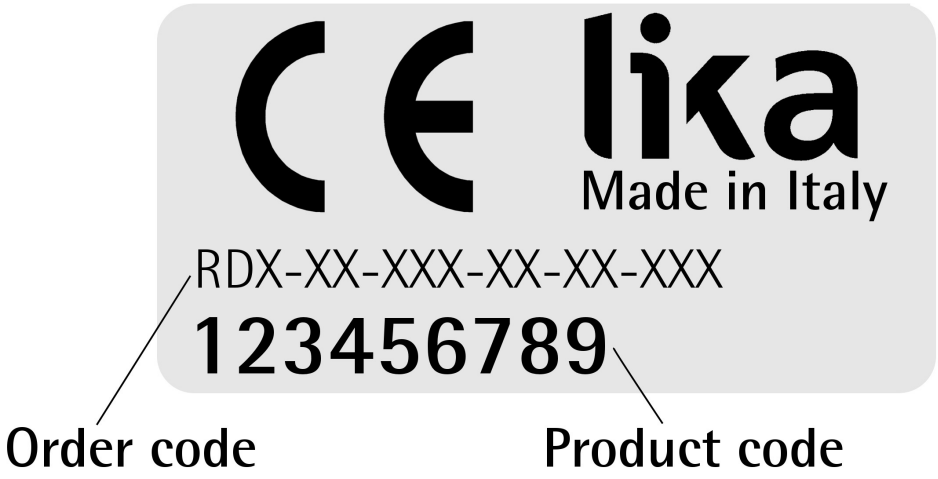
WARNING
The unit has been adjusted by performing a full-load mechanical running test; thence default values which has been set refer to a device running in such condition. Furthermore they are intended to ensure a standard and safe operation which not necessarily results in a smooth running and an optimum performance. Thus to suit the specific application requirements it may be advisable and even necessary to enter new parameters instead of the factory default settings; in particular it may be necessary to change velocity, acceleration, deceleration and gain values.

WARNING
The counter-electromotive force (back EMF) generated by the motor in case the shaft is forced to spin due to a manual external force can cause irreparable damages to the internal circuitry.

WARNING
Please evaluate attentively the characteristics of the 24V power supply pack as the counter-electromotive force (back EMF) generated by the motor flows back and directly discharge through the capacitor module of the 24V power supply pack.

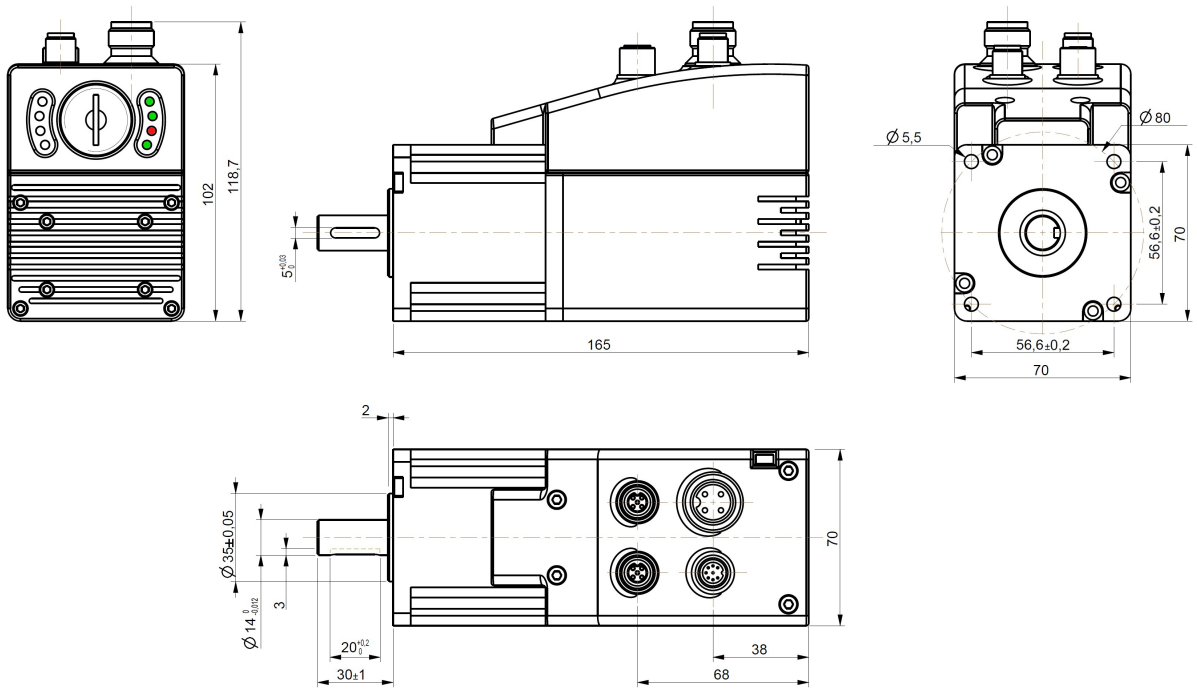
2 Identification

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



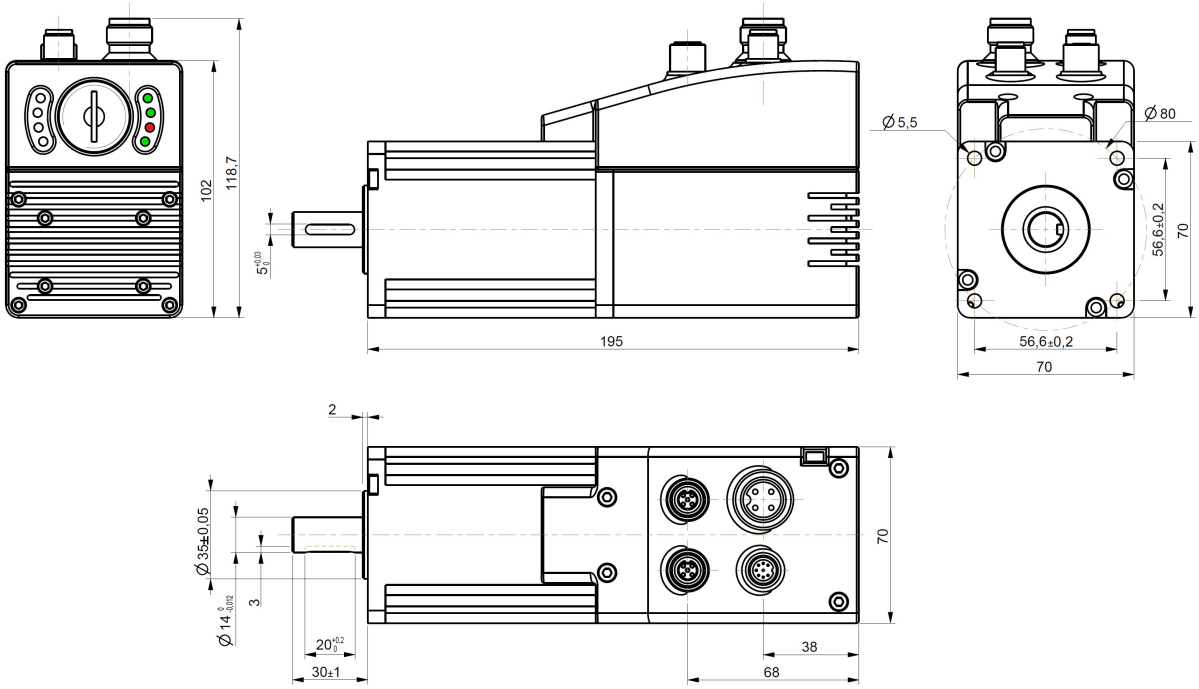
3 Mechanical installation

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



(values are expressed in mm)

Figure 1 - RD6-P8-157-... unit – Overall dimensions



(values are expressed in mm)

Figure 2 - RD6-P8-250-... unit – Overall dimensions

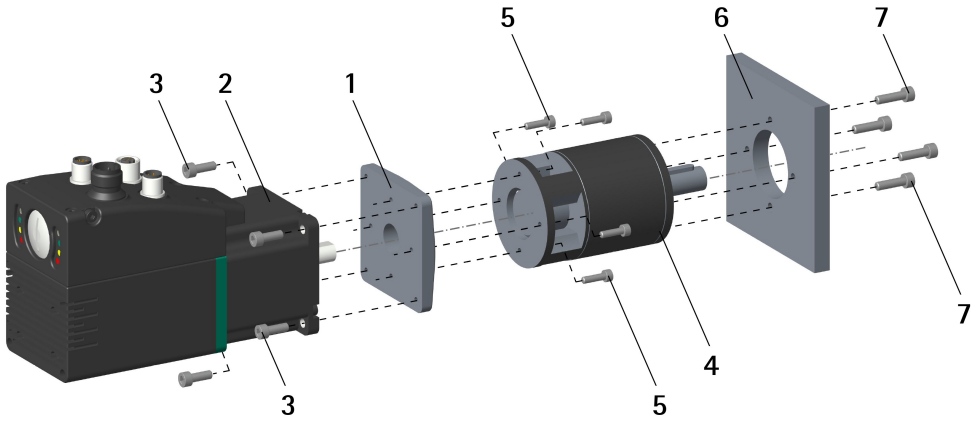
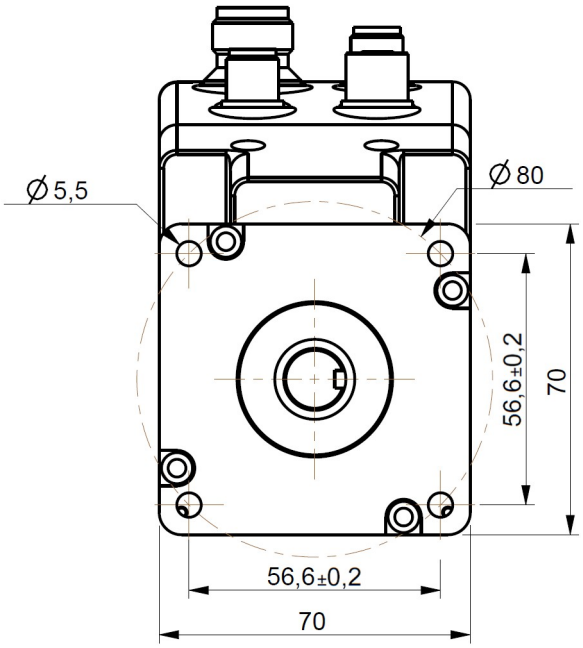


Figure 3 - Installation example of an RD6 unit

To properly install the DRIVECOD unit please read carefully and follow the instructions below; note anyway that the unit can be installed in several manners and according to the specific user's application.

- If required, mount an adapting flange **1**; the adapting flange **1** can be either fixed to the actuator's flange **2** first and then to the planetary gearbox **4**; or on the contrary it can be fixed to the planetary gearbox **4** first and then to the actuator's flange **2**, according to the mounting configuration;



  - use M5 type screws **3** to fix the adapting flange **1** to the actuator's flange **2**;
  - use the screws **5** to fix the planetary gearbox **4** to the adapting flange **1**;
  - couple and fasten the actuator and the planetary gearbox **4** together using the screws **3** or **5** according to the mounting configuration; properly secure the shaft of the actuator and the shaft of the planetary gearbox **4**;
- it could be advisable to install a coupling between the actuator and the planetary gearbox **4**;
- mount the shaft of the planetary gearbox **4** on the drive's shaft and properly secure them together; then fasten the planetary gearbox **4** to the flange or the mounting support **6** by means of the screws **7**.

WARNING
Never force manually the rotation of the shaft not to cause permanent damages! The counter-electromotive force (back EMF) generated by the motor in case the shaft is forced to spin due to a manual external force may cause irreparable damages to the internal circuitry.

4 Electrical connections

WARNING
Power supply must be turned off before performing any electrical connection!
When you send the **Start**, **Jog +** or **Jog -** commands, the unit and the shaft start moving! Before operating please make sure that there are no risks of personal injury and mechanical damages.
Each **Start** routine has to be checked carefully in advance!
Never force manually the rotation of the shaft not to cause permanent damages!

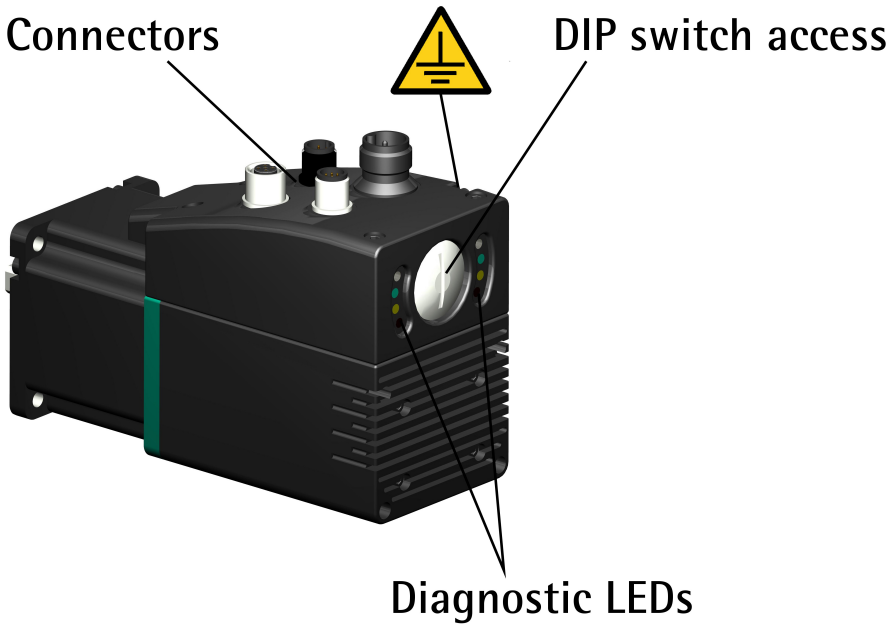
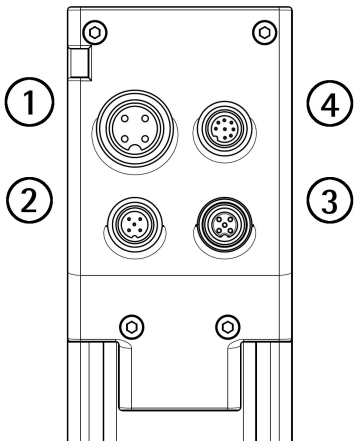


Figure 4: Connectors, diagnostic LEDs and DIP switches

4.1 Ground connection (Figure 5)

To minimize noise connect properly the frame to ground; we suggest using the ground screw provided in the frame (see the Figure above). 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. See also the note in the next paragraph. Anyway make sure that ground is not affected by noise. It is recommended to provide the ground connection as close as possible to the device.

4.2 Connectors (Figure 4 and Figure 5)



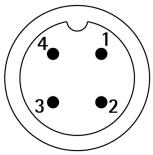
1	M18 4-pin male connector power supply
2	M12 5-pin male connector MODBUS BUS IN
3	M12 5-pin female connector MODBUS BUS OUT
4	M12 8-pin male connector, not used in this model

Figure 5: Connectors

4.2.1 Power supply connector

Power supply
M18 4-pin male connector

(frontal side)

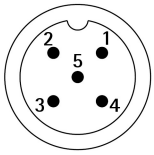


Pin	Description
1	**motor +24Vdc ±10% power supply**
2	**controller +24Vdc ±10% power supply**
3	**motor 0Vdc supply voltage**
4	**controller 0Vdc supply voltage**

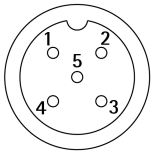
NOTE
Wire gauge of the mating connector cable: 1.50 mm^2 max. Please consider the consumption of both the motor and the controller to evaluate the better configuration, see the datasheet.

4.2.2 Modbus interface connectors (BUS IN and BUS OUT)

Interface
M12 5-pin connectors
A coding
(frontal side)



male
(BUS IN)

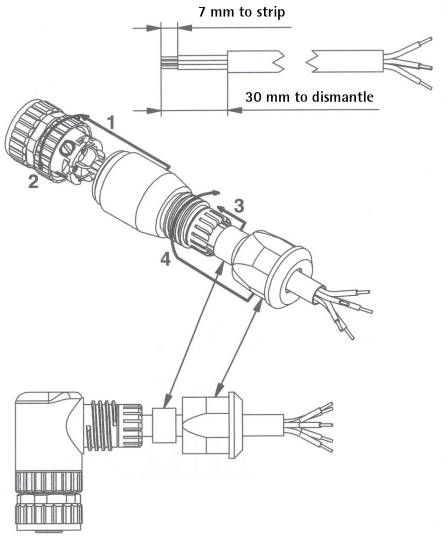


female
(BUS OUT)

Pin	Description
1	n.c.
2	n.c.
3	GND (RS-485)
4	Modbus A (RS-485), High, +
5	Modbus B (RS-485), Low, -
Case	Shielding[1]

[1] Lika's EC- pre-assembled cables only.
n.c. = not connected

NOTE
We suggest always connecting 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 (see the Figure). If metal connectors are used, connect the cable shield properly as recommended by the manufacturer.



4.3 Diagnostic LEDs (Figure 4 and Figure 6)

Eight LEDs located in the back of the actuator's enclosure are meant to show visually the operating or fault status of both the Modbus interface and the device. The meaning of each LED is explained in the following tables.

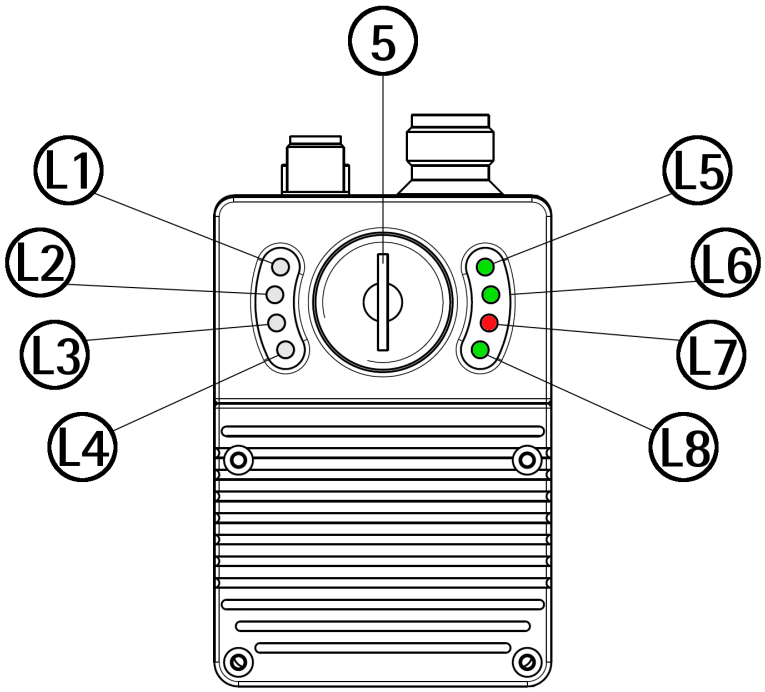


Figure 6: Diagnostic LEDs

	5	Screw plug for access to internal rotary / DIP switches	
L1	Not used	**L5**	Controller power supply information
L2	Not used	**L6**	Fieldbus interface status information
L3	Not used	**L7**	Active errors / faults information
L4	Not used	**L8**	Motor power supply information

LED L5 GREEN	Description
ON GREEN	It indicates that the power supply of the controller is turned on
OFF	It indicates that the power supply of the controller is turned off

LED L6 GREEN	Description
Blinking GREEN	The device is sending or receiving a message (network activity)
OFF	No send - receive activity

LED L7 RED	Description
ON GREEN	Active alarms, internal error. See the Alarms register [0x00] on page 80
OFF	No alarms active

LED L8 GREEN	Description
ON GREEN	It indicates that the motor is enabled (control loop activated)
OFF	It indicates that the motor is disabled (control loop deactivated)

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

4.4 Rotary / DIP switches (Figure 4 and Figure 6)

WARNING
The power supply must be turned off before performing this operation!

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

To access the rotary / DIP switches unscrew and pull out the screw plug **5** (Figure 6) in the back of the enclosure. Be careful to replace the screw plug at the end of the operation.

The rotary / DIP switches are located just beneath the screw plug.

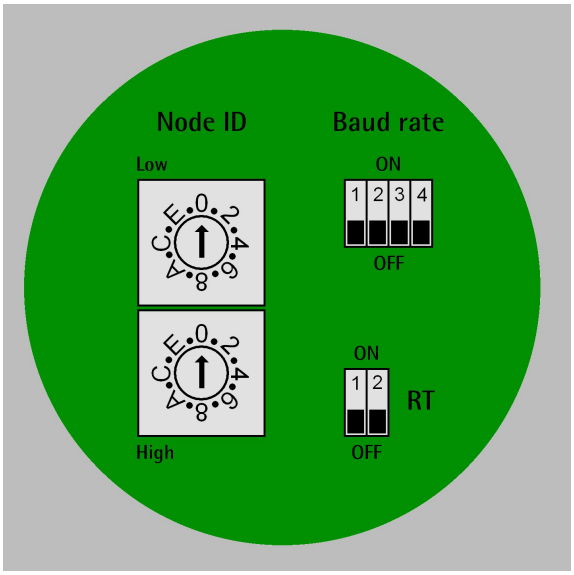


Figure 7: Rotary / DIP switches

4.4.1 Setting the node address: Node ID (Figure 7)

WARNING
The power supply must be turned off before performing this operation!

Set the node address expressed in hexadecimal notation.
The default address is "1".
The range of the node address is between "1" and "247" (247 = F7 hex).

EXAMPLE

Address 10 = 0A hex:	Address 25 = 19 hex:	Address 95 = 5F hex:
Low High	Low High	Low High

NOTE
The default address is "1".
The address "0" is reserved to identify a "broadcast" exchange (Master sends a request to all Slaves connected to the Modbus network). See the "7.1 Modbus Master / Slaves protocol principle" section on page 51.
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 an address higher than "247", the device will be set to "247" automatically.

The node address which is currently set in the unit can be read next to the **DIP switch node ID [0x0F]** register, see on page 86.

4.4.2 Setting the Baud rate and Parity bit (Figure 7)

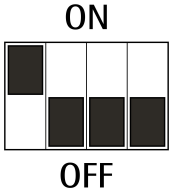
WARNING
The power supply must be turned off before performing this operation!

Set the binary value of the transmission rate and the parity bit according to the following table, please consider that: ON = 1; OFF = 0.

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

EXAMPLE
Set the baud rate to 9600 bits per second and the parity bit to Even:

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

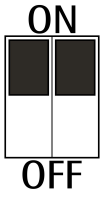
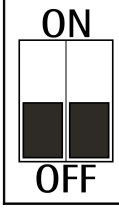


The data transmission rate that is currently set in the unit can be read next to the **DIP switch baud rate [0x0E]** register, see on page 86.

4.4.3 RT bus termination (Figure 7)

WARNING
The power supply must be turned off before performing this operation!

A bus termination resistor is provided and must be activated as line termination if the actuator is at the ends of the transmission line (i.e. it is either the first or the last device in the transmission line).
Use the RT switch to activate or deactivate the bus termination.

RT	Description
 1 = 2 = ON	Activated: when the actuator is either the first or the last device in the transmission line
 1 = 2 = OFF	Deactivated: when the actuator is neither the first nor the last device in the transmission line

5 Quick reference

5.1 Configuring the device using Lika's setting up software

RD6 Modbus DRIVECOD positioning units are supplied with a software expressly developed and released by Lika Electronic in order to allow an easy set up of the device. The program allows the operator to set the working parameters of the device; control manually some movements and functions; and monitor whether the device is running properly. The program is supplied for free and can be installed in any PC fitted with a Windows operating system (Windows XP or later). The executable file to launch the program is **ROTADRIVE INTERFACE.EXE** and is available in the enclosed documentation or at the address **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS/POSITIONING UNITS (DRIVECOD)**. The program is designed to be installed simply by copying the executable file to the desired location and there is **no installation** process. To launch it just double-click the file icon. To close the program press the **CLOSE** button in the title bar.

NOTE

Before starting the program, connect the device to the personal computer through serial ports. The serial interface of the DRIVECOD unit is a RS-485 type connector, while the serial standard in the personal computers (when available) is a RS-232 type connector. Therefore you must install a RS-485 / RS-232 converter, easily available in the market. Should the personal computer not be equipped with a serial port (RS-232 or RS-485), you must install a USB / RS-485 converter, easily available in the market too. For any information on the connection scheme and the cable pinout refer to the instruction sheet provided with the converter.

On the DRIVECOD side the cable must be connected to the M12 5-pin male connector (BUS IN). See the "Electrical connections" section on page 18. Always be sure that RX wire in the MODBUS DRIVECOD is connected up to TX wire in the PC and RX wire in the PC is connected up to TX wire in the MODBUS DRIVECOD.

A cable assembly fitted with M12 5-pin / USB connectors and integrated RS-485 converter is available on request; please contact Lika Electronic Technical Assistance & After Sale Service and quote the following code: **EXC-USB4-S54-GN-2-M12MC-S54**.

NOTE

If you use the EXC-USB4-S54-GN-2-M12MC-S54 USB connection cable, you are required to install the drivers of the USB Serial Converter and the USB Serial

Port first. The drivers are available in the Software folder of the actuator and downloadable from Lika's web site.

Default serial port settings as set at the factory by Lika Electronic for all RD6 Modbus positioning units are the followings:

RD6 MODBUS

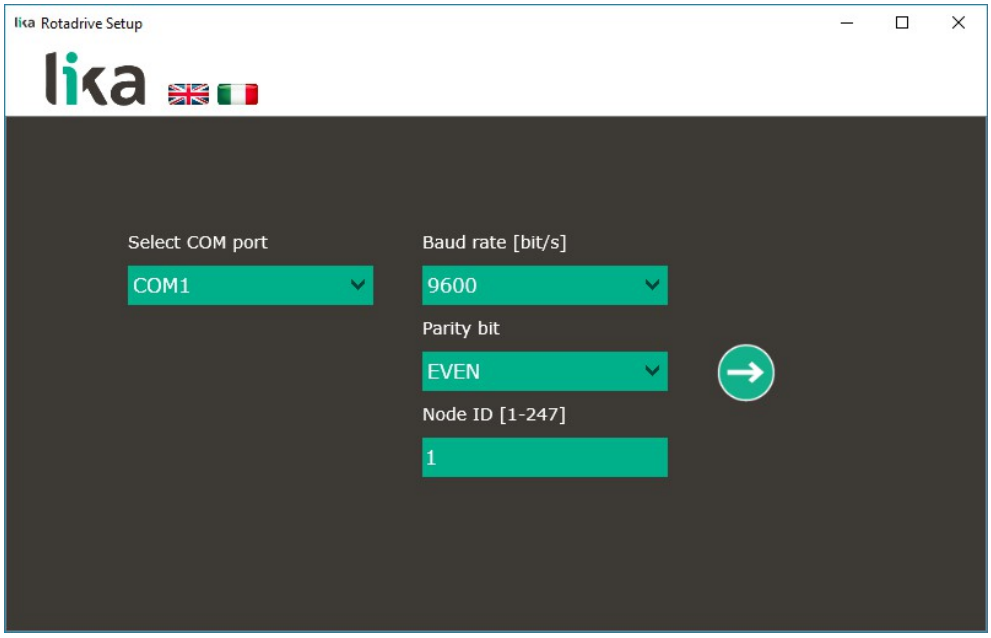
Serial port settings	**Default value**
Baud rate	9600
Byte size	8
Parity	Even
Stop bits	1

To configure the serial port of the RD6 device refer to the "4.4.2 Setting the Baud rate and Parity bit (Figure 7)" section on page 24.

The default node address for all RD6 MODBUS positioning units is "1". To set a custom node address of the RD6 device refer to the "4.4.1 Setting the node address: Node ID (Figure 7)" section on page 23.

5.2 "Serial configuration" page

When you start the program, the **Serial configuration** page appears on the screen.



First of all this page allows the operator to choose the language used to display texts and items in the user interface. Click on the **Italian flag** icon to choose the Italian language; click on the **UK flag** icon to choose the English language. The default language is according to the language of the operating system.

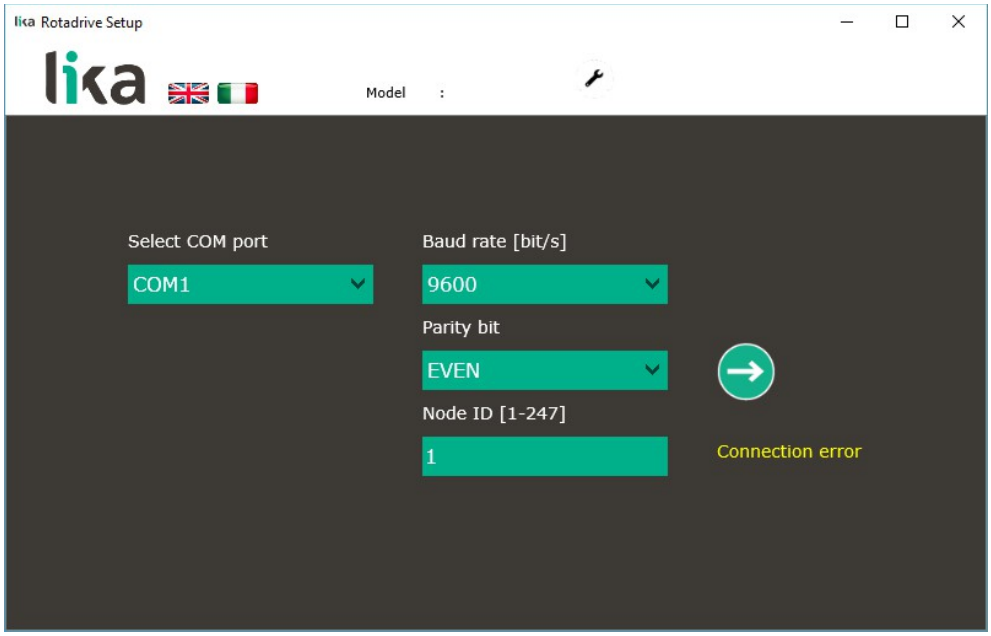
Furthermore, by clicking on Lika's logo button on the top left you enter Lika's web site **www.lika.biz**.

The **Serial configuration** page allows you to choose the serial port of the personal computer the RD6 unit is connected to (**Select COM port** drop-down box) and then set the configuration parameters. Serial port settings in the personal computer must compulsorily match those in the connected Lika device.

For serial port settings see the previous section.

Lastly set the node address of the device the personal computer is connected to through the **Node ID [1-247]** drop-down box (as previously stated, the default node address of RD6 rotary actuators with MODBUS interface is "1"). See on page 23.

Now you are ready to establish the connection to the Slave: press the **CONFIRMATION** button on the right to start searching for the connected device.



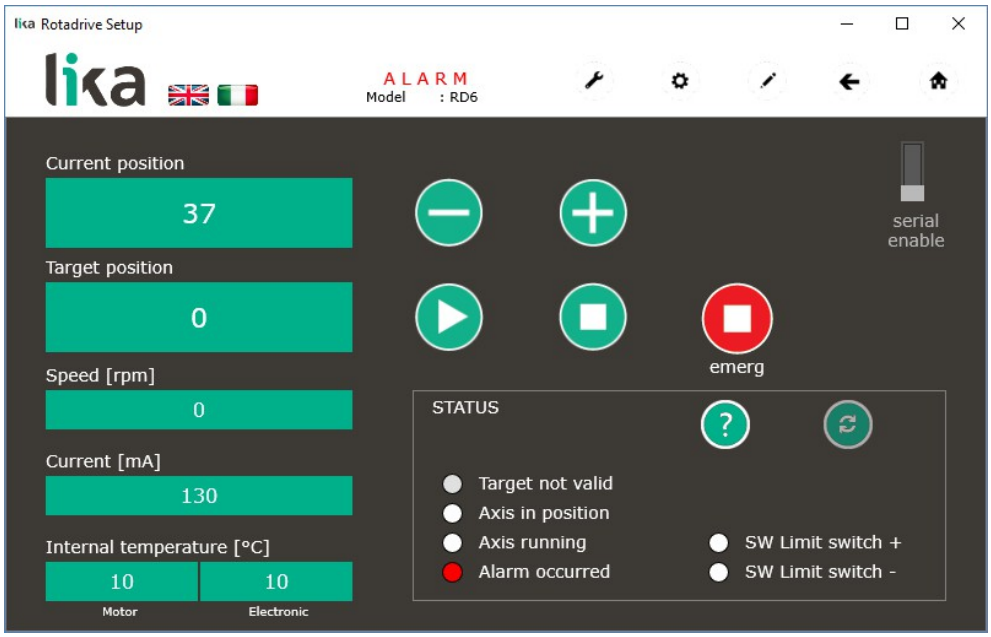
If the connection cannot be established (for instance, because of a wrong COM port setting or a wrong Node ID), the messages **Connection error**, **Device not responding**, **Error opening COM**, **Select COM port** or **Node ID error** may appear under the confirmation button . Please check the settings and try the connection again.

NOTE

Please note that the **FIRMWARE** button appears in the white bar at the top of the page (toolbar). It allows to enter the **Programming firmware** page. For complete information refer to the "5.7 "Programming firmware" page" section on page 35.

5.3 "Main" page

As soon as you press the **CONFIRMATION** button in the **Serial configuration** page, if the connection is established properly you enter the **Main** page.



In the white bar at the top of the page (toolbar) the DRIVECOD model you are connected to appears next to the **Model** label.
The ALARM warning message blinks as the unit is in emergency condition.
In the same bar the following buttons become available.

FIRMWARE

The **FIRMWARE** button allows to enter the **Programming firmware** page. This page allows the operator to upgrade the firmware of the DRIVECOD unit by downloading upgrading data to the flash memory. For complete information refer to the "5.7 "Programming firmware" page" section on page 35.

SETUP

The **SETUP** button allows to enter the **Parameters** page. In this page the list of the parameters available to set the RD6 positioning units (machine data) is displayed. For complete information refer to the "5.8 "Parameters" page" section on page 38.

SCHEDULE

The **SCHEDULE** button allows to enter the **Program** page. The functions available in the **Program** page allow the operator to create and save work programs in order to test the operation of the RD6 unit. For complete information refer to the "5.9 "Program" page" section on page 40.

ARROW BACK

The **ARROW BACK** button allows to go back to the page you looked through previously.

HOME

HOME button. When the **Main** page is displayed, press the **HOME** button to enter the **Serial configuration** page. When any other page is displayed, press the **HOME** button to enter the **Main** page.

In this page some further information on the position and states of the DRIVECOD unit appears.
The following items are available on the left.

Current position
See the **Current position [0x02-0x03]** registers on page 84.

Target position
See the **Target position [0x2B-0x2C]** registers on page 78. Set the position you need the unit to reach and then press the **ENTER** key in the keyboard to confirm it. As soon as you press the **START** button the device starts moving in order to reach the commanded position set next to this **Target position** item, then it stops and activates the **Axis in position** and **Target position reached** status bits (see the "5.5 "Status" box" section on page 33). For detailed

information on creating a complete cycle of movements and test the operation of the RD6 actuator (speed, acceleration, deceleration, etc.) refer to the "5.9 "Program" page" section on page 40.

Speed [rpm]
See the **Current velocity [0x04]** register on page 84.

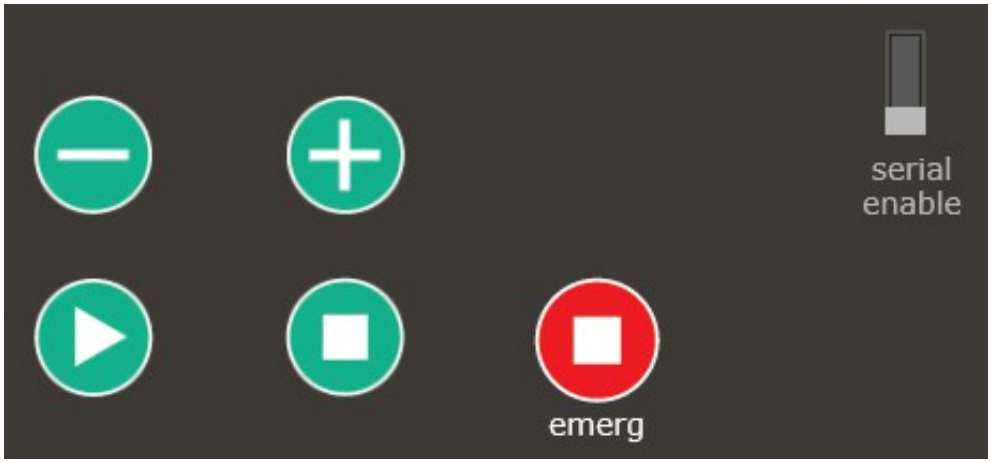
Current [mA]
See the **Current value [0x0B]** register on page 86.

Internal temperature [°C] Motor
See the **Temperature value [0x07]** register on page 84.

Internal temperature [°C] Electronic
See the **Temperature value [0x07]** register on page 84.

5.4 MODBUS commands

At the top right of the **Main** page some commands are available.



They allow to control manually and monitor the connected device.

JOG -
See the **Jog -** item on page 74. If the **Incremental JOG** item is enabled = ON, the **Jog step length** value that is set currently appears between the **JOG –** button and the **JOG +** button. See the "5.8 "Parameters" page" section on page 38.

JOG +

See the **Jog +** item on page 74. If the **Incremental JOG** item is enabled = ON, the **Jog step length** value that is set currently appears between the **JOG –** button and the **JOG +** button. See the "5.8 "Parameters" page" section on page 38.

START

Pressing the **START** button causes the unit to start running in order to reach the position set next to the **Target position** item. As soon as the commanded position is reached, the device stops and activates the **Axis in position** and **Target position reached** status bits (see the "5.5 "Status" box" section on page 33). For a normal halt of the device press the **STOP** button; for an immediate emergency halt press the **EMERGENCY** button. See the **Start** item on page 76.

STOP

Press the **STOP** button to force a normal halt of the device, respecting the deceleration values. See the **Stop** item on page 75.

EMERGENCY

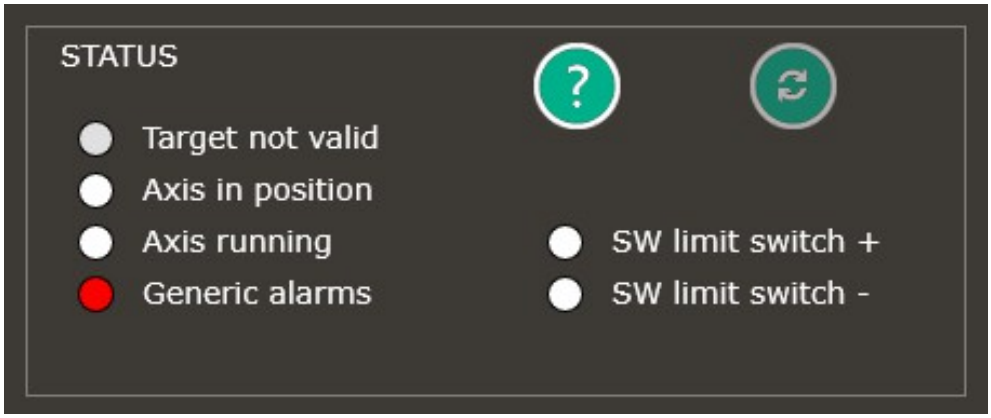
When the unit is running, press the **EMERGENCY** button to force an immediate halt in emergency condition. Press the **RESET** button to restore the normal work condition of the device (see the "5.5 "Status" box" section on page 33). See the **Emergency** item on page 76.

Serial enable

The operation of the **SERIAL ENABLE** slider is disabled (see Extra commands register [0x29] on page 73).

5.5 "Status" box

The **Status** box is available at the bottom right of the page.



Functions in the box provide short information about the status of the rotary actuator and allow to enter further pages containing more detailed information. The active alarms and serious states are lit red. The active ordinary states are lit green.

Please note that at power-on for safety reasons the RD6 unit is necessarily in an emergency condition: therefore when you first connect and enter the **Main** page the **Alarm occurred** warning is lit red. To know more about the specific active alarm enter the **Alarm and status** page by pressing the **STATUS & INFO** button, refer to the "5.6 "Alarm and status" page" section on page 34. To restore the **Idle** state of the device, press the **RESET** button in the box. Alarm warnings and emergencies will be reset and the normal work condition of the device will be restored.

For complete information on the states and alarms that appear in this box please refer to the **Status word [0x01]** register on page 82; and to the **Alarms register [0x00]** on page 80.

STATUS & INFO

Press this button to enter the **Alarm and status** page where specific information on the states and alarms that are active in the rotary actuator can be found. Refer to the "5.6 "Alarm and status" page" section on page 34.

RESET

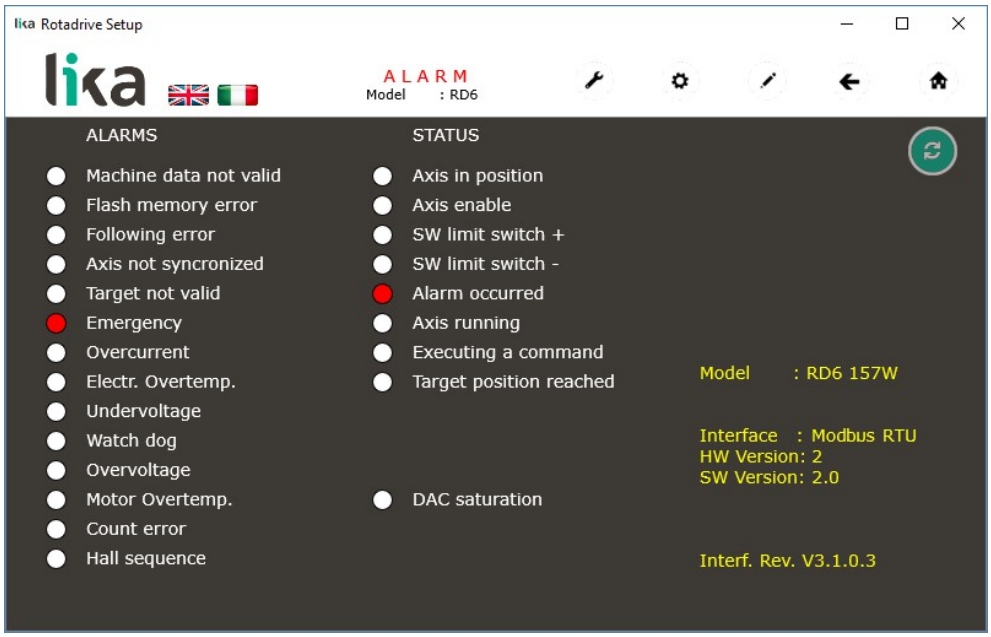
If an alarm is active, it is signalled through the generic warnings in the **Status** box. To know more about the specific active alarm enter the **Alarm and status** page by pressing the **STATUS & INFO** button, refer to the "5.6 "Alarm and status" page" section on page 34. Press this button to reset the alarm and restore the normal work condition of the device. This button is available only if

the MODBUS commands are enabled (see the **SERIAL ENABLE** slider 

). See the **Alarm reset** item on page 75.

5.6 "Alarm and status" page

When you press the **STATUS & INFO** button  in the **Status** box you enter the **Alarm and status** page.



As previously stated the **Status** box provides short information about the states and the alarms that are active in the rotary actuator. This **Alarm and status** page provides more detailed information on the active statuses and alarms. The active alarms and serious states are lit red. The active ordinary states are lit green.

For complete information on the states and alarms that appear in this page please refer to the Status word [0x01] register on page 82; and to the Alarms register [0x00] on page 80.

Furthermore the page provides detailed information on the connected actuator and the software tool. Data is listed in yellow at the bottom right of the page.

In particular you can found:

- **Model**: the model of the connected actuator;
- **Interface**: the protocol of the connected actuator;
- **HW version**: the hardware version of the connected actuator;

- **SW version**: the software version of the connected actuator;
- **Interf. Rev.**: the release of the software tool.

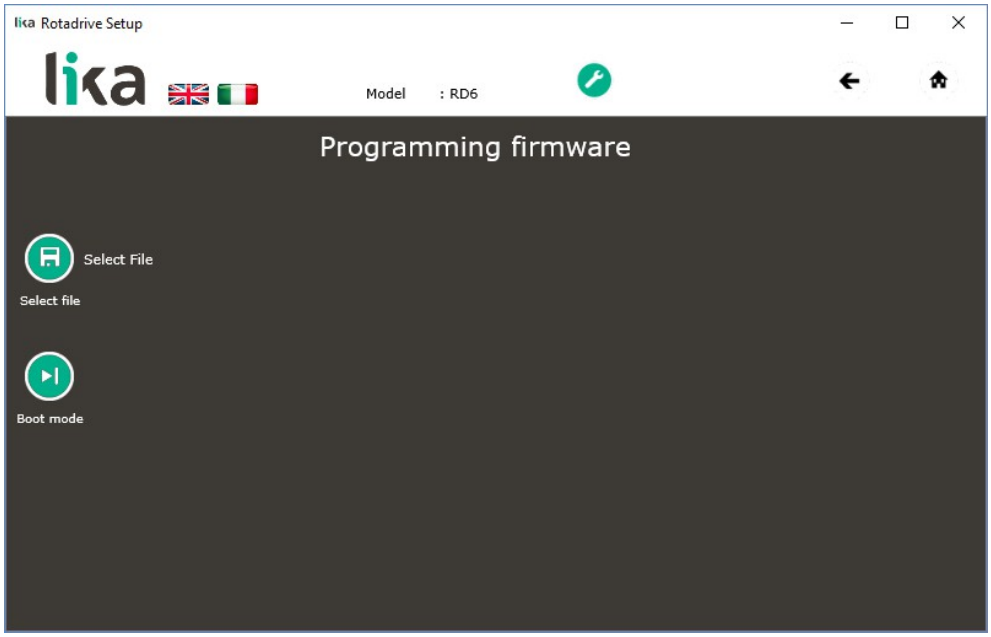
RESET

RESET button . See the previous page.

Press the **HOME** button to go back to the **Main** page.

5.7 "Programming firmware" page

By pressing the **FIRMWARE** button in the toolbar the operator enters the **Programming firmware** page.



The functions available in this page allow the operator to upgrade the firmware of the DRIVECOD unit by downloading upgrading data to the flash memory.
The firmware is a software program which controls the functions and operation of a device; the firmware program, sometimes referred to as "user program", is stored in the flash memory integrated inside the DRIVECOD unit. DRIVECOD units 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.

WARNING
The firmware upgrading process for any DRIVECOD unit has to be accomplished by skilled and competent personnel. If the upgrade is not performed according to the instructions provided or a wrong or incompatible firmware program is installed, then the unit may not be updated correctly, in some cases preventing the DRIVECOD unit from working.
If the latest firmware version is already installed in the DRIVECOD unit, you do not need to proceed with any new firmware installation. Current firmware version can be verified from the **SW Version** item in the **Alarm and status** page after connecting properly to the unit (see on page 34).
If you are not confident that you can perform the update successfully please contact Lika Electronic Technical Assistance & After Sale Service.

To upgrade the firmware program please proceed as follows:

1. make sure that the following configuration parameters are set in the serial port of the DRIVECOD unit: baud rate = 9600 bits/s; parity = even; if they are set otherwise, please set them following the instructions in the "4.4.2 Setting the Baud rate and Parity bit (Figure 7)" section on page 24;
2. make sure that the DRIVECOD unit you need to update is the only node connected to the personal computer;
3. connect to the unit, go online and then enter the **Programming firmware** page;
4. when you switch on the power supply, if user program is not present in the flash memory, you are not able to connect to the unit through the **Serial configuration** page; when this happens you need to enter directly the **Programming firmware** page; after the attempt to connect has failed the **FIRMWARE** button 

becomes available in the toolbar of the **Serial configuration** page; make sure that the correct serial port of the personal computer connected to the DRIVECOD unit is selected in the **Serial configuration** page;
5. press the **SELECT FILE** button 

; once you press the button the **Open** dialogue box appears on the screen: the operator must open the folder where the firmware upgrading .BIN file released by Lika Electronic is stored;

WARNING
Please note that for each DRIVECOD model having its own bus interface an appropriate firmware file is available. Make sure that you have the appropriate update for your DRIVECOD model. The .BIN file released by Lika Electronic has a file name that must be interpreted as follows.
For instance: RD6_EP_157_H1S1.BIN, where:

- RD6 = DRIVECOD unit model;
- EP = bus interface of the DRIVECOD unit (CB = CANopen; EC = EtherCAT; EP = EtherNet/IP; MB = MODBUS RTU; MT = MODBUS TCP; PB = Profibus; PL = POWERLINK; PT = Profinet);
- 157 = motor power;
- H1 = hardware version;
- S1 = firmware version.

6. select the .BIN file and confirm the choice by pressing the **OPEN** button, the dialog box closes;
7. the complete path of the file you have just confirmed appears next to the **SELECT FILE** button  ;
8. now press the **BOOT MODE** button  to enter the Boot Mode state;
9. the **UPLOAD** button  appears below in the page; press the button to start the firmware upgrading process;
10. a green download progress bar with percentage is displayed next to the **UPLOAD** button  ;

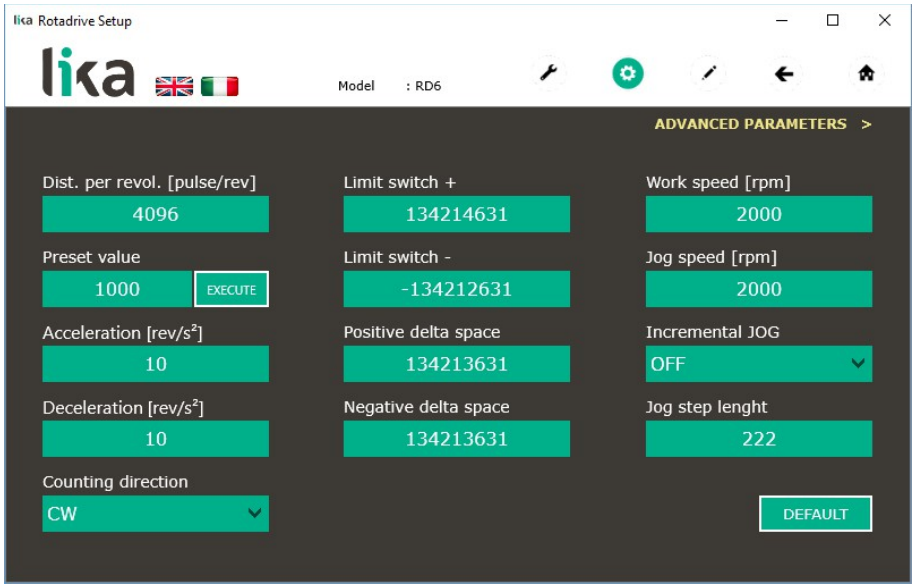
WARNING
Do not exit the **Programming firmware** page during installation, the process will be aborted!

11. as soon as the operation is carried out successfully, the **OK** message is displayed for a while; then the actuator reboots and the **Serial configuration** page appears on the screen;
12. reconnect to the DRIVECOD unit and restore the normal work condition.

Press the **HOME** button  to go back to the **Main** page.

5.8 "Parameters" page

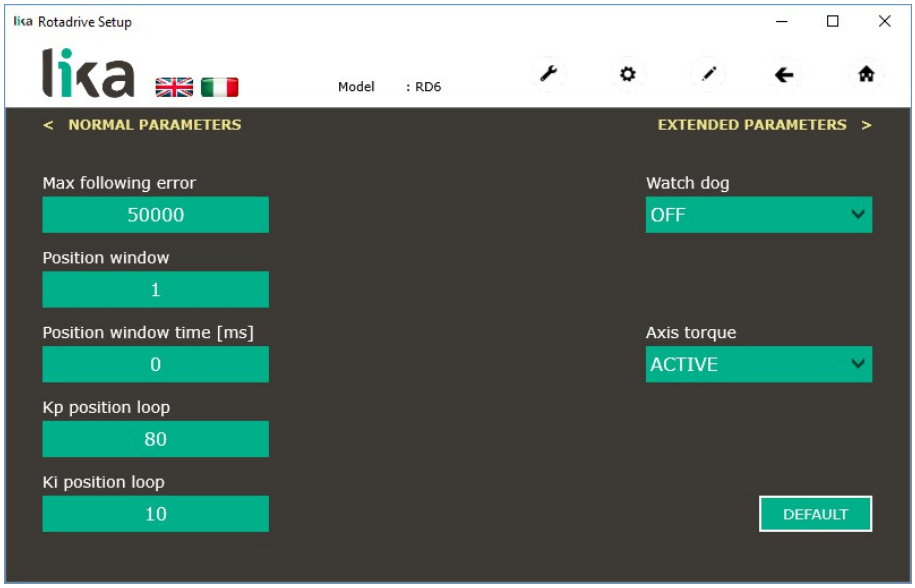
Press the **SETUP** button in the toolbar to configure the device and set the parameters. The page below will appear.



In this page the list of the "normal parameters" available to set the RD6 positioning units is displayed. Parameters in the page need to be set more usually when you configure and test a new program of the actuator.

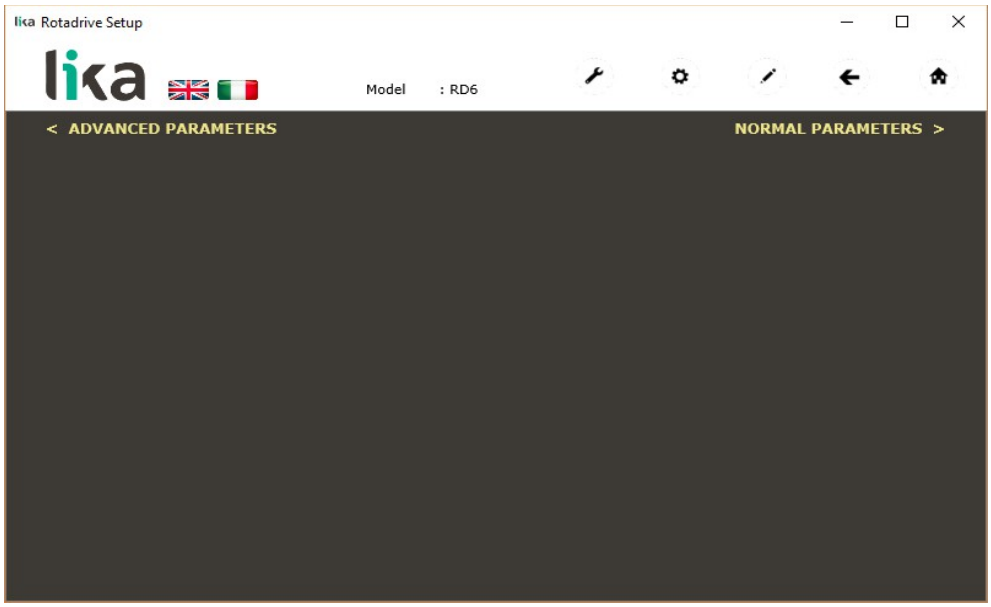
ADVANCED PARAMETERS > ADVANCED PARAMETERS >

A further page is accessible by pressing the **ADVANCED PARAMETERS >** button ADVANCED PARAMETERS >: the "advanced parameters" that need to be set more seldom are listed in the page.



EXTENDED PARAMETERS EXTENDED PARAMETERS >

Press the **EXTENDED PARAMETERS >** button EXTENDED PARAMETERS > to enter the page where the outputs can be enabled. RD6 rotary actuator does not provide output signals, so a blank page will appear.



NORMAL PARAMETERS NORMAL PARAMETERS

Press the **NORMAL PARAMETERS** button NORMAL PARAMETERS to go back to the first **Parameters** page.

For detailed information on the function and the setting of the parameters refer to the "8.1.1 Holding Register parameters" section on page 66.

The values that are currently set in the unit are shown under each field.
To enter a new value type it in the field and then press the **ENTER** key in the keyboard. If you set a value that is not allowed (out of range), at confirmation prompt the minimum or maximum value will be set instead. In some cases you are required to select the desired value by means of the drop-down box.
As soon as the new value is confirmed, it is downloaded to the unit and saved automatically.

DEFAULT DEFAULT

When you need to load the 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) press the **DEFAULT** button DEFAULT. For any further information on loading the default parameters refer to the **Load default**

parameters variable on page 77. The complete list of the machine data parameters and the relevant default values as set by Lika Electronic are available on page 96.

In the **Parameters** pages the following functions are also available.

EXECUTE 

The **EXECUTE** button 

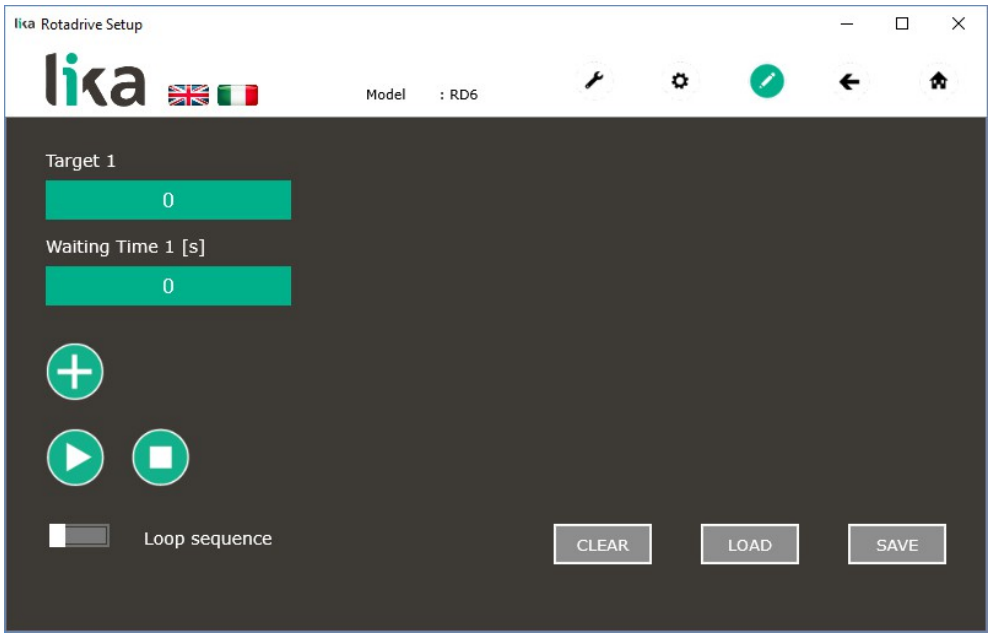
is placed next to the **Preset** item in the first **Parameters** page. When you enter a new value next to the **Preset** item, the Preset is saved but not activated. Press the **EXECUTE** button 

to activate the Preset value for the current position of the actuator's shaft. For any further information on activating the Preset value refer to the bit 11 **Setting the preset** in the Control Word [0x2A] register on page 77. Refer also to the Preset [0x12-0x13] registers on page 72.

5.9 "Program" page

Press the **SCHEDULE** button 

in the toolbar to enter the **Program** page. The page below will appear.



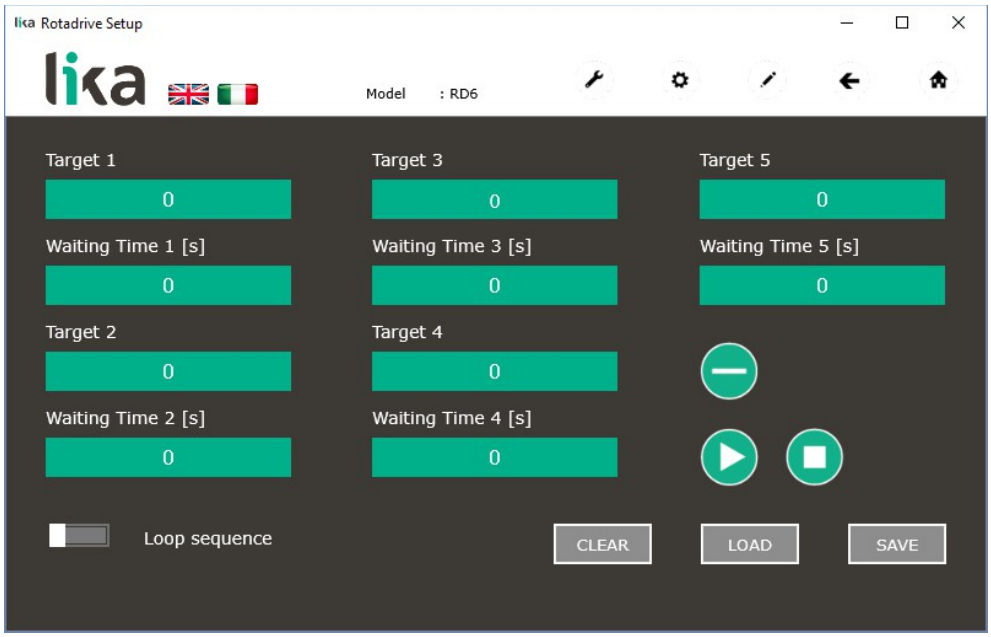
The functions available in the **Program** page allow the operator to create and save work programs for the RD6 unit. In this way it is possible to test the

operation of the actuator (speed, acceleration, deceleration, etc.) by setting up to five targets and commanding their execution.

Target 1 ... 5

The first position the device is commanded to reach (target position) must be set next to the **Target 1** item. Press the **ENTER** key in the keyboard to confirm.

It is possible to enter up to five subsequent positions. Press the **PLUS** button to add more target positions for the actuator to reach.

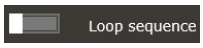


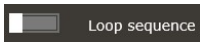
You can press the **MINUS** button to delete the last target.

Waiting Time 1 [s]

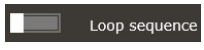
Next to the **Waiting Time 1 [s]** item you must set the interval between the first and the second step (commanded movement). Press the **ENTER** key in the keyboard to confirm.

Loop sequence



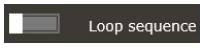
The **Loop sequence** slider 

enables / disables the "loop" function, i.e. after pressing the **START** button the device goes on running and executing the set steps without interruption, from **Target 1** to **Target 5** (if enabled) and again from **Target 1** to **Target 5**, until you press the **STOP** button .

If the **Loop sequence** slider 

is activated, when you press the **START** button , the device starts moving in order to reach the first commanded position **Target 1**; a green light appears next to the field and a

green progress bar is shown at the top of the page; as soon as the commanded position set next to the **Target 1** item is reached, the device stops. After the set interval **Waiting Time 1 [s]** has expired, a green light appears next to the **Target 2** item and the RD6 unit restarts running in order to reach the second commanded position **Target 2**; again a green progress bar is shown at the top of the page; and so on, from the first to the fifth commanded position (if enabled) and then again from the first to the fifth commanded position without interruption, until you press the **STOP** button .

If the **Loop sequence** slider 

is not activated, when you press the **START** button , the device starts running in order to reach the first commanded position **Target 1**; as soon as the first commanded position **Target 1** is reached, the device stops and waits for the set interval **Waiting Time 1 [s]** to expire: the sequence of movements is carried out; you must then press the **START** button again to command the unit to reach the second position **Target 2**; and so on.

The following buttons are available in the page:

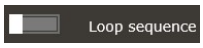
PLUS

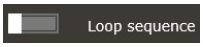
Press the **PLUS** button to add more target positions for the actuator to reach.

MINUS

Press the **MINUS** button to delete the last target.

START

Press the **START** button to command the unit to reach the set target position. If the **Loop sequence** slider 

is activated, you are required to press the **START** button just once: the actuator will reach in sequence all the commanded positions that are enabled. If the **Loop sequence** slider 

is not activated, you must press the **START** button after each step to go on.

STOP

Press the **STOP** button to stop the sequence of movements.

CLEAR 

Press the **CLEAR** button 

to zero-set the counter designed to count the steps during the execution of the running program: when the operator presses the **START** button the device will start running from step 1, i.e. in order to

reach the first commanded position **Target 1**, whatever the position reached before the counter was zero-set.

LOAD 

Press the **LOAD** button 

to load a work program that has been saved previously. Once you press the button, the **Open** dialog box appears on the screen: the operator must open the folder where the previously saved .dat file is stored, then select it and finally confirm the choice by pressing the **OPEN** button, the dialog box closes and the work values are loaded automatically.

SAVE 

Press the **SAVE** button 

to save the parameters of the work program you have just created. Once you press the button the **Save as** dialog box appears on the screen: the operator must type the name of the .dat file and specify the path where the file has to be stored. When you press the **SAVE** button to confirm, the dialog box closes. Set values are saved automatically.

5.10 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, see on page 15 ff;
- execute the electrical connections, see on page 18 ff;
- set the data transmission rate (baud rate; see on page 24); the default value set by Lika Electronic at factory set-up is "baud rate = 9600 bit/s, parity = Even";
- set the node address (node ID; see on page 23); the default value set by Lika Electronic at factory set-up is "1";
- switch on the +24Vdc power supply (in both motor and controller);
- check that the LED L5 is ON green while the LED L7 is ON red;
- to resume the normal work condition reset the active emergency: switch high ("=1") the **Emergency** bit 7 of the **Control Word [0x2A]** (see on page 74); reset the active alarms: switch high ("=1") the **Alarm reset** bit 3 of the **Control Word [0x2A]** (see on page 74). The LED L5 is ON green, the LED L7 switches off while the LED L6 starts blinking;
- set a proper value next to the **Distance per revolution [0x00]** item (register 1; see on page 67);
- set a proper value next to the **Jog speed [0x0D]** item (register 13; see on page 71);
- set a proper value next to the **Work speed [0x0E]** item (register 14; see on page 72);

- if required, set a proper value next to the **Preset [0x12-0x13]** item (registers 18-19; see on page 72);
- set the limit switch values next to the **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items; see on page 69);
- set the commanded position next to the **Target position [0x2B-0x2C]** item (registers 44-45; see on page 78);
- save the new setting values (**Save parameters** command; see on page 76).

Use the **Jog +**, **Jog -**, **Start** and **Stop** commands in the **Control Word [0x2A]** (see on page 74) to move the axis and reach the commanded position.

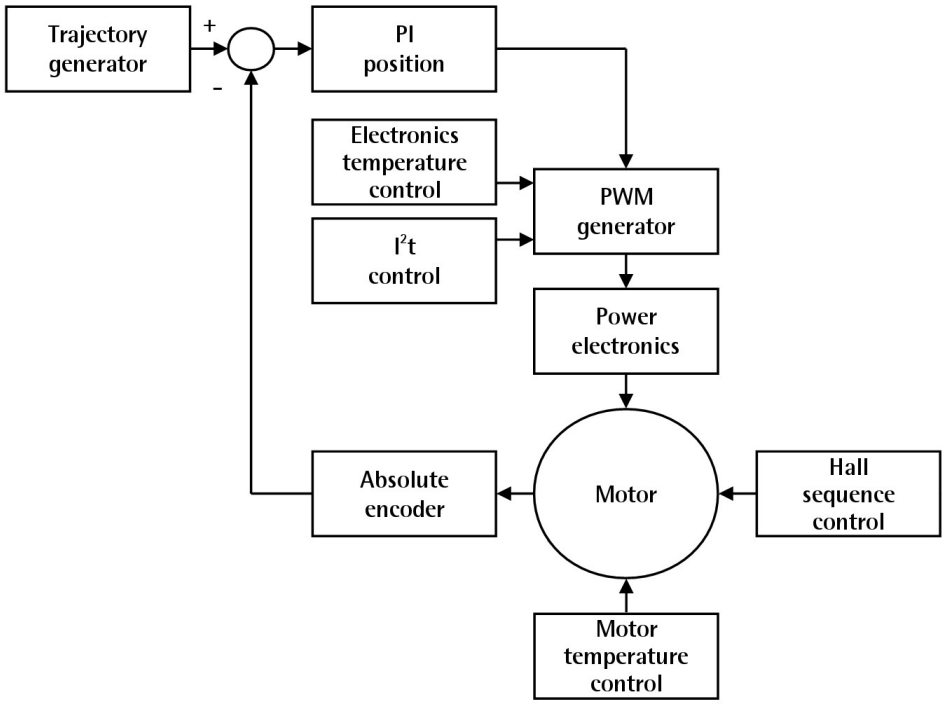
NOTE

The parameters **Distance per revolution [0x00]**, **Jog speed [0x0D]**, **Work speed [0x0E]**, **Preset [0x12-0x13]**, **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** are closely related, hence you have to be very attentive when you need to change the value in any of them. For any further information please refer to page 48.

6 Functions

6.1 Working principle

The following scheme is intended to show schematically the working principle of the system control logic.



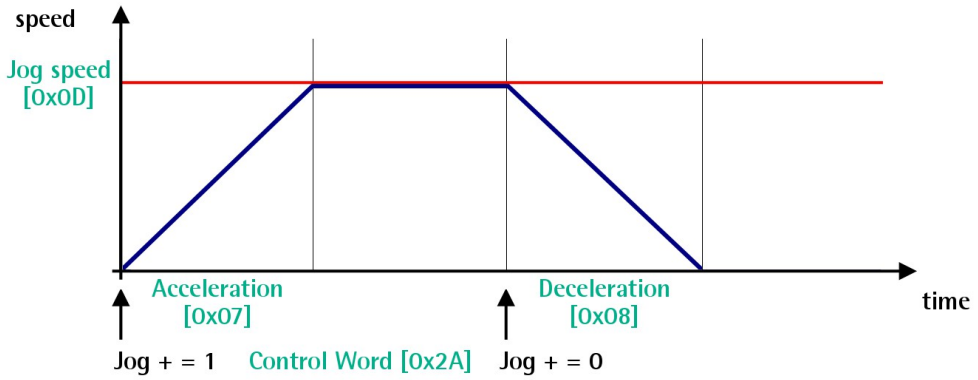
6.2 Movements: jog and positioning

Two kinds of movement are available in the DRIVECOD positioning unit, they are:

- Jog: speed control;
- Positioning: position and speed control.

Jog: speed control

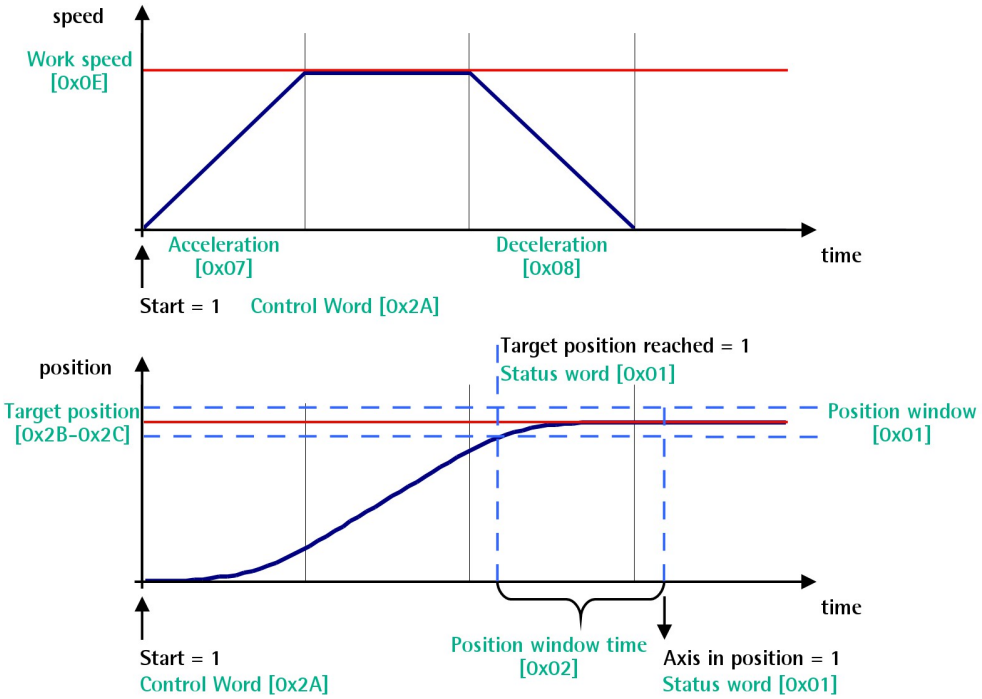
This kind of control is intended to generate a speed trajectory which allows the rotation speed of the DRIVECOD unit shaft to be equal to the value set in the **Jog speed [0x0D]** parameter.



When the bit 0 **Jog +** in the **Control Word [0x2A]** is "=1", the motor accelerates toward the positive direction according to the value set next to the **Acceleration [0x07]** register; if the available travel is long enough it reaches the speed set next to the **Jog speed [0x0D]** register. As soon as the bit 0 **Jog +** in the **Control Word [0x2A]** goes low ("=0"), the motor decelerates according to the value set next to the **Deceleration [0x08]** register until it stops.
Setting the bit 1 **Jog -** in the **Control Word [0x2A]** to "=1" causes the motor to run in the opposite direction (negative direction) respecting the work phases already described above.

Positioning: position and speed control

This kind of control is a point-to-point movement and the maximum reachable speed is equal to the value set in the **Work speed [0x0E]** parameter; the set speed can be reached only if the available travel is long enough.

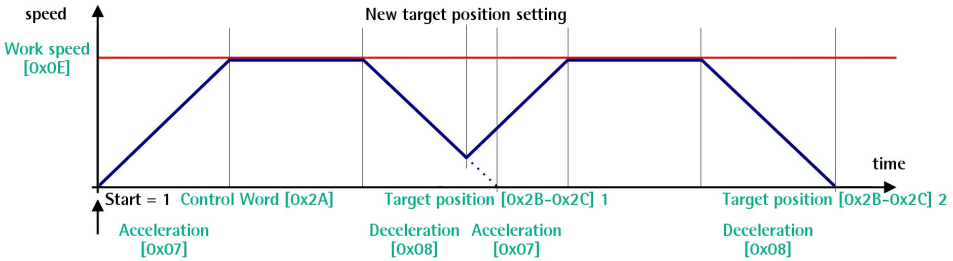


When the bit 6 **Start** in the **Control Word [0x2A]** is "=1", the motor starts moving and accelerates according to the value set next to the **Acceleration [0x07]** register in order to reach the target position as set next to the **Target position [0x2B-0x2C]** registers. If the available travel is long enough it reaches the speed set next to the **Work speed [0x0E]** register. The movement direction can be either positive or negative according to the target position to reach. As soon as the axis is within the tolerance window limits set next to the **Position window [0x01]** register, the bit 8 **Target position reached** in the **Status word [0x01]** goes high ("=1"). When the position is within the tolerance window limits set next to the **Position window [0x01]** register, after the delay set next to the **Position window time [0x02]** item, the bit 0 **Axis in position** in the **Status word [0x01]** goes high ("=1"). The motor decelerates according to the value set next to the **Deceleration [0x08]** register in order to reach the halt position according to the set target position.

NOTE

Position override function

It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the Target position [0x2B-0x2C] registers.



6.3 Distance per revolution [0x00], Jog speed [0x0D], Work speed [0x0E], Preset [0x12-0x13], Positive delta [0x09-0x0A] and Negative delta [0x0B-0x0C]

The variables Distance per revolution [0x00], Jog speed [0x0D], Work speed [0x0E], Preset [0x12-0x13], Positive delta [0x09-0x0A] and Negative delta [0x0B-0x0C] are closely related, hence you have to be very attentive every time you need to change the value in any of them.
Should a new setting be necessary, please comply with the following procedure:

- set a proper value next to the Distance per revolution [0x00] item (register 1, see on page 67);
- set a proper value next to the Jog speed [0x0D] item (register 14, see on page 71);
- set a proper value next to the Work speed [0x0E] item (register 15, see on page 72);
- set a proper value next to the Preset [0x12-0x13] item (registers 19-20, see on page 72);
- check the value next to the Positive delta [0x09-0x0A] item (registers 10-11, see on page 69);
- check the value next to the Negative delta [0x0B-0x0C] item (registers 12-13, see on page 70);
- save the new values (**Save parameters** item, bit 9 in the Control Word [0x2A] register, see on page 76).

Each time you change the value in **Distance per revolution [0x00]** you must then update the value in **Preset [0x12-0x13]** in order to define the zero of the axis as the system reference has changed.
After having changed the parameter in the **Preset [0x12-0x13]** registers it is not necessary to set new values for the travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]**.

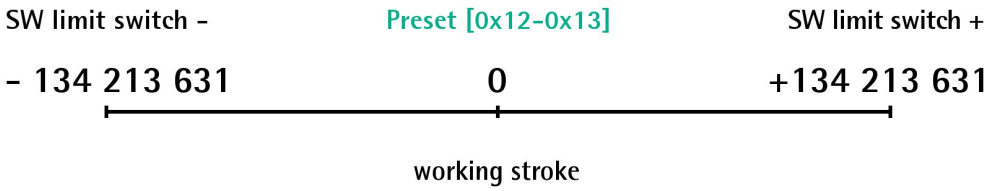
The number of revolutions managed by the system is 32,768 in negative direction and 32,768 in positive direction assuming the **Preset [0x12-0x13]** value as a reference.
The value set next to the **Positive delta [0x09-0x0A]** item plus the value set in the **Preset [0x12-0x13]** parameter is the maximum forward travel (positive travel) starting from the preset (the value is expressed in pulses).
The value set next to the **Negative delta [0x0B-0x0C]** item subtracted from the value set in the **Preset [0x12-0x13]** parameter is the maximum backward travel (negative travel) starting from the preset (the value is expressed in pulses).

WARNING
Please note that the parameters listed hereafter are closely related to the **Distance per revolution [0x00]** parameter; hence when you change the value in **Distance per revolution [0x00]** also the value in each of them necessarily changes. They are: **Position window [0x01]**, **Max following error [0x03-0x04]**, **Positive delta [0x09-0x0A]**, **Negative delta [0x0B-0x0C]**, **Target position [0x2B-0x2C]**, **Current position [0x02-0x03]** and **Position following error [0x05-0x06]**.

EXAMPLE 1
Default values:
Distance per revolution [0x00] = 4096 steps per revolution
Max. **Work speed [0x0E]**: 2000 rpm
Preset [0x12-0x13] = 0

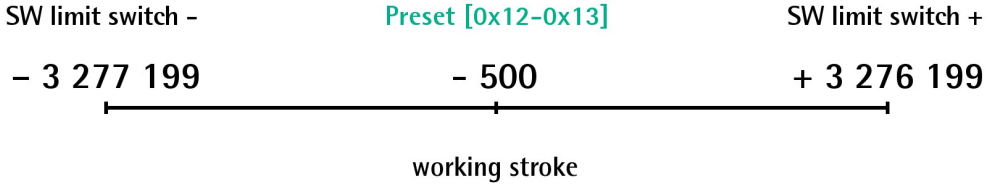
Positive delta [0x09-0x0A] and **Negative delta [0x0B-0x0C]** max. values = 134 213 631 = (4,096 steps per revolution x 32,768 revolutions) - 1 – 4,096 steps (i.e. 1 revolution for safety reasons) when **Preset [0x12-0x13]** = 0
Max. **SW limit switch +** = 0 + 134 213 631 = + 134 213 631 pulses (forward travel)
Max. **SW limit switch -** = 0 - 134 213 631 = - 134 213 631 pulses (backward travel)

Therefore, when **Preset [0x12-0x13]** = 0, the working stroke of the axis will span the overall positive and negative limits range, that is max. **SW limit switch +** = + 134 213 631 and max. **SW limit switch -** = – 134 213 631.



EXAMPLE 2
DRIVECOD RD6 positioning unit is joined to a worm screw having 1 mm (0.039") pitch and you need to have a hundredth of a millimetre resolution.

Distance per revolution [0x00] = 100 steps per revolution
Max. **Work speed [0x0E]** = 73 rpm (100 * 3000 / 4096 = 73)
Preset [0x12-0x13] = -500 (ex. thickness of the tool)
Max. **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** = (100 steps per revolution * 32,768 revolutions) - 1 – 100 steps (i.e. 1 revolution for safety reasons) = 3 276 699 pulses
Max. **SW limit switch +** = (-500) + 3 276 699 = 3 276 199 pulses (forward travel)
Max. **SW limit switch -** = (-500) – 3 276 699 = -3 277 199 pulses (backward travel)
Therefore, when **Preset [0x12-0x13]** = - 500, the working stroke of the axis will span the following positive and negative limits range, that is max. **SW limit switch +** = + 3 276 199 and max. **SW limit switch -** = – 3 277 199.



7 Modbus® interface

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

7.1 Modbus Master / Slaves protocol principle

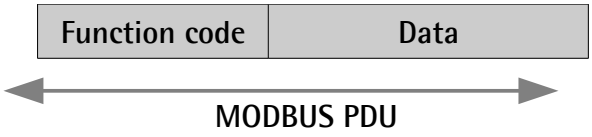
The Modbus Serial Line protocol is a Master – Slaves protocol. One only Master (at the same time) is connected to the bus and one or several (up to 247) Slave nodes are also connected to the same serial bus. A Modbus communication is always initiated by the Master. The Slave nodes will never transmit 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:

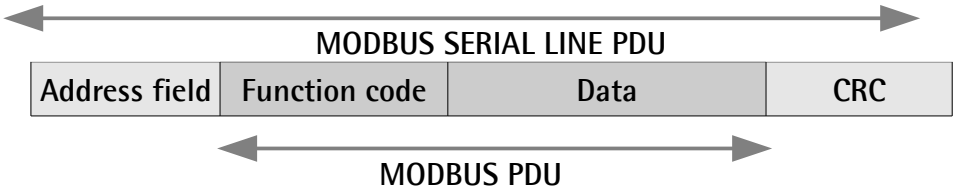
- **UNICAST mode**: in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must have a unique address (from 1 to 247) so that it can be addressed independently from the 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.

7.2 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 the "4.4.1 Setting the node address: Node ID (Figure 7)" section on page 23), 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 "7.4 Function codes" section on page 56.
- **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 "7.4 Function codes" section on page 56).
- **CRC (Cyclic Redundancy Check)**: error checking field is the result of a "Redundancy Check" 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".

7.3 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.

7.3.1 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.

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

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

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

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

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

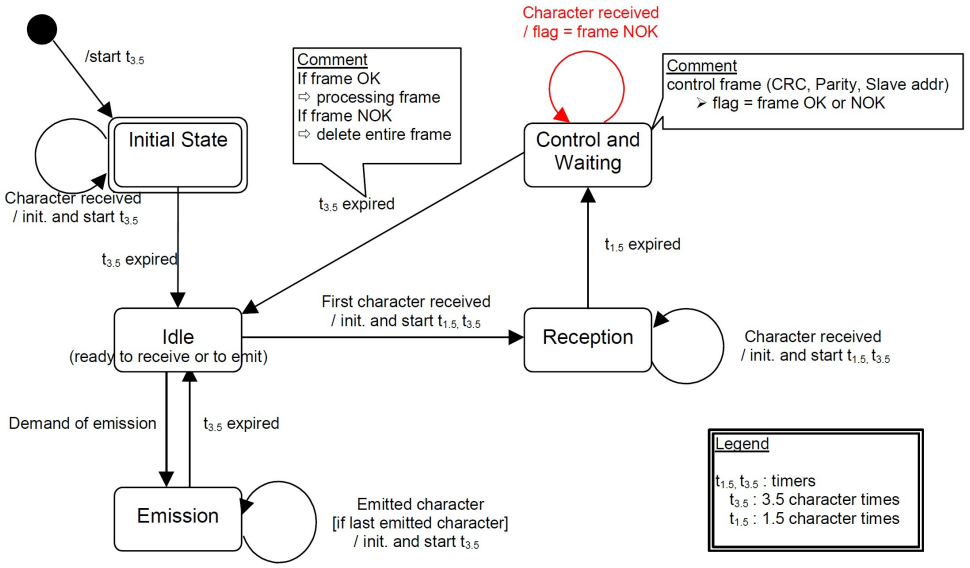
* When "No parity" is activated, the parity bit is replaced by a stop bit.

The default parity mode must be even parity.

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

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

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



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

7.4 Function codes

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

7.4.1 Implemented function codes

Lika RD6 Modbus positioning units 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 the "8.1.1 Holding Register parameters" section on page 66.

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 parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9).

Request	
Field name	(Hex)
Function	**03**
Starting address Hi	**00**
Starting address Lo	**07**
No. of registers Hi	**00**
No. of registers Lo	**02**

Response	
Field name	(Hex)
Function	**03**
Byte count	**04**
Register 8 value Hi	**00**
Register 8 value Lo	**0A**
Register 9 value Hi	**00**
Register 9 value Lo	**0A**

As you can see in the table, **Acceleration [0x07]** parameter (register 8) contains the value 00 0A hex, i.e. 10 in decimal notation; **Deceleration [0x08]** parameter (register 9) contains the value 00 0A hex, i.e. 10 in decimal notation.

The full frame needed for the request to read the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][03][00][07][00][02][75][CA]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)
[00][02] = number of requested registers
[75][CA] = CRC

The full frame needed to send back the values of the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][03][04][00][0A][00][0A][5A][36]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[00][0A] = value of register 8 **Acceleration [0x07]**, 00 0A hex = 10 dec
[00][0A] = value of register 9 **Deceleration [0x08]**, 00 0A hex = 10 dec
[5A][36] = 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 the "8.1.2 Input Register parameters" section on page 80.

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 [0x02-0x03]** parameter (input registers 3 and 4).

Request	
Field name	(Hex)
Function	**04**
Starting address Hi	**00**
Starting address Lo	**02**
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, the **Current position [0x02-0x03]** parameter (input registers 3 and 4) contains the value 00 00 2F F0 hex, i.e. 12,272 in decimal notation.

The full frame needed for the request to read the **Current position [0x02-0x03]** parameter (input registers 3 and 4) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][04][00][02][00][02][D0][0B]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[00][02] = starting address (**Current position [0x02-0x03]** parameter, register 3)
[00][02] = number of requested registers
[D0][0B] = CRC

The full frame needed to send back the value of the **Current position [0x02-0x03]** parameter (registers 3 and 4) 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 3 **Current position [0x02-0x03]**, 00 00 hex = 0 dec
[2F][F0] = value of register 4 **Current position [0x02-0x03]**, 2F F0 hex = 12,272 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 the "8.1.1 Holding Register parameters" section on page 66.

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 the value 00 96 hex (= 150 dec) in the **Acceleration [0x07]** parameter (register 8).

Request	
Field name	(Hex)
Function	**06**
Register address Hi	**00**
Register address Lo	**07**
Register value Hi	**00**
Register value Lo	**96**

Response	
Field name	(Hex)
Function	**06**
Register address Hi	**00**
Register address Lo	**07**
Register value Hi	**00**
Register value Lo	**96**

As you can see in the table, the value 00 96 hex, i.e. 150 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8).

The full frame needed for the request to write the value 00 96 hex (= 150 dec) in the **Acceleration [0x07]** parameter (register 8) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][06][00][07][00][96][B8][65]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[00][96] = value to be set in the register
[B8][65] = CRC

The full frame needed to send back a response following the request to write the value 00 96 hex (= 150 dec) in the **Acceleration [0x07]** parameter (register 8) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][06][00][07][00][96][B8][65]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[00][96] = value set in the register
[B8][65] = 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 the "8.1.1 Holding Register parameters" section on page 66.

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 values 150 and 100 dec in the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) respectively.

Request	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**07**
Quantity of registers Hi	**00**
Quantity of registers Lo	**02**
Byte count	**04**
Register 8 value Hi	**00**

Response	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**07**
Quantity of registers Hi	**00**
Quantity of registers Lo	**02**

Register 8 value Lo	**96**
Register 9 value Hi	**00**
Register 9 value Lo	**64**

As you can see in the table, the value 00 96 hex, i.e. 150 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8); the value 00 64 hex, i.e. 100 in decimal notation, is set in the **Deceleration [0x08]** parameter (register 9).

The full frame needed for the request to write the values 00 96 hex (= 150 dec) and 00 64 hex (= 100 dec) in the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][10][00][07][00][02][04][00][96][00][64][53][8E]
where:
[01] = Slave address
[10] = **16 Write Multiple Registers** function code
[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)
[00][02] = number of requested registers
[04] = number of bytes (2 bytes for each register)
[00][96] = value to be set in the register 8 **Acceleration [0x07]**, 00 96 hex = 150 dec
[00][64] = value to be set in the register 9 **Deceleration [0x08]**, 00 64 hex = 100 dec
[53][8E] = CRC

The full frame needed to send back a response following the request to write the values 00 96 hex (= 150 dec) and 00 64 hex (= 100 dec) in the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][10][00][07][00][02][F0][09]
where:
[01] = Slave address
[10] = **16 Write Multiple Registers** function code
[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)
[00][02] = number of written registers
[F0][09] = CRC

NOTE
For further examples refer to the "Programming examples" section on page 90.

WARNING
For safety reasons, when the DRIVECOD unit 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. The 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 -a jog command for example- the Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered. To enable the Watch dog function, set to "=1" the **Watch dog enable** bit in the **Control Word [0x2A]** variable. If "=0" is set the Watch dog is not enabled; 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).

8 Programming parameters

8.1 Parameters available

Hereafter the parameters available for RD6 Modbus devices 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

The MODBUS registers are 16-bit long; while the actuator parameters can be 1-register long, i.e. 16-bit long, or 2-register long, i.e. 32-bit long.

Unsigned16 parameter structure:

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

Unsigned32 parameter structure:

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

8.1.1 Holding Register parameters

Holding registers (**Machine data** parameters) are accessible for both writing and reading; to read the value set in a parameter use the **03 Read Holding Registers** function code (reading of multiple registers); to write a value in a parameter use the **06 Write Single Register** function code (writing of a single register) or the **16 Write Multiple Registers** (writing of multiple registers); for any further information on the implemented function codes refer to the "7.4.1 Implemented function codes" section on page 56.

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 [0x2A] register, see on page 74.
Should the power supply be turned off all data that has not been saved previously will be lost!

WARNING
For safety reasons the following holding register parameters Extra commands register [0x29], Control Word [0x2A] and Target position [0x2B-0x2C] are not stored in the memory. So they are required to be set after each power-on.

Distance per revolution [0x00]

[Register 1, Unsigned16, rw]
This parameter sets the number of pulses per each complete revolution of the shaft. It is useful to relate the revolution of the shaft and a linear measurement. For example: the unit is joined to a warm screw having 5 mm (0.196") pitch; by setting Distance per revolution [0x00] = 500, at each shaft revolution the system performs a 5 mm (0.196") pitch with one-hundredth of a millimetre resolution.
Default = 4096 (min. = 1, max. = 4096)

WARNING
After having changed this parameter you must then set new values also next to the Preset [0x12-0x13] parameter. For a detailed explanation see on page 48 and relevant parameters.
Please note that the parameters listed hereafter are closely related to the Distance per revolution [0x00] parameter; hence when you change the value in Distance per revolution [0x00] also the value in each of them necessarily changes. They are: Position window [0x01], Max following error [0x03-0x04], Positive delta [0x09-0x0A], Negative delta [0x0B-0x0C], Target position [0x2B-0x2C], Current position [0x02-0x03] and Position following error [0x05-0x06]. See also on page 72.

NOTE
If Distance per revolution [0x00] is not a power of 2 (2, 4,..., 2048, 4096), at position control a positioning error could occur having a value equal to one pulse.

Position window [0x01]

[Register 2, Unsigned16, rw]
This parameter defines the tolerance window limits for the **Target position [0x2B-0x2C]** value. As soon as the axis is within the tolerance window limits, the bit 8 **Target position reached** in the **Status word [0x01]** goes high ("=1"). When the axis is within the tolerance window limits for the time set in the **Position window time [0x02]** parameter, the bit 0 **Axis in position** in the **Status word [0x01]** goes high ("=1"). The parameter is expressed in pulses. See also the "Positioning: position and speed control" section on page 47.
Default = 1 (min. = 0, max. = 65535)

Position window time [0x02]

[Register 3, Unsigned16, rw]
It represents the time for which the axis has to be within the tolerance window limits set in the **Position window [0x01]** parameter before the state is signalled through the **Axis in position** status bit of the **Status word [0x01]**. The parameter is expressed in milliseconds. See also the "Positioning: position and speed control" section on page 47.
Default = 0 (min. = 0, max. = 10000)

Max following error [0x03-0x04]

[Registers 4-5, Unsigned32, rw]
This parameter defines the maximum allowable difference between the real position and the theoretical position of the device. If the device detects a value higher than the one set in this parameter, the **Following error** alarm is triggered and the unit stops. The parameter is expressed in pulses.
Default = 50000 (min. = 0, max. = 1000000)

Kp position loop [0x05]

[Register 6, Unsigned16, rw]
This parameter contains the proportional gain used by the PI controller for the position loop. The value has been optimized by Lika Electronic according to the technical characteristics of the device.
Default = 80 (min. = 0, max. = 1000)

Ki position loop [0x06]

[Register 7, Unsigned16, rw]
This parameter contains the integral gain used by the PI controller for the position loop. The value has been optimized by Lika Electronic according to the technical characteristics of the device.
Default = 10 (min. = 0, max. = 1000)

Acceleration [0x07]

[Register 8, Unsigned16, rw]
This parameter defines the acceleration value that has to be used by the device when reaching both the **Jog speed [0x0D]** and the **Work speed [0x0E]**. The parameter is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 46.
Default = 10 (min. = 1, max. = 500)

Deceleration [0x08]

[Register 9, Unsigned16, rw]
This parameter defines the deceleration value that has to be used by the device when stopping. The parameter is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 46.
Default = 10 (min. = 1, max. = 500)

Positive delta [0x09-0x0A]

[Registers 10-11, Unsigned32, rw]
This value is used to calculate the maximum forward (positive) limit the device is allowed to reach starting from the preset value. Should it happen that the maximum forward limit is reached, a signalling is activated through the **SW limit switch +** status bit of the **Status word [0x01]** (the bit is forced high). The parameter is expressed in pulses.
SW limit switch + = **Preset [0x12-0x13]** + **Positive delta [0x09-0x0A]**.
For further information please refer to the "6.3 Distance per revolution [0x00], Jog speed [0x0D], Work speed [0x0E], Preset [0x12-0x13], Positive delta [0x09-0x0A] and Negative delta [0x0B-0x0C]" section on page 48.
Default = 134 213 631 (min. = 0, max. = 134 213 631)

WARNING
Please mind the maximum acceptable value for this item depends on the set scaling.

EXAMPLE
When **Distance per revolution [0x00]** = 4,096 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Positive delta [0x09-0x0A]** is:
(4,096 steps per revolution * 32,768 revolutions) - 1 step – 4,096 steps (i.e. 1 revolution for safety reasons) = 134 213 631 (see the default value)
When **Distance per revolution [0x00]** = 1,024 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Positive delta [0x09-0x0A]** is:
(1,024 steps per revolution * 32,768 revolutions) - 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 33 553 407

When **Distance per revolution [0x00]** = 256 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Positive delta [0x09-0x0A]** is:
(256 steps per revolution * 32,768 revolutions) - 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 8 388 351
See further examples in the "6.3 Distance per revolution [0x00], Jog speed [0x0D], Work speed [0x0E], Preset [0x12-0x13], Positive delta [0x09-0x0A] and Negative delta [0x0B-0x0C]" section on page 48.

WARNING
Every time **Distance per revolution [0x00]** and **Preset [0x12-0x13]** parameters are changed, **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** values has to be checked carefully. Each time you change the value in **Distance per revolution [0x00]** you must then update the value in **Preset [0x12-0x13]** in order to define the zero of the shaft as the system reference has now changed.
After having changed the parameter in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 48.

Negative delta [0x0B-0x0C]

[Registers 12-13, Unsigned32, rw]
This value is used to calculate the maximum backward (negative) limit the device is allowed to reach starting from the preset value. Should it happen that the maximum backward limit is reached, a signalling is activated through the **SW limit switch -** status bit of the **Status word [0x01]** (the bit is forced high).
The parameter is expressed in pulses.
SW limit switch - = **Preset [0x12-0x13]** - **Negative delta [0x0B-0x0C]**.
For further information please refer to the "6.3 Distance per revolution [0x00], Jog speed [0x0D], Work speed [0x0E], Preset [0x12-0x13], Positive delta [0x09-0x0A] and Negative delta [0x0B-0x0C]" section on page 48.
Default = 134 213 631 (min. = 0, max. = 134 213 631)

WARNING
Please mind the maximum acceptable value for this item depends on the set scaling.

EXAMPLE

When **Distance per revolution [0x00]** = 4,096 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Negative delta [0x0B-0x0C]** is:
(4,096 steps per revolution * 32,768 revolutions) – 1 step – 4,096 steps (i.e. 1 revolution for safety reasons) = 134 213 631 (see the default value)
When **Distance per revolution [0x00]** = 1,024 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Negative delta [0x0B-0x0C]** is:
(1,024 steps per revolution * 32,768 revolutions) – 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 33 553 407
When **Distance per revolution [0x00]** = 256 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Negative delta [0x0B-0x0C]** is:
(256 steps per revolution * 32,768 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 8 388 351
See further examples in the "6.3 Distance per revolution [0x00], Jog speed [0x0D], Work speed [0x0E], Preset [0x12-0x13], Positive delta [0x09-0x0A] and Negative delta [0x0B-0x0C]" section on page 48.

WARNING

Every time **Distance per revolution [0x00]** and **Preset [0x12-0x13]** parameters are changed, **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** values has to be checked carefully. Each time you change the value in **Distance per revolution [0x00]** you must then update the value in **Preset [0x12-0x13]** in order to define the zero of the shaft as the system reference has now changed.
After having changed the parameter in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 48.

Jog speed [0x0D]

[Register 14, Unsigned16, rw]
This parameter contains the maximum speed the device is allowed to reach when using the **Jog +** and **Jog -** functions (see the **Control Word [0x2A]** parameter). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 46.
Default = 2000 (min. = 1, max. = 3000)

Work speed [0x0E]

[Register 15, Unsigned16, rw]
This parameter contains the maximum speed the device is allowed to reach in automatic work mode (movements are controlled using the **Start** and **Stop** commands -see the **Control Word [0x2A]** parameter- and are performed in order to reach the position set in **Target position [0x2B-0x2C]**). The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 47.
Default = 2000 (min. = 1, max. = 3000)

Code sequence [0x0F]

[Register 16, Unsigned16, rw]
It sets whether the position value output by the device increases (count up information) when the shaft rotates clockwise (0) or counter-clockwise (1). Clockwise and counter-clockwise rotations are viewed from shaft side.
0 = count up information with clockwise rotation (default)
1 = count up information with counter-clockwise rotation

WARNING
Changing this value causes also the position calculated by the controller to be necessarily affected. Therefore it is compulsory to set a new value in the **Preset [0x12-0x13]** parameter and then check the values set next to the **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items.

Offset [0x10-0x11]

[Registers 17-18, Integer32, ro]
This variable defines the difference between the position value transmitted by the device and the real position: real position – preset. The value is expressed in pulses.
Default = 0

Preset [0x12-0x13]

[Registers 19-20, Integer32, rw]
Use this parameter to set the Preset value. The Preset function is meant to assign a desired value to a physical position of the axis. The chosen physical position will get the value set next to this item and all the previous and the following positions will get a value according to it. The preset value will be set for the position of the axis in the moment when the value is entered. The preset value is activated when the bit 11 **Setting the preset** in the **Control Word [0x2A]** register is switched from logic level low ("0") to logic level high ("1").

Default = 0 (min. = -268 435 456, max. = 268 435 456)

NOTE
We suggest activating the preset when the actuator is in stop. See the **Setting the preset** command on page 77.

WARNING
A new value has to be set in the **Preset [0x12-0x13]** registers every time the **Distance per revolution [0x00]** value is changed. After having entered a new value in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in the **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 48.

Jog step length [0x14]

[Register 21, Unsigned16, rw]
If the incremental jog function is enabled (bit 4 **Incremental jog** in the **Control Word [0x2A]** = 1), the activation of the bits **Jog +** and **Jog -** causes a single step toward the positive or negative direction having the length, expressed in pulses, set next to this item to be executed at rising edge; then the actuator stops and waits for another command.
Default = 1000 (min. = 1, max. = 10000).

Extra commands register [0x29]

[Register 42, Unsigned16, rw]
Byte structure of the **Extra commands register [0x29]**:

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

Byte 0

bit 0: Not used.

Control by PC

bit 1: This function is reserved only for use and service of Lika Electronic engineers (only used with Modbus service port).

bits 2 ... 7 Not used.

Byte 1 Not used.

WARNING
For safety reasons the **Extra commands register [0x29]** holding register parameter is not stored in the memory. So it is required to be set after each power-on.

Control Word [0x2A]

[Register 43, 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 [0x2A]** register:

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

Byte 0

Jog +
bit 0

If the bit 4 **Incremental jog** = 0, as long as **Jog +** = 1, the Slave moves toward the positive direction; otherwise if the bit 4 **Incremental jog** = 1, the activation of this bit causes a single step toward the positive direction having the length, expressed in pulses, set next to the **Jog step length [0x14]** item to be executed at rising edge; then the actuator stops and waits for another command. Velocity, acceleration and deceleration are performed according to the values set next to the **Jog speed [0x0D]**, **Acceleration [0x07]** and **Deceleration [0x08]** parameters respectively. For a detailed description of the jog control see on page 46.

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the Slave while it is moving to the target position, the jog command will be ignored; if **Jog +** and **Jog -** commands are sent simultaneously, the device will not move or, if already moving, it will stop its movement.

Jog -
bit 1

If the bit 4 **Incremental jog** = 0, as long as **Jog -** = 1, the Slave moves toward the negative direction; otherwise if the bit 4 **Incremental jog** = 1, the activation of this bit causes a

single step toward the negative direction having the length, expressed in pulses, set next to the **Jog step length [0x14]** item to be executed at rising edge; then the actuator stops and waits for another command. Velocity, acceleration and deceleration are performed according to the value set next to the **Jog speed [0x0D]**, **Acceleration [0x07]** and **Deceleration [0x08]** parameters respectively. For a detailed description of the jog control see on page 46.

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the Slave while it is moving to the target position, the jog command will be ignored; if **Jog +** and **Jog -** commands are sent simultaneously, the device will not move or, if already moving, it will stop its movement.

Stop

bit 2

If set to "=1" the Slave is allowed to execute the movements as commanded. If, while the unit is running, this bit switches to "=0" then the Slave must stop executing the deceleration procedure set in **Deceleration [0x08]**. For an immediate halt in the movement, use the bit 7 **Emergency**.

Alarm reset

bit 3

This command is used to reset an alarm condition of the Slave but only if the fault condition has ceased. In a normal work condition this bit is set to "0". Setting this bit to "1" causes the normal work status of the device to be restored. The normal work status is resumed by switching this bit from "0" to "1".

Please note that should the alarm be caused by wrong parameter values (see **Machine data not valid** and **Wrong parameters list [0x08-0x09]**), the normal work status can be restored only after having set proper values. The **Flash memory error** alarm cannot be reset.

Incremental jog

bit 4

If set to "=0", the activation of the bits **Jog +** and **Jog -** causes the Slave to move as long as **Jog +** / **Jog -** = 1. Setting this bit to 1 the incremental jog function is enabled, that is: the activation of the bits **Jog +** and **Jog -** causes a single step toward the positive or negative direction having the length, expressed in pulses, set next to the **Jog step length [0x14]** item to be executed at rising edge; then the actuator stops and waits for another command.

bit 5

Not used.

Start

bit 6

When it is set to "=1" the device moves in order to reach the set target position (see Target position [0x2B-0x2C] on page 78). For a complete description of the position control see on page 47.

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the Slave while it is moving to the target position, the jog command will be ignored; if **Jog +** and **Jog -** commands are sent simultaneously, the device will not move or, if already moving, it will stop its movement.

Emergency

bit 7

This bit has to be normally high ("=1") otherwise it will cause the device to stop immediately. For a normal stop (not immediate) respecting the set deceleration see above the bit 2 **Stop**. At power-on it is forced low (="0") for safety reasons. Switch it high ("=1") to resume normal operation.

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). The 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 -a jog command for example- the Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered.

Save parameters

bit 9

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

Load default parameters

bit 10

The 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 loading 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 96.

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!

WARNING

The unit has been adjusted by performing a full-load mechanical running test; thence default values which has been set refer to a device running in such condition. Furthermore they are intended to ensure a standard and safe operation which not necessarily results in a smooth running and an optimum performance. Thus to suit the specific application requirements it may be advisable and even necessary to enter new parameters instead of the factory default settings; in particular it may be necessary to change velocity, acceleration, deceleration and gain values.

Setting the preset

bit 11

It sets the current position to the value set next to the **Preset [0x12-0x13]** registers. 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"). We suggest activating the preset when the actuator is in stop. For more information refer to page 72.

Release axis torque

bit 12

When the axis has reached the commanded position, it maintains the torque.

If set to "=0", when the axis is in position, the PWM is kept active.

If set to "=1", when the axis is in position, the PWM is deactivated (the torque is released).

bits 13 ... 15

Not used.

WARNING
For safety reasons the **Control Word [0x2A]** holding register parameter is not stored in the memory. So it is required to be set after each power-on.

Target position [0x2B-0x2C]

[Registers 44-45, Integer32, rw]
It sets the position to be reached, otherwise referred to as commanded position. When the **Start** command is sent while the **Stop** and **Emergency** bits are "=1" and the alarm condition is off, the device moves in order to reach the target position set next to this item.
As soon as the axis is within the tolerance window limits set next to the **Position window [0x01]** register, the bit 8 **Target position reached** in the **Status word [0x01]** goes high ("=1"). When the position is within the tolerance window limits set next to the **Position window [0x01]** register, after the delay set next to the **Position window time [0x02]** item, the bit 0 **Axis in position** in the **Status word [0x01]** goes high ("=1").
For more information refer also to the "Positioning: position and speed control" section on page 47.
Default = 0 (min. = 0, max. = within maximum positive limit / maximum negative limit)

NOTE
Position override function
It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the **Target position [0x2B-0x2C]** registers. See also on page 47.

NOTE
Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the Slave while it is moving to the target position, the jog command will be ignored; if **Jog +** and **Jog -** commands are sent simultaneously, the device will not move or, if already moving, it will stop its movement.
When the Watch dog function is enabled (**Watch dog enable** in **Control Word [0x2A]** is set to "=1"), should the device be disconnected from the Modbus network while it is moving (for instance because of a broken cable or a faulty wiring), the device stops moving immediately and activates the **Watch dog** alarm bit (the alarm is invoked to appear as soon as the Modbus network communication is restored).

WARNING
For safety reasons the **Target position [0x2B-0x2C]** holding register parameter is not stored in the memory. So it is required to be set after each power-on.

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

8.1.2 Input Register parameters

The **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 "7.4.1 Implemented function codes" section on page 56.

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 the normal work condition. See the list of the wrong parameters in the **Wrong parameters list [0x08-0x09]** item.

Flash memory error

bit 1 — Internal error, it cannot be restored.

Counting error

bit 2 — For safety reasons, both the absolute encoder position and the incremental encoder position are read and saved to two separate registers. If any difference between the values in the registers is found the error is signalled.

Following error

bit 3 — The difference between the real position and the theoretical position is greater than the value set in the **Max following error [0x03-0x04]** parameter; we suggest reducing the work speed.

Axis not synchronized

bit 4 — Internal error, it cannot be restored.

Target not valid

bit 5 — The set target position is over the maximum travel limits.

Emergency

bit 6 — Bit 7 **Emergency** in **Control Word [0x2A]** has been forced to low value (0); or alarms are active in the unit.

Overcurrent

bit 7 — Motor overcurrent.

Byte 1

Electronics Overtemperature

bit 8 — The temperature of the MOSFETs detected by an internal probe is exceeding the maximum ratings (see **Temperature value [0x07]** on page 84). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the range.

Motor Overtemperature

bit 9 — The temperature of the motor detected by an internal probe is exceeding the maximum ratings (see **Temperature value [0x07]** on page 84). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the range.

Undervoltage

bit 10 — The power supply voltage is under the minimum ratings allowed. Please ensure that the power supply voltage is within the range.

Watch dog

bit 11 — When the Watch dog function is enabled (bit 8 **Watch dog enable** in **Control Word [0x2A]** 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. The 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 -a jog command for example- the Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered.

bits 12 and 13 — Not used.

Hall sequence

bit 14 — An error has been detected in the Hall sensors commutation sequence.

Overvoltage

bit 15 — The power supply voltage is over the maximum ratings allowed. Please ensure that the power supply voltage is within the range.
If the alarm is triggered during the braking operation, please consider the counter-electromotive force (back EMF). To prevent such situation from arising, decrease the deceleration ramp or evaluate attentively the characteristics of the 24V power supply pack (capacitor module).

To reset a faulty condition use the **Alarm reset** command, Control Word [0x2A] bit 3. In a normal work condition the **Alarm reset** bit is set to "0". Setting the bit to "1" causes the normal work status of the device to be restored. The normal work status is resumed by switching this bit from "0" to "1". This command resets the alarm but only if the fault condition has ceased.

Please note that should the alarm be caused by wrong parameter values (see Machine data not valid and Wrong parameters list [0x08-0x09]), the normal work status can be restored only after having set proper values. The Flash memory error alarm cannot be reset.

Status word [0x01]

[Register 2, Unsigned16, ro]
This register contains information about the current state of the device.
Byte structure of the Status word [0x01] register:

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

Byte 0

Axis in position

bit 0 — The value is "=1" when the device reaches and keeps the set position (Target position [0x2B-0x2C]) for the time set next to the Position window time [0x02] register. It is kept active until the position error is lower than Position window [0x01]. For

further information please refer to the "Positioning: position and speed control" section on page 47.

bit 1 — Not used.

Axis enabled

bit 2 — It shows the enabling status of the motor. This bit is "=1" when the motor is enabled, that is: PWM is active and the axis is under closed-loop control (while reaching a target position or using a jog, for instance). It is "=0" when the motor is disabled, that is when the controller is off after a positioning or jog movement or because of an alarm condition.

SW limit switch +

bit 3 — The value is "=1" should it happen that the device reaches the maximum positive limit (positive limit switch). For more information see the **Positive delta [0x09-0x0A]** parameter.

SW limit switch -

bit 4 — The value is "=1" should it happen that the device reaches the maximum negative limit (negative limit switch). For more information see the **Negative delta [0x0B-0x0C]** parameter.

Alarm

bit 5 — The value is "=1" when an alarm occurs, see details in the **Alarms register [0x00]** variable.

Axis running

bit 6 — The value is "=0" while the device is not moving.
The value is "=1" while the device is moving.

Executing a command

bit 7 — The value is "=0" when the controller is not executing any command.
The value is "=1" while the controller is executing a command.

Byte 1

Target position reached

bit 8 — The value is "=1" when the device reaches the target position set next to the **Target position [0x2B-0x2C]** item (it is within the limits set next to the **Position window [0x01]**). The bit is kept active until a new **Target position [0x2B-0x2C]** value or the **Alarm reset** command are sent. For more information

refer also to the "Positioning: position and speed control" section on page 47.

bits 9 ... 11 — Not used.

PWM saturation

bit 12 — The current supplied for controlling the motor phases has reached the saturation point and cannot be increased further. The motor operation is affected by excessive dynamics or something is jamming the movement.

bits 13 ... 15 — Not used.

Current position [0x02-0x03]

[Registers 3-4, Integer32, ro]
Current position of the device expressed in pulses.

Current velocity [0x04]

[Register 5, Integer16, ro]
Speed of the device expressed in revolutions per minute [rpm], updated at every second.

Position following error [0x05-0x06]

[Registers 6-7, Integer32, ro]
This variable contains the difference between the target position and the current position step by step. If this value is greater than the one set in the **Max following error [0x03-0x04]** parameter, then the **Following error** alarm is triggered and the unit stops. The value is expressed in pulses.

Temperature value [0x07]

[Register 8, Integer16, ro]
This variable shows both the temperature of the motor and the temperature of the electronics as detected by internal probes. The value is expressed in Celsius degrees (°C). The minimum detectable temperature is -20°C.
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
MSB								LSB							
Major number								Minor number							
Temperature of the motor								Temperature of the electronics							

Value 18 1A hex in hexadecimal notation corresponds to the binary representation 0001 1000 0001 1010 and has to be interpreted as: temperature of the motor = 24°C; temperature of the electronics = 26°C.

Wrong parameters list [0x08-0x09]

[Registers 9-10, Unsigned32, ro]

The operator has set invalid data and the **Machine data not valid** alarm has been triggered. This variable is meant to show the list of the wrong parameters, respecting the structure shown in 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	Distance per revolution [0x00]
2	Acceleration [0x07]
3	Deceleration [0x08]
4	Positive delta [0x09-0x0A]
5	Negative delta [0x0B-0x0C]
6	Jog speed [0x0D]
7	Work speed [0x0E]
8	Code sequence [0x0F]
9	Preset [0x12-0x13]
10	Jog step length [0x14]
11	Kp position loop [0x05]
12	Ki position loop [0x06]
13	Position window time [0x02]
14	Max following error [0x03-0x04]
15	Not used

Motor voltage [0x0A]

[Register 11, Unsigned16, ro]

It shows the motor voltage expressed in millivolts (mV).

Current value [0x0B]

[Register 12, Unsigned16, ro]
This variable shows the value of the current absorbed by the motor (rated current). The value is expressed in milliamperes (mA).

Hall [0x0C]

[Register 13, Unsigned16, ro]
This function is reserved only for use and service of Lika Electronic engineers.

Duty cycle [0x0D]

[Register 14, Unsigned16, ro]
This function is reserved only for use and service of Lika Electronic engineers.

DIP switch baud rate [0x0E]

[Register 15, Unsigned16, ro]
This is meant to show the data transmission rate (baud rate) of the serial port the RD6 unit is equipped with; the data transmission rate has to be set through the provided DIP switch. For any further information on setting the baud rate refer to the "4.4.2 Setting the Baud rate and Parity bit (Figure 7)" section on page 24.

DIP switch node ID [0x0F]

[Register 16, Unsigned16, ro]
This is meant to show the node address set in the RD6 unit; the node address has to be set through the provided rotary switches. For any further information on setting the node ID refer to the "4.4.1 Setting the node address: Node ID (Figure 7)" section on page 23.

SW Version [0x10]

[Register 17, Unsigned16, ro]
This is meant to show the software version of the DRIVECOD unit.
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
MSB								LSB							
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 [0x11]

[Register 18, Unsigned16, ro]
This is meant to show the hardware version and model of the DRIVECOD unit.
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
not used		Motor size		Interface				not used				Hardware version			

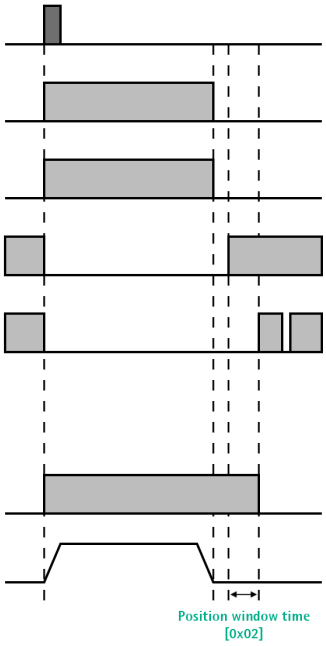
where:

00 ... 03	= hardware version
08 .. 11	= RD6 model equipped with the following interface: 0x00 = MODBUS RTU; 0x01 = Profibus; 0x02 = CANopen; 0x03 = POWERLINK; 0x04 = EtherCAT; 0x05 = MODBUS TCP; 0x06 = EtherNet/IP; 0x07 = Profinet
12	01h = 157 W motor size
13	01h = 250 W motor size

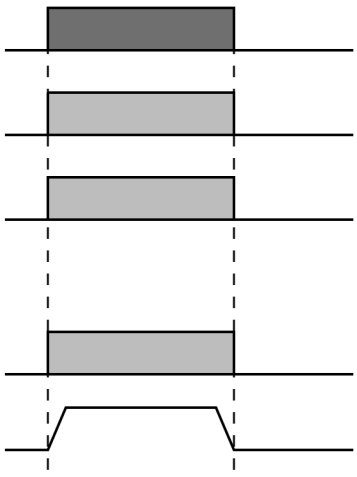
Value 10 01 hex in hexadecimal notation corresponds to the binary representation 0001 0000 0000 0001 and has to be interpreted as follows: hardware version 1 (LSbyte = 0x01); RD6 157 W model with MODBUS RTU interface (MSbyte = 0x10).

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

EXAMPLE 1

Positioning	RD6
Start (bit 6 - Control Word [0x2A])	
Executing a command (bit 7 - Status word [0x01])	
Axis running (bit 6 - Status word [0x01])	
Target position reached (bit 8 - Status word [0x01])	
Axis in position (bit 0 - Status word [0x01])	
Axis enabled (bit 2 - Status word [0x01]: PWM is active)	
Theoretical velocity	

EXAMPLE 2

Jog +	RD6
Jog + (bit 0 - Control Word [0x2A])	
Executing a command (bit 7 - Status word [0x01])	
Axis running (bit 6 - Status word [0x01])	
Axis enabled (bit 2 - Status word [0x01]: PWM is active)	
Theoretical velocity	

8.2 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 "MODBUS Exception Responses" section on page 48 of the "MODBUS Application Protocol Specification V1.1b" document.

9 Programming examples

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

9.1 Using the 03 Read Holding Registers function code

EXAMPLE 1
Request to read the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) to the Slave having the node address 1.

Request PDU
[01][03][00][07][00][02][75][CA]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)
[00][02] = number of requested registers
[75][CA] = CRC

Response PDU
[01][03][04][00][0A][00][0A][5A][36]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[00][0A] = value of register 8 **Acceleration [0x07]**, 00 0A hex = 10 dec
[00][0A] = value of register 9 **Deceleration [0x08]**, 00 0A hex = 10 dec
[5A][36] = CRC

Acceleration [0x07] parameter (register 8) contains the value 00 0A hex, i.e. 10 in decimal notation; **Deceleration [0x08]** parameter (register 9) contains the value 00 0A hex, i.e. 10 in decimal notation.

9.2 Using the 04 Read Input Register function code

EXAMPLE 1
Request to read the **Current position [0x02-0x03]** parameter (registers 3 and 4) to the Slave having the node address 1.

Request PDU
[01][04][00][02][00][02][D0][0B]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[00][02] = starting address (**Current position [0x02-0x03]** parameter, register 3)
[00][02] = number of requested registers
[D0][0B] = CRC

Response PDU
[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 3 **Current position [0x02-0x03]**, 00 00 hex = 0 dec
[2F][F0] = value of register 4 **Current position [0x02-0x03]**, 2F F0 hex = 12272 dec
[E7][F0] = CRC

Current position [0x02-0x03] parameter (registers 3 and 4) contains the value 00 00 2F F0 hex, i.e. 12272 in decimal notation.

EXAMPLE 2
Request to read the **Alarms register [0x00]** variable (register 1) to the Slave having the node address 1.

Request PDU
[01][04][00][00][00][01][31][CA]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[00][00] = starting address (**Alarms register [0x00]** variable, register 1)
[00][01] = number of requested registers

[31][CA] = CRC

Response PDU
[01][04][02][00][81][79][50]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[02] = number of bytes (2 bytes for each register)
[00][81] = value of register 1 **Alarms register [0x00]**, 00 81 hex = 0000 0000 1000 0001 bin
[79][50] = CRC

This means that in the **Alarms register [0x00]** variable (register 1) the bits 0 and 7 are active (logic level high = 1), i.e. (see on page 80): **Machine data not valid** and **Emergency**.

9.3 Using the 06 Write Single Register function code

EXAMPLE 1
Request to write the value 00 96 hex (= 150 dec) in the **Acceleration [0x07]** parameter (register 8) of the Slave having the node address 1.

Request PDU
[01][06][00][07][00][96][B8][65]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[00][96] = value to be set in the register
[B8][65] = CRC

Response PDU
[01][06][00][07][00][96][B8][65]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[00][96] = value set in the register
[B8][65] = CRC

The value 00 96 hex, i.e. 150 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8).

EXAMPLE 2
Request to write the value 00 84 hex in the **Control Word [0x2A]** variable (register 43) of the Slave having the node address 1.

Request PDU
[01][06][00][2A][00][84][A8][61]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[00][84] = value to be set in the register
[A8][61] = CRC

Response PDU
[01][06][00][2A][00][84][A8][61]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[00][84] = value set in the register
[A8][61] = CRC

The value 00 84 hex = 0000 0000 1000 0100 in binary notation is set in the **Control Word [0x2A]** variable (register 43). In other words, the **Stop** and **Emergency** bits are forced to the logical level high (bit 2 = 1; bit 7 = 1): the unit is ready to execute the motion command as requested.

EXAMPLE 3
Request to write the value 0A 80 hex in the **Control Word [0x2A]** variable (register 43) of the Slave having the node address 1.

Request PDU
[01][06][00][2A][0A][80][AF][02]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[0A][80] = value to be set in the register
[AF][02] = CRC

Response PDU
[01][06][00][2A][0A][80][AF][02]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[0A][80] = value set in the register
[AF][02] = CRC

The value 0A 80 hex = 0000 0010 1000 0000 in binary notation is set in the **Control Word [0x2A]** variable (register 43). In other words, the device is forced in stop (bit 2 **Stop** = 0) but not in emergency condition (bit 7 **Emergency** = 1); furthermore data save is requested (bit 9 **Save parameters** = 1).

9.4 Using the 16 Write Multiple Registers function code

EXAMPLE 1

Request to write the values 150 and 100 in the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) of the Slave having the node address 1.

Request PDU

[01][10][00][07][00][02][04][00][96][00][64][53][8E]
where:
[01] = Slave address
[10] = **16 Write Multiple Registers** function code
[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)
[00][02] = number of requested registers
[04] = number of bytes (2 bytes for each register)
[00][96] = value to be set in the register 8 **Acceleration [0x07]**, 00 96 hex = 150 dec
[00][64] = value to be set in the register 9 **Deceleration [0x08]**, 00 64 hex = 100 dec
[53][8E] = CRC

Response PDU

[01][10][00][07][00][02][F0][09]
where:
[01] = Slave address
[10] = **16 Write Multiple Registers** function code
[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)
[00][02] = number of written registers
[F0][09] = CRC

The value 00 96 hex, i.e. 150 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8); the value 00 64 hex, i.e. 100 in decimal notation, is set in the **Deceleration [0x08]** parameter (register 9).

10 Default parameters list

Parameters list	Default value		
Distance per revolution [0x00] PPR	4,096		
Position window [0x01] P	1		
Position window time [0x02] ms	0		
Max following error [0x03-0x04] P	50,000		
Kp position loop [0x05]	80		
Ki position loop [0x06]	10		
Acceleration [0x07] rev/s^2	10		
Deceleration [0x08] rev/s^2	10		
Positive delta [0x09-0x0A] P	134 213 631		
Negative delta [0x0B-0x0C] P	134 213 631		
Jog speed [0x0D] rpm	2,000		
Work speed [0x0E] rpm	2,000		
Code sequence [0x0F]	0		
Preset [0x12-0x13] P	0		
Jog step length [0x14] P	1,000		
Control Word [0x2A]	0		
Target position [0x2B-0x2C]	0		

This page intentionally left blank

This page intentionally left blank

This page intentionally left blank

Document release	Release date	Description	HW	SW	Interface
1.0	10.06.2016	First issue	1	1.0	1.0
1.1	25.07.2016	Connectors, new shaft dimensions, Modbus interface release 1.1	1	1.0	1.1
1.2	07.02.2019	General update, "Glossary terms" section added, Modbus examples, MODBUS interface updated	2	2.0	3.1.0.3

Dispose separately

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

Tel. +39 0445 806600
Fax +39 0445 806699

info@lika.biz • www.lika.biz