User's guide

RD5

RD53

Modbus

RS-485 version

Smart encoders & actuators

This publication was produced by Lika Electronic s.r.l. 2015. All rights reserved. Tutti i diritti riservati. Alle Rechte vorbehalten. Todos los derechos reservados. Tous droits réservés.

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

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

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

General contents

User's guide 1
General contents 3
Subject Index 6
Typographic and iconographic conventions 7
Preliminary information 8
1 Safety summary 9
1.1 Safety 9
1.2 Electrical safety 9
1.3 Mechanical safety 10
2 Identification 11
3 Mounting instructions 12
4 Electrical connections 15
4.1 Ground connection (Figure 2) 15
4.2 M12 connectors (Figure 2) 16
4.3 Dip-Switches and diagnostic LEDs (Figure 3) 18
4.3.1 Diagnostic LEDs (Figure 3) 19
4.3.2 Setting the address node: Node ID (Figure 3) 21
4.3.3 Setting data transmission rate: Baud rate and Parity bit (Figure 3) 22
4.3.4 RT bus termination (Figure 3) 23
5 Quick reference 24
5.1 Configuring the device using Lika setting up software 24
5.2 "Serial configuration" page 25
5.3 "Operative mode" page 29
5.4 "Machine data" page 33
5.5 "Message monitor" page 35
5.6 "Test Lika" page 36
5.7 "Upgrade Firmware" page 36
5.7.1 If there is an installation issue 39
5.8 Getting started 40
6 Functions 41
6.1 Working principle 41
6.2 Movements: jog and positioning 42
6.3 Distance per revolution [00 hex], Jog speed [0C hex], Work speed [0D hex], Preset [16-17 hex], Positive delta [08-09 hex] and Negative delta [0A-0B hex] 43
7 Modbus® interface 47
7.1 Modbus Master / Slaves protocol principle 47
7.2 Modbus frame description 48
7.3 Transmission modes 49
7.3.1 RTU transmission mode 50
7.4 Function codes 52
7.4.1 Implemented function codes 52
03 Read Holding Registers 52
04 Read Input Register 54
06 Write Single Register 56
16 Write Multiple Registers 58
8 Programming parameters 62
8.1 Parameters available 62

8.1.1 Machine data parameters....62
Distance per revolution [00 hex]....62
Position window [01 hex]....63
Position window time [02 hex]....63
Max following error [03 hex]....63
Kp position loop [04 hex]....64
Ki position loop [05 hex]....64
Acceleration [06 hex]....64
Deceleration [07 hex]....64
Positive delta [08-09 hex]....64
Negative delta [0A-0B hex]....65
Jog speed [0C hex]....65
Work speed [0D hex]....65
Start torque current time [0E hex]....66
Code sequence [0F hex]....66
Kp current loop [10 hex]....67
Ki current loop [11 hex]....67
Max current [12 hex]....67
Starting torque current [13 hex]....67
Offset [14-15 hex]....67
Preset [16-17 hex]....68
Gear ratio [18 hex]....68
Jog step length [19 hex]....68
Extra commands register [29 hex]....69
Absolute reading....69
Control Word [2A hex]....69
Jog +....69
Jog -....70
Stop....70
Alarm reset....70
Incremental jog....70
Start....71
Emergency....71
Watch dog enable....71
Save parameters....71
Load default parameters....71
Perform counting preset....72
Axis torque....72
Brake released....72
Target position [2B-2C hex]....72
8.1.2 Input Register parameters....74
Alarms register [00 hex]....74
Machine data not valid....74
Flash memory error....74
Following error....74
Axis not synchronized....74
Target not valid....74
Emergency....74
Overcurrent....75
Overtemperature....75
Undervoltage....75

Watch dog 75
Status word [01 hex] 76
Axis in position 76
Axis enabled 76
SW limit switch + ... 76
SW limit switch - 76
Alarm 76
Axis running 76
Executing a command 77
Target position reached 77
DAC saturation 77
Current position [02-03 hex] 77
Current velocity [04 hex] 77
Position following error [05-06 hex] 77
Current value [07 hex] 78
Temperature value [08 hex] 78
Wrong parameters list [09-0A hex] 78
I2t [0B hex] 79
Dip-switch baud rate [0C hex] 79
Dip-switch node ID [0D hex] 79
SW Version [0E hex] 79
HW Version [0F hex] 80
8.2 Exception codes 82
9 Programming examples 83
9.1 Using the 03 Read Holding Registers function code 83
9.2 Using the 04 Read Input Register function code 84
9.3 Using the 06 Write Single Register function code 86
9.4 Using the 16 Write Multiple Registers function code 88
10 Default parameters list 89

Subject Index

A
Absolute reading....69
Acceleration [06 hex]....64
Alarm....76
Alarm reset....70
Alarms register [00 hex]....74
Axis enabled....76
Axis in position....76
Axis not synchronized....74
Axis running....76
Axis torque....72
B
Brake released....72
C
Code sequence [0F hex]....66
Control Word [2A hex]....69
Current position [02-03 hex]....77
Current value [07 hex]....78
Current velocity [04 hex]....77
D
DAC saturation....77
Deceleration [07 hex]....64
Dip-switch baud rate [0C hex]....79
Dip-switch node ID [0D hex]....79
Distance per revolution [00 hex]....62
E
Emergency....71, 74
Executing a command....77
Extra commands register [29 hex]....69
F
Flash memory error....74
Following error....74
G
Gear ratio [18 hex]....68
H
HW Version [0F hex]....80
I
I2t [0B hex]....79
Incremental jog....70
J
Jog -....70
Jog +....69
Jog speed [0C hex]....65
Jog step length [19 hex]....68
K
Ki current loop [11 hex]....67
Ki position loop [05 hex]....64
Kp current loop [10 hex]....67
Kp position loop [04 hex]....64
L
Load default parameters....71
M
Machine data not valid....74
Max current [12 hex]....67
Max following error [03 hex]....63
N
Negative delta [0A-0B hex]....65
O
Offset [14-15 hex]....67
Overcurrent....75
Overtemperature....75
P
Perform counting preset....72
Position following error [05-06 hex]....77
Position window [01 hex]....63
Position window time [02 hex]....63
Positive delta [08-09 hex]....64
Preset [16-17 hex]....68
S
Save parameters....71
Start....71
Start torque current time [0E hex]....66
Starting torque current [13 hex]....67
Status word [01 hex]....76
Stop....70
SW limit switch -....76
SW limit switch +....76
SW Version [0E hex]....79
T
Target not valid....74
Target position [2B-2C hex]....72
Target position reached....77
Temperature value [08 hex]....78
U
Undervoltage....75
W
Watch dog....75
Watch dog enable....71
Work speed [0D hex]....65
Wrong parameters list [09-0A hex]....78

Typographic and iconographic conventions

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

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

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

	This icon, followed by the word **WARNING**, is meant to highlight the parts of the text where information of great significance for the user can be found: user must pay the greatest attention to them! Instructions must be followed strictly in order to guarantee the safety of the user and a correct use of the device. Failure to heed a warning or comply with instructions could lead to personal injury and/or damage to the unit or other equipment.
	This icon, followed by the word **NOTE**, is meant to highlight the parts of the text where important notes needful for a correct and reliable use of the device can be found. User must pay attention to them! Failure to comply with instructions could cause the equipment to be set wrongly: hence a faulty and improper working of the device could be the consequence.
	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 **ROTADRIVE positioning units RD5 and RD53 models**.
RD5 and RD53 units are positioning devices which integrate into one system a brushless motor fitted with gearbox, a drive, a multiturn absolute encoder and a position controller. They can be used in a variety of applications in any industrial sector and are suitable to drive secondary axes such as in mold changers, mobile stops, tools changers, suction cups motion units, conveyor and spindle positioning devices on packaging & woodworking machineries, among others.

An integrated brake differentiates RD53 model from RD5 model. The brake is designed to activate as soon as the motor comes to a stop in order to prevent it from moving even slightly.

The available interfaces for fieldbus communication are: **Modbus RTU**, **Profibus-DP** and **CANopen DS 301**.
In the Modbus version the configuration of the ROTADRIVE 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.

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.

1 Safety summary

1.1 Safety

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

1.2 Electrical safety

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

- minimize noise by connecting the shield and/or the connector housing and/or the frame to ground. Make sure that ground is not affected by noise. The connection point to ground can be situated both on the device side and on user's side. The best solution to minimize the interference must be carried out by the user.

1.3 Mechanical safety

- Install the device following strictly the information in the section "Mounting instructions";
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the unit;
- do not tool the unit or its shaft;
- delicate electronic equipment: handle with care; do not subject the device and the shaft to knocks or shocks;
- respect the environmental characteristics of the product;
- unit with hollow shaft: ROTADRIVE can be mounted directly onto a shaft whose diameter has to respect the technical characteristics specified in the purchase order and clamped by means of the collar and the hole into which an anti-rotation pin has to be inserted.

WARNING
The unit has been adjusted by performing a no-load mechanical running test; thence default values which has been set refer to an idle device, i.e. running disengaged from the load. Furthermore they are intended to ensure a standard and safe operation which not necessarily results in smooth running and 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.

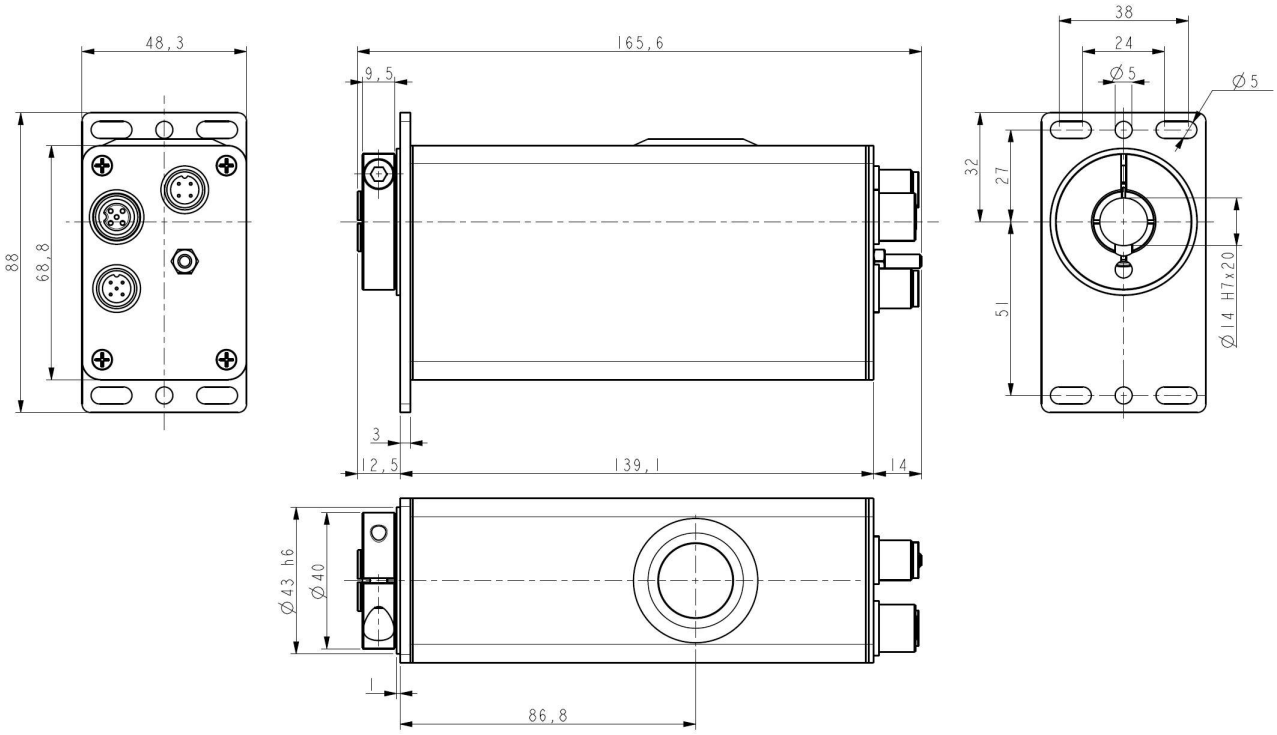
WARNING
The counter-electromotive force (back EMF) generated by the motor in case the shaft is forced to rotate due to a manual external force can cause irreparable damages to the internal circuitry.

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 s.r.l. for purchasing spare parts or needing assistance. For any information on the technical characteristics of the product refer to the technical catalogue.

3 Mounting instructions

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.



ROTADRIVE unit must be secured firmly only to the user's shaft using the provided collar. ROTADRIVE unit is supplied with a screw insulation and an anti-rotation pin; the anti-rotation pin (TE M4 screw) has to be inserted into the screw insulation. This will provide to the unit both stability and the mobility needed to absorb the mechanical tensions produced during operation. Do not fasten firmly the anti-rotation pin to the motor flange or the fixed support on user's side without using the screw insulation! If this occurs, the mechanical tensions would be transmitted completely to the motor shaft and this would lead to bearings damages and mechanical breakdowns!

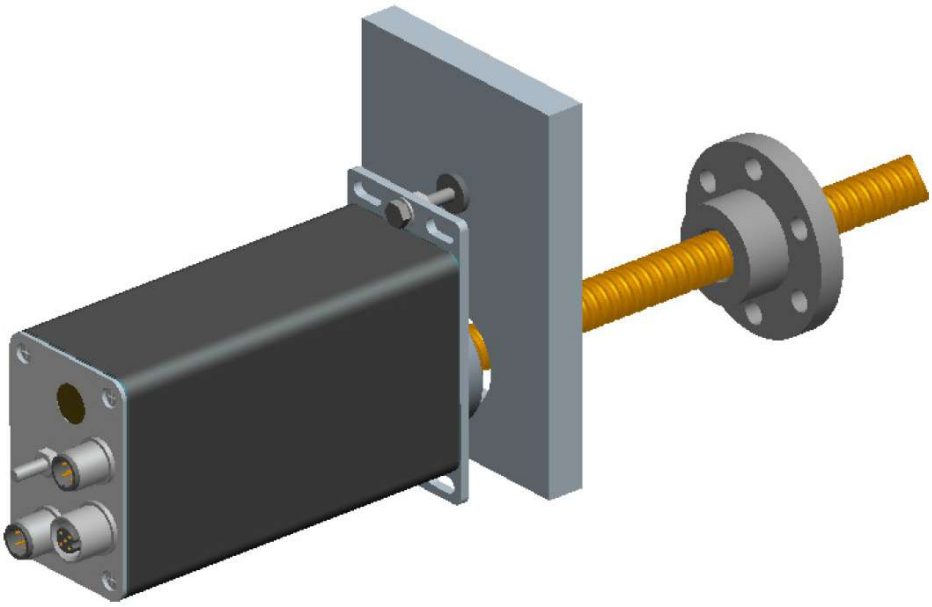
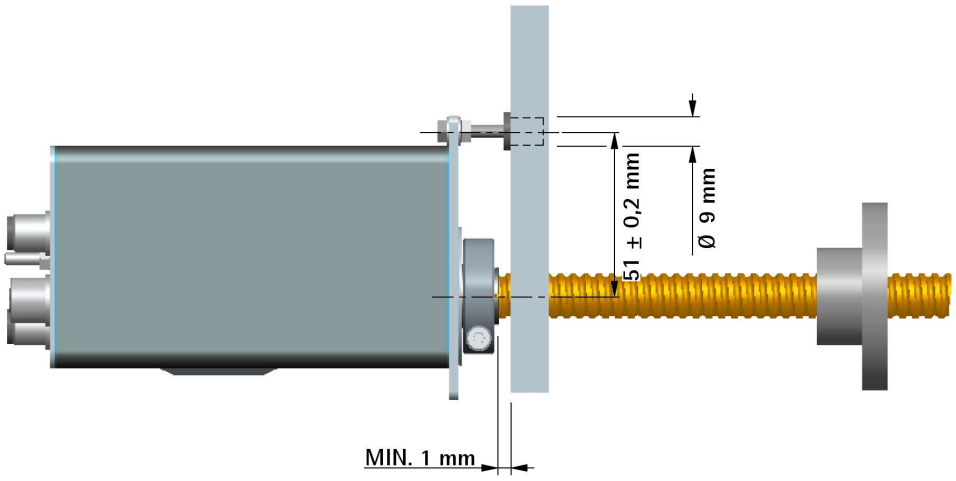


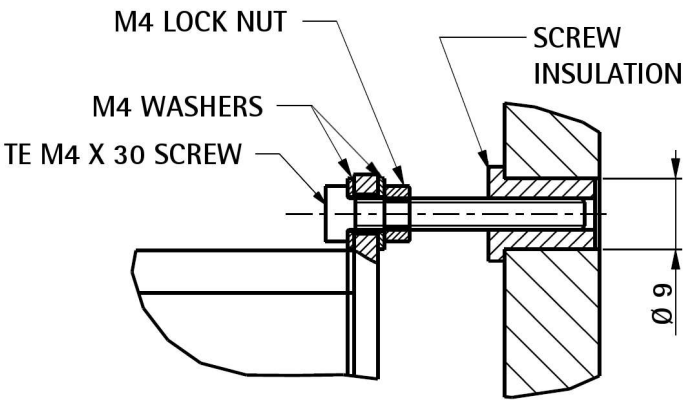
Figure 1 - Typical installation example of RD5-53 unit on worm screw

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

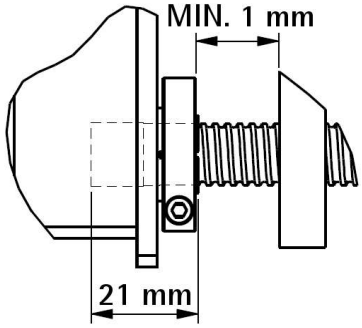
- Drill a 9 mm diameter hole in the motor flange or in the fixed support on user's side in order to insert the screw insulation and the anti-rotation pin. The distance between the axis of the shaft and the axis of the hole must be 51 ± 0,2 mm.



Make sure that the hole and the shaft are perfectly aligned on the vertical axis. If installation is not carried out properly, mechanical tensions would be produced on the motor shaft and this would lead to bearings damages and mechanical breakdowns!



- insert the screw insulation in the hole;
- insert the TE M4 x 30 screw and the two M4 washers in the middle hole designed in the flange of the ROTADRIVE unit; partially screw the M4 lock nut;
- insert the user's shaft in the hollow shaft of the ROTADRIVE unit; the maximum depth of the ROTADRIVE shaft is 21 mm; ascertain that the anti-rotation pin is inserted properly in the screw insulation; secure the user's shaft through the collar and the relevant fixing screw; the minimum distance between the collar and the fixed support on user's side must be not less than 1 mm in order to prevent the fixed parts from coming into contact;



- tighten the anti-rotation pin on the screw insulation;
- tighten the M4 lock nut in order to secure the anti-rotation pin to the flange of the ROTADRIVE unit.

ATTENTION

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 rotate due to a manual external force can cause irreparable damages to the internal circuitry.

4 Electrical connections

WARNING

When you send **Start**, **Jog +** or **Jog -** commands, the unit and the shaft start moving! Make sure 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!

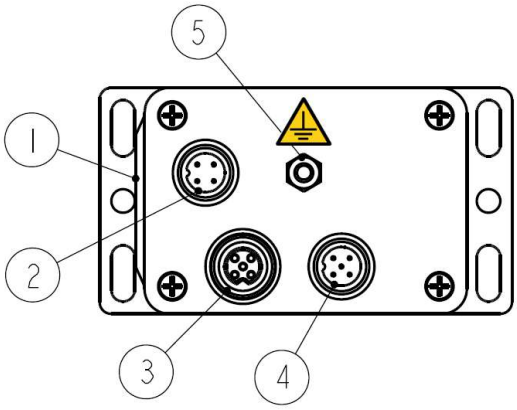


Figure 2: Electrical connections

Legend

1	Internal housing for dip-switches and diagnostic LEDs
2	M12 4-pin male connector for power supply
3	M12 5-pin female connector for BUS OUT
4	M12 5-pin male connector for BUS IN
5	Connection to earthing

4.1 Ground connection (Figure 2)

To minimize noise connect properly the frame to ground; we suggest using the ground screw provided in the frame (see 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 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.

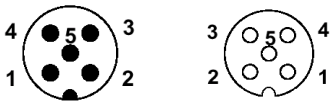
4.2 M12 connectors (Figure 2)

Power supply
M12 4-pin male connector
A coding
(frontal side)

Pin	Description
1	+24VDC ±10% Motor power supply
2	+24VDC ±10% Controller power supply
3	0VDC Power supply
4	n.c.

n.c. = not connected

Modbus interface
M12 5-pin connector
A coding
(frontal side)



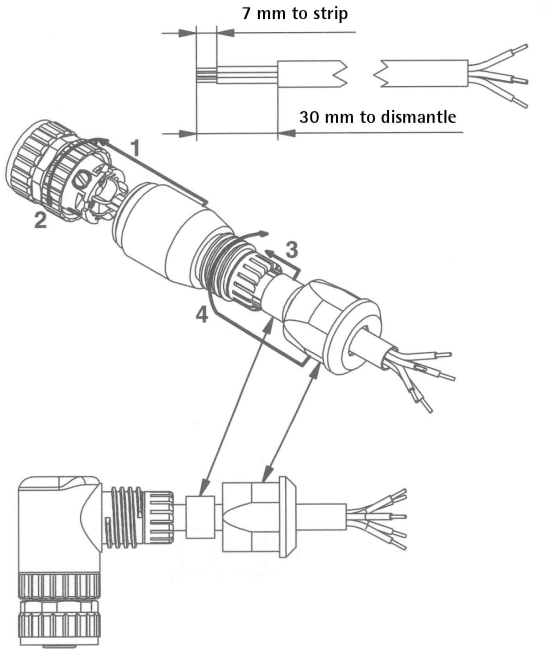
male (BUS IN) female (BUS OUT)

Pin	Description
1	n.c.
2	n.c.
3	GND (RS-485)
4	Modbus A (RS-485)
5	Modbus B (RS-485)
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 Figure below). If metal connectors are used, connect the cable shield properly as recommended by the manufacturer.

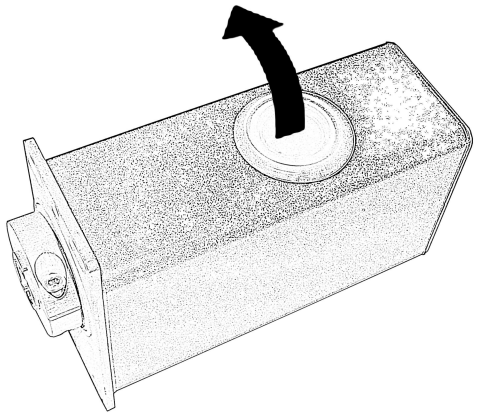


4.3 Dip-Switches and diagnostic LEDs (Figure 3)

WARNING
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 DIP-Switches and the diagnostic LEDs remove the plastic cover fitted on one side of the device.



The DIP-switches and the diagnostic LEDs are located just beneath.

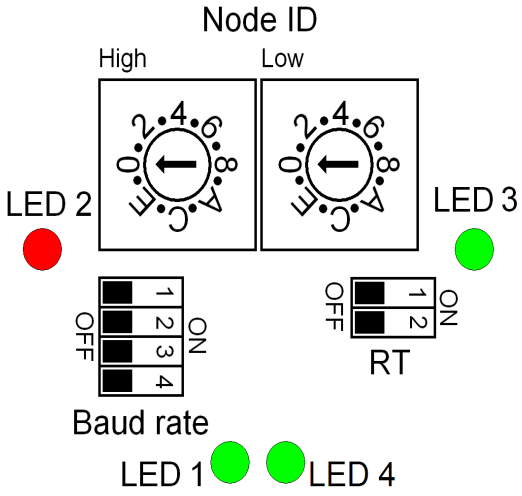


Figure 3: Dip-Switches and diagnostic LEDs

4.3.1 Diagnostic LEDs (Figure 3)

Four LEDs located in the inside of the device next to the provided switches and accessible by removing the plastic cover fitted on one side of the housing (see the previous page) are designed to show visually the operating or fault status of the Modbus interface and the unit as well. The meaning of each led is explained in the following table:

LED 1 GREEN	Description
ON	Indicates the power supply of the controller is turned on
OFF	Indicates the power supply of the controller is turned off

LED 2 RED	Description
ON	Active alarms, internal error, for a comprehensive list of alarms see on page 74
OFF	No alarms active

LED 3 GREEN	Description
Blinking led	Device is sending or receiving a message
OFF	No send - receive activity

LED 2	LED 3	Description
GREEN Blinking led	GREEN Blinking led	While downloading data to the flash memory for upgrading the firmware of the unit (see section "5.7 "Upgrade Firmware" page" on page 36), both LEDs 2 & 3 blink green at 5 Hz.
RED Blinking led	RED Blinking led	While downloading data to the flash memory for upgrading the firmware of the unit (see section "5.7 "Upgrade Firmware" page" on page 36), if an error occurs which stops the upgrading process (for instance: a voltage drop and/or the switching off of the ROTADRIVE unit), as soon as the power is turned on again both LEDs start blinking red at 5 Hz as the user program is not installed in the flash memory (it has been deleted previously). For any information on restoring the unit please refer to the section "5.7 "Upgrade Firmware" page" on page 36.
RED ON	RED ON	While downloading data to the flash memory for upgrading the firmware of the unit (see section "5.7 "Upgrade Firmware" page" on page 36), if data transmission is cut off (for instance, because of the disconnection of the serial cable), after 5 seconds both LEDs come on solidly red. For any information on restoring the unit please refer to the section "5.7 "Upgrade Firmware" page" on page 36.

LED 4 GREEN	Description
ON	Indicates the motor is enabled (control loop activated)
OFF	Indicates the motor is disabled (control loop deactivated)

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

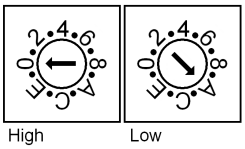
4.3.2 Setting the address node: Node ID (Figure 3)

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

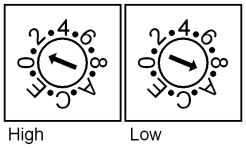
Set the node address expressed in hexadecimal notation.
The range of node addresses is between 1 and 247 (247 = F7 hex).

Example

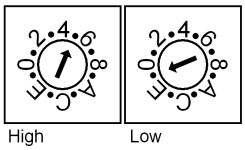
Address 10 = 0A hex:



Address 25 = 19 hex:



Address 95 = 5F hex:



NOTE
The default address is 1.
The address 0 is reserved to identify a "broadcast" exchange (Master sends a request to all Slaves connected to the Modbus network). See section "7.1 Modbus Master / Slaves protocol principle" on page 47.
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, device will be set it to 247 automatically.

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

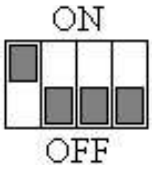
WARNING
Power supply must be turned off before performing this operation!

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

Switch	Baud rate	Parity bit
0000	9600 bit/s	No parity
1000 (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

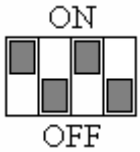
Example
Set the baud rate to 9600 bits per second and Even parity bit:

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



Set the baud rate to 19200 bits per second and Odd parity bit:

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



4.3.4 RT bus termination (Figure 3)

A bus termination resistor is provided and has to be activated as line termination in the last device of the transmission line.
Use RT Switch to activate or deactivate the bus termination.

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

5 Quick reference

5.1 Configuring the device using Lika setting up software

RD5 / RD53 Modbus ROTADRIVE 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. 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 **SW_RDX_MODBUS.EXE** and can be downloaded from **www.lika.biz > PRODUCTS > ROTARY ACTUATORS > POSITIONING UNITS**. 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 **DISCONNECT** button in the **Serial Configuration** page and then click the **CLOSE** button in the title bar.

WARNING

Please be aware that the following compatibilities between the HW-SW version of the actuator and the software release of the Modbus executable file have to be respected compulsorily.

Compatibility	HW-SW	EXE Modbus
	1-1.0	from V2.10 to ...

NOTE

Before starting the program, connect the device to the personal computer through a serial port. The serial interface of the ROTADRIVE 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.

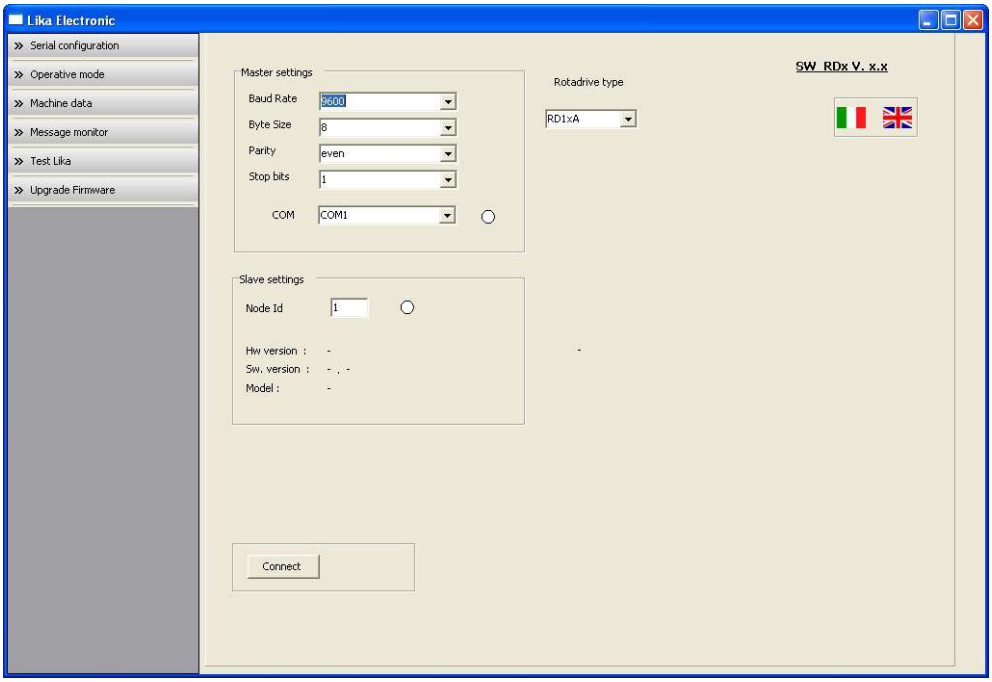
On the ROTADRIVE side the cable must be connected to the M12 5-pin male connector (BUS IN).

A cable assembly fitted with M12 5-pin / USB connectors and integrated RS-485 converter is available on request; please contact Lika Electronic s.r.l. Technical

Assistance & After Sale Service and quote the following code: **EXC-USB4-S54-GN-2-M12MC-S54**.

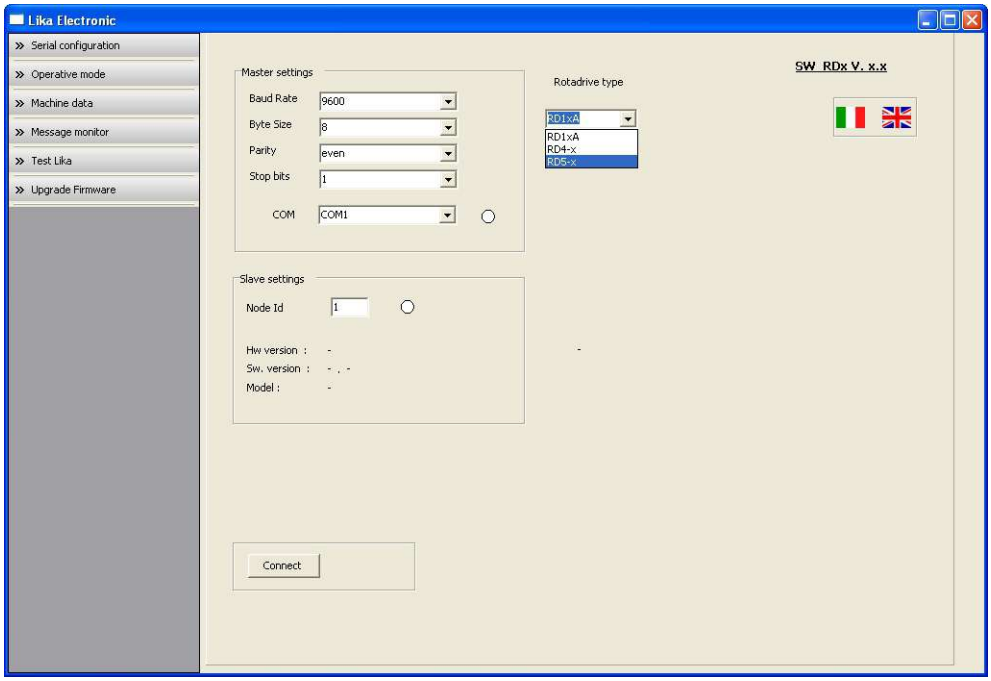
5.2 "Serial configuration" page

When you start the program, the **Serial configuration** page is first displayed.

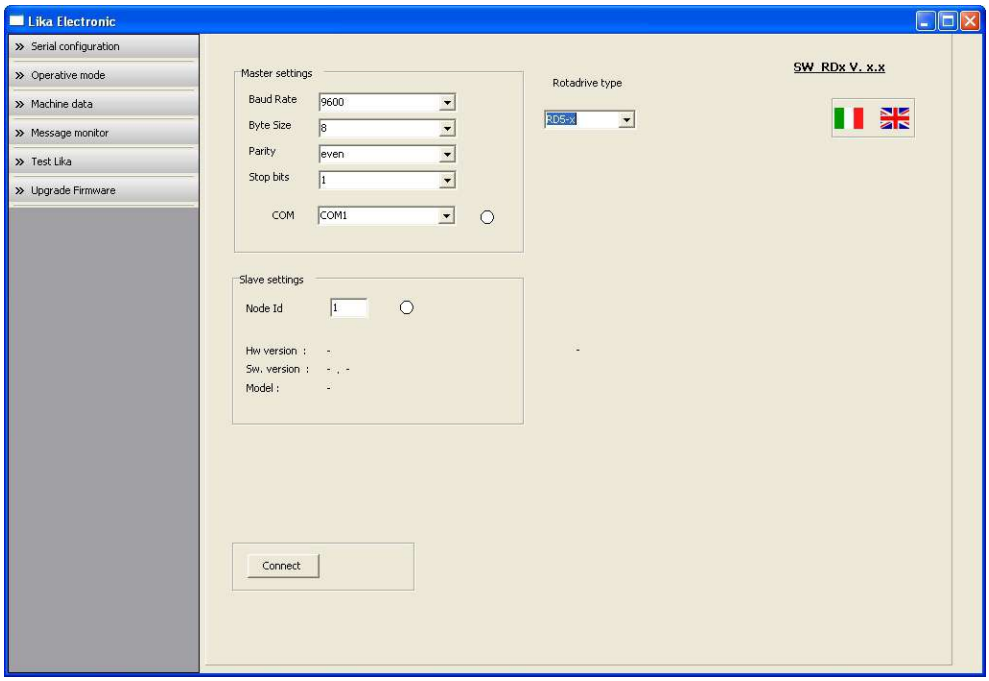


First of all this page allows the operator to choose the language used to display texts and items in the user interface. Click the **Italian flag** icon to choose the Italian language; click the **UK flag** icon to choose the English language.

Before entering the next pages pressing the buttons in the menu on the left, it is necessary to establish a serial connection with the RD5x unit. To do so, open the **Rotadrive type** drop-down box and select the **RD5-x** model.



On the left side of the drop-down box the **Master settings** box will activate; it allows you to choose the serial port of the personal computer the RD5x unit is connected to (**COM** 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.



Default serial port settings as set at the factory by Lika Electronic for all RD5x-Modbus positioning units are the following:

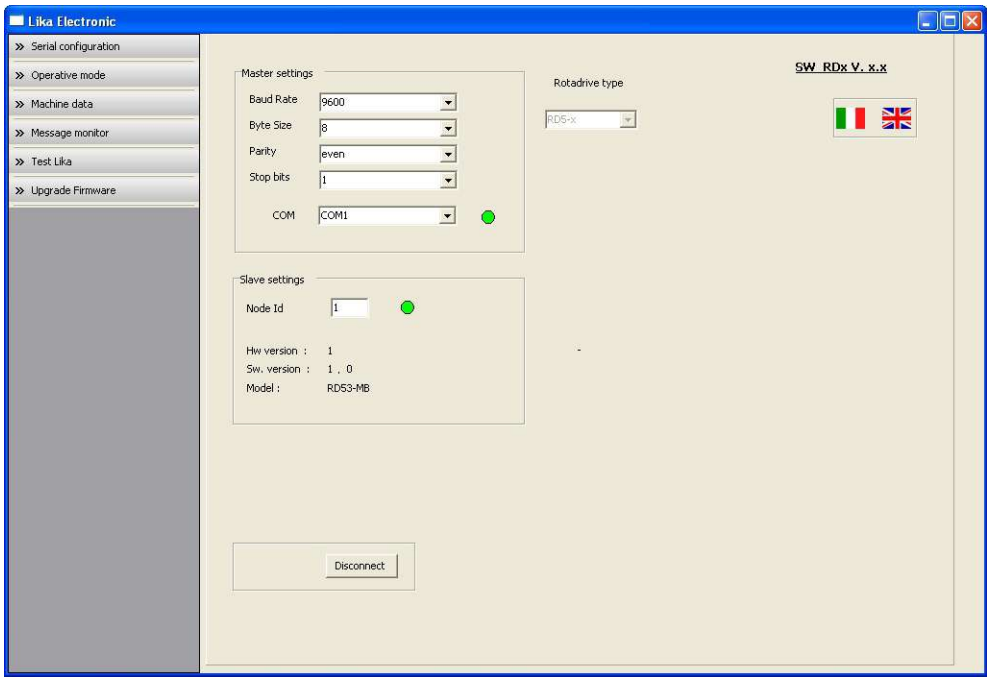
RD5x-Modbus **RS-485 serial port settings**	**Default value**
Baud rate	9600
Byte size	8
Parity	Even
Stop bits	1

To configure the serial port of the RD5x device refer to section "4.3.3 Setting data transmission rate: Baud rate and Parity bit (Figure 3)" on page 22.

Then set the node address of the device the personal computer is connected to through the **Slave settings** box (default value for all RD5x-Modbus positioning

units = 1). To set the node address of the RD5x device refer to section "4.3.2 Setting the address node: Node ID (Figure 3)" on page 21.

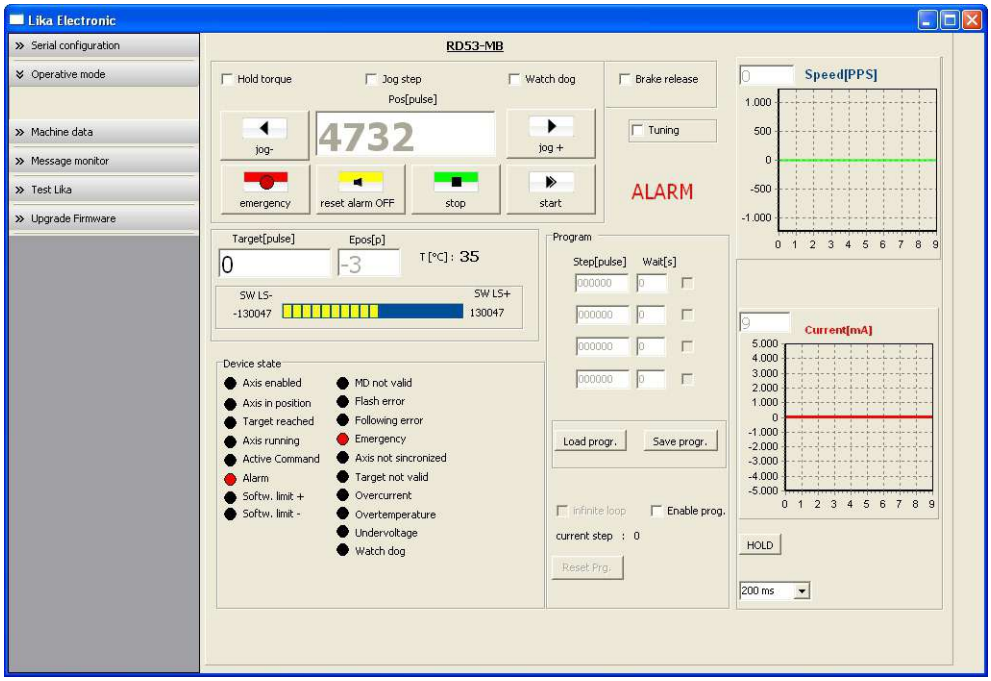
Now you are ready to establish the connection to the Slave: press the **CONNECT** button below in the page.



If the connection is established properly, two green lights placed next to the fields used to choose the **serial port** and set the **node ID** come on, while the **CONNECT** button disappears and is replaced by the **DISCONNECT** button. Furthermore the hardware version and the software version as well as the model of the device are shown in the **Slave settings** box.

5.3 "Operative mode" page

Press the **OPERATIVE MODE** button in the menu on the left side to start programming, controlling manually and monitoring the device. The page below will appear.



When you first enter the **Operative mode** page, RD5x unit is necessarily in an emergency condition: therefore the **Emergency** button is highlighted in red and the ALARM warning message blinks on the right; while the **Alarm** and **Emergency** warnings in the bottom left-hand **Device state** box are lit red; furthermore the red LED 2 in the RD5x unit is solidly lit. To restore the **Idle** state of the device, first press the **EMERGENCY** button and then press the **RESET ALARM** button in this page. Alarm warnings are reset while the green LED 3 in the RD5x unit starts blinking.

In the top left-hand **RD5x-MB** box the following functions are available.

Hold torque

See Axis torque item on page 72. This function is available only to RD5 version; in RD53 version this check-box is hidden.

Jog Step
See Incremental jog item on page 70.

Watch dog
See Watch dog enable item on page 71.

Jog -
See Jog - item on page 70.

Pos [pulse]
See Current position [02-03 hex] item on page 77.

Jog +
See Jog + item on page 69.

Emergency
When an emergency condition occurs, the **Emergency** button is highlighted in red; press the button to restore the normal work condition of the device. When the unit is running, press the button to force an immediate halt in emergency condition. See Emergency item on page 71.

Reset alarm
If an alarm is active, the **RESET ALARM** button is highlighted in yellow; press the button to reset the alarm. See Alarm reset item on page 70.

Stop
Press this button to force a normal halt of the device, respecting the acceleration and deceleration values. See Stop item on page 70.

Start
Pressing the button causes the unit to start running in order to reach the position set next to the **Target [pulse]** item; the MOVING warning message blinks on the right. As soon as the commanded position is reached, the device stops and activates the Axis in position and Target position reached status bits. For a normal halt of the device press the **STOP** button; for an immediate emergency halt press the **EMERGENCY** button. See Start item on page 71.

In the box just below the **RD5x-MB** box the following functions are available.

Target [pulse]
See Target position [2B-2C hex] item on page 72. 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 [pulse]** item, then it stops and activates the Axis in position and Target position reached status bits.

E pos [pulse]
See Position following error [05-06 hex] item on page 77.

T [°C]
See Temperature value [08 hex] item on page 78.

SW LS - / SW LS +
It shows visually the set positive and negative limit switch values. See Positive delta [08-09 hex] item on page 64 and Negative delta [0A-0B hex] item on page 65.

In the bottom left-hand **Device state** box the list of states and alarms available for the RD5x unit is displayed. Active states are highlighted in green; while active alarms are highlighted in red. For a detailed description of the states see Status word [01 hex] item on page 76; for a detailed description of the alarms see Alarms register [00 hex] item on page 74.

Functions available in the **Program** box allow the operator to create and then save a work program for the RD5x unit. Functions in the **Program** box are disabled by default; to enable them select the **ENABLE PROG.** check box.
Positions the device is commanded to reach (target positions) must be set next to the **STEP [pulse]** items; it is possible to enter up to four subsequent positions. Next to the **WAIT [s]** items you must set the gap between one step (commanded movement) and the next. All values that you set must then be confirmed by pressing the **ENTER** key in the keyboard. Before entering a value, each field must be previously enabled by selecting the check box on the right.

INFINITE LOOP check box below allows the operator to activate the "infinite loop" function, i.e. the device goes on running and executing the set steps without interruption.

If the **INFINITE LOOP** check box is selected, when you press the **START** button, the device starts moving in order to reach the first commanded position; **STEP [pulse]** and **WAIT [s]** items are highlighted in yellow; as soon as the commanded position set next to the **STEP [pulse]** item is reached, the device stops and the field is highlighted in green, as soon as the set gap has expired (a backward counter is displayed) also the **WAIT [s]** field is highlighted in green and the RD5x unit restarts running in order to reach the second commanded position; and so on, from the first to the fourth commanded position (if enabled) and then again from the first to the fourth commanded position without interruption, until you press the **STOP** button.
If the **INFINITE LOOP** check box is not selected, when you press the **START** button, the device starts running in order to reach the first commanded position; as soon as the commanded position is reached, the device stops and waits for the set gap to expire; you must then press the **START** button again to command the unit to reach the second position; and so on.

It is possible to save a work program you have created. To do so press the **SAVE PROGR.** button. Once you press the button the **Save as** dialogue box appears on the screen: the operator must type the .prg file name and specify the path where the file has to be located. When you press the **SAVE** button to confirm, the dialogue box closes. Set values are saved automatically.

To load a previously saved work program, press the **LOAD PROGR.** button. Once you press the button the **Open** dialogue box appears on the screen: the operator must open the folder where the previously saved .prg file is located, then select it and finally confirm the choice by pressing the **OPEN** button, the dialogue box closes and the work values are automatically loaded.

RESET PRG. button zero-sets the counter meant to detect the steps in 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, whatever the position reached previously.

To disable the execution of a work program deselect the **ENABLE PROG.** check box.

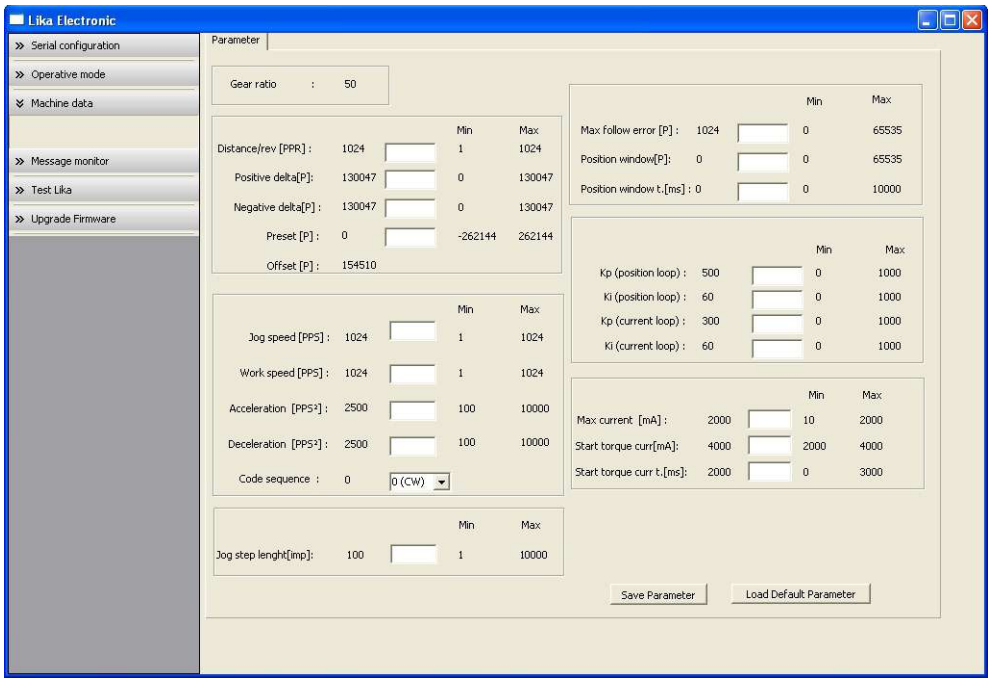
On the right side of the page the speed of the device (expressed in PPS) and the value of the current absorbed by the motor (expressed in milliamperes) are shown visually through charts. Press the **HOLD** button to disable charts visualization; press the same button (now labelled **GO**) to enable it again. The drop-down box below allows to choose the time scale in the horizontal axis of the graph.

On the right of the **RD5x-MB** box in the top of the screen, the **TUNING** check box is available. Once you select the check box, the speed and absorbed current charts on the right of the page disappear and four sliders replace them. The sliders are used to set the proportional and integral gain values concerning both the position loop and the current loop. For any further information refer to the explanation of each variable in this guide (see section "8.1.1 Machine data parameters" on page 62).

When you connect to a ROTADRIVE unit RD53 model, the **BRAKE RELEASE** check box appears just above the **TUNING** check box. Unlike RD5 model, RD53 model is fitted with a brake designed to activate as soon as the motor comes to a stop in order to prevent it from moving even slightly. When you select this check box, the brake is temporarily deactivated so, for instance, it is possible to move manually the shaft of the ROTADRIVE unit. See Brake released on page 72.

5.4 "Machine data" page

By pressing the **MACHINE DATA** button in the menu on the left side the operator enters the **Machine data** page.



In this page the list of the parameters available to set the RD5x-Modbus positioning units (machine data) is displayed. On the left of each field the values currently loaded in the unit are shown; while on the right the minimum and maximum values allowed are shown. For detailed information on the function and the setting of each parameter refer to the section "8.1.1 Machine data parameters" on page 62.

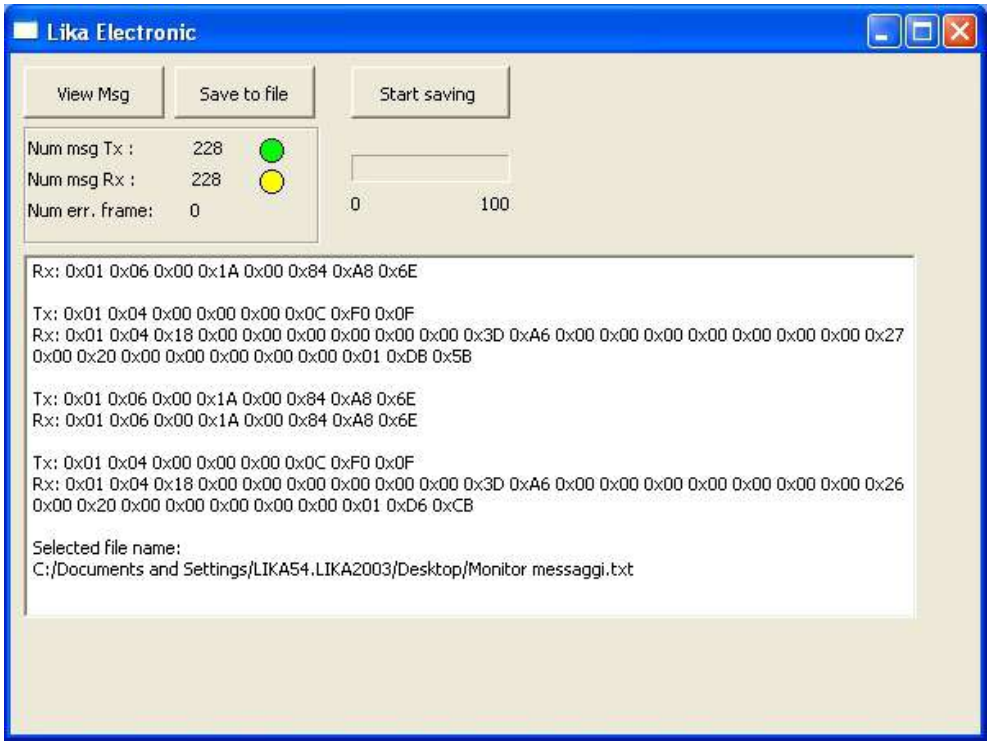
To enter a new value type it in the blank field and then press the **ENTER** key in the keyboard. If you set a value not allowed (out of range), at confirmation prompt the field is highlighted in red and the RD5x unit is forced in alarm condition (the **Alarm** status bit is activated and the **Machine data not valid** and/or **Emergency** error messages are invoked to appear). Enter a valid value and then press the **RESET ALARM** button in the **Operative mode** page to restore the normal work condition of the device.

To save the entered values on the non-volatile memory of the device press the **SAVE PARAMETER** button. If the power is turned off all data not saved will be lost! For any further information on saving the parameters refer to the **Save parameters** variable on page 71.

When you need to load 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 **LOAD DEFAULT PARAMETER** button. For any further information on loading the default parameters refer to the **Load default parameters** variable on page 71. On page 89 the complete list of the machine data parameters and the relevant default values as set by Lika Electronic are available.

5.5 "Message monitor" page

By pressing the **MESSAGE MONITOR** button in the menu on the left side the operator enters the **Message monitor** page.



This page allows the operator to monitor the communication between the Master and the Slave, by displaying the Request PDU (Tx) and the Response PDU (Rx) messages. When you first enter the page, the field meant to show the messages is blank. The box located just below the buttons shows the number of transmitted and received messages: Num msg TX = Request PDUs; Num msg Rx = Response PDUs; Num. Err. Frame = Exception Response PDUs.
Press the **VIEW MSG** button to display the flow of messages. Once you press the button, data throughput rate between the Master and the Slave starts appearing on the screen. Messages are displayed in hexadecimal notation. After pressing the **VIEW MSG** button, its descriptive label is replaced by **HOLD MSG** label. Press the **HOLD MSG** button to stop the flow of messages.

You can save the messages to a text file. As soon as you press the **SAVE TO FILE** button the **Open the log file** dialogue box appears on the screen: the operator must type the .txt file name and specify the path where the file has to be located. When you press the **OPEN** button to confirm, the dialogue box closes and the full path of the selected file is shown in the display box of the **Message**

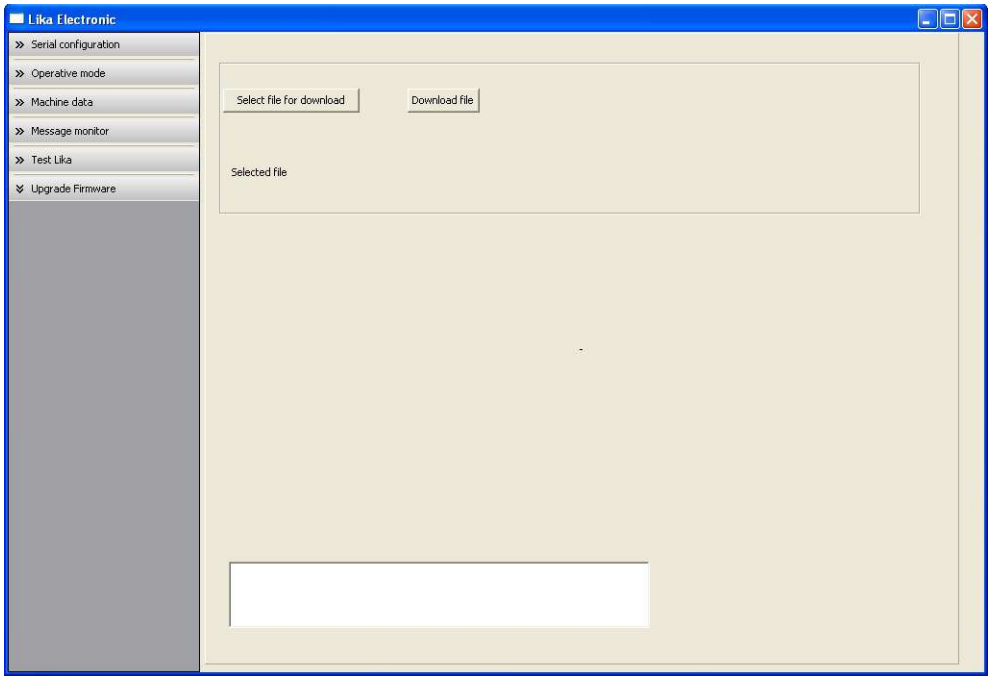
monitor page. Now press the **START SAVING** button to start saving the messages; the "File opened properly" message appears on the display box. After pressing the **START SAVING** button, its descriptive label is replaced by **STOP SAVING** label. Press the **STOP SAVING** button to stop saving the messages.

5.6 "Test Lika" page

Test Lika page is reserved for use by Lika Electronic engineers and is not accessible to users.

5.7 "Upgrade Firmware" page

By pressing the **UPGRADE FIRMWARE** button in the menu on the left side the operator enters the **Upgrade Firmware** page.



Functions available in this page allow the operator to upgrade the ROTADRIVE unit firmware by downloading upgrading data to the flash memory.

Firmware is a software program which controls the functions and operation of a device; the firmware program, sometimes referred to as "user program", is stored in the flash memory integrated inside the ROTADRIVE unit. ROTADRIVE 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 fix bugs, 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
Firmware upgrading process in any ROTADRIVE 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 ROTADRIVE unit from working.
If the latest firmware version is already installed in the ROTADRIVE 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 **Slave settings** box of the **Serial configuration** page after connecting properly to the unit (see on page 28).

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 the ROTADRIVE unit you need to update is the only node connected to the personal computer;
2. make sure to launch the 2.10 or latest release of the SW_RDX_MODBUS_X.EXE software for ROTADRIVE configuration via serial port in Modbus protocol;
3. connect to the unit, go online and then enter the **Upgrade Firmware** page;
4. when you switch on the power supply, if the LEDs 2 and 3 blink red at 5 Hz (the 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 **Upgrade Firmware** page; make sure the correct serial port of the personal computer connected to the ROTADRIVE unit is selected in the **Serial configuration**

page; for any further information please refer to the section "5.7.1 If there is an installation issue";

5. press the **SELECT FILE FOR DOWNLOAD** 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 located;

WARNING

Please note that for each ROTADRIVE model having its own bus interface an appropriate firmware file is available. Make sure you have the appropriate update for your ROTADRIVE model. The .BIN file released by Lika Electronic has a file name that has to be interpreted as follows.
For instance: RD1xA_MB_H1S2.0.BIN, where:

- RD1xA = ROTADRIVE unit model;
- MB = bus interface of the ROTADRIVE unit (MB = Modbus; I2 = CANopen; I3 = Profibus);
- H1 = hardware version;
- S2.0 = firmware version.

6. select the .BIN file and confirm the choice by pressing the **OPEN** button, the dialogue box closes;
7. the complete path for the file just confirmed appears next to the **SELECTED FILE** item;
8. now press the **DOWNLOAD FILE** button to start the firmware upgrading process;
9. a download progress bar is displayed in the centre of the page;
10. LEDs 2 and 3 blink green at 5 Hz during downloading operation;

WARNING

Do not exit the **Upgrade Firmware** page during installation, the process will be aborted!

11. as soon as the operation is carried out successfully, the **UPGRADE INSTALLATION COMPLETED SUCCESSFULLY** message is displayed;
12. the ROTADRIVE unit is now in an emergency condition;
13. close and then restart the SW_RDX_MODBUS_X.EXE program; connect to the ROTADRIVE unit and restore the normal work condition through the **Operative mode** page.

5.7.1 If there is an installation issue

While downloading the firmware upgrading program, unexpected conditions may arise which could lead to a failure of the installation process. When such a matter occurs, download process cannot be carried out successfully and thus the operation is aborted; LEDs 2 and 3 come on red (see on page 19), as explained hereafter.

LEDS 2 AND 3 BLINKING RED AT 5 Hz

While downloading data to the flash memory for upgrading the firmware of the unit, if an error occurs which stops the upgrading process (for instance: a voltage drop and/or the switching off of the ROTADRIVE unit), the COMMUNICATION ERROR. UPGRADE INSTALLATION ABORTED warning message is invoked to appear in the box in the bottom of the **Upgrade Firmware** page. The upgrading process is necessarily aborted and the unit cannot work as the firmware has been deleted before starting the update. As soon as the power is turned on again both LEDs 2 and 3 start blinking red at 5 Hz as the user program is not installed in the flash memory. To restore the work condition of the unit, the operator must close and then restart the SW_RDX_MODBUS_X.EXE program. It is not possible to connect to the unit through the **Serial configuration** page; the operator must enter the **Upgrade Firmware** page and restart the firmware installation process from point 5. Always make sure the correct serial port of the personal computer connected to the ROTADRIVE unit is selected in the **Serial configuration** page.

LEDS 2 AND 3 SOLIDLY LIT RED

While downloading data to the flash memory for upgrading the firmware of the unit, if data transmission is cut off (for instance, because of the disconnection of the serial cable), after 5 seconds both LEDs 2 and 3 come on solidly red. The COMMUNICATION ERROR. UPGRADE INSTALLATION ABORTED warning message is invoked to appear in the box in the bottom of the **Upgrade Firmware** page. The upgrading process is necessarily aborted and the unit cannot work as the firmware has been deleted before starting the update. To restore the work condition of the unit, the operator must shut down and then switch on the ROTADRIVE unit first, then close and restart the SW_RDX_MODBUS_X.EXE program. As soon as the power is turned on again both LEDs 2 and 3 start blinking red at 5 Hz as the user program is not installed in the flash memory. It is not possible to connect to the unit through the **Serial configuration** page; the operator must enter the **Upgrade Firmware** page and restart the firmware installation process from point 5. Always make sure the correct serial port of the personal computer connected to the ROTADRIVE unit is selected in the **Serial configuration** page.

5.8 Getting started

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

- Mechanically install the device;
- execute electrical connections;
- if requested, set data transmission rate (baud rate; see on page 22); the default value set by Lika Electronic at factory set-up is "baud rate = 9600 bit/s, parity = Even";
- if requested, set the node address (node ID; see on page 21); the default value set by Lika Electronic at factory set-up is "1";
- switch +24VDC power supply on (in both motor and controller);
- set a proper value next to the **Distance per revolution [00 hex]** item (register 1; see on page 62);
- set a proper value next to the **Jog speed [0C hex]** item (register 13; see on page 65);
- set a proper value next to the **Work speed [0D hex]** item (register 14; see on page 65);
- set a proper value next to the **Preset [16-17 hex]** item (registers 23-24; see on page 68);
- set the limit switch values next to the **Positive delta [08-09 hex]** and **Negative delta [0A-0B hex]** items; see on page 64);
- save new setting values (**Save parameters** register; see on page 71).

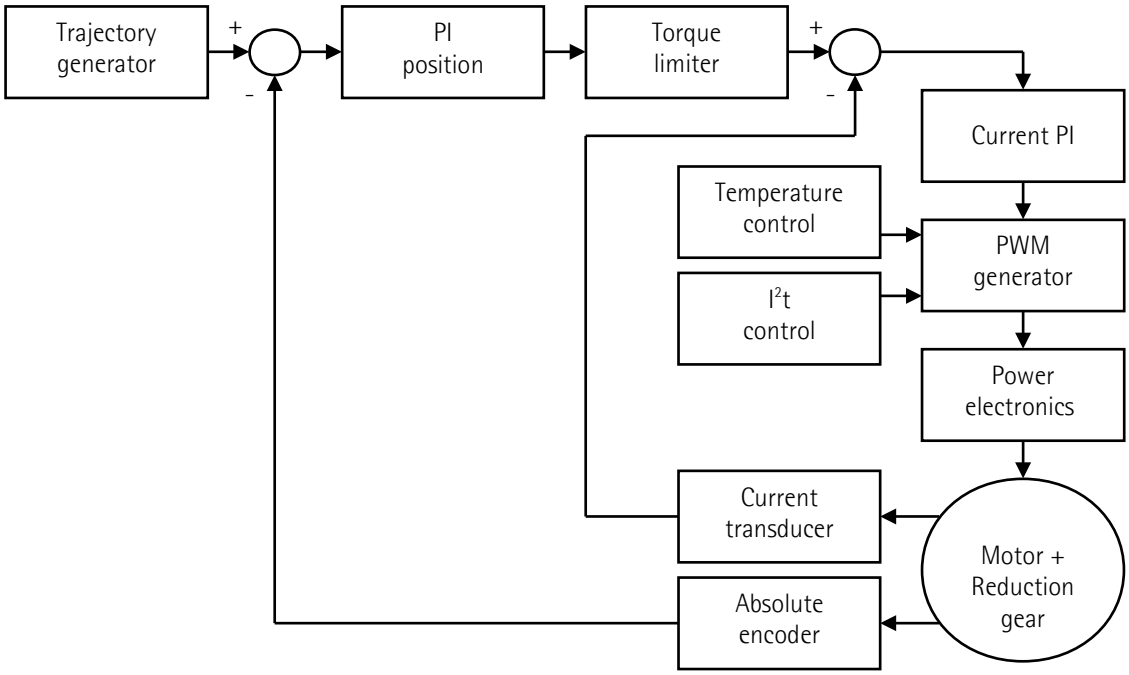
NOTE

Parameters **Distance per revolution [00 hex]**, **Jog speed [0C hex]**, **Work speed [0D hex]**, **Preset [16-17 hex]**, **Positive delta [08-09 hex]** and **Negative delta [0A-0B hex]** are closely related, hence you have to be very attentive when you need to change the value in one of them. For any further information please refer to page 43.

6 Functions

6.1 Working principle

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



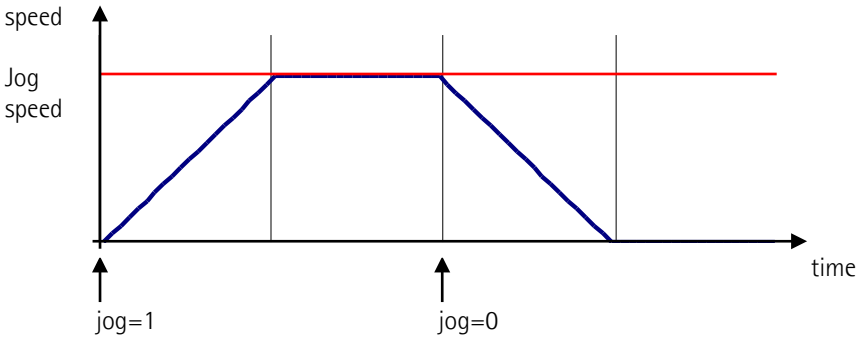
6.2 Movements: jog and positioning

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

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

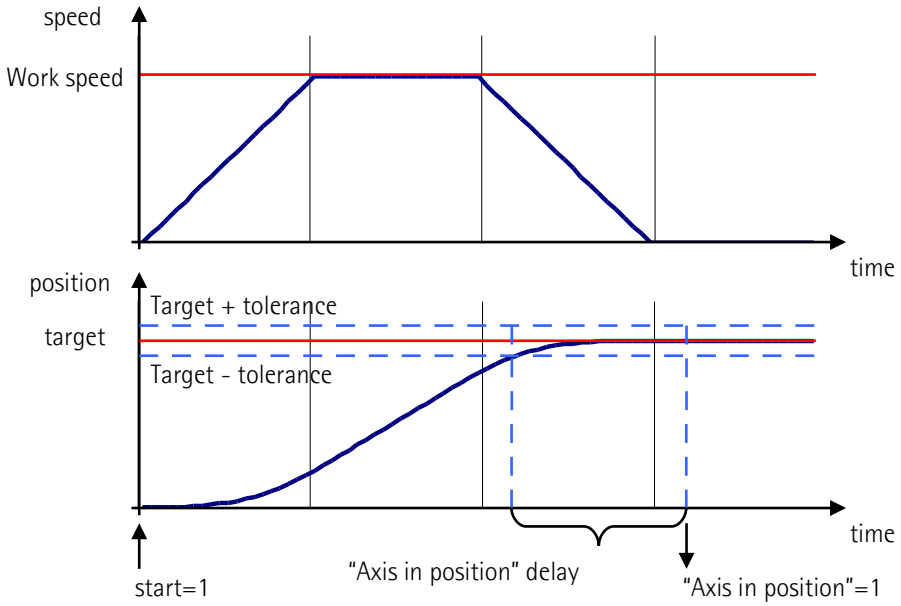
Jog: speed control

This kind of control is intended to generate a speed trajectory which is able to make the maximum rotation speed of the ROTADRIVE unit shaft to be equal to the value set in **Jog speed [0C hex]** (see on page 65).



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 **Work speed [0D hex]** (see on page 65); set speed can be reached only if space is long enough.



6.3 Distance per revolution [00 hex], Jog speed [0C hex], Work speed [0D hex], Preset [16-17 hex], Positive delta [08-09 hex] and Negative delta [0A-0B hex]

Variables Distance per revolution [00 hex], Jog speed [0C hex], Work speed [0D hex], Preset [16-17 hex], Positive delta [08-09 hex] and Negative delta [0A-0B hex] are closely related, hence you have to be very attentive every time you need to change the value in any of them.
Should it be necessary, you have to operate in compliance with the following procedure:

- set a proper value next to the Distance per revolution [00 hex] item (register 1, see on page 62);
- set a proper value next to the Jog speed [0C hex] item (register 13, see on page 65);
- set a proper value next to the Work speed [0D hex] item (register 14, see on page 65);
- set a proper value next to the Preset [16-17 hex] item (registers 23-24, see on page 68);
- check the value next to the Positive delta [08-09 hex] item (registers 9-10, see on page 64);
- check the value next to the Negative delta [0A-0B hex] item (registers 11-12, see on page 65);
- save new values (Save parameters item, bit 9 in the Control Word [2A hex] register, see on page 71).

WARNING
Each time you change the value in Distance per revolution [00 hex] you must then set new values also in Jog speed [0C hex] and Work speed [0D hex] as speed values are expressed in pulses per second (PPS). To calculate the speed values you have always to adhere to the following ratio:

$$\frac{\text{Min. Speed} * \text{Distance per revolution [00 hex]}}{1024} \leq \text{Speed} \leq \frac{\text{Max. Speed} * \text{Distance per revolution [00 hex]}}{1024}$$

where:

- Distance per revolution [00 hex]: this is the new value you want to set next to the Distance per revolution [00 hex] item, expressed in pulses
- Min. speed: minimum speed 1 [PPS]
- Max. speed: maximum speed 1024 [PPS]
- **1024**: this is the maximum value you can set next to the Distance per revolution [00 hex] item (expressed in pulses).

Each time you change the value in **Distance per revolution [00 hex]** then you must set a new value also in **Preset [16-17 hex]** in order to define the zero of the shaft as the system reference has now changed.
After having changed the parameter in **Preset [16-17 hex]** it is not necessary to set new values for travel limits as the Preset function then calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [08-09 hex]** and **Negative delta [0A-0B hex]**.

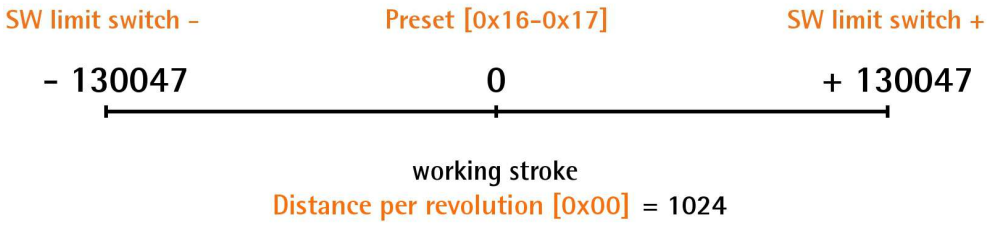
The number of revolutions managed by the system is 127 in negative direction and 127 in positive direction assuming **Preset [16-17 hex]** as reference.
Value set next to the **Positive delta [08-09 hex]** item plus value set in parameter **Preset [16-17 hex]** is the maximum forward travel (positive travel) starting from the preset (value is expressed in pulses).
Value set next to the **Negative delta [0A-0B hex]** item subtracted from value set in parameter **Preset [16-17 hex]** is the maximum backward travel (negative travel) starting from the preset (value is expressed in pulses).

WARNING
Please note that the parameters listed hereafter are closely related to the **Distance per revolution [00 hex]** parameter; hence when you change the value in **Distance per revolution [00 hex]** also the value expressed by each one is necessarily redefined. They are: **Position window [01 hex]**, **Max following error [03 hex]**, **Acceleration [06 hex]**, **Deceleration [07 hex]**, **Positive delta [08-09 hex]**, **Negative delta [0A-0B hex]**, **Target position [2B-2C hex]**, **Current position [02-03 hex]**, **Current velocity [04 hex]** and **Position following error [05-06 hex]**. See for instance the relationship between **Distance per revolution [00 hex]** and the speed values, explained in the previous page.

Example 1

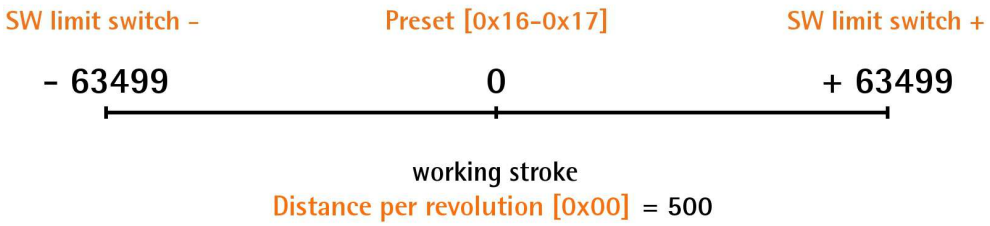
Default values:
Distance per revolution [00 hex] = 1024 steps per revolution
Max. **Work speed [0D hex]**: 1024 pulses per second (1024*1024/1024 = 1024)
Preset [16-17 hex] = 0
Positive delta [08-09 hex] and **Negative delta [0A-0B hex]** max. values = 130047 = (1024 steps per revolution x 127 revolutions) - 1 when **Preset [16-17 hex]** = 0
Max. **SW limit switch +** = 0 + 130047 = + 130047 pulses (forward travel)
Max. **SW limit switch -** = 0 - 130047 = - 130047 pulses (backward travel)

Therefore, when **Preset [16-17 hex]** = 0, the working stroke of the axis will span the maximum positive and negative limits range, that is max. **SW limit switch +** + 130047 and max. **SW limit switch -** – 130047.



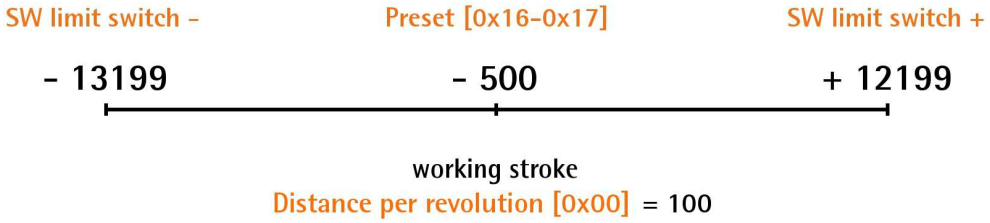
Example 2

ROTADRIVE positioning unit is joined to a worm screw having a 5 mm pitch and you need to have a hundredth of a millimetre resolution.
Distance per revolution [00 hex] = 500 steps per revolution
Max. **Work speed [0D hex]**: 500 pulses per second (1024*500/1024 = 500)
Preset [16-17 hex] = 0
Positive delta [08-09 hex] and **Negative delta [0A-0B hex]** max. values = (500 steps per revolution x 127 revolutions) - 1 = 63499
Max. **SW limit switch +** = 0 + 63499 = + 63499 pulses (forward travel)
Max. **SW limit switch -** = 0 - 63499 = - 63499 pulses (backward travel)
Therefore, when **Preset [16-17 hex]** = 0, the working stroke of the axis will span the maximum positive and negative limits range, that is max. **SW limit switch +** + 63499 and max. **SW limit switch -** – 63499.



Example 3

ROTADRIVE positioning unit is joined to a worm screw having a 1 mm pitch and you need to have a hundredth of a millimetre resolution.
Distance per revolution [00 hex] = 100 steps per revolution
Max. **Work speed [0D hex]**: 100 pulses per second (1024*100/1024 = 100)
Preset [16-17 hex] = -500 (ex. thickness of the tool)
Positive delta [08-09 hex] and **Negative delta [0A-0B hex]** max. values = (100 steps per revolution x 127 revolutions) - 1 = 12699
Max. **SW limit switch +** = (-500) + 12699 = + 12199 pulses (forward travel)
Max. **SW limit switch -** = (-500) - 12699 = - 13199 pulses (backward travel)
Therefore, when **Preset [16-17 hex]** = -500, the working stroke of the axis will span the maximum positive and negative limits range, that is max. **SW limit switch +** + 12199 and max. **SW limit switch -** – 13199.



7 Modbus® interface

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

7.1 Modbus Master / Slaves protocol principle

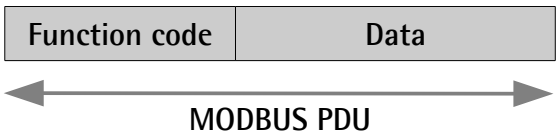
The Modbus Serial Line protocol is a Master – Slaves protocol. One only Master (at the same time) is connected to the bus and one or several (247 maximum number) Slave nodes are also connected to the same serial bus. A Modbus communication is always initiated by the Master. The Slave nodes will never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

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

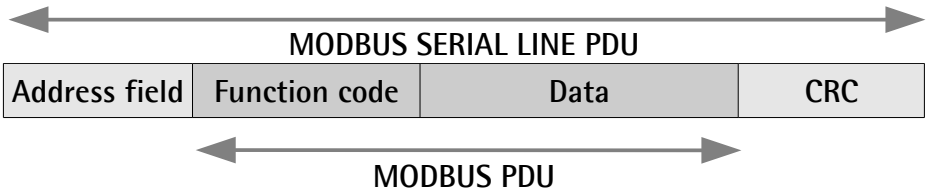
- **UNICAST mode**: in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must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

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 section "4.3.2 Setting the address node: Node ID (Figure 3)" on page 21), the valid Slave node addresses are in the range of 0 – 247 decimal. The individual Slave devices are assigned addresses in the range of 1 – 247. A Master addresses a Slave by placing the Slave address in the **ADDRESS FIELD** of the message. When the Slave returns its response, it places its own address in the response **ADDRESS FIELD** to let the Master know which Slave is responding.
- **FUNCTION CODE**: the function code indicates to the Server what kind of action to perform. The function code can be followed by a **DATA** field that contains request and response parameters. For any further information on the implemented function codes refer to the section "7.4 Function codes" on page 52.
- **DATA**: the **DATA** field of messages contains the bytes for additional information and transmission specifications that the server uses to take the action defined by the **FUNCTION CODE**. This can include items such as discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field. The structure of the **DATA** field depends on each **FUNCTION CODE** (refer to section "7.4 Function codes" on page 52).
- **CRC (Cyclical Redundancy Checking)**: error checking field is the result of a "Redundancy Checking" calculation that is performed on the message contents. This is intended to check whether transmission has

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

The Modbus protocol defines three PDUs. They are:

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

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

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

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

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

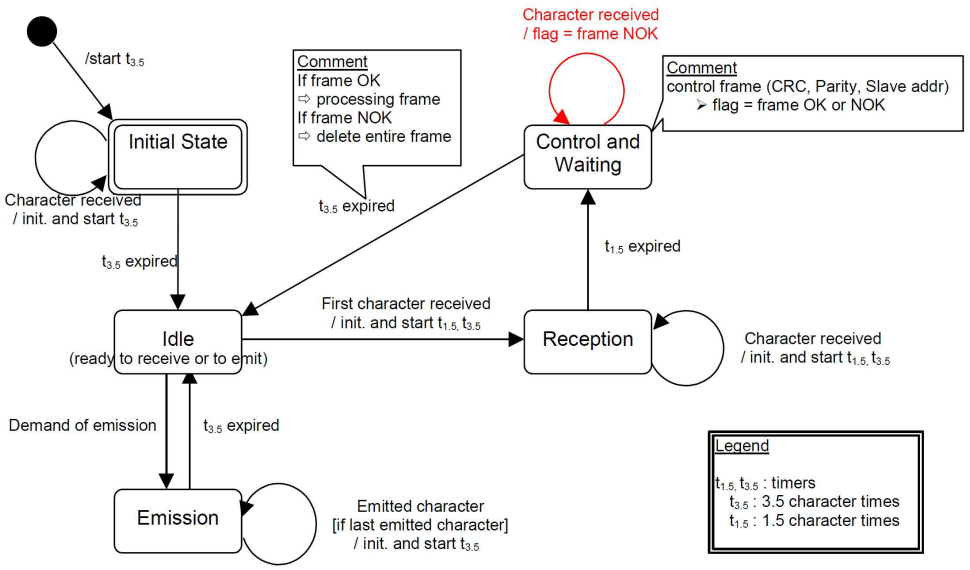
* When "No parity" is activated, the parity bit is replaced by a stop bit.

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 king of action to perform. Function code "0" is not valid.

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

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

7.4.1 Implemented function codes

Lika RD5x-Modbus positioning units only implement public function codes, they are described hereafter.

03 Read Holding Registers

FC = 03 (03 hex) ro

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

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

For the complete list of holding registers accessible using 03 Read Holding Registers function code please refer to section "8.1.1 Machine data parameters" on page 62.

Request PDU

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

Response PDU

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

*N = Quantity of registers

Exception Response PDU

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

Here is an example of a request to read parameters **Acceleration [06 hex]** (register 7) and **Deceleration [07 hex]** (register 8).

Request	
Field name	(Hex)
Function	**03**
Starting address Hi	**00**
Starting address Lo	**06**
No. of registers Hi	**00**
No. of registers Lo	**02**

Response	
Field name	(Hex)
Function	**03**
Byte count	**04**
Register 7 value Hi	**03**
Register 7 value Lo	**E8**
Register 8 value Hi	**05**
Register 8 value Lo	**DC**

As you can see in the table, **Acceleration [06 hex]** parameter (register 7) contains the value 03 E8 hex, i.e. 1000 in decimal notation; **Deceleration [07 hex]** parameter (register 8) contains the value 05 DC hex, i.e. 1500 in decimal notation.

The full frame needed for the request to read the parameters **Acceleration [06 hex]** (register 7) and **Deceleration [07 hex]** (register 8) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][03][00][06][00][02][24][0A]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[00][06] = starting address (**Acceleration [06 hex]** parameter, register 7)
[00][02] = number of requested registers
[24][0A] = CRC

The full frame needed to send back the values of the parameters **Acceleration [06 hex]** (register 7) and **Deceleration [07 hex]** (register 8) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][03][04][03][E8][05][DC][78][8A]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[03][E8] = value of register 7 **Acceleration [06 hex]**, 03 E8 hex = 1000 dec
[05][DC] = value of register 8 **Deceleration [07 hex]**, 05 DC hex = 1500 dec
[78][8A] = CRC

04 Read Input Register

FC = 04 (04 hex)

This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some results values and state / alarm messages in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).
For the complete list of input registers accessible using **04 Read Input Register** function code please refer to section "8.1.2 Input Register parameters" on page 74.

Request PDU

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

Response PDU

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

*N = Quantity of registers

Exception Response PDU

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

Here is an example of a request to read the **Current position [02-03 hex]** 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, **Current position [02-03 hex]** parameter (input registers 3 and 4) contains the value 00 00 2F F0 hex, i.e. 12272 in decimal notation.

The full frame needed for the request to read the **Current position [02-03 hex]** 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 [02-03 hex]** 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 [02-03 hex]** 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 [02-03 hex]**, 00 00 hex = 0 dec
[2F][F0] = value of register 4 **Current position [02-03 hex]**, 2F F0 hex = 12272 dec
[E7][F0] = CRC

06 Write Single Register

FC = 06 (06 hex)

This function code is used to WRITE a single holding register in a remote device. The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0.
The normal response is an echo of the request, returned after the register contents have been written.
For the complete list of registers accessible using **06 Write Single Register** function code please refer to section "8.1.1 Machine data parameters" on page 62.

Request PDU

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

Response PDU

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

Exception Response PDU

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

Here is an example of a request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [06 hex]** parameter (register 7).

Request	
Field name	(Hex)
Function	**06**
Register address Hi	**00**
Register address Lo	**06**
Register value Hi	**05**
Register value Lo	**DC**

Response	
Field name	(Hex)
Function	**06**
Register address Hi	**00**
Register address Lo	**06**
Register value Hi	**05**
Register value Lo	**DC**

As you can see in the table, the value 05 DC hex, i.e. 1500 in decimal notation, is set in the **Acceleration [06 hex]** parameter (register 7).

The full frame needed for the request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [06 hex]** parameter (register 7) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][06][00][06][05][DC][6B][02]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][06] = address of the register (**Acceleration [06 hex]** parameter, register 7)
[05][DC] = value to be set in the register
[6B][02] = CRC

The full frame needed to send back a response following the request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [06 hex]** parameter (register 7) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][06][00][06][05][DC][6B][02]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][06] = address of the register (**Acceleration [06 hex]** parameter, register 7)
[05][DC] = value set in the register
[6B][02] = CRC

16 Write Multiple Registers

FC = 16 (10 hex)

This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
The values to be written are specified in the request data field. Data is packed as two bytes per register.
The normal response returns the function code, starting address and quantity of written registers.
For the complete list of registers accessible using **16 Write Multiple Registers** function code please refer to section "8.1.1 Machine data parameters" on page 62.

Request PDU

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

*N = Quantity of registers

Response PDU

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

Exception Response PDU

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

Here is an example of a request to write the values 1500 and 1000 dec in the parameters **Acceleration [06 hex]** (register 7) and **Deceleration [07 hex]** (register 8) respectively.

Request	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**06**
Quantity of registers Hi	**00**
Quantity of registers Lo	**02**
Byte count	**04**

Response	
Field name	(Hex)
Function	**10**
Starting address Hi	**00**
Starting address Lo	**06**
Quantity of registers Hi	**00**
Quantity of registers Lo	**02**

Register 7 value Hi	**05**
Register 7 value Lo	**DC**
Register 8 value Hi	**03**
Register 8 value Lo	**E8**

As you can see in the table, the value 05 DC hex, i.e. 1500 in decimal notation, is set in the **Acceleration [06 hex]** parameter (register 7); the value 03 E8 hex, i.e. 1000 in decimal notation, is set in the **Deceleration [07 hex]** parameter (register 8).

The full frame needed for the request to write the values 05 DC hex (= 1500 dec) and 03 E8 hex (= 1000 dec) in the parameters **Acceleration [06 hex]** (register 7) and **Deceleration [07 hex]** (register 8) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)
[01][10][00][06][00][02][04][05][DC][03][E8][B2][0D]
where:
[01] = Slave address
[10] = **16 Write Multiple Registers** function code
[00][06] = starting address (**Acceleration [06 hex]** parameter, register 7)
[00][02] = number of requested registers
[04] = number of bytes (2 bytes for each register)
[05][DC] = value to be set in the register 7 **Acceleration [06 hex]**, 05 DC hex = 1000 dec
[03][E8] = value to be set in the register 8 **Deceleration [07 hex]**, 03 E8 hex = 1500 dec
[B2][0D] = CRC

The full frame needed to send back a response following the request to write the values 05 DC hex (= 1500 dec) and 03 E8 hex (= 1000 dec) in the parameters **Acceleration [06 hex]** (register 7) and **Deceleration [07 hex]** (register 8) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][10][00][06][00][02][A1][C9]
where:
[01] = Slave address

[10] = **16 Write Multiple Registers** function code
[00][06] = starting address (**Acceleration [06 hex]** parameter, register 7)
[00][02] = number of written registers
[A1][C9] = CRC

NOTE
For further examples refer to section "Programming examples" on page 83.

WARNING
For safety reasons, when ROTADRIVE 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. 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- 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 [2A hex]** 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 RD5x 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

8.1.1 Machine data parameters

Machine data parameters are accessible for both writing and reading; to read the value set in a parameter use the **03 Read Holding Registers** function code (reading of multiple registers); to write a value in a parameter use the **06 Write Single Register** function code (writing of a single register) or the **16 Write Multiple Registers** (writing of multiple registers); for any further information on the implemented function codes refer to the section "7.4.1 Implemented function codes" on page 52.

Distance per revolution [00 hex]
[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: unit is joined to a warm screw having a 5 mm pitch; by setting **Distance per revolution [00 hex]** = 500, at each shaft revolution system performs a 5 mm pitch with one-hundredth of a millimetre resolution.
Default = 1024 (min. = 1, max. = 1024)

WARNING
After having changed this parameter you must then set new values also in parameters **Jog speed [0C hex]**, **Work speed [0D hex]** and **Preset [16-17 hex]**. For a detailed explanation see on page 43 and relevant parameters.

Please note that the parameters listed hereafter are closely related to the **Distance per revolution [00 hex]** parameter; hence when you change the value in **Distance per revolution [00 hex]** also the value expressed by each one is necessarily redefined. They are: **Position window [01 hex]**, **Max following error [03 hex]**, **Acceleration [06 hex]**, **Deceleration [07 hex]**, **Positive delta [08-09 hex]**, **Negative delta [0A-0B hex]**, **Target position [2B-2C hex]**, **Current position [02-03 hex]**, **Current velocity [04 hex]** and **Position following error [05-06 hex]**. See for instance the relationship between **Distance per revolution [00 hex]** and the speed values, explained on page 43.

NOTE
If **Distance per revolution [00 hex]** is not a power of 2 (2, ..., 512, 1024), at position control a positioning error could occur having a value equal to one pulse.

Position window [01 hex]

[Register 2, Unsigned16, rw]
This parameter defines the tolerance window for the **Target position [2B-2C hex]** value. When the axis is within the tolerance window limits for the time set in the **Position window time [02 hex]** parameter, then the state is signalled through the **Axis in position** status bit. Parameter is expressed in pulses.
Default = 0 (min. = 0, max. = 65535)

Position window time [02 hex]

[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 [01 hex]** parameter before the state is signalled through the **Axis in position** status bit. Parameter is expressed in milliseconds.
Default = 0 (min. = 0, max. = 10000)

Max following error [03 hex]

[Register 4, Unsigned16, 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. Parameter is expressed in pulses.
Default = 1024 (min. = 0, max. = 65535)

Kp position loop [04 hex]

[Register 5, Unsigned16, rw]
This parameter contains the proportional gain used by the PI controller for the position loop. Value has been optimized by Lika Electronic according to the technical characteristics of the device.
Default = 500 (min. = 0, max. = 1000)

Ki position loop [05 hex]

[Register 6, Unsigned16, rw]
This parameter contains the integral gain used by the PI controller for the position loop. Value has been optimized by Lika Electronic according to the technical characteristics of the device.
Default = 60 (min. = 0, max. = 1000)

Acceleration [06 hex]

[Register 7, Unsigned16, rw]
This parameter defines the acceleration value that has to be used by the device. Parameter is expressed in pulses per second^2 [PPS^2].
Default = 2500 (min. = 100, max. = 10000)

Deceleration [07 hex]

[Register 8, Unsigned16, rw]
This parameter defines the deceleration value that has to be used by the device. Parameter is expressed in pulses per second^2 [PPS^2].
Default = 2500 (min. = 100, max. = 10000)

Positive delta [08-09 hex]

[Registers 9-10, Unsigned32, rw]
This value is used to calculate the maximum forward (positive) limit the device is allowed to reach starting from the preset value. When the maximum forward limit is reached, a signalling is activated through the **SW limit switch +** status bit. Parameter is expressed in encoder pulses. For further information see also on page 43.
SW limit switch + = **Preset [16-17 hex]** + **Positive delta [08-09 hex]**.
Default = 130047 (min. = 0, max. = 130047)

Negative delta [0A-0B hex]

[Registers 11-12, Unsigned32, rw]
This value is used to calculate the maximum backward (negative) limit the device is allowed to reach starting from the preset value. When the maximum backward limit is reached, a signalling is activated through the **SW limit switch -** status bit. Parameter is expressed in encoder pulses. For further information see also on page 43.
SW limit switch - = **Preset [16-17 hex]** - **Negative delta [0A-0B hex]**.
Default = 130047 (min. = 0, max. = 130047)

WARNING
Every time **Distance per revolution [00 hex]** and **Preset [16-17 hex]**. parameters are changed, **Positive delta [08-09 hex]** and **Negative delta [0A-0B hex]** values has to be checked carefully. Each time you change the value in **Distance per revolution [00 hex]** you must then update the value in **Preset [16-17 hex]** in order to define the zero of the shaft as the system reference has now changed.
After having changed the parameter in **Preset [16-17 hex]** it is not necessary to set new values for travel limits as the Preset function then calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [08-09 hex]** and **Negative delta [0A-0B hex]** items. For a detailed explanation see on page 43.

Jog speed [0C hex]

[Register 13, Unsigned16, rw]
This parameter contains the maximum speed of the device when using **Jog +** and **Jog -** functions. Parameter is expressed in pulses per second.
Default = 1024 (min. = 1, max. = 1024)

Work speed [0D hex]

[Register 14, Unsigned16, rw]
This parameter contains the maximum speed of the device in automatic work mode (movements are controlled using **Start** command and are performed in order to reach the position set in **Target position [2B-2C hex]**). Parameter is expressed in pulses per second.
Default = 1024 (min. = 1, max. = 1024)

WARNING
Each time you change the value in **Distance per revolution [00 hex]** you must then set new values also in **Jog speed [0C hex]** and **Work speed [0D hex]** as speed values are expressed in pulses per second (PPS). To calculate the speed values you have always to adhere to the following ratio:

$$\frac{\text{Min. Speed} * \text{Distance per revolution [00 hex]}}{1024} \leq \text{Speed} \leq \frac{\text{Max. Speed} * \text{Distance per revolution [00 hex]}}{1024}$$

For a detailed explanation see on page 43.

Start torque current time [0E hex]

[Register 15, Unsigned16, rw]
This parameter defines the maximum time for which the motor is supplied with starting torque current when it starts its movement (see **Starting torque current [13 hex]** item). Parameter is expressed in milliseconds.
Default = 2000 (min. = 0, max. = 3000)

Code sequence [0F hex]

[Register 16, Unsigned16, rw]
It sets the rotation direction of the shaft and consequently defines whether the position value output by the encoder increases when the shaft rotates clockwise (0) or counter-clockwise (1). Clockwise and counter-clockwise rotations are viewed from shaft.
0 = clockwise rotation (default)
1 = 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 **Preset [16-17 hex]** parameter and then check the values set next to the **Positive delta [08-09 hex]** and **Negative delta [0A-0B hex]** items.

Kp current loop [10 hex]

[Register 17, Unsigned16, rw]
This parameter contains the proportional gain used by the PI controller for the current loop. Value has been optimized by Lika Electronic according to the technical characteristics of the device.
Default = 300 (min. = 0, max. = 1000)

Ki current loop [11 hex]

[Register 18, Unsigned16, rw]
This parameter contains the integral gain used by the PI controller for the current loop. Value has been optimized by Lika Electronic according to the technical characteristics of the device.
Default = 60 (min. = 0, max. = 1000)

Max current [12 hex]

[Register 19, Unsigned16, rw]
This parameter defines the maximum current supplied by the power electronic for controlling the motor. Parameter is expressed in mA (milliamperes). This value cannot be greater than the one in **Starting torque current [13 hex]** item.
Default = 2000 (min. = 10, max. = 2000)

Starting torque current [13 hex]

[Register 20, Unsigned16, rw]
This parameter defines the maximum current supplied to the motor only when it starts its movement and for the maximum time set in the **Start torque current time [0E hex]** item. Parameter is expressed in mA (milliamperes).
Default = 4000 (min. = 10, max. = 4000)

Offset [14-15 hex]

[Registers 21-22, Integer32, ro]
This variable defines the difference between the position value transmitted by the device and the real position: real position – preset. Value is expressed in pulses.

Preset [16-17 hex]

[Registers 23-24, Integer32, rw]
Use this object to set the Preset value. Preset function is meant to assign a certain value to a desired physical position of the axis. The chosen physical position will get the value set next to this item and all the previous and following positions will get a value according to it. The preset value will be set for the position of the axis in the moment when the value is entered.
Default = 0 (min. = -262144, max. = 262144)

WARNING
A new value has to be set in **Preset [16-17 hex]** every time **Distance per revolution [00 hex]** value is changed. After having entered a new value in **Preset [16-17 hex]** it is not necessary to set new values for travel limits as the Preset function then calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [08-09 hex]** and **Negative delta [0A-0B hex]** items. For a detailed explanation see on page 43.

Gear ratio [18 hex]

[Register 25, Unsigned16, ro]
It sets the gear ratio of the reduction gear installed between the motor and the encoder shaft. This is a read only register.
Default = 50

Jog step length [19 hex]

[Register 26, Unsigned16, rw]
If the incremental jog function is enabled (bit 4 **Incremental jog** in **Control Word [2A hex]** = 1), the activation of bits **Jog +** and **Jog -** causes at rising edge the execution of a single step toward positive or negative direction having the length, expressed in pulses, set next to this item; then the slave stops and waits for another issue.
Default = 100 (min. = 1, max. = 10000).

Extra commands register [29 hex]

[Register 42, Unsigned16, rw]

Byte structure of the **Extra commands register [29 hex]**:

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

Byte 0

Absolute reading

bit 0: This function is reserved only for use and service of Lika Electronic engineers.

bits 1 ... 7 Not used.

Byte 1 Not used.

Control Word [2A hex]

[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 [2A hex]** register:

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

Byte 0

Jog +

bit 0 If bit 4 **Incremental jog** = 0, as long as **Jog +** = 1, the Slave moves toward positive direction; otherwise if bit 4 **Incremental jog** = 1, the activation of this bit causes at rising edge the execution of a single step toward positive direction having the length, expressed in pulses, set next to the **Jog step length [19 hex]** item; then the slave stops and waits for another issue. Velocity, acceleration and deceleration are set in parameters **Jog speed [0C hex]**, **Acceleration [06 hex]** and **Deceleration [07 hex]** respectively. For a detailed description of jog control see on page 42.

Jog -

bit 1

If bit 4 **Incremental jog** = 0, as long as **Jog -** = 1, the Slave moves toward negative direction; otherwise if bit 4 **Incremental jog** = 1, the activation of this bit causes at rising edge the execution of a single step toward negative direction having the length, expressed in pulses, set next to the **Jog step length [19 hex]** item; then the slave stops and waits for another issue. Velocity, acceleration and deceleration are set in parameters **Jog speed [0C hex]**, **Acceleration [06 hex]** and **Deceleration [07 hex]** respectively. For a detailed description of jog control see on page 42.

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 [07 hex]**. For an immediate halt in the movement, use bit 7 **Emergency**.

Alarm reset

bit 3

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. Normal work status is resumed by switching this bit from "0" to "1". This command is used to reset an alarm condition of the Slave 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 [09-0A hex]**), normal work status can be restored only after having set proper values. **Flash memory error** alarm cannot be reset.

Incremental jog

bit 4

If set to "=0", the activation of 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 bits **Jog +** and **Jog -** causes at rising edge the execution of a single step toward positive or negative direction having the length, expressed in pulses, set next to the **Jog step length [19 hex]** item; then the slave stops and waits for another issue.

bit 5

Not used.

Start

bit 6 — If set to "=1" device moves in order to reach the set target position (see **Target position [2B-2C hex]** on page 72). For a complete description of the position control see on page 41.

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 bit 2 **Stop**.

Byte 1

Watch dog enable

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

Load default parameters

bit 10 — Default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) are restored at each rising edge of the bit; in other words, the default parameters 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 89.

Perform counting preset

bit 11 — Counting acquires the value in the **Preset [16-17 hex]** variable. 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").

Axis torque

bit 12 — When the axis has reached the commanded position, it keeps the torque. This function is available only in RD5 version (model without brake); in the RD53 version (model fitted with brake) bit 12 is not used.
If set to "=0", when the axis is in position, PWM is deactivated.
If set to "=1", when the axis is in position, PWM is kept active.

bit 13 — Not used.

Brake released

bit 14 — This function is available only in RD53 version (model fitted with brake); in the RD5 version (model without brake) bit 14 is not used. RD53 model is fitted with a brake designed to activate as soon as the motor comes to a stop in order to prevent it from moving even slightly. Setting this bit to "=1" causes the brake to be disabled; setting this bit to "=0" causes the brake to be enabled and managed automatically by the system.

Please note that you can disengage the brake only when no alarm is active.

bit 15 — Not used.

Target position [2B-2C hex]

[Registers 44-45, Integer32, rw]
Position to be reached, otherwise referred to as commanded position. When the **Start** command is sent while **Stop** and **Emergency** bits are "=1" and the alarm condition is off, device moves in order to reach the target position.

Position override function
It is possible to change the target position value even while the device is still reaching it; to do this, send a **Start** command and the new target value in **Target position [2B-2C hex]**.

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, jog command will be ignored; if **Jog +** and **Jog -** commands are sent simultaneously, device does not move or, if already moving, it stops its movement.
When the watch dog function is enabled (**Watch dog enable** in **Control Word [2A hex]** is set to "=1"), should the device be disconnected from Modbus network while it is moving (for instance because of a broken cable or faulty wiring), 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).

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

8.1.2 Input Register parameters

Input Register parameters are accessible for reading only; to read the value set in an input register parameter use the **04 Read Input Register** function code (reading of multiple input registers); for any further information on the implemented function codes refer to the section "7.4.1 Implemented function codes" on page 52.

Alarms register [00 hex]

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

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

The available alarm error codes are listed hereafter:

Byte 0

Machine data not valid

bit 0 — One or more parameters are not valid, set proper values to restore normal work condition. See the list of wrong parameters in **Wrong parameters list [09-0A hex]**.

Flash memory error

bit 1 — Internal error, it cannot be restored.

bit 2 — Not used.

Following error

bit 3 — The difference between the real position and the theoretical position is greater than the value set in **Max following error [03 hex]** parameter; we suggest reducing the work speed.

Axis not synchronized

bit 4 — Internal error, it cannot be restored.

Target not valid

bit 5 — Target position is over maximum travel limits.

Emergency

bit 6 — Bit 7 **Emergency** in **Control Word [2A hex]** has been forced to low value (0); or alarms are active in the unit.

Overcurrent

bit 7 — The power supply current is exceeding maximum ratings allowed.

Byte 1

Overtemperature

bit 8 — The internal temperature of the device as sensed by a probe is exceeding maximum ratings (see **Temperature value [08 hex]**).

bit 9 — Not used.

Undervoltage

bit 10 — The power supply voltage is under minimum ratings allowed.

Watch dog

bit 11 — When the Watch dog function is enabled (**Watch dog enable** in **Control Word [2A hex]** is set to "=1"), if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm bit is activated). The alarm is invoked to appear as soon as the Modbus network communication is restored. Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running -a jog command for example- Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered.

bits 12 ... 15 — Not used.

To reset a faulty condition use the **Alarm reset** command, **Control Word [2A hex]** 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. 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 [09-0A hex]**), normal work status can be restored only after having set proper values. **Flash memory error** alarm cannot be reset.

Status word [01 hex]

[Register 2, Unsigned16, ro]
This register contains information about the current state of the device.
Byte structure of the **Status word [01 hex]** register:

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

Byte 0

Axis in position

bit 0 — If value is "=1" device has reached the set position for the time set in **Position window time [02 hex]**. It is kept active until the position error is lower than **Position window [01 hex]**.

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 — If value is "=1" device has reached the maximum positive limit (positive limit switch). See parameter **Positive delta [08-09 hex]**.

SW limit switch -

bit 4 — If value is "=1" device has reached the maximum negative limit (negative limit switch). See parameter **Negative delta [0A-0B hex]**.

Alarm

bit 5 — If value is "=1" an alarm has occurred, see details in the **Alarms register [00 hex]** variable.

Axis running

bit 6 — If value is "=0" device is not moving.
If value is "=1" device is moving.

Executing a command

bit 7	If value is "=0" controller is not executing any command. If value is "=1" controller is executing a command.

Byte 1

Target position reached

bit 8	If value is "=1" device has reached the target position set next to the **Target position [2B-2C hex]** item. Bit is kept active until new **Target position [2B-2C hex]** value or **Alarm reset** commands are sent.
bits 9 ... 11	Not used.

DAC saturation

bit 12	The current supplied by the power electronic for controlling the motor has reached the maximum value and cannot be increased further.
bits 13 ... 15	Not used.

Current position [02-03 hex]

[Registers 3-4, Integer32, ro]
Current position of the device in the moment in which the message is sent. Value is expressed in pulses.

Current velocity [04 hex]

[Register 5, Integer16, ro]
Speed of the device expressed in pulses per second [PPS], updated at every second.

Position following error [05-06 hex]

[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 [03 hex]** parameter, then the **Following error** alarm is triggered and the unit stops. Value is expressed in pulses.

Current value [07 hex]

[Register 8, Integer16, ro]

This variable shows the value of the current absorbed by the motor (rated current). Value is expressed in mA (milliamperes).

Temperature value [08 hex]

[Register 9, Integer16, ro]

This variable shows the value of the internal temperature of the device as sensed by a probe. Value is expressed in °C (Celsius degrees). The minimum detectable temperature is -20°C.

Wrong parameters list [09-0A hex]

[Registers 10-11, 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 normal work status can be restored only after having set proper values.

Bit	Parameter
1	Distance per revolution [00 hex]
2	Position window [01 hex]
3	Position window time [02 hex]
4	Max following error [03 hex]
5	Kp position loop [04 hex]
6	Ki position loop [05 hex]
7	Acceleration [06 hex]
8	Deceleration [07 hex]
9	Positive delta [08-09 hex]
10	Negative delta [0A-0B hex]
11	Jog speed [0C hex]
12	Work speed [0D hex]
13	Start torque current time [0E hex]
14	Code sequence [0F hex]

15	Kp current loop [10 hex]
16	Ki current loop [11 hex]
17	Max current [12 hex]
18	Starting torque current [13 hex]
20	Jog step length [19 hex]
26	Preset [16-17 hex]

I2t [0B hex]

[Register 12, Unsigned16, ro]
Thermal image or signal proportional to the current (for monitoring purposes only).

Dip-switch baud rate [0C hex]

[Register 13, Unsigned16, ro]
This is meant to show the data transmission rate (baud rate) of the serial port fitted in the RD5x unit; data transmission rate has to be set through the provided dip-switch. For any further information on setting the baud rate refer to the section "4.3.3 Setting data transmission rate: Baud rate and Parity bit (Figure 3)" on page 22.

Dip-switch node ID [0D hex]

[Register 14, Unsigned16, ro]
This is meant to show the node address set in the RD5x unit; node address has to be set through the provided dip-switch. For any further information on setting the node ID refer to the section "4.3.2 Setting the address node: Node ID (Figure 3)" on page 21.

SW Version [0E hex]

[Register 15, Unsigned16, ro]
This is meant to show the software version of the ROTADRIVE 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
Ms bit								Ls bit							
Major number								Minor number							

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

HW Version [0F hex]

[Register 16, Unsigned16, ro]
This is meant to show the hardware version of the ROTADRIVE 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
ROTADRIVE model				Interface				Brake	-			Hardware version			

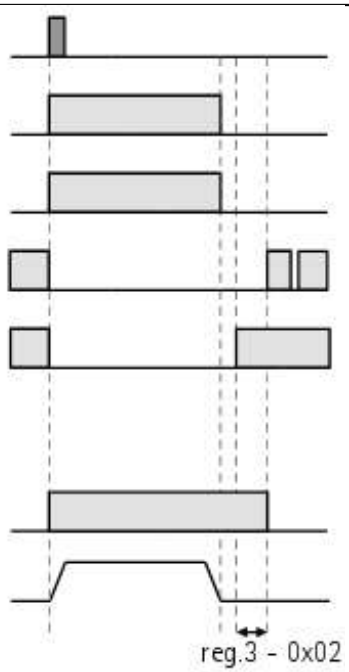
where:

00 ... 03	= hardware version
04 ... 06	= bits not used
07	= brake (0 = without brake; 1 = with brake)
08 .. 11	= interface (00 = Modbus; 01 = Profibus; 02 = CANopen; 03 ... 0F = bits not used)
12 ... 15	= ROTADRIVE model (00 = RD4; 01 = RD1xA; 02 = RD5; 03 ... 0F = bits not used)

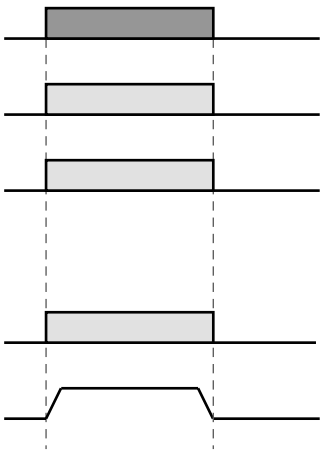
Value 11 81 hex in hexadecimal notation corresponds to the binary representation 00010001 10000001 and has to be interpreted as follows: hardware version 1 (bit 0 = 1); device fitted with brake (bit 7 = 1); Profibus interface (bit 8 = 1); RD1xA model (bit 12 = 1).

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

Example 1

Positioning	RD5x
Start (bit 6 - Control Word [2A hex]) Executing a command (bit 7 - Status word [01 hex]) Axis running (bit 6 - Status word [01 hex]) Axis in position (bit 0 - Status word [01 hex]) Target position reached (bit 8 - Status word [01 hex]) Axis enabled (bit 2 - Status word [01 hex]: PWM is active) Theoretical velocity	

Example 2

Jog +	RD5x
Jog + (bit 0 - Control Word [2A hex]) Executing a command (bit 7 - Status word [01 hex]) Axis running (bit 6 - Status word [01 hex]) Axis enabled (bit 2 - Status word [01 hex]: 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 query:

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

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

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

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

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 parameters **Acceleration [06 hex]** (register 7) and **Deceleration [07 hex]** (register 8) to the Slave having the node address 1.

Request PDU
[01][03][00][06][00][02][24][0A]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[00][06] = starting address (**Acceleration [06 hex]** parameter, register 7)
[00][02] = number of requested registers
[24][0A] = CRC

Response PDU
[01][03][04][03][E8][05][DC][78][8A]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[03][E8] = value of register 7 **Acceleration [06 hex]**, 03 E8 hex = 1000 dec
[05][DC] = value of register 8 **Deceleration [07 hex]**, 05 DC hex = 1500 dec
[78][8A] = CRC

Acceleration [06 hex] parameter (register 7) contains the value 03 E8 hex, i.e. 1000 in decimal notation; **Deceleration [07 hex]** parameter (register 8) contains the value 05 DC hex, i.e. 1500 in decimal notation.

9.2 Using the 04 Read Input Register function code

Example 1
Request to read the Current position [02-03 hex] 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 [02-03 hex] 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 [02-03 hex], 00 00 hex = 0 dec
[2F][F0] = value of register 4 Current position [02-03 hex], 2F F0 hex = 12272 dec
[E7][F0] = CRC

Current position [02-03 hex] 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 [00 hex] 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 [00 hex] 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 [00 hex]**, 00 81 hex = 0000 0000 1000 0001 bin
[79][50] = CRC

This means that in the **Alarms register [00 hex]** variable (register 1) bits 0 and 7 are active (logic level high = 1), i.e. (see on page 74): **Machine data not valid** and **Emergency**.

9.3 Using the 06 Write Single Register function code

Example 1
Request to write the value 05 DC hex (= 1500 dec) in the Acceleration [06 hex] parameter (register 7) of the Slave having the node address 1.

Request PDU
[01][06][00][06][05][DC][6B][02]
where:
[01] = Slave address
[06] = 06 Write Single Register function code
[00][06] = address of the register (Acceleration [06 hex] parameter, register 7)
[05][DC] = value to be set in the register
[6B][02] = CRC

Response PDU
[01][06][00][06][05][DC][6B][02]
where:
[01] = Slave address
[06] = 06 Write Single Register function code
[00][06] = address of the register (Acceleration [06 hex] parameter, register 7)
[05][DC] = value set in the register
[6B][02] = CRC

The value 05 DC hex, i.e. 1500 in decimal notation, is set in the Acceleration [06 hex] parameter (register 7).

Example 2
Request to write the value 00 84 hex in the Control Word [2A hex] 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 [2A hex] 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 [2A hex]** 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 [2A hex]** variable (register 43). In other words, **Stop** and **Emergency** bits are forced to 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 [2A hex]** 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 [2A hex]** 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 [2A hex]** 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 [2A hex]** 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 1500 and 1000 in the parameters Acceleration [06 hex] (register 7) and Deceleration [07 hex] (register 8) of the Slave having the node address 1.

Request PDU
[01][10][00][06][00][02][04][05][DC][03][E8][B2][0D]
where:
[01] = Slave address
[10] = 16 Write Multiple Registers function code
[00][06] = starting address (Acceleration [06 hex] parameter, register 7)
[00][02] = number of requested registers
[04] = number of bytes (2 bytes for each register)
[05][DC] = value to be set in the register 7 Acceleration [06 hex], 05 DC hex = 1000 dec
[03][E8] = value to be set in the register 8 Deceleration [07 hex], 03 E8 hex = 1500 dec
[B2][0D] = CRC

Response PDU
[01][10][00][06][00][02][A1][C9]
where:
[01] = Slave address
[10] = 16 Write Multiple Registers function code
[00][06] = starting address (Acceleration [06 hex] parameter, register 7)
[00][02] = number of written registers
[A1][C9] = CRC

The value 05 DC hex, i.e. 1500 in decimal notation, is set in the Acceleration [06 hex] parameter (register 7); the value 03 E8 hex, i.e. 1000 in decimal notation, is set in the Deceleration [07 hex] parameter (register 8).

10 Default parameters list

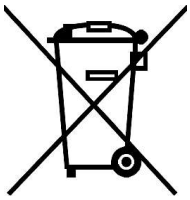
Parameters list	Default value		
Distance per revolution [00 hex] PPR	1024		
Position window [01 hex] P	0		
Position window time [02 hex] ms	0		
Max following error [03 hex] P	1024		
Kp position loop [04 hex]	500		
Ki position loop [05 hex]	60		
Acceleration [06 hex] PPS^2	2500		
Deceleration [07 hex] PPS^2	2500		
Positive delta [08-09 hex] P	130047		
Negative delta [0A-0B hex] P	130047		
Jog speed [0C hex] PPS	1024		
Work speed [0D hex] PPS	1024		
Start torque current time [0E hex] ms	2000		
Code sequence [0F hex]	0		
Kp current loop [10 hex]	300		
Ki current loop [11 hex]	60		
Max current [12 hex] mA	2000		
Starting torque current [13 hex] mA	4000		
Preset [16-17 hex] P	0		
Jog step length [19 hex] P	100		

This page intentionally left blank

This page intentionally left blank

HW-SW release	Document release	Description
1-1.0	1.0	First issue
1-1.0	1.1	Warning against back EMF; parameters Position window [01 hex], Max following error [03 hex], Jog step length [19 hex] updated

Dispose separately



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

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