

User's guide

RD1A

RD12A

CANopen



RS-232 version



..... **lika**

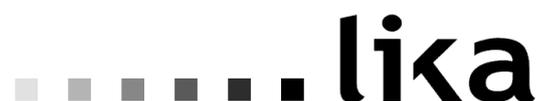
Smart encoders & actuators

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.....	9
Typographic and iconographic conventions.....	11
Preliminary information.....	12
1 Safety summary	13
1.1 Safety.....	13
1.2 Electrical safety.....	13
1.3 Mechanical safety.....	14
2 Identification	15
3 Mechanical installation	16
4 Electrical connections	20
4.1 Ground connection (Figure 1 and Figure 2).....	20
4.2 Connectors (Figure 4).....	22
4.3 Diagnostic LEDs (Figure 4).....	23
4.4 Dip-Switches and buttons (Figure 5).....	27
4.4.1 Setting the node address: Node ID (Figure 5).....	28
4.4.2 Setting data transmission rate: Baud rate (Figure 5).....	29
4.4.3 RT bus termination (Figure 5).....	30
4.4.4 JOG + and JOG – buttons (Figure 5).....	30
4.4.5 PRESET button (Figure 5).....	31
5 Quick reference	32
6 Functions	33
6.1 Working principle.....	33
6.2 Movements: jog and positioning.....	34
Jog: speed control.....	34
Positioning: position and speed control.....	34
6.3 Digital inputs and outputs.....	35
6.4 Distance per revolution, Jog speed, Work speed, Preset and limit switch values.....	36
7 Default parameters list	40
CANopen® interface	41
1 CANopen® interface	42
1.1 EDS files.....	42
1.2 NMT states.....	43
1.2.1 Initialization.....	43
1.2.2 Pre-operational state.....	44
1.2.3 Operational state.....	44
1.2.4 Stopped state.....	44
1.3 Communication messages.....	45
1.3.1 Generic pre-defined connection set.....	46
1.4 NMT messages.....	46
1.5 Boot-up messages.....	47
1.6 PDO messages.....	47
1.6.1 "RECEIVE PDO1" message sent by the Master to the Slave.....	48
Control Word.....	48
Jog +.....	48

Jog -	48
Stop.....	49
Alarm reset.....	49
Incremental jog.....	49
Start.....	49
Emergency.....	50
Axis torque.....	50
OUT 1.....	50
Brake released.....	50
Target position.....	51
1.6.2 "TRANSMIT PDO1" message sent by the Slave to the Master.....	52
Status word.....	52
Axis in position.....	52
Axis enabled.....	52
SW limit switch +.....	52
SW limit switch -.....	52
Alarm.....	52
Axis running.....	52
Executing a command.....	52
Target position reached.....	53
Button 1 Jog +.....	53
Button 2 Jog -.....	53
Button 3 Preset.....	53
DAC saturation.....	53
IN 1.....	53
IN 2.....	54
IN 3.....	54
Current velocity.....	54
Current position.....	54
1.7 SDO messages.....	56
1.7.1 Command.....	56
2 CANopen® programming parameters.....	57
2.1 Objects dictionary.....	57
2.1.1 Standard objects (DS 301).....	57
1000-00 Device type.....	57
1001-00 Error register.....	57
1003 Pre-defined error field.....	58
1005-00 COB-ID SYNC messages.....	58
1008-00 Device name.....	58
1009-00 Hardware version.....	58
100A-00 Software version.....	58
100C-00 Guard Time.....	58
100D-00 Life Time Factor.....	59
1010-01 Save Parameters.....	59
1011-01 Restore Default Parameters.....	59
1014-00 COB-ID EMCY.....	60
1015-00 Inhibit Time Emergency.....	60
1018 Identity object.....	60
1400 Receive PDO Communication Parameter 1.....	61
1600 Receive PDO Mapping Parameter 1.....	62
1800 Transmit PDO Communication Parameter 1.....	62

1A00 TPDO Mapping Parameter 1.....	64
2.1.2 Manufacturer's specific objects.....	66
3000-00 Baud rate.....	66
3001-00 Node ID.....	66
3108-00 Acceleration.....	67
3109-00 Deceleration.....	67
310C-00 Max following error.....	67
310D-00 Position window.....	67
310E-00 Position window time.....	68
310F Delta space.....	68
3111-00 Kp position loop.....	69
3112-00 Ki position loop.....	69
3114-00 Jog speed.....	69
3115-00 Work speed.....	70
3116-00 Max speed.....	70
3117-00 Jog step length.....	70
3118-00 Start Torque current time.....	71
3120-00 Distance per revolution.....	71
3300-00 Preset.....	72
3301-00 Offset.....	72
3302-00 Code sequence.....	72
3330-00 Kp current loop.....	73
3331-00 Ki current loop.....	73
3340-00 Max current.....	73
3341-00 Starting Torque current.....	73
3342-00 Gear ratio.....	73
3343-00 Positive absolute limit switch.....	74
3344-00 Negative absolute limit switch.....	74
3005-00 Current value [mA].....	75
3006-00 Temperature value.....	75
3100-00 Control word.....	75
3101-00 Status word.....	75
3102-00 Demanded position value.....	75
3103-00 Current position value.....	75
3105-00 Current velocity value.....	76
3106-00 Target position.....	76
3107-00 Target speed.....	76
310B-00 Position following error.....	76
3110-00 Cyclic Time.....	76
3200-00 Alarms.....	77
Machine data not valid.....	77
Flash memory error.....	77
Following error.....	77
Axis not synchronized.....	77
Target not valid.....	77
Emergency.....	77
Overcurrent.....	77
Overtemperature.....	78
Undervoltage.....	78
CAN Life guard error.....	78
2.2 Warning messages.....	79

2.3 Emergency messages.....	79
No active errors.....	79
Generic error.....	79
Power surge.....	79
Overvoltage.....	79
Undervoltage.....	79
Overtemperature.....	79
Flash memory.....	79
Life Guard.....	79
Following error.....	79
2.4 Node guarding protocol.....	80
3 CANopen® programming examples.....	81
Modbus® interface.....	82
1 Using the RS-232 service serial port.....	83
1.1 Configuring the device using Lika setting up software.....	83
1.2 "Serial configuration" page.....	85
1.3 "Operative mode" page.....	88
1.4 "Machine data" page.....	93
1.5 "Message monitor" page.....	95
1.6 "Test Lika" page.....	96
1.7 "Upgrade Firmware" page.....	96
1.7.1 If there is an installation issue.....	99
2 Modbus® interface.....	101
2.1 Modbus Master / Slaves protocol principle.....	101
2.2 Modbus frame description.....	102
2.3 Transmission modes.....	103
2.3.1 RTU transmission mode.....	104
2.4 Function codes.....	106
2.4.1 Implemented function codes.....	106
03 Read Holding Registers.....	106
04 Read Input Register.....	108
06 Write Single Register.....	110
16 Write Multiple Registers.....	112
3 Modbus® programming parameters.....	116
3.1 Parameters available.....	116
3.1.1 Machine data parameters.....	116
Distance per revolution [0x00].....	116
Position window [0x01].....	117
Position window time [0x02].....	117
Max following error [0x03].....	117
Kp position loop [0x04].....	118
Ki position loop [0x05].....	118
Acceleration [0x06].....	118
Deceleration [0x07].....	118
Positive delta [0x08-0x09].....	118
Negative delta [0x0A-0x0B].....	119
Jog speed [0x0C].....	120
Work speed [0x0D].....	121
Start torque current time [0x0E].....	121
Code sequence [0x0F].....	121
Kp current loop [0x10].....	122

Ki current loop [0x11].....	122
Max current [0x12].....	122
Starting torque current [0x13].....	122
Offset [0x14-0x15].....	123
Preset [0x16-0x17].....	123
Gear ratio [0x18].....	123
Jog step length [0x19].....	124
Extra commands register [0x29].....	124
Absolute reading.....	124
Control from PC.....	124
Control Word [0x2A].....	125
Jog +.....	125
Jog -.....	125
Stop.....	125
Alarm reset.....	126
Incremental jog.....	126
Start.....	126
Emergency.....	126
Watch dog enable.....	126
Save parameters.....	127
Load default parameters.....	127
Perform counting preset.....	127
Axis torque.....	127
OUT 1.....	127
Brake released.....	128
Target position [0x2B-0x2C].....	128
3.1.2 Input Register parameters.....	130
Alarms register [0x00].....	130
Machine data not valid.....	130
Flash memory error.....	130
Following error.....	130
Axis not synchronized.....	130
Target not valid.....	130
Emergency.....	130
Overcurrent.....	131
Overtemperature.....	131
Undervoltage.....	131
Watch dog.....	131
Status word [0x01].....	132
Axis in position.....	132
Axis enabled.....	132
SW limit switch +.....	132
SW limit switch -.....	132
Alarm.....	132
Axis running.....	132
Executing a command.....	133
Target position reached.....	133
Button 1 Jog +.....	133
Button 2 Jog -.....	133
Button 3 Preset.....	133
DAC saturation.....	133

IN 1.....	134
IN 2.....	134
IN 3.....	134
Current position [0x02-0x03].....	134
Current velocity [0x04].....	134
Position following error [0x05-0x06].....	134
Current value [0x07].....	135
Temperature value [0x08].....	135
Wrong parameters list [0x09-0x0A].....	135
I2t [0x0B].....	136
Dip-switch baud rate [0x0C].....	136
Dip-switch node ID [0x0D].....	136
SW Version [0x0E].....	136
HW Version [0x0F].....	137
3.2 Exception codes.....	139
4 Modbus® programming examples.....	140
4.1 Using the 03 Read Holding Registers function code.....	140
4.2 Using the 04 Read Input Register function code.....	141
4.3 Using the 06 Write Single Register function code.....	143
4.4 Using the 16 Write Multiple Registers function code.....	145

Subject Index

1	
1000-00 Device type.....	57
1001-00 Error register.....	57
1003 Pre-defined error field.....	58
1005-00 COB-ID SYNC messages.....	58
1008-00 Device name.....	58
1009-00 Hardware version.....	58
100A-00 Software version.....	58
100C-00 Guard Time.....	58
100D-00 Life Time Factor.....	59
1010-01 Save Parameters.....	59
1011-01 Restore Default Parameters.....	59
1014-00 COB-ID EMCY.....	60
1015-00 Inhibit Time Emergency.....	60
1018 Identity object.....	60
1400 Receive PDO Communication Parameter 1.....	61
1600 Receive PDO Mapping Parameter 1.....	62
1800 Transmit PDO Communication Parameter 1.....	62
1A00 TPDO Mapping Parameter 1.....	64
3	
3000-00 Baud rate.....	66
3001-00 Node ID.....	66
3005-00 Current value [mA].....	75
3006-00 Temperature value.....	75
3100-00 Control word.....	75
3101-00 Status word.....	75
3102-00 Demanded position value.....	75
3103-00 Current position value.....	75
3105-00 Current velocity value.....	76
3106-00 Target position.....	76
3107-00 Target speed.....	76
3108-00 Acceleration.....	67
3109-00 Deceleration.....	67
310B-00 Position following error.....	76
310C-00 Max following error.....	67
310D-00 Position window.....	67
310E-00 Position window time.....	68
310F Delta space.....	68
3110-00 Cyclic Time.....	76
3111-00 Kp position loop.....	69
3112-00 Ki position loop.....	69
3114-00 Jog speed.....	69
3115-00 Work speed.....	70
3116-00 Max speed.....	70
3117-00 Jog step length.....	70
3118-00 Start Torque current time.....	71
3120-00 Distance per revolution.....	71
3200-00 Alarms.....	77
3300-00 Preset.....	72
3301-00 Offset.....	72
3302-00 Code sequence.....	72
3330-00 Kp current loop.....	73
3331-00 Ki current loop.....	73
3340-00 Max current.....	73
3341-00 Starting Torque current.....	73
3342-00 Gear ratio.....	73
3343-00 Positive absolute limit switch.....	74
3344-00 Negative absolute limit switch.....	74
A	
Absolute reading.....	124
Acceleration [0x06].....	118
Alarm.....	52, 132
Alarm reset.....	49, 126
Alarms register [0x00].....	130
Axis enabled.....	52, 132
Axis in position.....	52, 132
Axis not synchronized.....	77, 130
Axis running.....	52, 132
Axis torque.....	50, 127
B	
Brake released.....	50, 128
Button 1 Jog +.....	53, 133
Button 2 Jog -.....	53, 133
Button 3 Preset.....	53, 133
C	
CAN Life guard error.....	78
Code sequence [0x0F].....	121
Control from PC.....	124
Control Word.....	48
Control Word [0x2A].....	125
Current position.....	54
Current position [0x02-0x03].....	134
Current value [0x07].....	135
Current velocity.....	54
Current velocity [0x04].....	134
D	
DAC saturation.....	53, 133
Deceleration [0x07].....	118
Dip-switch baud rate [0x0C].....	136
Dip-switch node ID [0x0D].....	136

Distance per revolution [0x00].....	116
E	
Emergency.....	50, 77, 126, 130
Executing a command.....	52, 133
Extra commands register [0x29].....	124
F	
Flash memory.....	79
Flash memory error.....	77, 130
Following error.....	77, 79, 130
G	
Gear ratio [0x18].....	123
Generic error.....	79
H	
HW Version [0x0F].....	137
I	
I2t [0x0B].....	136
IN 1.....	53, 134
IN 2.....	54, 134
IN 3.....	54, 134
Incremental jog.....	49, 126
J	
Jog -.....	48, 125
Jog +.....	48, 125
Jog speed [0x0C].....	120
Jog step length [0x19].....	124
K	
Ki current loop [0x11].....	122
Ki position loop [0x05].....	118
Kp current loop [0x10].....	122
Kp position loop [0x04].....	118
L	
Life Guard.....	79
Load default parameters.....	127
M	
Machine data not valid.....	77, 130
Max current [0x12].....	122
Max following error [0x03].....	117
N	
Negative delta [0x0A-0x0B].....	119

No active errors.....	79
O	
Offset [0x14-0x15].....	123
OUT 1.....	50, 127
Overcurrent.....	77, 131
Overtemperature.....	78 e seg., 131
Overvoltage.....	79
P	
Perform counting preset.....	127
Position following error [0x05-0x06].....	134
Position window [0x01].....	117
Position window time [0x02].....	117
Positive delta [0x08-0x09].....	118
Power surge.....	79
Preset [0x16-0x17].....	123
S	
Save parameters.....	127
Start.....	49, 126
Start torque current time [0x0E].....	121
Starting torque current [0x13].....	122
Status word.....	52
Status word [0x01].....	132
Stop.....	49, 125
SW limit switch -.....	52, 132
SW limit switch +.....	52, 132
SW Version [0x0E].....	136
T	
Target not valid.....	77, 130
Target position.....	51
Target position [0x2B-0x2C].....	128
Target position reached.....	53, 133
Temperature value [0x08].....	135
U	
Undervoltage.....	78 e seg., 131
W	
Watch dog.....	131
Watch dog enable.....	126
Work speed [0x0D].....	121
Wrong parameters list [0x09-0x0A].....	135

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 RD1A and RD12A models**.

RD1A and RD12A 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 RD12A model from RD1A 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. In the Profibus and CANopen versions configuration can be done using the same program through a **service RS-232 serial interface and in compliance with Modbus protocol**.

To make it easier to read the text, this guide is divided into three main sections.

In the first section general information concerning the safety, the mechanical installation and the electrical connection as well as tips for setting up and running properly and efficiently the unit are provided. In this section CANopen and Modbus logos located next to the text are meant to highlight the information which specifically concerns each interface. Further distinctions can be made through notes.

In the second section, entitled **CANopen Interface**, both general and specific information is given on the CANopen interface. In this section the interface features and the objects implemented in the unit are fully described.

In the third section, entitled **Modbus Interface**, both general and specific information is given on the Modbus interface. As previously stated, Profibus and CANopen versions are equipped with a service RS-232 serial interface and in compliance with Modbus protocol. Using a software expressly developed and released by Lika Electronic for free it allows the operator to configure the ROTADRIVE unit before installation in the Profibus or CANopen fieldbus networks. In the **Modbus Interface** 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 "Mechanical installation";
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the unit;
- do not tool the unit or its shaft;
- delicate electronic equipment: handle with care; do not subject the device and the shaft to knocks or shocks;
- respect the environmental characteristics of the product;
- unit with solid shaft: in order to guarantee maximum reliability over time of mechanical parts, we recommend a flexible coupling to be installed to connect ROTADRIVE and user's shaft; make sure the misalignment tolerances of the flexible coupling are respected;
- unit with hollow shaft: ROTADRIVE can be mounted directly on a shaft whose diameter has to respect the technical characteristics specified in the purchase order and clamped by means of the collar and 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 Mechanical installation



WARNING

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

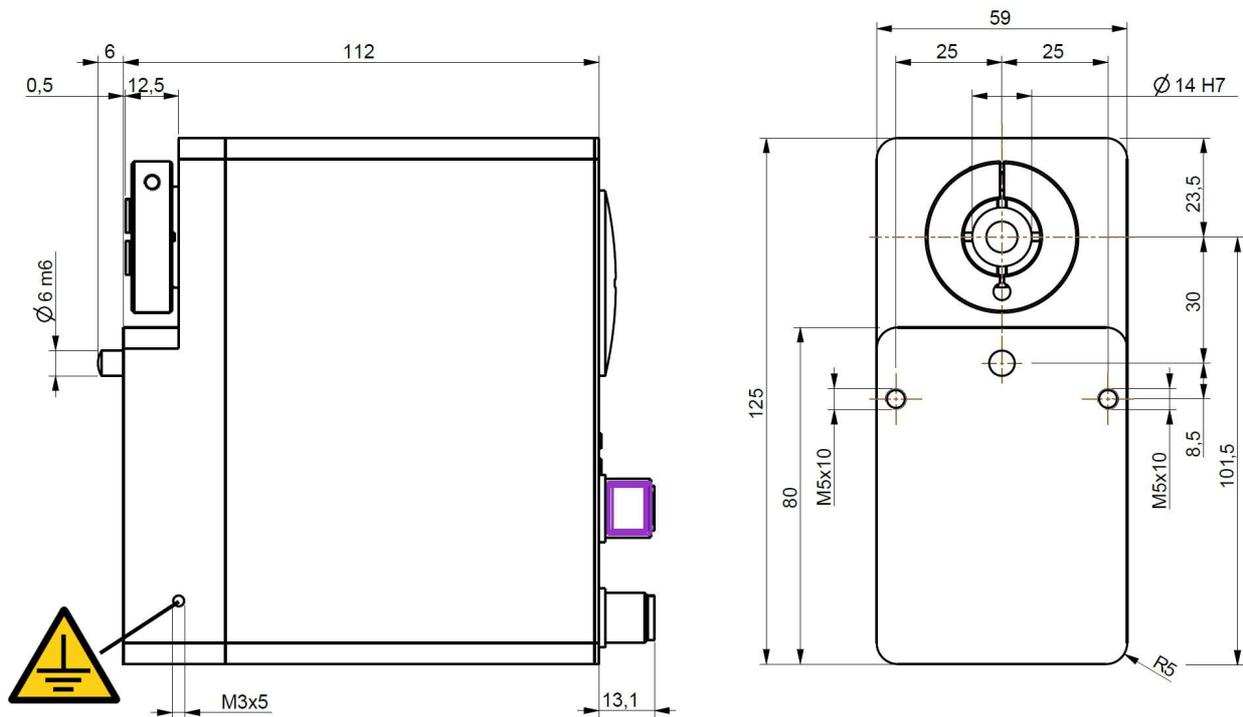


Figure 1 - RD1A unit – Overall dimensions

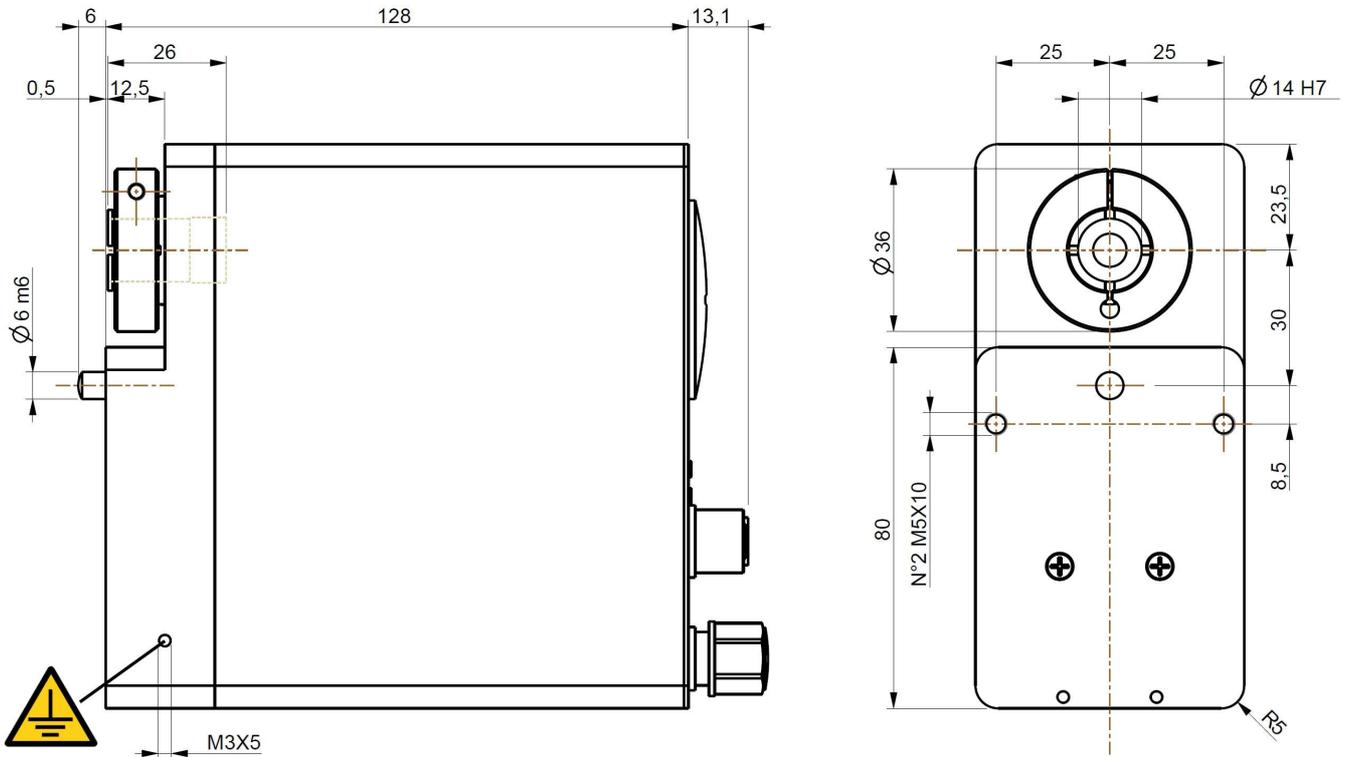


Figure 2 - RD12A unit – Overall dimensions



ROTADRIVE unit must be secured firmly only to the user's shaft using the provided collar. ROTADRIVE unit is supplied with a silicone isolator and an anti-rotation pin; the anti-rotation pin has to be inserted into the silicone isolator. 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 flange or the fixed support on user's side without using the silicone isolator! Furthermore do not place the flange of the positioning unit against the flange on user's side. If this occurs, the mechanical tensions would be transmitted completely to the motor shaft and this would lead to bearings damages and mechanical breakdowns!

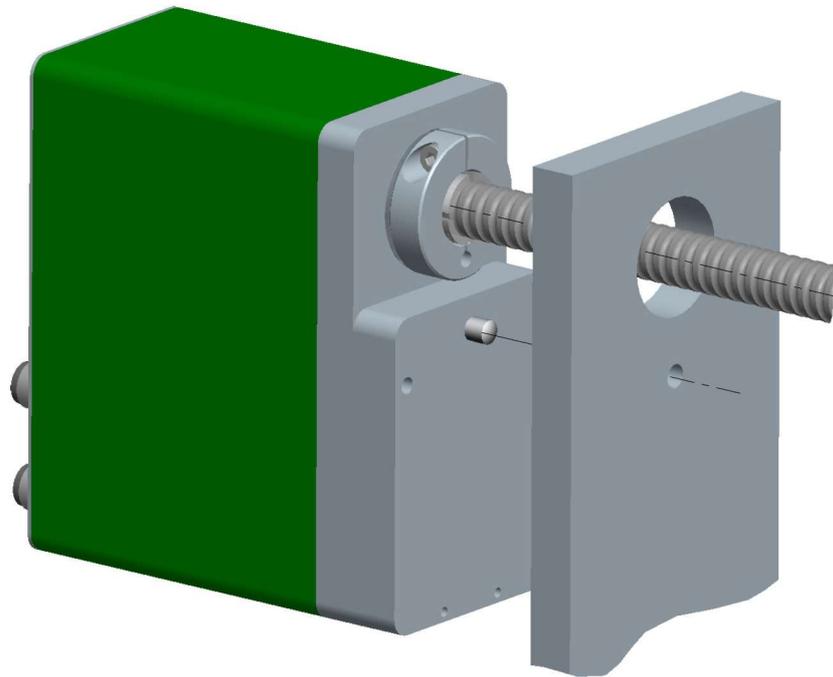
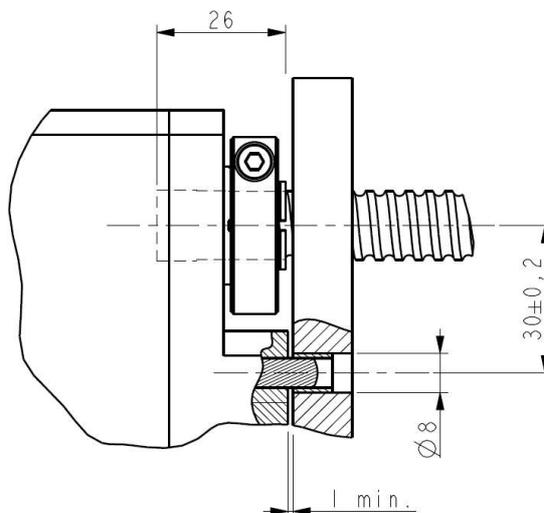


Figure 3 - Typical installation example of RD1xA unit on worm screw

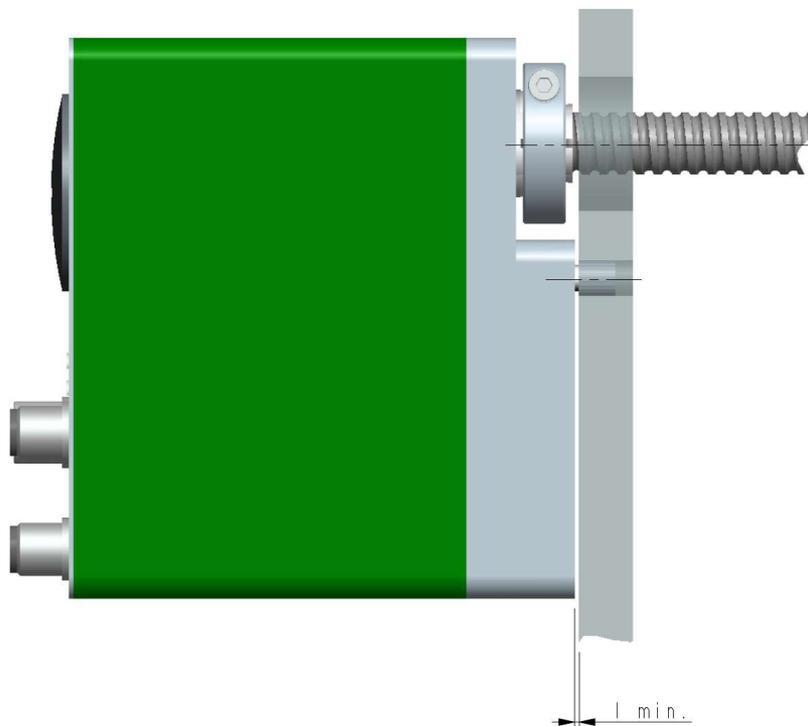
To install properly the ROTADRIVE unit please read carefully and follow these instructions; anyway note that the unit can be installed in several manners and according to the specific user's application.

- Drill a 8 mm diameter hole in the flange or in the fixed support on user's side in order to insert the silicone isolator and the anti-rotation pin. The distance between the axis of the shaft and the axis of the hole must be $30 \pm 0,2$ 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 silicone isolator in the hole;
- insert the user's shaft in the hollow shaft of the ROTADRIVE unit; the maximum depth of the ROTADRIVE shaft is 26 mm; ascertain that the anti-rotation pin is inserted properly in the silicone isolator;
- the minimum distance between the flange of the ROTADRIVE unit 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;
- secure the user's shaft through the collar and the relevant fixing screw.



WARNING

Never force manually the rotation of the shaft not to cause permanent damages! The counter-electromotive force (back EMF) generated by the motor in case the shaft is forced to 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!

4.1 Ground connection (Figure 1 and Figure 2)

To minimize noise connect properly the frame to ground; we suggest using the ground screw provided in the frame (see Figure 1 and Figure 2). 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.

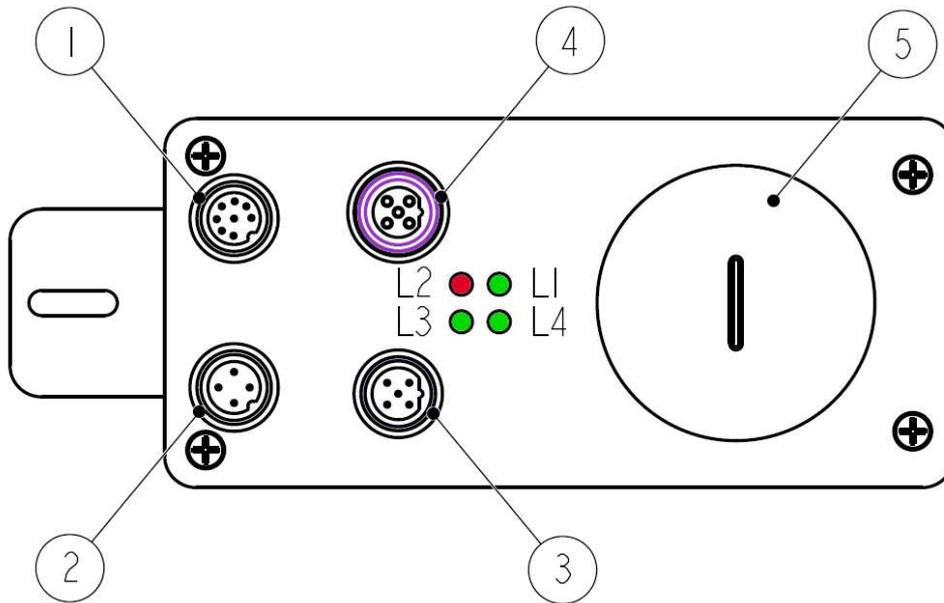


Figure 4: Electrical connections

Legend

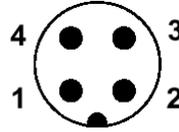
1	M12 8-pin male connector inputs / outputs + Modbus RS-232
2	M12 4-pin male connector power supply
3	M12 5-pin male connector CANopen BUS IN
4	M12 5-pin female connector CANopen BUS OUT
5	Internal housing of dip-switches and buttons
L1	LED 1 controller power supply information
L2	LED 2 active errors / faults information
L3	LED 3 fieldbus interface status information
L4	LED 4 motor power supply information

4.2 Connectors (Figure 4)

Power supply

M12 4-pin male connector

A coding
(frontal side)



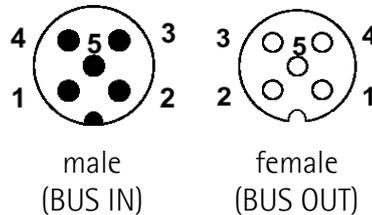
Pin	Description
1	+24VDC ±10% motor power supply
2	+24VDC ±10% controller power supply
3	motor and controller 0 VDC supply voltage
4	n.c.

n.c. = not connected

CANopen interface

M12 5-pin connectors

A coding
(frontal side)



Pin	Description
1	n.c.
2	n.c.
3	CAN GND ¹
4	CAN High
5	CAN Low
Case	Shielding ²

¹ CAN GND is the 0VDC reference of CAN signals, it is not connected to 0VDC supply voltage.

² Lika's EC- pre-assembled cables only.

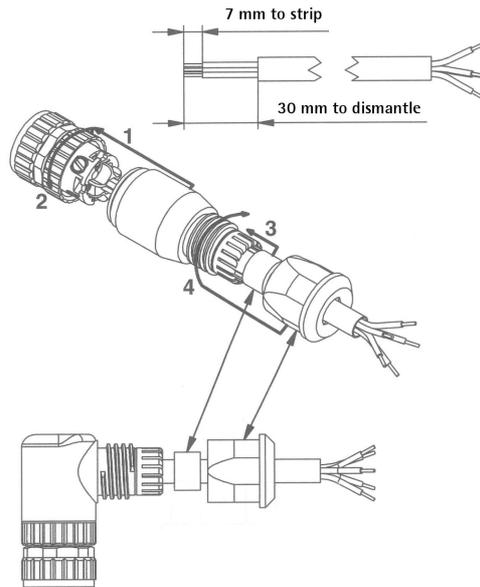
n.c. = not connected

We recommend CANopen certified cables to be used.



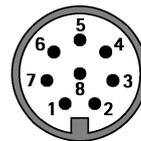
NOTE

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.



Inputs / outputs + RS-232 Modbus

M12 8-pin male connector



(frontal side)

Pin	Description
1	0 VDC
2	Input 1
3	Input 2
4	Input 3
5	Output 1
6	TD (RS-232) ¹
7	RD (RS-232) ¹
8	0 VDC (RS-232) ¹

¹ For Modbus service serial interface only. For any further information on configuring and using the RS-232 service serial port refer to the section "Modbus® interface" on page 82.

4.3 Diagnostic LEDs (Figure 4)

Four LEDs located next to the BUS IN & BUS OUT connectors (see Figure 4) are meant to show visually the operating or fault status of the fieldbus interfaces and the device as well. The meaning of each LED is explained in the following tables.

Please note the LEDs have different meanings depending on the active fieldbus interface.



CANopen

LED 1 GREEN		Description
ON		Indicates the controller power supply is turned on
OFF		Indicates the controller power supply is turned off
LED 2 RED ERR		Description
ON		Bus off
Double flash		Node guarding error (see on page 80)
Single flash		Maximum number of warning errors
Blinking led		General error or Flash memory error
OFF		No error
LED 3 GREEN RUN		Description
ON		Device in Operational state
Single flash		Device in Stopped state
Blinking led		Device in Pre-Operational state
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 "1.7 "Upgrade Firmware" page" on page 96), 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 "1.7 "Upgrade Firmware" page" on page 96), 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 "1.7 "Upgrade Firmware" page" on page 96.
RED ON	RED ON	While downloading data to the flash memory for upgrading the firmware of the unit (see section "1.7 "Upgrade Firmware" page" on page 96), 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 "1.7

		"Upgrade Firmware" page" on page 96.
LED 4 GREEN	Description	
ON	Indicates the motor power supply is turned on	
OFF	Indicates the motor power supply is turned off	



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

LED 2 RED	Description
ON	Active alarms, internal error
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 "1.7 "Upgrade Firmware" page" on page 96), 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 "1.7 "Upgrade Firmware" page" on page 96), 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 "1.7 "Upgrade Firmware" page" on page 96.
RED ON	RED ON	While downloading data to the flash memory for upgrading the firmware of the unit (see section "1.7 "Upgrade Firmware" page" on page 96), 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 "1.7 "Upgrade Firmware" page" on page 96.

LED 4 GREEN	Description
ON	Indicates the motor power supply is turned on

OFF	Indicates the motor power supply is turned off
-----	--

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

4.4 Dip-Switches and buttons (Figure 5)



WARNING

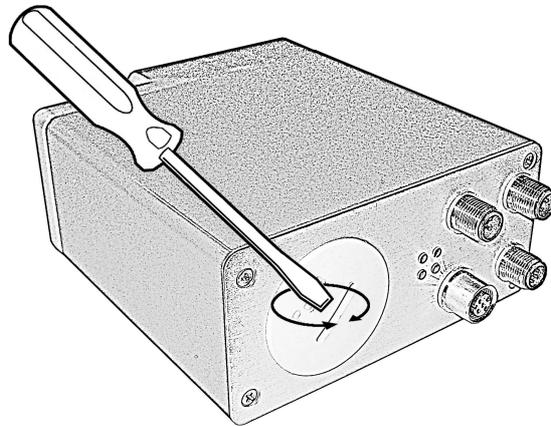
Power supply must be turned off before performing this operation!



NOTE

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

To access DIP-Switches and buttons loosen and remove the screw plug (PG 29 thread). Be careful to replace the screw plug at the end of the operation.



DIP-switches and buttons are located just beneath.

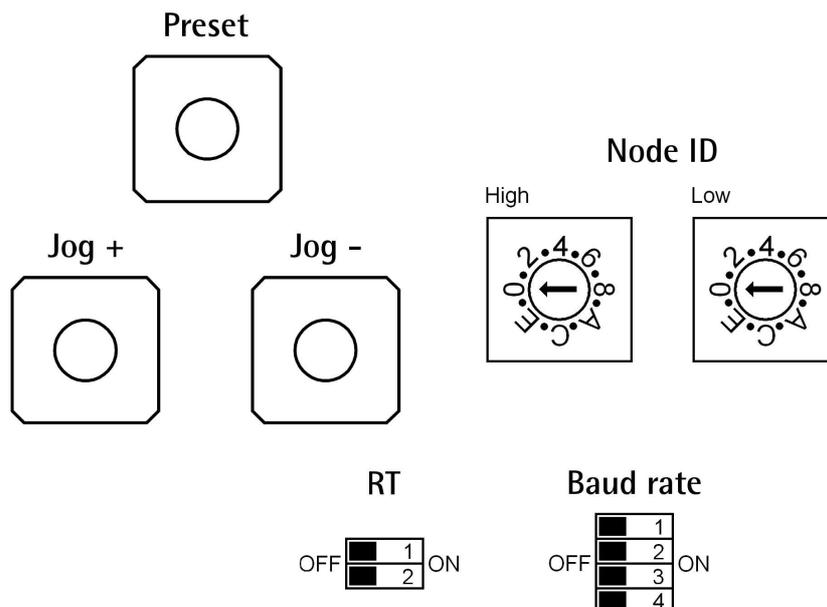


Figure 5: Dip-Switches and buttons

4.4.1 Setting the node address: Node ID (Figure 5)



WARNING

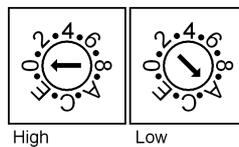
Power supply must be turned off before performing this operation!

Set the node address expressed in hexadecimal notation.
The range of node addresses is between 1 and 127 (127 = 7F hex).

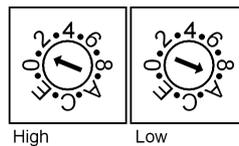


Example

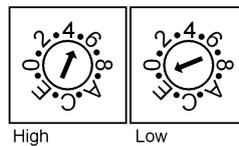
Address 10 = 0A hex:



Address 25 = 19 hex:



Address 95 = 5F hex:



NOTE

The node address which is set using this selector concerns both CANopen and Modbus fieldbus interfaces.

The default address is 1.

If you set the address to 0, device will be set to 1 automatically (address 0 is reserved for Master).

If you set an address higher than 127, device will be set to 127 automatically.

4.4.2 Setting data transmission rate: Baud rate (Figure 5)



WARNING

Power supply must be turned off before performing this operation!

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

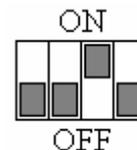
Switch 1 ... 4	Baud rate
0000	20 Kbit/s
1000	50 Kbit/s
0100	100 Kbit/s
1100	125 Kbit/s
0010	250 Kbit/s
1010 (default)	500 Kbit/s
0110	800 Kbit/s
1110	1000 Kbit/s



Example

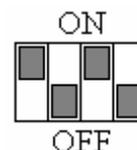
Set the baud rate to 250 Kbits per second:

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



Set the baud rate to 500 Kbits per second:

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



NOTE

The data transmission rate which is set using this selector concerns the CANopen fieldbus interface only.

Configuration parameters of Modbus service serial port have fixed values so the user cannot change them.

They are:

RS-232 Modbus service serial

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

For any further information on configuring and using the RS-232 service serial port refer to the section "Modbus® interface" on page 82.

4.4.3 RT bus termination (Figure 5)

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

4.4.4 JOG + and JOG - buttons (Figure 5)

Press these buttons to force the manual movements of the motor toward positive direction (JOG +) and toward negative direction (JOG -). For any further information see **Jog +** and **Jog -** commands on page 48 (CANopen interface) or page 125 (Modbus interface).



NOTE

Please note that when you use the manual buttons the "incremental jog" function (see **Incremental jog** in **Control Word** on page 49 -CANopen version; **Incremental jog** in **Control Word [0x2A]** on page 126 -Modbus version) is disabled; that is, jog step movements are not allowed using the manual buttons. Thus positive and negative movements are commanded only by keeping pressed the buttons continuously and come to an end when reaching the set limits.



WARNING

JOG and PRESET buttons are always active and available for use to the operator, even when the ROTADRIVE unit is in an alarm or emergency condition. Before pressing these buttons, please make sure that the device is free to move in the safest way and there are no risks that it could lead to personal injury and/or damage to the unit or other equipment.

4.4.5 PRESET button (Figure 5)

This button is meant to assign the value set next to the **Preset** item to the current position of the axis. The button has to be kept pressed for at least 3 seconds. For any further information see **Preset** object on page 72 (CANopen interface) or page 123 (Modbus interface).



WARNING

JOG and PRESET buttons are always active and available for use to the operator, even when the ROTADRIVE unit is in an alarm or emergency condition. Before pressing these buttons, please make sure that the device is free to move in the safest way and there are no risks that it could lead to personal injury and/or damage to the unit or other equipment.

5 Quick reference

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

- Mechanically install the device;
- execute electrical connections;
- set data transmission rate (baud rate; see on page 29); the default value set by Lika Electronic at factory set-up is "baud rate = 500 Kbits/s";
- set the node address (node ID; see on page 28); 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** item (see on page 71 -CANopen interface- or page 116 -Modbus interface);
- set a proper value next to the **Jog speed** item (see on page 69 -CANopen interface- or page 120 -Modbus interface);
- set a proper value next to the **Work speed** item (see on page 70 -CANopen interface- or page 121 -Modbus interface);
- set a proper value next to the **Preset** item (see on page 72 -CANopen interface- or page 123 -Modbus interface);
- set the limit switch values next to the **310F Delta space** item (objects **310F Delta space** sub 1 **Positive Delta** and **310F Delta space** sub 2 **Negative Delta**; see on page 68 -CANopen interface) or next to the **Positive delta [0x08-0x09]** and **Negative delta [0x0A-0x0B]** items, see on page 118 -Modbus interface);
- save the new configuration values using the object **1010-01 Save Parameters** 1010h sub 1 (see on page 59 -CANopen interface) or bit 9 **Save parameters** in the **Control Word [0x2A]** register; see on page 127 -Modbus interface).



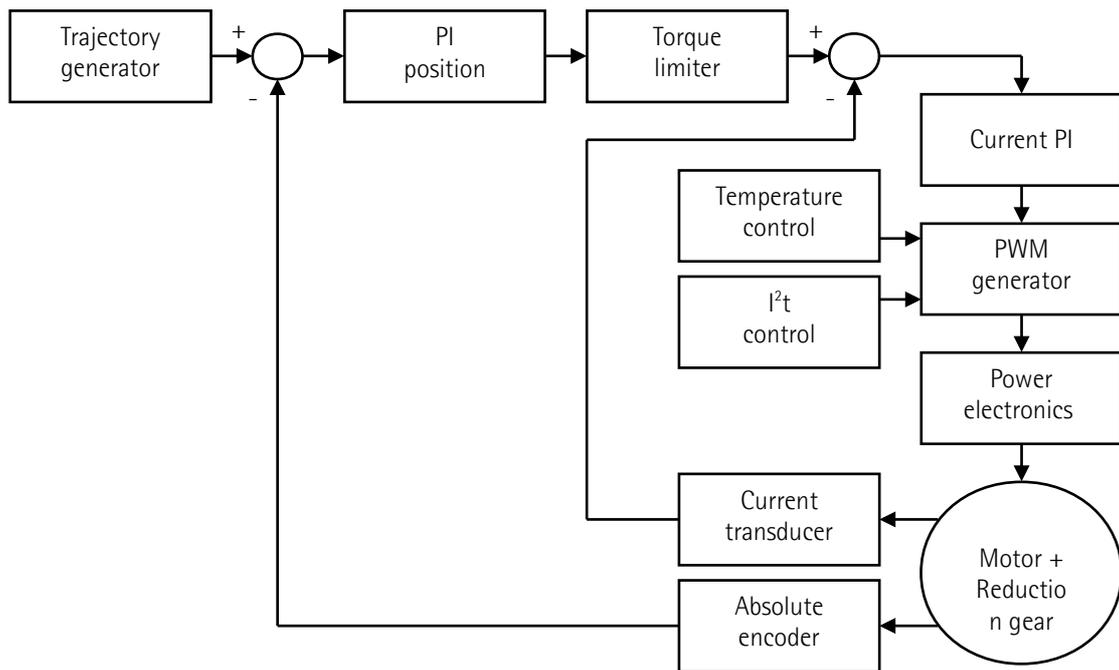
NOTE

Parameters **Distance per revolution**, **Jog speed**, **Work speed** and **Preset** as well as the limit switch values (objects **310F Delta space** sub 1 **Positive Delta** and **310F Delta space** sub 2 **Negative Delta** -CANopen interface- and registers **Positive delta [0x08-0x09]** and **Negative delta [0x0A-0x0B]** -Modbus interface) 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 36.

6 Functions

6.1 Working principle

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



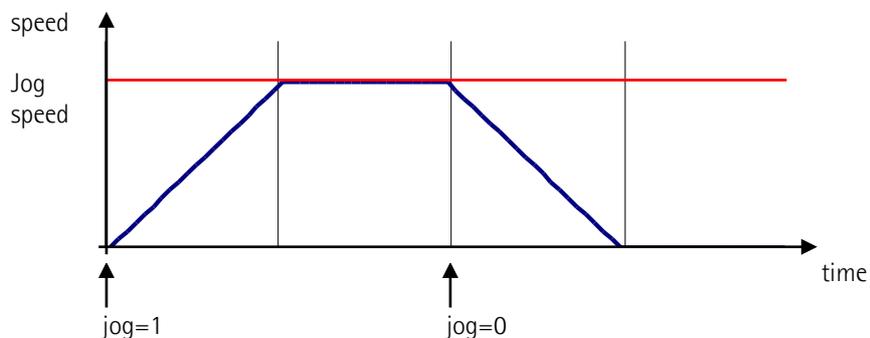
6.2 Movements: jog and positioning

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

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

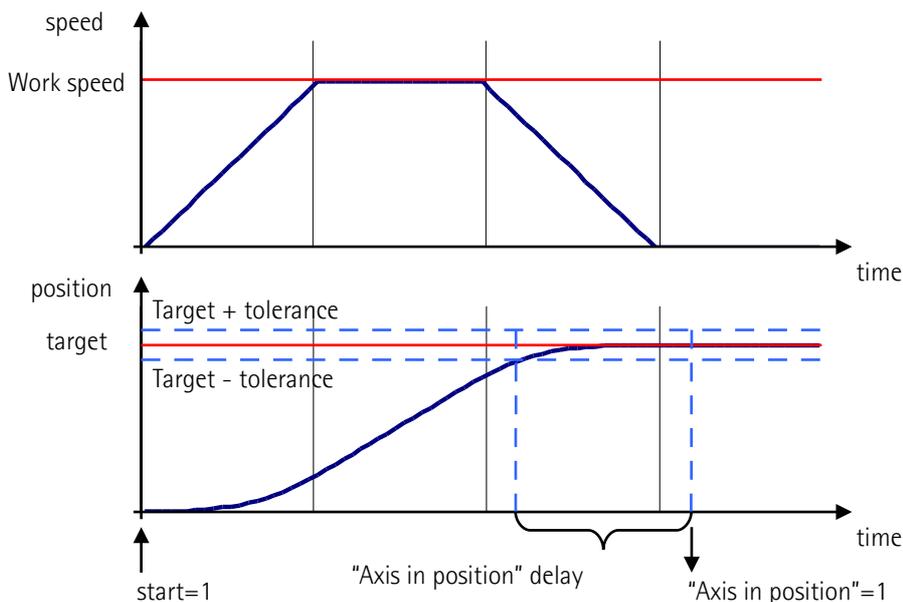
Jog: speed control

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** (see on page 69 -CANopen interface-, page 120 -Modbus interface).



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** (see on page 70 -CANopen interface-, page 121 -Modbus interface); set speed can be reached only if space is long enough.



6.3 Digital inputs and outputs

RD1xA unit is fitted with **three digital inputs and one digital output**.

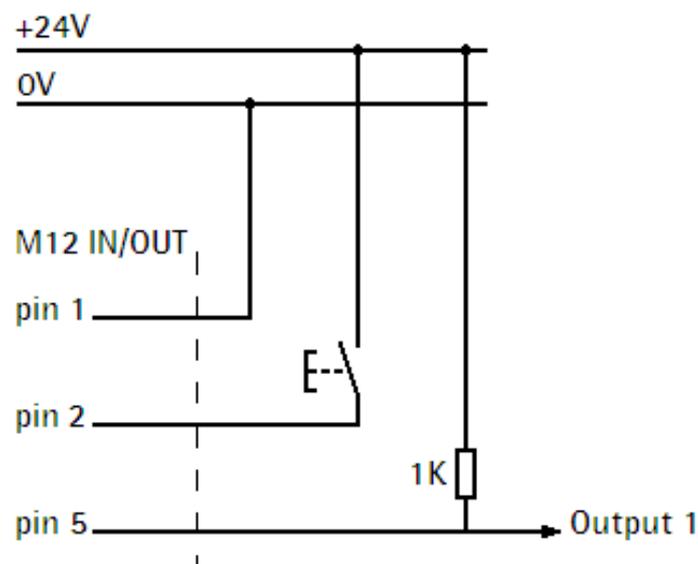
Inputs are read by the Slave device and transmitted to the Master through the **Status word** (bits 13-15; see on page 53 -CANopen interface- or page 134 -Modbus interface) when the device is running in **Operational** state -CANopen interface- / **Idle** state -Modbus interface.

"High" logic value is read when voltage is equal to $+24\text{VDC} \pm 10\%$.

The Slave output 1 is operated by the Master through the **Control word** (bit 13; see on page 50 -CANopen interface- or page 127 -Modbus interface) when the device is running in **Operational** state -CANopen interface- / **Idle** state -Modbus interface.

It is an "open collector" type output having $I_{\text{max}} = 150\text{mA}$.

Example of connection scheme:



6.4 Distance per revolution, Jog speed, Work speed, Preset and limit switch values

Variables **Distance per revolution**, **Jog speed**, **Work speed** and **Preset**, as well as the limit switch values (objects **310F Delta space** sub 1 **Positive Delta** and **310F Delta space** sub 2 **Negative Delta** -CANopen interface- and registers **Positive delta [0x08-0x09]** and **Negative delta [0x0A-0x0B]** -Modbus interface) are closely related, hence you have to be very attentive every time you need to change the value in any of them.

Should that be necessary, you have to operate in compliance with the following procedure:

- set a proper value next to the **Distance per revolution** item (see on page 71 -CANopen interface- or page 116 -Modbus interface);
- set a proper value next to the **Jog speed** item (see on page 69 -CANopen interface- or page 120 -Modbus interface);
- set a proper value next to the **Work speed** item (see on page 70 -CANopen interface- or page 121 -Modbus interface);
- set a proper value next to the **Preset** item (see on page 72 -CANopen interface- or page 123 -Modbus interface);
- check the value of the positive limit switch (see object **310F Delta space** sub 1 **Positive Delta** on page 68 -CANopen interface; **Positive delta [0x08-0x09]** item on page 118 -Modbus interface);
- check the value of the negative limit switch (see object **310F Delta space** sub 2 **Negative Delta** on page 68 -CANopen interface; **Negative delta [0x0A-0x0B]** item on page 119 -Modbus interface);
- save the new configuration values using the object **1010-01 Save Parameters** 1010h sub 1 (see on page 59 -CANopen interface) or bit 9 **Save parameters** in the **Control Word [0x2A]** register; see on page 127 -Modbus interface).



WARNING

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

$$\frac{\text{Min. speed} * \text{Distance per revolution}}{1024} \leq \text{Speed} \leq \frac{\text{Max. speed} * \text{Distance per revolution}}{1024}$$

where:

- **Distance per revolution**: this is the new value you want to set next to the **Distance per revolution** item, expressed in pulses
- Min. speed: minimum speed 1 [PPS] for all RD1xA units
- Max. speed: maximum speed 4266 [PPS] for RD1xA-...-T12-... model
2133 [PPS] for RD1xA-...-T24-... model
1066 [PPS] for RD1xA-...-T48-... model
556 [PPS] for RD1xA-...-T92-... model
- **1024**: this is the maximum value you can set next to the **Distance per revolution** item (expressed in pulses).

Each time you change the value in **Distance per revolution** you must then update the value in **Preset** in order to define the zero of the axis as the system reference has now changed.

After having changed the parameter in **Preset** 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 objects **310F Delta space** sub 1 **Positive Delta** and **310F Delta space** sub 2 **Negative Delta** -CANopen interface- or in registers **Positive delta [0x08-0x09]** and **Negative delta [0x0A-0x0B]** -Modbus interface.

The number of revolutions managed by the system is 511 in negative direction and 511 in positive direction assuming **Preset** as reference.

The value of the positive limit (as set next to the object **310F Delta space** sub 1 **Positive Delta** -CANopen interface- or the **Positive delta [0x08-0x09]** item -Modbus interface) plus the value set in parameter **Preset** is the maximum forward travel (positive travel) starting from the preset (value is expressed in pulses).

The value of the negative limit (as set next to the object **310F Delta space** sub 2 **Negative Delta** -CANopen interface- or the **Negative delta [0x0A-0x0B]** item -Modbus interface) subtracted from value set in parameter **Preset** 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** parameter; hence when you change the value in **Distance per revolution** also the value expressed by each one is necessarily redefined. They are: **Acceleration**, **Deceleration**, **Max following error**, **Position window**, **310F Delta space** (CANopen interface) and **Positive delta [0x08-0x09]** / **Negative delta [0x0A-0x0B]** (Modbus interface), **Max speed**, **3343-00 Positive absolute limit switch** (CANopen interface only), **3344-00 Negative absolute limit switch** (CANopen interface only), **Current position**,

Current velocity, **Target position** and **Target speed**. See for instance the relationship between **Distance per revolution** and the speed values, explained in this paragraph.



Example 1

Default values:

Distance per revolution = 1024 steps per revolution

Max. **Work speed**:

- = 4266 pulses per second for RD1xA-...T12-... model ($4266 \cdot 1024 / 1024 = 4266$)
- = 2133 pulses per second for RD1xA-...T24-... model ($2133 \cdot 1024 / 1024 = 2133$)
- = 1066 pulses per second for RD1xA-...T48-... model ($1066 \cdot 1024 / 1024 = 1066$)
- = 556 pulses per second for RD1xA-...T92-... model ($556 \cdot 1024 / 1024 = 556$)

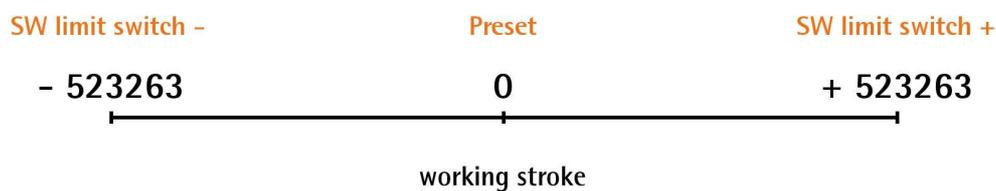
Preset = 0

Positive limit (object **310F Delta space** sub 1 **Positive Delta** -CANopen interface- or **Positive delta [0x08-0x09]** item -Modbus interface) and negative limit (object **310F Delta space** sub 2 **Negative Delta** -CANopen interface- or **Negative delta [0x0A-0x0B]** item -Modbus interface) max. values = $523263 = (1024 \text{ steps per revolution} \times 511 \text{ revolutions}) - 1$ when **Preset** = 0

Max. **SW limit switch +** = $0 + 523263 = + 523263$ pulses (forward travel)

Max. **SW limit switch -** = $0 - 523263 = - 523263$ pulses (backward travel)

Therefore, when **Preset** = 0, the working stroke of the axis will span the maximum positive and negative limits range, that is max. **SW limit switch +** = $+ 523263$ and max. **SW limit switch -** = $- 523263$.





Example 2

ROTADrive RD1xA-...T12-... 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 = 100 steps per revolution

Max. **Work speed** = 417 pulses per second ($4266 \cdot 100 / 1024 = 417$, rounded off to the nearest integer)

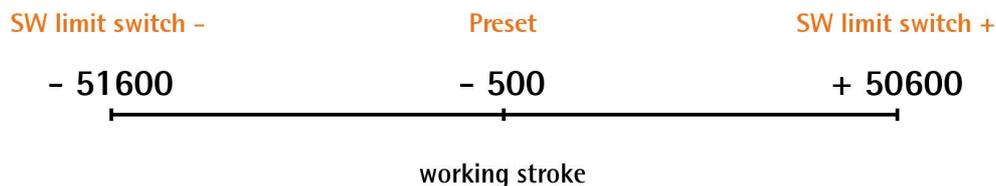
Preset = -500 (ex. thickness of the tool)

Positive limit (object **310F Delta space** sub 1 **Positive Delta** -CANopen interface- or **Positive delta [0x08-0x09]** item -Modbus interface) and negative limit (object **310F Delta space** sub 2 **Negative Delta** -CANopen interface- or **Negative delta [0x0A-0x0B]** item -Modbus interface) max. values = 100 steps per revolution x 511 revolutions = 51100 pulses

Max. **SW limit switch +** = (-500) + 51100 = 50600 pulses (forward travel)

Max. **SW limit switch -** = (-500) - 51100 = -51600 pulses (backward travel)

Therefore, when **Preset** = - 500, the working stroke of the axis will span the maximum positive and negative limits range, that is max. **SW limit switch +** = + 50600 and max. **SW limit switch -** = - 51600.



7 Default parameters list

Parameters list	CANopen object	Modbus register address	Default value
Distance per revolution PPR	3120-00	0x00	1024
Position window P	310D-00	0x01	0
Position window time ms	310E-00	0x02	0
Max following error P	310C-00	0x03	1024
Kp position loop	3111-00	0x04	400
Ki position loop	3112-00	0x05	100
Acceleration PPS ²	3108-00	0x06	500 (RD1xA-...T92-...) 1000 (RD1xA-...T48-...) 2500 (RD1xA-...T24-...) 5000 (RD1xA-...T12-...)
Deceleration PPS ²	3109-00	0x07	500 (RD1xA-...T92-...) 1000 (RD1xA-...T48-...) 2500 (RD1xA-...T24-...) 5000 (RD1xA-...T12-...)
Positive delta P	310F-01	0x08-0x09	523263
Negative delta P	310F-02	0x0A-0x0B	523263
Jog speed PPS	3114-00	0x0C	556 (RD1xA-...T92-...) 1066 (RD1xA-...T48-...) 2133 (RD1xA-...T24-...) 4266 (RD1xA-...T12-...)
Work speed PPS	3115-00	0x0D	556 (RD1xA-...T92-...) 1066 (RD1xA-...T48-...) 2133 (RD1xA-...T24-...) 4266 (RD1xA-...T12-...)
Start torque current time ms	3118-00	0x0E	2000
Code sequence	3302-00	0x0F	0
Kp current loop	3330-00	0x10	200
Ki current loop	3331-00	0x11	30
Max current mA	3340-00	0x12	2000
Starting torque current mA	3341-00	0x13	4000
Preset P	3300-00	0x16-0x17	0
Cyclic Time ms	3110-00	-	0
Gear ratio	3342-00	0x18	92 (RD1xA-...T92-...) 48 (RD1xA-...T48-...) 24 (RD1xA-...T24-...) 12 (RD1xA-...T12-...)
Jog step length P	3117-00	0x19	100

CANopen[®] interface

RD1A

RD12A

CANopen



Smart encoders & actuators

1 CANopen® interface

Lika ROTADRIVE positioning units are Slave devices and are designed in compliance with the "Application Layer and Communication Profile DS301". For any further information or omitted specifications please refer to "CiA Draft Standard 301" document available at www.can-cia.org.

1.1 EDS files

CANopen® devices are supplied with their own EDS file **LIKA_RD12A_Tx_CB_Vx.EDS** (see enclosed documentation or click www.lika.biz > **ROTARY ACTUATORS** > **ROTARY ACTUATORS (DRIVECOD)** > **RD1A / RD12A**). EDS file has to be installed on the CANopen® Master device.

Install the file **LIKA_RD12A_T12_CB_Vx.EDS** for devices fitted with T12 reduction gear.

Install the file **LIKA_RD12A_T24_CB_Vx.EDS** for devices fitted with T24 reduction gear.

Install the file **LIKA_RD12A_T48_CB_Vx.EDS** for devices fitted with T48 reduction gear.

Install the file **LIKA_RD12A_T92_CB_Vx.EDS** for devices fitted with T92 reduction gear.

EDS files are available in both English version (_en) and Italian version (_it).



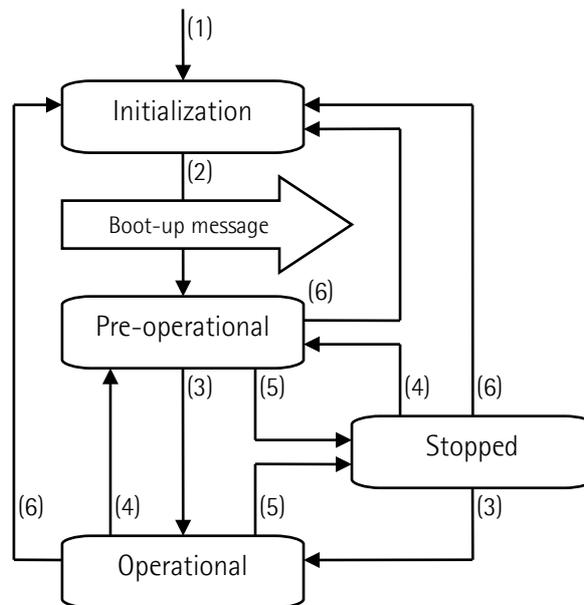
WARNING

Please be aware that the following compatibilities between a release of the EDS file and the release of the Modbus executable file have to be respected compulsorily.

Compatibility	EDS file	Modbus EXE
	V1	up to 2.1
	V2	up to 2.4
	V3	from 2.5 up to ...

1.2 NMT states

CANopen® devices are designed to operate using different states. Transition from one state to another is achieved by sending specific NMT messages (see Figure below).



(1)	Power turned on
(2)	Initialization carried out, boot-up message is sent automatically
(3)	NMT message: Start remote node
(4)	NMT message: Enter pre-operational
(5)	NMT message: Stop remote node
(6)	NMT message: Reset node or Reset communication

1.2.1 Initialization

This is the first state the CANopen® device enters after power is turned on or after performing a hardware reset by means of a **Reset node** command. During initialization, device reads and loads the parameters saved on EPROM. As soon as the basic CANopen® device initialization is carried out, device sends a boot-up message and then switches automatically to **Pre-operational** state.

1.2.2 Pre-operational state

In this state the communication between the Master and the Slave is possible using SDO messages. They allow the working parameters to be set. The Slave cannot send PDO messages.

To switch the Slave device to **Operational** state the Master must send a **Start remote node** command using a NMT message.

1.2.3 Operational state

In this state the Slave device is active and all communication objects are available. The Slave device can use the parameters available in the "Object dictionary" and is allowed to send process data using PDO messages. "Object dictionary" can be accessed using SDO messages. To switch the Slave device to **Pre-operational** state the Master must send an **Enter pre-operational** command using a NMT message.



WARNING

For safety reasons, in **Operational** state the Master must check the Slave device continuously and in a proper way. For a description of the correct procedure see section "1.6 PDO messages" on page 47.

1.2.4 Stopped state

In this state the Slave device is forced to interrupt the communication with the Master altogether (except node guarding, if active).

Communication using PDO and SDO messages is not allowed.

To switch the Slave device to **Pre-Operational** or **Operational** state the Master must send the specific commands **Enter pre-operational** or **Start remote node** using a NMT message.

1.3 Communication messages

Four different kinds of communication messages are used in a CANopen® network:

- Network management **NMT** service: through node control services, the NMT Master controls the NMT state of the NMT Slaves; see section "1.4 NMT messages" on page 46.
- **Process Data Object PDO** service: the real-time data transfer is performed by means of "Process Data Objects (PDO)". The PDOs correspond to entries in the object dictionary and provide the interface to the application objects; see section "1.6 PDO messages" on page 47.
- **Service Data Object SDO** service: SDOs used to provide direct access to entries of a CANopen® device "Object dictionary" (page 57); they allow to read and set parameters; see section "1.7 SDO messages" on page 56.
- **Special Function Object** services:
 - SYNC: synchronization object provides the basic network synchronization mechanism and is used by the Master to enable the Slave devices to transmit process data (position and velocity; see on page 58);
 - Emergency EMCY: emergency objects are triggered by the occurrence of a CANopen® device internal error situation, see on page 79;
 - Node guarding protocol: it is used to detect remote errors / faults in the network, see on page 80.

Relation between device states and communication objects:

	Initialization	Pre-operat.	Operational	Stopped
NMT		X	X	X
PDO			X	
SDO		X	X	
SYNC			X	
EMCY		X	X	
Boot-up	X			
Nodeguard		X	X	X

1.3.1 Generic pre-defined connection set

Broadcast objects of the generic pre-defined connection set		
Type of COB (Object)	Function code (binary)	COB-ID (hex)
NMT	0000	000
SYNC	0001	080

Peer-to-peer objects of the generic pre-defined connection set		
EMERGENCY	0001	081 - 0FF
PDO 1 (tx)	0011	181 - 1FF
PDO 1 (rx)	0100	200 - 27F
SDO (tx)	1011	581 - 5FF
SDO (rx)	1100	601 - 67F
Nodeguard	1110	701 - 77F
Boot-up	1110	701 - 77F

The type of COB (tx means COB-ID sent; rx means COB-ID received) is viewed from the Slave device.

1.4 NMT messages

Structure of NMT messages:

COB-ID (11 bit)		2 CAN Data Bytes	
Func.code	Node ID	Command	Slave ID
0000	0	NMT function	Slave ID

If Slave ID = 00h, the NMT message is issued to all network nodes.

Command	NMT service	Node state
01 hex	Start remote node	Operational
02 hex	Stop remote node	Stopped
80 hex	Enter pre-operational	Pre-operational
81 hex	Reset node	Pre-operational
82 hex	Reset communication	Pre-operational

1.5 Boot-up messages

Structure of Boot-up messages:

COB-ID(hex)	1 CAN Data Byte
700+Node ID	00

1.6 PDO messages



WARNING

For safety reasons, when the 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 monitoring the communication state in the network, among the possible methods the Node guarding protocol can be implemented (complying with DS301 specifications, see on page 80).

ROTADRIVE positioning unit sends PDO messages to the Master according to the set cyclic or synchronous work mode (see object **1800 Transmit PDO Communication Parameter 1**).



PDO messages have always a 8 CAN Data Bytes format. Please note that the structure of sent messages and received messages is different anyway.

Structure of **RECEIVE PDO1** messages received by the Slave device (sent by the Master):

IDENTIFIER		8 CAN data byte							
COB-ID (hex)		0	1	2	3	4	5	6	7
F.C.	Node-ID	Control Word				Target position			
200+ Node ID		Low	High	Low	High

Structure of **TRANSMIT PDO1** messages sent by the Slave device (received by the Master):

IDENTIFIER		8 CAN data byte							
COB-ID (hex)		0	1	2	3	4	5	6	7
F.C.	Node-ID	Status word		Current velocity		Current position			
180+ Node ID		Low	High	Low	High	Low	High

Structure of bytes:

bit	7	6	5	4	3	2	1	0
	M.S.bit							L.S.bit

1.6.1 "RECEIVE PDO1" message sent by the Master to the Slave

Control Word

Index 0x3100-00. 32 bits. It contains the commands to be sent in real time to the Slave in order to manage it. See also **3100-00 Control word** on page 75.

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 **3117-00 Jog step length** item; then the slave stops and waits for another issue. Velocity, acceleration and deceleration are set in objects **3114-00 Jog speed**, **3108-00 Acceleration** and **3109-00 Deceleration** respectively. For a detailed description of the jog control see on page 34.

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 **3117-00 Jog step length** item; then the slave stops and waits for another issue. Velocity, acceleration and

deceleration are set in objects **3114-00 Jog speed**, **3108-00 Acceleration** and **3109-00 Deceleration** respectively. For a detailed description of the jog control see on page 34.

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 **3109-00 Deceleration**. For an immediate halt in the movement, use bit 7 **Emergency**.

Alarm reset

bit 3

Setting this bit to "1" causes the normal work status of the device to be restored. This command is used to reset an alarm condition of the Slave but only if the faulty condition has ceased. Using SDO messages you can read further information about the alarm in the index **1003 Pre-defined error field**.

Please note that should the alarm be caused by wrong object values (**Machine data not valid**) 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 **3117-00 Jog step length** item; then the slave stops and waits for another issue.

Please note that when you use the manual buttons (see "4.4.4 JOG + and JOG - buttons (Figure 5)" on page 30) the "incremental jog" function is disabled; that is, jog step movements are not allowed using the manual buttons.



bit 5

Not used.

Start

bit 6

If set to "1" the device moves in order to reach the set target position. For a complete description of the position control see on page 33. For any information on target position see **Target position** on page 51.

Emergency

bit 7 This bit has to be normally high ("=1"), otherwise it will cause the device to stop immediately. For a normal halt (not immediate) which respects the set deceleration see above bit 2 **Stop**.

Byte 1

bit 8 ... 11 Not used.

Axis torque

bit 12 When the axis has reached the commanded position, it keeps the torque. This function is available only in RD1A version (model without brake); in the RD12A 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.

OUT 1

bit 13 This is intended to activate / deactivate the operation of the digital output 1. The meaning of the available outputs is described in section "CANopen® programming parameters" on page 57.
OUT 1 = 0 output 1 low (not active)
OUT 1 = 1 output 1 high (active)

Brake released

bit 14 This function is available only in RD12A version (model fitted with brake); in the RD1A version (model without brake) bit 14 is not used. RD12A 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.

Bytes 2 and 3 Not used.

Bytes 4 ... 7

Target position

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 while the device is still reaching it; to do this, send a **Start** command and the new target value in **Target position**.



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 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.

1.6.2 "TRANSMIT PDO1" message sent by the Slave to the Master

Status word

Index 0x3101-00. 16 bits. In these bytes the status of the PI controller in **Operational** work mode is shown. See also **3101-00 Status word** on page 75.

Byte 0

Axis in position

bit 0 If value is "=1" device has reached the set position for the time set in **310E-00 Position window time**. It is kept active until the position error is lower than **310D-00 Position window**.

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 object **310F Delta space** sub 1 **Positive delta**.

SW limit switch -

bit 4 If value is "=1" device has reached the maximum negative limit (negative limit switch). See object **310F Delta space** sub 2 **Negative delta**.

Alarm

bit 5 If value is "=1" an alarm has occurred, see details at indexes **1003 Pre-defined error field** and **3200-00 Alarms**.

Axis running

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

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** item. Bit is kept active until new **Target position** value or **Alarm reset** commands are sent.

Button 1 Jog +

bit 9

RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. Once you press the button 1 JOG +, bit 9 is forced high "1"; when the button 1 is not pressed, bit 9 is low "0". For further information see section "4.4 Dip-Switches and buttons (Figure 5)" on page 27.

Button 2 Jog -

bit 10

RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. Once you press the button 2 JOG -, bit 10 is forced high "1"; when the button 2 is not pressed, bit 10 is low "0". For further information see section "4.4 Dip-Switches and buttons (Figure 5)" on page 27.

Button 3 Preset

bit 11

RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. Once you press the button 3 PRESET, bit 11 is forced high "1"; when the button 3 is not pressed, bit 11 is low "0". For further information see section "4.4 Dip-Switches and buttons (Figure 5)" on page 27.

DAC saturation

bit 12

The current supplied by power electronic for controlling the motor has reached the maximum value and cannot be increased further.

IN 1

bit 13

This is meant to show the status of the digital input 1. The meaning of the available inputs is described in section "CANopen® programming parameters" on page 57.

IN 1 = 0 input 1 low (not active)

IN 1 = 1 input 1 high (active)

IN 2

bit 14

This is meant to show the status of the digital input 2. The meaning of the available inputs is described in section "CANopen® programming parameters" on page 57.

IN 2 = 0 input 2 low (not active)

IN 2 = 1 input 2 high (active)

IN 3

bit 15

This is meant to show the status of the digital input 3. The meaning of the available inputs is described in section "CANopen® programming parameters" on page 57.

IN 3 = 0 input 3 low (not active)

IN 3 = 1 input 3 high (active)

Bytes 2 and 3

Current velocity

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

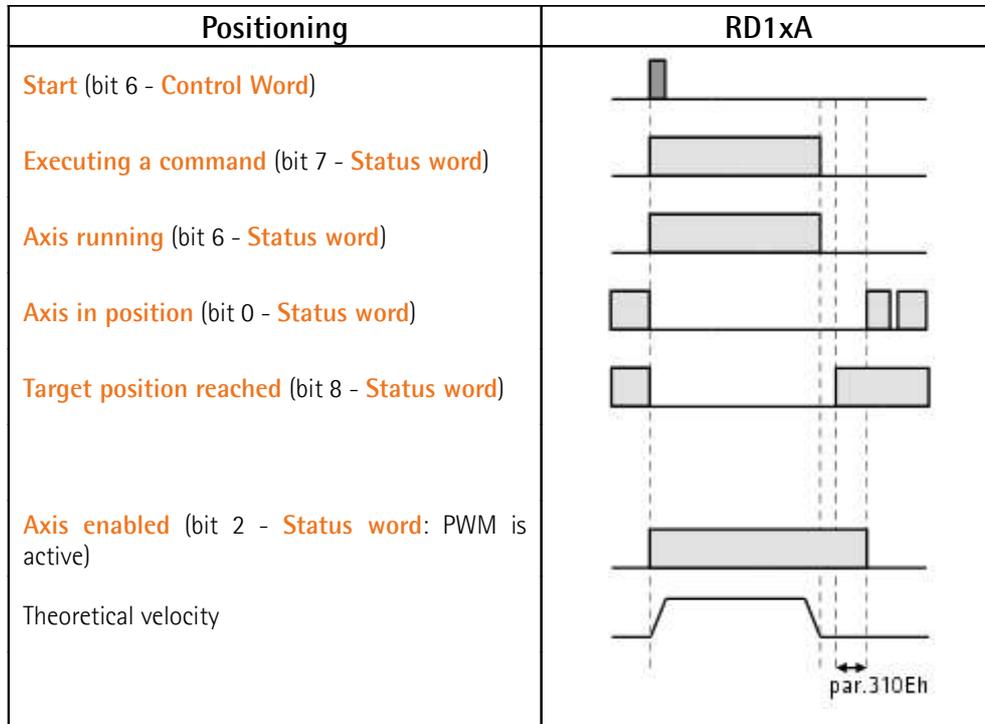
Bytes 4 ... 7

Current position

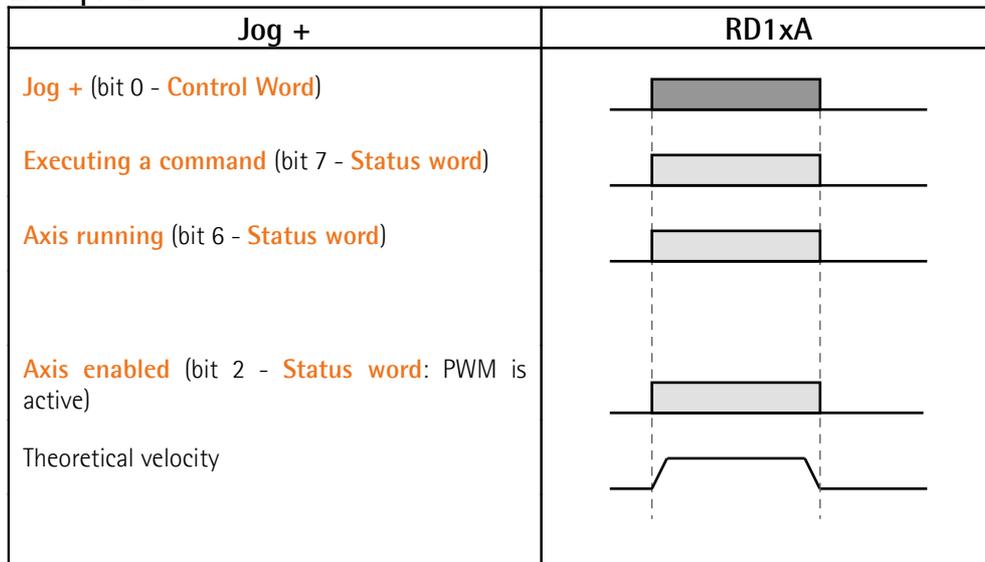
Absolute position of the device when the PDO message is sent.



Example 1



Example 2



1.7 SDO messages

SDO messages are used to set new values to and read values from the Slave. These parameters are described in the "Objects dictionary" (see on page 57). 4 bytes at the most are used for CAN data, other 4 bytes are used for Command, Index and Sub-index fields. SDO messages always need confirmation from the Slave. It follows that when the Master sends a SDO message to the Slave, the Slave always sends a reply (and a warning, should an error occur).

SDO structure:

IDENTIFIER		from 4 to 8 CAN data bytes							
COB-ID(hex)		0	1	2	3	4	5	6	7
F.C.	Node-ID	Com	Index		Sub	Data			
		1 byte	LSB	MSB	1 byte	LSB	MSB

Com command
Index parameter index
Sub parameter sub-index
Data parameter value (set or read value)

1.7.1 Command

Command byte contains the type of COB telegram transmitted to the CAN network.

Three types of telegram are available:

- **Set**: used to send configuration parameters to the Slave;
- **Req**: used by the Master to read data from the Slave;
- **Warnings**: used by the Slave to send error messages to the Master (e.g. "Index does not exist", "Process data value not valid", etc.).

Command	COB	COB type	Data length
22h	Set	M → S request	not specif.
23h	Set	M → S request	4 byte
2Bh	Set	M → S request	2 byte
2Fh	Set	M → S request	1 byte
60h	Set	S → M confirmation	0 byte
40h	Req	M → S request	0 byte
42h	Req	S → M reply	non specified
43h	Req	S → M reply	4 byte
4Bh	Req	S → M reply	2 byte
4Fh	Req	S → M reply	1 byte
41h	Req	S → M reply	segmented SDO
80h	Warning	S → M reply	4 byte

2 CANopen® programming parameters

2.1 Objects dictionary

In the following pages the objects implemented are listed and described as follows:

Index-subindex Object name

[data types, attribute]

- Index and subindex are expressed in hexadecimal notation.
- Attribute:
 - ro = read only access
 - rw = read and write access
 - const = ro + constant value

Unsigned16 data type:

Data byte	
byte 4	byte 5
LSByte	MSByte

Unsigned32 data type:

Data byte			
byte 4	byte 5	byte 6	byte 7
LSByte	MSByte

2.1.1 Standard objects (DS 301)

1000-00 Device type

[Unsigned32, const]

Default = 0000012Dh

1001-00 Error register

[Unsigned8, ro]

Should an error occur, bit 0 of this object will be set to "1".

Default = 00h

1003 Pre-defined error field

This object contains the list of the five previous errors which generated an EMCY emergency message.

- **00** Number of current errors [Unsigned8, rw]
 Enter 00h to delete the errors list.
- **01** Last error occurred [Unsigned32, ro]
- **02...05** Previous errors occurred [Unsigned32, ro]

1005-00 COB-ID SYNC messages

[Unsigned32, rw]

Default = 0000 0080h

1008-00 Device name

[String, const]

It displays the name of the device.

Default = "RD1A" or "RD12A"

1009-00 Hardware version

[String, const]

It shows the hardware version of the device.

100A-00 Software version

[String, const]

It shows the software version of the device.

100C-00 Guard Time

[Unsigned16, rw]

It contains the "node guard time" value expressed in msec (milliseconds). The Master polls each Slave at regular time intervals issuing RTR messages. This time-interval is called the "node guard time" and may be different for each Slave. For further details see section "2.4 Node guarding protocol" on page 80.

Default = 0

100D-00 Life Time Factor

[Unsigned8, rw]

100C-00 Guard Time and **100D-00 Life Time Factor** objects are used in the "Node guarding protocol" controlled by Master. For further details see section "2.4 Node guarding protocol" on page 80.

Default = 0

1010-01 Save Parameters

[Unsigned32, rw]

Use this object to save all parameters on non-volatile memory.

Write **save** (ASCII code in hexadecimal form) in the data bytes:

Master → Slave

COB-ID	Cmd	Index	Sub	Data bytes				
600+ID	23	10	10	01	73	61	76	65

Slave → Master (confirmation)

COB-ID	Cmd	Index	Sub	Data bytes				
580+ID	60	10	10	01	00	00	00	00

1011-01 Restore Default Parameters

[Unsigned32, rw]

This object allows you to restore all parameters to default values (default values are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode).

Write **load** (ASCII code in hexadecimal form) in the data bytes and then issue a **Reset node** command:

Master → Slave

COB-ID	Cmd	Index	Sub	Data bytes				
600+ID	23	11	10	01	6C	6F	61	64

Slave → Master (confirmation)

COB-ID	Cmd	Index	Sub	Data bytes				
580+ID	60	11	10	01	00	00	00	00

Master → Slave (reset node)

COB-ID	Cmd	Slave ID
000	81	ID

Slave → Master (Boot-up)

COB-ID	Cmd
700+ID	00



NOTE

Save default values using **1010-01 Save Parameters** function.

1014-00 COB-ID EMCY

[Unsigned32, rw]

This object defines the COB-ID used to send emergency messages (EMCY).

When the power is turned on, this object always contains the default value.

For a complete list of the emergency messages refer to section "2.3 Emergency messages" on page 79.

Default = 80h+Node ID

1015-00 Inhibit Time Emergency

[Unsigned16, rw]

Inhibit time of emergency messages (EMCY), i.e. minimum interval between subsequent emergency messages expressed in multiples of 100 µs.

Default = 0

1018 Identity object

- **01 Vendor Id** [Unsigned32, ro]
- **02 Product code** [Unsigned32, ro]
- **03 Revision number** [Unsigned32, ro]
- **04 Serial number** [Unsigned32, ro]

1400 Receive PDO Communication Parameter 1

This object contains the communication parameters for the PDOs the device is able to receive (Receive PDOs).

- **01 COB-ID** of PDO1 [Unsigned32, rw]

Bit number	Value	Meaning
31 (MSB)	0	PDO exists / is valid
	1	PDO does not exist / is not valid
30	0	RTR allowed on this PDO (not implemented)
	1	no RTR allowed on this PDO
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 ... 11	0	if bit 29 = 0
	X	if bit 29 = 1: bits 28-11 of 29-bit-COB-ID
10 ... 0 (LSB)	X	bits 10-0 of COB-ID

Default = 4000 0200h+Node ID (no RTR, COB-ID)



WARNING

It is mandatory to set the bit 30 of COB-ID to 1 (value 0 is not allowed). This means that "No RTR is allowed on the PDO".

At power on, this object always contains the default value.

- **02 Transmission Type** [Unsigned8, rw]

Transmission type	PDO transmission	
00h (0)	Acyclic, synchronous	not implemented
01h ... F0h (1 ... 240)	Cyclic, synchronous	implemented
F1h ... FBh (241 ... 251)	not implemented - reserved	
FCh (252)	Synchronous, RTR only	not implemented
FDh (253)	Asynchronous, RTR only	not implemented
FEh (254)	Asynchronous, manufacturer specific	implemented
FFh (255)	Asynchronous, device profile specific	not implemented

Default = FEh (event-driven, asynchronous, manufacturer specific)



WARNING

Following an attempt to set the **Transmission Type** to 0, the PDO message is not sent; following an attempt to change the **Transmission Type** to any other value that is not supported by the device, an abort message (abort code = 0609 0030h: **Value range of parameter exceeded**) is generated.



NOTE

- Before altering the value of **COB-ID** it is compulsory to deactivate the receipt of PDO1, then enter the new COB-ID value, finally activate again the receipt of PDO1.
- PDO1 receipt can be activated (deactivated) by setting to "0" ("1") the most significant bit of the object **1400 Receive PDO Communication Parameter 1** sub 1 **COB-ID**.

1600 Receive PDO Mapping Parameter 1

This object indicates the kind of parameter contained in the Receive PDO message.

- **01 3100-00 Control word** [Unsigned32, ro] Default = 3100 0020h.
- **02 3106-00 Target position** [Unsigned32, ro] Default = 3106 0020h.

1800 Transmit PDO Communication Parameter 1

These objects contain the communication parameters of Transmit PDO.

- **01 COB-ID** of PDO1 [Unsigned32, rw]

Bit number	Value	Meaning
31 (MSB)	0	PDO exists / is valid
	1	PDO does not exist / is not valid
30	0	RTR allowed on this PDO (not implemented)
	1	no RTR allowed on this PDO
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 ... 11	0	if bit 29 = 0
	X	if bit 29 = 1: bits 28-11 of 29-bit-COB-ID
10 ... 0 (LSB)	X	bits 10-0 of COB-ID

Default = 4000 0180h+Node ID (no RTR, COB-ID)



WARNING

It is mandatory to set the bit 30 of COB-ID to 1 (value 0 is not allowed). This means that "No RTR is allowed on the PDO". At power on, this object always contains the default value.

- 02 **Transmission Type** [Unsigned8, rw]

Transmission type	PDO transmission	
00h (0)	Acyclic, synchronous	not implemented
01h ... F0h (1 ... 240)	Cyclic, synchronous	implemented
F1h ... FBh (241 ... 251)	not implemented - reserved	
FCh (252)	Synchronous, RTR only	not implemented
FDh (253)	Asynchronous, RTR only	not implemented
FEh (254)	Asynchronous, manufacturer specific	implemented
FFh (255)	Asynchronous, device profile specific	not implemented

Default = FEh ("status, position, velocity exchange" transmission or cyclic transmission, see hereafter and object **3110-00 Cyclic Time**)



WARNING

Following an attempt to set the **Transmission Type** to 0, the PDO message is not sent; following an attempt to change the **Transmission Type** to any other value that is not supported by the device, an abort message (abort code = 0609 0030h: **Value range of parameter exceeded**) is generated.

If the value next to the object **3110-00 Cyclic Time** \neq 0, the PDO message is sent cyclically and the interval between two messages is the time set next to the object **3110-00 Cyclic Time**; otherwise, if the value next to the object **3110-00 Cyclic Time** = 0, the PDO message is sent only at each status exchange (see object **3101-00 Status word**) and/or position exchange (see object **3103-00 Current position value**) and/or velocity exchange (see object **3105-00 Current velocity value**); in this case the minimum interval between two PDO transmissions is set next to the following parameter **Inhibit Time** object 1800h sub 3.

- **03 Inhibit Time** [Unsigned16, rw]
Minimum interval between two PDO transmissions when "status, position, velocity exchange" transmission is set (see the parameter **Transmission Time** object 1800h sub 2 and the NOTE hereinafter); value is expressed in multiples of 100 µs.
Default = 0190h (40ms)



NOTE

- Before altering the value of **COB-ID** it is compulsory to deactivate the transmission of PDO1, then enter the new COB-ID value, finally activate again the transmission of PDO1.
- PDO1 transmission can be activated (deactivated) by setting to "0" ("1") the most significant bit of the object **1800 Transmit PDO Communication Parameter 1** sub 1 **COB-ID**.
- Cyclic transmission or synchronous transmission can be modified by setting a suitable value in the object **1800 Transmit PDO Communication Parameter 1** sub 2 **Transmission Type**. To have a PDO1 transmission every "n" SYNC commands, set "n" value in the object 1800h, sub 2.
01h = synchronous transmission at every SYNC command
02h = synchronous transmission every two SYNC commands
...
FEh = "cyclic transmission" or "status, position, velocity exchange transmission":
if **3110-00 Cyclic Time** ≠ 0 → "cyclic transmission": cycle time is set in object 3110h;
if **3110-00 Cyclic Time** = 0 → "status, position, velocity exchange" transmission: device issues a PDO message each time parameters mapped in PDO change (see object **1A00 TPDO Mapping Parameter 1**) with a minimum interval not lower than **1800 Transmit PDO Communication Parameter 1** sub 3 **Inhibit Time**.

1A00 TPDO Mapping Parameter 1

This object indicates the kind of parameter contained in the Transmit PDO message.

- **01 3101-00 Status word** [Unsigned32, ro] Default = 3101 0010h.
- **02 3105-00 Current velocity value** [Unsigned32, ro] Default = 3105 0010h.
- **03 3103-00 Current position value** [Unsigned32, ro] Default = 3103 0020h.



NOTE

Save set values using **1010-01 Save Parameters** function.

Should the power be turned off or **Reset node** or **Reset communication** commands be sent all data not saved will be lost!

2.1.2 Manufacturer's specific objects

Communication parameters

3000-00 Baud rate

[Unsigned8, ro]

This object is meant to show the baud rate (transmission rate) set by means of the dedicated switch, according to the following table; for any information on setting the baud rate see section "4.4.2 Setting data transmission rate: Baud rate (Figure 5)" on page 29.

Switch 1 ... 4	Baud rate
0000	20 Kbit/s
1000	50 Kbit/s
0100	100 Kbit/s
1100	125 Kbit/s
0010	250 Kbit/s
1010 (default)	500 Kbit/s
0110	800 Kbit/s
1110	1000 Kbit/s

3001-00 Node ID

[Unsigned8, ro]

This object is meant to show the node identifier (address) of the device set by means of the dedicated switch; for any information on setting the node-ID see section "4.4.1 Setting the node address: Node ID (Figure 5)" on page 28. The default address is 1.

Device profile objects

3108-00 Acceleration

[Unsigned32, rw]

This object defines the acceleration value that has to be used by the device. Parameter is expressed in pulses per second² [PPS²].

Default = 5000 for RD1xA-...-T12-... model

Default = 2500 for RD1xA-...-T24-... model

Default = 1000 for RD1xA-...-T48-... model

Default = 500 for RD1xA-...-T92-... model

3109-00 Deceleration

[Unsigned32, rw]

This object defines the deceleration value that has to be used by the device. Parameter is expressed in pulses per second² [PPS²].

Default = 5000 for RD1xA-...-T12-... model

Default = 2500 for RD1xA-...-T24-... model

Default = 1000 for RD1xA-...-T48-... model

Default = 500 for RD1xA-...-T92-... model

310C-00 Max following error

[Unsigned32, rw]

This object 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 object, then the alarm **Following error** is triggered and the unit stops. Parameter is expressed in pulses.

Default = 1024

310D-00 Position window

[Unsigned16, rw]

This object defines the tolerance window for the **3106-00 Target position** value. When the axis is within the tolerance window limits for the time set in the object **310E-00 Position window time**, then the state is signalled through the **Axis in position** status bit. Parameter is expressed in pulses.

Default = 0

310E-00 Position window time

[Unsigned16, rw]

It represents the time for which the axis has to be within the tolerance window limits set in the object **310D-00 Position window** before the state is signalled through the **Axis in position** status bit. Parameter is expressed in milliseconds. Default = 0

310F Delta space

- 01 Positive delta** [Integer32, 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.

$$\text{SW limit switch +} = \text{3300-00 Preset} + \text{310F Delta space sub 1 Positive delta.}$$
 Default = 523263
- 02 Negative delta** [Integer32, 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.

$$\text{SW limit switch -} = \text{3300-00 Preset} - \text{310F Delta space sub 2 Negative delta.}$$
 Default = 523263



WARNING

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



EXAMPLE

When **3120-00 Distance per revolution** = 1024 and **3300-00 Preset** = 0, the maximum acceptable value for **310F Delta space** (sub 1 **Positive delta** and sub 2 **Negative delta**) is:

$(1024 \text{ steps per revolution} * 511 \text{ revolutions}) - 1 = 523263$

When **3120-00 Distance per revolution** = 256 and **3300-00 Preset** = 0, the maximum acceptable value for **310F Delta space** (sub 1 **Positive delta** and sub 2 **Negative delta**) is:

$(256 \text{ steps per revolution} * 511 \text{ revolutions}) - 1 = 130815$

See further examples in the paragraph "6.4 Distance per revolution, Jog speed, Work speed, Preset and limit switch values" on page 36.



WARNING

Every time **3120-00 Distance per revolution** and **3300-00 Preset** parameters are changed, **310F Delta space** values has to be checked carefully. Each time you change the value in **3120-00 Distance per revolution** you must then update the value in **3300-00 Preset** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **3300-00 Preset** 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 **310F Delta space**. For a detailed explanation see on page 36.

3111-00 Kp position loop

[Unsigned32, rw]

This object 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 = 400

3112-00 Ki position loop

[Unsigned32, rw]

This object 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 = 100

3114-00 Jog speed

[Unsigned32, rw]

This object contains the maximum speed of the device when using **Jog +** and **Jog -** functions. Parameter is expressed in pulses per second.

Default = 4266 for RD1xA-...-T12-... model

Default = 2133 for RD1xA-...-T24-... model

Default = 1066 for RD1xA-...-T48-... model

Default = 556 for RD1xA-...-T92-... model

3115-00 Work speed

[Unsigned32, rw]

This object 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**). Parameter is expressed in pulses per second.

Default = 4266 for RD1xA-...-T12-... model

Default = 2133 for RD1xA-...-T24-... model

Default = 1066 for RD1xA-...-T48-... model

Default = 556 for RD1xA-...-T92-... model



WARNING

Each time you change the value in **3120-00 Distance per revolution** you must then set new values also in **3114-00 Jog speed** and **3115-00 Work speed** as speed values are expressed in pulses per second (PPS). To calculate the speed values you have always to adhere to the following ratio:

$$\frac{\text{Min. speed} * \text{Distance per revolution}}{1024} \leq \text{Speed} \leq \frac{\text{Max. speed} * \text{Distance per revolution}}{1024}$$

For a detailed explanation see on page 36.

3116-00 Max speed

[Unsigned32, ro]

This object shows the maximum speed you can set in the objects **3114-00 Jog speed** and **3115-00 Work speed**. Parameter is expressed in pulses per second.

Default = 4266 for RD1xA-...-T12-... model

Default = 2133 for RD1xA-...-T24-... model

Default = 1066 for RD1xA-...-T48-... model

Default = 556 for RD1xA-...-T92-... model

3117-00 Jog step length

[Unsigned32, rw]

If the incremental jog function is enabled (bit 4 **Incremental jog** in **Control Word** = 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

3118-00 Start Torque current time

[Unsigned32, rw]

This object defines the maximum time for which the motor is supplied with starting torque current when it starts its movement (see object **3341-00 Starting Torque current**). Parameter is expressed in milliseconds. Maximum value allowed is 5 seconds.

Default = 2000

3120-00 Distance per revolution

[Unsigned32, rw]

This object sets the number of pulses per each complete revolution of the shaft. This parameter is useful to relate the revolution of the shaft and a linear measurement. For example: unit is joined to a worm screw having a 5 mm pitch; by setting **3120-00 Distance per revolution** = 500, at each shaft revolution system performs a 5 mm pitch with one-hundredth of a millimetre resolution. Maximum value is 1024.

Default = 1024



WARNING

After having changed this parameter you must then set new values also in objects **3114-00 Jog speed**, **3115-00 Work speed** and **3300-00 Preset**. For a detailed explanation see on page 36 and relevant parameters.

Please note that the parameters listed hereafter are closely related to the **3120-00 Distance per revolution** parameter; hence when you change the value in **3120-00 Distance per revolution** also the value expressed by each one is necessarily redefined. They are: **3108-00 Acceleration**, **3109-00 Deceleration**, **310C-00 Max following error**, **310D-00 Position window**, **310F Delta space**, **3116-00 Max speed**, **3343-00 Positive absolute limit switch**, **3344-00 Negative absolute limit switch**, **3103-00 Current position value**, **3105-00 Current velocity value**, **3106-00 Target position** and **3107-00 Target speed**. See for instance the relationship between **3120-00 Distance per revolution** and the speed values, explained on page 70.



NOTE

If **3120-00 Distance per revolution** is not a power of 2 (2, ..., 512, 1024), at position control a positioning error could occur having a value equal to one pulse.

3300-00 Preset

[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



WARNING

A new value has to be set in **3300-00 Preset** every time **3120-00 Distance per revolution** value is changed. After having entered a new value in **3300-00 Preset** 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 objects **310F Delta space** sub 1 **Positive delta** and **310F Delta space** sub 2 **Negative delta**. For a detailed explanation see on page 36.

3301-00 Offset

[Integer32, ro]

This object defines the difference between the position value transmitted by the device and the real position: real position – preset. Value is expressed in pulses.

3302-00 Code sequence

[Boolean, rw]

It sets the rotation direction of the encoder shaft and consequently defines whether the position value output by the encoder increases when the encoder 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 **3300-00 Preset** parameter and then check the values set next to the **310F Delta space** item.

3330-00 Kp current loop

[Unsigned32, rw]

This object 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 = 200

3331-00 Ki current loop

[Unsigned32, rw]

This object 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 = 30

3340-00 Max current

[Unsigned32, rw]

This object defines the maximum current supplied by the power electronic to control the motor. Parameter is expressed in mA (milliamperes). This value cannot be greater than the one in the object **3341-00 Starting Torque current**. Maximum value is 2000.

Default = 2000

3341-00 Starting Torque current

[Unsigned32, rw]

This object defines the maximum current supplied to the motor only when it starts its movement and for the maximum time set in the object **3118-00 Start Torque current time**. Parameter is expressed in mA (milliamperes). Maximum value is 4000.

Default = 4000

3342-00 Gear ratio

[Unsigned32, ro]

It displays the gear ratio of the reduction gear installed between the motor and the encoder shaft. This is a read-only parameter.

Default = 12 for RD1xA-...-T12-... model

Default = 24 for RD1xA-...-T24-... model

Default = 48 for RD1xA-...-T48-... model

Default = 92 for RD1xA-...-T92-... model

3343-00 Positive absolute limit switch

[Integer32, ro]

This is the **SW limit switch +** value (maximum positive limit) calculated according to values set in parameters **3300-00 Preset** and **310F Delta space sub 1 Positive delta**. When the maximum forward limit is reached, a signalling is activated through the **SW limit switch +** status bit.

SW limit switch + = 3300-00 Preset + 310F Delta space sub 1 Positive delta.

Value is expressed in encoder pulses.

Default = +523263

3344-00 Negative absolute limit switch

[Integer32, ro]

This is the **SW limit switch -** value (maximum negative limit) calculated according to values set in parameters **3300-00 Preset** and **310F Delta space sub 2 Negative delta**. When the maximum backward limit is reached, a signalling is activated through the **SW limit switch -** status bit.

SW limit switch - = 3300-00 Preset - 310F Delta space sub 2 Negative delta.

Value is expressed in encoder pulses.

Default = -523263



NOTE

Save default values using **1010-01 Save Parameters** function.

Should the power be turned off or **Reset node** or **Reset communication** commands be sent all data not saved will be lost!

Operating parameters

3005-00 Current value [mA]

[Integer16, ro]

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

3006-00 Temperature value

[Integer16, ro]

This object 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.

3100-00 Control word

[Unsigned32, rw]

This object contains the commands to be sent in real time to the Slave in order to manage it. **3100-00 Control word** parameter is used to edit PDO messages received by the Slave (see further details in the section "1.6 PDO messages" on page 47).

3101-00 Status word

[Unsigned16, ro]

This object gives information on the device state. **3101-00 Status word** object is contained in the PDO messages sent by the Slave (see further details in the section "1.6 PDO messages" on page 47).

3102-00 Demanded position value

[Integer32, ro]

This object shows the value of the theoretical position calculated by the device while moving. This value is used by the PI controller to control the motor.

3103-00 Current position value

[Integer32, ro]

This object shows the value of the current position. **3103-00 Current position value** object is contained in the PDO messages sent by the Slave (see further details in the section "1.6 PDO messages" on page 47).

3105-00 Current velocity value

[Integer16, ro]

This object shows the value of the current speed. **3105-00 Current velocity value** object is contained in the PDO messages sent by the Slave (see further details in the section "1.6 PDO messages" on page 47). Parameter is expressed in pulses per second.

3106-00 Target position

[Integer32, rw]

This object defines the target position, otherwise referred to as commanded position. **3106-00 Target position** parameter is used to edit PDO messages received by the Slave (see further details in the section "1.6 PDO messages" on page 47).

3107-00 Target speed

[Integer32, ro]

This object shows the value of the theoretical speed used by the device for generating the position trajectory. Parameter is expressed in pulses per second.

310B-00 Position following error

[Integer32, ro]

This object 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 object **310C-00 Max following error**, then the alarm **Following error** is triggered and the unit stops.

3110-00 Cyclic Time

[Unsigned16, rw]

3110-00 Cyclic Time is used in asynchronous work mode and sets the interval between two PDO transmissions.

If the value next to this object **3110-00 Cyclic Time** $\neq 0$, the PDO message is sent cyclically and the interval between two messages is the time set here; otherwise, if the value next to this object **3110-00 Cyclic Time** = 0, the PDO message is sent only at each status exchange (see object **3101-00 Status word**) and/or position exchange (see object **3103-00 Current position value**) and/or velocity exchange (see object **3105-00 Current velocity value**); in this case the minimum interval between two PDO transmissions is set next to the parameter **Inhibit Time** object 1800h sub 3.

This parameter only concerns the PDO messages issued by the Slave, not by the Master. Parameter is expressed in milliseconds. For further information see object **1800 Transmit PDO Communication Parameter 1** on page 62.

Default = 0

3200-00 Alarms

[Unsigned16, ro]

This object is meant to show the alarms currently active in the device.

Structure of the alarms byte:

byte	Data byte 4			Data byte 5		
bit	7	...	0	15	...	8
			L.S.bit	M.S.bit		

Alarm codes available:

bit 0	0001h	Machine data not valid	One or more parameters are not valid, set proper values to restore the normal work condition.
bit 1	0002h	Flash memory error	Internal error, it cannot be restored.
bit 2		Not used	
bit 3	0008h	Following error	The difference between the real position and the theoretical position is greater than the value set in parameter 310C-00 Max following error ; we suggest reducing the work speed.
bit 4	0010h	Axis not synchronized	Internal error, it cannot be restored.
bit 5	0020h	Target not valid	Target position is over maximum travel limits.
bit 6	0040h	Emergency	Bit 7 Emergency in the Control Word has been forced to low value (0); or alarms are active in the unit.
bit 7	0080h	Overcurrent	The power supply current is exceeding maximum ratings.

bit 8	0100h	Overtemperature	The internal temperature of the device as sensed by a probe is exceeding maximum ratings (see 3006-00 Temperature value).
bit 9		Not used	
bit 10	0400h	Undervoltage	The power supply voltage is under minimum ratings.
bit 11	0800h	CAN Life guard error	Error in the "Node guarding protocol". For further information see section "2.4 Node guarding protocol" on page 80.
bits 12 ... 15		Not used	

To reset a faulty condition use **Alarm reset** bit. Setting this bit to "1" causes the normal work status of the device to be restored. This command is used to reset an alarm condition of the Slave but only if the faulty condition has ceased. Using SDO messages you can read further information about the alarm in the index **1003 Pre-defined error field**.



Please note that should the alarm be caused by wrong object values (see **Machine data not valid**), normal work status can be restored only after having set proper values. **Flash memory error** alarm cannot be reset.



NOTE

Save default values using **1010-01 Save Parameters** function. Should the power be turned off or **Reset node** or **Reset communication** commands be sent all data not saved will be lost!

2.2 Warning messages

For the complete list and the meaning of the warning messages please refer to "SDO abort codes" section in "CiA Draft Standard 301" document available at www.can-cia.org.

2.3 Emergency messages

Emergency (EMCY) objects are issued by the device when an internal error occurs.

Structure of EMCY object:

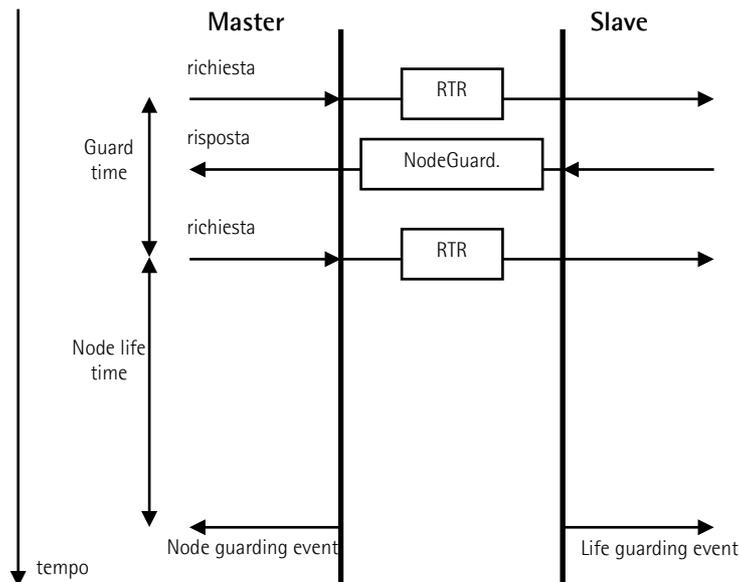
IDENTIFIER	CAN Data Byte			
COB-ID(hex)	0	1	2	3...7
See object 1014h 1014-00 COB-ID EMCY	Error code		Errors sub-register	Specific codes
	LSB	MSB	01	00...00

Emergency codes available:

0000h	No active errors
1000h	Generic error
2220h	Power surge
3110h	Overvoltage
3120h	Undervoltage
4310h	Overtemperature
5530h	Flash memory
8130h	Life Guard
8611h	Following error

2.4 Node guarding protocol

At system boot the "Node guarding protocol" is not active; this protocol is activated automatically as soon as the Master device sends a RTR (Remote Transmit Request) message the first time.



100C-00 Guard Time: interval between two RTR messages (see on page 58).

Node life time: maximum time by which the Slave device must receive the following RTR message issued by the Master device.

"Node life time" = "100C-00 Guard Time" * "100D-00 Life Time Factor".

"Node guarding" is enabled only if "Node life time" \neq 0.

If the Slave device does not receive a RTR message before the "Node life time" has expired, it warns activating a "Life Guarding Event". Furthermore the red LED on the back of the device starts flashing so indicating the "Node guarding error" (see on page 24), objects **1001-00 Error register** and **1003 Pre-defined error field** are updated and an error message is sent.

To reset the error send a **Reset node** command.

3 CANopen® programming examples

Hereafter are some examples of transmission between Master and Slave devices. A generic "ID" value is used to indicate the encoder address; Master address is always 0.

All values are expressed in hexadecimal notation.

Setting the Operational / Pre-operational state

NMT message

Master → Slave

	COB-ID	Cmd	Node
Operational:	000	01	ID
Pre-operational:	000	80	ID

Setting object 3300-00 Preset (preset = 1000 = 3E8h)

Master → Encoder (Set request)

COB-ID	Cmd	Index	Sub	Process data
600+ID	23	00 33	00	E8 03 00 00

Encoder → Master (Set confirmation)

COB-ID	Cmd	Index	Sub	Process data
580+ID	60	00 33	00	00 00 00 00



NOTE

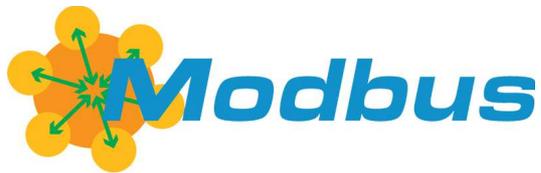
Save default values using **1010-01 Save Parameters** function.

Should the power be turned off or **Reset node** or **Reset communication** commands be sent all data not saved will be lost!

Modbus[®] interface

RD1A

RD12A



RS-232 version



Smart encoders & actuators

1 Using the RS-232 service serial port

1.1 Configuring the device using Lika setting up software

RD1A / RD12A ROTADRIVE positioning units can be equipped with the following fieldbus communication interfaces: Modbus RTU, Profibus-DP and CANopen DS 301. In the Profibus and CANopen versions configuration can be achieved through a RS-232 service serial interface and in compliance with Modbus protocol. For this purpose they 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 is available in the enclosed documentation or at the address www.lika.biz > **ROTARY ACTUATORS > ROTARY ACTUATORS (DRIVECOD) > RD1A / RD12A**. 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 a release of the EDS file and the release of the Modbus executable file have to be respected compulsorily.

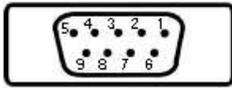
Compatibility	EDS file	Modbus EXE
	V1	up to 2.1
	V2	up to 2.4
	V3	from 2.5 up to ...



NOTE

Before starting the program, connect the device to the personal computer through a serial port. The serial interface of the ROTADRIVE unit for connection to the Modbus RTU fieldbus is a RS-232 type connector. Should the personal computer not be equipped with a serial port, you must install a USB / RS-232 converter, easily available in the market.

On the ROTADRIVE side the cable must be connected to the M12 8-pin male connector (INPUTS / OUTPUTS + Modbus RS-232). See section "Electrical connections" on page 20.

			
Function	Inputs / outputs + RS-232 M12 8-pin male connector	9-pin D-SUB female connector	Function
TD	6	2	RD
RD	7	3	TD
0 VDC	8	5	0 VDC

Always make sure that the RD of the ROTADRIVE unit is cross-wired to the TD of the PC and the TD of the PC is cross-wired to the RD of the ROTADRIVE unit. A M12F8 / D9F connectors cable assembly is available on request; please contact Lika Electronic s.r.l. Technical Assistance & After Sale Service and quote the following code: **EXC-M12F8-LK-xx-D9F-S51**.

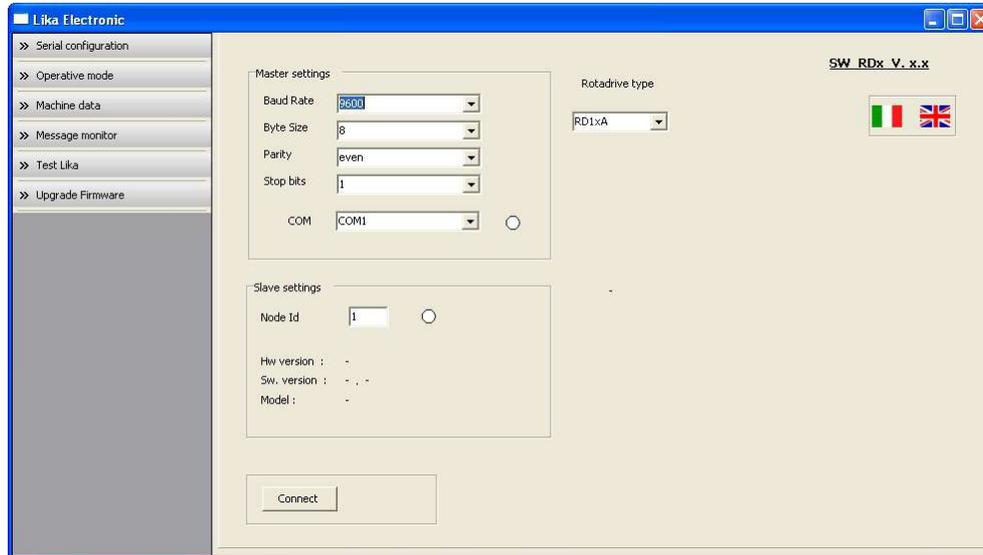
Please note that the configuration parameters of Modbus service serial port have fixed values so the user cannot change them. They are:

RS-232 Modbus service serial port settings	Default value
Baud rate	9600
Byte size	8
Parity	Even
Stop bits	1

Furthermore the data transmission rate which is set using the selector located inside the housing -see section "4.4.2 Setting data transmission rate: Baud rate (Figure 5)" on page 29- concerns the CANopen fieldbus interface only. Thus the selector does not affect the RS-232 Modbus service serial port settings.

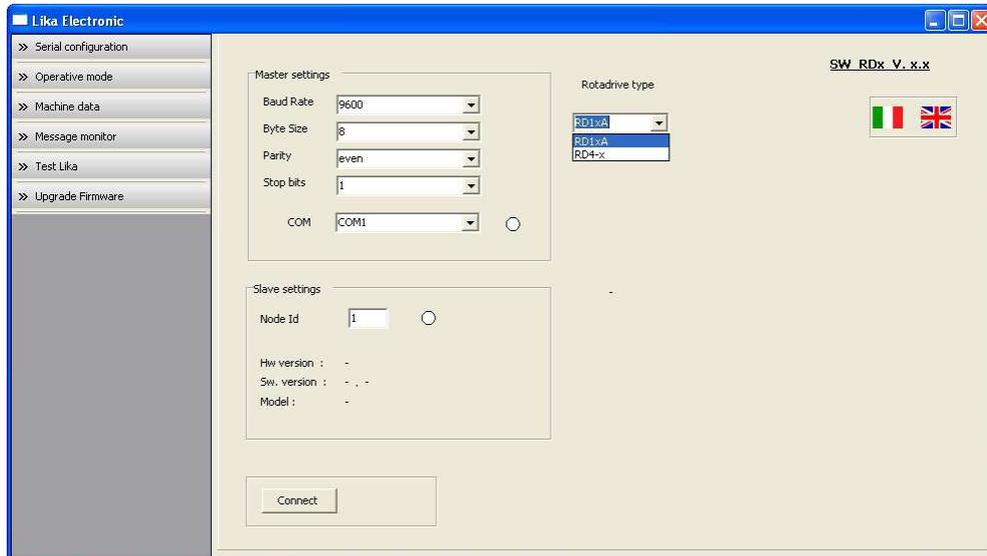
1.2 "Serial configuration" page

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

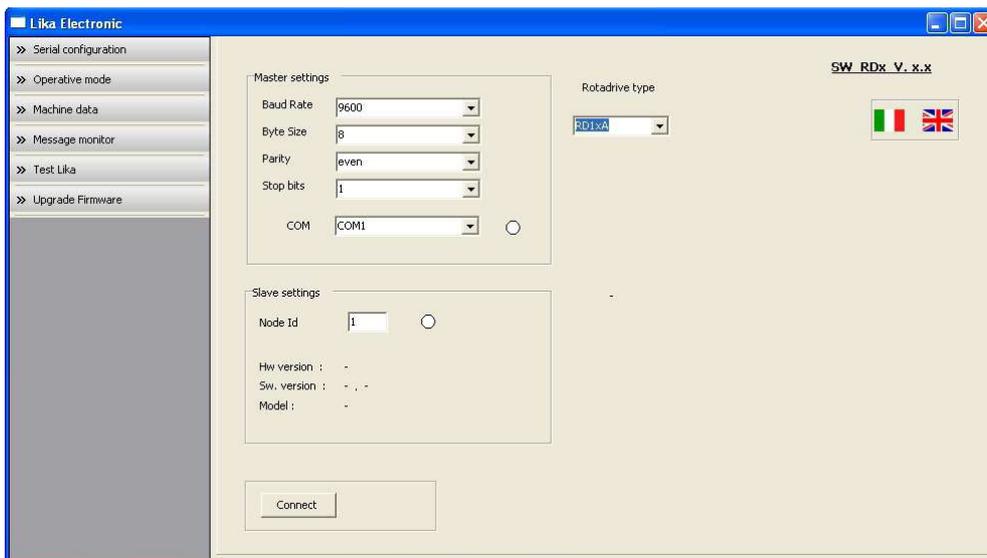


First of all this page allows the operator to choose the language used to display texts and items in the user interface. Click the **Italian flag**  to choose the Italian language; click the **UK flag**  icon to choose the English language.

Before entering the next pages pressing the buttons in the menu on the left, it is necessary to establish a serial connection with the RD1xA unit. To do so, open the **Rotadrive type** drop-down box and select the **RD1xA** 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 RD1xA unit is connected to (**COM** drop-down box). Serial port settings in the personal computer must compulsorily match those in the connected Lika device. Please note that the configuration parameters of Modbus service serial port have fixed values so the user cannot change them.



They are:

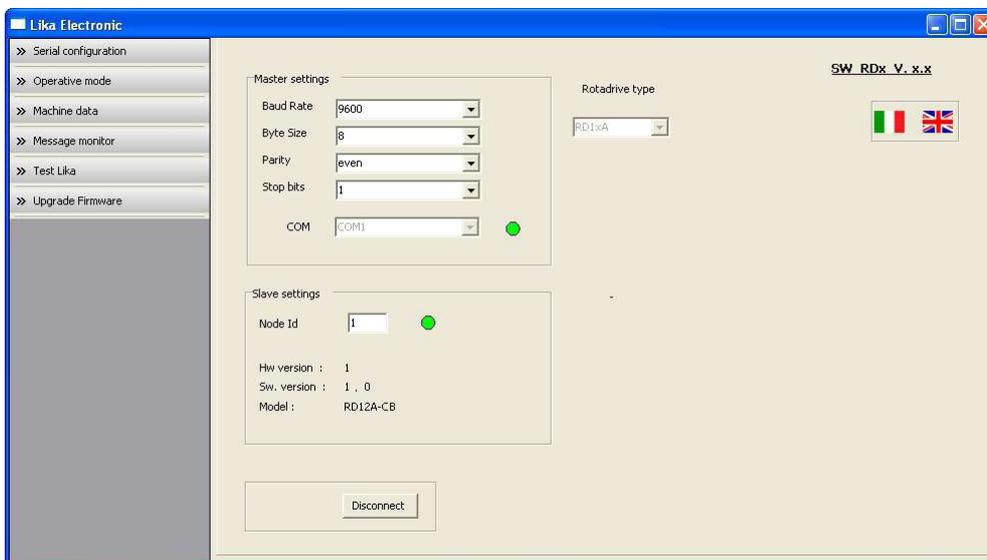
RS-232 Modbus service serial

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

Furthermore the data transmission rate which is set using the selector located inside the housing -see section "4.4.2 Setting data transmission rate: Baud rate (Figure 5)" on page 29- concerns the CANopen fieldbus interface only. Thus the selector does not affect the RS-232 Modbus service serial port settings.

Then set the node address of the device the personal computer is connected to through the **Slave settings** box (default value for all RD1xA-Modbus positioning units = 1). To set the node address of the RD1xA device refer to section "4.4.1 Setting the node address: Node ID (Figure 5)" on page 28.

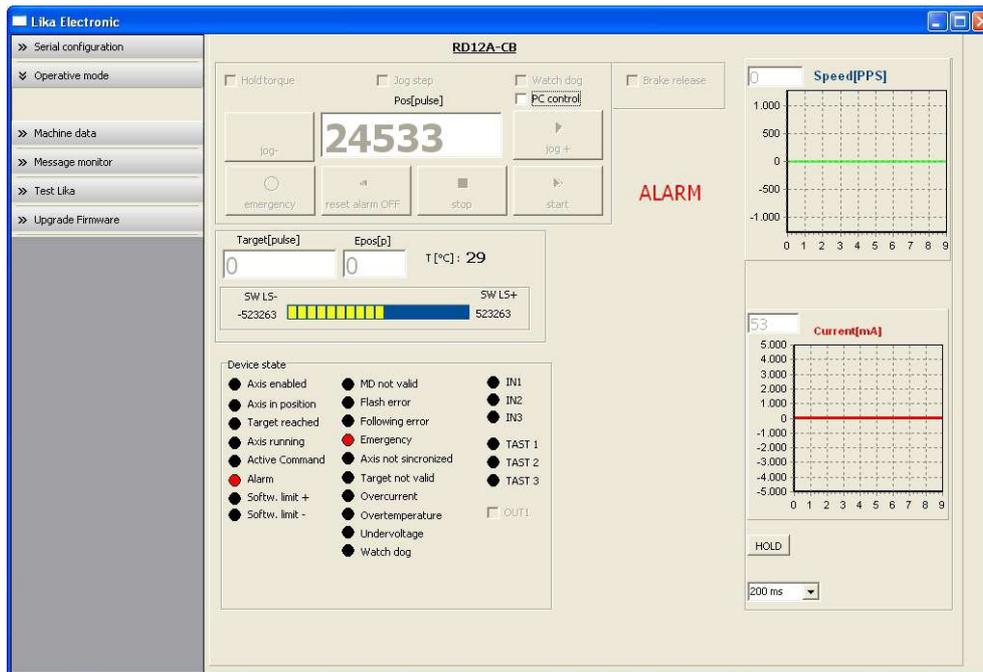
Now you are ready to establish the connection to the Slave: press the **CONNECT** button below in the page.



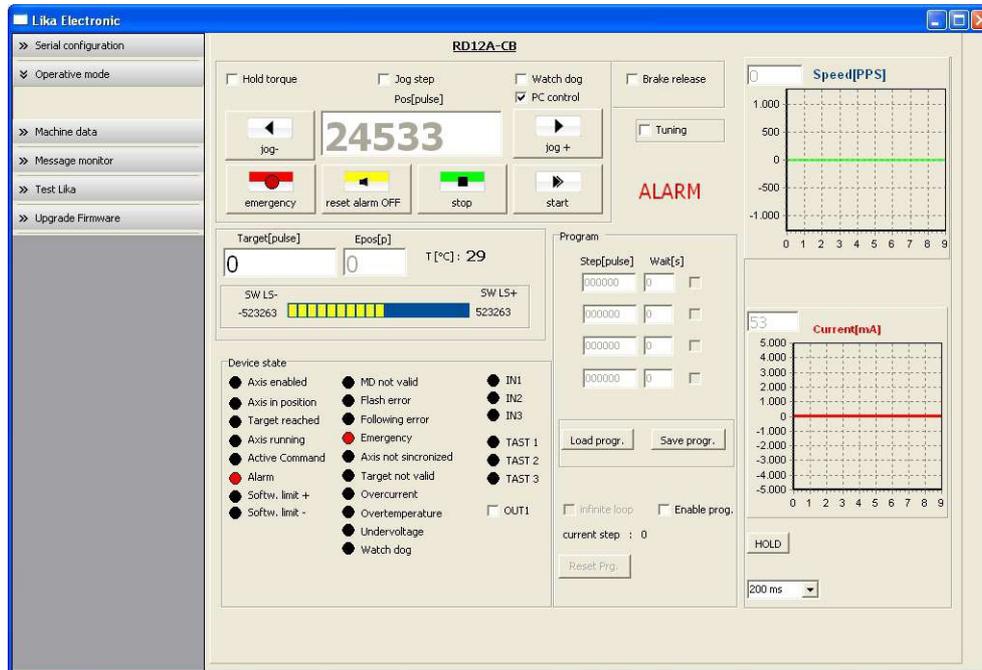
If the connection is established properly, two green lights placed next to the fields used to choose the **serial port** and set the **node ID** come on, while the **CONNECT** button disappears and is replaced by the **DISCONNECT** button. Furthermore the hardware version and the software version as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

1.3 "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, all commands are disabled as the unit is still under CANopen network control. To start programming, controlling manually and monitoring the device through the RS-232 service serial interface in Modbus protocol, it is necessary to enable the available commands by gaining control of the unit in the Modbus network via PC. To do this, select the **PC CONTROL** check box (see [Extra commands register \[0x29\]](#) on page 124).



All commands become immediately available for use.

When you first enter the **Operative mode** page, RD1xA 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 RD1xA 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 RD1xA unit starts blinking.

In the top left-hand **RD1xA-CB** box the following functions are available.

Hold torque

See **Axis torque** item on page 127. This function is available only to RD1A version; in RD12A version this check-box is hidden.

Jog step

See **Incremental jog** item on page 126.

Watch dog

See **Watch dog enable** item on page 126.

Jog -

See **Jog -** item on page 125.

Pos [pulse]

See **Current position [0x02-0x03]** item on page 134.

Jog +

See **Jog +** item on page 125.

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 126.

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 126.

Stop

Press this button to force a normal halt of the device, respecting the acceleration and deceleration values. See **Stop** item on page 125.

Start

Pressing the button causes the unit to start running in order to reach the position set next to the **Target [P]** 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 126.

In the box just below the **RD1xA-CB** box the following functions are available.

Target [pulse]

See **Target position [0x2B-0x2C]** item on page 128. 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 \[0x05-0x06\]](#) item on page 134.

T [°C]

See [Temperature value \[0x08\]](#) item on page 135.

SW LS - / SW LS +

It shows visually the set positive and negative limit switch values. See [Positive delta \[0x08-0x09\]](#) item on page 118 and [Negative delta \[0x0A-0x0B\]](#) item on page 119.

In the bottom left-hand **Device state** box the list of states and alarms available for the RD1xA 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 \[0x01\]](#) item on page 132; for a detailed description of the alarms see [Alarms register \[0x00\]](#) item on page 130. The **OUT1** check box is meant to activate / deactivate the operation of the digital output 1 in the device. For a detailed description see on page 127.

Functions available in the **Program** box allow the operator to create and then save a work program for the RD1xA 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 RD1xA 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.

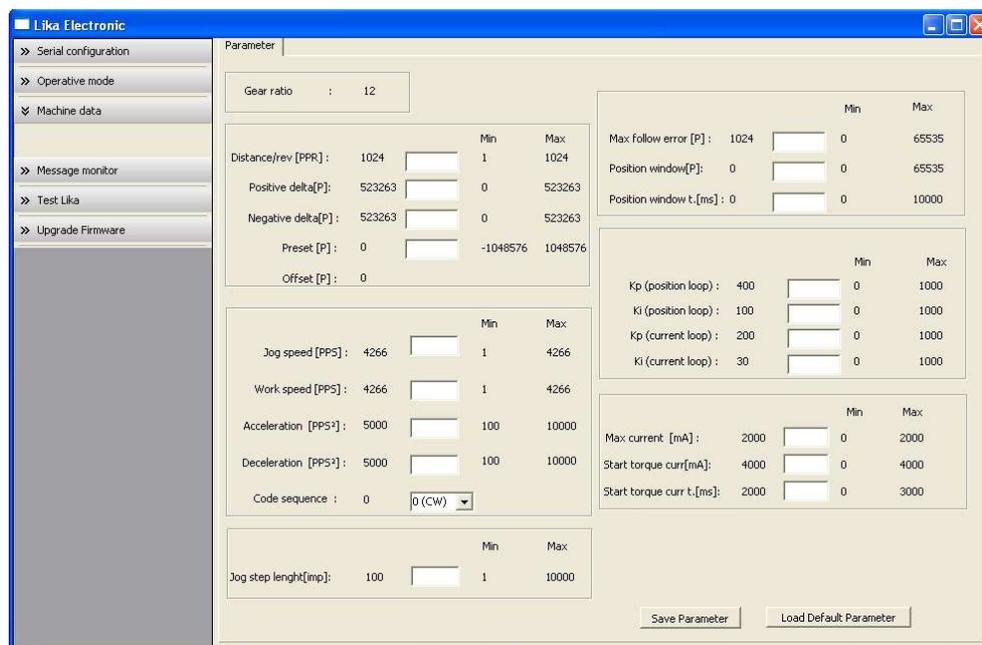
On the right side of the page the speed of the device (expressed in PPS) and the value of the current absorbed by the motor (expressed in milliamperes) are shown visually through charts. Press the **HOLD** button to disable charts visualization; press the same button (now labelled **GO**) to enable it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

On the right of the **RD1xA-CB** 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 "3.1.1 Machine data parameters" on page 116).

When you connect to a ROTADRIVE unit RD12A model, the **BRAKE RELEASE** check box appears just above the **TUNING** check box. Unlike RD1A model, RD12A 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. The brake is not kept deactivated for an indefinite period of time; a time-out has been implemented in order to reactivate automatically the brake after 20 seconds have expired.

1.4 "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 RD1xA 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 "3.1.1 Machine data parameters" on page 116.

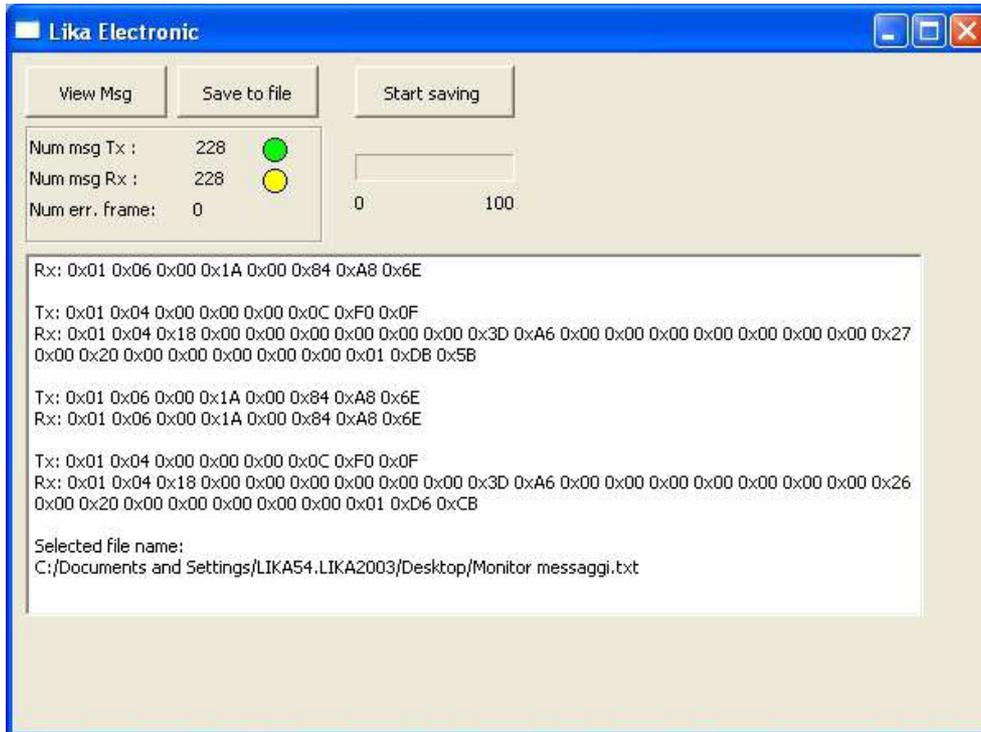
To enter a new value type it in the blank field and then press the **ENTER** key in the keyboard. If you set a value not allowed (out of range), at confirmation prompt the field is highlighted in red and the RD1xA unit is forced in alarm condition (the **Alarm** status bit is activated and the **Machine 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 127.

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 127. On page 40 the complete list of the machine data parameters and the relevant default values as set by Lika Electronic are available.

1.5 "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**

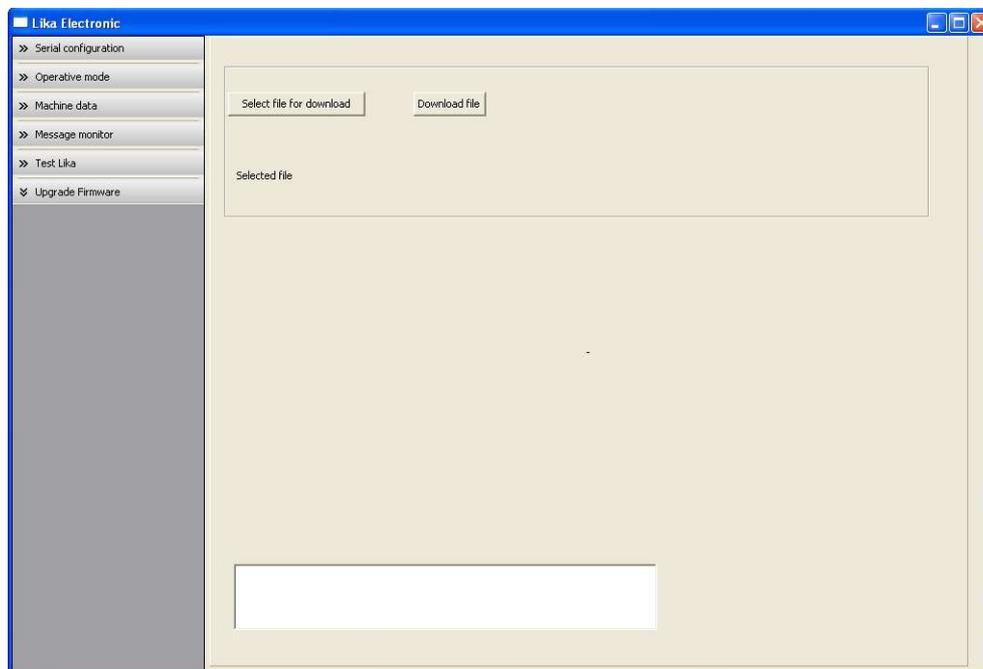
monitor page. Now press the **START SAVING** button to start saving the messages; the "File opened properly" message appears on the display box. After pressing the **START SAVING** button, its descriptive label is replaced by **STOP SAVING** label. Press the **STOP SAVING** button to stop saving the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

1.6 "Test Lika" page

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

1.7 "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 ROTADRIIVE 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 make corrections, improve and even add new functionalities to the device.

The firmware upgrading program consists of a single file having .BIN extension. It is released by Lika Electronic Technical Assistance & After Sale Service.



WARNING

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 87).

The firmware upgrading function has been implemented starting from:

- firmware version 1.0 in the ROTADRIVE units having CANopen and Modbus bus interface;
- firmware version 2.0 in the ROTADRIVE units having Profibus bus interface.

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



NOTE

The firmware upgrading function has been implemented starting from the version 2.0 of the **SW_RDX_MODBUS_2.0.EXE** software for ROTADRIVE configuration via serial port in Modbus protocol. Before proceeding with a new firmware installation please make sure you have the 2.0 or latest release of the program or get it from the following link: **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS (DRIVECOD) > RD1A / RD12A.**

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 2.0 or latest release of the SW_RDX_MODBUS_2.0.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 "1.7.1 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_PB_H1S2.0.BIN, where:

- RD1xA = ROTADRIVE unit model;
- PB = bus interface of the ROTADRIVE unit (MB = Modbus; CB = CANopen; PB = 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, otherwise 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.

1.7.1 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 24), 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.

2 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.

2.1 Modbus Master / Slaves protocol principle

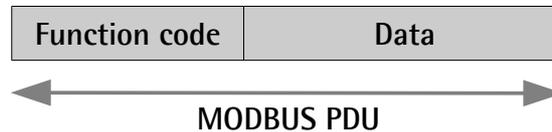
The Modbus Serial Line protocol is a Master – Slaves protocol. One only Master (at the same time) is connected to the bus and one or several (247 maximum number) Slave nodes are also connected to the same serial bus. A Modbus communication is always initiated by the Master. The Slave nodes will never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

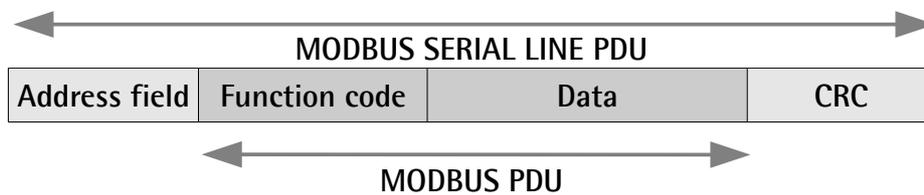
- **UNICAST mode:** in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must 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.

2.2 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 "4.4.1 Setting the node address: Node ID (Figure 5)" on page 28), 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 "2.4 Function codes" on page 106.
- **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 "2.4 Function codes" on page 106).
- **CRC (Cyclical Redundancy Checking):** error checking field is the result of a "Redundancy Checking" calculation that is performed on the message contents. This is intended to check whether transmission has

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

The Modbus protocol defines three PDUs. They are:

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

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

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

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

2.3 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.

2.3.1 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.

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
-------	---	---	---	---	---	---	---	---	---------	------

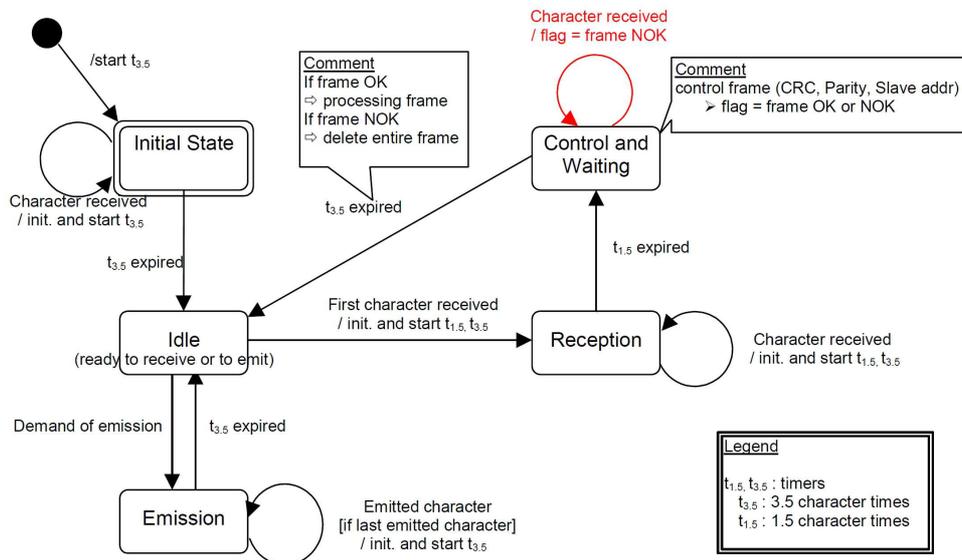
* When "No parity" is activated, the parity bit is replaced by a stop bit.

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.

2.4 Function codes

As previously stated, the function code indicates to the Server what kind of action to perform. The function code field of a Modbus data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (the range 128 ... 255 is reserved and used for Exception Responses). When a message is sent from a Client to a Server device the function code field tells the Server what kind of action to perform. Function code "0" is not valid.

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

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

2.4.1 Implemented function codes

Lika RD1xA positioning units only implement public function codes, they are described hereafter.

03 Read Holding Registers

FC = 03 (Hex = 0x03) ro

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

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

For the complete list of holding registers accessible using **03 Read Holding Registers** function code please refer to section "3.1.1 Machine data parameters" on page 116.

Request PDU

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

Response PDU

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

*N = Quantity of registers

Exception Response PDU

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



Here is an example of a request to read parameters **Acceleration [0x06]** (register 7) and **Deceleration [0x07]** (register 8).

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	03	Function	03
Starting address Hi	00	Byte count	04
Starting address Lo	06	Register 7 value Hi	03
No. of registers Hi	00	Register 7 value Lo	E8
No. of registers Lo	02	Register 8 value Hi	05
		Register 8 value Lo	DC

As you can see in the table, **Acceleration [0x06]** parameter (register 7) contains the value 03 E8 hex, i.e. 1000 in decimal notation; **Deceleration [0x07]** 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 [0x06]** (register 7) and **Deceleration [0x07]** (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 [0x06]** 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 [0x06]** (register 7) and **Deceleration [0x07]** (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 [0x06]**, 03 E8 hex = 1000 dec

[05][DC] = value of register 8 **Deceleration [0x07]**, 05 DC hex = 1500 dec

[78][8A] = CRC

04 Read Input Register

FC = 04 (Hex = 0x04)

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

For the complete list of input registers accessible using **04 Read Input Register** function code please refer to section "3.1.2 Input Register parameters" on page 130.

Request PDU

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

Response PDU

Function code	1 byte	0x04
Byte count	1 byte	2 x N*
Input register value	N* x 2 bytes	

*N = Quantity of registers

Exception Response PDU

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



Here is an example of a request to read **Current position [0x02-0x03]** parameter (input registers 3 and 4).

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	04	Function	04
Starting address Hi	00	Byte count	04
Starting address Lo	02	Register 3 value Hi	00
Quantity of Input Reg. Hi	00	Register 3 value Lo	00
Quantity of Input Reg. Lo	02	Register 4 value Hi	2F
		Register 4 value Lo	F0

As you can see in the table, **Current position [0x02-0x03]** 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 [0x02-0x03]** parameter (input registers 3 and 4) to the Slave having the node address 1 is as follows:

Request PDU (in hexadecimal format)

[01][04][00][02][00][02][D0][0B]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[00][02] = starting address (**Current position [0x02-0x03]** parameter, register 3)

[00][02] = number of requested registers

[D0][0B] = CRC

The full frame needed to send back the value of the **Current position [0x02-0x03]** parameter (registers 3 and 4) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)

[01][04][04][00][00][2F][F0][E7][F0]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[04] = number of bytes (2 bytes for each register)

[00][00] = value of register 3 **Current position [0x02-0x03]**, 00 00 hex = 0 dec

[2F][F0] = value of register 4 **Current position [0x02-0x03]**, 2F F0 hex = 12272 dec

[E7][F0] = CRC

06 Write Single Register

FC = 06 (Hex = 0x06)

This function code is used to WRITE a single holding register in a remote device. The Request PDU specifies the address of the register to be written. Registers are addressed starting at zero. Therefore register numbered 1 is addressed as 0.

The normal response is an echo of the request, returned after the register contents have been written.

For the complete list of registers accessible using **06 Write Single Register** function code please refer to section "3.1.1 Machine data parameters" on page 116.

Request PDU

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

Response PDU

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

Exception Response PDU

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



Here is an example of a request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [0x06]** parameter (register 7).

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	06	Function	06
Register address Hi	00	Register address Hi	00
Register address Lo	06	Register address Lo	06
Register value Hi	05	Register value Hi	05
Register value Lo	DC	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 [0x06]** parameter (register 7).

The full frame needed for the request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [0x06]** 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 [0x06]** 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 [0x06]** 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 [0x06]** parameter, register 7)

[05][DC] = value set in the register

[6B][02] = CRC

16 Write Multiple Registers

FC = 16 (Hex = 0x10)

This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.

The values to be written are specified in the request data field. Data is packed as two bytes per register.

The normal response returns the function code, starting address and quantity of written registers.

For the complete list of registers accessible using **16 Write Multiple Registers** function code please refer to section "3.1.1 Machine data parameters" on page 116.

Request PDU

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

*N = Quantity of registers

Response PDU

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

Exception Response PDU

Error code	1 byte	0x90 (= 0x10 + 0x80)
Exception code	1 byte	01 or 02 or 03 or 04



Here is an example of a request to write the values 1500 and 1000 dec in the parameters **Acceleration [0x06]** (register 7) and **Deceleration [0x07]** (register 8) respectively.

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

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 [0x06]** parameter (register 7); the value 03 E8 hex, i.e. 1000 in decimal notation, is set in the **Deceleration [0x07]** 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 [0x06]** (register 7) and **Deceleration [0x07]** (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 [0x06]** 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 [0x06]**, 05 DC hex = 1500 dec

[03][E8] = value to be set in the register 8 **Deceleration [0x07]**, 03 E8 hex = 1000 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 [0x06]** (register 7) and **Deceleration [0x07]** (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 [0x06]** parameter, register 7)

[00][02] = number of written registers

[A1][C9] = CRC



NOTE

For further examples refer to section "Modbus® programming examples" on page 140.



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 [0x2A]** variable. If "=0" is set the Watch dog is not enabled; if "=1" is set the Watch dog is enabled. When the Watch dog function is enabled, if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm message is invoked to appear as soon as the Modbus network communication is restored).

3 Modbus® programming parameters

3.1 Parameters available

Hereafter the parameters available for RD1xA 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

3.1.1 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 "2.4.1 Implemented function codes" on page 106.

Distance per revolution [0x00]

[Register 1, Unsigned16, rw]

This parameter sets the number of pulses per each complete revolution of the shaft. It is useful to relate the revolution of the shaft and a linear measurement. For example: unit is joined to a worm screw having a 5 mm pitch; by setting **Distance per revolution [0x00]** = 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 [0x0C]**, **Work speed [0x0D]** and **Preset [0x16-0x17]**. For a detailed explanation see on page 36 and relevant parameters.

Please note that the parameters listed hereafter are closely related to the **Distance per revolution [0x00]** parameter; hence when you change the value

in **Distance per revolution [0x00]** also the value expressed by each one is necessarily redefined. They are: **Position window [0x01]**, **Max following error [0x03]**, **Acceleration [0x06]**, **Deceleration [0x07]**, **Positive delta [0x08-0x09]**, **Negative delta [0x0A-0x0B]**, **Target position [0x2B-0x2C]**, **Current position [0x02-0x03]**, **Current velocity [0x04]** and **Position following error [0x05-0x06]**. See for instance the relationship between **Distance per revolution [0x00]** and the speed values, explained on page 121.



NOTE

If **Distance per revolution [0x00]** 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 [0x01]

[Register 2, Unsigned16, rw]

This parameter defines the tolerance window for the **Target position [0x2B-0x2C]** value. When the axis is within the tolerance window limits for the time set in the **Position window time [0x02]** 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 [0x02]

[Register 3, Unsigned16, rw]

It represents the time for which the axis has to be within the tolerance window limits set in the **Position window [0x01]** parameter before the state is signalled through the **Axis in position** status bit. Parameter is expressed in milliseconds.

Default = 0 (min. = 0, max. = 10000)

Max following error [0x03]

[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 [0x04]

[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 = 400 (min. = 0, max. = 1000)

Ki position loop [0x05]

[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 = 100 (min. = 0, max. = 1000)

Acceleration [0x06]

[Register 7, Unsigned16, rw]

This parameter defines the acceleration value that has to be used by the device. Parameter is expressed in pulses per second² [PPS²].

Default = 5000 for RD1xA-...-T12-... model (min. = 100, max. = 10000)

Default = 2500 for RD1xA-...-T24-... model (min. = 100, max. = 10000)

Default = 1000 for RD1xA-...-T48-... model (min. = 100, max. = 10000)

Default = 500 for RD1xA-...-T92-... model (min. = 100, max. = 10000)

Deceleration [0x07]

[Register 8, Unsigned16, rw]

This parameter defines the deceleration value that has to be used by the device. Parameter is expressed in pulses per second² [PPS²].

Default = 5000 for RD1xA-...-T12-... model (min. = 100, max. = 10000)

Default = 2500 for RD1xA-...-T24-... model (min. = 100, max. = 10000)

Default = 1000 for RD1xA-...-T48-... model (min. = 100, max. = 10000)

Default = 500 for RD1xA-...-T92-... model (min. = 100, max. = 10000)

Positive delta [0x08-0x09]

[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.

SW limit switch + = **Preset [0x16-0x17]** + **Positive delta [0x08-0x09]**.

Default = 523263 (min. = 0, max. = 523263)



WARNING

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



EXAMPLE

When **Distance per revolution [0x00]** = 1024 and **Preset [0x16-0x17]** = 0, the maximum acceptable value for **Positive delta [0x08-0x09]** is:

(1024 steps per revolution * 511 revolutions) - 1 = 523263

When **Distance per revolution [0x00]** = 256 and **Preset [0x16-0x17]** = 0, the maximum acceptable value for **Positive delta [0x08-0x09]** is:

(256 steps per revolution * 511 revolutions) - 1 = 130815

See further examples in the paragraph "6.4 Distance per revolution, Jog speed, Work speed, Preset and limit switch values" on page 36.



WARNING

Every time **Distance per revolution [0x00]** and **Preset [0x16-0x17]** parameters are changed, **Positive delta [0x08-0x09]** and **Negative delta [0x0A-0x0B]** values has to be checked carefully. Each time you change the value in **Distance per revolution [0x00]** you must then update the value in **Preset [0x16-0x17]** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **Preset [0x16-0x17]** 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 [0x08-0x09]** and **Negative delta [0x0A-0x0B]** items. For a detailed explanation see on page 36.

Negative delta [0x0A-0x0B]

[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.

SW limit switch - = **Preset [0x16-0x17]** - **Negative delta [0x0A-0x0B]**.

Default = 523263 (min. = 0, max. = 523263)



WARNING

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



EXAMPLE

When **Distance per revolution [0x00]** = 1024 and **Preset [0x16-0x17]** = 0, the maximum acceptable value for **Negative delta [0x0A-0x0B]** is:

(1024 steps per revolution * 511 revolutions) - 1 = 523263

When **Distance per revolution [0x00]** = 256 and **Preset [0x16-0x17]** = 0, the maximum acceptable value for **Negative delta [0x0A-0x0B]** is:

(256 steps per revolution * 511 revolutions) - 1 = 130815

See further examples in the paragraph "6.4 Distance per revolution, Jog speed, Work speed, Preset and limit switch values" on page 36.



WARNING

Every time **Distance per revolution [0x00]** and **Preset [0x16-0x17]** parameters are changed, **Positive delta [0x08-0x09]** and **Negative delta [0x0A-0x0B]** values has to be checked carefully. Each time you change the value in **Distance per revolution [0x00]** you must then update the value in **Preset [0x16-0x17]** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **Preset [0x16-0x17]** 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 [0x08-0x09]** and **Negative delta [0x0A-0x0B]** items. For a detailed explanation see on page 36.

Jog speed [0x0C]

[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 = 4266 for RD1xA-...-T12-... model (min. = 1, max. = 4266)

Default = 2133 for RD1xA-...-T24-... model (min. = 1, max. = 2133)

Default = 1066 for RD1xA-...-T48-... model (min. = 1, max. = 1066)

Default = 556 for RD1xA-...-T92-... model (min. = 1, max. = 556)

Work speed [0x0D]

[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 [0x2B-0x2C]**). Parameter is expressed in pulses per second.

Default = 4266 for RD1xA-...-T12-... model (min. = 1, max. = 4266)

Default = 2133 for RD1xA-...-T24-... model (min. = 1, max. = 2133)

Default = 1066 for RD1xA-...-T48-... model (min. = 1, max. = 1066)

Default = 556 for RD1xA-...-T92-... model (min. = 1, max. = 556)



WARNING

Each time you change the value in **Distance per revolution [0x00]** you must then set new values also in **Jog speed [0x0C]** and **Work speed [0x0D]** as speed values are expressed in pulses per second (PPS). To calculate the speed values you have always to adhere to the following ratio:

$$\frac{\text{Min. speed} * \text{Distance per revolution}}{1024} \leq \text{Speed} \leq \frac{\text{Max. speed} * \text{Distance per revolution}}{1024}$$

For a detailed explanation see on page 36.

Start torque current time [0x0E]

[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 [0x13]** item). Parameter is expressed in milliseconds.

Default = 2000 (min. = 0, max. = 3000)

Code sequence [0x0F]

[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 [0x16-0x17]** parameter and then check the values set next to the **Positive delta [0x08-0x09]** and **Negative delta [0x0A-0x0B]** items.

Kp current loop [0x10]

[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 = 200 (min. = 0, max. = 1000)

Ki current loop [0x11]

[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 = 30 (min. = 0, max. = 1000)

Max current [0x12]

[Register 19, Unsigned16, rw]

This parameter defines the maximum current supplied by power electronic for controlling the motor. Parameter is expressed in mA (milliamperes). This value cannot be greater than the one in **Starting torque current [0x13]** item.

Default = 2000 (min. = 10, max. = 2000)

Starting torque current [0x13]

[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 [0x0E]** item. Parameter is expressed in mA (milliamperes).

Default = 4000 (min. = 10, max. = 4000)

Offset [0x14-0x15]

[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 [0x16-0x17]

[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. = -1048576, max. = 1048576)



WARNING

A new value has to be set in **Preset [0x16-0x17]** every time **Distance per revolution [0x00]** value is changed. After having entered a new value in **Preset [0x16-0x17]** 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 [0x08-0x09]** and **Negative delta [0x0A-0x0B]** items. For a detailed explanation see on page 36.

Gear ratio [0x18]

[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 = 12 for RD1xA-...-T12-... model

Default = 24 for RD1xA-...-T24-... model

Default = 48 for RD1xA-...-T48-... model

Default = 92 for RD1xA-...-T92-... model

Jog step length [0x19]

[Register 26, Unsigned16, rw]

If the incremental jog function is enabled (bit 4 **Incremental jog** in **Control Word [0x2A]** = 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 [0x29]

[Register 42, Unsigned16, rw]

Byte structure of the **Extra commands register [0x29]**:

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

Byte 0

Absolute reading

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

Control from PC

bit 1 If set to "0" the device is intended to communicate in the CANopen network; if set to "1" the device is intended to communicate in the Modbus network through the RS-232 service serial interface (see section "Modbus® interface" on page 82).

bits 2 ... 7 Not used.

Byte 1 Not used.

Control Word [0x2A]

[Register 43, Unsigned16, rw]

This variable contains the commands to be sent in real time to the Slave in order to manage it.

Byte structure of the **Control Word [0x2A]** register:

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

Byte 0

Jog +

bit 0

If 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 [0x19]** item; then the slave stops and waits for another issue. Velocity, acceleration and deceleration are set in parameters **Jog speed [0x0C]**, **Acceleration [0x06]** and **Deceleration [0x07]** respectively. For a detailed description of jog control see on page 34.

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 [0x19]** item; then the slave stops and waits for another issue. Velocity, acceleration and deceleration are set in parameters **Jog speed [0x0C]**, **Acceleration [0x06]** and **Deceleration [0x07]** respectively. For a detailed description of jog control see on page 34.

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 [0x07]**. 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 [0x09-0x0A]**), 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 [0x19]** item; then the slave stops and waits for another issue.

Please note that when you use the manual buttons (see "4.4.4 JOG + and JOG - buttons (Figure 5)" on page 30) the "incremental jog" function is disabled; that is, jog step movements are not allowed using the manual buttons.



bit 5

Not used.

Start

bit 6

If set to "=1" device moves in order to reach the set target position (see **Target position [0x2B-0x2C]** on page 128). For a complete description of the position control see on page 33.

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

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 40.

Perform counting preset

bit 11 Counting acquires the value in the **Preset [0x16-0x17]** 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 RD1A version (model without brake); in the RD12A 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.

OUT 1

bit 13 This is intended to activate / deactivate the operation of the digital output 1. The meaning of the available outputs is

described in section "Modbus® programming parameters" on page 116.

OUT 1 = 0 output 1 low (not active)

OUT 1 = 1 output 1 high (active)

Brake released

bit 14

This function is available only in RD12A version (model fitted with brake); in the RD1A version (model without brake) bit 14 is not used. RD12A 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 [0x2B-0x2C]

[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 while the device is still reaching it; to do this, send a **Start** command and the new target value in **Target position [0x2B-0x2C]**.



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 [0x2A]** 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!

3.1.2 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 "2.4.1 Implemented function codes" on page 106.

Alarms register [0x00]

[Register 1, Unsigned16, ro]

This variable is meant to show the alarms currently active in the device.

Structure of the alarms byte:

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

The available alarm error codes are listed hereafter:

Byte 0

Machine data not valid

bit 0 One or more parameters are not valid, set proper values to restore normal work condition. See the list of wrong parameters in **Wrong parameters list [0x09-0x0A]**.

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 [0x03]** 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 [0x2A]** 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 [0x08]**).

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 [0x2A]** is set to "1"), if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm bit is activated). The alarm is invoked to appear as soon as the Modbus network communication is restored. 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 [0x2A]** bit 3. In a normal work condition the **Alarm reset** bit is set to "0". Setting the bit to "1" causes the normal work status of the device to be restored. 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 [0x09-0x0A]**), normal work status can be restored only after having set proper values. **Flash memory error** alarm cannot be reset.

Status word [0x01]

[Register 2, Unsigned16, ro]

This register contains information about the current state of the device.

Byte structure of the **Status word [0x01]** register:

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

Byte 0

Axis in position

bit 0

If value is "1" device has reached the set position for the time set in **Position window time [0x02]**. It is kept active until the position error is lower than **Position window [0x01]**.

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 [0x08-0x09]**.

SW limit switch -

bit 4

If value is "1" device has reached the maximum negative limit (negative limit switch). See parameter **Negative delta [0x0A-0x0B]**.

Alarm

bit 5

If value is "1" an alarm has occurred, see details in the **Alarms register [0x00]** variable.

Axis running

bit 6

If value is "0" device is not moving.
If value is "1" device is running.

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 [0x2B-0x2C]** item. Bit is kept active until a new **Target position [0x2B-0x2C]** value or **Alarm reset** commands are sent.

Button 1 Jog +

bit 9 RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. Once you press the button 1 JOG +, bit 9 is forced high "=1"; when the button 1 is not pressed, bit 9 is low "=0". For further information see section "4.4 Dip-Switches and buttons (Figure 5)" on page 27.

Button 2 Jog -

bit 10 RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. Once you press the button 2 JOG -, bit 10 is forced high "=1"; when the button 2 is not pressed, bit 10 is low "=0". For further information see section "4.4 Dip-Switches and buttons (Figure 5)" on page 27.

Button 3 Preset

bit 11 RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. Once you press the button 3 PRESET, bit 11 is forced high "=1"; when the button 3 is not pressed, bit 11 is low "=0". For further information see section "4.4 Dip-Switches and buttons (Figure 5)" on page 27.

DAC saturation

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

IN 1

bit 13

This is meant to show the status of the digital input 1. The meaning of the available inputs is described in section "Modbus® programming parameters" on page 116.

IN 1 = 0 input 1 low (not active)

IN 1 = 1 input 1 high (active)

IN 2

bit 14

This is meant to show the status of the digital input 2. The meaning of the available inputs is described in section "Modbus® programming parameters" on page 116.

IN 2 = 0 input 2 low (not active)

IN 2 = 1 input 2 high (active)

IN 3

bit 15

This is meant to show the status of the digital input 3. The meaning of the available inputs is described in section "Modbus® programming parameters" on page 116.

IN 3 = 0 input 3 low (not active)

IN 3 = 1 input 3 high (active)

Current position [0x02-0x03]

[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 [0x04]

[Register 5, Integer16, ro]

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

Position following error [0x05-0x06]

[Registers 6-7, Integer32, ro]

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

Current value [0x07]

[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 [0x08]

[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 [0x09-0x0A]

[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 the normal work status can be restored only after having set proper values.

Bit	Parameter
1	Distance per revolution [0x00]
7	Acceleration [0x06]
8	Deceleration [0x07]
9	Positive delta [0x08-0x09]
10	Negative delta [0x0A-0x0B]
11	Jog speed [0x0C]
12	Work speed [0x0D]
13	Start torque current time [0x0E]
14	Code sequence [0x0F]
17	Max current [0x12]
18	Starting torque current [0x13]
19	Gear ratio [0x18]
20	Jog step length [0x19]
26	Preset [0x16-0x17]

I2t [0x0B]

[Register 12, Unsigned16, ro]

Thermal image or signal proportional to the current (for monitoring purposes only).

Dip-switch baud rate [0x0C]

[Register 13, Unsigned16, ro]

This is meant to show the data transmission rate (baud rate) of the serial port fitted in the RD1xA unit; data transmission rate has to be set through the provided dip-switch (CANopen network only). For any further information on setting the baud rate refer to the section "4.4.2 Setting data transmission rate: Baud rate (Figure 5)" on page 29.

Dip-switch node ID [0x0D]

[Register 14, Unsigned16, ro]

This is meant to show the node address set in the RD1xA 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 "4.4.1 Setting the node address: Node ID (Figure 5)" on page 28.

SW Version [0x0E]

[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 [0x0F]

[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).



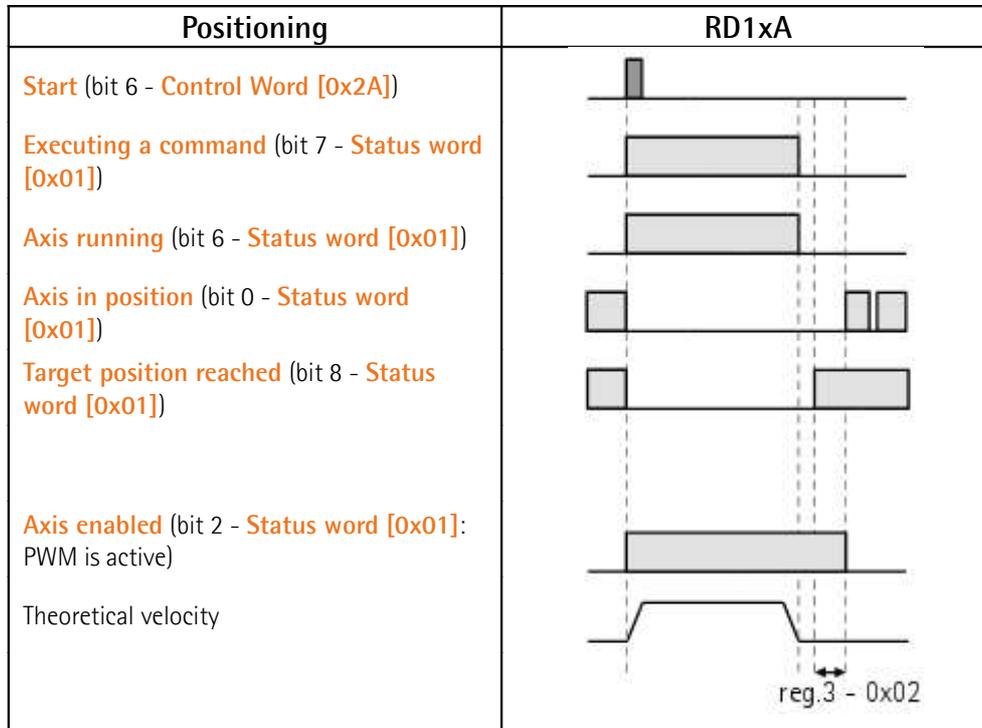
NOTE

Save the set values using **Save parameters** function.

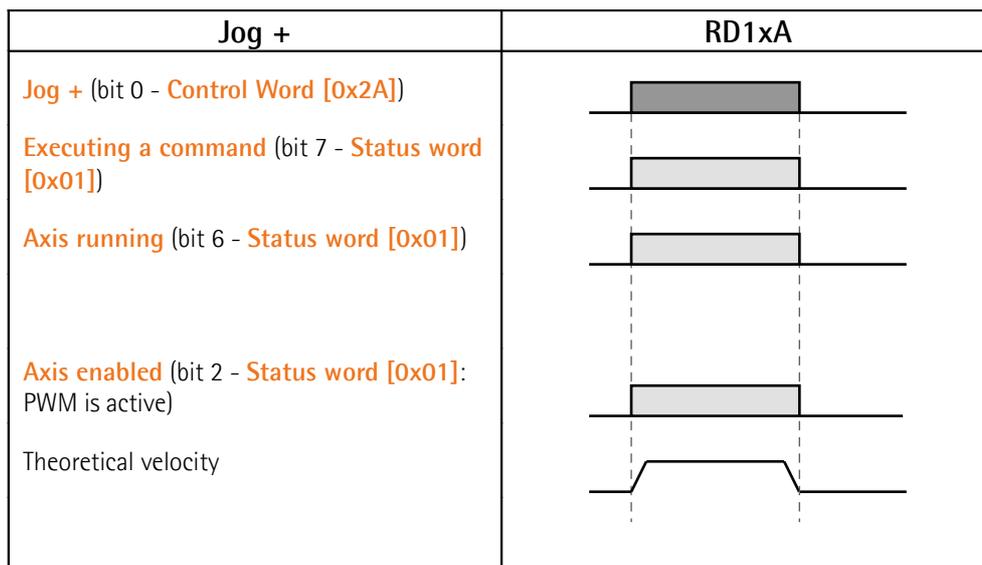
Should the power be turned off all data not saved will be lost!



Example 1



Example 2



3.2 Exception codes

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

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

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

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

DATA FIELD: in a normal response, the Server may return data or statistics in the data field (any information that was requested in the request). In an exception code, the Server returns an exception code in the data field. This defines the Server condition that caused the exception.

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

4 Modbus® programming examples

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

4.1 Using the 03 Read Holding Registers function code



Example 1

Request to read parameters **Acceleration [0x06]** (register 7) and **Deceleration [0x07]** (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 [0x06]** 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 [0x06]**, 03 E8 hex = 1000 dec

[05][DC] = value of register 8 **Deceleration [0x07]**, 05 DC hex = 1500 dec

[78][8A] = CRC

Acceleration [0x06] parameter (register 7) contains the value 03 E8 hex, i.e. 1000 in decimal notation; **Deceleration [0x07]** parameter (register 8) contains the value 05 DC hex, i.e. 1500 in decimal notation.

4.2 Using the 04 Read Input Register function code



Example 1

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

Request PDU

[01][04][00][02][00][02][D0][0B]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[00][02] = starting address (**Current position [0x02-0x03]** parameter, register 3)

[00][02] = number of requested registers

[D0][0B] = CRC

Response PDU

[01][04][04][00][00][2F][F0][E7][F0]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[04] = number of bytes (2 bytes for each register)

[00][00] = value of register 3 **Current position [0x02-0x03]**, 00 00 hex = 0 dec

[2F][F0] = value of register 4 **Current position [0x02-0x03]**, 2F F0 hex = 12272 dec

[E7][F0] = CRC

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



Example 2

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

Request PDU

[01][04][00][00][00][01][31][CA]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[00][00] = starting address (**Alarms register [0x00]** variable, register 1)

[00][01] = number of requested registers
 [31][CA] = CRC

Response PDU

[01][04][02][00][81][79][50]

where:

[01] = Slave address

[04] = **04 Read Input Register** function code

[02] = number of bytes (2 bytes for each register)

[00][81] = value of register 1 **Alarms register [0x00]**, 00 81 hex = 0000 0000
 1000 0001 bin

[79][50] = CRC

This means that in the **Alarms register [0x00]** variable (register 1) bits 0 and 7 are active (logic level high = 1), i.e. (see on page 130): **Machine data not valid** and **Emergency**.

4.3 Using the **06 Write Single Register** function code



Example 1

Request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [0x06]** 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 [0x06]** 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 [0x06]** 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 [0x06]** parameter (register 7).



Example 2

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

Request PDU

[01][06][00][2A][00][84][A8][61]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)

[00][84] = value to be set in the register

[A8][61] = CRC

Response PDU

[01][06][00][2A][00][84][A8][61]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)

[00][84] = value set in the register

[A8][61] = CRC

The value 00 84 hex = 0000 0000 1000 0100 in binary notation is set in the **Control Word [0x2A]** variable (register 43). In other words, **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 [0x2A]** variable (register 43) of the Slave having the node address 1.

Request PDU

[01][06][00][2A][0A][80][AF][02]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)

[0A][80] = value to be set in the register

[AF][02] = CRC

Response PDU

[01][06][00][2A][0A][80][AF][02]

where:

[01] = Slave address

[06] = **06 Write Single Register** function code

[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)

[0A][80] = value set in the register

[AF][02] = CRC

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

4.4 Using the **16 Write Multiple Registers** function code



Example 1

Request to write the values 1500 and 1000 in the parameters **Acceleration [0x06]** (register 7) and **Deceleration [0x07]** (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 [0x06]** 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 [0x06]**, 05 DC hex = 1000 dec

[03][E8] = value to be set in the register 8 **Deceleration [0x07]**, 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 [0x06]** 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 [0x06]** parameter (register 7); the value 03 E8 hex, i.e. 1000 in decimal notation, is set in the **Deceleration [0x07]** parameter (register 8).

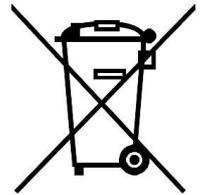
This page intentionally left blank

This page intentionally left blank



HW-SW release	Document release	Description
1-1	1.0	First issue
1-2 1-3 1-4	1.1	Added "Jog step" function in Control Word and "Jog step length" parameter. Updated jog entries. Updated jog buttons entry. Updated "Axis enabled" entry. Updated EDS file (V2) and Modbus.exe file (V2.2/V2.3/V2.4).
2-5	1.2	Updated information about brake, updated EDS file (V3) and Modbus.exe file (V2.5).
3-0	1.3	Updated section "Electrical connections". Updated "KP current loop" entry. Updated EDS file.
3-1 3-2	1.4	-T92 reduction gear model. EDS file for -T92 model. Updated Modbus.exe file (V2.7).
3-2	1.5	Updated information about objects 1400 Receive PDO Communication Parameter 1, 1800 Transmit PDO Communication Parameter 1, 3110-00 Cyclic Time
3-2	1.6	Updated information about LEDs
3-2	1.7	Warning against back EMF, Modbus registers Position window [0x01], Max following error [0x03], Jog step length [0x19] 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