

## RD1A RD12A



- Rotary actuator with Profinet-RT / -IRT connectivity
- Supports Media Redundancy Protocol (MRP)
- Brushless motor, nominal torque 5 Nm
- 18-bit real multiturn absolute encoder
- Also with integrated brake, braking torque 17 Nm
- For change-over operations and automated positioning systems

#### Suitable for the following models:

- RD1A PT
- RD12A PT

#### General Contents

Safety summary	23
Identification	25
Mechanical installation	26
Electrical connections	30
Profinet interface	50
Modbus® interface	103
Default parameters list	167

This publication was produced by Lika Electronic s.r.l. 2018. 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](mailto:info@lika.it) for submitting your comments, suggestions and criticisms.

The logo for Lika Electronic s.r.l. features the word "lika" in a bold, lowercase, sans-serif typeface. The letters are black and have a modern, clean appearance.

# General contents

User's guide.....	1
General contents.....	3
Subject Index.....	9
Typographic and iconographic conventions.....	11
Preliminary information.....	12
Glossary of Profinet terms.....	13
List of Profinet abbreviations.....	18
Glossary of MODBUS terms.....	20
<b>1 Safety summary.....</b>	<b>23</b>
1.1 Safety.....	23
1.2 Electrical safety.....	23
1.3 Mechanical safety.....	24
<b>2 Identification.....</b>	<b>25</b>
<b>3 Mechanical installation.....</b>	<b>26</b>
<b>4 Electrical connections.....</b>	<b>30</b>
4.1 Ground connection (Figure 1 and Figure 2).....	31
4.2 Connectors (Figure 4 and Figure 5).....	31
4.2.1 Power supply connector.....	32
4.2.2 Profinet interface connectors (PORT 1 and PORT 2).....	32
4.2.3 Network configuration: topologies, cables, hubs, switches - Recommendations.....	33
4.2.4 MAC address and IP address.....	34
4.2.5 Line Termination.....	34
4.2.6 Inputs / output + MODBUS RS-232 service port.....	35
4.3 Diagnostic LEDs (Figure 4 and Figure 6).....	36
4.4 Screw plug for internal access (Figure 4 and Figure 7).....	39
4.5 Preset / Jog buttons (Figure 8).....	40
4.5.1 JOG + and JOG – buttons (Figure 8).....	40
4.5.2 PRESET button (Figure 8).....	41
<b>5 Quick reference.....</b>	<b>42</b>
<b>6 Functions.....</b>	<b>43</b>
6.1 Working principle.....	43
6.2 Movements: jog and positioning.....	44
Jog: speed control.....	44
Positioning: position and speed control.....	45
6.3 Digital inputs and output.....	46
6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta.....	47
<b>7 Profinet interface.....</b>	<b>50</b>
7.1 Network and communication settings.....	50
7.1.1 MAC address.....	50
7.2 Configuring the device using TIA PORTAL.....	50
7.2.1 Preliminary information.....	50
7.2.2 About TIA Portal.....	51
7.2.3 Project overview.....	51
7.2.4 Device view.....	54
7.2.5 Network view.....	55

7.2.6 Topology view.....	56
7.2.7 Installing the GSDML file.....	56
7.2.8 Adding a node to the project.....	59
7.2.9 Establishing the bus connection.....	60
7.2.10 Device name and IP address at delivery.....	61
7.2.11 Setting the device name and the IP address.....	61
7.2.12 Compiling and transferring the project.....	62
7.2.13 Establishing an online connection (Online mode).....	63
7.2.14 Closing an online connection.....	65
7.2.15 Diagnostics.....	65
7.2.16 Watch Table.....	66
7.2.17 Module parameters.....	69
7.2.18 Setting and reading parameter values.....	70
7.2.18.1 Reading and setting the 20 Preset parameter.....	70
7.2.18.2 Reading and setting the 01 Distance per revolution parameter.....	73
7.2.18.3 Reading the Current Position (Bytes 0 to 3) parameter.....	75
7.2.18.4 Setting the Target Position (Bytes 2 to 5) value.....	75
7.4 Cyclic Data Exchange channel.....	76
7.3.1 Controller → Device cyclic transmission.....	77
<b>Control Word (Bytes 0 and 1)</b> .....	77
Jog +.....	77
Jog -.....	78
Stop.....	78
Alarm reset.....	78
Incremental jog.....	79
Start.....	79
Emergency.....	79
Save parameters.....	79
Load default parameters.....	79
Setting the preset.....	80
Release axis torque.....	80
OUT 1.....	80
Brake disabled.....	81
<b>Target Position (Bytes 2 to 5)</b> .....	81
<b>Parameter number (Byte 6)</b> .....	82
<b>Parameter value (Bytes 7 ... 10)</b> .....	83
7.3.2 Device → Controller cyclic transmission.....	84
<b>Current Position (Bytes 0 to 3)</b> .....	84
<b>Current Speed [rpm] (Bytes 4 to 7)</b> .....	84
<b>Status Word (Bytes 8 and 9)</b> .....	84
Axis in position.....	84
Drive enabled.....	84
SW limit switch +.....	84
SW limit switch -.....	85
Alarm.....	85
Axis running.....	85
Executing a command.....	85
Target position reached.....	85
Button 1 Jog +.....	85
Button 2 Jog -.....	86
Button 3 Preset.....	86



PWM saturation.....	86
IN 1.....	86
IN 2.....	86
IN 3.....	86
<b>Alarms List (Bytes 10 and 11).....</b>	<b>87</b>
Machine data not valid.....	87
Flash memory error.....	87
Counting error.....	87
Following error.....	87
Axis not synchronized.....	87
Target not valid.....	87
Emergency.....	87
Overcurrent.....	87
Electronics Overtemperature.....	88
Motor Overtemperature.....	88
Undervoltage.....	88
Address not valid.....	88
Read-only.....	88
Hall sequence.....	88
Overvoltage.....	88
<b>Following error [pulse] (Bytes 12 to 15).....</b>	<b>89</b>
<b>Current [mA] (Bytes 16 to 19).....</b>	<b>89</b>
<b>Electronics Temperature [°C] (Byte 20).....</b>	<b>89</b>
<b>Motor Temperature [°C] (Byte 21)].....</b>	<b>89</b>
<b>Parameter Error List (Bytes 22 to 25).....</b>	<b>89</b>
<b>Parameter number (Byte 26).....</b>	<b>90</b>
<b>Parameter value (Bytes 27 to 30).....</b>	<b>91</b>
7.4 Profinet module parameters.....	92
01 Distance per revolution.....	92
02 Position window.....	93
03 Position window time.....	93
04 Max following error.....	93
05 Proportional gain.....	94
06 Integral gain.....	94
07 Acceleration.....	94
08 Deceleration.....	94
09 Positive delta.....	94
10 Negative delta.....	95
11 Jog speed.....	96
12 Work speed.....	97
13 Code sequence.....	97
14 Jog step length.....	98
20 Preset.....	98
21 Position Offset.....	99
22 Pos. Limit Switch [pulse].....	99
23 Neg. Limit Switch [pulse].....	99
24 HMS Serial Number.....	100
25 Software version.....	100
26 Hardware version.....	100
27 Gear ratio.....	101

<b>8 Modbus® interface.....</b>	<b>103</b>
8.1 Configuring the device using Lika's setting up software.....	103
8.2 "Serial configuration" page.....	105
8.3 "Operative mode" page.....	107
8.4 "Parameter" page.....	114
8.5 "Message monitor" page.....	116
8.6 "Test Lika" page.....	117
8.7 "Upgrade Firmware" page.....	117
8.7.1 If an installation issue occurs.....	119
8.8 Modbus Master / Slaves protocol principle.....	120
8.9 Modbus frame description.....	121
8.10 Transmission modes.....	123
8.10.1 RTU transmission mode.....	123
8.11 Function codes.....	125
8.11.1 Implemented function codes.....	125
<b>03 Read Holding Registers.....</b>	<b>125</b>
<b>04 Read Input Register.....</b>	<b>127</b>
<b>06 Write Single Register.....</b>	<b>130</b>
<b>16 Write Multiple Registers.....</b>	<b>132</b>
8.12 Programming parameters.....	135
8.12.1 Holding Register parameters.....	135
<b>Distance per revolution [0x00].....</b>	<b>136</b>
<b>Position window [0x01].....</b>	<b>137</b>
<b>Position window time [0x02].....</b>	<b>137</b>
<b>Max following error [0x03-0x04].....</b>	<b>137</b>
<b>Kp position loop [0x05].....</b>	<b>137</b>
<b>Ki position loop [0x06].....</b>	<b>137</b>
<b>Acceleration [0x07].....</b>	<b>138</b>
<b>Deceleration [0x08].....</b>	<b>138</b>
<b>Positive delta [0x09-0x0A].....</b>	<b>138</b>
<b>Negative delta [0x0B-0x0C].....</b>	<b>139</b>
<b>Jog speed [0x0D].....</b>	<b>140</b>
<b>Work speed [0x0E].....</b>	<b>140</b>
<b>Code sequence [0x0F].....</b>	<b>141</b>
<b>Offset [0x10-0x11].....</b>	<b>141</b>
<b>Preset [0x12-0x13].....</b>	<b>142</b>
<b>Jog step length [0x14].....</b>	<b>142</b>
<b>Extra commands register [0x29].....</b>	<b>143</b>
Control by PC.....	143
<b>Control Word [0x2A].....</b>	<b>143</b>
Jog +.....	143
Jog -.....	144
Stop.....	144
Alarm reset.....	144
Incremental jog.....	145
Start.....	145
Emergency.....	145
Watch dog enable.....	145
Save parameters.....	146
Load default parameters.....	146

Setting the preset.....	146
Release axis torque.....	147
OUT 1.....	147
Brake disabled.....	147
<b>Target position [0x2B-0x2C]</b> .....	147
8.12.2 Input Register parameters.....	150
<b>Alarms register [0x00]</b> .....	150
Machine data not valid.....	150
Flash memory error.....	150
Counting error.....	150
Following error.....	150
Axis not synchronized.....	150
Target not valid.....	151
Emergency.....	151
Overcurrent.....	151
Electronics Overtemperature.....	151
Motor Overtemperature.....	151
Undervoltage.....	151
Watch dog.....	151
Hall sequence.....	152
Overvoltage.....	152
<b>Status word [0x01]</b> .....	152
Axis in position.....	152
Drive enabled.....	153
SW limit switch +.....	153
SW limit switch -.....	153
Alarm.....	153
Axis running.....	153
Executing a command.....	153
Target position reached.....	153
Button 1 Jog +.....	154
Button 2 Jog -.....	154
Button 3 Preset.....	154
PWM saturation.....	154
IN 1.....	154
IN 2.....	155
IN 3.....	155
<b>Current position [0x02-0x03]</b> .....	155
<b>Current velocity [0x04]</b> .....	155
<b>Position following error [0x05-0x06]</b> .....	155
<b>Temperature value [0x07]</b> .....	156
<b>Wrong parameters list [0x08-0x09]</b> .....	156
<b>Motor voltage [0x0A]</b> .....	157
<b>Current value [0x0B]</b> .....	157
<b>Hall [0x0C]</b> .....	157
<b>Duty cycle [0x0D]</b> .....	157
<b>DIP switch baud rate [0x0E]</b> .....	157
<b>DIP switch node ID [0x0F]</b> .....	157
<b>SW Version [0x10]</b> .....	158
<b>HW Version [0x11]</b> .....	158
8.13 Exception codes.....	160

8.14 Programming examples.....	161
8.14.1 Using the 03 Read Holding Registers function code.....	161
8.14.2 Using the 04 Read Input Register function code.....	162
8.14.3 Using the 06 Write Single Register function code.....	164
8.14.4 Using the 16 Write Multiple Registers function code.....	166
<b>9 Default parameters list.....</b>	<b>167</b>

# Subject Index

## 0

01 Distance per revolution.....	92
02 Position window.....	93
03 Position window time.....	93
04 Max following error.....	93
05 Proportional gain.....	94
06 Integral gain.....	94
07 Acceleration.....	94
08 Deceleration.....	94
09 Positive delta.....	94

## 1

10 Negative delta.....	95
11 Jog speed.....	96
12 Work speed.....	97
13 Code sequence.....	97
14 Jog step length.....	98

## 2

20 Preset.....	98
21 Position Offset.....	99
22 Pos. Limit Switch [pulse].....	99
23 Neg. Limit Switch [pulse].....	99
24 HMS Serial Number.....	100
25 Software version.....	100
26 Hardware version.....	100
27 Gear ratio.....	101

## A

Acceleration [0x07].....	138
Address not valid.....	88
Alarm.....	85, 153
Alarm reset.....	78, 144
Alarms List (Bytes 10 and 11).....	87
Alarms register [0x00].....	150
Axis in position.....	84, 152
Axis not synchronized.....	87, 150
Axis running.....	85, 153

## B

Brake disabled.....	81, 147
Button 1 Jog +.....	85, 154
Button 2 Jog -.....	86, 154
Button 3 Preset.....	86, 154

## C

Code sequence [0x0F].....	141
Control by PC.....	143
Control Word (Bytes 0 and 1).....	77
Control Word [0x2A].....	143
Counting error.....	87, 150

Current [mA] (Bytes 16 to 19).....	89
Current Position (Bytes 0 to 3).....	84
Current position [0x02-0x03].....	155
Current Speed [rpm] (Bytes 4 to 7).....	84
Current value [0x0B].....	157
Current velocity [0x04].....	155

## D

Deceleration [0x08].....	138
DIP switch baud rate [0x0E].....	157
DIP switch node ID [0x0F].....	157
Distance per revolution [0x00].....	136
Drive enabled.....	84, 153
Duty cycle [0x0D].....	157

## E

Electronics Overtemperature.....	88, 151
Electronics Temperature [°C] (Byte 20).....	89
Emergency.....	79, 87, 145, 151
Executing a command.....	85, 153
Extra commands register [0x29].....	143

## F

Flash memory error.....	87, 150
Following error.....	87, 150
Following error [pulse] (Bytes 12 to 15).....	89

## H

Hall [0x0C].....	157
Hall sequence.....	88, 152
HW Version [0x11].....	158

## I

IN 1.....	86, 154
IN 2.....	86, 155
IN 3.....	86, 155
Incremental jog.....	79, 145

## J

Jog -.....	78, 144
Jog +.....	77, 143
Jog speed [0x0D].....	140
Jog step length [0x14].....	142

## K

Ki position loop [0x06].....	137
Kp position loop [0x05].....	137

## L

Load default parameters.....	79, 146
------------------------------	---------

## M

Machine data not valid.....	87, 150
Max following error [0x03-0x04].....	137
Motor Overtemperature.....	88, 151

Motor Temperature [°C] (Byte 21)].....	89
Motor voltage [0x0A].....	157
<b>N</b>	
Negative delta [0x0B-0x0C].....	139
<b>O</b>	
Offset [0x10-0x11].....	141
OUT 1.....	80, 147
Overcurrent.....	87, 151
Overvoltage.....	88, 152
<b>P</b>	
Parameter Error List (Bytes 22 to 25).....	89
Parameter number (Byte 26).....	90
Parameter number (Byte 6).....	82
Parameter value (Bytes 27 to 30).....	91
Parameter value (Bytes 7 ... 10).....	83
Position following error [0x05-0x06].....	155
Position window [0x01].....	137
Position window time [0x02].....	137
Positive delta [0x09-0x0A].....	138
Preset [0x12-0x13].....	142
PWM saturation.....	86, 154
<b>R</b>	
Read-only.....	88
Release axis torque.....	80, 147




<b>S</b>	
Save parameters.....	79, 146
Setting the preset.....	80, 146
Start.....	79, 145
Status Word (Bytes 8 and 9).....	84
Status word [0x01].....	152
Stop.....	78, 144
SW limit switch -.....	85, 153
SW limit switch +.....	84, 153
SW Version [0x10].....	158
<b>T</b>	
Target not valid.....	87, 151
Target Position (Bytes 2 to 5).....	81
Target position [0x2B-0x2C].....	147
Target position reached.....	85, 153
Temperature value [0x07].....	156
<b>U</b>	
Undervoltage.....	88, 151
<b>W</b>	
Watch dog.....	151
Watch dog enable.....	145
Work speed [0x0E].....	140
Wrong parameters list [0x08-0x09].....	156

# Typographic and iconographic conventions

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

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

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

	This icon, followed by the word <b>WARNING</b> , 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 <b>NOTE</b> , 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 <b>EXAMPLE</b> when instructions for setting parameters are accompanied by examples to clarify the explanation.

# Preliminary information

This guide is designed to provide the most complete information the operator needs to correctly and safely install and operate the **DRIVECOD rotary actuators RD1A and RD12A models with Profinet interface**.

RD1A and RD12A units are positioning devices which integrate into one system a brushless motor fitted with gearbox, drive, multiturn absolute encoder and position controller. RD1A and RD12A rotary actuators are designed to drive positioning systems and change-over applications. Typical uses are packaging lines, food processing and pharmaceutical industries, wood & metalworking machinery, paper machinery, material handling equipment, bending machines, filling and bottling plants, printing machines, mold changers, mobile stops, tool changers, spindle positioning devices, among others.

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.

RD1A and RD12A rotary actuators can be equipped with the following interfaces:

- RD1xA-x-xxx-**CB**-... = CANopen DS301 interface;
- RD1xA-x-xxx-**EC**-... = EtherCAT interface;
- RD1xA-x-xxx-**MB**-... = Modbus RTU (RS-485) interface;
- RD1xA-x-xxx-**PB**-... = Profibus-DP interface;
- RD1xA-x-xxx-**PL**-... = POWERLINK interface;
- RD1xA-x-xxx-**PT**-... = Profinet interface.

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

In the Modbus version the configuration of the DRIVECOD unit can be done through a software expressly developed and released by Lika Electronic in order to allow an easy set up of the device. The program is supplied for free and can be installed in any PC fitted with a Windows operating system (Windows XP or later). It allows the operator to set the working parameters of the device; control manually some movements and functions; and monitor whether the device is running properly. In the Profinet version configuration can be done using the same program through a **service RS-232 serial interface, in compliance with Modbus protocol**.

To make it easier to read the text, this guide can be divided into two main sections.

In the first section general information concerning the safety, the mechanical installation and the electrical connection as well as tips for setting up and running properly and efficiently the unit are provided.

In the second section, entitled **Profinet Interface**, both general and specific information is given on the Profinet 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, Profinet version is equipped with a service RS-232 serial interface, 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 Profinet network. In the **Modbus Interface** section the interface features and the registers implemented in the unit are fully described.



# Glossary of Profinet terms

Profinet, like many other networking systems, has a set of unique terminology. Table below contains a few of the technical terms used in this guide to describe the Profinet interface. Sometimes they also refer more specifically to the S7 programming environment. They are listed in alphabetical order.

<b>Acyclic Communications</b>	Unscheduled, on demand communications. Diagnostic messages from an IO Supervisor to an IO Device are Acyclic.
<b>AP</b>	Application Process - The application process running in the device. PROFINET supports a default Application Processes and additional profile specific application processes.
<b>API</b>	The value of the API (Application Process Identifier) parameter specifies the application that is processing the IO data. PROFINET standard IEC 61158 assigns profiles to certain APIs (PROFIdrive, PROFIslave) which are defined by the PROFINET User Organization. The standard API is 0.
<b>Application class</b>	An application class specifies a number of mandatory functions and addition optional functions to be supported by an IO device. For instance, Profinet encoders can be configured as CLASS 3 and CLASS 4 PROFINET IO devices according to the encoder profile.
<b>AR</b>	Application Relation - The relationship between a PROFINET IO Controller and an IO device. A PROFINET IO device can support more than one Application Relationship.
<b>Bus</b>	A bus is a communication medium connecting several nodes. Data can be transferred via serial or parallel circuits, that is, via electrical conductors or fiber optic.
<b>Channel</b>	A single IO point. A Channel can be discrete or analog.
<b>Consumer Status</b>	The Status an IO device provides to an IO Controller for the data it consumes from IO Controller.
<b>CR</b>	Communication Relationship - A virtual communication channel within an AR.
<b>Cyclic Communications</b>	Scheduled, repetitive communications. IO data and alarm transfers are cyclic.
<b>Data block</b>	In contrast to code blocks, data blocks (DB) do not contain Step 7 statements. They are used to save data, i.e. variable data which are processed by the user program. Global data blocks serve to accommodate user data which can be used by all other blocks.
<b>DCP</b>	Discovery Control Protocol - A communications protocol with PROFINET IO that allows an IO Controller or Supervisor to find every PROFINET IO device on a subnet.
<b>Determinism</b>	Determinism means that a system responds in a predictable

	(deterministic) manner.
<b>Device name</b>	Before an IO device can be addressed by an IO controller, it must have a device name. In PROFINET, this method was selected because it is simpler to work with names than with complex IP addresses.
<b>Encoder Profile</b>	The PROFINET profile for Encoders is intended to define a standard application interface for encoders. The profile is a supplement to the PROFIdrive profile, so it is mandatory to read the PROFIdrive profile before implementing the encoder profile. Profinet encoders from Lika Electronic comply with the Encoder Profile Specifications V4.1 version 3.162. See also "Profile".
<b>Function</b>	Functions (FC) are code blocks which can be programmed by the user. A FC does not have a "memory". Temporary variables as well as parameters transferred to the function when the latter is called are saved in a L stack. They are lost following processing of the FC.
<b>Function block</b>	Function blocks (FB) are code blocks with a "memory" which are programmed by the user. They have an assigned instance data block (instance DB) as memory. Parameters transferred to a FB as well as the static variables are saved in this data block. An FB contains a program which is always executed when the FB is called by another code block. Function blocks facilitate the programming of frequently repeated, complex functions.
<b>Frame ID</b>	The two-byte field in the Ethernet frame which defines the type of PROFINET IO message.
<b>GSD</b>	The properties of a PROFINET device are described in a GSD file (General Station Description) that contains all the information required for configuration. In PROFINET IO, the GSD file is in XML format. The structure of the GSD file conforms to ISO 15734, which is the world-wide standard for device descriptions.
<b>GSDML</b>	General Station Description Markup Language – The file containing the XML description of the PROFINET IO device.
<b>IO Controller</b>	Device used to address the connected IO devices. This means that the IO controller exchanges input and output signals with assigned field devices. The IO controller is often the controller on which the automation program runs.
<b>IO Device</b>	A decentralized field device that is assigned to one of the IO controllers (e.g. remote IO, encoders, valve terminals, frequency converters, switches, etc.).
<b>IO Parameter Server</b>	An IO Parameter Server is a server station, usually a PC, for loading and saving the configuration data (records) of IO Devices.
<b>IO Supervisor</b>	Programming device, PC or HMI device used for commissioning and diagnostics of IO Controllers and IO Devices.

<b>IP address</b>	The IP address is the name of the unit in a network using the Internet protocol.
<b>IRT</b>	Synchronized transmission procedure for the cyclic exchange of IRT data between PROFINET devices. A reserved bandwidth within the send clock is available for the IRT IO data. The reserved bandwidth ensures that the IRT data can be transmitted at reserved, synchronized intervals whilst remaining uninfluenced even by other greater network loads (e.g. TCP/IP communication or additional real time communication). The "high flexibility" enables simple planning and expansion of the system. A topological configuration is not required.
<b>MAC address</b>	The MAC address is an identifier unique worldwide consisting of two parts: the first 3 bytes are the manufacturer ID and are provided by IEE standard authority; the last three bytes represent a consecutive number of the manufacturer.
<b>Module</b>	Modules are user defined components that plug into slots. Modules can be real or virtual.
<b>MRP (Media Redundancy Protocol)</b>	The Media Redundancy Protocol (MRP) enables implementing a redundant PROFINET communication through ring topology without the need for switches. MRP uses the basic principles of the Hiper ring. This ring redundancy protocol was developed by Hirschmann and Siemens and first presented in 1999. Since 2008, MRP is defined in the IEC 62439 standard. MRP is integrated into the PROFINET devices, so that no additional switches are needed. MRP can compensate an individual failure in a PROFINET / Industrial Ethernet in a simple ring topology. As meshed networks are not supported, MRP is both simple and deterministic. MRP in PROFINET networks can achieve reconfiguration times of just 200 ms.
<b>NRT</b>	Non Real Time - The non Real Time PROFINET IO Channel. Configuration and diagnostic messages are transferred over the NRT Channel.
<b>Organization block</b>	A range of organization blocks (OB) are designed to execute the user program. OBs are the interface between the user program and the operating system of a CPU. They permit event-controlled processing of special program components within the user program. The order in which the user program is executed is defined in the organization blocks.
<b>Profile</b>	Profiles define application-specific functionality to ensure the openness of PROFIBUS and PROFINET is utilized consistently. PI Profiles can cover simple devices such as encoders by defining how signals are used and how they are physically connected. However, profiles are increasingly covered more complex systems or requirements. Profiles such as PROFIdrive and PROFIsafe deliver active functionality as well. An advanced profile covering active power management for end devices like lasers and robots is now under development with the aim of

	bringing significant reductions in energy consumption for the automotive industry. Profiles guarantee quicker system design and they support faster device interchange, promoting competition amongst vendors, increased choice for users and full interoperability.
<b>Provider Status</b>	The Status an IO device provides to an IO Controller with the data transferred to the Controller.
<b>Proxy</b>	A device which maps non PROFINET IO data to PROFINET.
<b>Real-time</b>	Real-time means that a system processes external events within a defined time. If the reaction of a system is predictable, one speaks of a deterministic system. The general requirements for real-time are therefore: deterministic response and defined response time.
<b>RT</b>	Real Time - The Real Time PROFINET IO Channel. I/O and Alarm Data are transferred over the RT Channel.
<b>Slot</b>	A group of one or more Subslots. Slots can be real or virtual.
<b>Standard signal</b>	The encoder profile defines a series of standard signals which are used to configure the IO data.
<b>Submodule</b>	A component of a module that is plugged into a subslot. A submodule is real or virtual.
<b>Subslot</b>	A group of one or more channels. Subslots can be real or virtual.
<b>Sync domain</b>	All PROFINET devices that are to be synchronized via PROFINET IO with IRT must belong to a sync domain. The sync domain consists of precisely one sync master and at least one sync slave. IO controllers and switches can hold the role of a sync master or sync slave. Other IO devices support only the role as sync slave.
<b>System function</b>	System functions (SFC) are integral functions in the operating system of a S7 CPU. In addition, SFCs are frequently called implicitly by SFBs. SFCs can be called by the user program like normal functions. SFCs are used to implement a number of important system functions for Profinet IO.
<b>System function block</b>	System function blocks (SFB) are integral functions in the operating system of a S7 CPU. SFBs can be called by the user program like normal function blocks. SFBs are used to implement a number of important system functions for Profinet IO.
<b>TCP/IP</b>	<p>The Ethernet system is designed solely to carry data. It is comparable to a highway as a system for transporting goods and passengers. The data is actually transported by protocols. This is comparable to cars and commercial vehicles transporting passengers and goods on the highway.</p> <p>Tasks handled by the basic Transmission Control Protocol (TCP) and Internet Protocol (IP) (abbreviated to TCP/IP):</p> <ol style="list-style-type: none"> <li>1. The sender splits the data into a sequence of packets.</li> </ol>

	<ol style="list-style-type: none"> <li>2. The packets are transported over the Ethernet to the correct recipient.</li> <li>3. The recipient reassembles the data packets in the correct order.</li> <li>4. Faulty packets are sent again until the recipient acknowledges that they have been transferred successfully.</li> </ol>
<b>Telegram</b>	A telegram is a rigidly defined bit stream carrying data. A telegram specifies the data length and the type of data which is sent to and from the IO controller. The encoder profile supports Standard Telegrams 81, 82, 83 and 84.
<b>Topology</b>	<p>Network structure. Commonly used structures:</p> <ul style="list-style-type: none"> <li>• Line topology;</li> <li>• Ring topology;</li> <li>• Star topology;</li> <li>• Tree topology.</li> </ul>
<b>Transmission rate</b>	Data transfer rate (in bps).
<b>User program</b>	The user program contains all instructions, declarations and data for signal processing required to control a plant or a process. It is assigned to a programmable module (for example CPU) and can be structured in smaller units (blocks).

# List of Profinet abbreviations

Table below contains a list of abbreviations (in alphabetical order) which may be used in this guide to describe the Profinet interface. Sometimes they also refer more specifically to the S7 programming environment.

<b>AR</b>	Application Relation
<b>API</b>	Application Process Identifier
<b>C-LS</b>	Controller's Sign-Of-Life
<b>CR</b>	Communication Relation
<b>DB</b>	Data block
<b>DO</b>	Drive Object
<b>DO-LS</b>	Driver Object Sign-Of-Life
<b>DU</b>	Drive Unit
<b>EO</b>	Encoder Object
<b>EU</b>	Encoder Unit
<b>FB</b>	Function block
<b>FC</b>	Function
<b>I&amp;M</b>	Identification & Maintenance
<b>IRT</b>	Isochronous Real Time Ethernet
<b>IRT Flex</b>	IRT "High Flexibility"
<b>IRT Top</b>	IRT "High Performance"
<b>GSDML</b>	General Station Description Markup Language
<b>IO</b>	Input/Output
<b>IP</b>	Internet Protocol
<b>LLDP</b>	Link Layer Discovery Protocol
<b>LS</b>	Sign-Of-Life
<b>MAC</b>	Media Access Control
<b>MAP</b>	Module Access Point
<b>MLS</b>	Master Sign-Of-Life
<b>MRP</b>	Media Redundancy Protocol
<b>OB</b>	Organization block
<b>PAP</b>	Parameter Access Point
<b>PI</b>	PROFIBUS and PROFINET International
<b>RT</b>	Real Time Ethernet

<b>SFB</b>	System function block
<b>SFC</b>	System function
<b>TCP</b>	Transmission Control Protocol
<b>T<sub>MAPC</sub></b>	Master Application Cycle Time

# Glossary of MODBUS terms

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

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



	<p>specified by function codes. The function code is sent from a Client to the Server and indicates which kind of action the Server must perform. MODBUS function codes are elements of MODBUS request/reply PDUs.</p> <p>The function code field of a MODBUS data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (the range 128 – 255 is reserved and used for exception responses). Function code "0" is not valid. Like actuators only implement public function codes.</p>
<b>Holding register</b>	In the MODBUS data model, a Holding register is the output data. A Holding register has a 16-bit quantity, is alterable by an application program, and allows either read-write or read-only access.
<b>IEEE 1588</b>	This standard defines a protocol enabling synchronisation of clocks in distributed networked devices (e.g. connected via Ethernet).
<b>Input register</b>	In the MODBUS data model, an Input register is the input data. An Input register has a 16-bit quantity, is provided by an I/O system, and allows read-only access.
<b>LRC Checking</b>	In ASCII mode, messages include an error-checking field that is based on a Longitudinal Redundancy Checking (LRC) calculation that is performed on the message contents, exclusive of the beginning 'colon' and terminating CRLF pair characters. It is applied regardless of any parity checking method used for the individual characters of the message.
<b>Master</b>	A Master is any network device that sends data requests to Slaves.
<b>Message</b>	<p>The MODBUS messaging service provides a Client/Server communication between devices connected on the network. The Client / Server model is based on four types of messages:</p> <ul style="list-style-type: none"> <li>• MODBUS Request</li> <li>• MODBUS Confirmation</li> <li>• MODBUS Indication</li> <li>• MODBUS Response</li> </ul> <p>The MODBUS messaging services are used for information exchange.</p>
<b>MODBUS Confirmation</b>	A MODBUS Confirmation is the Response Message received on the Client side.
<b>MODBUS Indication</b>	A MODBUS Indication is the Request message received on the Server side.
<b>MODBUS Request</b>	A MODBUS Request is the message sent on the network by the Client to initiate a transaction.
<b>MODBUS Response</b>	A MODBUS Response is the Response message sent by the Server.
<b>Network</b>	Network is a group of computers on a single physical network segment.

<b>PDU</b>	<p>The Protocol Data Unit (PDU) is the MODBUS function code and data field. It is packed together with the Address Field and the CRC (or LRC) to form the Modbus Serial Line PDU.</p> <p>The MODBUS protocol defines three PDUs. They are:</p> <ul style="list-style-type: none"> <li>• MODBUS Request PDU, mb_req_pdu</li> <li>• MODBUS Response PDU, mb_rsp_pdu</li> <li>• MODBUS Exception Response PDU, mb_excep_rsp_pdu</li> </ul>
<b>Read Holding Registers (03, 0003hex)</b>	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.
<b>Read Input Register (04, 0004hex)</b>	This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some result values and state / alarm messages in a remote device.
<b>Register</b>	MODBUS functions operate on memory registers to configure, monitor, and control device I/O.
<b>RTU transmission mode</b>	Remote Terminal Unit. When devices communicate on a MODBUS serial line using the RTU mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII mode for the same baud rate. Each message must be transmitted in a continuous stream of characters.
<b>Server</b>	<p>A Server is any program that awaits data requests to be sent to it. Servers do not initiate contacts with Clients, but only respond to them.</p> <p>MODBUS follows the Client/Server model. MODBUS Masters are referred to as clients, while MODBUS Slaves are servers.</p>
<b>Service request</b>	It is the MODBUS Request, i.e. the message sent on the network by the Client to initiate a transaction.
<b>Slave</b>	A Slave is any program that awaits data requests to be sent to it. Slaves do not initiate contacts with Masters, but only respond to them.
<b>Transmission rate</b>	Data transfer rate (in bps).
<b>Write Multiple Registers (16, 0010hex)</b>	This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
<b>Write Single Register (06, 0006hex)</b>	This function code is used to WRITE a single holding register in a remote device.

# 1 Safety summary



## 1.1 Safety

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



## 1.2 Electrical safety

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



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



### 1.3 Mechanical safety

- Install the device following strictly the information in the "Mechanical installation" section on page 26;
- 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.



#### WARNING

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

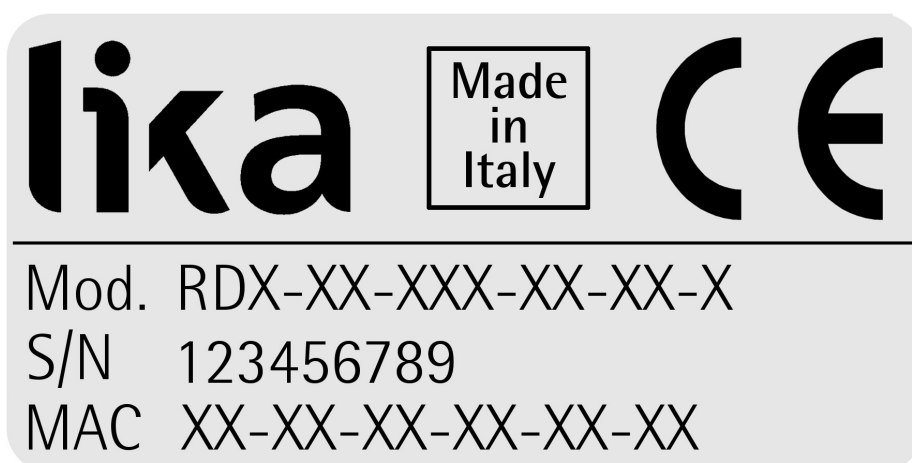


#### WARNING

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

## 2 Identification

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

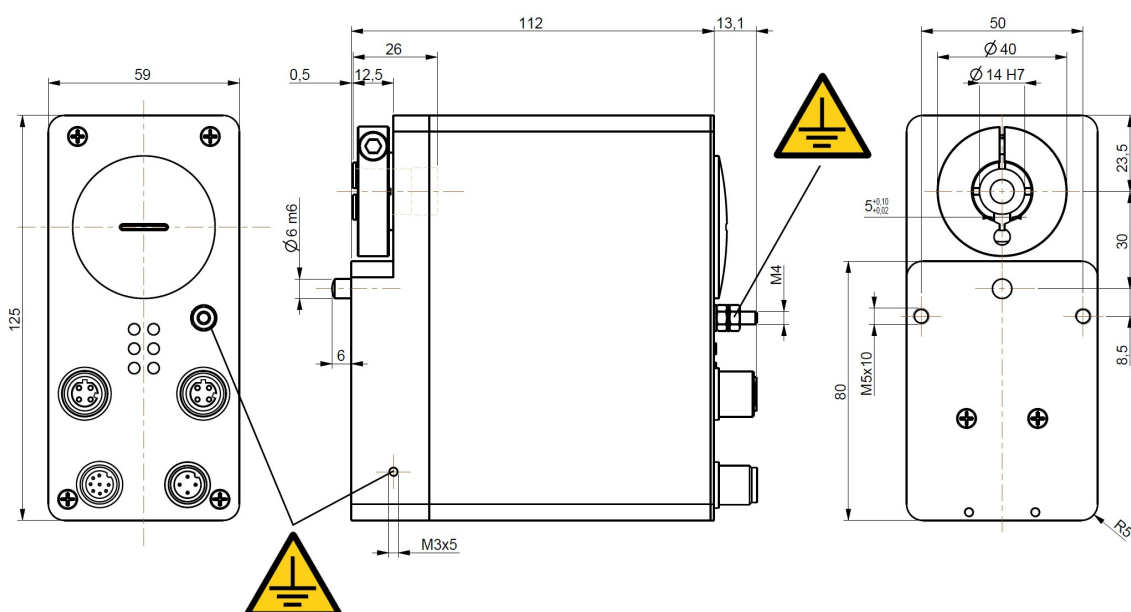


### 3 Mechanical installation



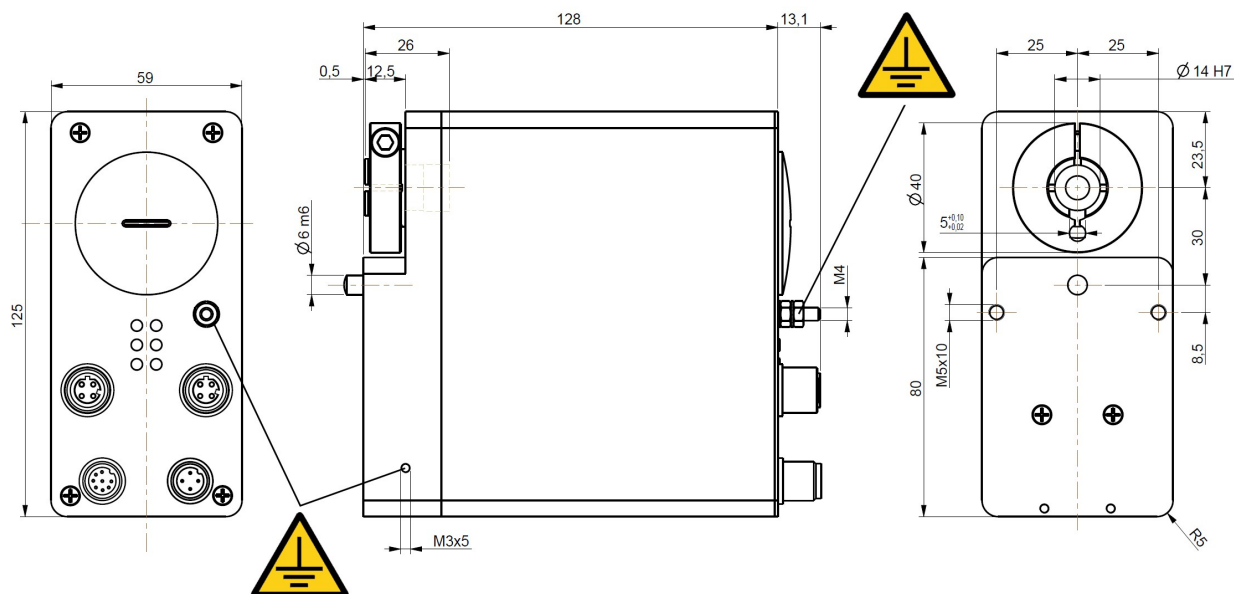
#### WARNING

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



(values are expressed in mm)

Figure 1 - RD1A unit – Overall dimensions



(values are expressed in mm)

Figure 2 - RD12A unit – Overall dimensions



DRIVECOD unit must be secured firmly only to the user's shaft using the provided collar. DRIVECOD 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 the 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!

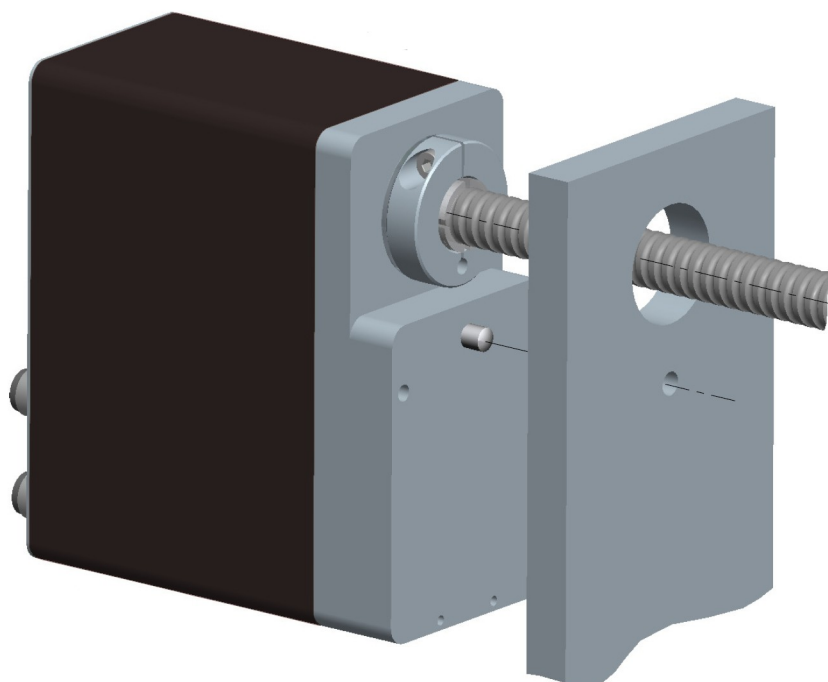
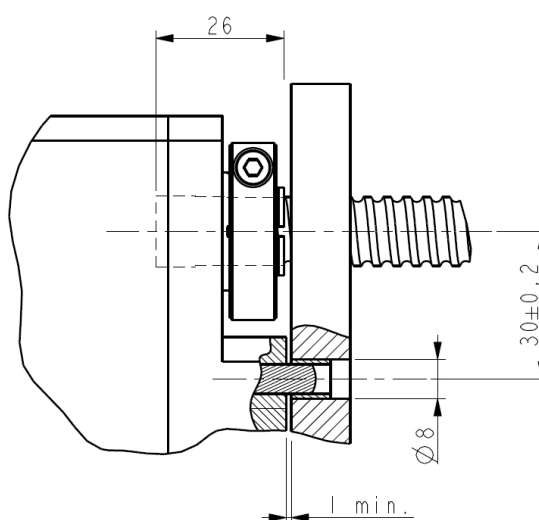


Figure 3 - Typical installation example of RD1xA unit on worm screw

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

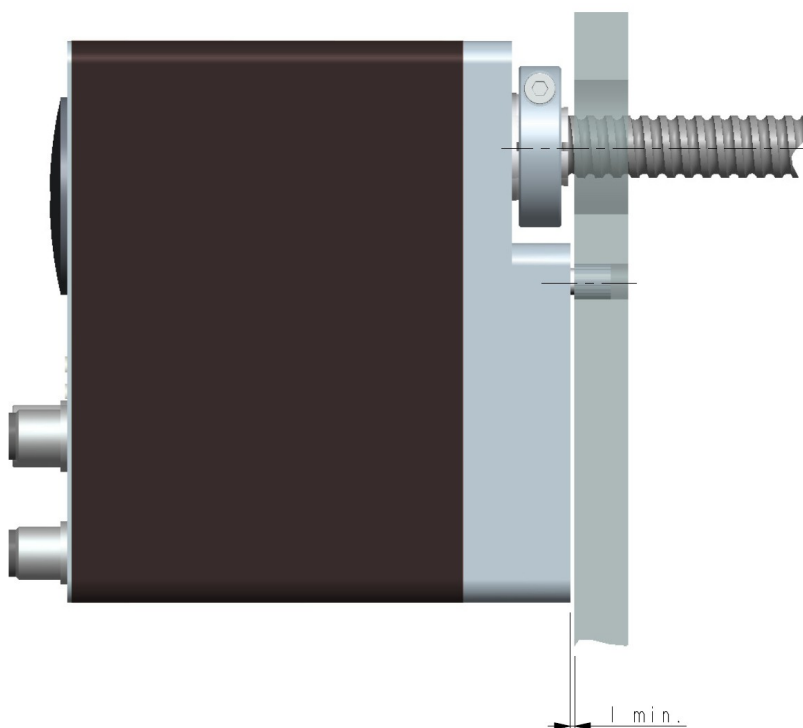
- Drill a 8 mm (0.315") diameter hole in the flange or in the fixed support on user's side in order to insert the silicone isolator and the anti-rotation pin. The distance between the axis of the shaft and the axis of the hole must be 30 mm (1.18")  $\pm$  0.2 mm (0.008"). Make sure that the hole and 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 DRIVECOD unit; the maximum depth of the DRIVECOD shaft is 26 mm (1.02"); ascertain that the anti-rotation pin is inserted properly in the silicone isolator;
- the minimum distance between the flange of the DRIVECOD unit and the fixed support on user's side must be not less than 1 mm (0.039") 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 the **Start**, **Jog +** or **Jog -** commands, the unit and the shaft start moving! Before operating please make sure that there are no risks of personal injury and mechanical damages.

Each **Start** routine has to be checked carefully in advance!

Never force manually the rotation of the shaft not to cause permanent damages!

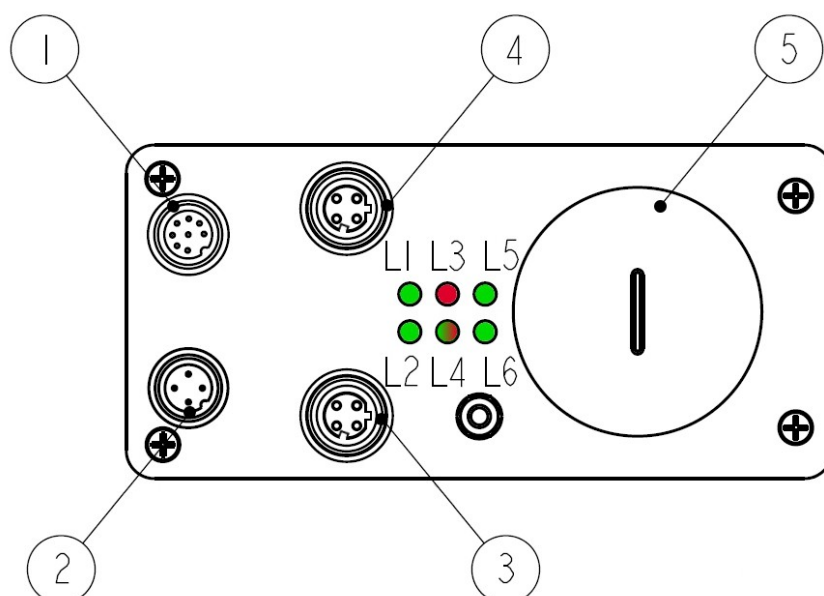


Figure 4: Connectors and diagnostic LEDs

### Legend

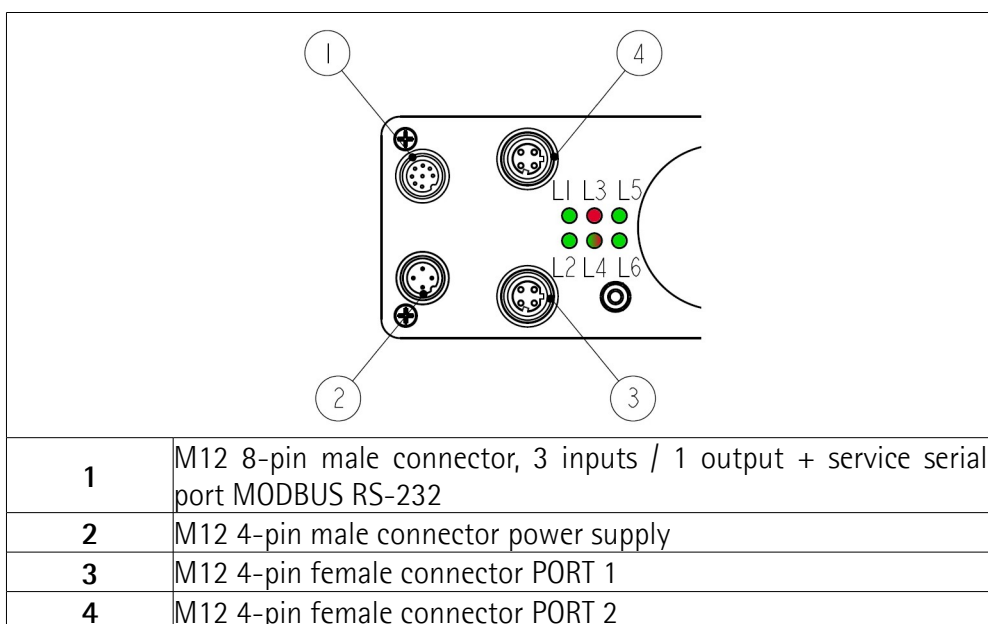
1	M12 8-pin male connector inputs / output + MODBUS RS-232
2	M12 4-pin male connector power supply
3	M12 4-pin female connector PORT 1
4	M12 4-pin female connector PORT 2
5	Internal housing of Preset / Jog buttons
L1	LED 1 Link / Activity in PORT 2
L2	LED 2 Link / Activity in PORT 1
L3	LED 3 Network Status

L4	LED 4 Module Status
L5	LED 5 Controller power supply information
L6	LED 6 Motor power supply information

#### 4.1 Ground connection (Figure 1 and Figure 2)

To minimize noise connect properly the frame to ground; we suggest using the ground screw points provided in the frame (see Figure 1 and Figure 2). Connect properly the cable shield to ground on user's side. Lika 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 E- connectors have a plastic gland, thus grounding is not possible. If metal connectors are used, connect the cable shield properly as recommended by the manufacturer. See also the note in the next paragraph. Anyway make sure that ground is not affected by noise. It is recommended to provide the ground connection as close as possible to the device.

#### 4.2 Connectors (Figure 4 and Figure 5)



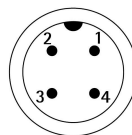
**Figure 5: Connectors**

#### 4.2.1 Power supply connector

##### Power supply

M12 4-pin male connector

(frontal side)



Pin	Description
1	motor +24Vdc $\pm$ 10% power supply
2	controller +24Vdc $\pm$ 10% power supply
3	motor and controller 0Vdc supply voltage
4	not connected

#### 4.2.2 Profinet interface connectors (PORT 1 and PORT 2)

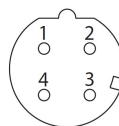
Two M12 4-pin female connectors with D coding are used for Ethernet network connection through PORT 1 and PORT 1.

##### Interface

M12 4-pin connectors

D coding, female

(frontal side)



Pin	Description
1	Tx Data +
2	Rx Data +
3	Tx Data -
4	Rx Data -
Case	Shielding <sup>1</sup>

<sup>1</sup> Lika EC- pre-assembled cables only

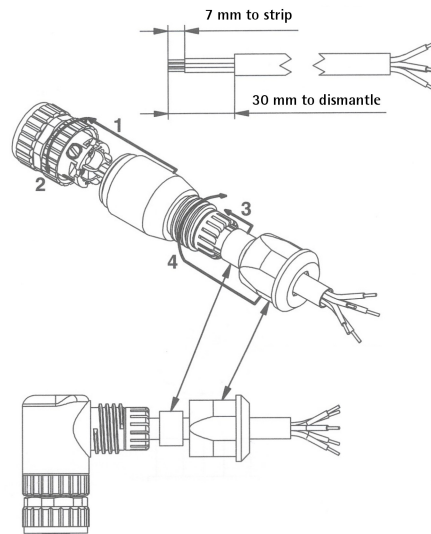
The ports are equal and interchangeable – if only one connection is required, either port can be used. The Ethernet interface supports 100 Mbit/s, full-duplex operation.



#### NOTE

We suggest always connecting the cable shield to ground on user's side.

Lika 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 E-connectors have a plastic gland, thus grounding is not possible (see the Figure). If metal connectors are used, connect the cable shield properly as recommended by the manufacturer.



#### 4.2.3 Network configuration: topologies, cables, hubs, switches – Recommendations

PROFINET is based on a 100 Mbps, full-duplex Ethernet network. Faster communication is also possible on all transmission sections (e.g., between switches, PC systems, or camera systems).

Using Ethernet several topologies of connection are supported by Profinet networks: line, tree, daisy-chain, star, ... Furthermore Profinet networks can be configured in almost any topology in the same structure.

The connection of PROFINET IO field devices occurs exclusively with switches as network components. Switches typically integrated in the field device are used for this (with 2 ports assigned). PROFINET-suitable switches must support "autonegotiation" (negotiating of transmission parameters) and "autocrossover" (autonomous crossing of send and receive lines).

Cables and connectors comply with the Profinet specifications. The cabling guide defines for all Conformance Classes a 2-pair cable according to IEC 61784-5-3.

Standard Profinet cables commercially available can be used.

The maximum segment length for electrical data transmission with copper cables between two nodes (field devices or switches) is 100 m. The copper cables are designed uniformly in AWG 22. The installation guide defines different cable types, whose range has been optimally adapted to general requirements for industry. Sufficient system reserves allow industry-compatible installation with no limitation on transmission distance.

The PROFINET cables conform to the cable types used in industry:

- PROFINET Type A: Standard permanently routed cable, no movement after installation

- PROFINET Type B: Standard flexible cable, occasional movement or vibration
- PROFINET Type C: Special applications: for example, highly-flexible, constant movement (trailing cable or torsion)

For complete information please refer to IEC 61918, IEC 61784-5-13 and IEC 61076-2-101.

To increase noise immunity only S/FTP or SF/FTP cables must be used (CAT-5).

The maximum cable length (100 meters) predefined by Ethernet 100Base-TX must be compulsorily fulfilled.

Regarding wiring and EMC measures, the IEC 61918 and IEC 61784-5-13 must be considered.

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

#### 4.2.4 MAC address and IP address

The unit can be identified in the network through the **MAC address**, the **IP address** and the **device name**. MAC address has to be intended as a permanent and globally unique identifier assigned to the unit for communication on the physical layer; while the IP address is the name of the unit in a network using the Internet protocol. MAC address is 6-byte long and cannot be modified. It consists of two parts, numbers are expressed in hexadecimal notation: the first three bytes are used to identify the manufacturer (OUI, namely Organizationally Unique Identifier), while the last three bytes are the specific identifier of the unit. The MAC address can be found on the label applied to the encoder. The IP address (and the subnet mask) must be assigned by the user to each interface of the unit to be connected in the network. For additional information on the MAC address, the IP address and the device name refer to the "7.1 Network and communication settings" section on page 50.

#### 4.2.5 Line Termination

Profinet network needs no line termination because the line is terminated automatically; in fact every Device is able to detect the presence of the downstream Devices.

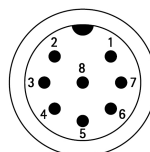
#### 4.2.6 Inputs / output + MODBUS RS-232 service port

##### Inputs / output + MODBUS RS-232 service port

M12 8-pin connector

A coding, male

(frontal side)



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

For any information on the three digital inputs and the digital output please refer to the "6.3 Digital inputs and output" section on page 46.

The configuration parameters of the 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

The MODBUS address is 1. It cannot be changed. See the "8.2 "Serial configuration" page" section on page 105.

For any further information on configuring and using the RS-232 service serial port refer to the "Modbus® interface" section on page 103.

### 4.3 Diagnostic LEDs (Figure 4 and Figure 6)

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

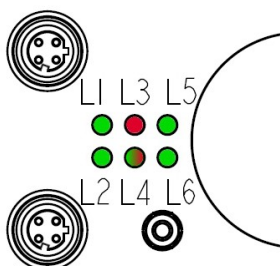


Figure 6: Diagnostic LEDs

<b>L1</b>	Link / Activity in PORT 2	<b>L4</b>	Module State
<b>L2</b>	Link / Activity in PORT 1	<b>L5</b>	Controller power supply information
<b>L3</b>	Network State	<b>L6</b>	Motor power supply information



#### NOTE

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



LED L1 GREEN		Description
LED state		
OFF	No Link	<ul style="list-style-type: none"> <li>There is neither network link nor communication</li> <li>The cable is disconnected</li> </ul>
ON	Link	Link to the Ethernet network established properly through port 2, but there is no communication
FLICKERING	Activity	Link to the Ethernet network established properly through port 2, communication is present, incoming and outgoing traffic through port 2 (data exchange)



LED L2 <b>GREEN</b>		It shows the state of the physical link and the activity in PORT 1 (connector 3)
LED state	Description	
OFF	No Link	<ul style="list-style-type: none"> <li>There is neither network link nor communication</li> <li>The cable is disconnected</li> </ul>
ON	Link	Link to the Ethernet network established properly through port 1, but there is no communication
FLICKERING	Activity	Link to the Ethernet network established properly through port 1, communication is present, incoming and outgoing traffic through port 1 (data exchange)

LED L3 NETWORK STATUS		It shows the state of the network connection
LED state	Description	
OFF	Offline	<ul style="list-style-type: none"> <li>There is no power supply</li> <li>There is no connection with IO Controller</li> </ul>
Lit <b>green</b>	Online (RUN)	<ul style="list-style-type: none"> <li>The connection with the IO Controller has been established properly</li> <li>The IO Controller is in RUN state</li> </ul>
Single flash <b>green</b>	Online (STOP)	<ul style="list-style-type: none"> <li>The connection with the IO Controller has been established properly</li> <li>The IO Controller is in STOP state</li> <li>IO data is corrupted</li> <li>IRT synchronization not finished</li> </ul>
Blinking <b>green</b>	Blink	<ul style="list-style-type: none"> <li>Used by engineering tools to identify the node on the network</li> </ul>
<b>Red</b>	Fatal event	<ul style="list-style-type: none"> <li>Major internal error (this indication is combined with a red module status LED)</li> </ul>
Single flash <b>red</b>	Station Name error	<ul style="list-style-type: none"> <li>Station name not set</li> </ul>
Double flash <b>red</b>	IP address error	<ul style="list-style-type: none"> <li>IP address not set</li> </ul>
Triple flash <b>red</b>	Configuration error	<ul style="list-style-type: none"> <li>Expected identification differs from Real Identification</li> </ul>

LED L4 MODULE STATUS		It shows the state of the Profinet module
LED state	Description	
OFF	Not initialized	<ul style="list-style-type: none"> <li>There is no power supply</li> <li>The module is in SETUP state</li> <li>The module is in NW_INIT state</li> </ul>
Lit <b>green</b>	Normal operation	<ul style="list-style-type: none"> <li>The module has shifted from the NW_INIT state</li> </ul>
Single flash	Diagnostic	<ul style="list-style-type: none"> <li>Diagnostic event(s) present</li> </ul>

green	event(s)	
Red	Exception error	<ul style="list-style-type: none"> <li>The device is in EXCEPTION state</li> </ul>
	Fatal event	<ul style="list-style-type: none"> <li>Major internal error (this indication is combined with a red network status LED)</li> </ul>
Alternating red / green	Firmware update	<ul style="list-style-type: none"> <li>Do NOT power off the module. Turning the module off during this phase could cause permanent damage</li> </ul>

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

LED L6 GREEN	It shows whether the power supply of the motor is switched on
ON	It indicates that the power supply of the motor is turned on
OFF	It indicates that the power supply of the motor is turned off



LED L3 GREEN	<b>Description</b>
	It shows whether the actuator is switched on
ON	It indicates that the power supply of the actuator is turned on
OFF	It indicates that the power supply of the actuator is turned off

LED L4 GREEN	<b>Description</b>
ON	It indicates that the firmware upgrade operation is being performed
OFF	It indicates that there is no firmware upgrade operation in progress

LED L5 GREEN	<b>Description</b>
	It shows whether the power supply of the controller is switched on
ON	It indicates that the power supply of the controller is turned on

OFF	It indicates that the power supply of the controller is turned off
-----	--

LED L6 GREEN	Description
	It shows whether the power supply of the motor is switched on
ON	It indicates that the power supply of the motor is turned on
OFF	It indicates that the power supply of the motor is turned off

#### 4.4 Screw plug for internal access (Figure 4 and Figure 7)



##### WARNING

Please be careful: the power supply is ON when you need to operate the Preset / Jog buttons!



##### NOTE

When you perform this operation please be careful not to damage the internal connections.

To access the Preset / Jog buttons unscrew and pull out the screw plug (PG 29 thread) in the back of the enclosure. Be careful to always replace the screw plug at the end of the operation.

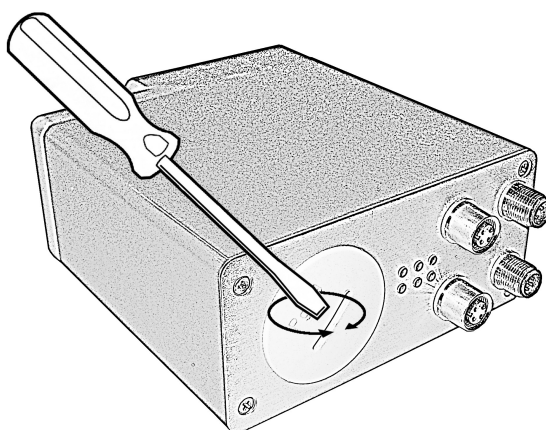


Figure 7: Screw plug for internal access

#### 4.5 Preset / Jog buttons (Figure 8)

The Preset / Jog buttons are located just beneath the screw plug.

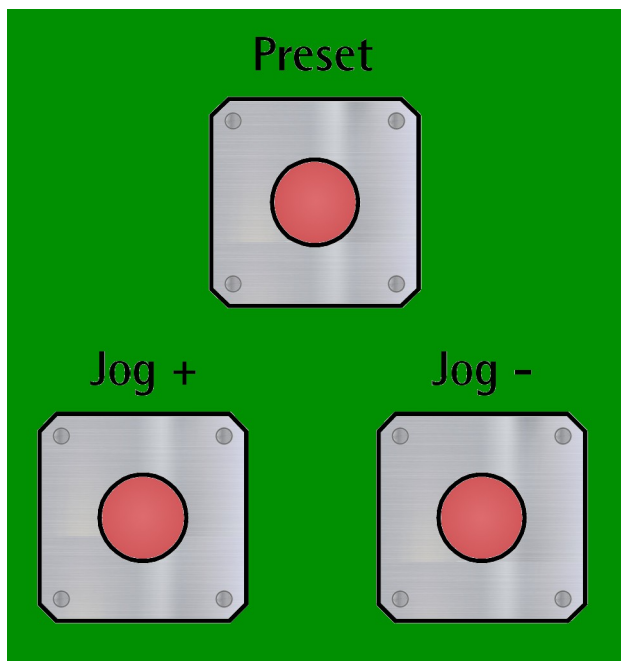


Figure 8: Preset / Jog buttons

##### 4.5.1 JOG + and JOG – buttons (Figure 8)

Press these buttons to force the manual movements of the motor toward the positive direction (**Jog +**) or toward the negative direction (**Jog -**). For any further information see the **Jog +** and the **Jog -** commands on page 77 ff (Profinet interface) or on page 143 ff (Modbus interface).



#### WARNING

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



#### NOTE

Please note that when you use the manual buttons the "incremental jog" function (see **Incremental jog** in **Control Word (Bytes 0 and 1)** on page 79

-Profinet version; **Incremental jog** in **Control Word [0x2A]** on page 145 (Modbus version) is disabled; that is, the jog step movements are not allowed using the manual buttons. Thus the positive and negative movements are commanded only by keeping pressed the buttons continuously and come to an end when the buttons are released.

**NOTE**

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

**4.5.2 PRESET button (Figure 8)**

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 3 seconds at least. We suggest setting the preset when the actuator is in stop. For any further information on the preset function see the **Preset** item on page 98 (Profinet interface) or on page 142 (Modbus interface).

**WARNING**

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

## 5 Quick reference

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

- Mechanically install the device;
- execute the electrical connections;
- switch on the +24Vdc power supply (in both the motor and the controller);
- check the operating condition shown through the LEDs;
- to resume the normal work condition reset the active emergency: switch high ("=1") the **Emergency** bit 7 of the **Control Word** (see on page 77 -Profinet interface; see on page 143 -MODBUS interface); reset the active alarms: switch high ("=1") the **Alarm reset** bit 3 of the **Control Word** (see on page 77 -Profinet interface; see on page 143 -MODBUS interface). Check the operating condition shown through the LEDs;
- set a proper value next to the **Distance per revolution** item (see on page 92 -Profinet interface; see on page 136 -MODBUS interface);
- set a proper value next to the **Jog speed** item (see on page 96 -Profinet interface; see on page 140 -MODBUS interface);
- set a proper value next to the **Work speed** item (see on page 97 -Profinet interface; see on page 140 -MODBUS interface);
- if required, set a proper value next to the **Preset** item (see on page 98 -Profinet interface; see on page 142 -MODBUS interface);
- set the limit switch values next to the **Max delta pos / Positive delta** and **Max delta neg / Negative delta** items (see on page 94 ff -Profinet interface; see on page 138 ff -MODBUS interface);
- set the commanded position next to the **Target position** item (see on page 81 -Profinet interface; see on page 147 -MODBUS interface);
- save the new setting values (**Save parameters** command; see on page 79 -Profinet interface; see on page 146 -MODBUS interface).

Use the **Jog +**, **Jog -**, **Start** and **Stop** commands in the **Control Word** (see on page 77 -Profinet interface; see on page 143 -MODBUS interface) to move the axis and reach the commanded position.



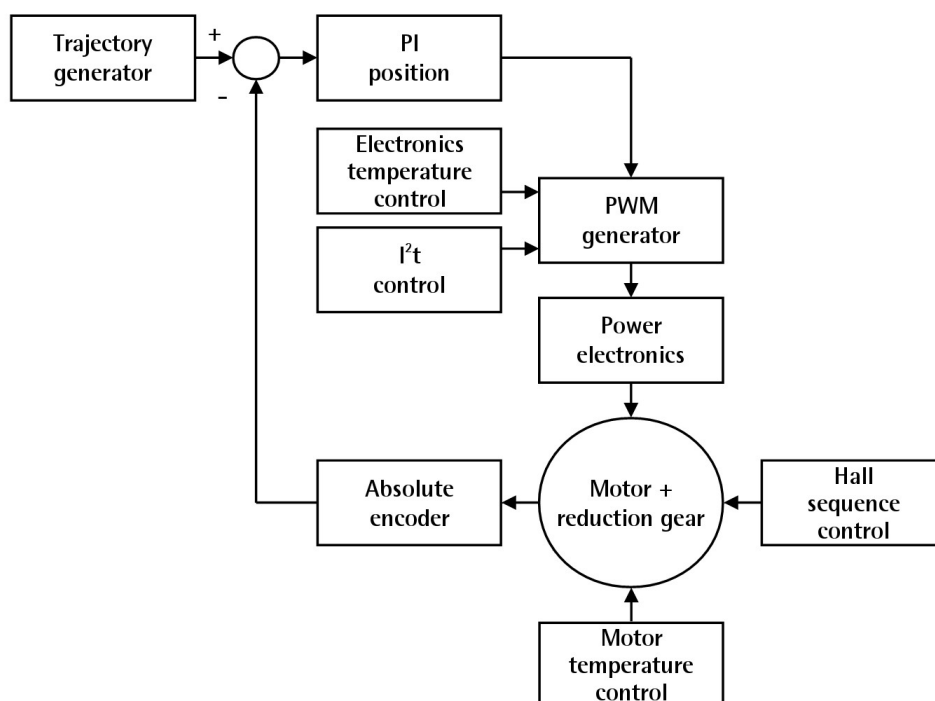
### NOTE

The parameters **Distance per revolution**, **Jog speed**, **Work speed**, **Preset**, **Max delta pos / Positive delta** and **Max delta neg / Negative delta** are closely related, hence you have to be very attentive when you need to change the value in any of them. For any further information please refer to page 47.

## 6 Functions

### 6.1 Working principle

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



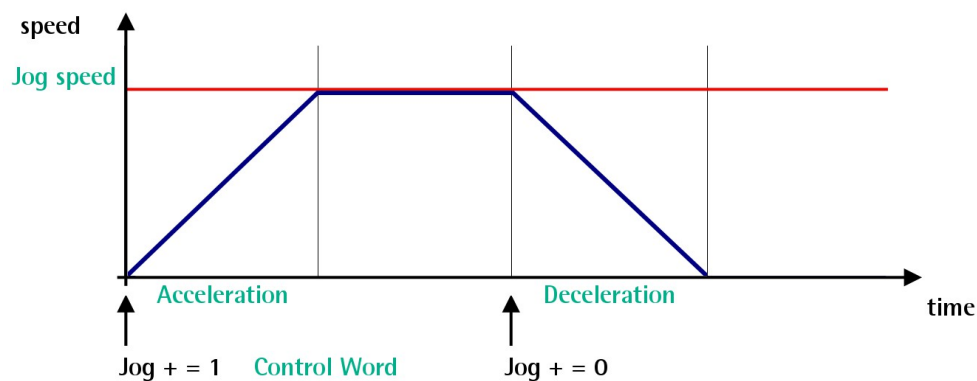
## 6.2 Movements: jog and positioning

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

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

### Jog: speed control

This kind of control is intended to generate a speed trajectory which allows the rotational speed of the DRIVECOD unit shaft to be equal to the value set in the **11 Jog speed / Jog speed [0x0D]** parameter.



When the bit 0 **Jog +** in the **Control Word (Bytes 0 and 1) / Control Word [0x2A]** is "1", the motor accelerates toward the positive direction according to the value set next to the **07 Acceleration / Acceleration [0x07]** item; if the available travel is long enough it reaches the speed set next to the **11 Jog speed / Jog speed [0x0D]** item. As soon as the bit 0 **Jog +** in the **Control Word (Bytes 0 and 1) / Control Word [0x2A]** goes low ("0"), the motor decelerates according to the value set next to the **08 Deceleration / Deceleration [0x08]** item until it stops.

Setting the bit 1 **Jog -** in the **Control Word (Bytes 0 and 1) / Control Word [0x2A]** to "1" causes the motor to run in the opposite direction (negative direction) respecting the work phases already described above.



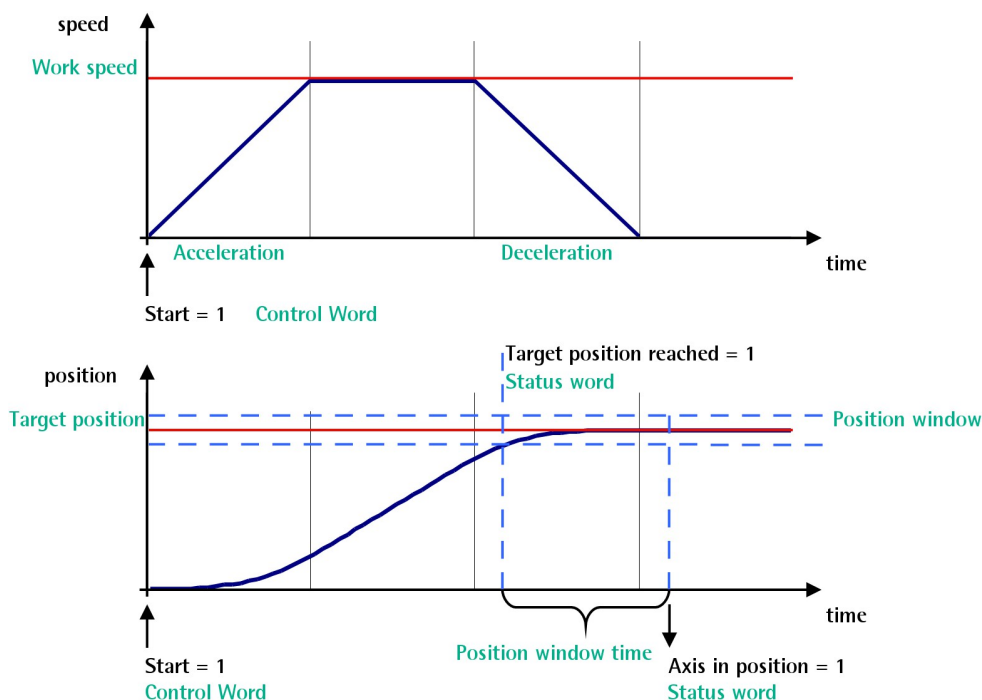
#### NOTE

Please note that the value in the **11 Jog speed / Jog speed [0x0D]** parameter is the speed of the motor, not the speed of the output shaft after the reduction gears, see on page 101.



## Positioning: position and speed control

This kind of control is a point-to-point movement and the maximum reachable speed is equal to the value set in the **12 Work speed / Work speed [0x0E]** parameter; the set speed can be reached only if the available travel is long enough.



When the bit 6 **Start** in the **Control Word (Bytes 0 and 1) / Control Word [0x2A]** is "1", the motor starts moving and accelerates according to the value set next to the **07 Acceleration / Acceleration [0x07]** item in order to reach the target position as set next to the **Target Position (Bytes 2 to 5) / Target position [0x2B-0x2C]** item. If the available travel is long enough it reaches the speed set next to the **12 Work speed / Work speed [0x0E]** item. The movement direction can be either positive or negative according to the target position to reach. As soon as the axis is within the tolerance window limits set next to the **02 Position window / Position window [0x01]** item, the bit 8 **Target position reached** in the **Status Word (Bytes 8 and 9) / Status word [0x01]** goes high ("1"). When the position is within the tolerance window limits set next to the **02 Position window / Position window [0x01]** item, after the delay set next to the **03 Position window time / Position window time [0x02]** item, the bit 0 **Axis in position** in the **Status Word (Bytes 8 and 9) / Status word [0x01]** goes high ("1"). The motor decelerates according to

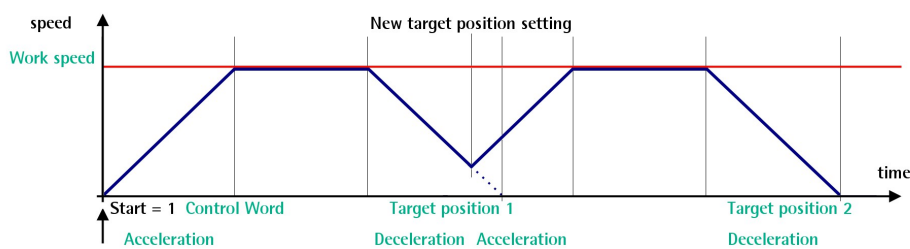
the value set next to the **08 Deceleration / Deceleration [0x08]** item in order to reach the halt position according to the set target position.



**NOTE**

**Position override function**

It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the **Target Position (Bytes 2 to 5) / Target position [0x2B-0x2C]** item.



**NOTE**

Please note that the value in the **12 Work speed / Work speed [0x0E]** parameter is the speed of the motor, not the speed of the output shaft after the reduction gears, see on page 101.

### 6.3 Digital inputs and output

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

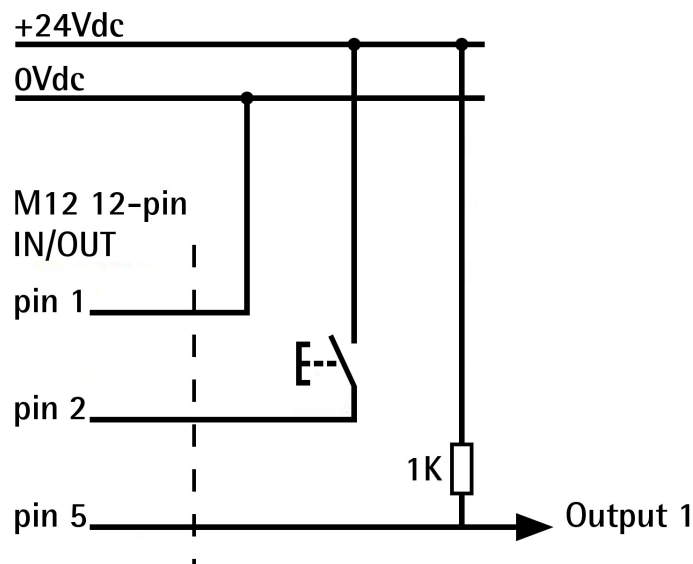
**Inputs** are read by the Device/Slave and transmitted to the Controller/Master through the **Status word** (bits 13 ... 15; see on page 86 ff -Profinet interface- or page 154 ff -Modbus interface) when the device is running in **Operation** state -Profinet interface- / **Idle** state -Modbus interface.

"High" logic value is read when the voltage is equal to +24Vdc  $\pm 10\%$ .

The Device/Slave **output** 1 is operated by the Controller/Master through the **Control word** (bit 13; see on page 77 -Profinet interface- or page 143 -Modbus interface) when the device is running in **Operation** state -Profinet interface- / **Idle** state -Modbus interface.

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

Example of connection scheme:



#### 6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta

The parameters/variables **Distance per revolution**, **Preset**, **Max delta pos / Positive delta** and **Max delta neg / Negative delta** are closely related, hence you have to be very attentive every time you need to change the value in any of them.

Should a new setting be necessary, please comply with the following procedure:

- set a proper value next to the **Distance per revolution** item (see on page 92 -Profinet interface; see on page 136 -MODBUS interface);
- set a proper value next to the **Jog speed** item (see on page 96 -Profinet interface; see on page 140 -MODBUS interface);
- set a proper value next to the **Work speed** item (see on page 97 -Profinet interface; see on page 140 -MODBUS interface);
- set a proper value next to the **Preset** item (see on page 98 -Profinet interface; see on page 142 -MODBUS interface);
- check the value next to the **Max delta pos / Positive delta** item (see on page 94 -Profinet interface; see on page 138 -MODBUS interface);
- check the value next to the **Max delta neg / Negative delta** item (see on page 95 -Profinet interface; see on page 139 -MODBUS interface);

- save the new values (**Save parameters** command, bit 9 in the **Control Word (Bytes 0 and 1)** / **Control Word [0x2A]** item, see on page 77 -Profinet interface; see on page 146 -MODBUS interface).

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

After having changed the parameter in the **Preset** item it is not necessary to set new values for the travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Max delta pos / Positive delta** and **Max delta neg / Negative delta**.

The number of revolutions managed by the system is 512 in negative direction and 512 in positive direction assuming the **Preset** value as a reference (the max. number of revolutions is 1,024).

The value set next to the **Max delta pos / Positive delta** item plus the value set in the **Preset** parameter is the maximum forward travel (positive travel) starting from the preset (the value is expressed in pulses).

The value set next to the **Max delta neg / Negative delta** item subtracted from the value set in the **Preset** parameter is the maximum backward travel (negative travel) starting from the preset (the value is expressed in pulses).



#### WARNING

Please note that the parameters listed hereafter are closely related to the **Distance per revolution** parameter; hence when you change the value in **Distance per revolution** also the value in each of them necessarily changes. They are: **Position tolerance / Position window**, **Max following error**, **Max delta pos / Positive delta**, **Max delta neg / Negative delta**, **Target position**, **Real position / Current position** and **Following error [pulse] / Position following error**.



#### EXAMPLE 1

Default values:

**Distance per revolution** = 1024 steps per revolution

**Work speed**: 2000 rpm

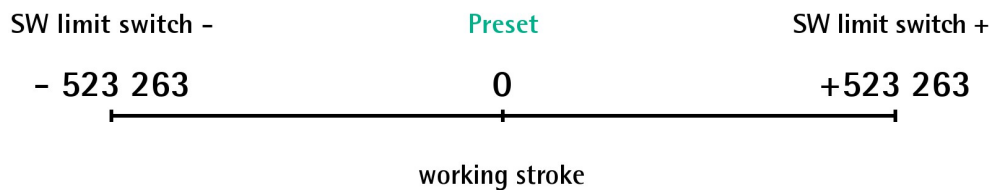
**Preset** = 0

**Max delta pos / Positive delta** and **Max delta neg / Negative delta** max. values =  $523\ 263 = (1,024 \text{ steps per revolution} \times 512 \text{ revolutions}) - 1 - 1,024 \text{ steps}$  (i.e. 1 revolution for safety reasons) when **Preset** = 0

Max. **SW limit switch +** =  $0 + 523\,263 = +523\,263$  pulses (forward travel)

Max. **SW limit switch -** =  $0 - 523\,263 = -523\,263$  pulses (backward travel)

Therefore, when **Preset** = 0, the working stroke of the axis will span the overall positive and negative limits range, that is max. **SW limit switch +** = + 523 263 and max. **SW limit switch -** = - 523 263.



#### EXAMPLE 2

DRIVECOD RD1A positioning unit is joined to a worm screw having 1 mm (0.039") pitch and you need to have a hundredth of a millimetre resolution.

**Distance per revolution** = 100 steps per revolution

Max. **Work speed** = 293 rpm ( $100 * 3000 / 1024 = 293$ )

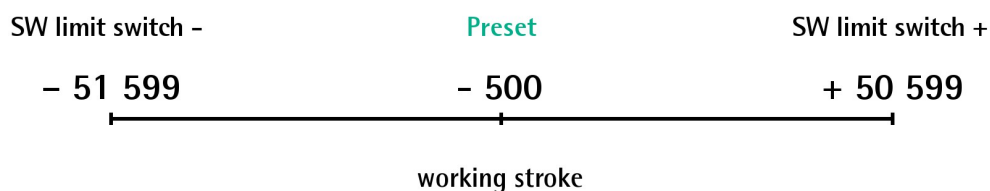
**Preset** = -500 (ex. thickness of the tool)

**Max. delta pos / Positive delta** and **Max delta neg / Negative delta** = (100 steps per revolution \* 512 revolutions) - 1 - 100 steps (i.e. 1 revolution for safety reasons) = 51 099 pulses

Max. **SW limit switch +** =  $(-500) + 51\,099 = +50\,599$  pulses (forward travel)

Max. **SW limit switch -** =  $(-500) - 51\,099 = -51\,599$  pulses (backward travel)

Therefore, when **Preset** = - 500, the working stroke of the axis will span the following positive and negative limits range, that is max. **SW limit switch +** = + 50 599 and max. **SW limit switch -** = - 51 599.



## 7 Profinet interface

Lika ROTADrive positioning units with Profinet interface are Profinet devices with Profinet-RT / -IRT connectivity.

For any further information or omitted specifications please refer to documentation issued by Profibus International and available at the address [www.profibus.com](http://www.profibus.com).

### 7.1 Network and communication settings

The **MAC address** of the device is reported in the label applied to the actuator enclosure. See the following section.

The IP address and the subnet mask as well as the Profinet device name must be assigned by the user to each interface of the unit to be connected in the network. By default, before delivery the device name of the actuator is set to a **blank string** and its IP address is set to **0.0.0.0**. See on page 61.

#### 7.1.1 MAC address

The MAC address is an identifier unique worldwide.

The MAC-ID consists of two parts: the first three bytes are the manufacturer ID and are provided by IEE standard authority; the last three bytes represent a consecutive number of the manufacturer.



#### NOTE

The MAC address is always printed on the actuator's label for commissioning purposes.

The MAC address has the following structure:

Bit value 47 ... 24			Bit value 23 ... 0		
10	B9	FE	X	X	X
Company code (OUI)			Consecutive number		

### 7.2 Configuring the device using TIA PORTAL

#### 7.2.1 Preliminary information

In this manual some screenshots are shown to explain how to install and configure the actuator in a supervisor. In the specific example the development

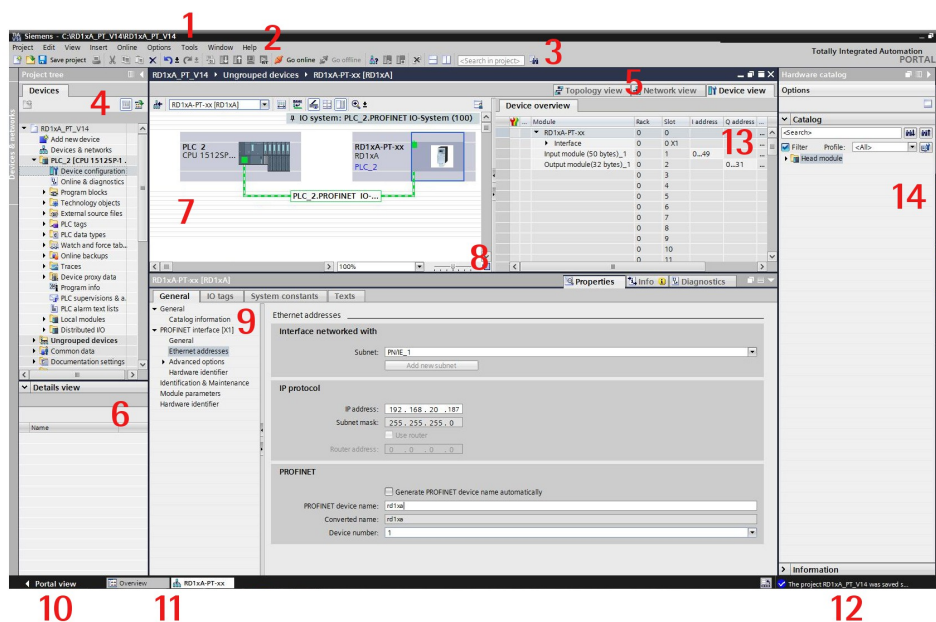
environment is TIA PORTAL V14 with SIEMENS PLC CPU 1512SP-1 PN. If you need to install the actuator using a different configuration tool, please read and follow carefully the instructions given in the documentation provided by the manufacturer.

### 7.2.2 About TIA Portal

TIA Portal stands for Totally Integrated Automation Portal. It is an integrated engineering framework for controllers, HMI and drives. It integrates several SIMATIC products into a single software in order to increase productivity and efficiency.

TIA portal can be used to configure both the PLC and the visualization in an homogeneous system. Data is saved in a single project. Tools for programming (STEP 7) and displaying (WinCC) are not distinct programs, but editors of a system that has access to and uses a common database. One single user interface is used to enter all functions used for displaying and programming.

### 7.2.3 Project overview



1. **Title bar:** the name of the project is displayed in the title bar.
2. **Menu bar:** the menu bar contains all the commands that you require for your work.
3. **Toolbar:** the toolbar provides you with buttons for commands you will use frequently. This gives you faster access to these commands.

4. **Project Tree:** using the Project Tree features gives you access to all components and project data. You can perform the following tasks in the Project Tree:
  - add new components
  - edit existing components
  - scan and modify the properties of existing components
5. **Changeover switches:** they allow to switch among the three working areas of the **Hardware and network editor**: Topology view, Network view and Device view. See point 7 for more information.
6. **Details view:** it shows certain content of the selected object in the **Overview Window** or in the **Project Tree**. This might include text lists or tags. The content of the folders is not shown, however. To display the content of the folders, use the **Project Tree** or the **Inspector Window**.
7. **Graphic Area** of the **Hardware and network editor**. The **Hardware and network editor** opens when you double-click on the **Devices and Networks** entry in the **Project Tree**. The **Hardware and network editor** is the integrated development environment for configuring, networking and assigning parameters to devices and modules. It provides maximum support for the realization of the automation project. This pane is the graphic area where the current configuration of the installed devices with information on the topology and the network can be found. The **Hardware and network editor** provides you with three views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.  
 See the **Changeover switches**, point 5: **Device view** for parametrisation and configuration of the individual devices, it allows to configure and assign both device and module parameters, see on page 54; **Network view** for graphical connections between devices, it allows to configure and assign device parameters and to network the devices with one another, see on page 55; and **Topology view** for current interconnection of Profinet devices, it allows to display and configure the Ethernet topology as well as to identify and minimize differences between the desired and actual topology, see on page 56. In the Figure above the SIEMENS PLC CPU 1512SP-1 PN is the Controller and is connected to the RD1xA-PT rotary actuator, i.e. the Device, through the PLC\_2.PROFINET IO network connection.
8. **Overview Navigation**, it allows to quickly scroll through the objects available in the **Work Area** by pressing the left button of the mouse.
9. **Inspector window:** additional information on an object selected or on actions executed are displayed in the **Inspector window**, the available properties and parameters shown for the object selected can be edited in the Inspector window using the **Properties** tab.

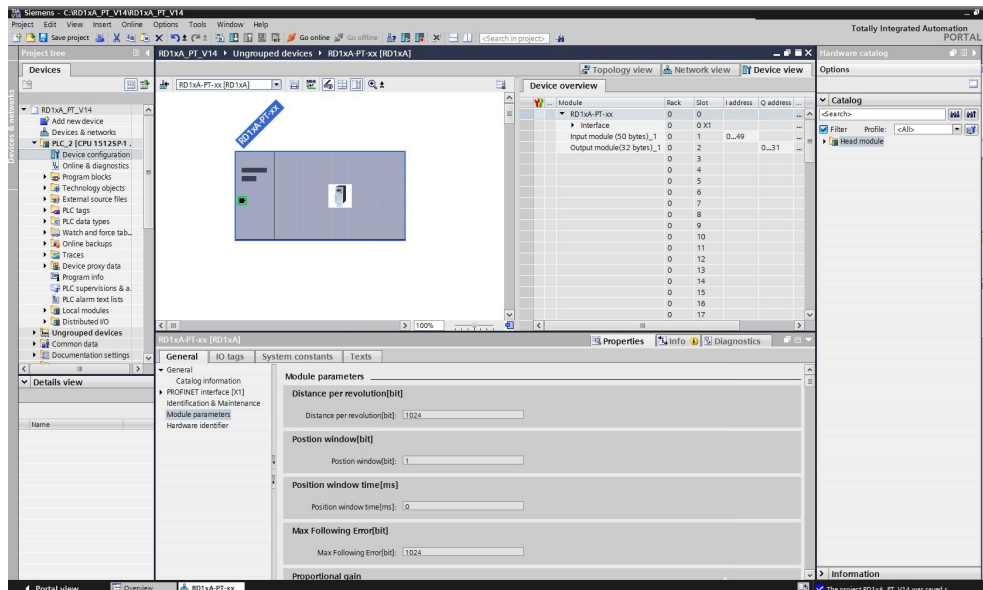


10. It allows to enter the **Portal view**. The Portal view provides you with a task-oriented view of the tools.
11. **Editor bar**: it displays the open editors. If you have opened a lot of editors, they are shown grouped together. You can use the Editor bar to change quickly between the open elements.
12. **Status bar with progress display**. In the status bar, you will find the progress display for processes that are currently running in the background. This also includes a progress bar that shows the progress graphically. Hover the mouse pointer over the progress bar to display a tooltip providing additional information on the active background process. You can cancel the background processes by clicking the button next to the progress bar. If no background processes are currently running, the status bar displays the last generated alarm.
13. **Table Area of the Hardware and network editor**: it offers a general overview of the characteristics of the Device (when **Device view** is selected), of the Network (when **Network view** is selected) and of the Topology (when **Topology view** is selected).
14. **Task Cards**: depending on the edited or selected object, task cards that allow you to perform additional actions are available. These actions include:
  - selecting objects from a library or from the hardware catalog
  - searching for and replacing objects in the project
  - dragging predefined objects to the work area

The task cards available can be found in a bar on the right-hand side of the screen. You can collapse and reopen them at any time. Which task cards are available depends on the products installed. More complex task cards are divided into panes that you can also collapse and reopen.

The **Hardware catalog** can be selected in the **Task Cards**; it allows to install the available components just dragging and dropping them onto the **Work Area**. Customarily the field devices that have been integrated into the TIA Portal via XML files are listed under **Other field devices > Profinet IO**.

## 7.2.4 Device view

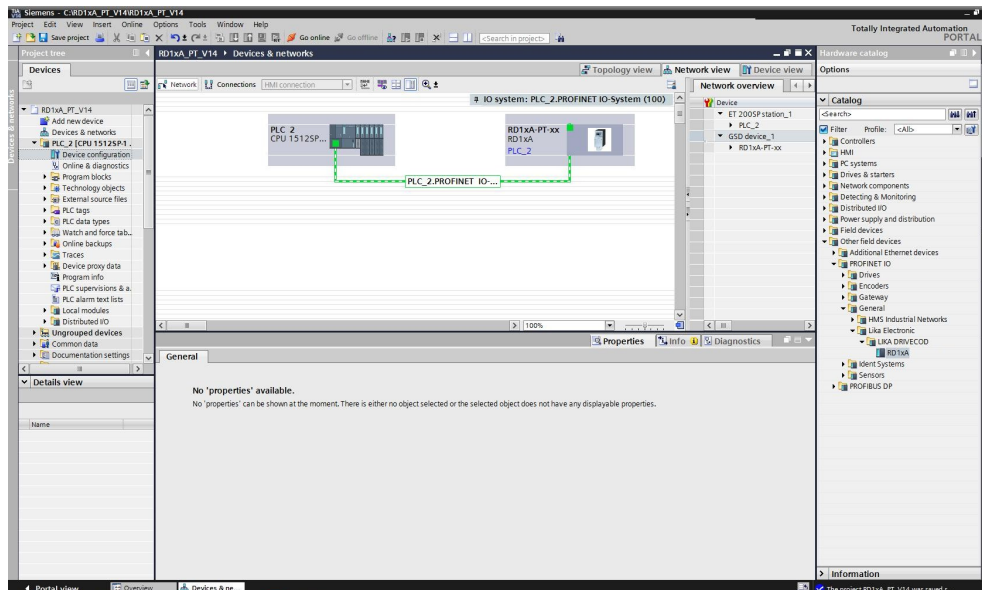


Press the **Device view** changeover switch in the **Hardware and network editor** to enter the **Device view**.

The configuration of devices and assigning of addresses etc. is performed in the **Device view**. All devices are represented in a photo-realistic way.

- Buffering of configured hardware modules and reuse with module clipboard
- When zoomed to at least 200%, I/Os are displayed with the symbolic names / addresses
- Automatic readout of available hardware with hardware detect
- Full text search in the Hardware catalogue
- Option of filtering the Hardware catalogue to show modules that can currently be used
- All parameters and configuration data are displayed on a hierarchical and context-sensitive basis

## 7.2.5 Network view



Press the **Network view** changeover switch in the **Hardware and network editor** to enter the **Network view**.

The **Network view** enables the configuration of plant communication. The communication links between individual stations are displayed here graphically and very clearly.

- Combined view of all network resources and network components
- Fully graphical configuration of the individual stations
- Resources are networked by linking communication interfaces using drag & drop
- Multiple controllers, peripherals, HMI devices, SCADA stations, PC stations and drives possible in a single project
- Procedure for integrating AS-i devices identical to PROFIBUS/PROFINET
- Zoom and page navigation
- Copying/pasting entire stations, incl. configuration, or individual hardware modules

A subnet (PLC\_2.PROFINET IO) is added to the operator panel. Click the subnet (PLC\_2.PROFINET IO) to apply the network settings. Specify the required network settings under **Properties > Network Settings** in the **Properties** area (see point 9 on page 51). Make sure that you use the same settings throughout the entire network.

### 7.2.6 Topology view

Press the **Topology view** changeover switch in the **Hardware and network editor** to enter the **Topology view**.

Decentralised peripherals on Profinet are configured in the Network view. The controllers and the decentralised peripherals assigned to them can be shown graphically. During ongoing operation, however, it is not possible to see which ports are actually connected and communicating with each other.

Yet this is precisely what is often important for diagnostics. For Profinet networks, the **Topology view** enables this information to be displayed quickly and easily. An offline/online comparison identifies the communicating ports. By detecting, presenting and monitoring the physical connections between devices on Profinet, the administrator can easily monitor and maintain even complex networks.

### 7.2.7 Installing the GSDML file

The functionality of a PROFINET IO device is always described in a GSDML file. This file contains all data that are relevant for engineering as well as for data exchange with the IO device.

PROFINET IO devices can be described using XML-based GSD. The description language of the GSD file, i.e. GSDML (General Station Description Markup Language) is based on international standards. As the name suggests, the GSD file is a language-independent XML file (Extensible Markup Language).

RD1xA rotary actuators equipped with Profinet interface are supplied with their own GSDML file **GSDML-V2.32-LIKA-0239-DRIVECOD-RD1xA-XXXXXXXX.xml** where XXXXXXXX is the release date of the file in a 8-digit format encompassing information about year (4 digits), month (2 digits) and day (2 digits): **20170911** is the first GSDML file released by Lika Electronic for Profinet actuators. To download the file please enter **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS / POSITIONING UNITS (DRIVECOD) > RD1xA**).

The XML file has to be installed in the Profinet Controller.

**Version structure of GSDML files**

The GSDML file structure is in compliance with the ISO 15745 "Open Systems Application Integration Framework" and is oriented on the defined profile of a field device via the following model:

<b>GSDML-</b>	<b>V2.32-</b>	<b>LIKA-0239-</b>	<b>DRIVECOD- RD1xA-</b>	<b>20170911</b>	<b>.xml</b>
GSD data identification	Version of GSDML scheme	Manufacturer	Name of device	Version number, format: yyyymmdd	File extension

- The version of the GSDML model used defines which scope of language a GSD file uses.
- The version date is updated, if, for example, an error is cleared or a function extended.

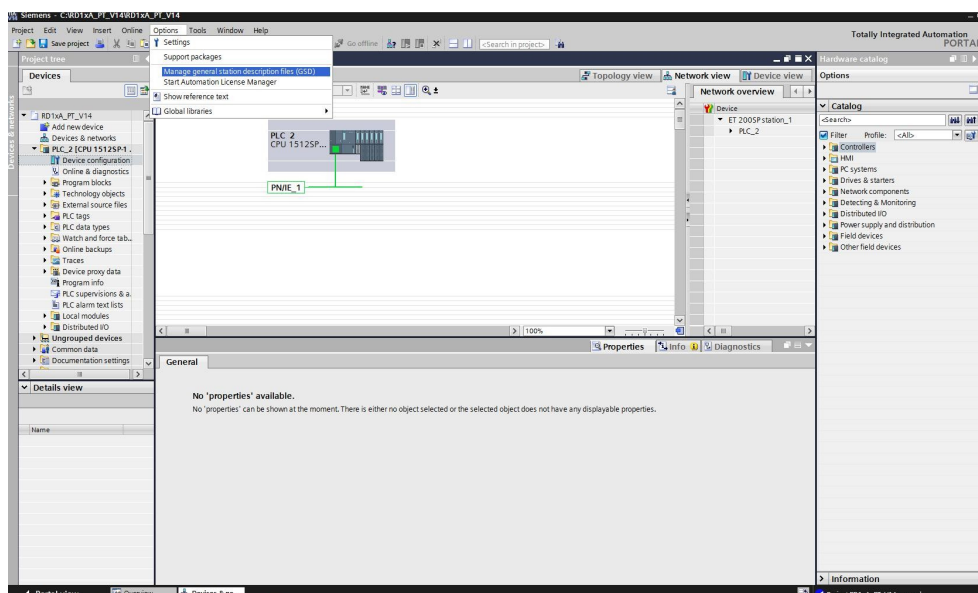

**WARNING**

Please be aware that you compulsorily comply with the following compatibilities:

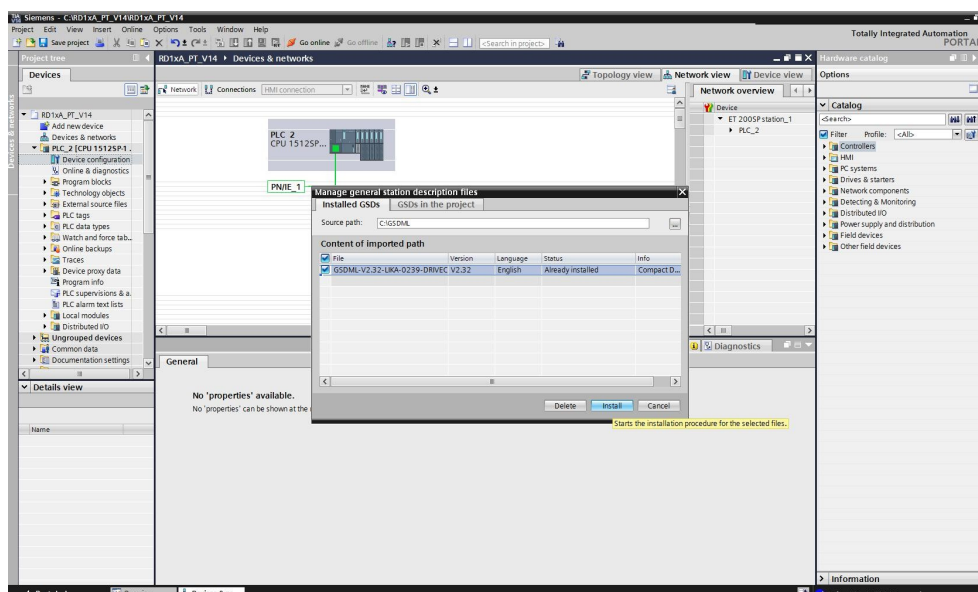
<b>GSDML file version</b>	<b>Actuator HW version</b>	<b>Actuator SW version</b>	<b>User's guide version</b>	<b>Modbus EXE</b>
From release 20170911 to ...	2	1.0	1.0	From 1.2 up to ...

To install the RD1xA unit on TIA Portal proceed as follows.

1. In the TIA Portal, select the **Options > Manage general station description files (GSD)** menu.



2. In the **Manage general station description files** dialog box, select the directory containing the GSDML files.
3. Select the GSDML file specific to the unit you need to install.
4. Click the **INSTALL** button.

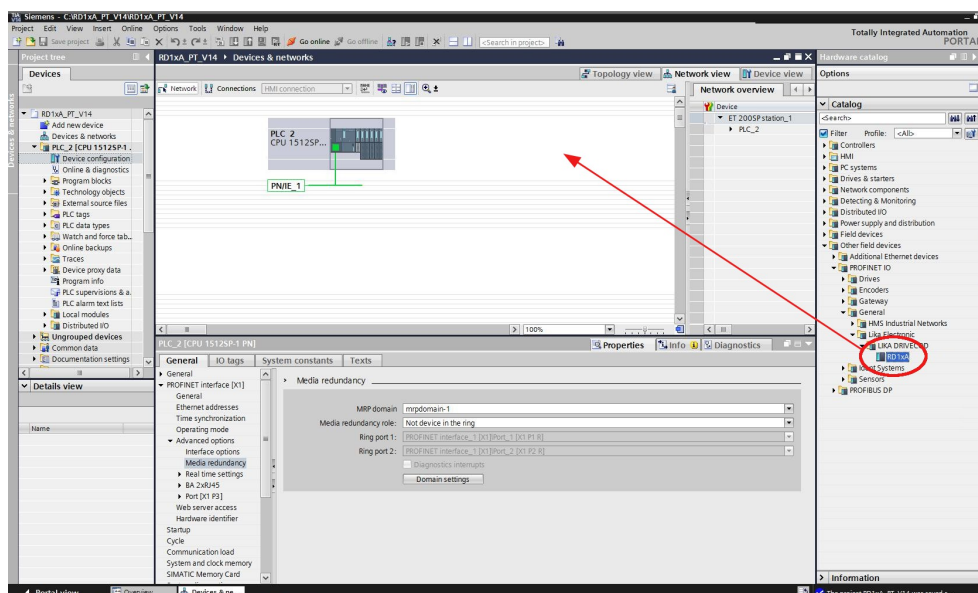


Customarily the field devices that have been integrated into the TIA Portal via GSDML files are listed under **Other field devices > Profinet IO**, see the "7.2.3 Project overview" section on page 51.

### 7.2.8 Adding a node to the project

On the right side of the TIA Portal, open the **Hardware catalog** task card. The field devices integrated into the TIA Portal via the Profinet file (GSDML file) can then be found under **Other field devices\PROFINET IO\General\Lika Electronic\Lika DRIVECOD**.

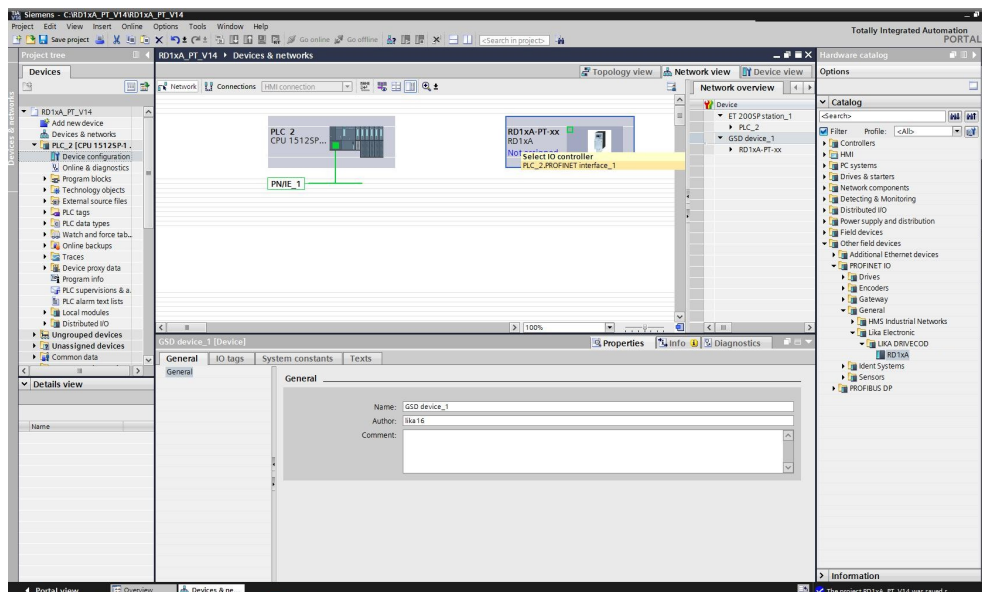
Select the desired device from the **Lika DRIVECOD** directory (for instance, **RD1xA**) and use drag-and-drop to move the item from the editing window to the **Network view**. This creates the device in the project. Detailed information on the selected device is available at the bottom of the Hardware catalog in **Information**.



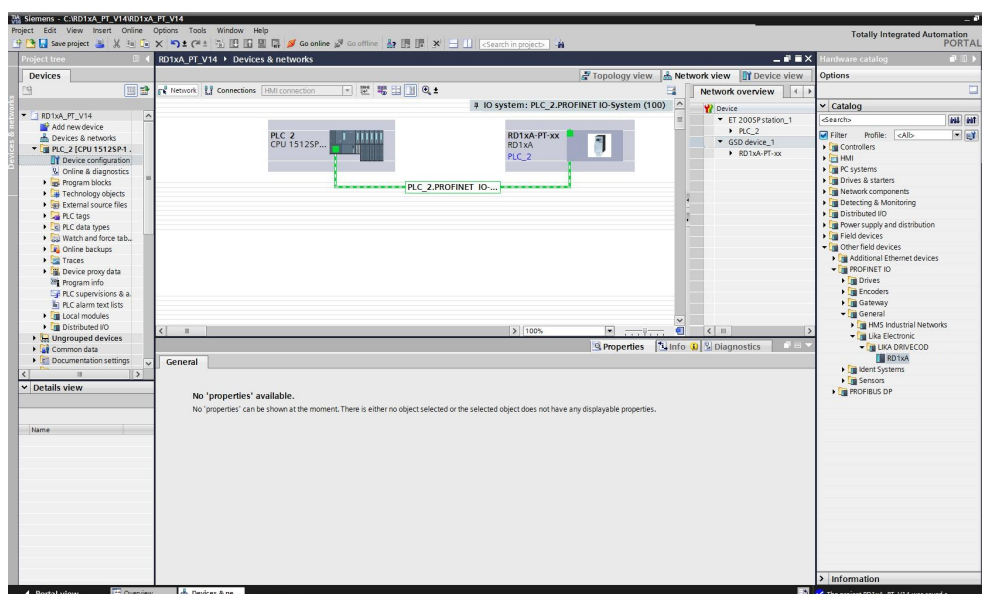
### 7.2.9 Establishing the bus connection

As soon as the device has been inserted into the project, the bus connection with the PLC can be established in the **Network view**.

The "**Not assigned**" information message appears in the node picture: it warns that the connection between the PLC and the Slave device is not established yet. Right-click on the message and select, through the **Select IO controller** drop-down box, the PLC the node has to be connected to. When doing so, make sure that you are in the **Network** function mode in the **Network view**.



After configuring the networking, the device is connected to the PLC via the PROFINET network.





### 7.2.10 Device name and IP address at delivery

In a Profinet network it is mandatory that each IO device is provided with its own Device name and IP address. By default, before delivery the device name of the actuator is set to a **blank string** and its IP address is set to **0.0.0.0**.

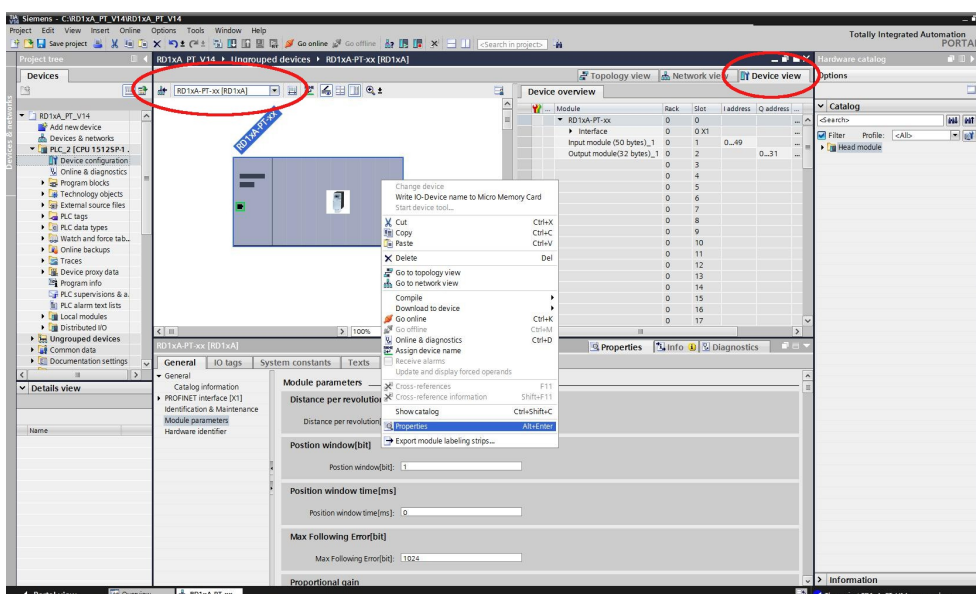
Before the PROFINET IO controller can address a PROFINET IO device, a name has to be assigned to the PROFINET IO device. PROFINET uses this method because names are easier to use and recall than complex IP addresses. Devices on an Ethernet subnet must have unique names.

The device names must satisfy DNS (Domain Name System) conventions:

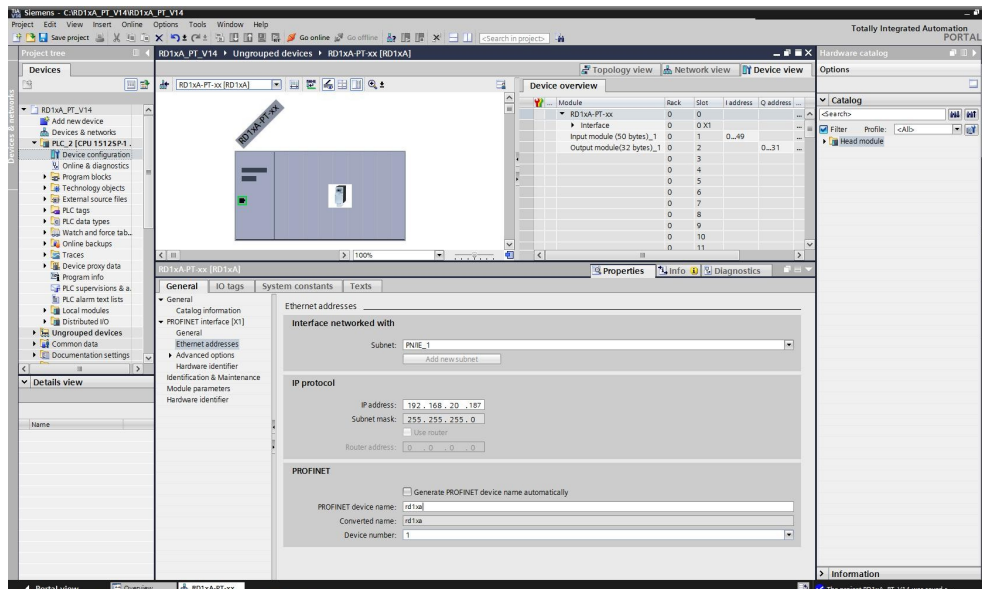
- Names are limited to a total of 127 characters (letters, numbers, dashes or dots).
- Any component part (that is, a character string between two dots) of the device name may only be up to 63 characters long.
- Names cannot contain any special character such as umlauts, parentheses, underscores, forward or backward slashes, empty spaces, etc. The dash is the only special character allowed.
- Names must neither start nor end with the minus "-" sign.

### 7.2.11 Setting the device name and the IP address

As stated, to completely establish the connection, you have to assign the IP address and the Profinet device name to the Slave device. To do so, enter the **Device view** working area, select the device you need to configure in the drop-down box on the top left of the graphic area, right-click on the image of the module and select the **Properties** command from the shortcut menu.



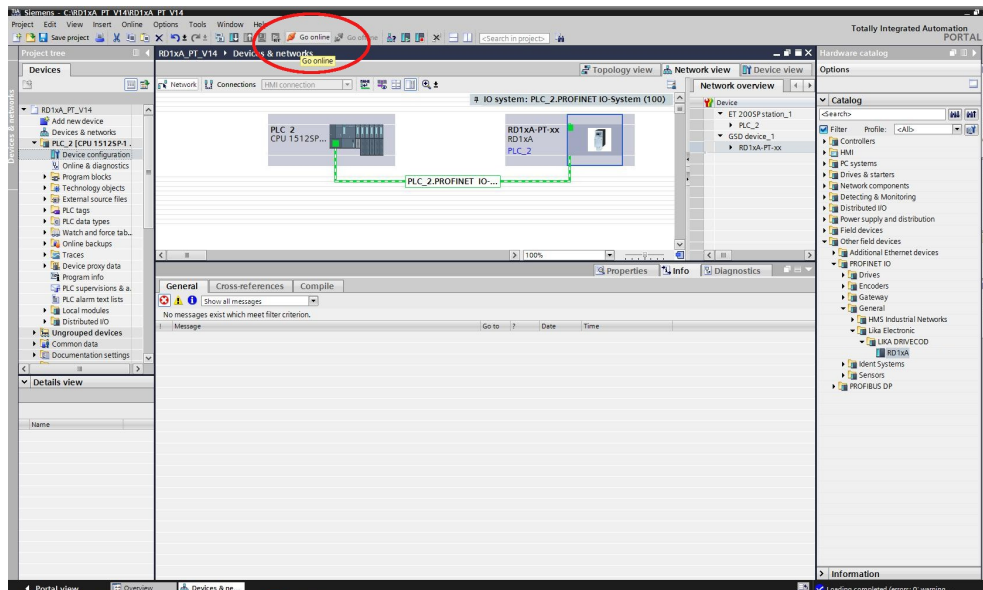
In the **Properties** inspector window, **General** tab, you can now use the **Ethernet addresses** menu option to set the Ethernet address (IP address, subnet mask, ...) and assign the Profinet name of the Device.



## 7.2.12 Compiling and transferring the project

After setting you must compile and then transfer the project to the device.

### 7.2.13 Establishing an online connection (Online mode)



In online mode, there is an online connection between the PLC and one or more devices. An online connection between the PLC and the device is required, for example, for the following tasks:

- Using the Control Table
- Testing user programs
- Displaying and changing the operating mode of the device
- Displaying module information
- Comparing blocks
- Hardware diagnostics

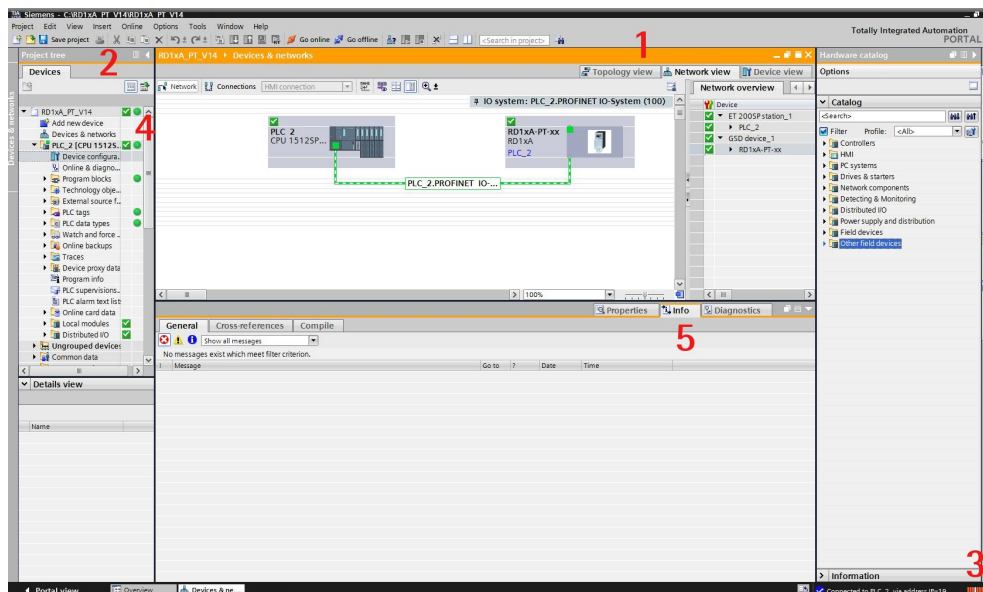
Before you can establish an online connection, the PLC and the device must be physically or remotely connected.

After establishing a connection, you can use the **Online and Diagnostics view** or the **Online tools** task card to access the data on the device. The current online status of a device is indicated by an icon to the right of the device in the **Project Tree**.

To establish an online connection between the PLC (Profinet Controller) and the device (Profinet Device) proceed as follows.

- In the **Project Tree** (see point 4 in the "7.2.3 Project overview" section on page 51) mark the folder of the PLC that is configured as the Controller.
- Select the **Go online** command in the **Online** menu bar to establish an online connection to the PLC (Controller) and to the device (Device).

- If the device has already been connected online, the online connection is automatically established using the previously specified connection path.
- If there was no previous connection, the **Go online** dialog opens.
- Select the connection path:
  - select the type of interface;
  - select the interface of the PLC;
  - select the interface or the subnet for the connection.
- Click the **START SEARCH** button. Devices which can be reached by the set connection path are displayed in the **Compatible devices in target subnet**. The connection line in the graphic is displayed as solid.
- Select the device in the **Compatible devices in target subnet table** and confirm the selection with **Go online**. The online connection to the selected target device is established.



After the online connection has been established successfully, the user interface changes (see the Figure above).

1. The title bar of the active window gets an orange background as soon as at least one of the devices currently displayed in the editor has been successfully connected online. If one or more devices are unavailable, a symbol for a broken connection appears in the title bar of the editor.
2. The title bars of inactive windows for the relevant station now have an orange line below them.
3. An orange, pulsing bar appears at the right-hand edge of the status bar. If the connection has been established but it is not working properly, an icon for an interrupted connection is displayed instead of the bar. You

will find more information on the error in **Diagnostics** in the **Inspector window**.

4. Operating mode symbols or diagnostics symbols for the stations connected online and their underlying objects are shown in the **Project Tree**. A comparison of the online and offline status is also made automatically. Differences between online and offline objects are also displayed in the form of symbols.
5. The **Diagnostics > Device information** area is brought to the foreground in the **Inspector window**.

#### 7.2.14 Closing an online connection

To close the existing online connection, follow these steps.

1. Select the device for which you want to disconnect the online connection in the **Project Tree**.
2. Select the **Go offline** command in the **Online** menu bar. The online connection is disconnected.

#### 7.2.15 Diagnostics

Configuration of the diagnostics is integrated in the system in a user-friendly way and activated with just one click. When new hardware components are introduced, the diagnostic information is updated automatically via the engineering system (HWCN). System diagnostics outputs all relevant information on existing errors in the system. This information is packaged automatically in messages containing the following elements:

- Module
- Message text
- Message status

To access the diagnostics function please proceed as follows.

1. Right-click on the module to process.
2. Select the **Online & diagnostics** command from the shortcut menu.
3. If there is no online connection established, click the **Connect online** button in the **Diagnostics** entry.
4. The diagnostic status of the module will be displayed in the **Diagnostic status** group in the **Diagnostics** folder in the **Online and diagnostics view** of the module to be diagnosed.

The following status information is displayed in the **Diagnostic status** area:

- Status of the module as viewed by the CPU, for example:
  - Module available and OK.
  - Module defective.  
If the module experiences a fault and you have enabled the diagnostic error interrupt during configuration, the "Module defective" status is displayed.
  - Module configured, but not available.  
Example: Diagnostics data is not available because the current online configuration differs from the offline configuration.
- Detected differences between the configured and the inserted module.  
Provided it can be ascertained, the article number will be displayed for the set and actual type.

The scope of the displayed information depends on the selected module.

## 7.2.16 Watch Table

The "watch tables" are used to monitor and modify the values of a user program being executed by the online CPU. You can create and save different watch tables in your project to support a variety of test environments. This allows you to reproduce tests during commissioning or for service and maintenance purposes.

With a watch table, you can monitor and interact with the CPU as it executes the user program. You can display or change values not only for the tags of the code blocks and data blocks, but also for the memory areas of the CPU, including the inputs and outputs (I and Q), peripheral inputs (I:P), bit memory (M), and data blocks (DB).

With the watch table, you can enable the physical outputs (Q:P) of a CPU in STOP mode. For example, you can assign specific values to the outputs when testing the wiring for the CPU.

To create a watch table:

- Double-click **Add new watch table** in the **Project Tree** to open a new watch table.
- Enter the tag name to add a tag to the watch table.

To monitor the tags, you must have an online connection to the CPU.

Siemens - C:\RD1A_P1_V14\RD1A_P1_V14							
RD1A_P1_V14 - PLC_2 (PLC 1512P-1 PN) - Watch and force tables - Watch table							
#	Name	Address	Display format	Monitor value	Modify value	Comment	Tag comment
IF CYCLIC CHANNEL - DEVICE => CONTROLLER							
1	"Actual Position [mm]"	%D0	DEC+/-	204651			
2	"Actual Speed [mm/s]"	%D4	DEC+/-	0			
3	"Status word"	%W8	Hex	16#0020			
4	"Alarms"	%W10	Hex	16#0040			
5	"Following error[bit]"	%D12	Hex	16#0000_000A			
6	"Current[mA]"	%D16	DEC+/-	111			
7	"Electronic temperature [°C]"	%B20	DEC	25			
8	"Motor temperature [°C]"	%B21	DEC	230			
9	"Parameter error list"	%D22	Hex	16#0000_0000			
10	"Parameter number [read]"	%B26	DEC	0			
11	"Parameter value [write]"	%D27	DEC+/-	0			
IF CYCLIC CHANNEL - CONTROLLER => DEVICE							
12	"Control word"	%QW0	Hex	16#0084			
13	"TargetPosition[mm]"	%QD2	DEC+/-	3000	3000		
14	"Parameter number [write]"	%QD6	Hex	16#000	16#000		
15	"Parameter value [write]"	%QD7	DEC+/-	0			
----- CTRL WORD -----							
16	"CTRL word bit 00: jog"	%Q1.0	Bool	<input type="checkbox"/>	FALSE		
17	"CTRL word bit 01: jog +"	%Q1.1	Bool	<input type="checkbox"/>	FALSE		
18	"CTRL word bit 02: stop"	%Q1.2	Bool	<input checked="" type="checkbox"/>	TRUE		
19	"CTRL word bit 03: reset alarms"	%Q1.3	Bool	<input type="checkbox"/>	FALSE		
20	"CTRL word bit 04: incremental jog (jog step)"	%Q1.4	Bool	<input type="checkbox"/>	FALSE		
21	"CTRL word bit 05: not used"	%Q1.5	Bool	<input type="checkbox"/>	FALSE		
22	"CTRL word bit 06: start positioning"	%Q1.6	Bool	<input type="checkbox"/>	FALSE		
23	"CTRL word bit 07: Emergency"	%Q1.7	Bool	<input checked="" type="checkbox"/>	TRUE		
24	"CTRL word bit 08: not used"	%Q1.8	Bool	<input type="checkbox"/>	FALSE		
25	"CTRL word bit 09: save parameter"	%Q1.9	Bool	<input type="checkbox"/>	FALSE		
26	"CTRL word bit 10: load default"	%Q2.0	Bool	<input type="checkbox"/>	FALSE		
27	"CTRL word bit 11: preset cont"	%Q2.3	Bool	<input type="checkbox"/>	FALSE		
28	"CTRL word bit 12: Release torque"	%Q2.4	Bool	<input type="checkbox"/>	FALSE		
29	"CTRL word bit 13: not used"	%Q2.5	Bool	<input type="checkbox"/>	FALSE		
30	"CTRL word bit 14: not used"	%Q2.6	Bool	<input type="checkbox"/>	FALSE		
31	"CTRL word bit 15: not used"	%Q2.7	Bool	<input type="checkbox"/>	FALSE		
----- TARGET POSITION -----							
32	"TargetPosition[mm]"	%QD2	Hex	16#0000_0088			
----- STATUS WORD -----							
33	"Status Reg bit 00: RD in position"	%I0.0	Bool	<input type="checkbox"/>	FALSE		
34	"Status Reg bit 01: not used"	%I0.1	Bool	<input type="checkbox"/>	FALSE		
35	"Status Reg bit 02: driver enable"	%I0.2	Bool	<input type="checkbox"/>	FALSE		
36	"Status Reg bit 03: f.c. pos"	%I0.3	Bool	<input type="checkbox"/>	FALSE		
37	"Status Reg bit 04: f.c. neg"	%I0.4	Bool	<input type="checkbox"/>	FALSE		
38	"Status Reg bit 05: Alarm"	%I0.5	Bool	<input checked="" type="checkbox"/>	TRUE		
39	"Status Reg bit 06: Moving"	%I0.6	Bool	<input type="checkbox"/>	FALSE		
40	"Status Reg bit 07: Command active"	%I0.7	Bool	<input type="checkbox"/>	FALSE		
41	"Status Reg bit 08: target reached"	%I0.8	Bool	<input type="checkbox"/>	FALSE		
42	"Status Reg bit 09: KEY 1"	%I0.9	Bool	<input type="checkbox"/>	FALSE		
43	"Status Reg bit 10: KEY 2"	%I1.0	Bool	<input type="checkbox"/>	FALSE		
44	"Status Reg bit 11: KEY 3"	%I1.1	Bool	<input type="checkbox"/>	FALSE		
45	"Status Reg bit 12: dac saturation"	%I1.2	Bool	<input type="checkbox"/>	FALSE		
46	"Status Reg bit 13: IN1"	%I1.3	Bool	<input type="checkbox"/>	FALSE		
47	"Status Reg bit 14: IN2"	%I1.4	Bool	<input type="checkbox"/>	FALSE		
48	"Status Reg bit 15: IN3"	%I1.5	Bool	<input type="checkbox"/>	FALSE		
----- ALARMS -----							
49	"Alarm word bit 00: Parameter fail"	%I1.0	Bool	<input type="checkbox"/>	FALSE		
50	"Alarm word bit 01: flash error"	%I1.1	Bool	<input type="checkbox"/>	FALSE		
51	"Alarm word bit 02: Count error"	%I1.2	Bool	<input type="checkbox"/>	FALSE		
52	"Alarm word bit 03: Following error"	%I1.3	Bool	<input type="checkbox"/>	FALSE		
53	"Alarm word bit 04: Encoder not synchronized"	%I1.4	Bool	<input type="checkbox"/>	FALSE		
54	"Alarm word bit 05: Target out of range"	%I1.5	Bool	<input type="checkbox"/>	FALSE		
55	"Alarm word bit 06: Emergency"	%I1.6	Bool	<input checked="" type="checkbox"/>	TRUE		
56	"Alarm word bit 07: Overcurrent"	%I1.7	Bool	<input type="checkbox"/>	FALSE		
57	"Alarm word bit 08: Electronic overtemperature"	%I1.8	Bool	<input type="checkbox"/>	FALSE		
58	"Alarm word bit 09: Motor overtemperature"	%I1.9	Bool	<input type="checkbox"/>	FALSE		
59	"Alarm word bit 10: Undervoltage"	%I2.0	Bool	<input type="checkbox"/>	FALSE		
60	"Alarm word bit 11: invalid parameter number"	%I2.3	Bool	<input type="checkbox"/>	FALSE		
61	"Alarm word bit 12: write a read-only parameter"	%I2.4	Bool	<input type="checkbox"/>	FALSE		
62	"Alarm word bit 13: not used"	%I2.5	Bool	<input type="checkbox"/>	FALSE		
63	"Alarm word bit 14: Hall error"	%I2.6	Bool	<input type="checkbox"/>	FALSE		
64	"Alarm word bit 15: overvoltage"	%I2.7	Bool	<input type="checkbox"/>	FALSE		
----- I_Q15 -----							
65	"Digitalinput_Q15"	%I80	Hex	16#84			
66	"Digitalinput_Q15"	%I81	Hex	16#00			
----- PARAMETER ERROR LIST -----							
67	"Parameter error (bit 00)"	%I25.0	Bool	<input type="checkbox"/>	FALSE		
68	"Parameter error (bit 01: Wrong dist. rev)"	%I25.1	Bool	<input type="checkbox"/>	FALSE		
69	"Parameter error (bit 02: Wrong acceleration)"	%I25.2	Bool	<input type="checkbox"/>	FALSE		
70	"Parameter error (bit 03: Wrong Deceleration)"	%I25.3	Bool	<input type="checkbox"/>	FALSE		
71	"Parameter error (bit 04: Wrong Delta pos)"	%I25.4	Bool	<input type="checkbox"/>	FALSE		
72	"Parameter error (bit 05: Wrong Delta neg)"	%I25.5	Bool	<input type="checkbox"/>	FALSE		
73	"Parameter error (bit 06: Wrong jog speed)"	%I25.6	Bool	<input type="checkbox"/>	FALSE		
74	"Parameter error (bit 07: Wrong work speed)"	%I25.7	Bool	<input type="checkbox"/>	FALSE		
75	"Parameter error (bit 08: Wrong count direction)"	%I25.8	Bool	<input type="checkbox"/>	FALSE		
76	"Parameter error (bit 09: Wrong Preset)"	%I25.9	Bool	<input type="checkbox"/>	FALSE		
77	"Parameter error (bit 10: Wrong jog step)"	%I26.0	Bool	<input type="checkbox"/>	FALSE		
78	"Parameter error (bit 11: Wrong Kp)"	%I26.1	Bool	<input type="checkbox"/>	FALSE		
79	"Parameter error (bit 12: Wrong Ki)"	%I26.2	Bool	<input type="checkbox"/>	FALSE		
80	"Parameter error (bit 13: Wrong Time pas)"	%I26.3	Bool	<input type="checkbox"/>	FALSE		
81	"Parameter error (bit 14: Wrong max follow err)"	%I26.4	Bool	<input type="checkbox"/>	FALSE		
82	"Parameter error (bit 15)"	%I26.5	Bool	<input type="checkbox"/>	FALSE		
83	"Parameter error (bit 16)"	%I26.6	Bool	<input type="checkbox"/>	FALSE		
84	"Parameter error (bit 17)"	%I26.7	Bool	<input type="checkbox"/>	FALSE		
85	"Parameter error (bit 18)"	%I26.8	Bool	<input type="checkbox"/>	FALSE		
86	"Parameter error (bit 19)"	%I26.9	Bool	<input type="checkbox"/>	FALSE		
87	"Parameter error (bit 20)"	%I27.0	Bool	<input type="checkbox"/>	FALSE		
88	"Parameter error (bit 21)"	%I27.1	Bool	<input type="checkbox"/>	FALSE		
89	"Parameter error (bit 22)"	%I27.2	Bool	<input type="checkbox"/>	FALSE		
90	"Parameter error (bit 23)"	%I27.3	Bool	<input type="checkbox"/>	FALSE		
91	"Parameter error (bit 24)"	%I27.4	Bool	<input type="checkbox"/>	FALSE		
92	"Parameter error (bit 25)"	%I27.5	Bool	<input type="checkbox"/>	FALSE		
93	"Parameter error (bit 26)"	%I27.6	Bool	<input type="checkbox"/>	FALSE		
94	"Parameter error (bit 27)"	%I27.7	Bool	<input type="checkbox"/>	FALSE		
95	"Parameter error (bit 28)"	%I27.8	Bool	<input type="checkbox"/>	FALSE		
96	"Parameter error (bit 29)"	%I27.9	Bool	<input type="checkbox"/>	FALSE		
97	"Parameter error (bit 30)"	%I28.0	Bool	<input type="checkbox"/>	FALSE		
98	"Parameter error (bit 31)"	%I28.1	Bool	<input type="checkbox"/>	FALSE		
<Add new>							

The Watch Table shown above is only for sample purpose. You can create your own Watch table as the sample one above to monitor and modify the current values of individual parameters. You can assign values to individual items for



testing and run the program in a variety of different situations. The values are not saved permanently.

The example of the Watch table is divided into eight sections:

**1. CYCLIC CHANNEL: DEVICE => CONTROLLER**

In this section data sent by the Device to the Controller via the Cyclic Data Exchange channel is listed. In particular, for instance, the **Current Position (Bytes 0 to 3)** -ACTUAL POSITION [BIT]-, the **Status Word (Bytes 8 and 9)** -STATUS WORD-, the **Alarms List (Bytes 10 and 11)** -ALARMS-, the requested **Parameter number (Byte 26)** -PARAMETER NUMBER [READ]-, and the **Parameter value (Bytes 27 to 30)** -PARAMETER VALUE [READ]- are found. The interval between transmissions is set in the PLC.

**2. CYCLIC CHANNEL: CONTROLLER => DEVICE**

In this section data sent by the Controller to the Device via the Cyclic Data Exchange channel is listed. In particular the **Control Word (Bytes 0 and 1)** -CONTROL\_WORD-, the **Target Position (Bytes 2 to 5)** -TARGET POSITION [BIT]-, the requested **Parameter number (Byte 6)** -PARAMETER NUMBER [WRITE]-, and the **Parameter value (Bytes 7 ... 10)** -PARAMETER VALUE [WRITE]- are found. You can set and transmit new values to the device. The interval between transmissions is set in the PLC.

**3. CONTROL WORD**

In this section the meaning and value of the 16 bits of the **Control Word (Bytes 0 and 1)** item are detailed. You can set and transmit new values to the device. See on page 77.

**4. TARGET POSITION**

In this section the target position value can be set and transmitted to the device. See the **Target Position (Bytes 2 to 5)** item on page 81.

**5. STATUS WORD**

In this section the meaning and value of the 16 bits of the **Status Word (Bytes 8 and 9)** item are detailed. See on page 84.

**6. ALARMS**

In this section the value of the 16 bits of the **Alarms List (Bytes 10 and 11)** item is shown. See on page 87.

**7. IO...I15**

In this section the value of the digital inputs can be forced. See on page 87.

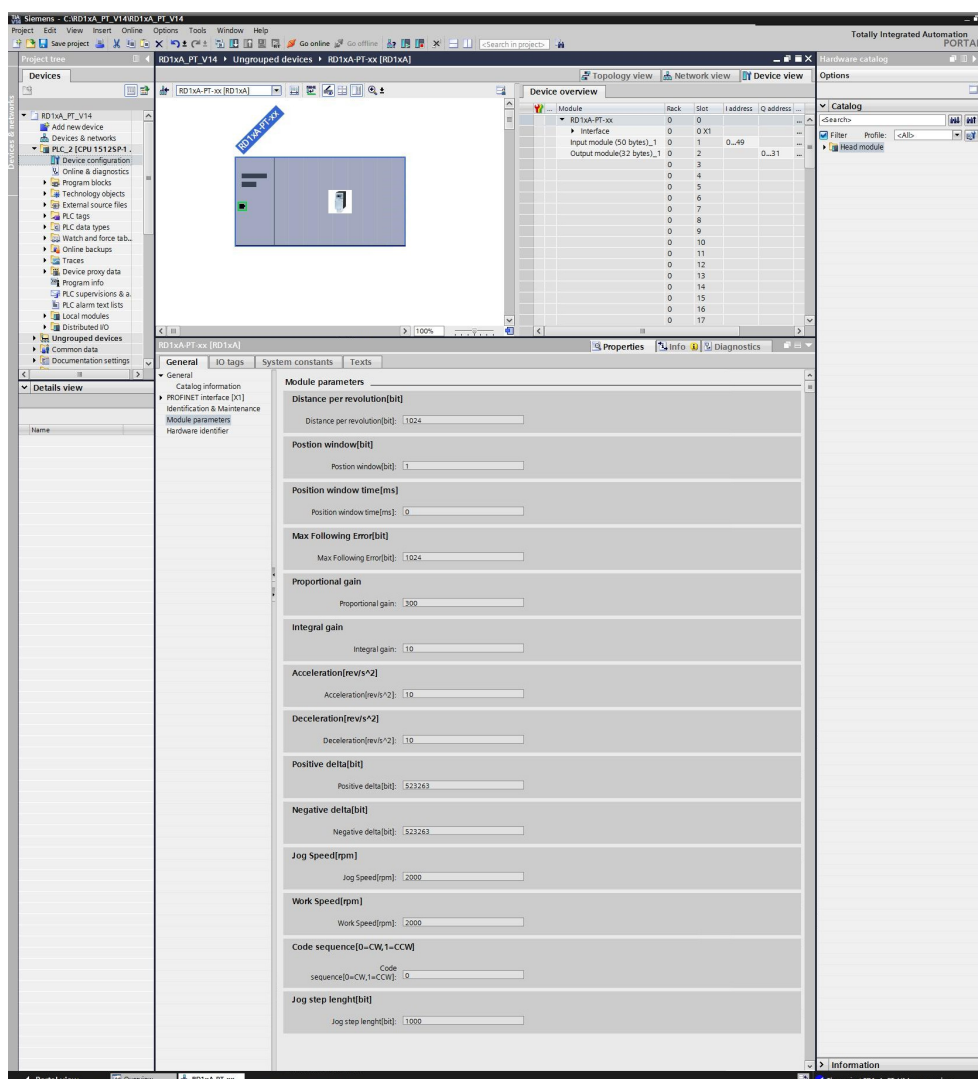


## 8. PARAMETER ERROR LIST

In this section the value of the 32 bits of the [Parameter Error List \(Bytes 22 to 25\)](#) item is shown. See on page 89.

It is possible to choose the display format of each item.

### 7.2.17 Module parameters



Press the **Device view** changeover switch in the **Hardware and network editor** to enter the **Device view** working area, then select the device you need to configure in the drop-down box on the top left of the graphic area. In the **Properties** inspector window, **General** tab, press the **Module parameters** menu option to see and set if required the actuator's parameters.

The parameters listed in this page are sent in the Acyclic Data Exchange mode at switching on.

You can change the value of each parameter in the edit field. The new value will be transmitted to the Device in the Acyclic Data Exchange mode at switching on.

You can change the value of the module parameters also while the device is operational in the Cyclic Data Exchange mode via the Watch table. Please note that the value however will be overwritten at switching on by the value set in the **Module parameters** tabbed page.

For a comprehensive description of the parameters and how to set them properly refer to the specific explanation in the "7.4 Profinet module parameters" section on page 92.

## 7.2.18 Setting and reading parameter values

Here following are some examples of how you can read and set the parameters.



### 7.2.18.1 Reading and setting the 20 Preset parameter



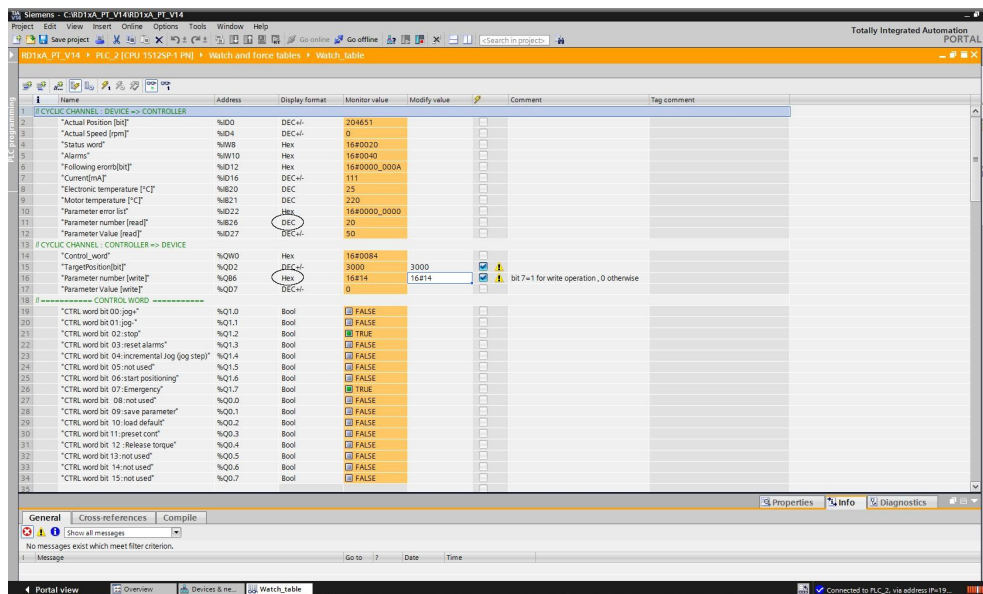
#### NOTE

Please note that the **20 Preset** parameter is the only R/W parameter that is not available in the **Module parameters** tabbed page otherwise it would be sent at each switching on.

In the example, first we want to read the preset value that is actually set in the device and then enter the value 100 dec.

To read the parameter value we must enter the number of the parameter we want to read in hexadecimal format next to the "PARAMETER NUMBER [WRITE]" in the CYCLIC CHANNEL: CONTROLLER => DEVICE section. The number of the **20 Preset** parameter is 14h, 20 dec. Find more information next to the **Parameter number (Byte 6)** item on page 82.

The value that is actually set in the **20 Preset** parameter will be shown next to the "PARAMETER VALUE [READ]" in the CYCLIC CHANNEL: DEVICE => CONTROLLER section: 50 dec. Next to the "PARAMETER NUMBER [READ]" in the CYCLIC CHANNEL: DEVICE => CONTROLLER section the number of the parameter will be displayed: 20 dec, 14h. Find more information next to the **Parameter number (Byte 26)** and **Parameter value (Bytes 27 to 30)** items on page 90.



Now you must enter the preset value you need to set in the device. As stated, we want to set the **20 Preset** parameter = 100 dec. The Preset value must be entered next to the "PARAMETER VALUE [WRITE]" item in the CYCLIC CHANNEL: CONTROLLER => DEVICE section.

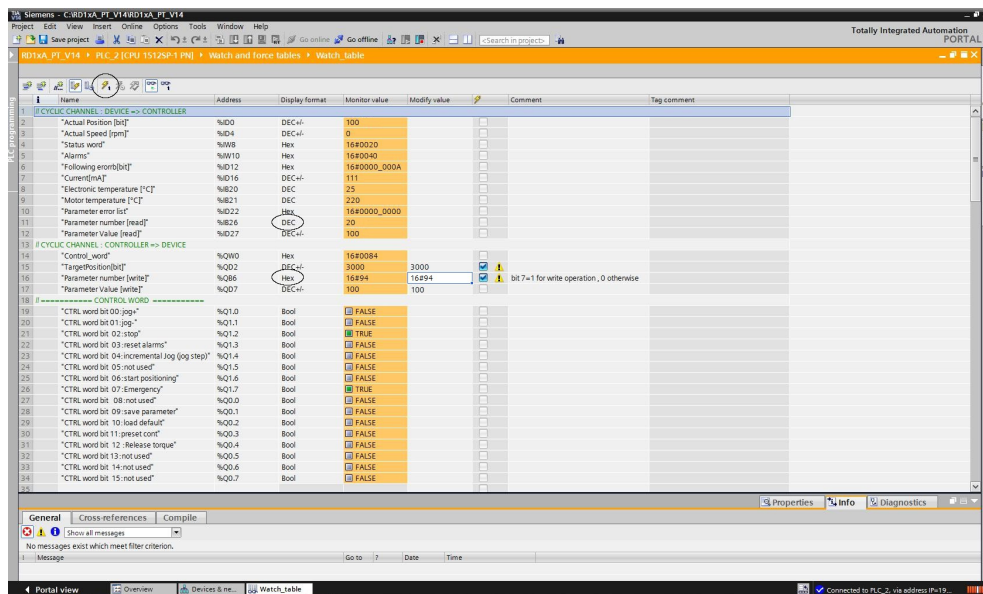
Then you must send a request to WRITE in the **20 Preset** parameter the value previously set next to the "PARAMETER VALUE [WRITE]" item. To do this you must enter both the index of the **20 Preset** parameter (=14h, 20 dec) and the command 80h for writing values (so, 14h + 80h = 94h) next to the "PARAMETER NUMBER [WRITE]" item in the CYCLIC CHANNEL: CONTROLLER => DEVICE section. Find more information next to the **Parameter number (Byte 6)** and **Parameter value (Bytes 7 ... 10)** items on page 82.

**Parameter value** = 100

(Preset value, by way of example).

**Parameter number** = 94hex

the index of the **20 Preset** parameter is 14h (see on page 98); then command 80h for writing values must be added (14h + 80h = 94h).



Finally press the **MODIFY ALL SELECTED PARAMETERS IMMEDIATELY AND ONCE** button in the toolbar to confirm the setting. This command is executed once and as quickly as possible without reference to any particular point in the user program. The operation of the button is available only in online mode.

To activate the new preset value set next to the **20 Preset** parameter, you must switch high and then low again the value of the **Setting the preset bit**, see on page 80.

The set value can be read next to the "PARAMETER VALUE [READ]" item in the CYCLIC CHANNEL: DEVICE => CONTROLLER section. The number of the parameter is displayed next to the "PARAMETER NUMBER [READ]" item in the CYCLIC CHANNEL: DEVICE => CONTROLLER section.

Furthermore, as **Current Position (Bytes 0 to 3) = 20 Preset** (the actuator should be in stop, as suggested), the same value can be read also next to the "ACTUAL POSITION [BIT]" item in the CYCLIC CHANNEL: DEVICE => CONTROLLER section.

Following a Preset operation, the Offset value is automatically stored in the memory.



#### NOTE

Please note that the **20 Preset** parameter is not available in the **Module parameters** tabbed page (otherwise it would be set at each switching on).



## 7.2.18.2 Reading and setting the 01 Distance per revolution parameter

In the example, first we want to read the **01 Distance per revolution** value that is actually set in the device and then enter the value 1000 dec.

To read the parameter value we must enter the number of the parameter we want to read in hexadecimal format next to the "PARAMETER NUMBER [WRITE]" in the CYCLIC CHANNEL: CONTROLLER => DEVICE section. The number of the **01 Distance per revolution** parameter is 01h, 1 dec. Find more information next to the **Parameter number (Byte 6)** item on page 82.

The value that is actually set in the **01 Distance per revolution** parameter will be shown next to the "PARAMETER VALUE [READ]" in the CYCLIC CHANNEL: DEVICE => CONTROLLER section: 1024 dec. Next to the "PARAMETER NUMBER [READ]" in the CYCLIC CHANNEL: DEVICE => CONTROLLER section the number of the parameter will be displayed: 1 dec, 01h. Find more information next to the **Parameter number (Byte 26)** and **Parameter value (Bytes 27 to 30)** items on page 90.

Name	Address	Display format	Monitor value	Modify value	Comment	Tag comment
<b>IF CYCLIC CHANNEL: DEVICE =&gt; CONTROLLER</b>						
"Actual Position [m]"	%D0	DEC+	204651			
"Actual Speed [rpm]"	%D4	DEC+	0			
"Status word"	%W8	Hex	16#0020			
"Alarm"	%B10	Hex	16#0040			
"Following error bit"	%D12	Hex	16#0000_000A			
"Current [mA]"	%D16	DEC+	111			
"Electric temperature [°C]"	%B20	DEC	25			
"Motor temperature [°C]"	%B21	DEC	220			
"Parameter error list"	%D22	Hex	16#0000_0000			
"Parameter number [read]"	%B26	DEC	1			
"Parameter value [read]"	%D27	DEC+	1024			
<b>IF CYCLIC CHANNEL: CONTROLLER =&gt; DEVICE</b>						
"Control word"	%QW0	Hex	16#0084			
"Target position [m]"	%QD2	DEC+	3000	3000		
"Parameter number [write]"	%QB6	Hex	16#14	16#01	bit 7=1 for write operation, 0 otherwise	
"Parameter value [write]"	%QD7	DEC+	0			
<b>CONTROL WORD</b>						
"CTRL word bit 00: jog"	%Q1.0	Bool	FALSE			
"CTRL word bit 01: jog"	%Q1.1	Bool	FALSE			
"CTRL word bit 02: stop"	%Q1.2	Bool	TRUE			
"CTRL word bit 03: reset alarms"	%Q1.3	Bool	FALSE			
"CTRL word bit 04: incremental jog (jog step)"	%Q1.4	Bool	FALSE			
"CTRL word bit 05: not used"	%Q1.5	Bool	FALSE			
"CTRL word bit 06: start positioning"	%Q1.6	Bool	FALSE			
"CTRL word bit 07: Emergency"	%Q1.7	Bool	TRUE			
"CTRL word bit 08: not used"	%Q0.0	Bool	FALSE			
"CTRL word bit 09: save parameter"	%Q0.1	Bool	FALSE			
"CTRL word bit 10: load default"	%Q0.2	Bool	FALSE			
"CTRL word bit 11: preset cont"	%Q0.3	Bool	FALSE			
"CTRL word bit 12: Release torque"	%Q0.4	Bool	FALSE			
"CTRL word bit 13: not used"	%Q0.5	Bool	FALSE			
"CTRL word bit 14: not used"	%Q0.6	Bool	FALSE			
"CTRL word bit 15: not used"	%Q0.7	Bool	FALSE			

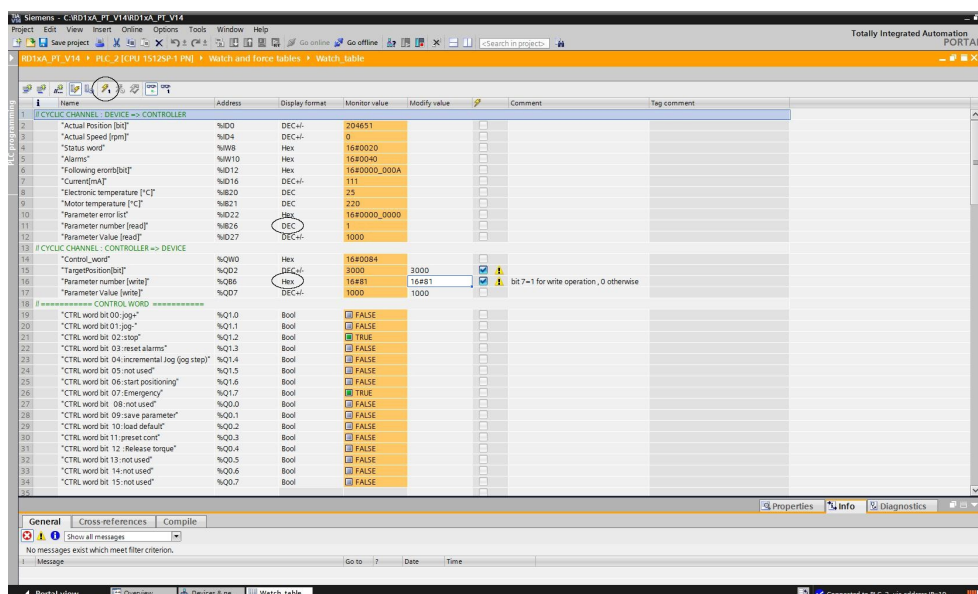
Now you must enter the **01 Distance per revolution** value you need to set in the device. As stated, we want to set the **01 Distance per revolution** parameter = 1000 dec. The **01 Distance per revolution** value must be entered next to the "PARAMETER VALUE [WRITE]" item in the CYCLIC CHANNEL: CONTROLLER => DEVICE section.

Then you must send a request to WRITE in the **01 Distance per revolution** parameter the value previously set next to the "PARAMETER VALUE [WRITE]" item. To do this you must enter both the index of the **01 Distance per**

**revolution** parameter (=01h, 1 dec) and the command 80h for writing values (so, 01h + 80h = 81h) next to the "PARAMETER NUMBER [WRITE]" item in the CYCLIC CHANNEL: CONTROLLER => DEVICE section. Find more information next to the **Parameter number (Byte 6)** and **Parameter value (Bytes 7 ... 10)** items on page 82.

**Parameter value = 1000** (01 Distance per revolution value, by way of example).

**Parameter number = 81hex** the index of the **01 Distance per revolution** parameter is 01h (see on page 92); then command 80h for writing values must be added (01h + 80h = 81h).



Finally press the **MODIFY ALL SELECTED PARAMETERS IMMEDIATELY AND ONCE** button in the toolbar to confirm the setting. This command is executed once and as quickly as possible without reference to any particular point in the user program. The operation of the button is available only in online mode.

The set value can be read next to the "PARAMETER VALUE [READ]" item in the CYCLIC CHANNEL: DEVICE => CONTROLLER section. The number of the parameter is displayed next to the "PARAMETER NUMBER [READ]" item in the CYCLIC CHANNEL: DEVICE => CONTROLLER section.



#### NOTE

Please note that the value set in the Cyclic Data Exchange mode via the Watch table will be overwritten at switching on by the value set in the **Module parameters** tabbed page.

If required, you can change the **01 Distance per revolution** value in the **Module parameters** tabbed page, see on page 69. This value will be sent to the device at each switching on.



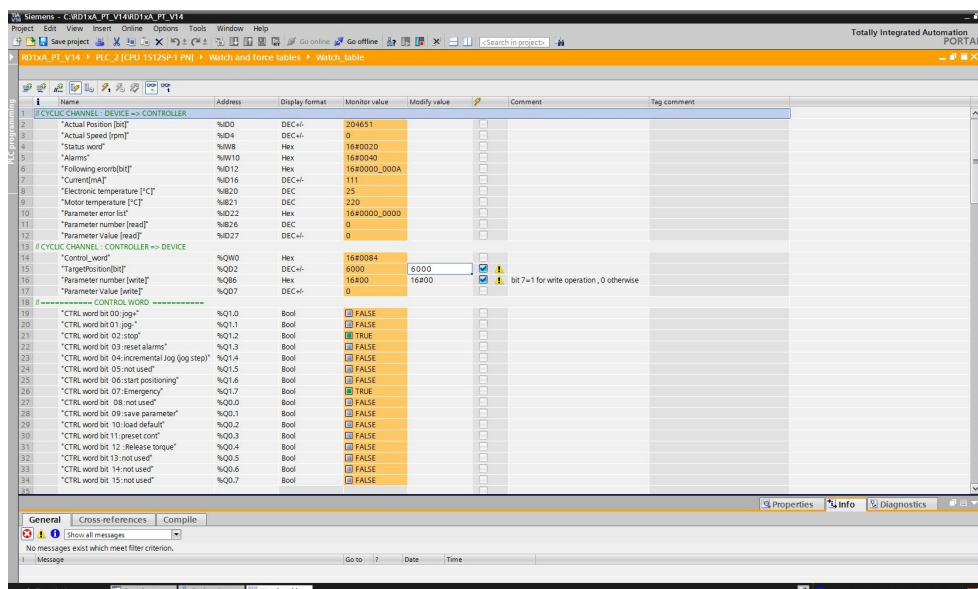
#### 7.2.18.3 Reading the **Current Position (Bytes 0 to 3)** parameter

The **Current Position (Bytes 0 to 3)** value is sent by the Device to the Controller via the Cyclic Data Exchange channel. It can be read next to the "ACTUAL POSITION [BIT]" item in the CYCLIC CHANNEL: DEVICE => CONTROLLER section.



#### 7.2.18.4 Setting the **Target Position (Bytes 2 to 5)** value

The **Target Position (Bytes 2 to 5)** value is sent by the Controller to the Device via the Cyclic Data Exchange channel. If you need to set a new target position value just enter a new value in the **Modify value** edit field next to the "TARGET POSITION [BIT]" item in the CYCLIC CHANNEL: CONTROLLER => DEVICE section.





## 7.4 Cyclic Data Exchange channel

Data described in the following pages is sent by either the Controller to the Device or by the Device to the Controller via the Cyclic Data Exchange channel.

The controller sends the following data to the device cyclically. Data is 32-byte long. See the "7.3.1 Controller → Device cyclic transmission" section on page 77.

Byte	Controller → Device (32 bytes)
0	Control Word (Bytes 0 and 1)
1	
2	Target Position (Bytes 2 to 5)
3	
4	
5	
6	Parameter number (Byte 6)
7	Parameter value (Bytes 7 ... 10)
8	
9	
10	
11 to 31	Not used

The device sends the following data to the controller cyclically. Data is 50-byte long. See the "7.3.2 Device → Controller cyclic transmission" section on page 84.

Byte	Device → Controller (50 bytes)
0	Current Position (Bytes 0 to 3)
1	
2	
3	
4	Current Speed [rpm] (Bytes 4 to 7)
5	
6	
7	
8	Status Word (Bytes 8 and 9)
9	
10	Alarms List (Bytes 10 and 11)
11	
12	Following error [pulse] (Bytes 12 to 15)
13	



14	Current [mA] (Bytes 16 to 19)
15	
16	
17	
18	
19	Electronics Temperature [°C] (Byte 20)
20	
21	Motor Temperature [°C] (Byte 21)]
22	Parameter Error List (Bytes 22 to 25)
23	
24	
25	
26	Parameter number (Byte 26)
27	Parameter value (Bytes 27 to 30)
28	
29	
30	
31 to 49	Not used

### 7.3.1 Controller → Device cyclic transmission

Data described in this section is sent cyclically by the Controller (PLC) to the Device (RD actuator).

#### Control Word (Bytes 0 and 1)

[Unsigned16, rw]

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

#### Byte 0

##### Jog +

bit 0

If the bit 4 **Incremental jog** = 0, as long as **Jog +** = 1, the Device moves toward the positive direction; otherwise if the bit 4 **Incremental jog** = 1, the activation of this bit causes a single step toward the positive direction having the length, expressed in pulses, set next to the **14 Jog step length** parameter to be executed at rising edge; then the actuator stops and waits for another command. Velocity, acceleration and deceleration are performed according to the values set next to the **11 Jog speed**, **07 Acceleration** and **08 Deceleration** parameters respectively. For a detailed description of the jog control see on page 44.



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

**Jog -**  
bit 1

If the bit 4 **Incremental jog** = 0, as long as **Jog -** = 1, the Device moves toward the negative direction; otherwise if the bit 4 **Incremental jog** = 1, the activation of this bit causes a single step toward the negative direction having the length, expressed in pulses, set next to the **14 Jog step length** parameter to be executed at rising edge; then the actuator stops and waits for another command. Velocity, acceleration and deceleration are performed according to the value set next to the **11 Jog speed**, **07 Acceleration** and **08 Deceleration** parameters respectively. For a detailed description of the jog control see on page 44.



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

**Stop**  
bit 2

If set to "1" the Device is allowed to execute the movements as commanded. If, while the unit is running, this bit switches to "0", then the Device must stop executing the deceleration procedure set in **08 Deceleration**. For an immediate halt in the movement, use the bit 7 **Emergency**.

**Alarm reset**  
bit 3

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



Please note that should the alarm be caused by wrong parameter values (see **Machine data not valid** and **Parameter Error List (Bytes 22 to 25)**), the normal work status can be restored only after having set proper values. The **Flash memory error** alarm cannot be reset.

### Incremental jog

bit 4

If set to "0", the activation of the bits **Jog +** and **Jog -** causes the Device to move as long as **Jog + / Jog -** = 1. Setting this bit to 1 the incremental jog function is enabled, that is: the activation of the bits **Jog +** and **Jog -** causes a single step toward the positive or negative direction having the length, expressed in pulses, set next to the **14 Jog step length** parameter to be executed at rising edge; then the actuator stops and waits for another command.

bit 5

Not used.

### Start

bit 6

When it is set to "1" the device moves in order to reach the set target position (see **Target Position (Bytes 2 to 5)** on page 81). For a complete description of the position control see on page 45.



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

### Emergency

bit 7

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

### Byte 1

bit 8

Not used.

### Save parameters

bit 9



Data is saved on non-volatile memory at each rising edge of the bit; in other words, save is performed each time this bit is switched from logic level low ("0") to logic level high ("1"). Always save the new values after setting in order to store them in the non-volatile memory permanently. Should the power supply be turned off all data that has not been saved previously will be lost!

### Load default parameters

bit 10

The default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) are restored at each



rising edge of the bit; in other words, the default parameters loading operation is performed each time this bit is switched from logic level low ("0") to logic level high ("1"). The complete list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 167.

Always save the new values after setting in order to store them in the non-volatile memory permanently. Should the power supply be turned off all data that has not been saved previously will be lost!

#### WARNING

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

#### Setting the preset

bit 11

It sets the current position to the value set next to the **20 Preset** parameter. The operation is performed at each rising edge of the bit, i.e. each time this bit is switched from logic level low ("0") to logic level high ("1"). We suggest activating the preset when the actuator is in stop. For more information refer to page 98.

#### Release axis torque

bit 12

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

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

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

#### OUT 1

bit 13

This is intended to activate / deactivate the operation of the digital output 1. The meaning of the available output is described in the "6.3 Digital inputs and output" section on page 46.

**OUT 1 = 0**      output 1 low (not active)

**OUT 1 = 1**      output 1 high (active)

**Brake disabled**

bit 14

This function is available only in the RD12A version (model equipped with brake); in the RD1A version (model without brake) the bit 14 is not used. RD12A model is equipped with a brake designed to activate as soon as the motor comes to a stop in order to prevent it from moving. Setting this bit to "1" causes the brake to be disabled and not operating; 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 (Bytes 2 to 5)**

[Signed32, rw]

It sets the position to be reached, otherwise referred to as commanded position. When the **Start** command is sent while the **Stop** and **Emergency** bits are "1" and the alarm condition is off, the device moves in order to reach the target position set next to this parameter.

As soon as the axis is within the tolerance window limits set next to the **02 Position window** parameter, the bit 8 **Target position reached** in the **Status Word (Bytes 8 and 9)** goes high ("1"). When the position is within the tolerance window limits set next to the **02 Position window** parameter, after the delay set next to the **03 Position window time** parameter, the bit 0 **Axis in position** in the **Status Word (Bytes 8 and 9)** goes high ("1").

For more information refer also to the "Positioning: position and speed control" section on page 45.

Default = 0 (min. = 0, max. = within **22 Pos. Limit Switch [pulse]** / **23 Neg. Limit Switch [pulse]**)

**NOTE****Position override function**

It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the **Target Position (Bytes 2 to 5)** parameter. See also on page 45.



#### NOTE

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

#### Parameter number (Byte 6)

This byte is used to set the index of the parameter / variable whose value is set / read in the next four bytes 7...10.

For the complete list of the available parameters and variables and their meaning please refer to the "7.4 Profinet module parameters" section on page 92.

bit 7	bit 6	bit 5 ... 0
R/W	0	Parameter / variable index

R/W = 0 reading the parameter

R/W = 1 writing the parameter



#### EXAMPLE 1

You need to write a value next to the **14 Jog step length** parameter (index 0Eh), so set 0x0E = 1000 1110:

	R/W	-	Parameter / variable index					
bit	7	6	5	4	3	2	1	0
binary	1	0	0	0	1	1	1	0

where:

bit 7 = R/W = 1, i.e. "writing the parameter"

bit 6 = bit always set to 0

bit 5 ... 0 = parameter index = 001110 binary value = 0Eh = **14 Jog step length**

The value to be set must be entered in the next four bytes 7...10 - **Parameter value (Bytes 7 ... 10)**.

The set **14 Jog step length** variable value will be sent back through **Parameter number (Byte 26)** and **Parameter value (Bytes 27 to 30)**, refer to "7.3.2 Device → Controller cyclic transmission" section on page 84.



### EXAMPLE 2

You need to read the value next to the **27 Gear ratio** variable (index 1Bh), so set 0x1B = 0001 1011:

	R/W	-	Parameter / variable index					
bit	7	6	5	4	3	2	1	0
binary	0	0	0	1	1	0	1	1

where:

bit 7 = R/W = 0, i.e. "reading the parameter"

bit 6 = bit always set to 0

bit 5 ... 0 = variable index = 011011 binary value = 1Bh = **27 Gear ratio**

When a request to read a parameter is sent the next four bytes 7...10 -**Parameter value (Bytes 7 ... 10)**- are ignored.

The **27 Gear ratio** variable value will be sent back through **Parameter number (Byte 26)** and **Parameter value (Bytes 27 to 30)**, refer to "7.3.2 Device → Controller cyclic transmission" section on page 84.

### Parameter value (Bytes 7 ... 10)

These bytes contain the value to be assigned to the parameter whose index is set in the previous byte 6. Four data bytes are available for each parameter.

For the complete list of the available parameters and variables and their meaning please refer to the "7.4 Profinet module parameters" section on page 92.

byte 7	byte 8	byte 9	byte 10
Low	...	...	High

**Bytes 11 to 31**      Not used

### 7.3.2 Device → Controller cyclic transmission

Data described in this section is sent cyclically by the Device (RD actuator) to the Controller (PLC).

#### Current Position (Bytes 0 to 3)

[Signed32, ro]

Current position of the device expressed in pulses.

#### Current Speed [rpm] (Bytes 4 to 7)

[Signed32, ro]

Speed of the output shaft after the reduction gears expressed in revolutions per minute [rpm]. The speed of the motor will be, for example, as follows:

Speed at output: T12 = 166 rpm → Motor speed = 2000 rpm (166 \* 12).

For further information refer to the **11 Jog speed** / **12 Work speed** parameters.

#### Status Word (Bytes 8 and 9)

[Unsigned16, ro]

These bytes provide information about the current state of the device.

#### Byte 8

##### Axis in position

bit 0

The bit value is "1" when the device reaches and keeps the commanded position (**Target Position (Bytes 2 to 5)**) for the time set next to the **03 Position window time** parameter. It is kept active until the position error is lower than **02 Position window**. For further information please refer to the "Positioning: position and speed control" section on page 45.

bit 1

Not used.

##### Drive enabled

bit 2

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

##### SW limit switch +

bit 3

The bit value switches to "1" as soon as the device reaches the maximum positive limit (positive limit



switch). For more information see the **09 Positive delta** parameter on page 94.

**SW limit switch –**

bit 4

The bit value switches to "1" as soon as the device reaches the maximum negative limit (negative limit switch). For more information see the **10 Negative delta** parameter on page 95.

**Alarm**

bit 5

The value is "1" when an alarm occurs, see details in the **Alarms List (Bytes 10 and 11)** bytes on page 87.

**Axis running**

bit 6

The value is "0" when the device is not moving.  
The value is "1" while the device is moving.

**Executing a command**

bit 7

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

**Byte 9**

**Target position reached**

bit 8

The value is "1" when the device reaches the target position set next to the **Target Position (Bytes 2 to 5)** bytes (it is within the limits set next to the **02 Position window** parameter). The bit is kept active until a new **Target Position (Bytes 2 to 5)** value or the **Alarm reset** command are sent. For more information refer also to the "Positioning: position and speed control" section on page 45.

**Button 1 Jog +**

bit 9

RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. As long as the button 1 JOG + is pressed, the bit 9 is forced high "1"; when the button 1 is not pressed, the bit 9 is low "0". For further information see the "4.4 Screw plug for internal access (Figure 4 and Figure 7)" section on page 39 and the "4.5.1 JOG + and JOG – buttons (Figure 8)" section on page 40.

## Button 2 Jog –

bit 10

RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. As long as the button 2 JOG – is pressed, the bit 10 is forced high "1"; when the button 2 is not pressed, the bit 10 is low "0". For further information see the "4.4 Screw plug for internal access (Figure 4 and Figure 7)" section on page 39 and the "4.5.1 JOG + and JOG – buttons (Figure 8)" section on page 40.

## 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, the bit 11 is forced high "1"; when the button 3 is not pressed, the bit 11 is low "0". For further information see the "4.4 Screw plug for internal access (Figure 4 and Figure 7)" section on page 39 and the "4.5.2 PRESET button (Figure 8)" section on page 41.

## PWM saturation

bit 12

The bit value switches to "1" when the current supplied for controlling the motor phases reaches the saturation point and cannot be increased further. The motor operation is affected by excessive dynamics or something is impeding the movement.

## 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 the "6.3 Digital inputs and output" section on page 46.

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 the "6.3 Digital inputs and output" section on page 46.

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

the "6.3 Digital inputs and output" section on page 46.

IN 3 = 0            input 3 low (not active)

IN 3 = 1            input 3 high (active)

### Alarms List (Bytes 10 and 11)

[Unsigned16, ro]

These bytes are meant to signal the alarms that are currently active in the device.

#### Byte 10

##### Machine data not valid

bit 0            One or more parameters are not valid, set proper values to restore the normal work condition. See the list of the wrong parameters in the [Parameter Error List \(Bytes 22 to 25\)](#) bytes.

##### Flash memory error

bit 1            Internal error, it cannot be restored.

##### Counting error

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

##### Following error

bit 3            The difference between the real position and the theoretical position is greater than the value set in the [04 Max following error](#) parameter; we suggest reducing the work speed.

##### Axis not synchronized

bit 4            Internal error, it cannot be restored.

##### Target not valid

bit 5            The set target position (see [Target Position \(Bytes 2 to 5\)](#)) is over the maximum travel limits.

##### Emergency

bit 6            Bit 7 **Emergency** in [Control Word \(Bytes 0 and 1\)](#) has been forced to low value (0); or alarms are active in the unit.

##### Overcurrent

bit 7            Motor overcurrent.

## Byte 11

### Electronics Overtemperature

bit 8                      The temperature of the MOSFETs detected by an internal probe is exceeding the maximum ratings (see [Electronics Temperature \[°C\] \(Byte 20\)](#) on page 89). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the allowed range.

### Motor Overtemperature

bit 9                      The temperature of the motor detected by an internal probe is exceeding the maximum ratings (see [Motor Temperature \[°C\] \(Byte 21\)](#) on page 89). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the allowed range.

### Undervoltage

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

### Address not valid

bit 11                     The number of the requested parameter (address) is not valid.

### Read-only

bit 12                     You tried to set a read-only parameter (ro parameter), it cannot be written to.

bit 13                     Not used.

### Hall sequence

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

### Overvoltage

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

To reset a faulty condition use the **Alarm reset** command, bit 3 in the [Control Word \(Bytes 0 and 1\)](#). In a normal work condition the **Alarm reset** bit is set to "0". Setting the bit to "1" causes the normal work status of the device to be

restored. The normal work status is resumed by switching this bit from "0" to "1". This command resets the alarm but only if the fault condition has ceased.



Please note that should the alarm be caused by wrong parameter values (see **Machine data not valid** and **Parameter Error List (Bytes 22 to 25)**), the normal work status can be restored only after having set proper values. **Flash memory error** and **Axis not synchronized** alarms cannot be reset.

#### Following error [pulse] (Bytes 12 to 15)

[Signed32, ro]

These bytes contain the difference between the target position and the current position step by step. If this value is greater than the one set in the **04 Max following error** parameter, then the **Following error** alarm is triggered and the unit stops. The value is expressed in pulses.

#### Current [mA] (Bytes 16 to 19)

[Signed32, ro]

These bytes inform about the value of the current absorbed by the motor (rated current). The value is expressed in milliamperes (mA).

#### Electronics Temperature [°C] (Byte 20)

[Signed8, ro]

This byte informs about the temperature of the electronics as detected by internal probes. The value is expressed in Celsius degrees (°C). The minimum detectable temperature is -20°C.

#### Motor Temperature [°C] (Byte 21)

[Signed8, ro]

This byte informs about the temperature of the motor as detected by internal probes. The value is expressed in Celsius degrees (°C). The minimum detectable temperature is -20°C.

#### Parameter Error List (Bytes 22 to 25)

[Unsigned32, ro]

The operator has set invalid data and the **Machine data not valid** alarm has been triggered. These bytes are meant to inform about the specific wrong parameters, according to the information in the following table.

Please note that the normal work status can be restored only after having set proper values.

Bit	Parameter
0	Not used
1	01 Distance per revolution
2	07 Acceleration
3	08 Deceleration
4	09 Positive delta
5	10 Negative delta
6	11 Jog speed
7	12 Work speed
8	13 Code sequence
9	20 Preset
10	14 Jog step length
11	05 Proportional gain
12	06 Integral gain
13	03 Position window time
14	04 Max following error
15 to 31	Not used

#### Parameter number (Byte 26)

This byte contains the index of the parameter / variable whose value is shown in the four following bytes 27 to 30.

Indexes are listed in the "7.4 Profinet module parameters" section on page 92.

### Parameter value (Bytes 27 to 30)

These bytes contain the value assigned to the parameter / variable whose number is shown in the previous byte 26. Four data bytes are available for each parameter.

For the complete list of the available parameters and their meaning please refer to the "7.4 Profinet module parameters" section on page 92.

byte 27	byte 28	byte 29	byte 30
Low	...	...	High

**Bytes 31 to 49**    Not used

#### 7.4 Profinet module parameters

When the system is turned on, machine data set by the operator is sent to the Device by the Controller via the Acyclic Data Exchange. Parameters can be found and set if required in the **Module parameters** inspector window of TIA Portal (see on page 69).

They are described as follows.

##### Index Parameter/Variable name

[data type, attribute]

- Index is expressed in decimal notation
- Attribute:
  - ro = read only access
  - rw = read and write access



##### NOTE

Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the **Save parameters** function available in the **Control Word (Bytes 0 and 1)**, see on page 77.

Should the power supply be turned off all data that has not been saved previously will be lost!

Parameters 01 to 14 are sent to the device at each switching on and can be changed in the **Module parameters** page, see on page 69.

#### 01 Distance per revolution

[Unsigned16, rw]

This parameter sets the number of pulses per each complete revolution of the shaft. It is useful to relate the revolution of the shaft and a linear measurement. For example: the unit is joined to a worm screw having 5 mm (0.196") pitch; by setting **01 Distance per revolution** = 500, at each shaft revolution the system performs a 5 mm (0.196") 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 next to the **20 Preset** variable. For a detailed explanation see on page 47 and the relevant parameters.

Please note that the parameters/variables listed hereafter are closely related to the **01 Distance per revolution** parameter; hence when you change the value



in **01 Distance per revolution** also the value in each of them necessarily changes. They are: **02 Position window**, **04 Max following error**, **09 Positive delta**, **10 Negative delta**, **Target Position (Bytes 2 to 5)**, **Current Position (Bytes 0 to 3)** and **Following error [pulse] (Bytes 12 to 15)**.



#### NOTE

If **01 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.

#### 02 Position window

[Unsigned16, rw]

This parameter defines the tolerance window limits for the **Target Position (Bytes 2 to 5)** value. As soon as the axis is within the tolerance window limits, the bit 8 **Target position reached** in the **Status Word (Bytes 8 and 9)** goes high ("=1"). When the axis is within the tolerance window limits for the time set in the **03 Position window time** parameter, the bit 0 **Axis in position** in the **Status Word (Bytes 8 and 9)** goes high ("=1"). The parameter is expressed in pulses. See also the "Positioning: position and speed control" section on page 45. Default = 1 (min. = 0, max. = 65535)

#### 03 Position window time

[Unsigned16, rw]

It represents the time for which the axis has to be within the tolerance window limits set in the **02 Position window** parameter before the state is signalled through the **Axis in position** status bit of the **Status Word (Bytes 8 and 9)**. The parameter is expressed in milliseconds (ms). See also the "Positioning: position and speed control" section on page 45. Default = 0 (min. = 0, max. = 10000)

#### 04 Max following error

[Unsigned32, rw]

This parameter defines the maximum allowable difference between the real position and the theoretical position of the device. If the device detects a value higher than the one set in this parameter, the **Following error** alarm is triggered and the unit stops. The parameter is expressed in pulses. Default = 1024 (min. = 0, max. = 65535)

### 05 Proportional gain

[Unsigned16, rw]

This parameter contains the proportional gain used by the PI controller for the position loop. The value has been optimized by Lika Electronic according to the technical characteristics of the device.

Default = 300 (min. = 0, max. = 1000)

### 06 Integral gain

[Unsigned16, rw]

This parameter contains the integral gain used by the PI controller for the position loop. The value has been optimized by Lika Electronic according to the technical characteristics of the device.

Default = 10 (min. = 0, max. = 1000)

### 07 Acceleration

[Unsigned16, rw]

This parameter defines the acceleration value that has to be used by the device when reaching either the **11 Jog speed** or the **12 Work speed**. The parameter is expressed in revolutions per second<sup>2</sup> [rev/s<sup>2</sup>]. See also the "6.2 Movements: jog and positioning" section on page 44.

Default = 10 (min. = 1, max. = 500)

### 08 Deceleration

[Unsigned16, rw]

This parameter defines the deceleration value that has to be used by the device when stopping. The parameter is expressed in revolutions per second<sup>2</sup> [rev/s<sup>2</sup>]. See also the "6.2 Movements: jog and positioning" section on page 44.

Default = 10 (min. = 1, max. = 500)

### 09 Positive delta

[Unsigned32, rw]

This parameter is used to calculate the maximum forward (positive) limit the device is allowed to reach starting from the preset value. As soon as the maximum forward limit is reached, the condition is signalled through the **SW limit switch +** status bit of the **Status Word (Bytes 8 and 9)** (the bit is forced high). The parameter is expressed in pulses.

**SW limit switch +** = **20 Preset** + **09 Positive delta**. The maximum positive limit can be read next to the **22 Pos. Limit Switch [pulse]** variable.

For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.

Default = 523 263 (min. = 0, max. = 523 263)


**WARNING**

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


**EXAMPLE**

When **01 Distance per revolution** = 1,024 and **20 Preset** = 0, the maximum acceptable value for **09 Positive delta** is:

$(1,024 \text{ steps per revolution} * 512 \text{ revolutions}) - 1 \text{ step} - 1,024 \text{ steps (i.e. 1 revolution for safety reasons)} = 523\,263$

When **01 Distance per revolution** = 256 and **20 Preset** = 0, the maximum acceptable value for **09 Positive delta** is:

$(256 \text{ steps per revolution} * 512 \text{ revolutions}) - 1 \text{ step} - 256 \text{ steps (i.e. 1 revolution for safety reasons)} = 130\,815$

See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.


**WARNING**

Every time **01 Distance per revolution** parameter and **20 Preset** variable are changed, **09 Positive delta** and **10 Negative delta** values have to be checked carefully. Each time you change the value in **01 Distance per revolution**, then you must update the value in **20 Preset** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **20 Preset** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **09 Positive delta** and **10 Negative delta** parameters. For a detailed explanation see on page 47.

**10 Negative delta**

[Unsigned32, rw]

This value is used to calculate the maximum backward (negative) limit the device is allowed to reach starting from the preset value. As soon as the maximum backward limit is reached, the condition is signalled through the **SW limit switch** – status bit of the **Status Word (Bytes 8 and 9)** (the bit is forced high). The parameter is expressed in pulses.

**SW limit switch** – = **20 Preset** - **10 Negative delta**. The maximum negative limit can be read next to the **23 Neg. Limit Switch [pulse]** variable.

For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.

Default = 523 263 (min. = 0, max. = 523 263)



#### WARNING

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



#### EXAMPLE

When **01 Distance per revolution** = 1,024 and **20 Preset** = 0, the maximum acceptable value for **10 Negative delta** is:

$(1,024 \text{ steps per revolution} * 512 \text{ revolutions}) - 1 \text{ step} - 1,024 \text{ steps (i.e. 1 revolution for safety reasons)} = 523\,263$

When **01 Distance per revolution** = 256 and **20 Preset** = 0, the maximum acceptable value for **10 Negative delta** is:

$(256 \text{ steps per revolution} * 512 \text{ revolutions}) - 1 \text{ step} - 256 \text{ steps (i.e. 1 revolution for safety reasons)} = 130\,815$

See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.



#### WARNING

Every time **01 Distance per revolution** parameter and **20 Preset** variable are changed, **09 Positive delta** and **10 Negative delta** values have to be checked carefully. Each time you change the value in **01 Distance per revolution**, then you must update the value in **20 Preset** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **20 Preset** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **09 Positive delta** and **10 Negative delta** parameters. For a detailed explanation see on page 47.

### 11 Jog speed

[Unsigned16, rw]

This parameter contains the maximum speed the motor is allowed to reach when using the **Jog +** and **Jog -** functions (see **Control Word (Bytes 0 and 1)**). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 44.

Default = 2000 (min. = 10, max. = 3000)



**NOTE**

Please note that this is the speed of the motor, not the speed of the output shaft after the reduction gears.

The speed at output will be as follows:

Motor speed = 2000 rpm

Speed at output: T12 = 166 rpm

T24 = 83 rpm

T48 = 41 rpm

T92 = 21 rpm

## 12 Work speed

[Unsigned16, rw]

This parameter contains the maximum speed the motor is allowed to reach in automatic work mode (movements are controlled using the **Start** and **Stop** commands -see **Control Word (Bytes 0 and 1)**- and are performed in order to reach the position set in **Target Position (Bytes 2 to 5)**). The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 45.

Default = 2000 (min. = 10, max. = 3000)



**NOTE**

Please note that this is the speed of the motor, not the speed of the output shaft after the reduction gears.

The speed at output will be as follows:

Motor speed = 2000 rpm

Speed at output: T12 = 166 rpm

T24 = 83 rpm

T48 = 41 rpm

T92 = 21 rpm

## 13 Code sequence

[Unsigned16, rw]

It sets whether the position value output by the device increases (count up information) when the shaft rotates clockwise (0) or counter-clockwise (1). Clockwise and counter-clockwise rotations are viewed from the shaft side.

0 = count up information with clockwise rotation (default)

1 = count up information with counter-clockwise rotation



**WARNING**

Changing this value causes also the position calculated by the controller to be necessarily affected. Therefore it is compulsory to set a new value in the **20**

**Preset** variable and then check the values set next to the **09 Positive delta** and **10 Negative delta** parameters.

#### 14 Jog step length

[Unsigned16, rw]

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

Default = 1000 (min. = 1, max. = 10000).

Parameters 20 to 27 can be accessed by the Controller but are not available in the **Module parameters** page, see on page 69.

#### 20 Preset

[Signed32, rw]

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

Default = 0 (min. = -1 048 576, max. = +1 048 576)



#### NOTE

We suggest activating the preset when the actuator is in stop. See the **Setting the preset** command on page 80.



#### WARNING

A new value has to be set in the **20 Preset** variable every time the **01 Distance per revolution** and **13 Code sequence** values are changed. After having entered a new value in **20 Preset** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in the **09 Positive**

**delta** and **10 Negative delta** parameters. For a detailed explanation see on page 47.

## 21 Position Offset

[Signed32, ro]

This variable informs about the difference between the position value transmitted by the device and the real position: real position – preset. The value is expressed in pulses.

## 22 Pos. Limit Switch [pulse]

[Signed32, ro]

This is the **SW limit switch +** value (maximum positive limit) calculated according to values set next to the **20 Preset** variable and the **09 Positive delta** parameter. When the maximum forward limit is reached, the condition is signalled through the **SW limit switch +** status bit 3 of the **Status Word (Bytes 8 and 9)**.

**SW limit switch + = 20 Preset + 09 Positive delta.**

The value is expressed in pulses.

Refer also to the EXAMPLE 1 in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.

Default = 523 263

## 23 Neg. Limit Switch [pulse]

[Signed32, ro]

This is the **SW limit switch -** value (maximum negative limit) calculated according to values set next to the **20 Preset** variable and the **10 Negative delta** parameter. When the maximum backward limit is reached, the condition is signalled through the **SW limit switch -** status bit 4 of the **Status Word (Bytes 8 and 9)**.

**SW limit switch - = 20 Preset - 10 Negative delta.**

The value is expressed in pulses.

Refer also to the EXAMPLE 1 in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.

Default = - 523 263

## 24 HMS Serial Number

[Unsigned32, ro]

It shows the serial number of the HMS module.

Value = device dependent

## 25 Software version

[Unsigned16, ro]

It contains the software version of the device.

The meaning of the 16 bits in the index 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 258 in decimal notation corresponds to the binary representation 00000001 00000010 and has to be interpreted as: 01 02 hex, i.e. version 1.2.

## 26 Hardware version

[Unsigned16, ro]

It contains the hardware version of the device.

The meaning of the 16 bits in the index is as follows:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
RD1xA interface								Brake	-		Hardware version				

where:

00 ... 03	= hardware version
04 ... 06	= bits not used
07	= brake (0 = without brake; 1 = with brake)
08 ... 15	= RD1xA interface (33h = POWERLINK; 34h = EtherCAT; 35h = MODBUS TCP; 36h = EtherNet/IP; 37h = Profinet; 00h ... 32h & 38h ... FF = not used)

Value 3781 in hexadecimal notation corresponds to the binary representation 0011 0111 1000 0001 and has to be interpreted as follows: hardware version 1 (bit 0-3 = 1); device fitted with brake (bit 7 = 1); RD12A model with Profinet interface (bit 8-15 = 37h).



## 27 Gear ratio

[Unsigned32, ro]

It informs about the gear ratio of the reduction gear installed between the motor and the encoder shaft of the unit. 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



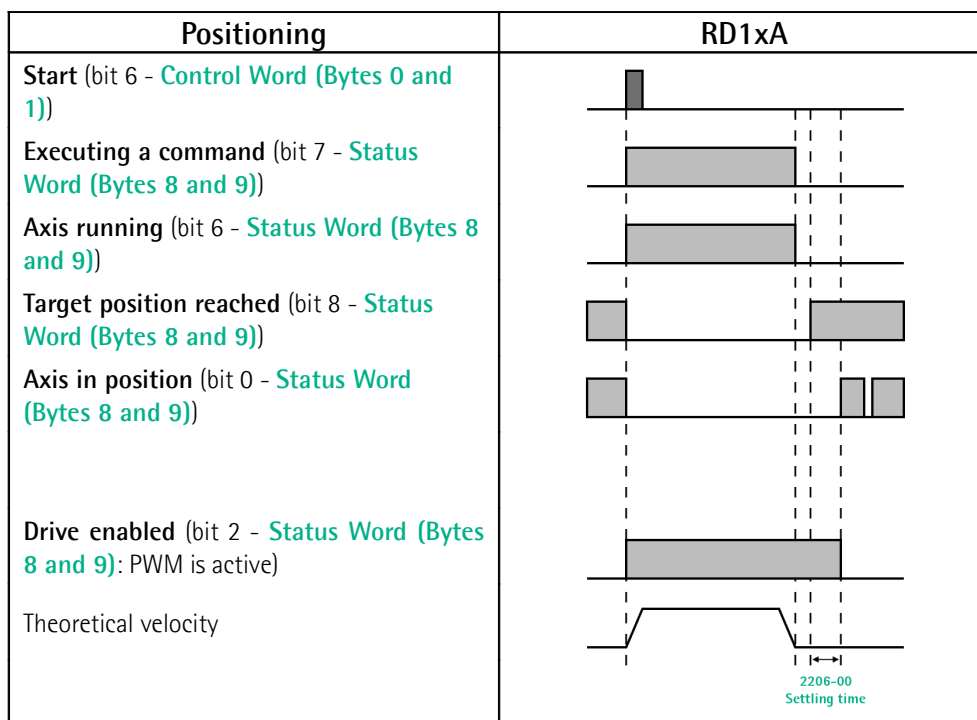
### NOTE

Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the **Save parameters** function available in the **Control Word (Bytes 0 and 1)**, see on page 77.

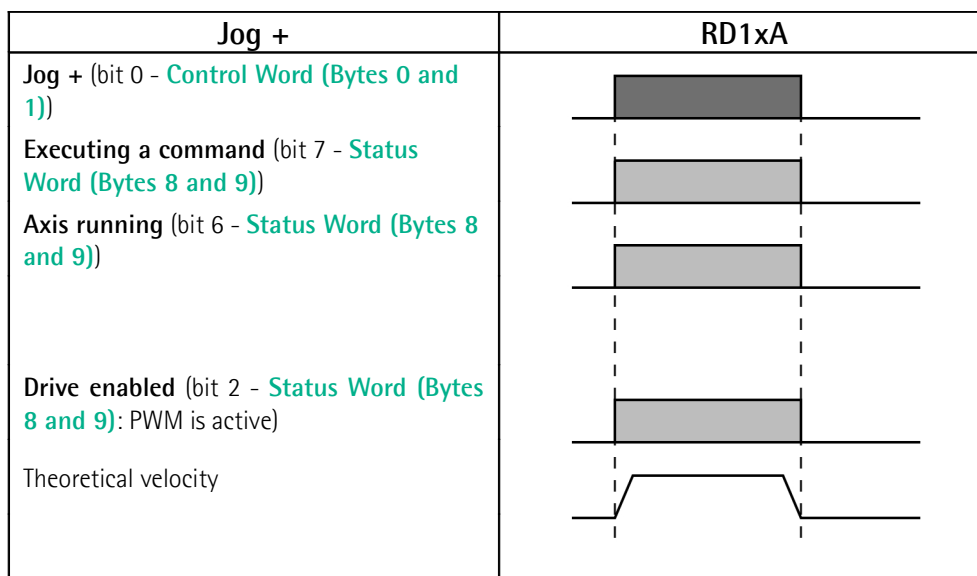
Should the power supply be turned off all data that has not been saved previously will be lost!



### EXAMPLE 1



### EXAMPLE 2



## 8 Modbus® interface

Lika DRIVECOD positioning units are Slave devices and implement the Modbus application protocol (level 7 of the OSI model) and the "Modbus over Serial Line" protocol (levels 1 & 2 of the OSI model).

For any further information or omitted specifications please refer to the "Modbus Application Protocol Specification V1.1b" and "Modbus over Serial Line. Specification and Implementation Guide V1.02" available at [www.modbus.org](http://www.modbus.org).

### 8.1 Configuring the device using Lika's setting up software

RD1xA DRIVECOD positioning units can be equipped with several communication interfaces such as EtherCAT, POWERLINK, MODBUS RTU, Profibus-DP, CANopen DS 301, etc. All versions except the MODBUS RTU one are equipped with an RS-232 service serial port in compliance with the MODBUS protocol. It can be used to configure the actuator. For this purpose all versions are supplied with a software expressly developed and released by Lika Electronic in order to allow an easy set up of the device. The program allows the operator to set the working parameters of the device; control manually some movements and functions; and monitor whether the device is running properly. The program is supplied for free and can be installed in any PC fitted with a Windows operating system (Windows XP or later). The executable file to be used to launch the program is **SW\_RD1xA\_LKC742\_Vx.EXE** and is available in the enclosed documentation or at the address [www.lika.biz](http://www.lika.biz) > **ROTARY ACTUATORS** > **ROTARY ACTUATORS/POSITIONING UNITS (DRIVECOD)**. The program is designed to be installed simply by copying the executable file to the desired location and there is **no installation** process. To launch it just double-click the file icon. To close the program press the **DISCONNECT** button in the **Serial Configuration** page and then click the **CLOSE** button in the title bar.



#### WARNING

Please be sure to comply with the following compatibilities between the HW-SW version of the actuator and the software release of the Modbus executable file.

Compatibility	HW-SW	EXE Modbus
	2 - 1.0	from V1.2 to ...



**NOTE**

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

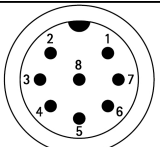
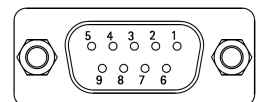
**On the DRIVECOD side the cable must be connected to the M12 8-pin male connector service serial port.** See the "Electrical connections" section on page 30.

A connection assembly fitted with M12 8-pin / USB connectors is available on request; please contact Lika Electronic Technical Assistance & After Sale Service and quote the following items: **IF92 converter + EC-RD1A-M12M8 connection cable.**



**NOTE**

If you use the IF92 converter + connection cable, you are required to install the drivers of the USB Serial Converter and the USB Serial Port first. The drivers are available in the Software folder of the actuator and downloadable from Lika's web site.

			
Function	RS-232 M12 8-pin male connector (actuator)	9-pin D-SUB female connector (PC)	Function
TD	6	2	RD
RD	7	3	TD
0Vdc	8	5	0Vdc

Always make sure that the RD of the DRIVECOD unit is cross-wired to the TD of the PC and the TD of the PC is cross-wired to the RD of the DRIVECOD unit.

Please note that the configuration parameters of the MODBUS service serial port have fixed values, so the user cannot change them.

They are:

### RS-232 Modbus

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

The MODBUS address is 1. It cannot be changed. See the "8.2 "Serial configuration" page" section hereafter.

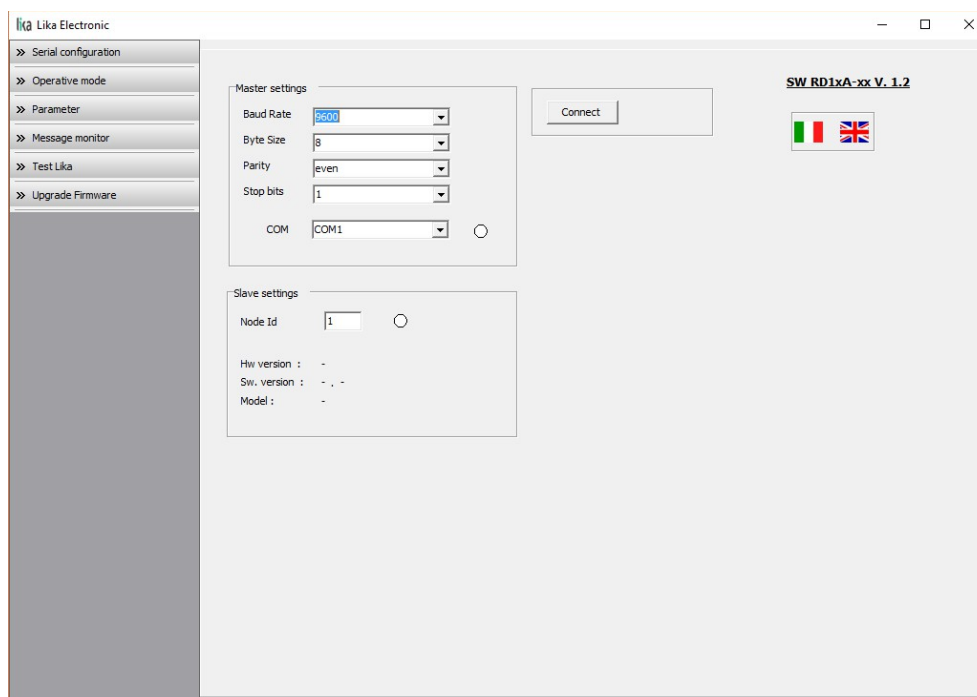




### NOTE

Please note that only the L5 LED (controller power supply) and the L6 LED (motor power supply) operation is available when the MODBUS interface is active.

## 8.2 "Serial configuration" page

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

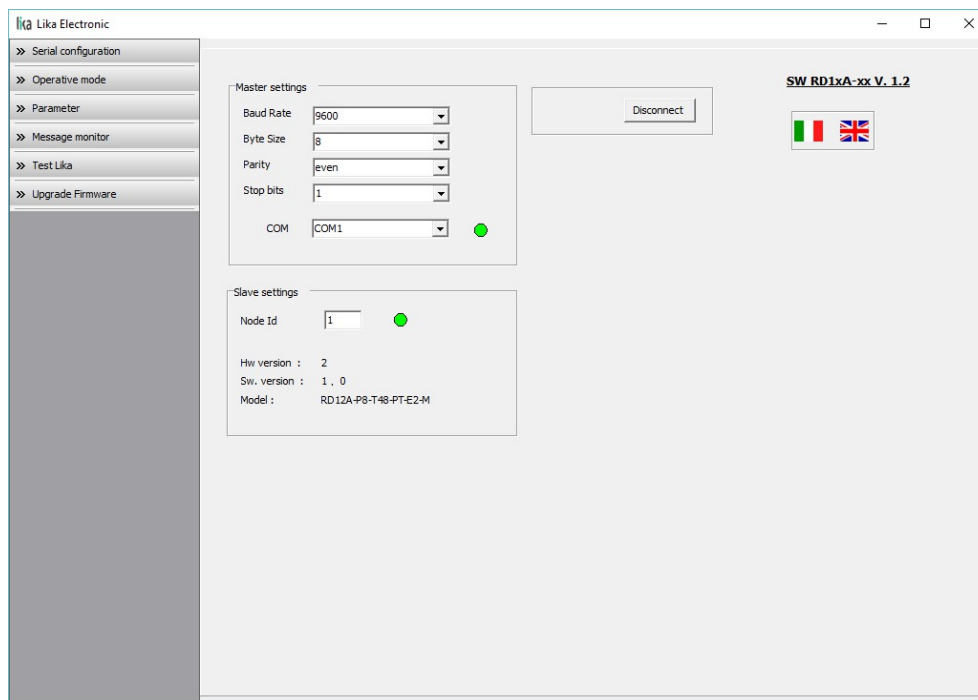


First of all this page allows the operator to choose the language used to display texts and items in the user interface. Click on the **Italian flag**  icon to choose the Italian language; click on the **UK flag**  icon to choose the English language.

The **Master settings** box in the left side of the page allows you to choose the serial port of the personal computer the RD1xA unit is connected to (**COM** drop-down box) and then set the configuration parameters. Serial port settings in the personal computer must compulsorily match those in the connected Lika device.

**For serial port settings see the previous section.**

Then set the node address of the device the personal computer is connected to through the **Slave settings** box (for all RD1xA-Modbus positioning units = **1**). Now you are ready to establish the connection to the Slave: press the **CONNECT** button on the top of the page.



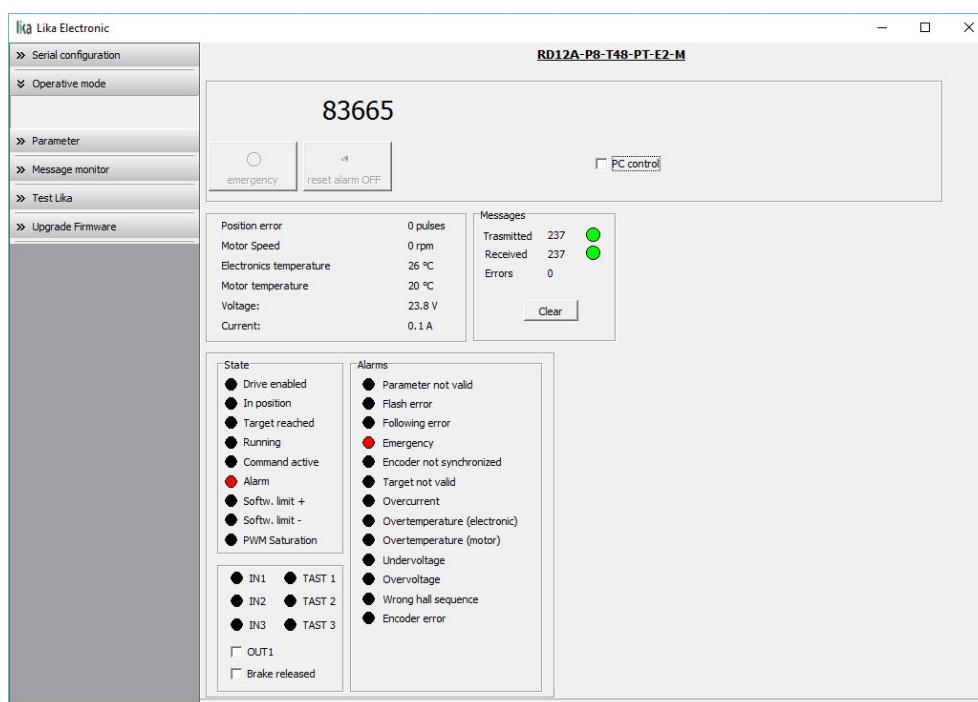
If the connection is established properly, two green lights placed next to the fields used to choose the **serial port** and set the **node ID** come on, while the **CONNECT** button disappears and is replaced by the **DISCONNECT** button. Furthermore the hardware version and the software version as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

The green light next to the **COM** item indicates that the COM port is open successfully.

The green light next to the **Node ID** item indicates that the DRIVECOD unit has been detected and the communication has been established successfully.

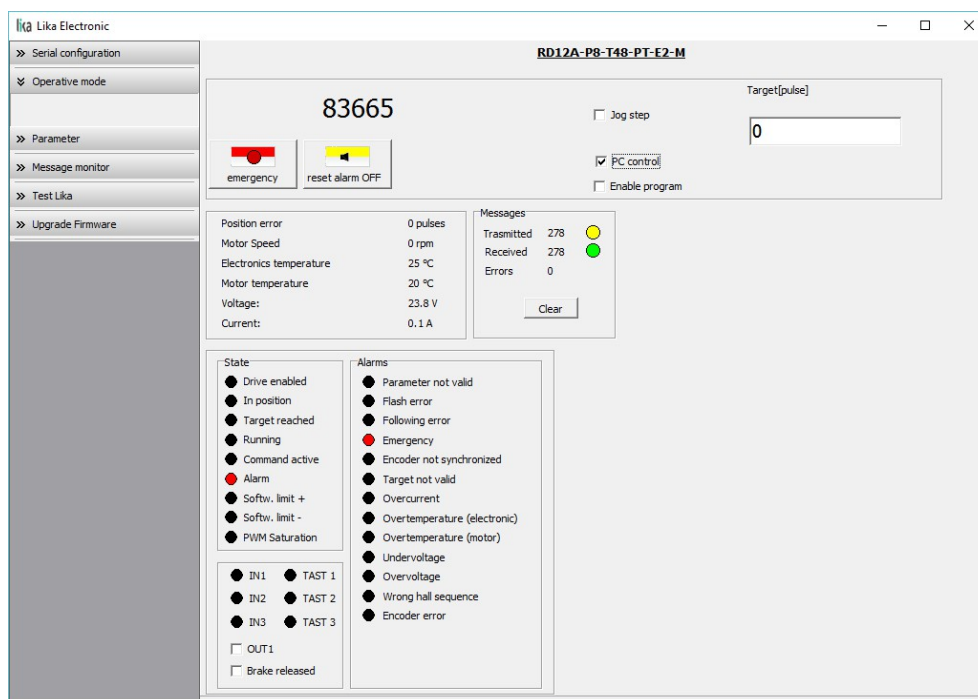
### 8.3 “Operative mode” page

Press the **OPERATIVE MODE** button in the sidebar menu 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 Profinet 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 143).

The following page will appear:

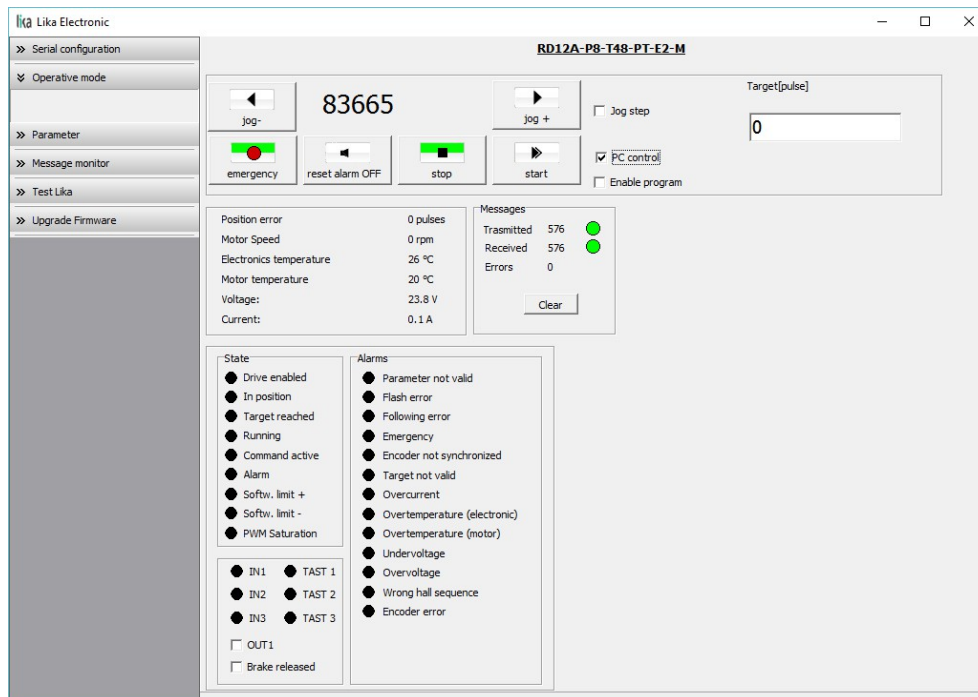


All commands become immediately available for use.

When you first enter the **Operative mode** page, for safety reasons the RD1xA unit is necessarily in an emergency condition: therefore the **Emergency** button is highlighted in red; furthermore the **Emergency** warning in the bottom left-hand **Alarms** box is lit red while the **Alarm** warning in the bottom left-hand **State** box flashes. To restore the **Idle** state of the device, press the **EMERGENCY** button first and then press the **RESET ALARM** button in this page. Alarm warnings will be reset.



The following page will appear:



In the top left-hand **RD1A** box the following functions are available.

### Jog -

See the **Jog -** item on page 144.

### Pos [pulse]

See the **Current position [0x02-0x03]** item on page 155.

### Jog +

See the **Jog +** item on page 143.

### 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 the **Emergency** item on page 145.

### Reset alarm

If an alarm is active, the **RESET ALARM** button is highlighted in yellow; press the button to reset the alarm. See the **Alarm reset** item on page 144.

### Stop

Press this button to force a normal halt of the device, respecting the acceleration and deceleration values. See the **Stop** item on page 144.

### Start

Pressing the button causes the unit to start running in order to reach the position set next to the **Target [pulse]** item. 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 the **Start** item on page 145.

### Release torque

See the **Release axis torque** item on page 147.

### Jog step

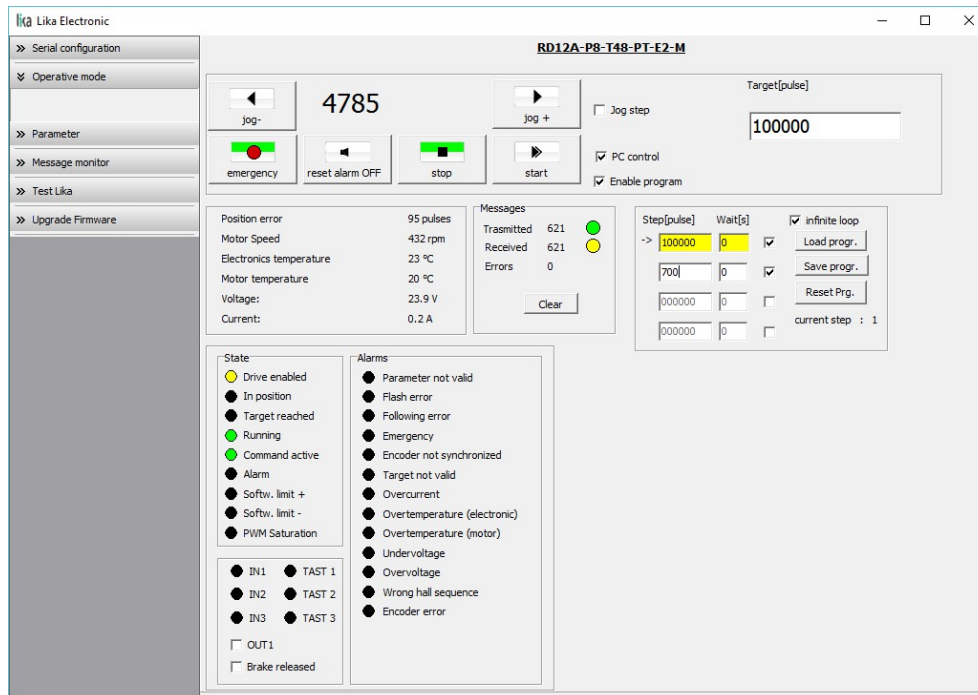
See the **Incremental jog** item on page 145.

### Target [pulse]

See the **Target position [0x2B-0x2C]** item on page 147. 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.

## Enable program

The **ENABLE PROGRAM** check box is used to enable the functions of the **Program** box. The **Program** box does not appear if the **ENABLE PROGRAM** check box is not selected.



The functions available in the **Program** box allow the operator to create and then save work programs for the RD1xA unit.

The positions that 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 interval between one step (commanded movement) and the next. All set values must 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.

The **INFINITE LOOP** check box 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 interval 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 interval to expire; you must then press the **START** button again to command the unit to reach the second position; and so on.

The movement (step) that the actuator is currently performing is shown next to the **CURRENT STEP** item. The interval between a movement and the following one is shown next to the **WAIT** item.

It is possible to save a work program you created. To do so press the **SAVE PROGR.** button. Once you press the button the **Save as** dialog 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 dialog 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** dialog 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 dialog 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 PROGRAM** check box.

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

#### Position error [pulses]

See the [Position following error \[0x05–0x06\]](#) item on page 155.

#### Motor Speed [rpm]

See the [Current velocity \[0x04\]](#) item on page 155.

### Electronics temperature [°C]

See the [Temperature value \[0x07\]](#) item on page 156.

### Motor temperature [°C]

See the [Temperature value \[0x07\]](#) item on page 156.

### Voltage [V]

See the [Motor voltage \[0x0A\]](#) item on page 157.

### Current [A]

It shows the value of the current absorbed by the motor (rated current). The value is expressed in amperes (A). See the [Current value \[0x0B\]](#) item on page 157.

The right-hand **Messages** box allows the operator to have a brief description of the communication between the Master and the Slave, by displaying the Request PDU (Transmitted) and the Response PDU (Received) messages. The fields in the box are meant to show the number of transmitted messages, the number of received messages and the errors: **Transmitted** = Request PDUs; **Received** = Response PDUs; **Errors** = Exception Response PDUs. For complete information on the communication between the Master and the Slave, please refer to the **Message monitor** page (see the "8.5 "Message monitor" page" section on page 116).

In the bottom left-hand **States** and **Alarms** boxes 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 the [Status word \[0x01\]](#) item on page 152; for a detailed description of the alarms see the [Alarms register \[0x00\]](#) item on page 150.

In the bottom left-hand box the current state of the three inputs and the three buttons is shown. The lights are ON when either the relevant input is high (active) or the relevant button is pressed (forced high). For complete information on the digital inputs refer to the **IN 1** item on page 154 and ff. For complete information on the buttons refer to the **Button 1 Jog +** item on page 154 and ff.

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 147.

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. When you select the **BRAKE RELEASED** check box, the brake is deactivated so, for instance, it is possible to move manually the shaft of the DRIVECOD unit.

## 8.4 "Parameter" page

By pressing the **PARAMETER** button in the sidebar menu the operator enters the **Parameter** page.

Parameter	Current Value	Min	Max
Distance/rev [pulses/rev]:	1024	1	1024
Positive delta[pulses]:	523263	0	523263
Negative delta[pulses]:	523263	0	523263
Preset [pulses]:	0	-1048576	1048576
Offset[pulses]:	897264		
Max follow error [pulses]:	1024	0	65535
Position window[pulses]:	1	0	65535
Pos. window time[ms]:	0	0	10000
Jog speed motor [rpm]:	2000	1	3000
Motor work speed [rpm]:	2000	1	3000
Acceleration [rev/s²]:	10	1	500
Deceleration [rev/s²]:	10	1	500
Code sequence:	0	[0 (CW) ▼]	
Kp (position loop):	300	0	1000
Ki (position loop):	10	0	1000
SW LS+	523263		
SW LS-	-523263		
Jog step length[pulses]:	1000	1	10000

Buttons: Save Parameter, Load Default

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 the minimum and maximum values allowed are shown on the right. For detailed information on the function and the setting of each parameter refer to the "8.12.1 Holding Register parameters" section on page 135.

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 that is 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 before saving the values all data not saved will be lost! For any further information on saving the parameters refer to the **Save parameters** variable on page 146.

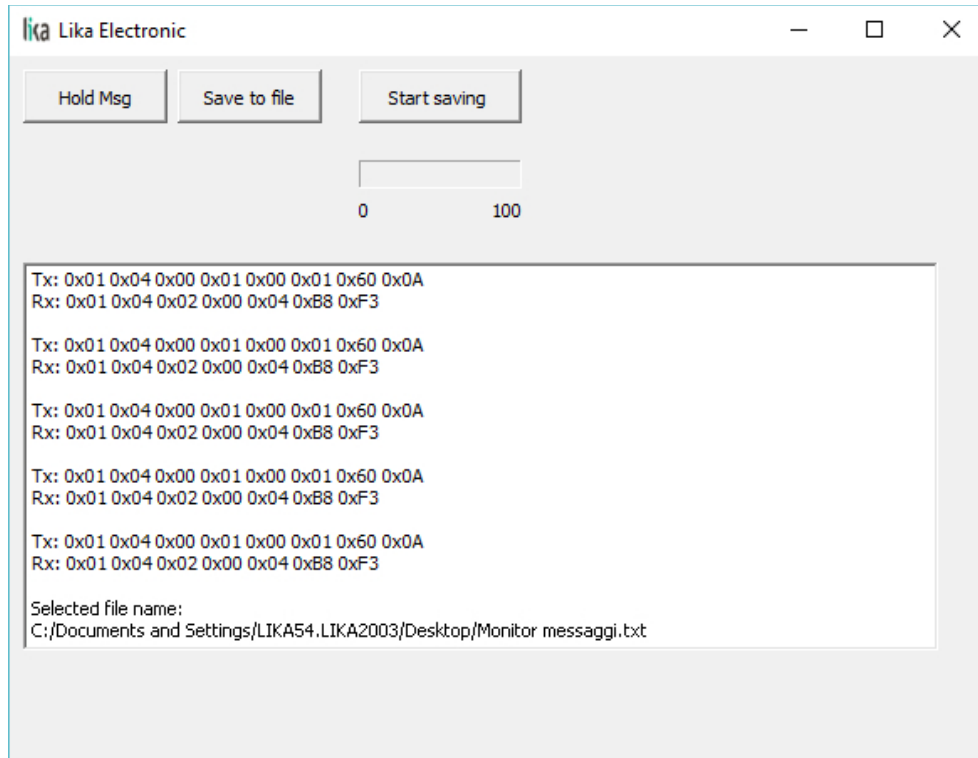
When you need to load the default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) press the **LOAD DEFAULT** button. For any further information on loading the default parameters refer to the **Load default parameters** variable on page 146. The complete list of the machine data parameters and the relevant default values as set by Lika Electronic are available on page 167.

#### **SW LS + / SW LS -**

They are available in the box over the **SAVE PARAMETER** and **LOAD DEFAULT** buttons. They show visually the positive and negative limit switch values. See the **Positive delta [0x09-0x0A]** item on page 138 and the **Negative delta [0x0B-0x0C]** item on page 139.

## 8.5 “Message monitor” page

By pressing the **MESSAGE MONITOR** button in the sidebar menu 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.

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. The messages are displayed in hexadecimal notation. After pressing the **VIEW MSG** button, its descriptive label is replaced by the **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** dialog 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 dialog 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.



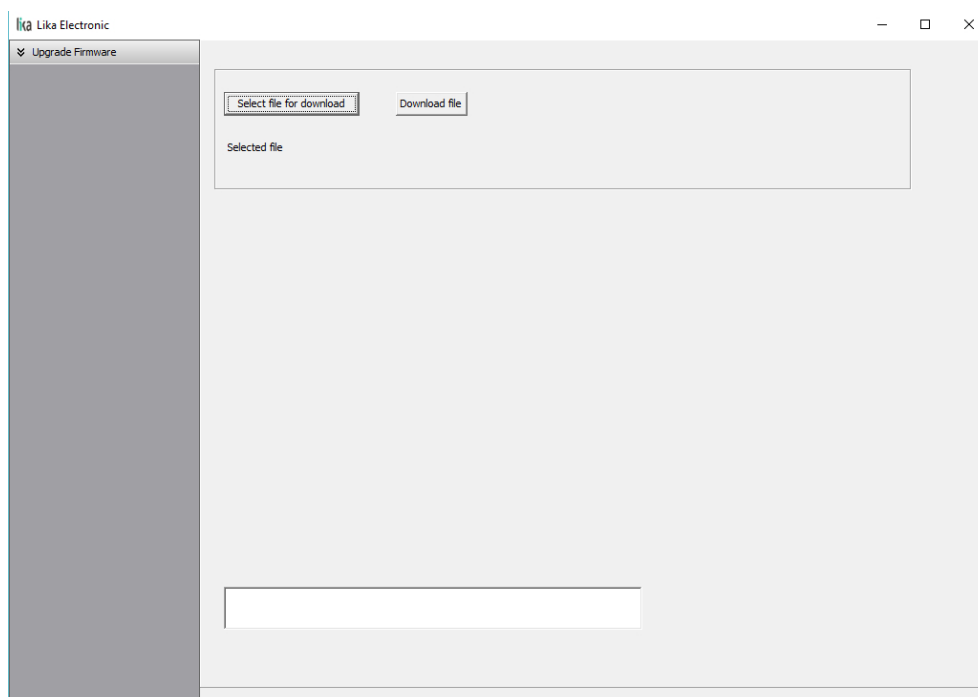
A box in the **Parameter** page offers a brief description of the communication between the Master and the Slave, by displaying the Request PDU (Transmitted) and the Response PDU (Received) messages. The fields in the box are meant to show the number of transmitted messages, the number of received messages and the errors: **Transmitted** = Request PDUs; **Received** = Response PDUs; **Errors** = Exception Response PDUs.

## 8.6 “Test Lika” page

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

## 8.7 “Upgrade Firmware” page

By pressing the **UPGRADE FIRMWARE** button in the sidebar menu the operator enters the **Upgrade Firmware** page.



The functions available in this page allow the operator to upgrade the DRIVECOD unit firmware by downloading upgrading data to the flash memory.

The firmware is a software program which controls the functions and operation of a device; the firmware program, sometimes referred to as "user program", is stored in the flash memory integrated inside the DRIVECOD unit. DRIVECOD units are designed so that the firmware can be easily updated by the user himself. This allows Lika Electronic to make new improved firmware programs available during the lifetime of the product.

Typical reasons for the release of new firmware programs are the necessity to make corrections, improve and even add new functionalities to the device.

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



#### WARNING

The firmware upgrading process in any DRIVECOD unit has to be accomplished by skilled and competent personnel. If the upgrade is not performed according to the instructions provided or a wrong or incompatible firmware program is installed, then the unit may not be updated correctly, in some cases preventing the DRIVECOD unit from working.

If the latest firmware version is already installed in the DRIVECOD unit, you do not need to proceed with any new firmware installation. Current firmware version can be verified from the **SW VERSION** item in the **Slave settings** box of the **Serial configuration** page after connecting properly to the unit (see on page 105).

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

To upgrade the firmware program please proceed as follows:

1. make sure that the following configuration parameters are set (they are unmodifiable in the serial port of the DRIVECOD unit): baud rate = 9600 bits/s; parity = even; if they are set otherwise in your PC, please set them; see the "4.2.6 Inputs / output + MODBUS RS-232 service port" section on page 35;
2. make sure that the DRIVECOD unit you need to update is the only node connected to the personal computer;
3. connect to the unit, go online and then enter the **Upgrade Firmware** page;
4. when you switch on the power supply, if user program is not present in the flash memory, you are not able to connect to the unit through the **Serial configuration** page; when this happens you need to enter directly the **Upgrade Firmware** page; make sure that the correct serial port of

the personal computer connected to the DRIVECOD unit is selected in the **Serial configuration** page; for any further information please refer to the "8.7.1 If an installation issue occurs" section;

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 DRIVECOD model having its own bus interface an appropriate firmware file is available. Make sure that you have the appropriate update for your DRIVECOD model. The .BIN file released by Lika Electronic has a file name that must be interpreted as follows.

For instance: RD1xA\_P8\_Tx\_PT.BIN, where:

- RD1xA = DRIVECOD unit model;
- P8 = power supply
- Tx = reduction gear ratio
- PT = bus interface of the DRIVECOD unit (CB = CANopen; EC = EtherCAT; MB = MODBUS; PB = Profibus; PL = POWERLINK; PT = Profinet).

6. select the .BIN file and confirm the choice by pressing the **OPEN** button, the dialog box closes;
7. the complete path of the file you 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;



#### WARNING

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

10. as soon as the operation is carried out successfully, the **UPGRADE INSTALLATION COMPLETED SUCCESSFULLY** message is displayed;
11. the DRIVECOD unit is now in an emergency condition;
12. close and then restart the SW\_RD1xA\_LKC742\_Vx.EXE program; connect to the DRIVECOD unit and restore the normal work condition through the **Operative mode** page.

#### 8.7.1 If an installation issue occurs

While downloading the firmware upgrading program, unexpected conditions may arise which could lead to a failure of the installation process. When such a

matter occurs, the download process cannot be carried out successfully and thus the operation is aborted.

#### CONDITION 1

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 DRIVECOD 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. At next power-on the unit is out of order because 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\_RD1xA\_LKC742\_Vx.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 6. Always make sure that the correct serial port of the personal computer connected to the DRIVECOD unit is selected in the **Serial configuration** page.

#### CONDITION 2

While downloading data to the flash memory for upgrading the firmware of the unit, if data transmission is cut off (for instance, due to the disconnection of the serial cable or because the process is terminated), 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 DRIVECOD unit first, then close and restart the SW\_RD1xA\_LKC742\_Vx.EXE program. At next power-on the unit is out of order because 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 6. Always make sure that the correct serial port of the personal computer connected to the DRIVECOD unit is selected in the **Serial configuration** page.

### 8.8 Modbus Master / Slaves protocol principle

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

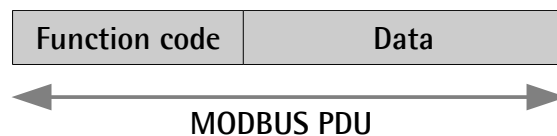
communicate with each other. The Master node initiates only one Modbus transaction at the same time.

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

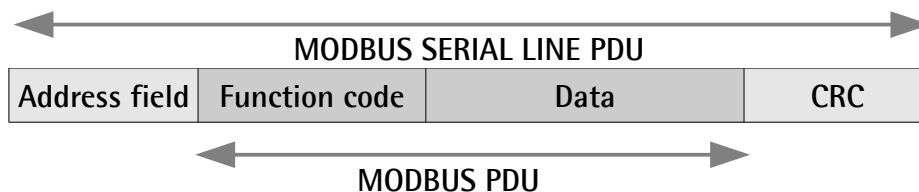
- **UNICAST mode:** in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must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

### 8.9 Modbus frame description

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



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



- **ADDRESS FIELD:** on Modbus Serial Line the address field only contains the Slave address. As previously stated (see the "8.8 Modbus Master / Slaves protocol principle" section on page 120), 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 "8.11 Function codes" section on page 125.
- **DATA:** the **DATA** field of messages contains the bytes for additional information and transmission specifications that the server uses to take the action defined by the **FUNCTION CODE**. This can include items such as discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field. The structure of the **DATA** field depends on each **FUNCTION CODE** (refer to the "8.11 Function codes" section on page 125).
- **CRC (Cyclic Redundancy Check):** error checking field is the result of a "Redundancy Check" calculation that is performed on the message contents. This is intended to check whether transmission has been performed properly. The CRC field is two bytes, containing 16-bit binary value. The CRC value is calculated by the transmitting device, which appends the CRC to the message. The device that receives recalculates a CRC during receipt of the message and compares the calculated value to the actual value it received in the CRC field. If the two values are not equal, an error results.

The Modbus protocol defines three PDUs. They are:

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

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

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

The **Modbus Exception Response PDU** is defined as {exception-function\_code, exception\_code}, where:

exception-function\_code = Modbus function code + 0x80 [1 byte];  
exception\_code = Modbus Exception code, refer to the table "Modbus Exception Codes" in the section 7 of the document "Modbus Application Protocol Specification V1.1b".

### 8.10 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.

#### 8.10.1 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 = 9,600 bit/s the "silent interval" is 4 ms.

When baud rate = 19,200 bit/s the "silent interval" is 2 ms.

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

**Coding system:** 8-bit binary

**Bits per Byte:** 1 start bit;

8 data bits, least significant bit (lsb) sent first;

1 bit for parity completion (= Even);

1 stop bit.

Modbus protocol uses a "big-Endian" representation for addresses and data items. This means that when a numerical quantity greater than a single byte is transmitted, the most significant byte (MSB) is sent first.

Each character or byte is sent in this order (left to right):

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

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

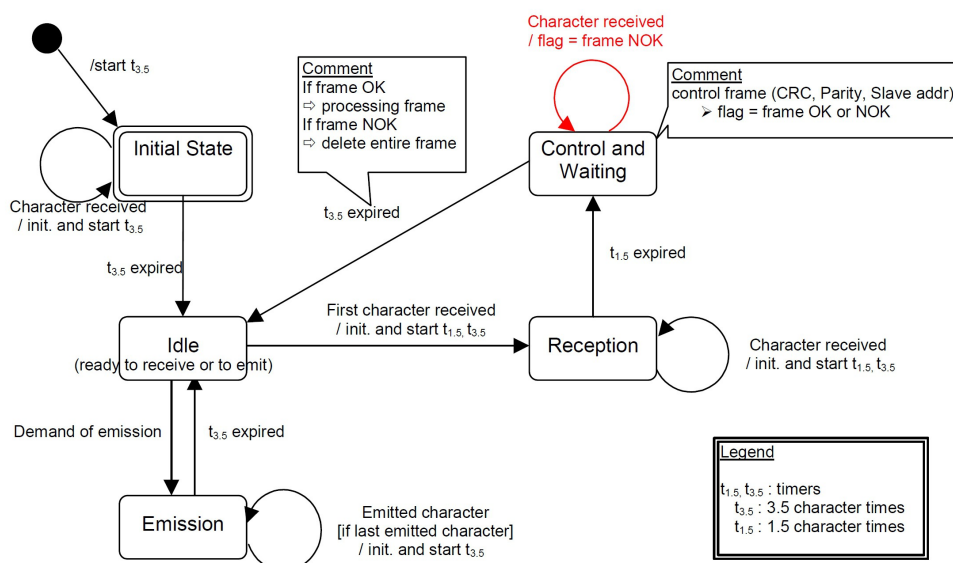
\* When "No parity" is activated, the parity bit is replaced by a stop bit.

The default parity mode must be even parity.

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

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

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



- Transition from **Initial State** to **Idle** state needs an interval of at least 3.5 character times (time-out expiration =  $t_{3.5}$ ).
- **Idle** state is the normal state when neither emission nor reception is active. In RTU mode, the communication link is declared in **Idle** state when there is no transmission activity after a time interval equal to at least 3.5 characters ( $t_{3.5}$ ).



- A request can only be sent in **Idle** state. After sending a request, the Master leaves the **Idle** state and cannot send a second request at the same time.
- When the link is in **Idle** state, each transmitted character detected on the link is identified as the start of the frame. The link goes to **Active** state. Then the end of the frame is identified when no more character is transmitted on the link after the time interval of at least  $t_{3.5}$ .
- After detection of the end of frame, the CRC calculation and checking is completed. Afterwards the address field is analysed to determine if the frame is addressed to the device. If not, the frame is discarded. In order to reduce the reception processing time the address field can be analysed as soon as it is received without waiting the end of frame. In this case the CRC will be calculated and checked only if the frame is actually addressed to the Slave.

### 8.11 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.

#### 8.11.1 Implemented function codes

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

#### 03 Read Holding Registers

FC = 03 (Hex = 0x03) r/o

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

addressed starting at zero. Therefore registers numbered 1-16 are addressed as 0-15.

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

For the complete list of holding registers accessible using **03 Read Holding Registers** function code please refer to the "8.12.1 Holding Register parameters" section on page 135.

### Request PDU

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

### Response PDU

Function code	1 byte	<b>0x03</b>
Byte count	1 byte	2 x N*
Register value	N* x 2 bytes	

\*N = Quantity of registers

### Exception Response PDU

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



Here is an example of a request to read the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9).

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	<b>03</b>	Function	<b>03</b>
Starting address Hi	<b>00</b>	Byte count	<b>04</b>
Starting address Lo	<b>07</b>	Register 8 value Hi	<b>00</b>
No. of registers Hi	<b>00</b>	Register 8 value Lo	<b>0A</b>

No. of registers Lo	02	Register 9 value Hi	00
		Register 9 value Lo	0A

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

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

**Request PDU** (in hexadecimal format)

[01][03][00][07][00][02][75][CA]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)

[00][02] = number of requested registers

[75][CA] = CRC

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

**Response PDU** (in hexadecimal format)

[01][03][04][00][0A][00][0A][5A][36]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

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

[00][0A] = value of register 8 **Acceleration [0x07]**, 00 0A hex = 10 dec

[00][0A] = value of register 9 **Deceleration [0x08]**, 00 0A hex = 10 dec

[5A][36] = CRC

#### 04 Read Input Register

FC = 04 (Hex = 0x04)

This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some results values and state / alarm messages in a remote device. The Request PDU specifies the starting

register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15. The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).

For the complete list of input registers accessible using **04 Read Input Register** function code please refer to the "8.12.2 Input Register parameters" section on page 150.

### Request PDU

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

### Response PDU

Function code	1 byte	<b>0x04</b>
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	<b>0x84 (=0x04 + 0x80)</b>
Exception code	1 byte	01 or 02 or 03 or 04



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

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	<b>04</b>	Function	<b>04</b>
Starting address Hi	<b>00</b>	Byte count	<b>04</b>
Starting address Lo	<b>02</b>	Register 3 value Hi	<b>00</b>

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, the **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 the "8.12.1 Holding Register parameters" section on page 135.

**Request PDU**

Function code	1 byte	<b>0x06</b>
Register address	2 bytes	0x0000 to 0xFFFF
Register value	2 bytes	0x0000 to 0xFFFF

**Response PDU**

Function code	1 byte	<b>0x06</b>
Register address	2 bytes	0x0000 to 0xFFFF
Register value	2 bytes	0x0000 to 0xFFFF

**Exception Response PDU**

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



Here is an example of a request to write the value 00 96 hex (= 150 dec) in the **Acceleration [0x07]** parameter (register 8).

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	<b>06</b>	Function	<b>06</b>

Register address Hi	00	Register address Hi	00
Register address Lo	07	Register address Lo	07
Register value Hi	00	Register value Hi	00
Register value Lo	96	Register value Lo	96

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

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

**Request PDU** (in hexadecimal format)

[01][06][00][07][00][96][B8][65]

where:

[01] = Slave address

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

[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)

[00][96] = value to be set in the register

[B8][65] = CRC

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

**Response PDU** (in hexadecimal format)

[01][06][00][07][00][96][B8][65]

where:

[01] = Slave address

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

[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)

[00][96] = value set in the register

[B8][65] = CRC

## 16 Write Multiple Registers

FC = 16 (Hex = 0x10)

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

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

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

For the complete list of registers accessible using **16 Write Multiple Registers** function code please refer to the "8.12.1 Holding Register parameters" section on page 135.

### Request PDU

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

### Exception Response PDU

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



Here is an example of a request to write the values 150 and 100 dec in the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) respectively.



Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	10	Function	10
Starting address Hi	00	Starting address Hi	00
Starting address Lo	07	Starting address Lo	07
Quantity of registers Hi	00	Quantity of registers Hi	00
Quantity of registers Lo	02	Quantity of registers Lo	02
Byte count	04		
Register 8 value Hi	00		
Register 8 value Lo	96		
Register 9 value Hi	00		
Register 9 value Lo	64		

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

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

**Request PDU** (in hexadecimal format)

[01][10][00][07][00][02][04][00][96][00][64][53][8E]

where:

[01] = Slave address

[10] = **16 Write Multiple Registers** function code

[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)

[00][02] = number of requested registers

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

[00][96] = value to be set in the register 8 **Acceleration [0x07]**, 00 96 hex = 150 dec

[00][64] = value to be set in the register 9 **Deceleration [0x08]**, 00 64 hex = 100 dec

[53][8E] = CRC

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

**Response PDU** (in hexadecimal format)

[01][10][00][07][00][02][F0][09]

where:

[01] = Slave address

[10] = **16 Write Multiple Registers** function code

[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)

[00][02] = number of written registers

[F0][09] = CRC



#### NOTE

For further examples refer to the "8.14 Programming examples" section on page 161.



#### WARNING

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

For this purpose the Watch dog function is implemented and can be activated as optional. The Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running -a jog command for example- the Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered. To enable the Watch dog function, set to "=1" the **Watch dog enable** bit in the **Control Word [0x2A]** variable. If "=0" is set the Watch dog is not enabled; if "=1" is set the Watch dog is enabled. When the Watch dog function is enabled, if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm message is invoked to appear as soon as the Modbus network communication is restored).

## 8.12 Programming parameters

Hereafter the parameters available for RD1xA Modbus devices are listed and described as follows:

### Parameter name [Register address]

[Register number, data types, attribute]

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

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

### Unsigned16 parameter structure:

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

### Unsigned32 parameter structure:

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

### 8.12.1 Holding Register parameters

**Holding registers (Machine data parameters)** are accessible for both writing and reading; to read the value set in a parameter use the **03 Read Holding Registers** function code (reading of multiple registers); to write a value in a parameter use the **06 Write Single Register** function code (writing of a single register) or the **16 Write Multiple Registers** (writing of multiple registers); for any further information on the implemented function codes refer to the "8.11.1 Implemented function codes" section on page 125.



#### NOTE

Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the **Save parameters** function available in the **Control Word [0x2A]** register, see on page 143.

Should the power supply be turned off all data that has not been saved previously will be lost!



#### WARNING

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

#### Distance per revolution [0x00]

[Register 1, Unsigned16, rw]

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

Default = 1024 (min. = 1, max. = 1024)



#### WARNING

After having changed this parameter you must then set new values also next to the **Preset [0x12-0x13]** parameter. For a detailed explanation see on page 47 and relevant parameters.

Please note that the parameters listed hereafter are closely related to the **Distance per revolution [0x00]** parameter; hence when you change the value in **Distance per revolution [0x00]** also the value in each of them necessarily changes. They are: **Position window [0x01]**, **Max following error [0x03-0x04]**, **Positive delta [0x09-0x0A]**, **Negative delta [0x0B-0x0C]**, **Target position [0x2B-0x2C]**, **Current position [0x02-0x03]** and **Position following error [0x05-0x06]**. See also on page 140.



#### NOTE

If **Distance per revolution [0x00]** is not a power of 2 (2, 4, ..., 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 limits for the **Target position [0x2B-0x2C]** value. As soon as the axis is within the tolerance window limits, the bit 8 **Target position reached** in the **Status word [0x01]** goes high ("=1"). When the axis is within the tolerance window limits for the time set in the **Position window time [0x02]** parameter, the bit 0 **Axis in position** in the **Status word [0x01]** goes high ("=1"). The parameter is expressed in pulses. See also the "Positioning: position and speed control" section on page 45.

Default = 1 (min. = 0, max. = 65535)

**Position window time [0x02]**

[Register 3, Unsigned16, rw]

It represents the time for which the axis has to be within the tolerance window limits set in the **Position window [0x01]** parameter before the state is signalled through the **Axis in position** status bit of the **Status word [0x01]**. The parameter is expressed in milliseconds (ms). See also the "Positioning: position and speed control" section on page 45.

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

**Max following error [0x03-0x04]**

[Registers 4-5, Unsigned32, rw]

This parameter defines the maximum allowable difference between the real position and the theoretical position of the device. If the device detects a value higher than the one set in this parameter, the **Following error** alarm is triggered and the unit stops. The parameter is expressed in pulses.

Default = 1024 (min. = 0, max. = 65535)

**Kp position loop [0x05]**

[Register 6, Unsigned16, rw]

This parameter contains the proportional gain used by the PI controller for the position loop. The value has been optimized by Lika Electronic according to the technical characteristics of the device.

Default = 300 (min. = 0, max. = 1000)

**Ki position loop [0x06]**

[Register 7, Unsigned16, rw]

This parameter contains the integral gain used by the PI controller for the position loop. The value has been optimized by Lika Electronic according to the technical characteristics of the device.

Default = 10 (min. = 0, max. = 1000)

### Acceleration [0x07]

[Register 8, Unsigned16, rw]

This parameter defines the acceleration value that has to be used by the motor when reaching both the **Jog speed [0x0D]** and the **Work speed [0x0E]**. The parameter is expressed in revolutions per second<sup>2</sup> [rev/s<sup>2</sup>]. See also the "6.2 Movements: jog and positioning" section on page 44.

Default = 10 (min. = 1, max. = 500)

### Deceleration [0x08]

[Register 9, Unsigned16, rw]

This parameter defines the deceleration value that has to be used by the motor when stopping. The parameter is expressed in revolutions per second<sup>2</sup> [rev/s<sup>2</sup>]. See also the "6.2 Movements: jog and positioning" section on page 44.

Default = 10 (min. = 1, max. = 500)

### Positive delta [0x09–0x0A]

[Registers 10–11, Unsigned32, rw]

This value is used to calculate the maximum forward (positive) limit the device is allowed to reach starting from the preset value. Should it happen that the maximum forward limit is reached, a signalling is activated through the **SW limit switch +** status bit of the **Status word [0x01]** (the bit is forced high). The parameter is expressed in pulses.

**SW limit switch +** = **Preset [0x12–0x13]** + **Positive delta [0x09–0x0A]**.

For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.

Default = 523 263 (min. = 0, max. = 523 263)



### WARNING

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



### EXAMPLE

When **Distance per revolution [0x00]** = 1,024 and **Preset [0x12–0x13]** = 0, the maximum acceptable value for **Positive delta [0x09–0x0A]** is:

(1,024 steps per revolution \* 512 revolutions) – 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 523 263

When **Distance per revolution [0x00]** = 256 and **Preset [0x12–0x13]** = 0, the maximum acceptable value for **Positive delta [0x09–0x0A]** is:

(256 steps per revolution \* 512 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 130 815

See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.


**WARNING**

Every time **Distance per revolution [0x00]** and **Preset [0x12-0x13]** parameters are changed, **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** values have to be checked carefully. Each time you change the value in **Distance per revolution [0x00]** you must then update the value in **Preset [0x12-0x13]** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 47.

**Negative delta [0x0B-0x0C]**

[Registers 12-13, Unsigned32, rw]

This value is used to calculate the maximum backward (negative) limit the device is allowed to reach starting from the preset value. Should it happen that the maximum backward limit is reached, a signalling is activated through the **SW limit switch** – status bit of the **Status word [0x01]** (the bit is forced high). The parameter is expressed in pulses.

**SW limit switch** – = **Preset [0x12-0x13]** – **Negative delta [0x0B-0x0C]**.

For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.

Default = 523 263 (min. = 0, max. = 523 263)


**WARNING**

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


**EXAMPLE**

When **Distance per revolution [0x00]** = 1,024 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Negative delta [0x0B-0x0C]** is:

(1,024 steps per revolution \* 512 revolutions) – 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 523 263

When **Distance per revolution [0x00]** = 256 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Negative delta [0x0B-0x0C]** is:

(256 steps per revolution \* 512 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 130 815

See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 47.



#### WARNING

Every time **Distance per revolution [0x00]** and **Preset [0x12-0x13]** parameters are changed, **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** values have to be checked carefully. Each time you change the value in **Distance per revolution [0x00]** you must then update the value in **Preset [0x12-0x13]** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 47.

#### Jog speed [0x0D]

[Register 14, Unsigned16, rw]

This parameter contains the maximum speed the motor is allowed to reach when using the **Jog +** and **Jog -** functions (see the **Control Word [0x2A]** parameter). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 44.

Default = 2000 (min. = 1, max. = 3000)



#### NOTE

Please note that this is the speed of the motor, not the speed of the output shaft after the reduction gears.

The speed at output will be as follows:

Motor speed = 2000 rpm

Speed at output:

T12	= 166 rpm
T24	= 83 rpm
T48	= 41 rpm
T92	= 21 rpm

#### Work speed [0x0E]

[Register 15, Unsigned16, rw]

This parameter contains the maximum speed the motor is allowed to reach in automatic work mode (movements are controlled using the **Start** and **Stop**



commands -see the **Control Word [0x2A]** parameter- and are performed in order to reach the position set in **Target position [0x2B-0x2C]**. The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 45.

Default = 2000 (min. = 1, max. = 3000)



#### NOTE

Please note that this is the speed of the motor, not the speed of the output shaft after the reduction gears.

The speed at output will be as follows:

Motor speed = 2000 rpm

Speed at output: T12 = 166 rpm

T24 = 83 rpm

T48 = 41 rpm

T92 = 21 rpm

#### Code sequence [0x0F]

[Register 16, Unsigned16, rw]

It sets whether the position value output by the device increases (count up information) when the shaft rotates clockwise (0) or counter-clockwise (1). Clockwise and counter-clockwise rotations are viewed from shaft side.

0 = count up information with clockwise rotation (default)

1 = count up information with counter-clockwise rotation



#### WARNING

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

#### Offset [0x10-0x11]

[Registers 17-18, Integer32, ro]

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

Default = 0

**Preset [0x12-0x13]**

[Registers 19-20, Integer32, rw]

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

Default = 0 (min. = -1 048 576, max. = +1 048 576)

**NOTE**

We suggest activating the preset when the actuator is in stop. See the **Setting the preset** command on page 146.

**WARNING**

A new value has to be set in the **Preset [0x12-0x13]** registers every time the **Distance per revolution [0x00]** value is changed. After having entered a new value in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in the **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 47.

**Jog step length [0x14]**

[Register 21, Unsigned16, rw]

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

Default = 1000 (min. = 1, max. = 10000).

**Extra commands register [0x29]**

[Register 42, Unsigned16, rw]

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

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

**Byte 0**

bit 0: Not used.

**Control by PC**

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

bits 2 ... 7 Not used.

**Byte 1** Not used.**WARNING**

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

**Control Word [0x2A]**

[Register 43, Unsigned16, rw]

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

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

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

**Byte 0****Jog +**

bit 0 If the bit 4 **Incremental jog** = 0, as long as **Jog +** = 1, the Slave moves toward the positive direction; otherwise if the bit 4 **Incremental jog** = 1, the activation of this bit causes a single step toward the positive direction having the length, expressed in pulses, set next to the **Jog step length [0x14]** item to be executed at rising edge; then the actuator stops

and waits for another command. Velocity, acceleration and deceleration are performed according to the values set next to the **Jog speed [0x0D]**, **Acceleration [0x07]** and **Deceleration [0x08]** parameters respectively. For a detailed description of the jog control see on page 44.



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

#### Jog - bit 1

If the bit 4 **Incremental jog** = 0, as long as **Jog -** = 1, the Slave moves toward the negative direction; otherwise if the bit 4 **Incremental jog** = 1, the activation of this bit causes a single step toward the negative direction having the length, expressed in pulses, set next to the **Jog step length [0x14]** item to be executed at rising edge; then the actuator stops and waits for another command. Velocity, acceleration and deceleration are performed according to the value set next to the **Jog speed [0x0D]**, **Acceleration [0x07]** and **Deceleration [0x08]** parameters respectively. For a detailed description of the jog control see on page 44.



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

#### Stop bit 2

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

#### Alarm reset bit 3

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



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

### Incremental jog

bit 4

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

bit 5

Not used.

### Start

bit 6

When it is set to "1" the device moves in order to reach the set target position (see **Target position [0x2B-0x2C]** on page 147). For a complete description of the position control see on page 45.



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

### Emergency

bit 7

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

### Byte 1

#### Watch dog enable

bit 8

Setting the **Watch dog enable** bit to "1" causes the Watch dog function to be enabled; setting the **Watch dog enable** bit to "0" causes the Watch dog function to be disabled. When the Watch dog function is enabled, if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm is invoked to appear as soon as the Modbus

network communication is restored). The Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running -a jog command for example- the Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered.

### Save parameters

bit 9



Data is saved on non-volatile memory at each rising edge of the bit; in other words, save is performed each time this bit is switched from logic level low ("0") to logic level high ("1"). Always save the new values after setting in order to store them in the non-volatile memory permanently. Should the power supply be turned off all data that has not been saved previously will be lost!

### Load default parameters

bit 10



The default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) are restored at each rising edge of the bit; in other words, the default parameters loading operation is performed each time this bit is switched from logic level low ("0") to logic level high ("1"). The complete list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 167.

Always save the new values after setting in order to store them in the non-volatile memory permanently. Should the power supply be turned off all data that has not been saved previously will be lost!



#### WARNING

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

### Setting the preset

bit 11

It sets the current position to the value set next to the **Preset [0x12-0x13]** registers. The operation is performed at each rising edge of the bit, i.e. each time this bit is

switched from logic level low ("0") to logic level high ("1"). We suggest activating the preset when the actuator is in stop. For more information refer to page 142.

#### Release axis torque

bit 12 When the axis has reached the commanded position, it maintains the torque.  
If set to "=0", when the axis is in position, the PWM is kept active.  
If set to "=1", when the axis is in position, the PWM is deactivated (the torque is released).

#### OUT 1

bit 13 This is intended to activate / deactivate the operation of the digital output 1. The meaning of the available output is described in the "6.3 Digital inputs and output" section on page 46.  
**OUT 1 = 0** output 1 low (not active)  
**OUT 1 = 1** output 1 high (active)

#### Brake disabled

bit 14 This function is available only in the RD12A version (model fitted with brake); in the RD1A version (model without brake) the 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. Setting this bit to "=1" causes the brake to be disabled and not operational; 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.



#### WARNING

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

#### Target position [0x2B-0x2C]

[Registers 44-45, Integer32, rw]

It sets the position to be reached, otherwise referred to as commanded position. When the **Start** command is sent while the **Stop** and **Emergency** bits are "=1"

and the alarm condition is off, the device moves in order to reach the target position set next to this item.

As soon as the axis is within the tolerance window limits set next to the **Position window [0x01]** register, the bit 8 **Target position reached** in the **Status word [0x01]** goes high ("=1"). When the position is within the tolerance window limits set next to the **Position window [0x01]** register, after the delay set next to the **Position window time [0x02]** item, the bit 0 **Axis in position** in the **Status word [0x01]** goes high ("=1").

For more information refer also to the "Positioning: position and speed control" section on page 45.

Default = 0 (min. = 0, max. = within maximum positive limit / maximum negative limit)



#### NOTE

##### Position override function

It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the **Target position [0x2B-0x2C]** registers. See also on page 45.



#### NOTE

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

When the Watch dog function is enabled (**Watch dog enable** in **Control Word [0x2A]** is set to "=1"), should the device be disconnected from the Modbus network while it is moving (for instance because of a broken cable or a faulty wiring), the device stops moving immediately and activates the **Watch dog** alarm bit (the alarm is invoked to appear as soon as the Modbus network communication is restored).



#### WARNING

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



**NOTE**

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

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

### 8.12.2 Input Register parameters

The **Input Register** parameters are accessible for reading only; to read the value set in an input register parameter use the **04 Read Input Register** function code (reading of multiple input registers); for any further information on the implemented function codes refer to the "8.11.1 Implemented function codes" section on page 125.

#### Alarms register [0x00]

[Register 1, Unsigned16, ro]

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

Structure of the alarms byte:

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

The available alarm error codes are listed hereafter:

#### Byte 0

##### Machine data not valid

bit 0 One or more parameters are not valid, set proper values to restore the normal work condition. See the list of the wrong parameters in the [Wrong parameters list \[0x08-0x09\]](#) item.

##### Flash memory error

bit 1 Internal error, it cannot be restored.

##### Counting error

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

##### Following error

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

##### Axis not synchronized

bit 4 Internal error, it cannot be restored.

**Target not valid**

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

**Emergency**

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

**Overcurrent**

bit 7                      Motor overcurrent.

**Byte 1**

**Electronics Overtemperature**

bit 8                      The temperature of the MOSFETs detected by an internal probe is exceeding the maximum ratings (see **Temperature value [0x07]** on page 156). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the range.

**Motor Overtemperature**

bit 9                      The temperature of the motor detected by an internal probe is exceeding the maximum ratings (see **Temperature value [0x07]** on page 156). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the range.

**Undervoltage**

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

**Watch dog**

bit 11                     When the Watch dog function is enabled (bit 8 **Watch dog enable** in **Control Word [0x2A]** is set to "1"), if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm bit is activated). The alarm is invoked to appear as soon as the Modbus network communication is restored. The Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running –a jog command for example– the Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered.

bits 12 and 13           Not used.

**Hall sequence**

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

**Overvoltage**

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

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



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

**Status word [0x01]**

[Register 2, Unsigned16, ro]

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

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

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

**Byte 0****Axis in position**

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

further information please refer to the "Positioning: position and speed control" section on page 45.

bit 1

Not used.

### Drive enabled

bit 2

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

### SW limit switch +

bit 3

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

### SW limit switch -

bit 4

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

### Alarm

bit 5

The value is "1" when an alarm occurs, see details in the [Alarms register \[0x00\]](#) variable.

### Axis running

bit 6

The value is "0" when the device is not moving.  
The value is "1" while the device is moving.

### Executing a command

bit 7

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

### Byte 1

#### Target position reached

bit 8

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

refer also to the "Positioning: position and speed control" section on page 45.

### **Button 1 Jog +** bit 9

RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. As long as the button 1 JOG + is kept pressed, the bit 9 is forced high "=1"; when the button 1 is not pressed, the bit 9 is low "=0". For further information see the "4.4 Screw plug for internal access (Figure 4 and Figure 7)" section on page 39 and the "4.5.1 JOG + and JOG – buttons (Figure 8)" section on page 40.

### **Button 2 Jog –** bit 10

RD1xA positioning unit is equipped with three buttons located inside the housing and accessible by removing a screw plug. As long as the button 2 JOG – is kept pressed, the bit 10 is forced high "=1"; when the button 2 is not pressed, the bit 10 is low "=0". For further information see the "4.4 Screw plug for internal access (Figure 4 and Figure 7)" section on page 39 and the "4.5.1 JOG + and JOG – buttons (Figure 8)" section on page 40.

### **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, the bit 11 is forced high "=1"; when the button 3 is not pressed, the bit 11 is low "=0". For further information see the "4.4 Screw plug for internal access (Figure 4 and Figure 7)" section on page 39 and the "4.5.2 PRESET button (Figure 8)" section on page 41.

### **PWM saturation** bit 12

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

### **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

the "6.3 Digital inputs and output" section on page 46.

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 the "6.3 Digital inputs and output" section on page 46.

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 the "6.3 Digital inputs and output" section on page 46.

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 expressed in pulses.

## Current velocity [0x04]

[Register 5, Integer16, ro]

Speed of the device expressed in revolutions per minute [rpm], updated at every second. This is the speed of the motor, not the speed of the output shaft after the reduction gears. The speed at output will be as follows:

Motor speed = 2000 rpm

Speed at output:      T12 = 166 rpm

                                 T24 = 83 rpm

                                 T48 = 41 rpm

                                 T92 = 21 rpm

## Position following error [0x05-0x06]

[Registers 6-7, Integer32, ro]

This variable contains the difference between the target position and the current position step by step. If this value is greater than the one set in the **Max**

following error [0x03–0x04] parameter, then the **Following error** alarm is triggered and the unit stops. The value is expressed in pulses.

#### Temperature value [0x07]

[Register 8, Integer16, ro]

This variable shows both the temperature of the motor and the temperature of the electronics as detected by internal probes. The value is expressed in Celsius degrees (°C). The minimum detectable temperature is -20°C.

The meaning of the 16 bits in the register is as follows:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MSB								LSB							
Major number								Minor number							
Temperature of the motor								Temperature of the electronics							

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

#### Wrong parameters list [0x08–0x09]

[Registers 9–10, Unsigned32, ro]

The operator has set invalid data and the **Machine data not valid** alarm has been triggered. This variable is meant to show the list of the wrong parameters, respecting the structure shown in the following table.

Please note that the normal work status can be restored only after having set proper values.

Bit	Parameter
0	Not used
1	Distance per revolution [0x00]
2	Acceleration [0x07]
3	Deceleration [0x08]
4	Positive delta [0x09–0x0A]
5	Negative delta [0x0B–0x0C]
6	Jog speed [0x0D]
7	Work speed [0x0E]
8	Code sequence [0x0F]
9	Preset [0x12–0x13]



10	Jog step length [0x14]
11	Kp position loop [0x05]
12	Ki position loop [0x06]
13	Position window time [0x02]
14	Max following error [0x03-0x04]
15	Not used

#### Motor voltage [0x0A]

[Register 11, Unsigned16, ro]

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

#### Current value [0x0B]

[Register 12, Unsigned16, ro]

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

#### Hall [0x0C]

[Register 13, Unsigned16, ro]

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

#### Duty cycle [0x0D]

[Register 14, Unsigned16, ro]

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

#### DIP switch baud rate [0x0E]

[Register 15, Unsigned16, ro]

This is meant to show the data transmission rate (baud rate) of the serial port the RD1xA unit is equipped with; the data transmission rate has to be set through the provided DIP switch. In this model the baud rate DIP switch has fixed value and is not accessible to the user.

#### DIP switch node ID [0x0F]

[Register 16, Unsigned16, ro]

This is meant to show the node address set in the RD1xA unit; the node address has to be set through the provided DIP switch. In this model the node ID DIP switch has fixed value and is not accessible to the user.

### SW Version [0x10]

[Register 17, Unsigned16, ro]

This is meant to show the software version of the DRIVECOD unit.

The meaning of the 16 bits in the register is as follows:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MSB								LSB							
Major number								Minor number							

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

### HW Version [0x11]

[Register 18, Unsigned16, ro]

This is meant to show the hardware version and model of the DRIVECOD unit.

The meaning of the 16 bits in the register is as follows:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DRIVECOD model								Brake	Gear ratio		Hardware version				

where:

00 ... 03	= hardware version
04 ... 06	= gear ratio: 1 = T12; 2 = T24; 3 = T48; 4 = T92
07	= brake: 0 = RD1A model without brake; 1 = RD12A model with brake
08 .. 15	= RD1xA model equipped with the following interface: 0x30 = Modbus RTU; 0x31 = Profibus; 0x32 = CANopen; 0x33 = POWERLINK; 0x34 = EtherCAT; 0x35 = Modbus TCP; 0x36 = EtherNet/IP; 0x37 = Profinet

Value 37 B2 hex in hexadecimal notation corresponds to the binary representation 0011 0111 1011 0010 and has to be interpreted as follows: RD1xA model with Profinet interface, T48 reduction gear, equipped with brake, hardware version 2.



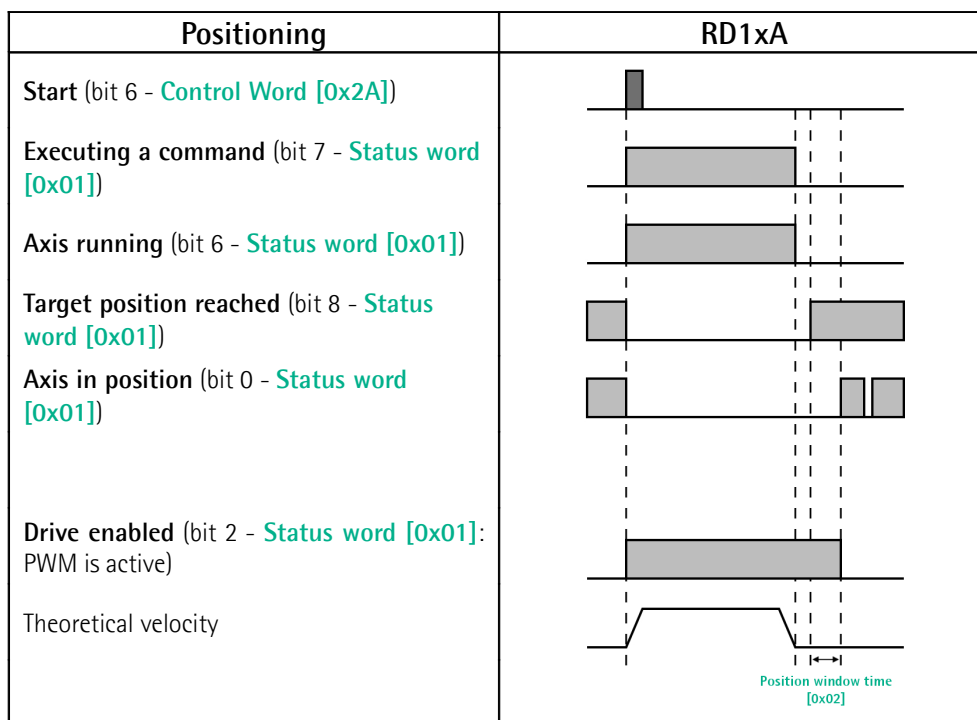
#### NOTE

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

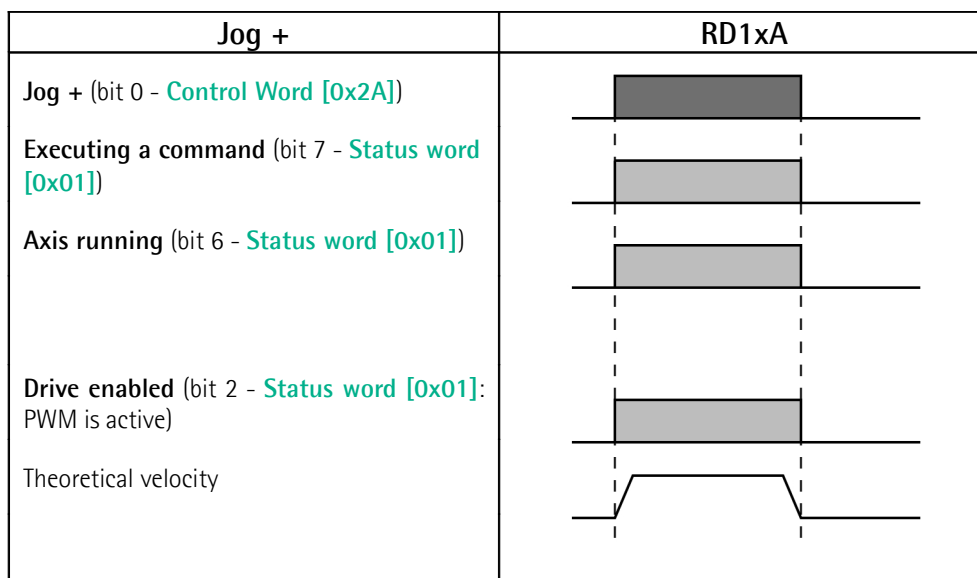
Should the power be turned off all data not saved will be lost!



### EXAMPLE 1



### EXAMPLE 2



### 8.13 Exception codes

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

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

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

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

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

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

## 8.14 Programming examples

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

### 8.14.1 Using the 03 Read Holding Registers function code



#### EXAMPLE 1

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

##### Request PDU

[01][03][00][07][00][02][75][CA]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)

[00][02] = number of requested registers

[75][CA] = CRC

##### Response PDU

[01][03][04][00][0A][00][0A][5A][36]

where:

[01] = Slave address

[03] = **03 Read Holding Registers** function code

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

[00][0A] = value of register 8 **Acceleration [0x07]**, 00 0A hex = 10 dec

[00][0A] = value of register 9 **Deceleration [0x08]**, 00 0A hex = 10 dec

[5A][36] = CRC

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

### 8.14.2 Using the 04 Read Input Register function code



#### EXAMPLE 1

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

##### Request PDU

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

where:

[01] = Slave address

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

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

[00][02] = number of requested registers

[D0][0B] = CRC

##### Response PDU

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

where:

[01] = Slave address

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

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

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

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

[E7][F0] = CRC

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



#### EXAMPLE 2

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

##### Request PDU

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

where:

[01] = Slave address

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

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

[00][01] = number of requested registers

[31][CA] = CRC

#### Response PDU

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

where:

[01] = Slave address

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

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

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

[79][50] = CRC

This means that in the **Alarms register [0x00]** variable (register 1) the bits 0 and 7 are active (logic level high = 1), i.e. (see on page 150): **Machine data not valid** and **Emergency**.

### 8.14.3 Using the 06 Write Single Register function code



#### EXAMPLE 1

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

##### Request PDU

[01][06][00][07][00][96][B8][65]

where:

[01] = Slave address

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

[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)

[00][96] = value to be set in the register

[B8][65] = CRC

##### Response PDU

[01][06][00][07][00][96][B8][65]

where:

[01] = Slave address

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

[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)

[00][96] = value set in the register

[B8][65] = CRC

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



#### EXAMPLE 2

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

##### Request PDU

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

where:

[01] = Slave address

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

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

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

[A8][61] = CRC



### Response PDU

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

where:

[01] = Slave address

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

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

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

[A8][61] = CRC

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



### EXAMPLE 3

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

### Request PDU

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

where:

[01] = Slave address

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

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

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

[AF][02] = CRC

### Response PDU

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

where:

[01] = Slave address

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

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

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

[AF][02] = CRC

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

#### 8.14.4 Using the 16 Write Multiple Registers function code



##### EXAMPLE 1

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

##### Request PDU

[01][10][00][07][00][02][04][00][96][00][64][53][8E]

where:

[01] = Slave address

[10] = **16 Write Multiple Registers** function code

[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)

[00][02] = number of requested registers

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

[00][96] = value to be set in the register 8 **Acceleration [0x07]**, 00 96 hex = 150 dec

[00][64] = value to be set in the register 9 **Deceleration [0x08]**, 00 64 hex = 100 dec

[53][8E] = CRC

##### Response PDU

[01][10][00][07][00][02][F0][09]

where:

[01] = Slave address

[10] = **16 Write Multiple Registers** function code

[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)

[00][02] = number of written registers

[F0][09] = CRC

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

## 9 Default parameters list

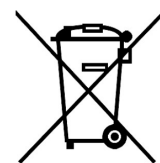


Parameters list	Default value		
Control Word (Bytes 0 and 1)	0		
Target Position (Bytes 2 to 5)	0		
01 Distance per revolution PPR	1,024		
02 Position window P	1		
03 Position window time ms	0		
04 Max following error P	1,024		
05 Proportional gain	300		
06 Integral gain	10		
07 Acceleration $\text{rev/s}^2$	10		
08 Deceleration $\text{rev/s}^2$	10		
09 Positive delta P	523 263		
10 Negative delta P	523 263		
11 Jog speed rpm	2,000		
12 Work speed rpm	2,000		
13 Code sequence	0		
20 Preset P	0		
14 Jog step length P	1,000		



Parameters list	Default value		
Distance per revolution [0x00] PPR	1,024		
Position window [0x01] P	1		
Position window time [0x02] ms	0		
Max following error [0x03-0x04] P	1,024		
Kp position loop [0x05]	300		
Ki position loop [0x06]	10		
Acceleration [0x07] $\text{rev/s}^2$	10		
Deceleration [0x08] $\text{rev/s}^2$	10		
Positive delta [0x09-0x0A] P	523 263		
Negative delta [0x0B-0x0C] P	523 263		
Jog speed [0x0D] rpm	2,000		
Work speed [0x0E] rpm	2,000		
Code sequence [0x0F]	0		
Preset [0x12-0x13] P	0		
Jog step length [0x14] P	1,000		
Control Word [0x2A]	0		
Target position [0x2B-0x2C]	0		

Document release	Release date	Description	HW	SW	Interface
1.0	16.02.2018	First issue	2	1.0	1.2



Dispose separately

**lika**

**Lika Electronic**

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

Tel. +39 0445 806600

Fax +39 0445 806699



info@lika.biz • www.lika.biz