

RD1A RD12A



ETHERNET 
POWERLINK

- Rotary actuator with POWERLINK interface
- Complies with DS 301 profile
- 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 PL
- RD12A PL

General Contents

Safety summary	29
Identification	31
Mechanical installation	32
Electrical connections	36
POWERLINK interface	58
Modbus® interface	143
Default parameters list	208

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 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.....	11
Typographic and iconographic conventions.....	15
Preliminary information.....	16
Glossary of POWERLINK terms.....	17
List of POWERLINK abbreviations.....	23
POWERLINK references.....	25
Glossary of MODBUS terms.....	26
1 Safety summary.....	29
1.1 Safety.....	29
1.2 Electrical safety.....	29
1.3 Mechanical safety.....	30
2 Identification.....	31
3 Mechanical installation.....	32
4 Electrical connections.....	36
4.1 Ground connection (Figure 5).....	37
4.2 Connectors (Figure 4 and Figure 5).....	37
4.2.1 Power supply connector.....	38
4.2.2 POWERLINK interface connectors (P1 PORT 1 and P2 PORT 2).....	38
4.2.3 Network configuration: cables, hubs, switches - Recommendations.....	39
4.2.4 Addressing.....	40
4.2.5 Line Termination.....	40
4.2.6 Modbus RS-232 service port.....	40
4.3 Diagnostic LEDs (Figure 4 and Figure 6).....	41
4.4 Screw plug for internal access (Figure 4 and Figure 7).....	44
4.4.1 Node address (Node ID): DIP A (Figure 8).....	45
4.4.2 MAC address and IP address.....	46
4.5 Preset / Jog buttons (Figure 9).....	47
4.5.1 JOG + and JOG - buttons (Figure 9).....	47
4.5.2 PRESET button (Figure 9).....	48
5 Quick reference.....	49
6 Functions.....	51
6.1 Working principle.....	51
6.2 Movements: jog and positioning.....	52
Jog: speed control.....	52
Positioning: position and speed control.....	53
6.3 Digital inputs and output.....	54
6.4 Distance per revolution, Preset, Positive delta and Negative delta.....	55
7 POWERLINK interface.....	58
7.1 Before starting.....	58
7.1.1 Network identity.....	58
7.1.2 Network and communication settings.....	59
7.2 Configuring the actuator with Automation Studio V. 4.3 from B&R.....	59
7.3 MAC address.....	59
7.4 Installing the actuator under Automation Studio environment.....	60

7.4.1 Description of the XDD file.....	60
7.4.2 Installing the XDD file.....	61
7.4.3 Setting the device node address in the project.....	65
7.4.4 Configuring the CN device.....	66
7.4.5. Downloading the parameters to the actuator.....	69
7.4.6 Parameter values download at PLC start.....	71
7.4.7 Monitoring input and output values.....	72
7.4.8 Setting the cycle time.....	74
7.4.9 Preset setting program.....	75
7.4.10 Entering the System Diagnostics Manager (SDM).....	78
7.4.11 Logger Monitor.....	81
7.5 Overview.....	82
7.6 Physical layer.....	83
7.7 Reference model.....	84
7.8 CANopen over Ethernet.....	84
7.9 POWERLINK nodes.....	84
7.9.1 POWERLINK Managing Node (MN).....	84
7.9.2 POWERLINK Controlled Node CN).....	85
7.10 POWERLINK Basic Frame.....	85
7.11 Message types.....	86
7.11.1 Start of Cycle (SoC).....	86
7.11.2 PollRequest (PReq).....	86
7.11.3 PollResponse (PRes).....	86
7.11.4 Start of Asynchronous (SoA).....	86
7.11.5 Asynchronous Send (Asnd).....	86
7.12 POWERLINK Cycle.....	86
7.12.1 Isochronous Period.....	86
7.12.2 Asynchronous Period.....	87
7.12.3 Idle Period.....	87
7.13 CN Node NMT States.....	87
7.13.1 NMT_CS_NOT_ACTIVE.....	88
7.13.2 NMT_CS_PRE_OPERATIONAL_1.....	89
7.13.3 NMT_CS_PRE_OPERATIONAL_2.....	89
7.13.4 NMT_CS_READY_TO_OPERATE.....	89
7.13.5 NMT_CS_OPERATIONAL.....	90
7.13.6 NMT_CS_STOPPED.....	90
7.13.7 NMT_CS_BASIC_ETHERNET.....	91
7.14 XDD file.....	91
7.15 Communication objects.....	92
7.15.1 NMT Network Management.....	92
7.15.2 PDO objects.....	92
7.15.3 PDO Mapping.....	93
7.15.4 SDO objects.....	94
7.16 Object Dictionary.....	94
7.16.1 Communication Profile Area objects (DS 301).....	96
1000-00 NMT_DeviceType_U32.....	96
1001-00 ERR_ErrorRegister_U8.....	96
1006-00 NMT_CycleLen_U32.....	96
1008-00 NMT_ManufactDevName_VS.....	97
1009-00 NMT_ManufactHwVers_VS.....	97
100A-00 NMT_ManufactSwVers_VS.....	97

1018-00 NMT_IdentityObject_REC.....	97
1018-01 NMT_IdentityObject_REC.VendorID_U32.....	97
1018-02 NMT_IdentityObject_REC.ProductCode_U32.....	97
1018-03 NMT_IdentityObject_REC.RevisionNo_U32.....	97
1018-04 NMT_IdentityObject_REC.SerialNo_U32.....	98
1020-00 CFM_VerifyConfiguration_REC.....	98
1020-01 CFM_VerifyConfiguration_REC.ConfDate_U32.....	98
1020-02 CFM_VerifyConfiguration_REC.ConfTime_U32.....	98
1021-00 CFM_StoreDevDescrFile_DOM.....	98
1022-00 CFM_StoreDevDescrFormat_U16.....	99
1030-00 NMT_InterfaceGroup_0h_REC.....	99
1030-01 NMT_InterfaceGroup_0h_REC.InterfaceIndex_U16.....	99
1030-02 NMT_InterfaceGroup_0h_REC.InterfaceDescription_VSTR.....	99
1030-03 NMT_InterfaceGroup_0h_REC.InterfaceType_U8.....	99
1030-04 NMT_InterfaceGroup_0h_REC.InterfaceMtu_U16.....	99
1030-05 NMT_InterfaceGroup_0h_REC.InterfacePhysAddress_OSTR.....	100
1030-06 NMT_InterfaceGroup_0h_REC.InterfaceName_VSTR.....	100
1030-07 NMT_InterfaceGroup_0h_REC.InterfaceOperStatus_U8.....	100
1030-08 NMT_InterfaceGroup_0h_REC.InterfaceAdminState_U8.....	100
1030-09 NMT_InterfaceGroup_0h_REC.Valid_BOOL.....	100
1300-00 SDO_SequLayerTimeout_U32.....	100
1400-00 PDO_RxCommParam_00h_REC.....	101
1400-01 PDO_RxCommParam_00h_REC.NodeID_U8.....	101
1400-02 PDO_RxCommParam_00h_REC.MappingVersion_U8.....	101
1600-00 PDO_RxMappParam_00h_AU64.....	101
1600-01 PDO_RxMappParam_00h_AU64.ObjectMapping.....	102
1600-02 PDO_RxMappParam_00h_AU64.ObjectMapping.....	102
1800-00 PDO_TxCommParam_00h_REC.....	102
1800-01 PDO_TxCommParam_00h_REC.NodeID_U8.....	103
1800-02 PDO_TxCommParam_00h_REC.MappingVersion_U8.....	103
1A00-00 PDO_TxMappParam_00h_AU64.....	103
1A00-01 PDO_TxMappParam_00h_AU64.ObjectMapping.....	103
1A00-02 PDO_TxMappParam_00h_AU64.ObjectMapping.....	104
1C0B-00 DLL_CNLossSoC_REC.....	104
1C0B-01 DLL_CNLossSoC_REC.CumulativeCnt_U32.....	104
1C0B-02 DLL_CNLossSoC_REC.ThresholdCnt_U32.....	104
1C0B-03 DLL_CNLossSoC_REC.Threshold_U32.....	105
1C0F-00 DLL_CNCRCErrror_REC.....	105
1C0F-01 DLL_CNCRCErrror_REC.CumulativeCnt_U32.....	105
1C0F-02 DLL_CNCRCErrror_REC.ThresholdCnt_U32.....	105
1C0F-03 DLL_CNCRCErrror_REC.Threshold_U32.....	105
1C14-00 DLL_CNLossOfSocTolerance_U32.....	106
1F50-00 PDL_DownloadProgData_ADOM.....	106
1F50-01 PDL_DownloadProgData_ADOM.Program.....	106
1F51-00 PDL_ProgCtrl_AU8.....	106
1F51-01 PDL_ProgCtrl_AU8.ProgCtrl.....	106
1F52-00 PDL_LocVerApplSw_REC.....	107
1F52-01 PDL_LocVerApplSw_REC.ApplSwDate_U32.....	107
1F52-02 PDL_LocVerApplSw_REC.ApplSwTime_U32.....	107

1F81-00 NMT_NodeAssignment_AU32.....	107
1F81-01 NMT_NodeAssignment_AU32.NodeAssignment.....	107
1F82-00 NMT_FeatureFlags_U32.....	109
1F83-00 NMT_EPLVersion_U8.....	110
1F8C-00 NMT_CurrNMTState_U8.....	111
1F8D-00 NMT_PResPayloadLimitList_AU16.....	111
1F8D-01 NMT_PResPayloadLimitList_AU16.PResPayloadLimit.....	112
1F93-00 NMT_EPLNodeID_REC.....	112
1F93-01 NMT_EPLNodeID_REC.NodeID_U8.....	112
1F93-02 NMT_EPLNodeID_REC.NodeIDByHW_BOOL.....	112
1F98-00 NMT_CycleTiming_REC.....	112
1F98-01 NMT_CycleTiming_REC.IsochrTxMaxPayload_U16.....	113
1F98-02 NMT_CycleTiming_REC.IsochrRxMaxPayload_U16.....	113
1F98-03 NMT_CycleTiming_REC.PresMaxLatency_U32.....	113
1F98-04 NMT_CycleTiming_REC.PReqActPayloadLimit_U16.....	113
1F98-05 NMT_CycleTiming_REC.PResActPayloadLimit_U16.....	114
1F98-06 NMT_CycleTiming_REC.ASndMaxLatency_U32.....	114
1F98-07 NMT_CycleTiming_REC.MultiplCycleCnt_U8.....	114
1F98-08 NMT_CycleTiming_REC.AsyncMTU_U16.....	114
1F99-00 NMT_CNBasicEthernetTimeout_U32.....	115
1F9B-00 NMT_MultiplCycleAssign_AU8.....	115
1F9B-01 NMT_MultiplCycleAssign_AU8.CycleNo.....	115
1F9E-00 NMT_ResetCmd_U8.....	115
7.16.2 Manufacturer Specific Profile Area objects.....	117
2101-00 HMS Serial Number.....	117
2102-00 HMS_FW_Major.....	117
2103-00 HMS_FW_Minor.....	117
2104-00 HMS_FW_Build.....	117
2105-00 Position Offset.....	117
2106-00 Real Speed [rpm].....	117
2107-00 Electronics Temperature [°C].....	118
2108-00 Motor Temperature [°C].....	118
2109-00 Real Current.....	118
210A-00 Following error [pulse].....	118
210B-00 Pos. Limit Switch [pulse].....	118
210C-00 Neg. Limit Switch [pulse].....	119
210D-00 Parameter Error List.....	119
210E-00 Node ID.....	120
210F-00 Alarms List.....	120
Machine data not valid.....	121
Flash memory error.....	121
Counting error.....	121
Following error.....	121
Encoder not synchronized.....	121
Target not valid.....	121
Emergency.....	121
Overcurrent.....	121
Electronics Overtemperature.....	121
Motor Overtemperature.....	121

Undervoltage.....	122
Hall sequence.....	122
Overvoltage.....	122
2200-00 Control Word.....	122
Jog +.....	123
Jog -.....	123
Stop.....	123
Alarm reset.....	124
Incremental jog.....	124
Start.....	124
Emergency.....	124
Save parameters.....	125
Load default parameters.....	125
Setting the preset.....	125
Release axis torque.....	126
OUT 1.....	126
Brake disabled.....	126
2201-00 Target position.....	126
2202-00 Status word.....	127
Axis in position.....	128
Drive enabled.....	128
SW limit switch +.....	128
SW limit switch -.....	128
Alarm.....	128
Axis running.....	128
Executing a command.....	129
Target position reached.....	129
Button 1 Jog +.....	129
Button 2 Jog -.....	129
Button 3 Preset.....	129
PWM saturation.....	130
IN 1.....	130
IN 2.....	130
IN 3.....	130
2203-00 Position.....	130
2204-00 Distance per revolution-pulse.....	131
2205-00 Position tolerance.....	131
2206-00 Settling time-ms.....	132
2207-00 Max following error-pulse.....	132
2208-00 Proportional gain.....	132
2209-00 Integral gain.....	132
220A-00 Acceleration-rev/s².....	132
220B-00 Deceleration-rev/s².....	133
220C-00 Max delta pos-pulse.....	133
220D-00 Max delta neg-pulse.....	134
220E-00 Jog speed-rpm.....	135
220F-00 Work speed-rpm.....	135
2210-00 Count direction 0=CW,1=CCW.....	136
2211-00 Preset-pulse.....	136
2212-00 Jog step-pulse.....	137

7.17 SDO abort codes.....	139
7.18 Storing parameters.....	140
7.19 Restoring default parameters.....	140
7.20 Executing the preset.....	141
8 Modbus® interface.....	143
8.1 Configuring the device using Lika's setting up software.....	143
8.2 "Serial configuration" page.....	145
8.3 "Operative mode" page.....	147
8.4 "Parameter" page.....	154
8.5 "Message monitor" page.....	156
8.6 "Test Lika" page.....	157
8.7 "Upgrade Firmware" page.....	157
8.7.1 If an installation issue occurs.....	159
8.8 Modbus Master / Slaves protocol principle.....	160
8.9 Modbus frame description.....	161
8.10 Transmission modes.....	163
8.10.1 RTU transmission mode.....	163
8.11 Function codes.....	165
8.11.1 Implemented function codes.....	165
03 Read Holding Registers.....	165
04 Read Input Register.....	167
06 Write Single Register.....	170
16 Write Multiple Registers.....	171
8.12 Programming parameters.....	175
8.12.1 Holding Register parameters.....	175
Distance per revolution [0x00].....	176
Position window [0x01].....	177
Position window time [0x02].....	177
Max following error [0x03-0x04].....	177
Kp position loop [0x05].....	177
Ki position loop [0x06].....	177
Acceleration [0x07].....	178
Deceleration [0x08].....	178
Positive delta [0x09-0x0A].....	178
Negative delta [0x0B-0x0C].....	179
Jog speed [0x0D].....	180
Work speed [0x0E].....	180
Code sequence [0x0F].....	181
Offset [0x10-0x11].....	181
Preset [0x12-0x13].....	181
Jog step length [0x14].....	182
Extra commands register [0x29].....	182
Control by PC.....	183
Control Word [0x2A].....	183
Jog +.....	183
Jog -.....	184
Stop.....	184
Alarm reset.....	184
Incremental jog.....	184
Start.....	185

Emergency.....	185
Watch dog enable.....	185
Save parameters.....	186
Load default parameters.....	186
Setting the preset.....	186
Release axis torque.....	187
OUT 1.....	187
Brake disabled.....	187
Target position [0x2B-0x2C]	188
8.12.2 Input Register parameters.....	190
Alarms register [0x00]	190
Machine data not valid.....	191
Flash memory error.....	191
Counting error.....	191
Following error.....	191
Encoder not synchronized.....	191
Target not valid.....	191
Emergency.....	191
Overcurrent.....	191
Electronics Overtemperature.....	191
Motor Overtemperature.....	192
Undervoltage.....	192
Watch dog.....	192
Hall sequence.....	192
Overvoltage.....	192
Status word [0x01]	193
Axis in position.....	193
Drive enabled.....	193
SW limit switch +.....	193
SW limit switch -.....	194
Alarm.....	194
Axis running.....	194
Executing a command.....	194
Target position reached.....	194
Button 1 Jog +.....	194
Button 2 Jog -.....	194
Button 3 Preset.....	195
PWM saturation.....	195
IN 1.....	195
IN 2.....	195
IN 3.....	195
Current position [0x02-0x03]	196
Current velocity [0x04]	196
Position following error [0x05-0x06]	196
Temperature value [0x07]	196
Wrong parameters list [0x08-0x09]	196
Motor voltage [0x0A]	197
Current value [0x0B]	197
Hall [0x0C]	197
Duty cycle [0x0D]	198
DIP switch baud rate [0x0E]	198

DIP switch node ID [0x0F].....	198
SW Version [0x10].....	198
HW Version [0x11].....	198
8.13 Exception codes.....	201
8.14 Programming examples.....	202
8.14.1 Using the 03 Read Holding Registers function code.....	202
8.14.2 Using the 04 Read Input Register function code.....	203
8.14.3 Using the 06 Write Single Register function code.....	205
8.14.4 Using the 16 Write Multiple Registers function code.....	207
9 Default parameters list.....	208

Subject Index

1

1000-00 NMT_DeviceType_U32.....	96
1001-00 ERR_ErrorRegister_U8.....	96
1006-00 NMT_CycleLen_U32.....	96
1008-00 NMT_ManufactDevName_VS.....	97
1009-00 NMT_ManufactHwVers_VS.....	97
100A-00 NMT_ManufactSwVers_VS.....	97
1018-00 NMT_IdentityObject_REC.....	97
1018-01 NMT_IdentityObject_REC.VendorID_U32	97
1018-02 NMT_IdentityObject_REC.ProductCode_U32.	97
1018-03 NMT_IdentityObject_REC.RevisionNo_U32.....	97
1018-04 NMT_IdentityObject_REC.SerialNo_U32	98
1020-00 CFM_VerifyConfiguration_REC.....	98
1020-01 CFM_VerifyConfiguration_REC.ConfDate_U32	98
1020-02 CFM_VerifyConfiguration_REC.ConfTime_U32	98
1021-00 CFM_StoreDevDescrFile_DOM.....	98
1022-00 CFM_StoreDevDescrFormat_U16.....	99
1030-00 NMT_InterfaceGroup_0h_REC.....	99
1030-01 NMT_InterfaceGroup_0h_REC.InterfaceIndex_ U16.....	99
1030-02 NMT_InterfaceGroup_0h_REC.InterfaceDescrip tion_VSTR.....	99
1030-03 NMT_InterfaceGroup_0h_REC.InterfaceType_U 8.....	99
1030-04 NMT_InterfaceGroup_0h_REC.InterfaceMtu_U 16.....	99
1030-05 NMT_InterfaceGroup_0h_REC.InterfacePhysAd dress_OSTR.....	100
1030-06 NMT_InterfaceGroup_0h_REC.InterfaceName_ VSTR.....	100

1030-07 NMT_InterfaceGroup_0h_REC.InterfaceOperSt atus_U8.....	100
1030-08 NMT_InterfaceGroup_0h_REC.InterfaceAdminS tate_U8.....	100
1030-09 NMT_InterfaceGroup_0h_REC.Valid_BOOL...	100
1300-00 SDO_SequLayerTimeout_U32.....	100
1400-00 PDO_RxCommParam_00h_REC.....	101
1400-01 PDO_RxCommParam_00h_REC.NodeID_U8.	101
1400-02 PDO_RxCommParam_00h_REC.MappingVersio n_U8.....	101
1600-00 PDO_RxMappParam_00h_AU64.....	101
1600-01 PDO_RxMappParam_00h_AU64.ObjectMappin g.....	102
1600-02 PDO_RxMappParam_00h_AU64.ObjectMappin g.....	102
1800-00 PDO_TxCommParam_00h_REC.....	102
1800-01 PDO_TxCommParam_00h_REC.NodeID_U8.	103
1800-02 PDO_TxCommParam_00h_REC.MappingVersion _U8.....	103
1A00-00 PDO_TxMappParam_00h_AU64.....	103
1A00-01 PDO_TxMappParam_00h_AU64.ObjectMapping	103
1A00-02 PDO_TxMappParam_00h_AU64.ObjectMapping	104
1C0B-00 DLL_CNLossSoC_REC.....	104
1C0B-01 DLL_CNLossSoC_REC.CumulativeCnt_U32...	104
1C0B-02 DLL_CNLossSoC_REC.ThresholdCnt_U32	104
1C0B-03 DLL_CNLossSoC_REC.Threshold_U32	105
1C0F-00 DLL_CNCRCErrror_REC.....	105
1C0F-01 DLL_CNCRCErrror_REC.CumulativeCnt_U32.	105

1C0F-02	DLL_CNCRCErrror_REC.ThresholdCnt_U32.....	105
1C0F-03	DLL_CNCRCErrror_REC.Threshold_U32.....	105
1C14-00	DLL_CNLossOfSocTolerance_U32.....	106
1F50-00	PDL_DownloadProgData_ADOM.....	106
1F50-01	PDL_DownloadProgData_ADOM.Program.....	106
1F51-00	PDL_ProgCtrl_AU8.....	106
1F51-01	PDL_ProgCtrl_AU8.ProgCtrl.....	106
1F52-00	PDL_LocVerApplSw_REC.....	107
1F52-01	PDL_LocVerApplSw_REC.ApplSwDate_U32... ..	107
1F52-02	PDL_LocVerApplSw_REC.ApplSwTime_U32. .	107
1F81-00	NMT_NodeAssignment_AU32.....	107
1F81-01	NMT_NodeAssignment_AU32.NodeAssignment.....	107
1F82-00	NMT_FeatureFlags_U32.....	109
1F83-00	NMT_EPLVersion_U8.....	110
1F8C-00	NMT_CurrNMTState_U8.....	111
1F8D-00	NMT_PResPayloadLimitList_AU16.....	111
1F8D-01	NMT_PResPayloadLimitList_AU16.PResPayloadLimit.....	112
1F93-00	NMT_EPLNodeID_REC.....	112
1F93-01	NMT_EPLNodeID_REC.NodeID_U8.....	112
1F93-02	NMT_EPLNodeID_REC.NodeIDByHW_BOOL..	112
1F98-00	NMT_CycleTiming_REC.....	112
1F98-01	NMT_CycleTiming_REC.IsochrTxMaxPayload_U16.....	113
1F98-02	NMT_CycleTiming_REC.IsochrRxMaxPayload_U16.....	113
1F98-03	NMT_CycleTiming_REC.PresMaxLatency_U32.....	113
1F98-04	NMT_CycleTiming_REC.PReqActPayloadLimit_U16.....	113
1F98-05	NMT_CycleTiming_REC.PResActPayloadLimit_U16.....	114
1F98-06	NMT_CycleTiming_REC.ASndMaxLatency_U32.....	114
1F98-07	NMT_CycleTiming_REC.MultiplCycleCnt_U8	114

1F98-08	NMT_CycleTiming_REC.AsyncMTU_U16.....	114
1F99-00	NMT_CNBasicEthernetTimeout_U32..	115
1F9B-00	NMT_MultiplCycleAssign_AU8.....	115
1F9B-01	NMT_MultiplCycleAssign_AU8.CycleNo.....	115
1F9E-00	NMT_ResetCmd_U8.....	115
2		
2101-00	HMS Serial Number.....	117
2102-00	HMS_FW_Major.....	117
2103-00	HMS_FW_Minor.....	117
2104-00	HMS_FW_Build.....	117
2105-00	Position Offset.....	117
2106-00	Real Speed [rpm].....	117
2107-00	Electronics Temperature [°C].....	118
2108-00	Motor Temperature [°C].....	118
2109-00	Real Current.....	118
210A-00	Following error [pulse].....	118
210B-00	Pos. Limit Switch [pulse].....	118
210C-00	Neg. Limit Switch [pulse].....	119
210D-00	Parameter Error List.....	119
210E-00	Node ID.....	120
210F-00	Alarms List.....	120
2200-00	Control Word.....	122
2201-00	Target position.....	126
2202-00	Status word.....	127
2203-00	Position.....	130
2204-00	Distance per revolution-pulse.....	131
2205-00	Position tolerance.....	131
2206-00	Settling time-ms.....	132
2207-00	Max following error-pulse.....	132
2208-00	Proportional gain.....	132
2209-00	Integral gain.....	132
220A-00	Acceleration-rev/s ²	132
220B-00	Deceleration-rev/s ²	133
220C-00	Max delta pos-pulse.....	133
220D-00	Max delta neg-pulse.....	134
220E-00	Jog speed-rpm.....	135
220F-00	Work speed-rpm.....	135
2210-00	Count direction 0=CW,1=CCW.....	136
2211-00	Preset-pulse.....	136
2212-00	Jog step-pulse.....	137
A		
	Acceleration [0x07].....	178
	Alarm.....	128, 194
	Alarm reset.....	124, 184
	Alarms register [0x00].....	190
	Axis in position.....	128, 193
	Axis running.....	128, 194
B		
	Brake disabled.....	126, 187

Button 1 Jog +.....	129, 194
Button 2 Jog -.....	129, 194
Button 3 Preset.....	129, 195

C

CFM_VerifyConfiguration_REC.....	98
CFM_VerifyConfiguration_REC.ConfDate_U32.....	98
CFM_VerifyConfiguration_REC.ConfTime_U32.....	98
Code sequence [0x0F].....	181
Control by PC.....	183
Control Word [0x2A].....	183
Counting error.....	121, 191
Current position [0x02-0x03].....	196
Current value [0x0B].....	197
Current velocity [0x04].....	196

D

Deceleration [0x08].....	178
DIP switch baud rate [0x0E].....	198
DIP switch node ID [0x0F].....	198
Distance per revolution [0x00].....	176
DLL_CNCRCErrror_REC.....	105
DLL_CNCRCErrror_REC.CumulativeCnt_U32.....	105
DLL_CNCRCErrror_REC.Threshold_U32.....	105
DLL_CNCRCErrror_REC.ThresholdCnt_U32.....	105
DLL_CNLossOfSocTolerance_U32.....	106
DLL_CNLossSoC_REC.....	104
DLL_CNLossSoC_REC.CumulativeCnt_U32.....	104
DLL_CNLossSoC_REC.Threshold_U32.....	105
DLL_CNLossSoC_REC.ThresholdCnt_U32.....	104
Drive enabled.....	128, 193
Duty cycle [0x0D].....	198

E

Electronics Overtemperature.....	121, 191
Emergency.....	121, 124, 185, 191
Encoder not synchronized.....	121, 191
ERR_ErrorRegister_U8.....	96
Executing a command.....	129, 194
Extra commands register [0x29].....	182

F

Flash memory error.....	121, 191
Following error.....	121, 191

H

Hall [0x0C].....	197
Hall sequence.....	122, 192
HW Version [0x11].....	198

I

IN 1.....	130, 195
IN 2.....	130, 195
IN 3.....	130, 195
Incremental jog.....	124, 184

J

Jog -.....	123, 184
------------	----------

Jog +.....	123, 183
Jog speed [0x0D].....	180
Jog step length [0x14].....	182

K

Ki position loop [0x06].....	177
Kp position loop [0x05].....	177

L

Load default parameters.....	125, 186
------------------------------	----------

M

Machine data not valid.....	121, 191
Max following error [0x03-0x04].....	177
Motor Overtemperature.....	121, 192
Motor voltage [0x0A].....	197

N

Negative delta [0x0B-0x0C].....	179
NMT_CNBasicEthernetTimeout_U32.....	115
NMT_CS_BASIC_ETHERNET.....	91
NMT_CS_NOT_ACTIVE.....	88
NMT_CS_OPERATIONAL.....	90
NMT_CS_PRE_OPERATIONAL_1.....	89
NMT_CS_PRE_OPERATIONAL_2.....	89
NMT_CS_READY_TO_OPERATE.....	89
NMT_CS_STOPPED.....	90
NMT_CurrNMTState_U8.....	111
NMT_CycleLen_U32.....	96
NMT_CycleTiming_REC.....	112
NMT_CycleTiming_REC.ASndMaxLatency_U32.....	114
NMT_CycleTiming_REC.AsyncMTU_U16.....	114
NMT_CycleTiming_REC.IsochrRxMaxPayload_U16.....	113
NMT_CycleTiming_REC.IsochrTxMaxPayload_U16.....	113
NMT_CycleTiming_REC.MultiplCycleCnt_U8.....	114
NMT_CycleTiming_REC.PReqActPayloadLimit_U16.....	113
NMT_CycleTiming_REC.PResActPayloadLimit_U16.....	114
NMT_CycleTiming_REC.PresMaxLatency_U32.....	113
NMT_DeviceType_U32.....	96
NMT_EPLNodeID_REC.....	112
NMT_EPLNodeID_REC.NodeID_U8.....	112
NMT_EPLNodeID_REC.NodeIDByHW_BOOL.....	112
NMT_EPLVersion_U8.....	110
NMT_FeatureFlags_U32.....	109
NMT_IdentityObject_REC.....	97
NMT_IdentityObject_REC.ProductCode_U32.....	97
NMT_IdentityObject_REC.RevisionNo_U32.....	97
NMT_IdentityObject_REC.SerialNo_U32.....	98
NMT_IdentityObject_REC.VendorID_U32.....	97
NMT_InterfaceGroup_0h_REC.....	99

NMT_InterfaceGroup_0h_REC.InterfaceAdminSta	
te_U8.....	100
NMT_InterfaceGroup_0h_REC.InterfaceDescripti	
on_VSTR.....	99
NMT_InterfaceGroup_0h_REC.InterfaceIndex_U1	
6.....	99
NMT_InterfaceGroup_0h_REC.InterfaceMtu_U16	
.....	99
NMT_InterfaceGroup_0h_REC.InterfaceName_VS	
TR.....	100
NMT_InterfaceGroup_0h_REC.InterfaceOperStat	
us_U8.....	100
NMT_InterfaceGroup_0h_REC.InterfacePhysAddr	
ess_OSTR.....	100
NMT_InterfaceGroup_0h_REC.InterfaceType_U8	
.....	99
NMT_InterfaceGroup_0h_REC.Valid_BOOL.....	100
NMT_ManufactDevName_VS.....	97
NMT_ManufactHwVers_VS.....	97
NMT_ManufactSwVers_VS.....	97
NMT_MultiplCycleAssign_AU8.....	115
NMT_MultiplCycleAssign_AU8.CycleNo.....	115
NMT_NodeAssignment_AU32.....	107
NMT_NodeAssignment_AU32.NodeAssignment	
.....	107
NMT_PResPayloadLimitList_AU16.....	111
NMT_PResPayloadLimitList_AU16.PResPayloadLi	
mit.....	112
NMT_ResetCmd_U8.....	115
O	
Offset [0x10-0x11].....	181
OUT 1.....	126, 187
Overcurrent.....	121, 191
Overvoltage.....	122, 192
P	
PDO_TxCommParam_00h_REC.....	102




PDO_TxCommParam_00h_REC.MappingVersion_	
U8.....	103
PDO_TxCommParam_00h_REC.NodeID_U8.....	103
PDO_TxMappParam_00h_AU64.....	103
PDO_TxMappParam_00h_AU64.ObjectMapping	
.....	103 e seg.
Position following error [0x05-0x06].....	196
Position window [0x01].....	177
Position window time [0x02].....	177
Positive delta [0x09-0x0A].....	178
Preset [0x12-0x13].....	181
PWM saturation.....	130, 195
R	
Release axis torque.....	126, 187
S	
Save parameters.....	125, 186
SDO_SequLayerTimeout_U32.....	100
Setting the preset.....	125, 186
Start.....	124, 185
Status word [0x01].....	193
Stop.....	123, 184
SW limit switch -.....	128, 194
SW limit switch +.....	128, 193
SW Version [0x10].....	198
T	
Target not valid.....	121, 191
Target position [0x2B-0x2C].....	188
Target position reached.....	129, 194
Temperature value [0x07].....	196
U	
Undervoltage.....	122, 192
W	
Watch dog.....	192
Watch dog enable.....	185
Work speed [0x0E].....	180
Wrong parameters list [0x08-0x09].....	196

Typographic and iconographic conventions

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

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

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

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

Preliminary information

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

RD1A and RD12A units are positioning devices which integrate into one system a brushless motor fitted with gearbox, a drive, a multiturn absolute encoder and a position controller. 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.

RD1xA 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 POWERLINK 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 POWERLINK 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 **POWERLINK Interface**, both general and specific information is given on the POWERLINK 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, POWERLINK 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 POWERLINK network. In the **Modbus Interface** section the interface features and the registers implemented in the unit are fully described.

Glossary of POWERLINK terms

POWERLINK, 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 POWERLINK interface. They are listed in alphabetical order.

Ageing	Ageing is a common mechanism to maintain (cache) tables. Entries which are not used or refreshed are removed after a specified time.
Application Process	The Application Process is the task on the Application Layer.
Async-only CN	An Async-only CN is operated in a way, that it is not accessed cyclically in the isochronous slot by the MN. It is polled during the asynchronous period by a StatusRequest message.
Asynchronous Data	Data in a POWERLINK network which is not time critical. Within the POWERLINK cycle there is a specific period reserved for Asynchronous Data which is shared by all nodes. Each node connected to the network can send asynchronous data by requesting it to the Managing Node. The Managing Node keeps a list of all asynchronous data requests and will subsequently grant the network access to one node after the other. Refer also to page 87.
Asynchronous Period	The Asynchronous Period is the second part of the POWERLINK cycle, starting with a Start of Asynchronous (SoA) frame. Refer to page 87.
Asynchronous Scheduling	The MN's asynchronous scheduler decides when a requested asynchronous data transfer will happen.
Basic Ethernet Mode	Basic Ethernet Mode provides the Legacy Ethernet communication. Refer also to page 91.
Bus	A bus is a communication medium connecting several nodes. Data can be transferred via serial or parallel circuits, that is, via electrical conductors or fiber optic.
CANopen	CANopen is a network technology optimized for the usage in industrial control environments, in machine internal networks and in embedded systems (any control unit deeply "embedded" in a device with electronics). The lower-layer implementation of CANopen is based upon CAN (Controller Area Network).
Continuous	Continuous is a POWERLINK communication class where isochronous communication takes place every cycle (the opposite to multiplexed).
Controlled Node (CN)	Node in a POWERLINK network without the ability to manage the SCNM mechanism. Refer to page 85.

Cycle State Machine	The Cycle State Machine controls the POWERLINK cycle on the Data Link Layer and is itself controlled by the NMT state machine defining the current operating mode.
Cycle Time	The time between two consecutive Start of Cyclic (SoC) frames – i.e. repeating – process. The Cycle Time includes the time for data transmission and some idle time before the beginning of the next cycle. Refer also to page 74.
Destination NAT (D-NAT)	D-NAT (Destination- Network Address Translation) changes the destination address of the IP / ICMP packet.
Determinism	Determinism means that a system responds in a predictable (deterministic) manner.
Deterministic Communication	It describes a communication process whose timing behaviour can be predicted exactly. I.e. the time when a message reaches the recipient is predictable. Refer to page 86.
Device Configuration File	The configuration parameters of a specific device are stored in the Device Configuration File (XDC).
Device Description File	All device dependent information is stored in the Device Description File (XDD) of each device. Refer to page 91.
Domain	In the context of CANopen: A Domain is a data object of arbitrary type and length which can be transferred over a POWERLINK network. In the context of internet protocols: A Domain is a part of the internet name space which is supported by the Domain Name System (DNS).
Encoder Profile	POWERLINK integrates with CANopen Profiles "DS301 CANopen Application Layer and Communication Profile" and "DS406 Device Profile for Encoders" for device interoperability. The "DS406 Device Profile for Encoders" is intended to define a standard application interface for encoders. The profile is a supplement to the "DS301 CANopen Application Layer and Communication Profile", so it is mandatory to read the DS301 profile before implementing the encoder profile. POWERLINK encoders from Lika Electronic comply with the "EPSG Draft Standard 301 Ethernet POWERLINK Communication Profile Specification Version 1.2.0". See also "Profile". Refer to page 84.
Ethernet POWERLINK (EPL)	An extension to Legacy Ethernet on layer 2, to exchange data under hard real-time constraints. It was developed for deterministic data exchange, short cycle time and isochronous operation in industrial automation.
IdentRequest	IdentRequests are POWERLINK frames sent by the MN in order to identify active CNs waiting to be included into the network.
IdentResponse	The IdentResponse is a special form of an ASnd frame in response to an IdentRequest.
Idle Period	The Idle Period is time interval remaining between the completed asynchronous period and the beginning of the next cycle. Refer also to page 87.
IEEE 1588	This standard defines a protocol enabling synchronisation of

	clocks in distributed networked devices (e.g. connected via Ethernet).
IP address	The IP address is the name of the unit in a network using the Internet protocol. Refer to page 46.
Isochronous	Pertains to processes that require timing coordination to be successful. Isochronous data transfer ensures that data flows continuously and at a steady rate in close timing with the ability of connected devices.
Isochronous Data	Data in a POWERLINK network which is to be transmitted every cycle (or every nth cycle in case of multiplexed isochronous data). Refer also to page 86.
Isochronous Period	The Isochronous Period of a POWERLINK cycle offers deterministic operation, i.e. it is reserved for the exchange of (continuous or multiplexed) isochronous data. Refer to page 86.
Legacy Ethernet	Ethernet as standardised in IEEE 802.3 (non-deterministic operation in non-time-critical environments).
MAC address	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. Refer to page 46.
Managing Node (MN)	A node capable to manage the SCNM mechanism in a POWERLINK network. Refer to page 84.
Media Access Control (MAC)	One of the sub-layers of the Data Link Layer in the POWERLINK reference model that controls who gets access to the medium to send a message.
Multiplexed	Multiplexed is a POWERLINK communication class where cyclic communication takes place in such a way that m nodes are served in s cycles (the opposite to continuous).
Multiplexed CN	A node which is allowed to send isochronous data every n th cycle.
Multiplexed Timeslot	A slot destined to carry multiplexed isochronous data, i.e. the timeslot is shared among multiple nodes.
NetTime	The MN's clock time is distributed to all CNs within the SoC frame.
Network Management (NMT)	Network Management functions and services in the POWERLINK model. It performs initialisation, configuration and error handling in a POWERLINK network. Refer to page 92.
NMT State Machine	The state machine controlling the overall operating mode and status of a POWERLINK node. Refer to page 92.
Object Directory	The repository of all data objects accessible over POWERLINK communications.
PollRequest	A PollRequest is a frame, which is used in the isochronous part of the cyclic communication. The MN request with this frame

	the CN to send its data. Refer to page 86.
PollResponse	A PollResponse is a frame, which is used in the isochronous part of the cyclic communication. The CN responds with this frame to a PollRequest frame from an MN. Refer to page 86.
POWERLINK Command Layer	The POWERLINK Command Layer defines commands to access parameters of the object dictionary. This layer is on top of the Sequence Layer and distinguishes between an expedited and a segmented transfer.
POWERLINK Cycle	Data exchange within a POWERLINK network is structured in fix intervals, called cycles. The cycle is subdivided into the isochronous and the asynchronous period and is organized by the MN. Refer to page 86.
POWERLINK Mode	The POWERLINK Mode includes all NMT states in which POWERLINK cycles are run. Refer to page 87.
POWERLINK Node ID	Each POWERLINK node (MN, CN and Router) is addressed by an 8 bit POWERLINK Node ID on the POWERLINK layer. This ID has only local significance (i.e. it is unique within a POWERLINK segment). Refer to page 46.
Precision Time Protocol (PTP)	IEEE 1588, Standard for a Precision Clock Synchronisation Protocol for Networked Measurement and Control Systems.
Process Data Object (PDO)	Object for isochronous data exchange between POWERLINK nodes.
Profile	Profiles define application-specific functionality to ensure the openness of POWERLINK is utilized consistently. 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 guarantee quicker system design and they support faster device interchange, promoting competition amongst vendors, increased choice for users and full interoperability. POWERLINK encoders from Lika Electronic comply with the "EPSG Draft Standard 301 Ethernet POWERLINK Communication Profile Specification Version 1.2.0". See also "Encoder profile". See on page 82.
Real-time	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. Refer to page 86.
Reserved	Reserved bits shall be set 0 by the sender. The receiver shall not interpret such bits. It is not allowed to use reserved bits. Their use is reserved for further development or by extensions of this specification.
Router Type 1	A Type 1 POWERLINK Router is a coupling element in a network that allows IP communication between a POWERLINK segment and any other datalink layer protocol carrying IP e.g.

	legacy Ethernet, POWERLINK etc. It is usually a separate network element acting as Controlled Node within the POWERLINK segment.
Router Type 2	A Type 2 POWERLINK Router is a router between a POWERLINK segment and a CANopen network.
Sequence Layer	The POWERLINK Sequence Layer provides the service of a reliable bidirectional connection that guarantees that no messages are lost or duplicated and that all messages arrive in the correct order.
Service Data Object (SDO)	Peer to peer communication with access to the object dictionary of a device. Refer to page 94.
Slot Communication Network Management (SCNM)	In a POWERLINK network, the managing node allocates data transfer time for data from each node in a cyclic manner within a guaranteed cycle time. Within each cycle there are slots for Isochronous Data, as well as for Asynchronous Data for ad hoc communication. The SCNM mechanism ensures that there are no collisions during physical network access of any of the networked nodes thus providing deterministic communication via Legacy Ethernet. Refer to page 82.
Source NAT (S-NAT)	S-NAT (Source - Network Address Translation) changes the source address of the IP / ICMP packet.
StatusRequest	A StatusRequest frame is a special SoA frame used to poll the status of a node.
StatusResponse	A StatusResponse frame is transmitted by a CN upon assignment of the asynchronous slot via the StatusRequest in the SoA frame.
TCP/IP	<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"> 1. The sender splits the data into a sequence of packets. 2. The packets are transported over the Ethernet to the correct recipient. 3. The recipient reassembles the data packets in the correct order. 4. Faulty packets are sent again until the recipient acknowledges that they have been transferred successfully.
Topology	<p>Network structure. Commonly used structures:</p> <ul style="list-style-type: none"> • Line topology; • Ring topology; • Star topology; • Tree topology. <p>Refer to page 82.</p>

Transmission rate	Data transfer rate (in bps). Refer to page 83.
-------------------	--

List of POWERLINK abbreviations

Table below contains a list of abbreviations (in alphabetical order) which may be used in this guide to describe the POWERLINK interface.

ACL	Access Control List
ARP	Address Resolution Protocol
ASnd	Asynchronous Send (POWERLINK frame type)
CAN	Controller Area Network
CiA	CAN in Automation
CN	POWERLINK Controlled Node
DCF	Device Configuration File
EA	Exception Acknowledge (flag in POWERLINK frame)
EIA	Electronic Industries Association
EMC	Electro Magnetic Compatibility
EN	Exception New (flag in POWERLINK frame)
EPL	Ethernet POWERLINK
EPSCG	Ethernet POWERLINK Standardisation Group
ICMP	Internet Control Message Protocol
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
MAC	Media Access Control
MIB	Management Information Base
MN	POWERLINK Managing Node
MS	Multiplexed Slot (flag in POWERLINK frame)
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NIL	Not in List (Basic Data Type)
NMT	Network Management
PDO	Process Data Object
PR	Priority (bit field in POWERLINK frame)
PReq	PollRequest (POWERLINK frame type)

PRes	PollResponse (POWERLINK frame type)
PS	Prescaled Slot (flag in POWERLINK frame)
PTP	Precision Time Protocol
RD	Ready (flag in POWERLINK frame)
RFC	Requests for Comments
RPDO	Receive Process Data Object
RS	Request to Send (flag in POWERLINK frame)
SCNM	Slot Communication Network Management
SDO	Service Data Object
SNMP	Simple Network Management Protocol
SoA	Start of Asynchronous (POWERLINK frame type)
SoC	Start of Cyclic (POWERLINK frame type)
TCP	Transmission Control Protocol
TIA	Telecommunications Industry Association
TPDO	Transmit Process Data Object
UDP	User Datagram Protocol
VPN	Virtual Private Network
XDC	XML device configuration file
XDD	XML device description file

POWERLINK references

- [1] EPSG Draft Standard 301, Ethernet POWERLINK, Communication Profile Specification, Version 1.2.0
- [2] EPSG Draft Standard 311, Ethernet POWERLINK, XML Device Description, Version 1.0.0
- [3] CiA Draft Standard Proposal 301, Application layer and communication profile, Version 4.2.0
- [4] CiA Draft Standard 406, Device profile for encoders, Version 4.0.1
- [5] EPSG Draft Standard 302-A (EPSG DS 302-A), Ethernet POWERLINK, Part A: High Availability, Version 1.1.0
- [6] EPSG Draft Standard 302-B (EPSG DS 302-B), Ethernet POWERLINK, Part B: Multiple-ASnd, Version 1.0.0
- [7] EPSG Draft Standard 302-C (EPSG DS 302-C), Ethernet POWERLINK, Part C: PollResponse Chaining, Version 1.0.0
- [8] EPSG Draft Standard 302-D (EPSG WDP 302-D), Ethernet POWERLINK, Part D: Multiple PReq/Pres, Version 1.0.0
- [9] EPSG Draft Standard 302-E (EPSG WDP 302-E), Ethernet POWERLINK, Part E: Dynamic Node Allocation, Version 1.0.0
- [10] IEC 61918 Industrial communication networks – Installation of communication networks in industrial premises
- [11] IEC 61784-5-13 Industrial communication networks – Profiles – Part 5-13: Installation of fieldbuses – Installation profiles for CPF 13

Glossary of MODBUS terms

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

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

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

PDU	<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"> • MODBUS Request PDU, mb_req_pdu • MODBUS Response PDU, mb_rsp_pdu • MODBUS Exception Response PDU, mb_except_rsp_pdu
Read Holding Registers (03, 0003hex)	This function code is used to READ the contents of a contiguous block of holding registers in a remote device; in other words, it allows to read the values set in a group of work parameters placed in order.
Read Input Register (04, 0004hex)	This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some result values and state / alarm messages in a remote device.
Register	MODBUS functions operate on memory registers to configure, monitor, and control device I/O.
RTU transmission mode	Remote Terminal Unit. When devices communicate on a MODBUS serial line using the RTU mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII mode for the same baud rate. Each message must be transmitted in a continuous stream of characters.
Server	<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>
Service request	It is the MODBUS Request, i.e. the message sent on the network by the Client to initiate a transaction.
Slave	A Slave is any program that awaits data requests to be sent to it. Slaves do not initiate contacts with Masters, but only respond to them.
Transmission rate	Data transfer rate (in bps).
Write Multiple Registers (16, 0010hex)	This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
Write Single Register (06, 0006hex)	This function code is used to WRITE a single holding register in a remote device.

1 Safety summary



1.1 Safety

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



1.2 Electrical safety

- Turn OFF power supply before connecting the device;
- connect according to explanation in the "Electrical connections" section on page 36;
- 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 32;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the unit;
- do not tool the unit or its shaft;
- delicate electronic equipment: handle with care; do not subject the device and the shaft to knocks or shocks;
- respect the environmental characteristics of the product;
- the actuator can be mounted directly on the drive shaft or coupled with planetary gearboxes. An adapting flange can be further interposed.



WARNING

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



WARNING

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

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](#).

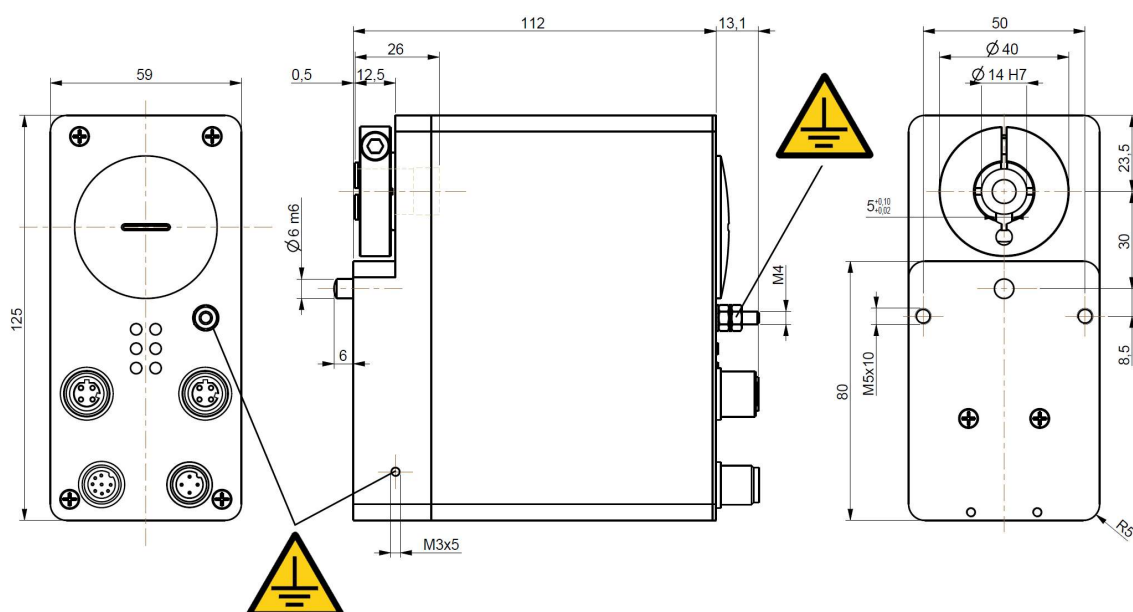


3 Mechanical installation



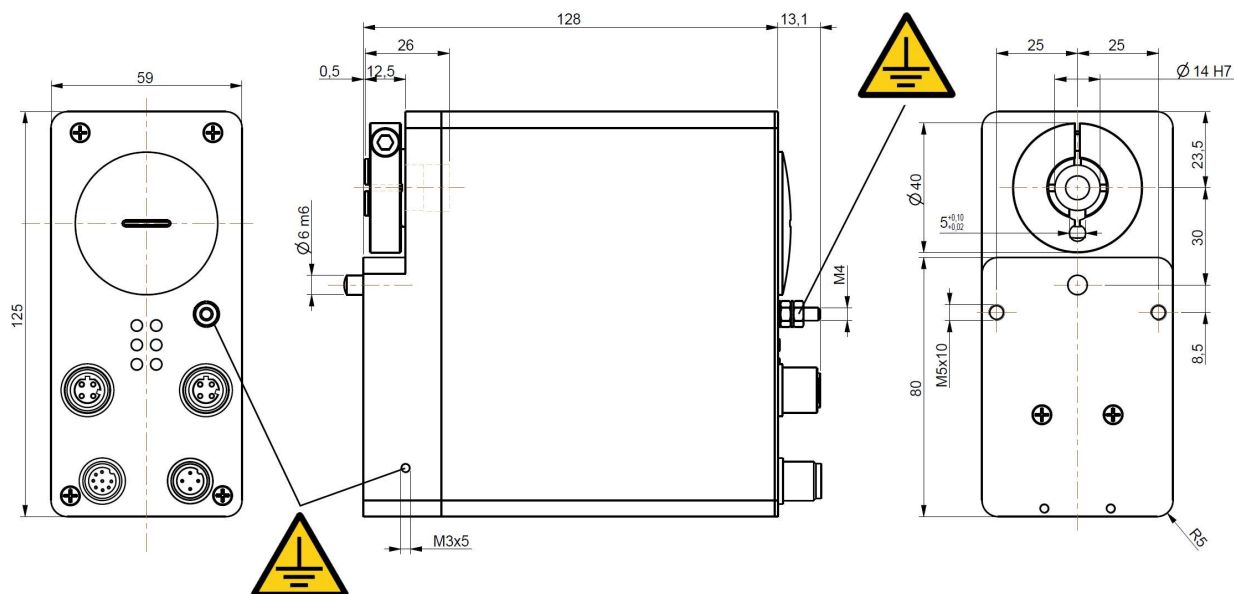
WARNING

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!

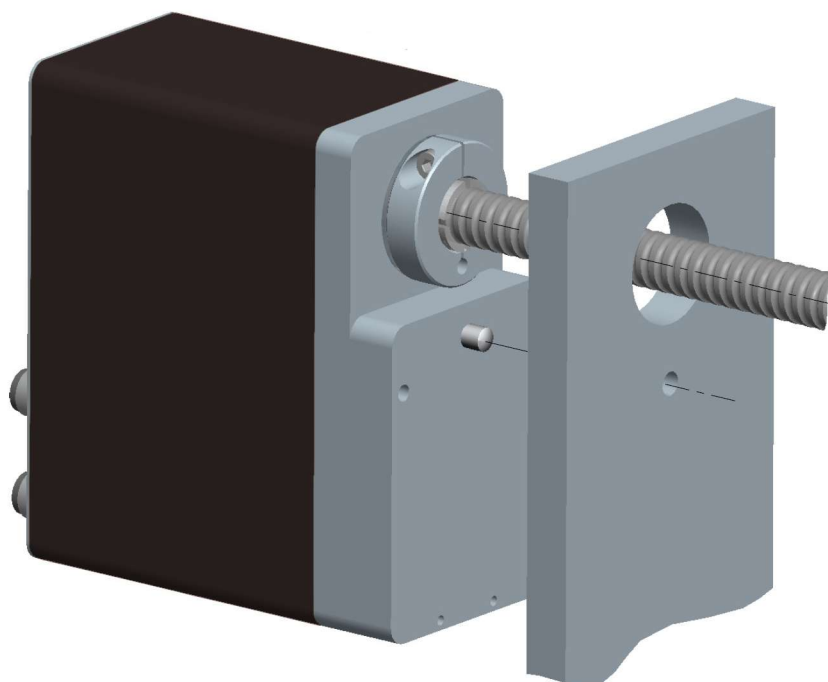
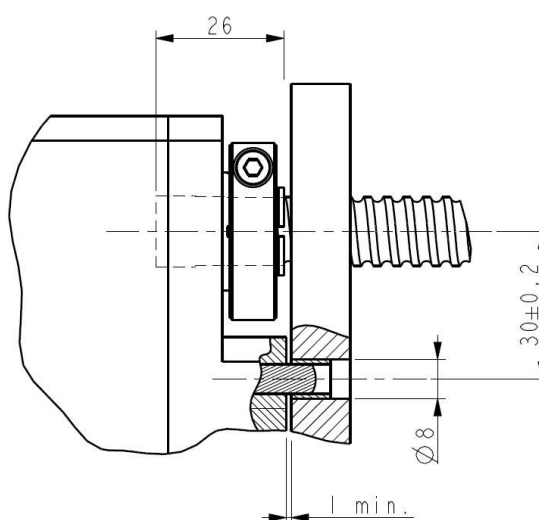


Figure 3 - Typical installation example of RD1xA unit on worm screw

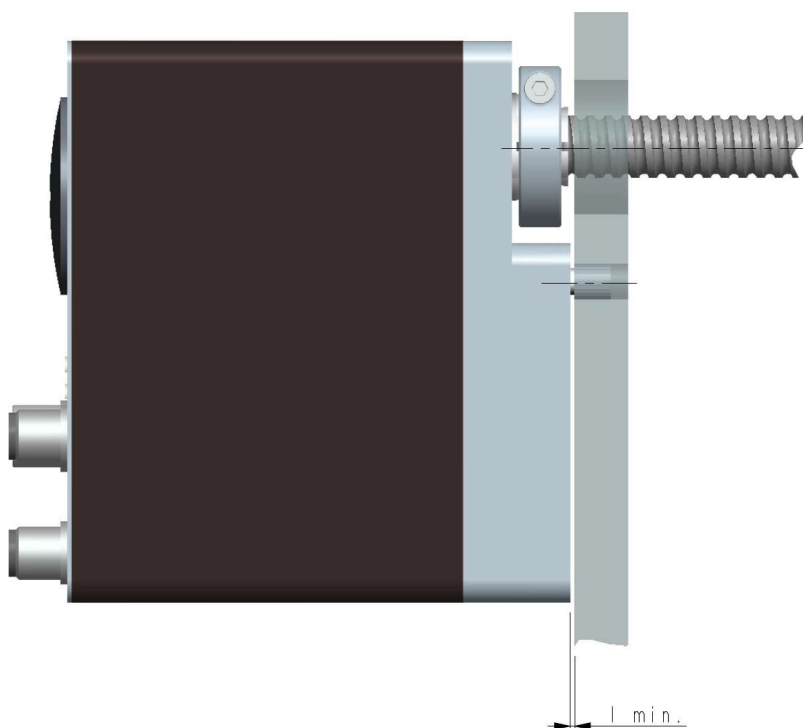
To install properly the DRIVECOD unit please read carefully and follow the instructions hereafter; anyway note that the unit can be installed in several manners and according to the specific user's application.

- 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!

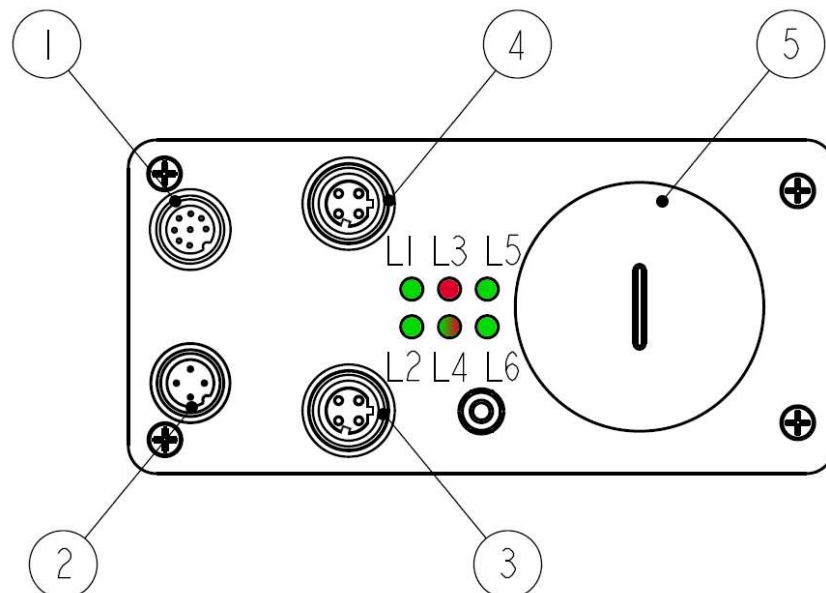


Figure 4: Connectors and diagnostic LEDs

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 2 (P2)
4	M12 4-pin female connector PORT 1 (P1)
5	Internal housing of Preset / Jog buttons
L1	LED 1 Link / Activity in PORT 1
L2	LED 2 Link / Activity in PORT 2
L3	LED 3 Active errors / faults information

L4	LED 4 Fieldbus interface status information
L5	LED 5 Controller power supply information
L6	LED 6 Motor power supply information

4.1 Ground connection (Figure 5)

To minimize noise connect properly the frame to ground; we suggest using the ground screw provided in the frame (see the Figure above). Connect properly the cable shield to ground on user's side. Lika 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)

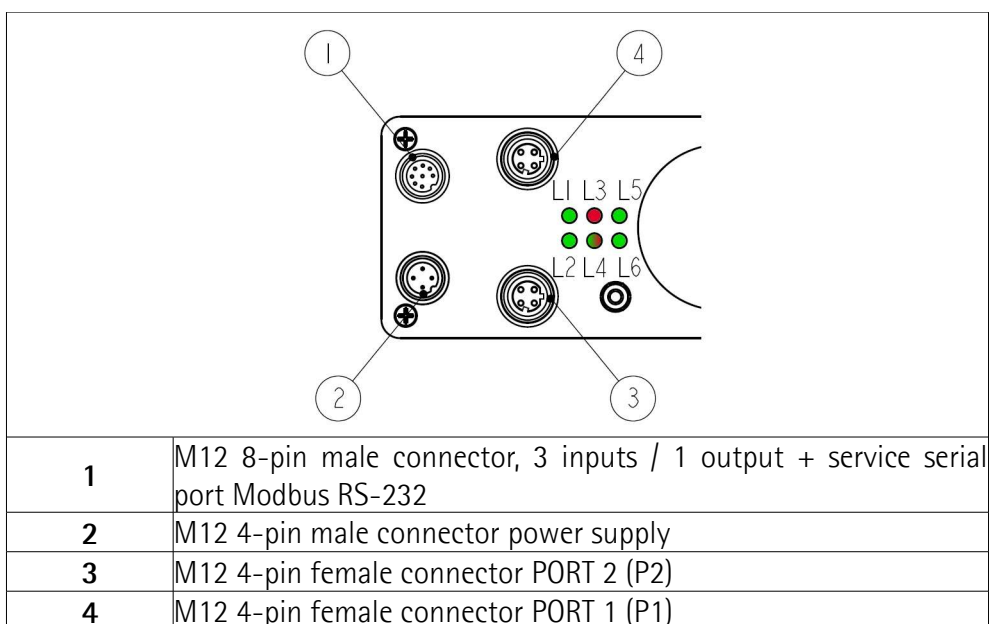


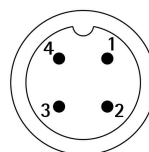
Figure 5: Connectors

4.2.1 Power supply connector

Power supply

M18 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 POWERLINK interface connectors (P1 PORT 1 and P2 PORT 2)

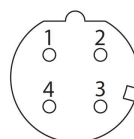
Two M12 4-pin female connectors with D coding are used for Ethernet connection through PORT 1 and PORT 2.

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 ¹

¹ Lika EC- pre-assembled cables only

The Ethernet interface supports 100 Mbit/s, half duplex operation.

P1 PORT 1 **4** and P2 PORT 2 **3** M12 connectors have pin-out in compliance with the POWERLINK standard. Therefore you can use standard POWERLINK cables commercially available.

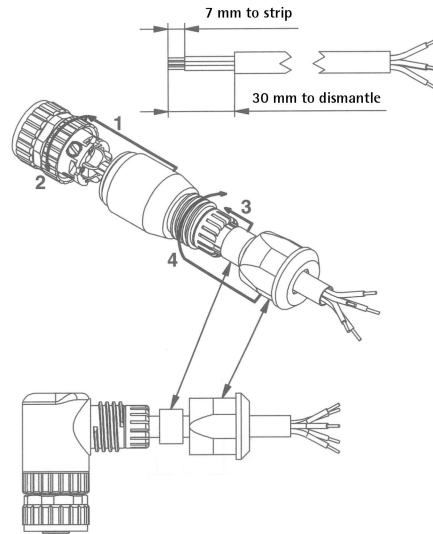
P1 PORT 1 **4** and P2 PORT 2 **3** connectors are interchangeable.



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 Figure below). If metal connectors are used, connect the cable shield properly as recommended by the manufacturer.



4.2.3 Network configuration: cables, hubs, switches - Recommendations

Cables and connectors comply with the POWERLINK specifications. Cables are CAT-5 shielded cables.

Standard POWERLINK cables commercially available can be used.

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, 328 ft) predefined by Ethernet 100Base-TX must be compulsorily fulfilled.

EPL recommends the use of hubs to fit POWERLINK jitter requirements.

Switches may be used to build a POWERLINK network.

It has to be considered that any POWERLINK network constructed with anything but Class 2 Repeater Devices does not conform to the POWERLINK standard.

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 Addressing

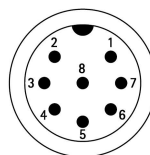
For complete information please refer to the "4.4.1 Node address (Node ID): DIP A (Figure 8)" section on page 45.

4.2.5 Line Termination

POWERLINK network needs no line termination because the line is terminated automatically; in fact every Slave is able to detect the presence of the downstream Slaves.

4.2.6 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 54.

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

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

The MODBUS address is according to the rotary switch setting. To set the node address of the RD1xA device refer to the "4.4.1 Node address (Node ID): DIP A (Figure 8)" section on page 45. See also the "8.2 "Serial configuration" page" section on page 145.

For any further information on configuring and using the RS-232 service serial port refer to the "Modbus® interface" section on page 143.

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 POWERLINK and the MODBUS interfaces and the device. The meaning of each LED is explained in the following tables.

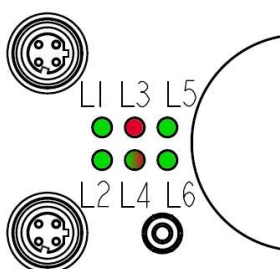


Figure 6: Diagnostic LEDs

L1	L/A Link/Activity in P1 PORT 1	L4	MS Module Status LED
L2	L/A Link/Activity in P2 PORT 2	L5	Controller power supply information
L3	NS Network State Error LED	L6	Motor power supply information


NOTE

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

ETHERNET
POWERLINK

LED L1 GREEN	Description
L/A Link/Activity LED for port 1 P1 (green)	It shows the state and the activity of the physical link (port 1 P1).
BLINKING	Activity on port 1 P1.
ON	Port 1 P1 link active, no activity.

LED L2 GREEN	Description
L/A Link/Activity LED for port 2 P2 (green)	It shows the state and the activity of the physical link (port 2 P2).
BLINKING	Activity on port 2 P2.
ON	Port 2 P2 link active, no activity.

LED L3 RED	Description
NS Network State Error LED (red)	It shows the current state of the network.
OFF	No error is currently active.
ON	<ul style="list-style-type: none"> If the MS Status LED is green: a non fatal error has occurred. If the MS Status LED is red: a fatal error has occurred.

LED L4 GREEN/RED	Description
MS Module Status LED (green / red)	It shows the state of the POWERLINK device.
OFF	The device is switched OFF; the device is initializing; the device is not active.
FLICKERING green (50ms ON, 50ms OFF)	The network communication is in Basic Ethernet Mode , see NMT_CS_BASIC_ETHERNET , no POWERLINK traffic, see on page 91.
SINGLE FLASH green	The device is in Pre-Operational 1 state, see NMT_CS_PRE_OPERATIONAL_1 , see on page 89, asynchronous communication and no PDO exchange.
DOUBLE FLASH green	The device is in Pre-Operational 2 state, see NMT_CS_PRE_OPERATIONAL_2 , see on page 89, asynchronous and synchronous communication and no PDO exchange.
TRIPLE FLASH green	The device is in Ready to Operate state, see NMT_CS_READY_TO_OPERATE , see on page 89, there is no PDO exchange.
ON green	The device is in Operational state, see NMT_CS_OPERATIONAL , see on page 90, there is PDO exchange.
BLINCKING green (200ms ON, 200ms OFF)	The device is in Stopped state (for example, due to a controlled shutdown), see NMT_CS_STOPPED , see on page 90, there is no PDO exchange.
ON red	If the NS Network State Error LED is also lit, a fatal error has occurred.

LED L5 GREEN	Description
	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 the enabling state of the motor
ON	It indicates that the motor is enabled (control loop activated)
OFF	It indicates that the motor is disabled (control loop deactivated)

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

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



WARNING

The power supply must be turned off before setting the rotary switches!



WARNING

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



NOTE

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

To access the rotary switches and 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.

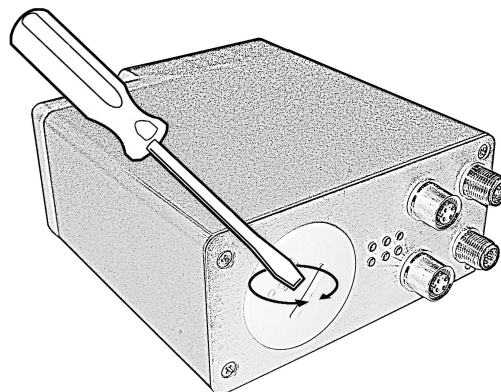


Figure 7: Screw plug for internal access

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

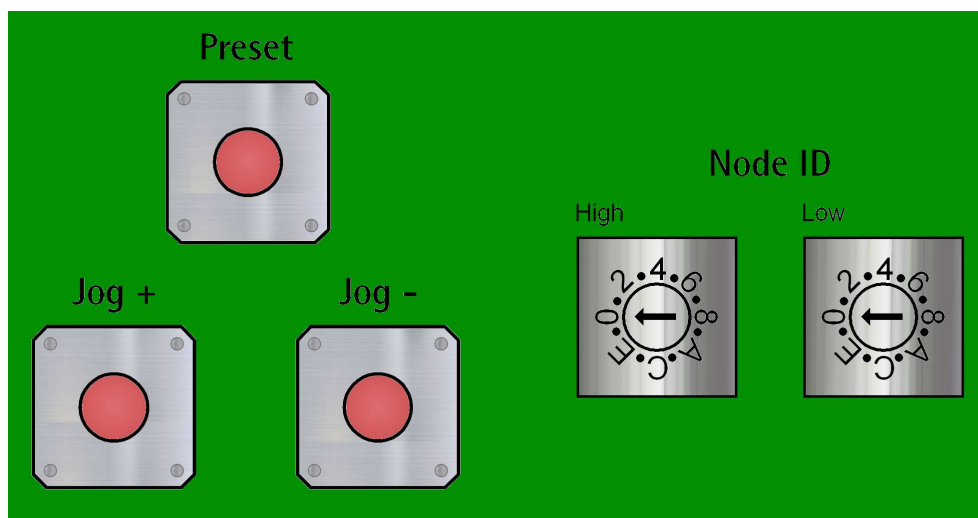


Figure 8: Preset / Jog buttons and rotary switches

4.4.1 Node address (Node ID): DIP A (Figure 8)



WARNING

Power supply must be turned off before performing this operation!
The POWERLINK Node ID cannot be set via software.

The private class C Net ID **192.168.100.0** is used in the POWERLINK network. A class C network provides 254 (1-254) IP addresses which match the number of valid POWERLINK Node IDs. As they are CNs, Lika actuators accept values from 1 to 239, in compliance with the POWERLINK specifications. Address 0 is invalid, addresses from 240 to 255 are reserved to MNs or special functions. The POWERLINK MN (Active MN) is addressed to 240. The Host ID of the private class C Net ID 192.168.100.0 is identical to the POWERLINK Node ID. Hence the last byte of the IP address (Host ID) has the same value as the POWERLINK Node ID. The following figure illustrates the construction of the IP address.

192.168.100.	POWERLINK Node
Net ID	Host ID

The following table summarises the default IP parameters.

IP Parameter	IP address
IP address	192.168.100.<POWERLINK Node ID>
Subnet mask	255.255.255.0
Default Gateway	192.168.100.254 (it may be modified)

The POWERLINK node ID is set via hardware using the DIP A rotary switches located inside the enclosure. To access the DIP A rotary switches please refer to the "4.4 Screw plug for internal access (Figure 4 and Figure 7)" section on page 44.

Allowed node addresses range between 1 and 239. Address 0 is invalid. The default value is 1.

If you set an invalid address or any value greater than 239 the address will be forced to the default value.

The node address which is set using this selector affects both the POWERLINK and the MODBUS interfaces.

The node address which is currently set in the actuator can be read next to the **210E-00 Node ID** object.

Set the node address (POWERLINK node ID) expressed in hexadecimal notation.



EXAMPLE

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

4.4.2 MAC address and IP address

The unit can be identified in the network through the **MAC address** and the **IP address**.

The 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 actuator.

The IP address must be assigned by the user to each interface of the unit to be connected in the network, while the subnet mask is always 255.255.255.0 as in a class C net.

For additional information on the MAC address refer to the "7.3 MAC address" section on page 59.

For additional information on the IP address refer to the "4.4.1 Node address (Node ID): DIP A (Figure 8)" section on page 45.

4.5 Preset / Jog buttons (Figure 9)

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

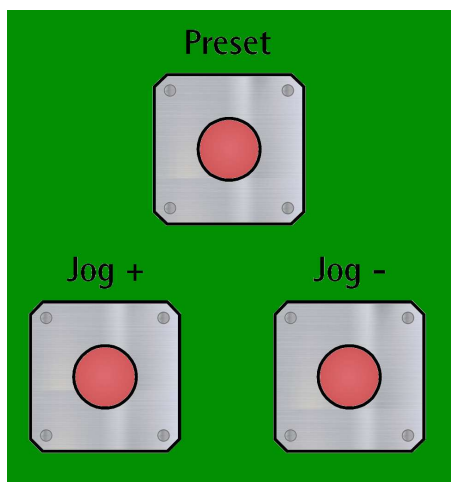


Figure 9: Preset / Jog buttons

4.5.1 JOG + and JOG - buttons (Figure 9)

Press the JOG + / JOG - 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 123 ff (POWERLINK interface) or on page 183 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 **2200-00 Control Word** on page 124 -POWERLINK version; **Incremental jog** in **Control Word [0x2A]** on page 184 -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 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 9)

This button is meant to assign the value set next to the **Preset** item to the current position of the axis. The button must 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 136 (POWERLINK interface) or on page 181 (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 given 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;
- if required, set the node address (node ID; see on page 45); the value set by Lika Electronic at factory set-up is "1";
- 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 122 -POWERLINK interface; see on page 183 -MODBUS interface); reset the active alarms: switch high ("=1") the **Alarm reset** bit 3 of the **Control Word** (see on page 122 -POWERLINK interface; see on page 183 -MODBUS interface). Check the operating condition shown through the LEDs;
- set a proper value next to the **Distance per revolution-pulse / Distance per revolution** item (see on page 131 -POWERLINK interface; see on page 176 -MODBUS interface);
- set a proper value next to the **Jog speed-rpm / Jog speed** item (see on page 135 -POWERLINK interface; see on page 180 -MODBUS interface);
- set a proper value next to the **Work speed-rpm / Work speed** item (see on page 135 -POWERLINK interface; see on page 180 -MODBUS interface);
- if required, set a proper value next to the **Preset-pulse / Preset** item (see on page 136 -POWERLINK interface; see on page 181 -MODBUS interface) and activate the preset by switching high ("=1") the **Setting the preset** bit 11 of the **Control Word** (see on page 122 -POWERLINK interface; see on page 183 -MODBUS interface);
- set the limit switch values next to the **Max delta pos-pulse / Positive delta** and **Max delta neg-pulse / Negative delta** items (see on page 133 ff -POWERLINK interface; see on page 178 ff -MODBUS interface);
- set the commanded position next to the **Target position** item (see on page 126 -POWERLINK interface; see on page 188 -MODBUS interface);
- save the new setting values (**Save parameters** command; see on page 125 -POWERLINK interface; see on page 186 -MODBUS interface).

Use the **Jog+**, **Jog-**, **Start** and **Stop** commands in the **Control Word** (see on page 122 -POWERLINK interface; see on page 183 -MODBUS interface) to move the axis, check its operation and reach the commanded position.

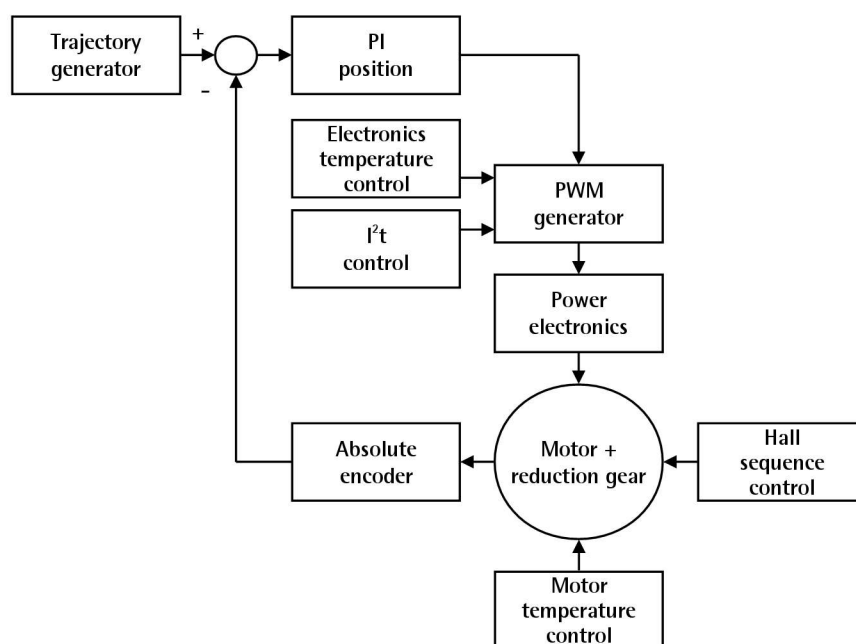
**NOTE**

The parameters **Distance per revolution-pulse / Distance per revolution**, **Position tolerance / Position window**, **Max following error-pulse / Max following error**, **Max delta pos-pulse / Positive delta**, **Max delta neg-pulse / Negative delta**, **Jog step-pulse / Jog step length** and **Preset-pulse / Preset** 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 55.

6 Functions

6.1 Working principle

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



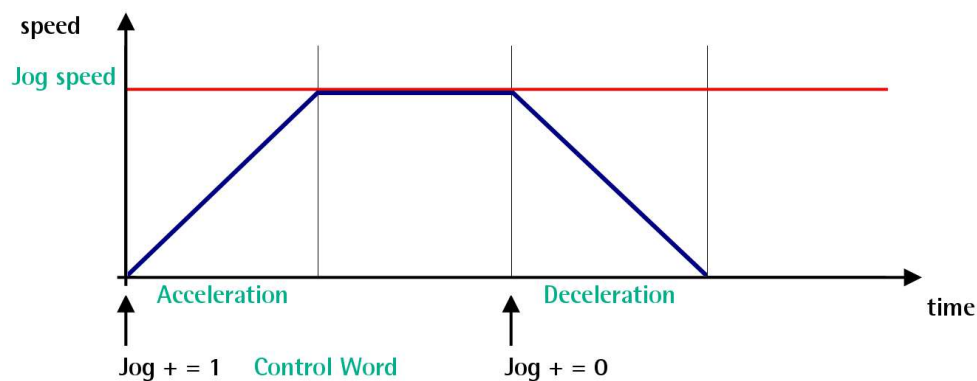
6.2 Movements: jog and positioning

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 **220E-00 Jog speed-rpm / Jog speed [0x0D]** parameter.



When the bit 0 **Jog +** in the **2200-00 Control Word / Control Word [0x2A]** is "1", the motor accelerates toward the positive direction according to the value set next to the **220A-00 Acceleration- rev/s^2 / Acceleration [0x07]** item; if the available travel is long enough it reaches the speed set next to the **220E-00 Jog speed-rpm / Jog speed [0x0D]** item. As soon as the bit 0 **Jog +** in the **2200-00 Control Word / Control Word [0x2A]** goes low ("0"), the motor decelerates according to the value set next to the **220B-00 Deceleration- rev/s^2 / Deceleration [0x08]** item until it stops.

Setting the bit 1 **Jog -** in the **2200-00 Control Word / Control Word [0x2A]** to "1" causes the motor to run in the opposite direction (negative direction) respecting the work phases already described above.

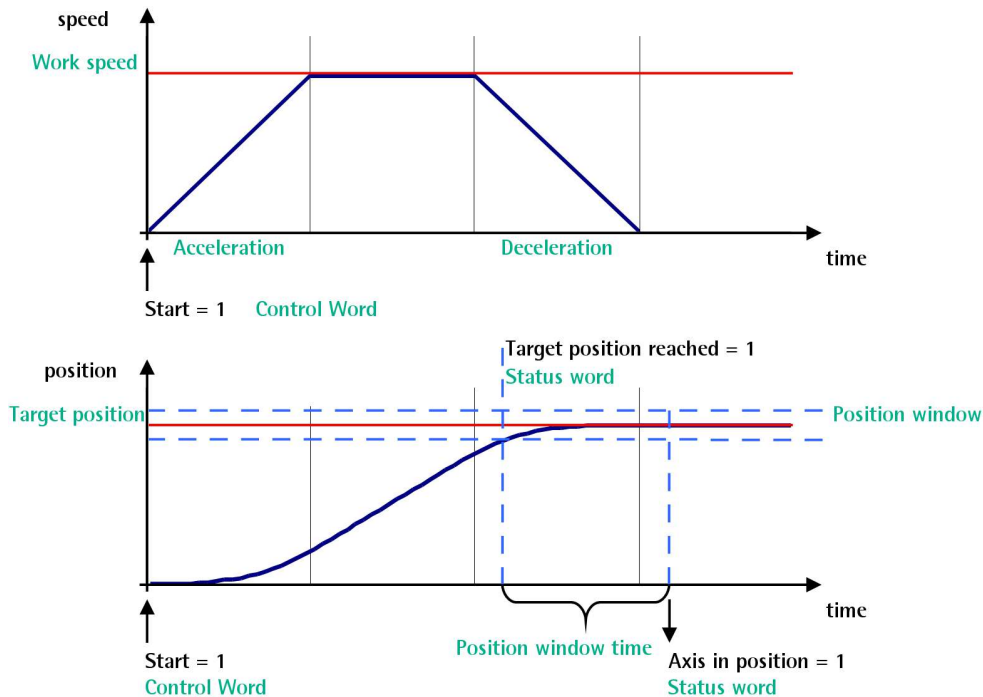


NOTE

Please note that the value in the **220E-00 Jog speed-rpm / Jog speed [0x0D]** parameter is the speed of the motor, not the speed of the output shaft after the reduction gears.

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 **220F-00 Work speed-rpm / 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 **2200-00 Control Word / Control Word [0x2A]** is "1", the motor starts moving and accelerates according to the value set next to the **220A-00 Acceleration-rev/s² / Acceleration [0x07]** item in order to reach the target position as set next to the **2201-00 Target position / Target position [0x2B-0x2C]** item. If the available travel is long enough it reaches the speed set next to the **220F-00 Work speed-rpm / 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 **2205-00 Position tolerance / Position window [0x01]** item, the bit 8 **Target position reached** in the **2202-00 Status word / Status word [0x01]** goes high ("1"). When the position is within the tolerance window limits set next to the **2205-00 Position tolerance / Position window [0x01]** item, after the delay set next to the **2206-00 Settling time-ms / Position window time [0x02]** item, the bit 0 **Axis in position** in the **2202-00 Status word / Status word [0x01]** goes high ("1"). The motor decelerates according to the value set next to the **220B-00 Deceleration-rev/s² /**

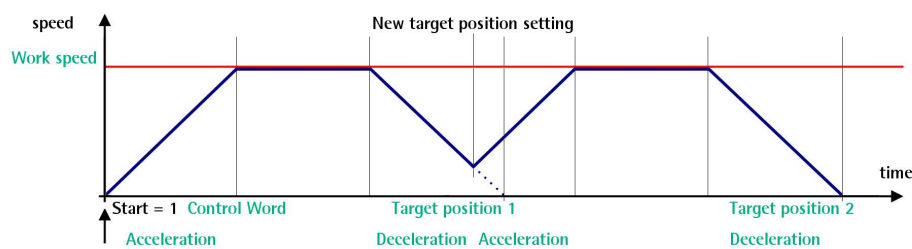
Deceleration [0x08] item in order to reach the halt position according to the set target position.



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 **2201-00 Target position / Target position [0x2B-0x2C]** item.



NOTE

Please note that the value in the **220F-00 Work speed-rpm / Work speed [0x0E]** parameter is the speed of the motor, not the speed of the output shaft after the reduction gears.

6.3 Digital inputs and output

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

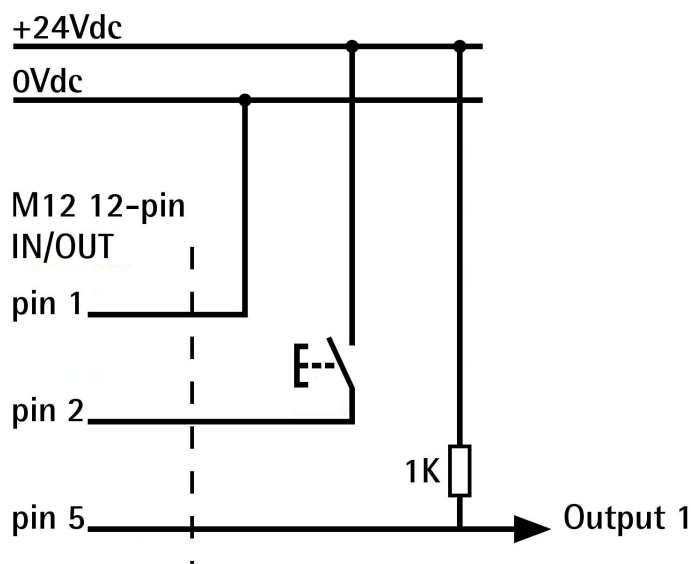
Inputs are read by the Slave device and transmitted to the Master through the **Status word** (bits 13-15; see on page 130 ff -POWERLINK interface- or page 195 ff -Modbus interface) when the device is running in **NMT_CS_OPERATIONAL** state -POWERLINK interface- / **Idle** state -Modbus interface.

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

The Slave **output 1** is operated by the Master through the **Control word** (bit 13; see on page 126 -POWERLINK interface- or page 187 -Modbus interface) when the device is running in **NMT_CS_OPERATIONAL** state -POWERLINK 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, Positive delta and Negative delta

The variables **Distance per revolution**, **Preset**, **Positive delta** and **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 131 -POWERLINK interface; see on page 176 -MODBUS interface);
- set a proper value next to the **Jog speed** item (see on page 135 -POWERLINK interface; see on page 180 -MODBUS interface);
- set a proper value next to the **Work speed** item (see on page 135 -POWERLINK interface; see on page 180 -MODBUS interface);
- set a proper value next to the **Preset** item (see on page 136 -POWERLINK interface; see on page 181 -MODBUS interface) and activate the preset by switching high ("=1") the **Setting the preset** bit 11 of the **Control Word** (see on page 122 -POWERLINK interface; see on page 183 -MODBUS interface);
- check the value next to the **Positive delta** item (see on page 133 -POWERLINK interface; see on page 178 -MODBUS interface);

- check the value next to the **Negative delta** item (see on page 134 -POWERLINK interface; see on page 179 -MODBUS interface);
- save the new values (**Save parameters** command, bit 9 in the **2200-00 Control Word / Control Word [0x2A]** item, see on page 125 -POWERLINK interface; see on page 186 -MODBUS interface).

Each time you change the value in **Distance per revolution** then you must update the **Preset** value 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 **Positive delta** and **Negative delta**.

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

The value set next to the **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 **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 window**, **Max following error**, **Positive delta**, **Negative delta**, **Target position** and **Position following error**. The **Current position** value is also affected.



EXAMPLE 1

Default values:

Distance per revolution = 1,024 steps per revolution

Max. **Work speed**: 2,000 rpm

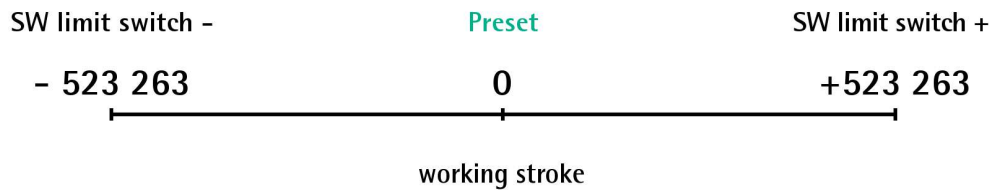
Preset = 0

Positive delta and **Negative delta** max. values = $523\ 263 = (1,024 \text{ steps per revolution} * 512 \text{ revolutions}) - 1 \text{ step} - 1,024 \text{ steps}$ (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 ($100 * 3000 / 1024 = 293$)

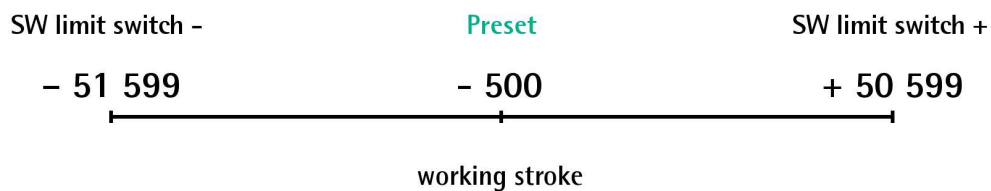
Preset = -500 (ex. thickness of the tool)

Positive delta and **Negative delta** = (100 steps per revolution * 512 revolutions) - 1 step - 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 POWERLINK interface

7.1 Before starting

Lika rotary actuators are **CN devices** (Controlled Nodes e.g. Slaves: nodes without the ability to manage the SCNM mechanism, see on page 85) and comply with the "EPSG Draft Standard 301 Ethernet POWERLINK Communication Profile Specification Version 1.2.0" as well as with the CANopen Profiles "DS301 CANopen Application Layer and Communication Profile" according to the POWERLINK specifications. They require an **MN device** installed in the POWERLINK network (Managing Node, e.g. a Master: a node capable to manage the SCNM mechanism, see on page 84).

7.1.1 Network identity

Lika POWERLINK rotary actuators use the following identity settings:

Identity Name: **Vendor ID**

Object: **1018-01 NMT_IdentityObject_REC.VendorID_U32**

Setting: **0000 0012Eh** Lika Electronic's CANopen vendor ID

Identity Name: **Device Type**

Object: **1000-00 NMT_DeviceType_U32**

Setting: **0000 012Dh** RD rotary actuators series

Identity Name: **Product Code**

Object: **1018-02 NMT_IdentityObject_REC.ProductCode_U32**

Setting: **0000 2010h** RD1xA series

Identity Name: **Manufacturer Device Name**

Object: **1008-00 NMT_ManufactDevName_VS**

Setting: **RD1xA-PL-xxx** RD1xA POWERLINK series

7.1.2 Network and communication settings

The **MAC address** of the device is reported in the label applied to the actuator enclosure. See below in this page.

The **POWERLINK Node ID** is set via hardware using the DIP A rotary switch located inside the actuator enclosure. The default value is 1. See on page 45.

7.2 Configuring the actuator with Automation Studio V. 4.3 from B&R

In this manual some screenshots are shown to explain how to install and configure the rotary actuator in a supervisor. In the specific example the development environment is Automation Studio V. 4.3.3.195 from B&R; it is used in combination with the X20CP1584 PLC from B&R. Therefore, the information on the installation of the XDD file, the assignment of the IP address and the device name, the configuration of the actuator in the network, topology, diagnostics, etc. will always refer to the aforementioned development tools. 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.



Lika Electronic POWERLINK rotary actuator documentation is complete with an **example project** supplied free of charge. This program is designed to make your own project planning, programming, communication and diagnostics with the Automation

Studio V. 4.3 development environment user-friendly and reliable. It allows to set the Preset value and execute it (refer to page 75 and following). You can find it in the **SW_RD1A_PL_Example.zip** compressed file.

7.3 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 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		

The MAC address can be read next to the **1030-05 NMT_InterfaceGroup_0h_REC.InterfacePhysAddress_OSTR** object.

7.4 Installing the actuator under Automation Studio environment

7.4.1 Description of the XDD file

The functionality of a POWERLINK device is always described in a XDD file (XML Device Description file). The Device Description File provides information about the device basic communication and functional properties. It must be installed in the MN device.

The file name is primarily built up as follows:

0xVendorID_ProductName.xdd

e.g. 0000012E_Lika_RD1xA_PL_EthernetPOWERLINK

POWERLINK actuators from Lika Electronic are supplied with their own XDD file. It is:

- **0000012E_Lika_RD1xA_PL_EthernetPOWERLINK.xdd**: it is intended for installation of **RD1xA series rotary actuators** ("0000012E_Lika" shows the Vendor ID -expressed in hexadecimal notation- and name; "RD1xA" is the actuator series; "PL" is the abbreviation for POWERLINK).

XDD files are paired with the **RD1xA.bmp** picture file available inside the file folder.

Follow the path **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS/POSITIONING UNITS (DRIVECOD)** to download the XDD files from Lika's corporate web site.

7.4.2 Installing the XDD file

In the menu bar of the main page, press **Tools** and then the **Import Fieldbus Device ...** command.

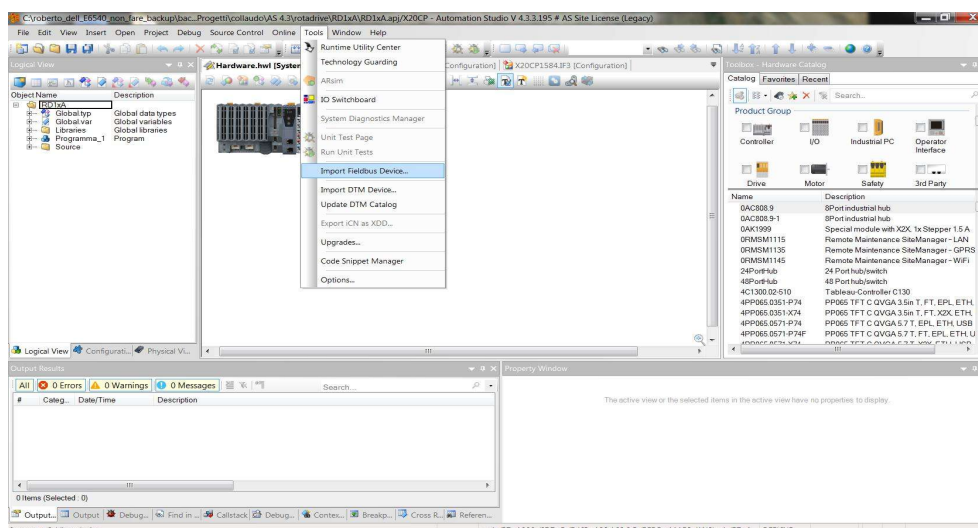


Figure 1 - Installing the XDD file

The **Select fieldbus file(s) for import** dialog box will appear. Browse through the folders and search for the directory where the XDD file is located. Please make sure that the RD1xA.bmp bitmap file representing the actuator is located in the same folder as the XDD file. Select the file corresponding to the actuator to be installed (if several files are available, please check the order code) and press the **Open** button to install it.

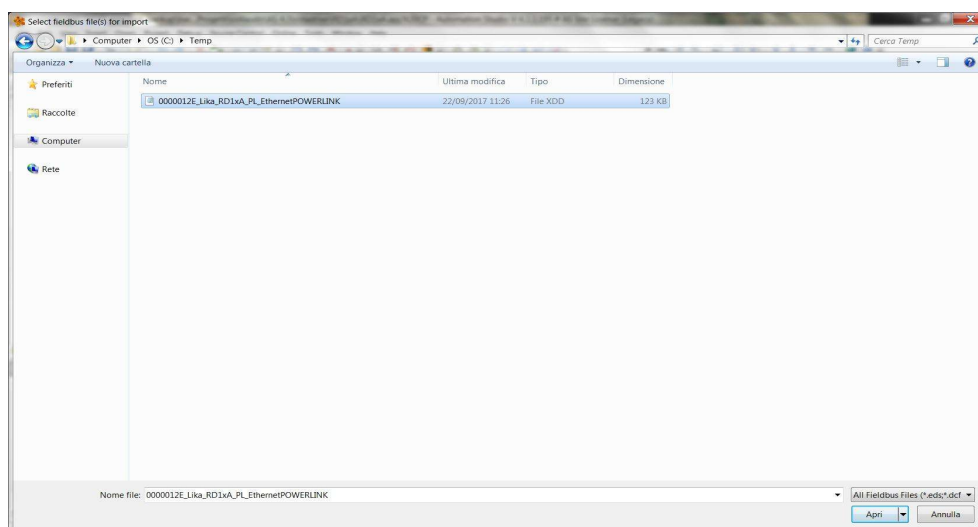


Figure 2 - Selecting the XDD file

As soon as the operation is carried out, a confirmation message will appear in the **Output Results** window.

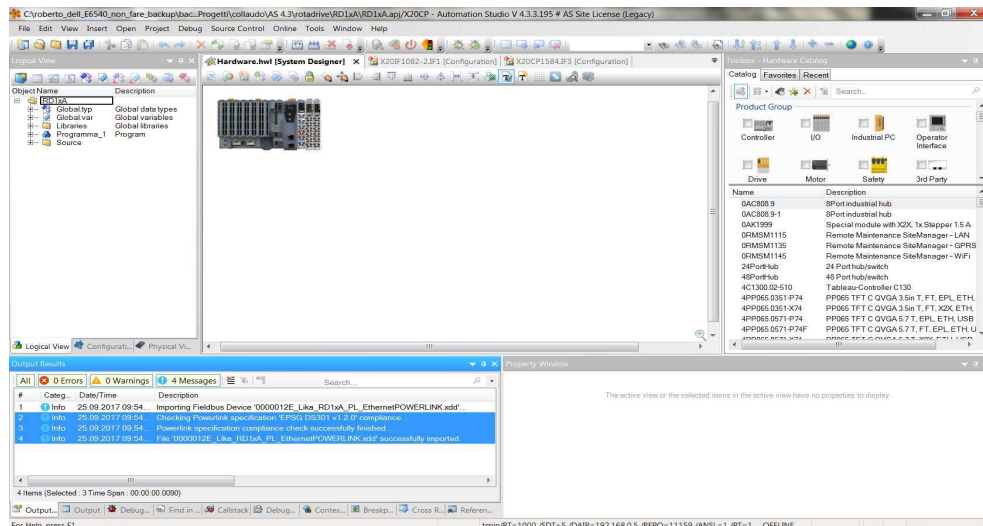


Figure 3 - XDD file installation

The installed device will be listed in the **Toolbox – Hardware Catalog** window (at the top right corner in the snapshot above, see Figure 3).



NOTE

To display the **Hardware.hwl [System Designer]** window in the main page, double click the **Hardware.hwl** item in the **Configuration View** window. To display the **Toolbox – Hardware Catalog** window in the main page, right click in the **Hardware.hwl [System Designer]** window and press the **Add Hardware Module...** command.

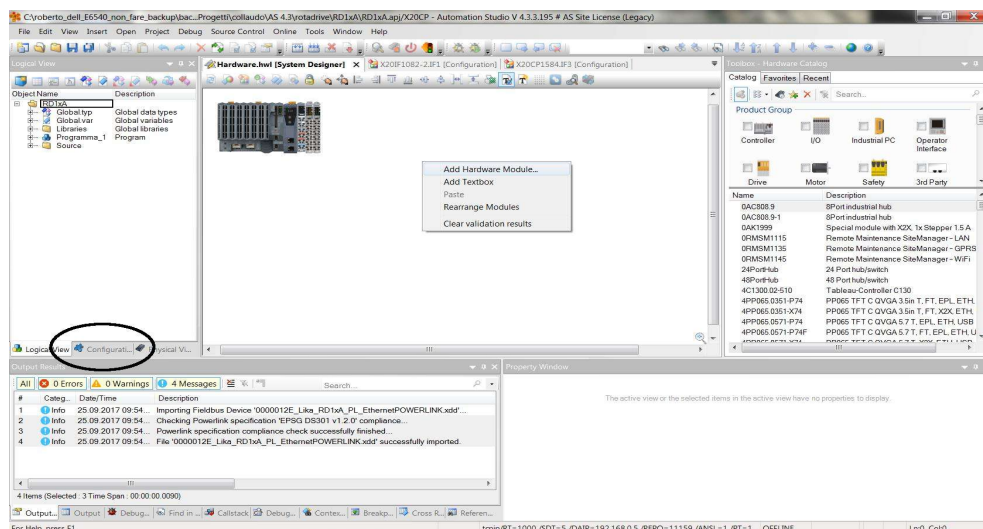


Figure 4 - Adding a hardware module

Now you need to link the installed actuator (that is: the module) to the PLC.
There are two ways.

1. In the **Toolbox – Hardware Catalog** window, enter the name of the installed device (or just "Lika") in the **Search** box; all Lika modules that have been installed will be listed in the **Model Number** window.
2. Or check the **Powerlink** option in the **Network Type** section of the **Toolbox – Hardware Catalog** window and then scroll through the list of the installed POWERLINK devices in the **Model Number** window.

Select the required Lika CN device in the **Model Number** window ("Lika Electronic RD1xA Ethernet POWERLINK" module); a preview will appear in the **Properties** window.

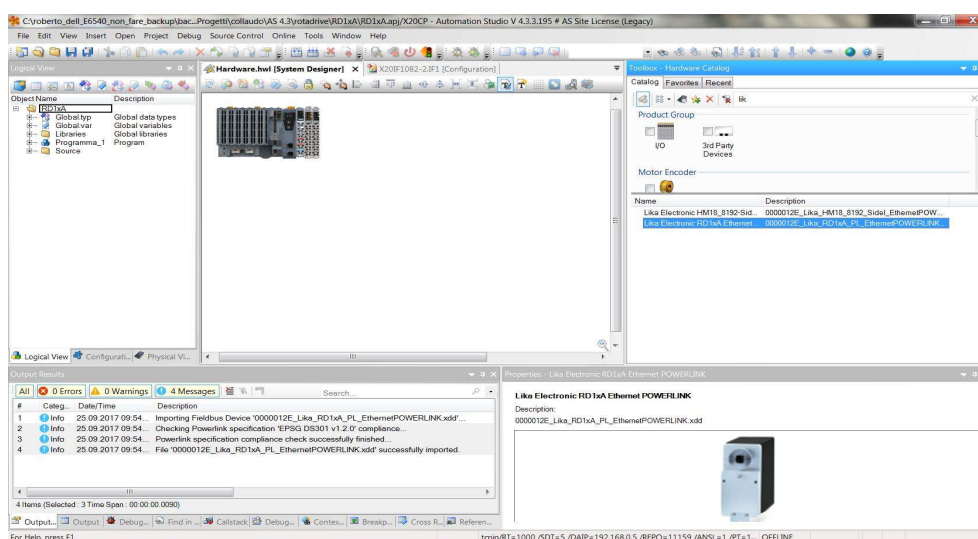


Figure 5 - Adding a module

Drag the module to the **Hardware.hwl [System Designer]** window and drop it to the desired position.

The Lika device icon will appear in the **Hardware.hwl [System Designer]** window.

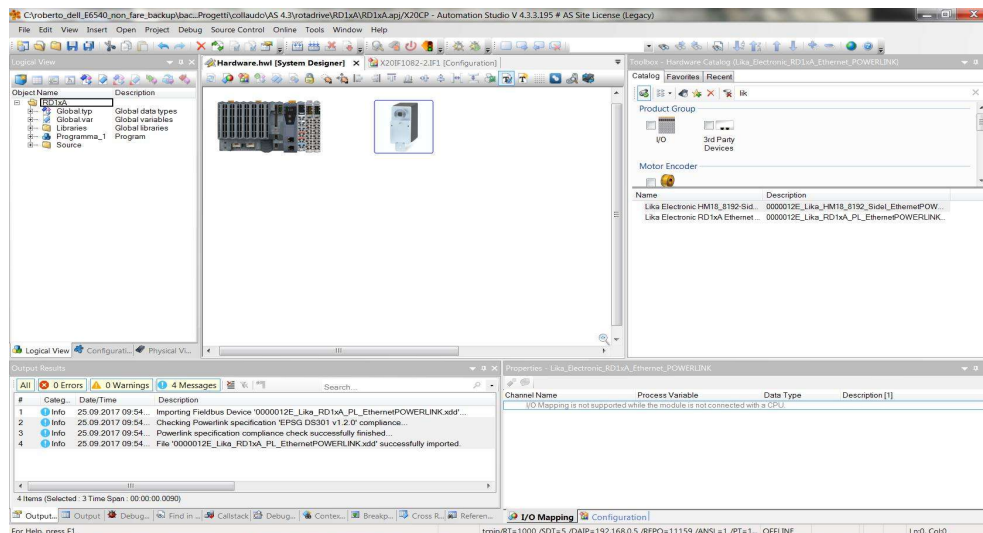


Figure 6 - Installed module

To link the MN device to the CN device, move the cursor over the MN icon; a small circle will appear on the icon; drag it (a blue line will appear) and drop it onto the small circle that is shown when you move the cursor over the CN icon, as shown in the snapshot below, see Figure 7.

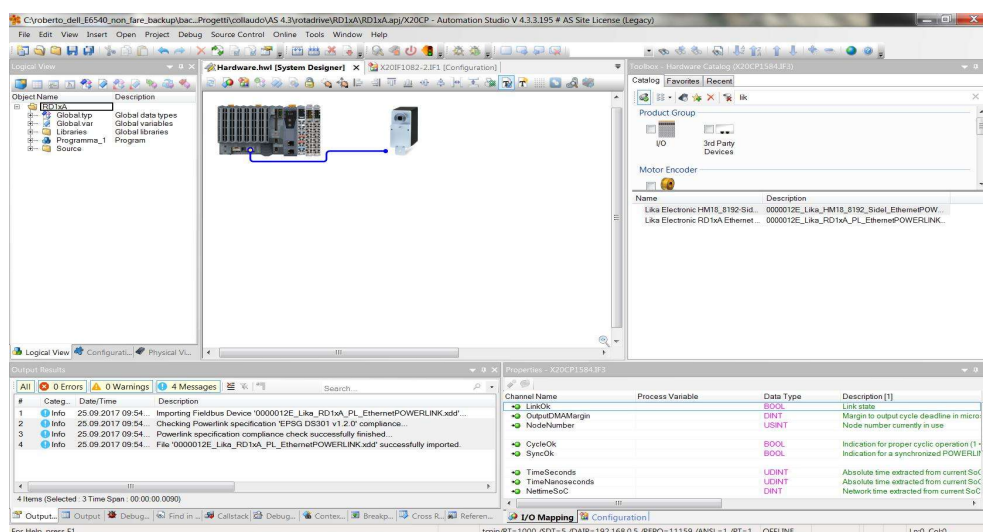


Figure 7 - Adding the module to the network

7.4.3 Setting the device node address in the project

You are required to enter in the project the node address that has been set physically in the installed CN device. For information on the POWERLINK Node ID setting refer to the "4.4.1 Node address (Node ID): DIP A (Figure 8)" section on page 45.

Enter the **Physical View** window by pressing the **Physical View** tab.

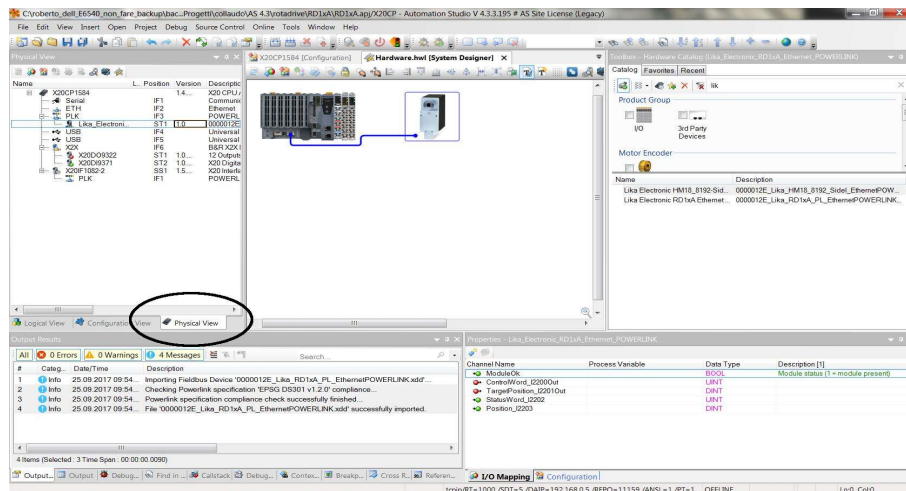


Figure 8 - Physical View window

Scroll through the tree of the installed devices, extend the PLK (POWERLINK) group and select the module (for instance: **Lika_Electronic_RD1xA...**). Right-click the **Lika_Electronic_RD1xA...** module in the list and then press the **Node Number - Change Node Number** command.

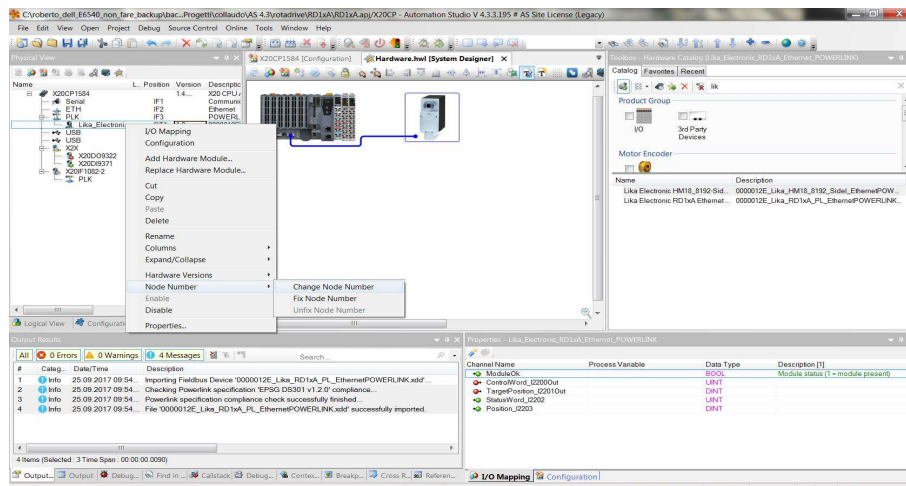


Figure 9 - Setting the node address

Double click the field in the **Position** column and set the required address value in the box.

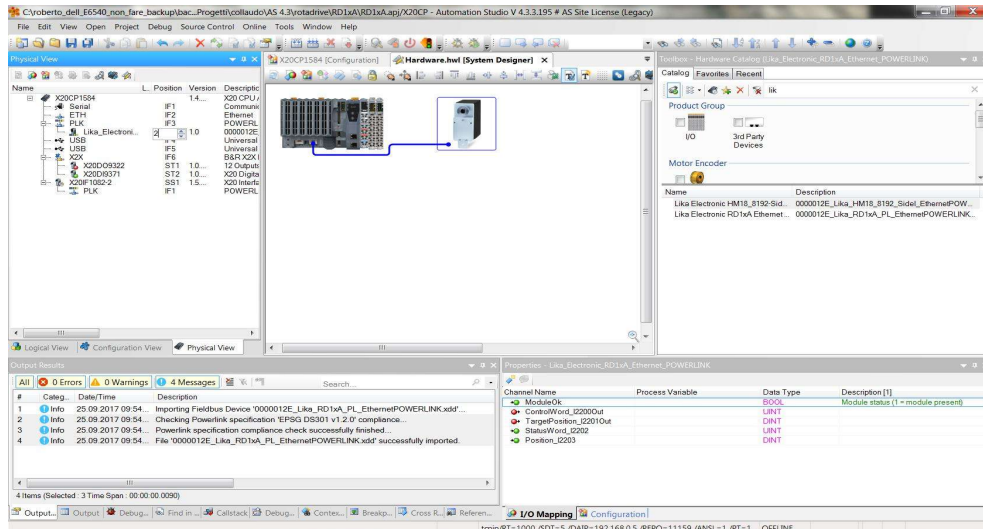


Figure 10 - Setting the node address

7.4.4 Configuring the CN device

The Configuration objects are grouped in the **Object Dictionary** (refer also to page 94).

The Object Dictionary is the most important part of a device profile. It is essentially a grouping of objects accessible via the network in an ordered, pre-defined fashion. Each object within the dictionary is addressed using a 16-bit index.

The Object Dictionary can contain a maximum of 65,536 entries.

The user-related objects are grouped in three main areas: the Communication Profile Area, the Manufacturer Specific Profile Area and the Standardised Device Profile Area. The objects are described in the XDD file.

The **Communication Profile Area** at indexes from 1000h to 1FFFh contains the communication specific parameters for the POWERLINK network. These entries are common to all POWERLINK devices. The Communication Profile Area objects comply with the "CiA Draft Standard Proposal 301 CANopen Application layer and communication profile". Refer to the "7.16.1 Communication Profile Area objects (DS 301)" section on page 96.

The **Manufacturer Specific Profile Area** at indexes from 2000h to 5FFFh is free to add manufacturer-specific functionality. The objects that are specifically intended to be used for configuring Lika's actuators can be found in this group.

Refer to the "7.16.2 Manufacturer Specific Profile Area objects" section on page 117.

The **Standardised Device Profile Area** at indexes from 6000h to 9FFFh contains all data objects common to a class of devices that can be read or written via the network. RD1xA rotary actuators have no parameters in this profile area.

Right click the Lika actuator icon and press the **Configuration** button to enter the actuator's **Configuration** window.

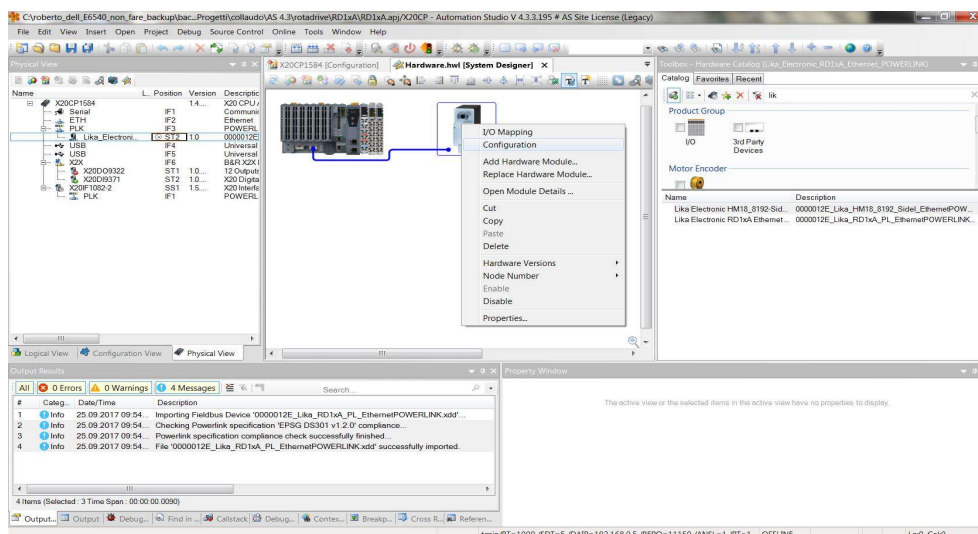


Figure 11 - Entering the Configuration window

In the **Configuration** window the implemented objects are listed.

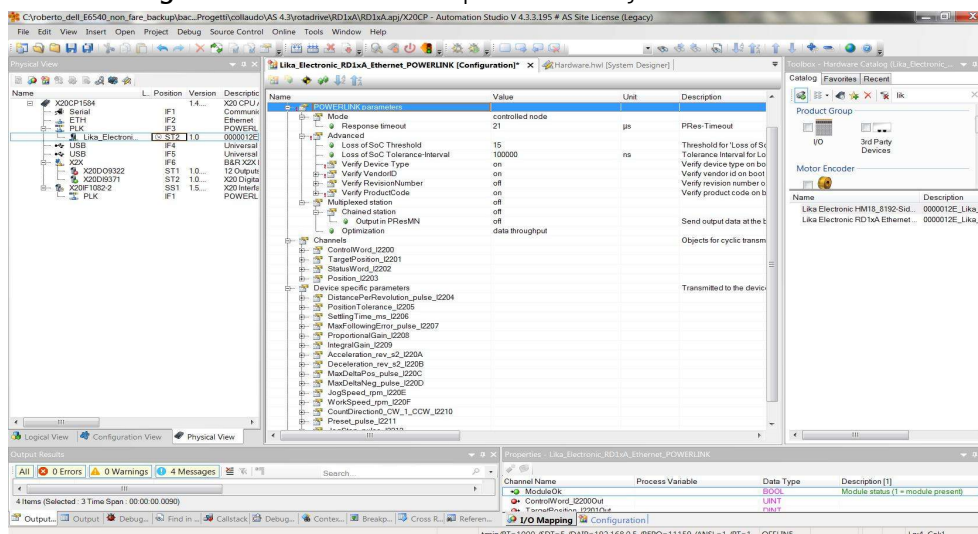


Figure 12 - Configuration window

Under the **POWERLINK Parameters** group the Communication Profile Area objects are found.

Under the **Channels** group the **2200-00 Control Word**, the **2201-00 Target position**, the **2202-00 Status word** and the **2203-00 Position** entries can be found.

Under the **Device Specific Parameters** group the Manufacturer Specific Profile Area objects are found.

In the **POWERLINK Parameters** group we suggest setting the following items to ON:

- **Verify Device Type**: the Device Type compliance is checked on boot;
- **Verify VendorID**: the VendorID compliance is checked on boot;
- **Verify ProductCode**: the Product Code compliance is checked on boot.

This is useful to avoid mismatch errors. See the Figure 13.

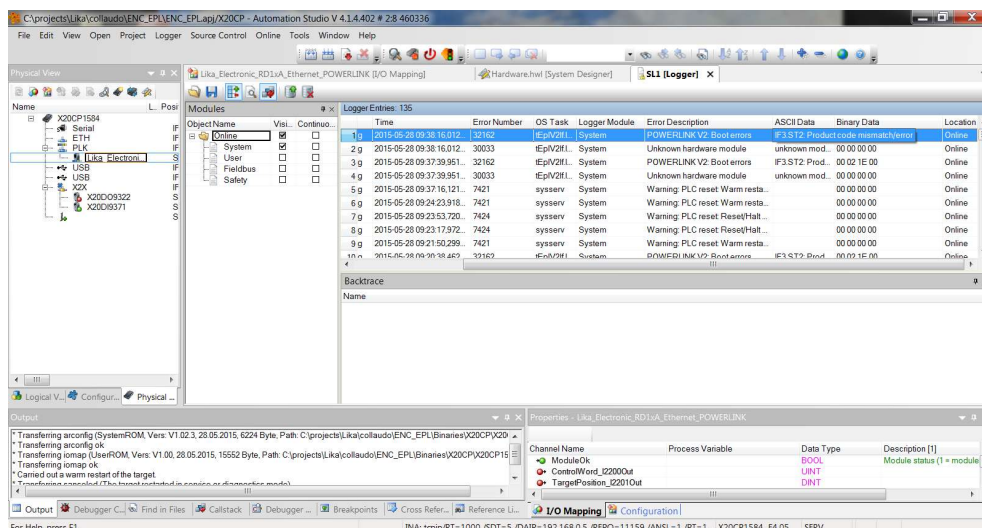


Figure 13 - Mismatch error

7.4.5. Downloading the parameters to the actuator

After having set the parameters in the **Configuration** window, you must download the new values to the actuator to make them effective.

To do this you must press **Project** in the menu bar of the main page and then the **Build Configuration** command.

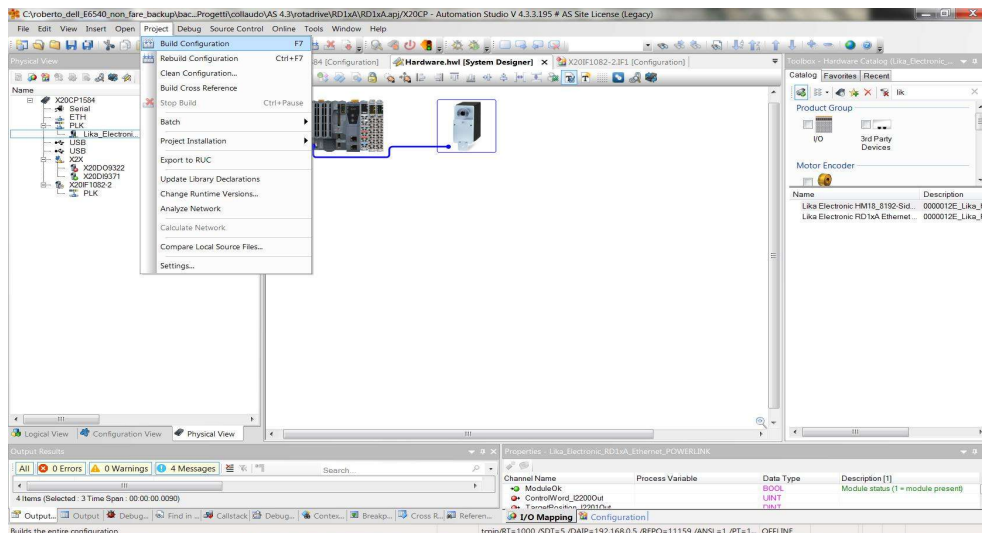


Figure 14 - Building the project

At the end of the process the **Project Build** message will appear on the display. Press the **Transfer** button to start the download of the project.

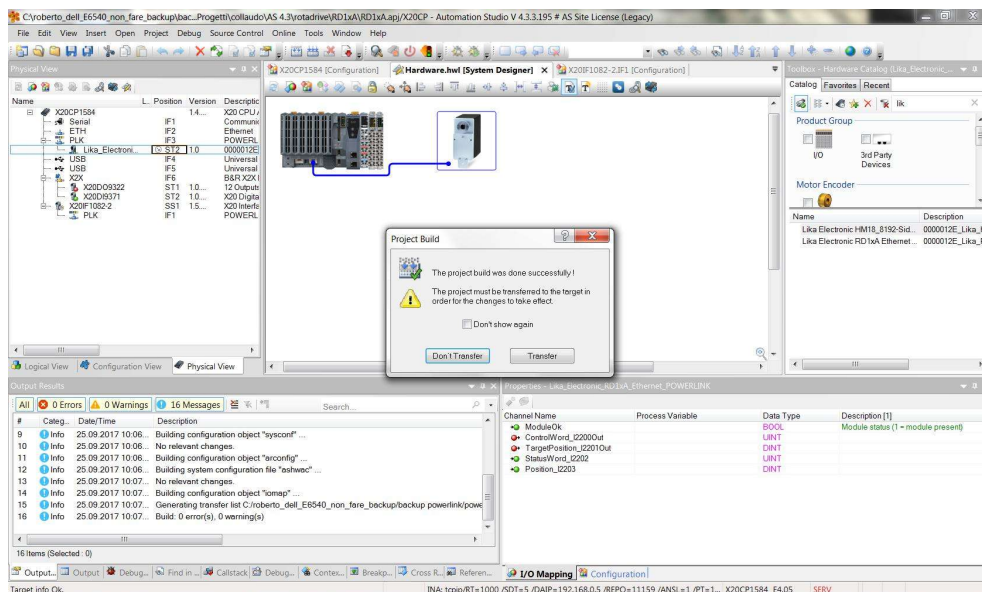


Figure 15 - Transferring the project

As soon as the download is complete, the **Transfer Project** message will appear on the display.

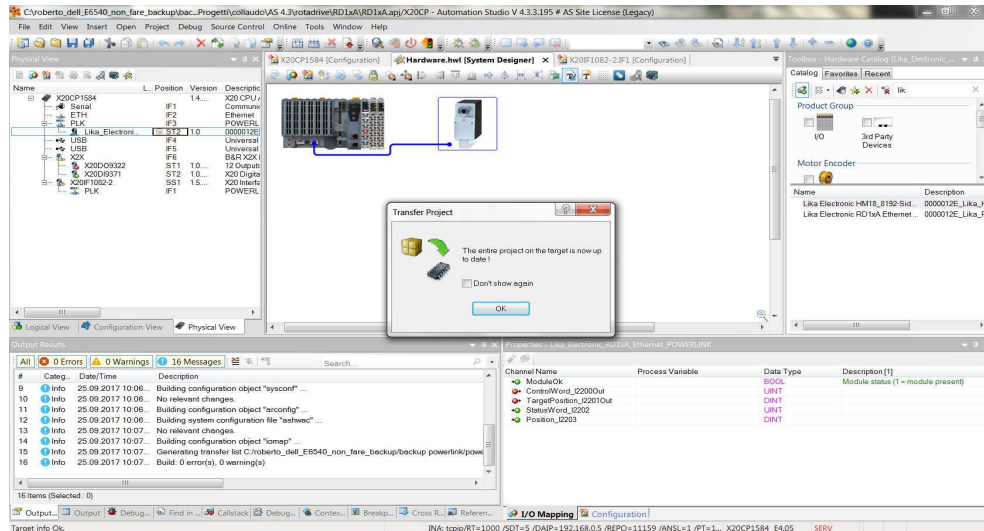


Figure 16 - Download completed

7.4.6 Parameter values download at PLC start



NOTE

Please consider attentively this PLC behaviour.

During the initialisation process at the PLC start, B&R PLC downloads to the actuator the parameter values that have been set next to the entries in the **Device Specific Parameters** group.

All values that are set in the **Init value** line of each parameter are sent to the control node and overwrite the values currently saved.

To avoid transferring a value and overwriting the parameter stored in the actuator memory, the **Init value** line under the entry in the **Device Specific Parameters** group has to be kept blank.

In the example shown in Figure 17, the **2211-00 Preset-pulse** next to the **Init value** line is blank, thus it is not transferred to the actuator during the initialisation process at the PLC start. The preset stored in the actuator memory is preserved.

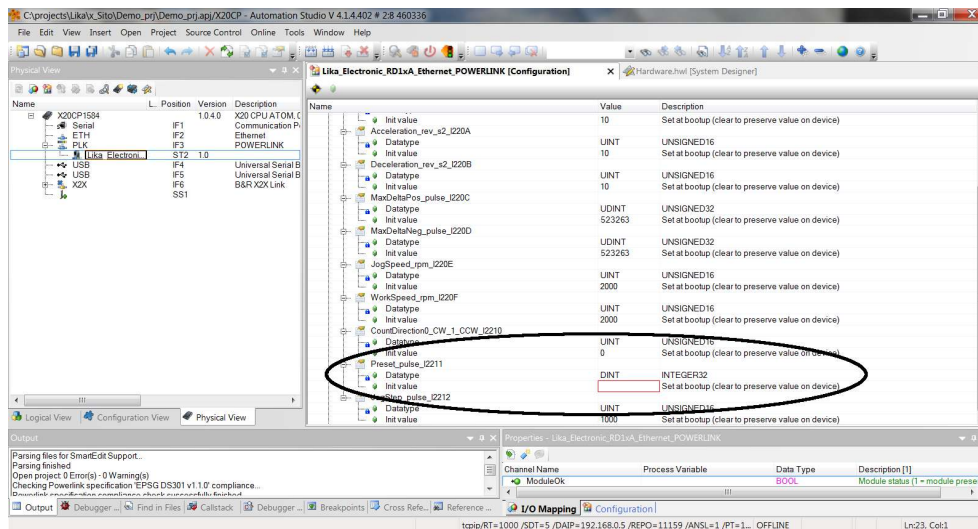


Figure 17 – Parameter values download at PLC start

7.4.7 Monitoring input and output values

To monitor the input and output values of the actuator, you must enter the **I/O Mapping** window first. Right click the Lika actuator icon and press the **I/O Mapping** button to enter the **I/O Mapping** window.

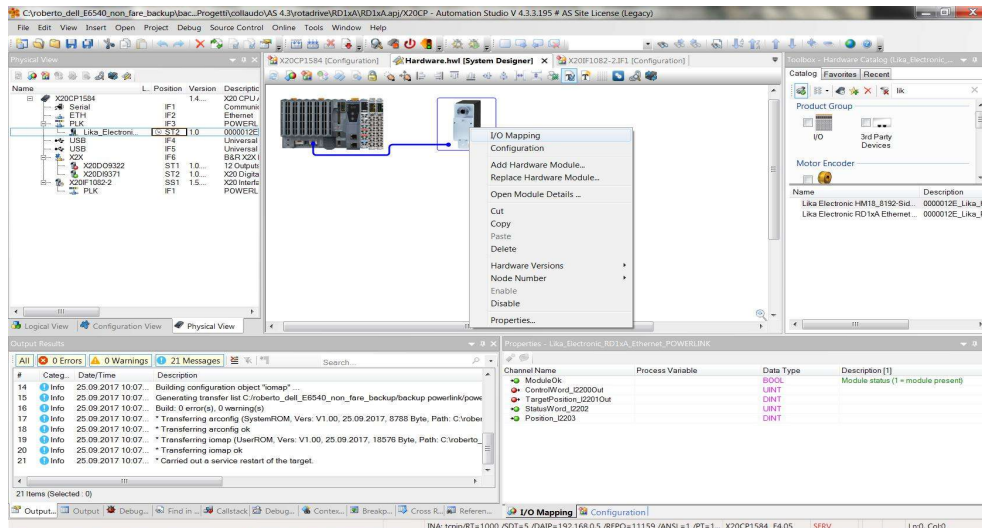


Figure 18 - Entering the I/O Mapping window

You can display the **I/O Mapping** window also by scrolling through the tree of the installed devices in the **Physical View** window, extending the **PLK (POWERLINK)** group and selecting the Lika module (for instance: **Lika_Electronic_RD1xA...**). Right-click the **Lika_Electronic_RD1xA...** module in the list and then press the **I/O Mapping** command.

Now press the **Monitor** button in the Toolbar to activate the Monitor mode and start monitoring the system.

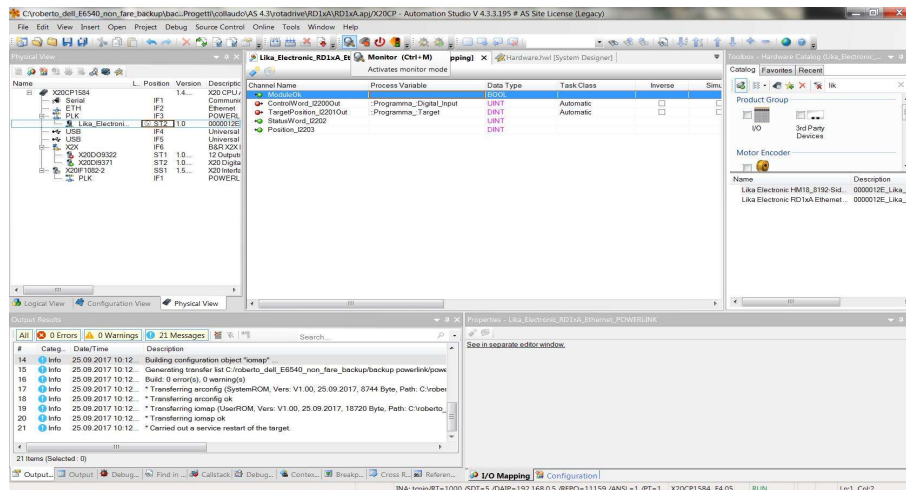


Figure 19 - Monitoring the device

As soon as the system is in Monitor mode, the background of the windows becomes grey.

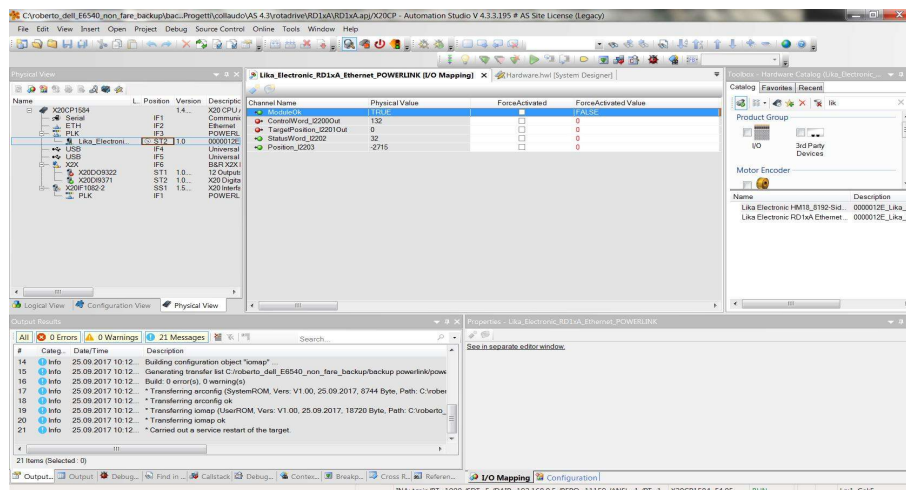


Figure 20 - Monitoring the device

The objects **2200-00 Control Word**, **2201-00 Target position**, **2202-00 Status word** and **2203-00 Position** are displayed in the **I/O Mapping** window and can be monitored.

7.4.8 Setting the cycle time

In a POWERLINK network, the managing node allocates data transfer time for data from each node in a cyclic manner within a guaranteed cycle time. The Cycle Time is the time between two consecutive Start of Cyclic (SoC) frames – i.e. repeating – process. The Cycle Time includes the time for data transmission and some idle time before the beginning of the next cycle. See the [1006-00 NMT_CycleLen_U32](#) object on page 96.

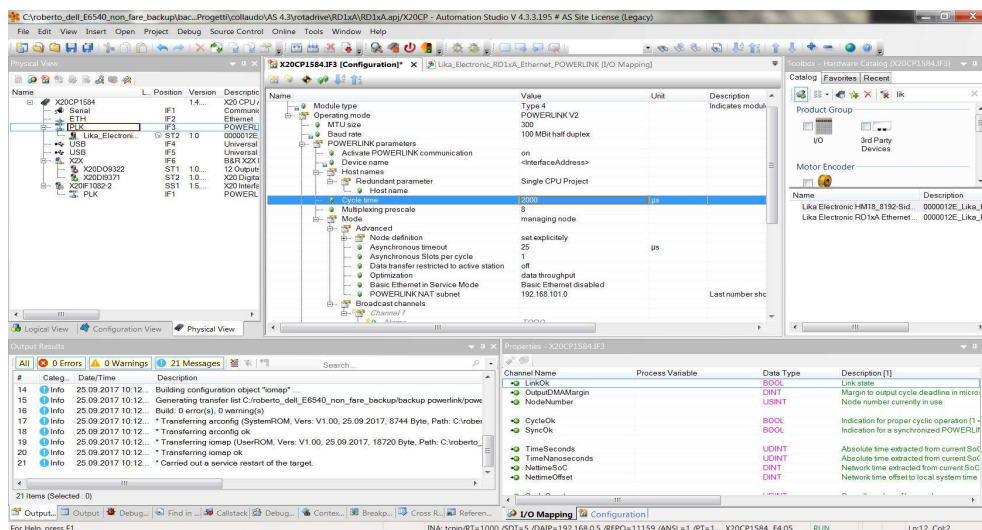


Figure 21 - Setting the cycle time

7.4.9 Preset setting program

As previously stated (see the "7.4.6 Parameter values download at PLC start" section on page 71), during the initialisation process at the PLC start, B&R PLC downloads to the actuator the parameter values that have been set next to the entries in the **Device Specific Parameters** group.

If any value is set in the **Init value** line of the **Preset_pulse_I2211** parameter, it is sent to the control node at each power-on.

At power-on the preset value, if it is set in the **Init value** line, is sent but the preset operation is not executed as the actuator is not in operational state (**NMT_CS_OPERATIONAL**) yet.

To execute the preset the value must be transmitted to the **2211-00 Preset-pulse** object in the asynchronous phase via SDO when the actuator is in operational state (**NMT_CS_OPERATIONAL**) and then you must execute the **Save parameters** function (see the **Save parameters** function on page 125).

As an alternative you can execute the preset by pressing the PRESET button, it is located beneath the screw plug in the enclosure of the unit, see on page 44. The PRESET button is meant to assign the value set next to the **2211-00 Preset-pulse** item to the current position of the axis. The button must be kept pressed for 3 seconds at least. We suggest setting the preset when the actuator is in stop. For more information please refer to the "4.5.2 PRESET button (Figure 9)" section on page 48.

You can set and execute the preset also by using the Preset sample task provided by Lika Electronic (AsEPL Sample Task using the AsEPL library available in Automation Studio).

To open the Preset sample task you must enter the **Logical View** window by pressing the **Logical View** tab.

Scroll through the tree of the available EPLs, extend the **Source** directory and select the **Preset** AsEPL Sample Task. Right-click the **Preset** AsEPL Sample Task in the list, press the **Open** command first and then the **Watch** command.

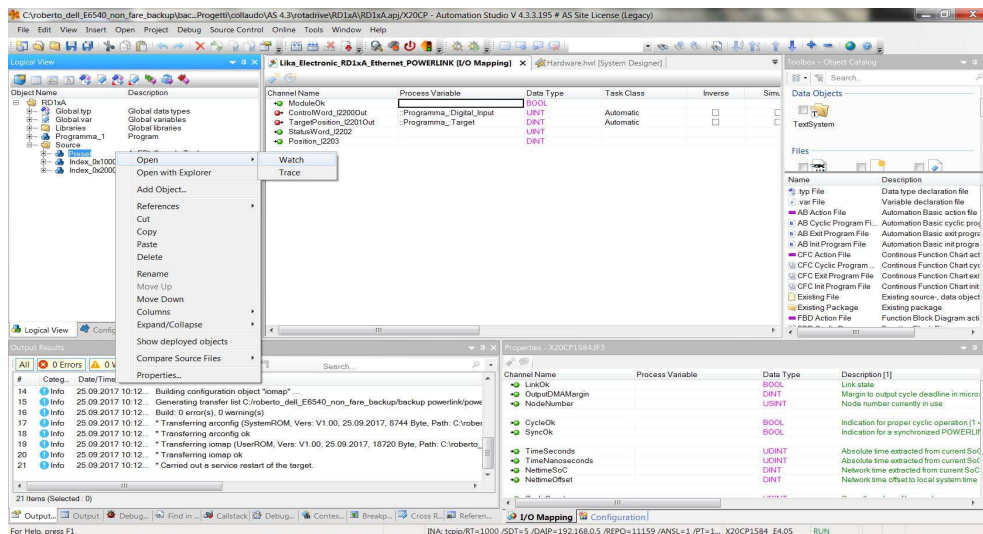


Figure 22 - Opening the Preset AsEPL Sample Task

The **Preset.pvm [Watch]** window will be displayed.

Activate the Monitor mode by pressing the **Monitor** button in the Toolbar.

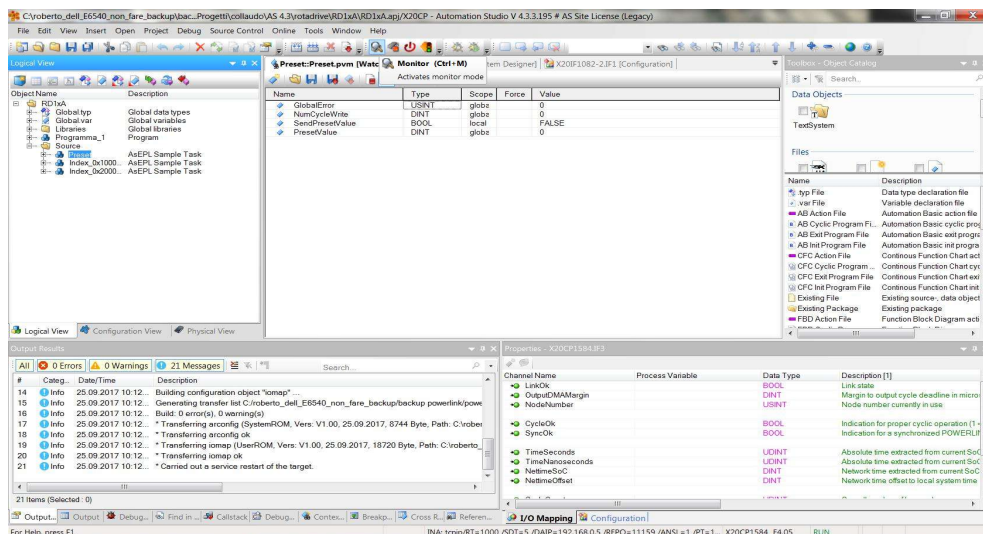


Figure 23 - Preset.pvm [Watch] window

The background of the windows becomes grey.

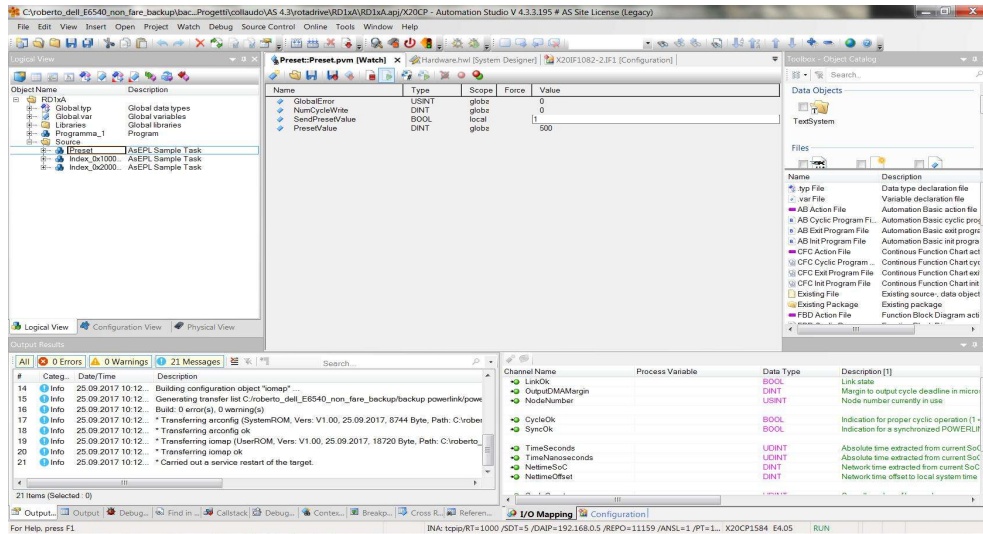


Figure 24 - Activating the Monitor mode

Enter the preset value next to the **PresetValue** entry ("500" in the sample shown in the snapshot, see Figure 24). Set the value "1" (TRUE) next to the **SendPresetValue** entry (see Figure 24) and then set it to "0" (FALSE) again.

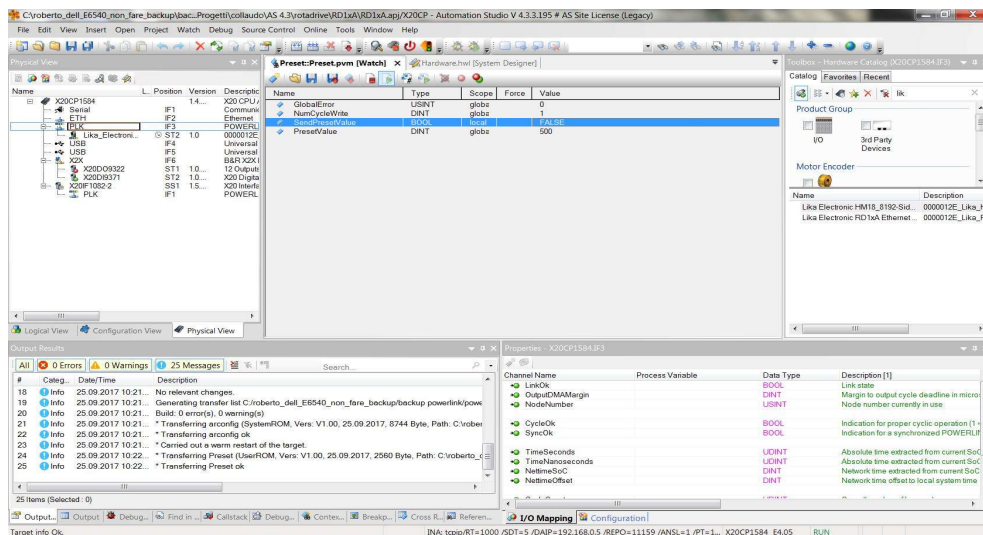


Figure 25 - Preset activated

You can enter the **I/O Mapping** window (see the "7.4.7 Monitoring input and output values" section on page 72) and check that the **Position_I2203** entry is "500".

After setting the preset, you are required to save the parameters in order to store on memory the preset value. If you do not save the new preset value, at next power-on the system will calculate the actuator position using the preset value previously saved. To store the parameters you must set the bit 9 **Save parameters** in the **2200-00 Control Word** object in the asynchronous phase via SDO when the actuator is in operational state (**NMT_CS_OPERATIONAL**). As soon as the new preset value is entered, the calculated offset (**2105-00 Position Offset**) is automatically stored on memory.

7.4.10 Entering the System Diagnostics Manager (SDM)

Automation Studio provides the user with a wide variety of diagnostic tools for commissioning applications and searching for errors:

- System Diagnostics Manager (SDM);
- Status bar;
- Logger Monitor of the software and hardware configuration, see on page 81;
- ...

These tools range from simple monitoring of operating states (variable and program status, I/O, etc.), forcing I/O channels, and tracing variable states over time to profiling the entire runtime behaviour, debugging operations for programs and libraries, and simulating and commissioning axes.

The **System Diagnostics Manager (SDM)** is a diagnostic tool used to diagnose the system by means of a standard web browser from any location (Intranet or Internet).

It only requires a web browser and a TCP/IP connection to the controller.

It can be used for:

- hardware analysis for detection of configuration or hardware problems on the target system;
- analysis of system configuration and runtime parameters (e.g. configured IP address, etc.);
- software analysis (software modules and versions on the target system);
- display and upload of error logbook for target system.

The user profits not just from being able to access information about system hardware and software from anywhere in the world, but also from ready-made diagnostic applets that can be easily integrated directly into applications.

Since all service functions necessary for a machine or system are already integrated in the System Diagnostics Manager, it can be launched from any PC

without any installation whatsoever. The only requirement is a conventional web browser.

To enter the System Diagnostics Manager, press **Tools** in the menu bar of the main page and then the **System Diagnostics Manager** command.

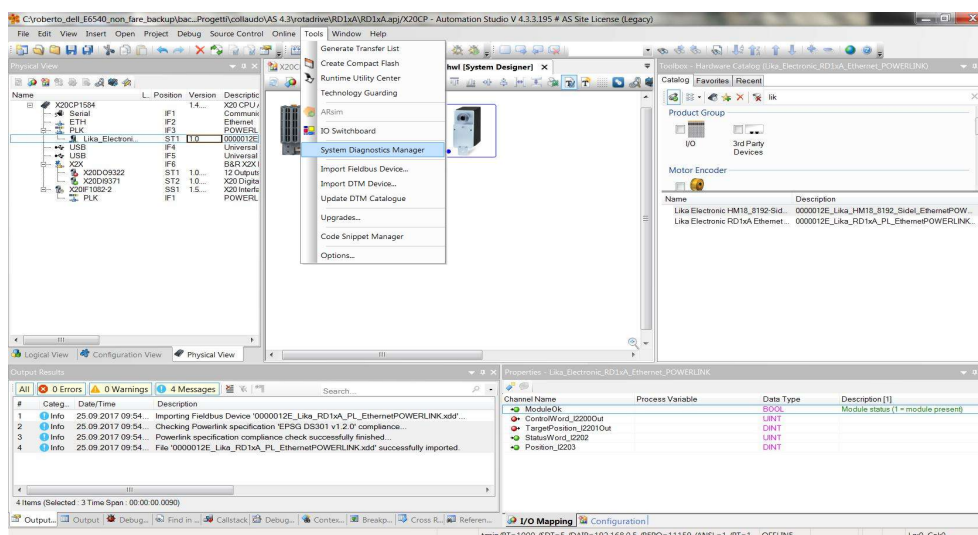


Figure 26 – Entering the System Diagnostics Manager

The first page of B&R System Diagnostics Manager will appear in your predefined web browser.

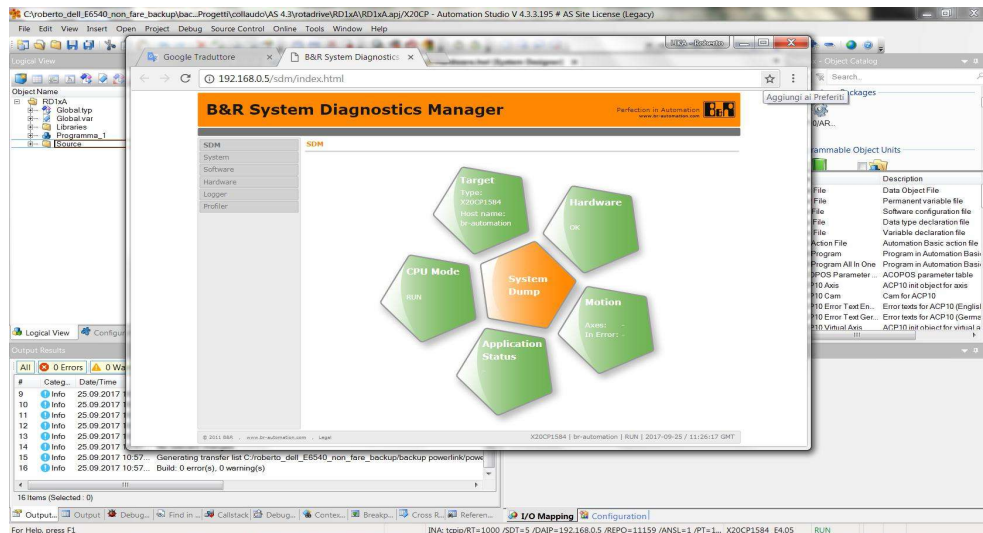


Figure 27 - System Diagnostics Manager

Press either the buttons in the left navigation bar or the graphic figures to enter the specific pages.

Press the **Hardware** button in the left navigation bar to collect some diagnostic information on the installed Lika module.

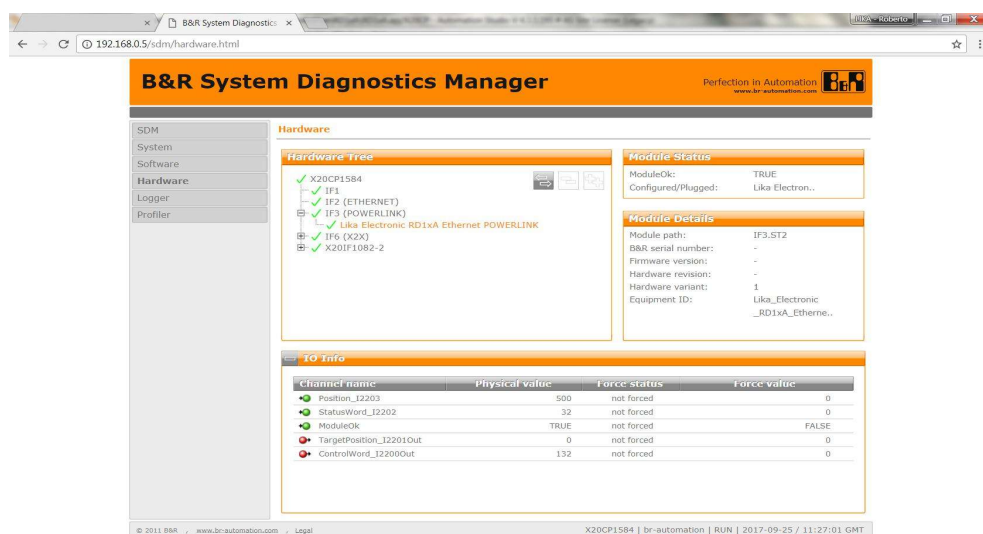


Figure 28 - Hardware diagnostic page

7.4.11 Logger Monitor

Among the diagnostic tools provided by Automation Studio is the Logger Monitor.

It is used to record system information, the system messages are automatically entered in the list in the **SL1 [Logger]** window.

To enter the Logger Monitor window, press **Open** in the menu bar of the main page and then the **Logger** command.

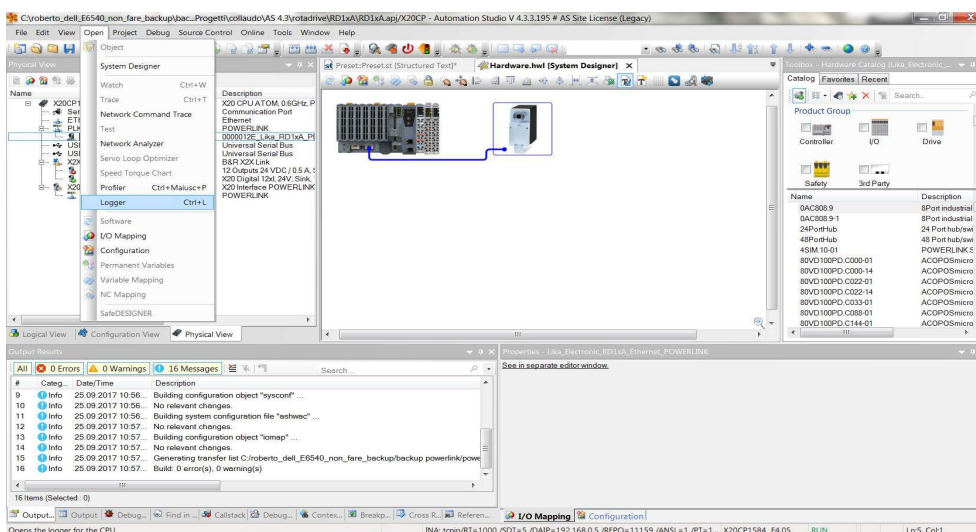


Figure 29 - Entering the Logger Monitor

The **SL1 [Logger]** window will be displayed.

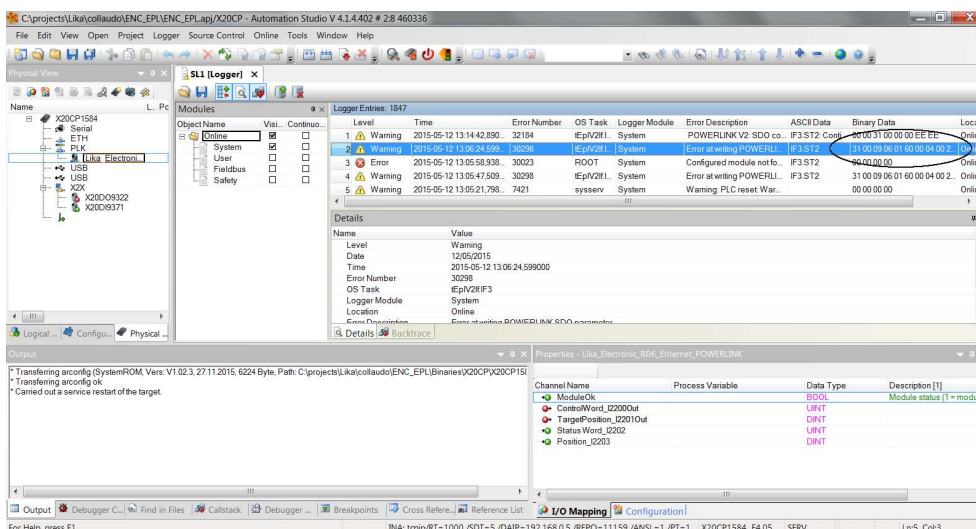


Figure 30 - Logger entries

In the highlighted logger entry the SDO abort code is shown in the Binary Data column (6090031h abort code: "Value of parameter written too high", for the complete list of the SDO abort codes see on page 139).

7.5 Overview

Ethernet POWERLINK (EPL) is a communication profile for Real-Time Ethernet (RTE). It extends Ethernet according to the IEEE 802.3 standard with mechanisms to transfer data with predictable timing and precise synchronisation. The communication profile meets timing demands typical of high-performance automation and motion applications. It does not change basic principles of the Fast Ethernet Standard IEEE 802.3 but extends it towards RTE. Thus it is possible to leverage and continue to use any standard Ethernet silicon, infrastructure component or test and measurement equipment like a network analyser.

EPL was originally designed by B&R GmbH and the first version was released in 2001. Later on, the Ethernet POWERLINK Standardization Group (EPSG) was in charge of continuing its management and published the first EPSG Draft version of EPL as an open standard in 2003 fostering for a free distribution of its specifications as well as for the open-source distribution of the protocol source code. The current version of this communication standard can actually be downloaded from the EPSG web site.

EPL is able to provide the real-time capabilities required by critical processes, control tasks and management functions typical of the industrial scenario. As a matter of fact, which is compliant with the Industrial Automation Open Network Alliance (IAONA) real-time class 4 (highest performance) recommendations, it is able to cope with communication cycles in the order of hundreds of microseconds, ensuring, at the same time, jitters below 1 μ s.

Also, the EPL application layer is based on the popular and proven CANopen standard (practitioners often refer to EPL as "CANopen over Ethernet", see below on page 84). This feature ensures, at high layer of the protocol stack, compatibility with several other industrial communication devices.

POWERLINK provides mechanisms to achieve the following aims:

1. transmit time-critical data in precise isochronous cycles. Data exchange is based on a publish/subscribe relationship. Isochronous data communication can be used for exchanging position data of motion applications of the automation industry;
2. synchronise networked nodes with high accuracy;
3. transmit less time-critical data asynchronously on request. Asynchronous data communication can be used to transfer IP-based protocols like TCP or UDP and higher layer protocols such as HTTP, FTP, etc.

POWERLINK manages the network traffic in a way that there are dedicated time-slots for isochronous and asynchronous data. It takes care that always only one networked device gains access to the network media. Thus transmission of isochronous and asynchronous data will never interfere and precise communication timing is guaranteed. The mechanism is called Slot Communication Network Management (SCNM). SCNM is managed and supervised by exactly one particular networked device – the Managing Node (MN, e.g. the Master) – which includes the MN functionality. All other nodes (up to 240) are called Controlled Nodes (CNs, e.g. the Slaves) and are deployed in various topologies (networks may have a star, tree, daisy chain or ring structure, or any combination of these topologies). Find further information in the "7.9 POWERLINK nodes" section on page 84.

7.6 Physical layer

POWERLINK is a protocol residing on top of the standard IEEE 802.3 MAC layer. The physical layer is 100Base-X (copper and fiber, see IEEE 802.3). Half-Duplex transmission mode is used.

POWERLINK uses Ethernet as it is, without any modifications. Hence any advancement in Ethernet Technology can be exploited (e.g. Gigabit Ethernet).

To increase noise immunity only S/FTP or SF/FTP cables must be used (CAT-5). The maximum cable length (100 meters, 328 ft) predefined by Ethernet 100Base-TX must be compulsorily fulfilled.

EPL recommends the use of hubs to fit POWERLINK jitter requirements.

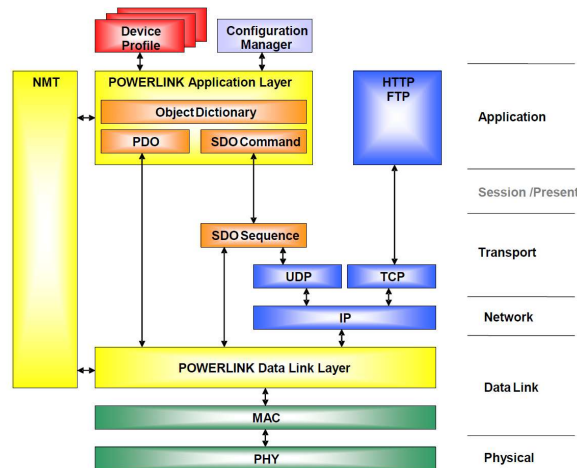
Switches may be used to build a POWERLINK network.

It has to be considered that any POWERLINK network constructed with anything but Class 2 Repeater Devices does not conform to the POWERLINK standard.

Refer also to the "4.2.3 Network configuration: cables, hubs, switches – Recommendations" section on page 39.

7.7 Reference model

POWERLINK-based networks use the following reference model.



7.8 CANopen over Ethernet

The Ethernet POWERLINK Standardisation Group (EPSG) is working closely with the CiA (CAN in Automation) organisation to integrate CANopen with POWERLINK. CANopen standards define widely deployed communication profiles, device profiles and application profiles. These profiles are in use millions of times all over the world. Integration of POWERLINK with CANopen combines profiles, high performance data exchange and open transparent communication with TCP/UDP/IP protocols.

The POWERLINK communication profile implemented in the actuator is based on CANopen communication profile DS301.

7.9 POWERLINK nodes

The node managing the permission to send messages to the Ethernet is called the POWERLINK Managing Node (MN).

All other nodes transmit only within communication slots assigned by the MN. They are thus called Controlled Nodes (CN).

7.9.1 POWERLINK Managing Node (MN)

Only the MN is allowed to send messages independently – i.e. not as a response to a received message. Controlled Nodes are only allowed to send when requested to by the MN.

The Controlled Nodes are accessed cyclically by the MN. Unicast data are sent from the MN to each configured CN (frame: PReq), which then publishes its data via multicast to all other nodes (frame: PRes).

All available nodes in the network are configured on the MN.

Only one active MN is permitted in a POWERLINK network.

7.9.2 POWERLINK Controlled Node CN)

CNs are passive bus nodes. They only send when requested by the MN.

The ability of a node to perform CN functions is indicated by the device description entry **D_DLL_FeatureCN_BOOL**.

Lika actuators are CN devices and comply with the "EPSG Draft Standard 301 Ethernet POWERLINK Communication Profile Specification Version 1.2.0" as well as with the CANopen Profiles "DS301 CANopen Application Layer and Communication Profile" according to the POWERLINK specifications.

7.10 POWERLINK Basic Frame

The POWERLINK Basic Frame format contains 5 fields:

- Reserved (1 bit)
- MessageType (7 bits)
- Destination node address (1 byte)
- Source node address (1 byte)
- Data (n bytes)

The POWERLINK Basic Frame format is encapsulated by the Ethernet II wrapper consisting of 14 bytes of leading Ethernet header (Destination and Source MAC addresses, EtherType) and 4 bytes of terminating CRC32 checksum.

Allowed frame sizes are ranging between 64 bytes to 1518 bytes, thus the POWERLINK frame can be between 46 bytes and 1500 bytes (see the [1030-04 NMT_InterfaceGroup_0h_REC.InterfaceMtu_U16](#) object on page 99).

	Bit offset								Entry defined by
Byte offset	7	6	5	4	3	2	1	0	
0 ... 5	Destination MAC Address								Ethernet Type II
6 ... 11	Source MAC Address								
12 ... 13	EtherType (0x88AB)								
14	res	MessageType							Ethernet POWERLINK
15	Destination (8 bit node ID)								
16	Source (8 bit node ID)								
17 ... n	Data								
n+1 ... n+4	CRC32								Ethernet Type II

POWERLINK is identified via the EtherType 88ABh.

7.11 Message types

7.11.1 Start of Cycle (SoC)

Start of Cycle (SoC, ID = 01h) is a broadcast frame sent by the MN to begin the POWERLINK cycle (see the "7.12 POWERLINK Cycle" section on page 86). To maintain a fixed cycle time, the SoC frame is issued on a precise periodic basis, keeping jitter on it as low as possible; this also serves to the purpose of providing time synchronisation for all the devices.

7.11.2 PollRequest (PReq)

After the SoC frame has been issued, the Isochronous Period is entered (see the "7.12.1 Isochronous Period" section on page 86). In this key part of the cycle, the MN polls each CN by means of the PollRequest (PReq, ID = 03h) frame which is sent only to the selected CN and carry output data for it.

7.11.3 PollResponse (PRes)

Following a PollRequest (PReq), the accessed CN responds to the query by issuing a PollResponse (PRes, ID = 04h) frame which is instead a multicast frame carrying input data, made available to all nodes in the network.

7.11.4 Start of Asynchronous (SoA)

Once the Isochronous Period is concluded, the MN sends a broadcast frame called Start of Asynchronous (SoA, ID = 05h) which informs all CNs about the start of the Asynchronous Period (see the "7.12.2 Asynchronous Period" section on page 87). This second phase ensures the transmission of only one asynchronous message by a selected node. The SoA frame is also used to inform which node has been selected for the acyclic communication.

7.11.5 Asynchronous Send (Asnd)

The Asynchronous Send (Asnd, ID = 06h) frame transports asynchronous data via the POWERLINK/ASnd protocol (e.g. NMT commands).

7.12 POWERLINK Cycle

Data exchange within a POWERLINK network is structured in fixed intervals, called POWERLINK cycles. The cycle is subdivided into the Isochronous Period, the Asynchronous Period and the Idle Period and is managed by the MN.

7.12.1 Isochronous Period

The Isochronous Period of a POWERLINK cycle offers deterministic operation, i.e. it is reserved for the exchange of (continuous or multiplexed) isochronous data. Isochronous data exchange between nodes occurs cyclically. It is repeated in a fixed interval, called the POWERLINK cycle. The POWERLINK cycle is controlled by the MN.

At the beginning of a POWERLINK cycle, the MN sends a SoC (Start of Cycle, see on page 86) frame to all nodes via Ethernet multicast. The send and receive time of this frame is the basis for the common timing of all the nodes.

Only the SoC frame is generated on a periodic basis. The generation of all other frames is event controlled (with additional time monitoring per node).

The MN starts the isochronous data exchange after the SoC frame has been sent. A PReq frame (see on page 86) is sent to every configured and active node. The accessed node responds by means of a PRes frame (see on page 86).

7.12.2 Asynchronous Period

The Asynchronous Period is the second part of the POWERLINK cycle, starting with a Start of Asynchronous (SoA) frame (see on page 86).

In the asynchronous phase of the cycle, access to the POWERLINK network may be granted to one CN or to the MN for the transfer of a single asynchronous message only.

There are two types of asynchronous frames available:

- the POWERLINK ASnd frame uses the POWERLINK addressing scheme and is sent via unicast or broadcast to any other node (see on page 86);
- a Legacy Ethernet message can be sent.

If no asynchronous message transmission request is pending at the MN scheduling queues, the MN issues a SoA without assignment of the right to send to any node. No ASnd frame follows to the SoA frame in this case.

The MN starts the asynchronous phase with the SoA. The SoA is used to identify CNs, request status information of a CN, to poll async-only CNs and to grant the asynchronous transmit right to one CN.

The SoA frame is the first frame in the asynchronous phase and is a signal to all CNs that all isochronous data has been exchanged during the isochronous phase.

The asynchronous phase is calculated from the start of SoA to the end of the asynchronous response. If no asynchronous response is allowed for any node, the asynchronous phase is terminated by the end of SoA.

7.12.3 Idle Period

After both the SoA and the acyclic frames have been transmitted, the Idle Period is entered. All the nodes wait for the new cycle to start, i.e. to receive a new SoC frame from the MN.

7.13 CN Node NMT States

After the Initialization NMT State Machine (it is common to both MN and CNs), the POWERLINK devices enter specific MN and CN states.

Here follows the list of the available CN states.

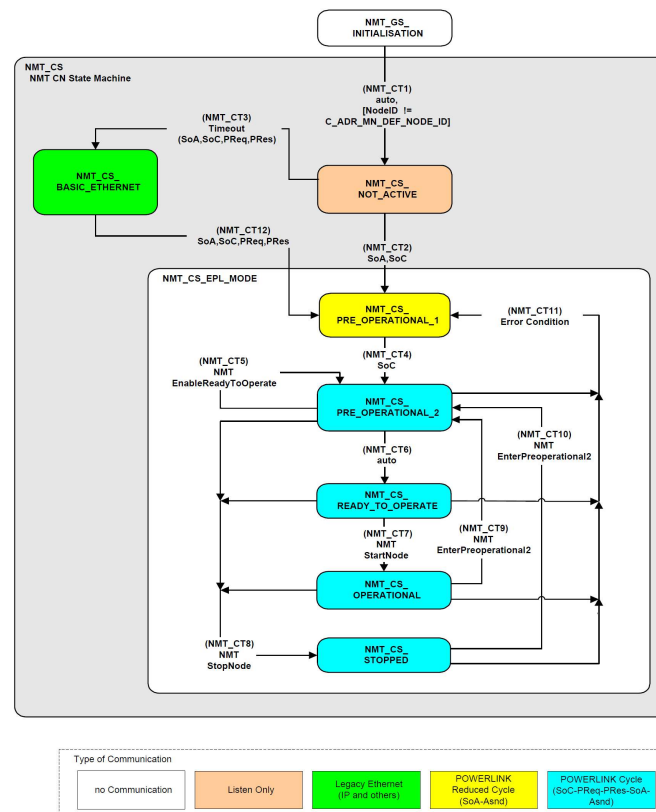


Figure 31 - CN NMT State Machine

7.13.1 NMT_CS_NOT_ACTIVE

NMT_CS_NOT_ACTIVE is a non-permanent state which allows a starting node to recognize the current network state. The MS (Module Status) LED is off. The CN observes network traffic. The node is not authorised to send frames autonomously. There is no Legacy Ethernet frame transmission allowed at **NMT_CS_NOT_ACTIVE**. The node is able to recognize **NMTReset** commands sent via Asnd.

The transition from **NMT_CS_NOT_ACTIVE** to **NMT_CS_PRE_OPERATIONAL_1** is triggered by a SoA or SoC frame being received.

The transition from **NMT_CS_NOT_ACTIVE** to **NMT_CS_BASIC_ETHERNET** is triggered by timeout for SoC, PReq, PRes and SoA frames.

7.13.2 NMT_CS_PRE_OPERATIONAL_1

In the state **NMT_CS_PRE_OPERATIONAL_1**, the CN sends a frame only if the MN has authorised it to do so by a **SoA AsyncInvite** command. The MS (Module Status) LED gives single green flashes.

In **NMT_CS_PRE_OPERATIONAL_1** the node is identified by the MN via **IdentRequest**. If required the CN downloads its configuration data from a configuration server. Both processes may be completely or partially shifted to **NMT_CS_PRE_OPERATIONAL_2**, if the MN is not in **NMT_CS_PRE_OPERATIONAL_1** or leaves **NMT_CS_PRE_OPERATIONAL_1** before the CN has completed its configuration.

The transition from **NMT_CS_PRE_OPERATIONAL_1** to the following state is triggered by a SoC frame being received.

There is no PDO communication in **NMT_CS_PRE_OPERATIONAL_1**.

7.13.3 NMT_CS_PRE_OPERATIONAL_2

In the state **NMT_CS_PRE_OPERATIONAL_2**, the CN waits for the configuration to be completed. The MS (Module Status) LED gives double green flashes.

The node is queried by the MN via PReq. The received PDO data may be invalid.

The PDO data received from the MN via PReq and from other CNs and the MN via PRes is ignored by the CN.

CNs respond to **AsyncInvite** commands via SoA. If not invited by the MN, there is no Ethernet frame transmission allowed at the **NMT_CS_PRE_OPERATIONAL_2** state.

Precondition for the transition from **NMT_CS_PRE_OPERATIONAL_2** to **NMT_CS_READY_TO_OPERATE** is the reception of an **NMTEnableReadyToOperate** command. The transition is triggered if the application is ready for operation.

The transition from **NMT_CS_PRE_OPERATIONAL_2** to **NMT_CS_PRE_OPERATIONAL_1** is triggered by an error recognition.

The transition from **NMT_CS_PRE_OPERATIONAL_2** to **NMT_CS_STOPPED** is triggered by reception of NMT state command **NMTStopNode**.

7.13.4 NMT_CS_READY_TO_OPERATE

With the state **NMT_CS_READY_TO_OPERATE**, the CN signals its readiness to operation to the MN. The MS (Module Status) LED gives triple green flashes.

The node may participate in cyclic frame exchange. Nodes respond via PRes when queried via PReq by the MN.

CNs respond to **AsyncInvite** commands via SoA. If not invited by the MN, there is no Ethernet frame transmission allowed at the **NMT_CS_READY_TO_OPERATE** state.

The transition from **NMT_CS_READY_TO_OPERATE** to **NMT_CS_OPERATIONAL** is triggered by the reception of NMT state command **NMTStartNode**.

The transition from **NMT_CS_READY_TO_OPERATE** to **NMT_CS_PRE_OPERATIONAL_1** is triggered by an error recognition.

The transition from **NMT_CS_READY_TO_OPERATE** to **NMT_CS_STOPPED** is triggered by reception of NMT state command **NMTStopNode**.

7.13.5 **NMT_CS_OPERATIONAL**

NMT_CS_OPERATIONAL is the normal operating state of a CN. The MS (Module Status) LED is solidly lit green.

The CN may participate in cyclic frame exchange. The CN responds via PRes when queried via PReq by the MN.

CNs respond to **AsyncInvite** commands via SoA. If not invited by the MN, there is no Ethernet frame transmission allowed at the **NMT_CS_OPERATIONAL** state.

The PDO data received from the MN via PReq and from other CNs and the MN via PRes shall be interpreted if selected by the CN application.

The transition from **NMT_CS_OPERATIONAL** to **NMT_CS_PRE_OPERATIONAL_2** is triggered by the reception of NMT state command **NMTEnterPreOperational2**.

The transition from **NMT_CS_OPERATIONAL** to **NMT_CS_PRE_OPERATIONAL_1** is triggered by an error recognition.

The transition from **NMT_CS_OPERATIONAL** to **NMT_CS_STOPPED** is triggered by reception of NMT state command **NMTStopNode**.

7.13.6 **NMT_CS_STOPPED**

In the **NMT_CS_STOPPED** state, the node is largely passive. The MS (Module Status) LED blinks green (200 ms ON, 200 ms OFF).

NMT_CS_STOPPED is used for controlled shutdown of a selected CN while the system is still running.

The node does not participate in cyclic frame exchange, but still observes SoA frames.

It is not queried by the MN via PReq.

The node does not respond via PRes when queried by the MN via PReq.

The node responds to **AsyncInvite** commands via SoA. If not invited by the MN, there is no Ethernet frame transmission allowed at the **NMT_CS_STOPPED** state.

The transition from **NMT_CS_STOPPED** to **NMT_CS_PRE_OPERATIONAL_2** is triggered by the reception of NMT state command **NMTEnterPreOperational2**.

The transition from **NMT_CS_STOPPED** to **NMT_CS_PRE_OPERATIONAL_1** is triggered by an error recognition.

7.13.7 NMT_CS_BASIC_ETHERNET

In the **NMT_CS_BASIC_ETHERNET** state the node is allowed to perform Legacy Ethernet communication according to IEEE 802.3. There is no POWERLINK specific network traffic control. The MS (Module Status) LED flickers green (50 ms ON, 50 ms OFF).

The node is allowed to transmit autonomously.

Any Legacy Ethernet protocol can be applied.

Asnd frames can be transmitted by a CN in state **NMT_CS_BASIC_ETHERNET**.

To avoid disturbance of POWERLINK network traffic when the node is in **NMT_CS_BASIC_ETHERNET**, the node recognizes SoC, PReq, PRes and SoA frames. On the reception of such a frame, the CN immediately stalls any autonomous frame transmission and changes over to **NMT_CS_PRE_OPERATIONAL_1**.

7.14 XDD file

The functionality of a POWERLINK device is always described in a XDD file (XML Device Description file). The Device Description File provides information about the device basic communication and functional properties. It must be installed in the MN device.

The file name is primarily built up as follows:

0xVendorID_ProductName.xdd

e.g. 0000012E_Lika_RD1xA_PL_EthernetPOWERLINK

POWERLINK actuators from Lika Electronic are supplied with their own XDD file. It is:

- **0000012E_Lika_RD1xA_PL_EthernetPOWERLINK.xdd**: it is intended for installation of **RD1xA series rotary actuators** ("0000012E_Lika" shows the Vendor ID -expressed in hexadecimal notation- and name; "RD1xA" is the actuator series; "PL" is the abbreviation for POWERLINK).

XDD files are paired with the **RD1xA.bmp** picture file available inside the file folder.

Follow the path **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS/POSITIONING UNITS (DRIVECOD)** to download the XDD files from Lika's corporate web site.

7.15 Communication objects

POWERLINK uses the same device description files as CANopen, the same Object Dictionaries and the same communication mechanisms, such as Network Management (NMT), Process Data Objects (PDO) and Service Data Objects (SDO).

Three main kinds of communication messages are used in a POWERLINK network:

- **Network Management NMT** protocol: NMT protocols are used to issue state machine change commands (i.e. to start and stop the devices), detect remote device boot-ups and error conditions.
- **Process Data Objects PDO** protocol: used to process real time data (transmission of process data in real time).
- **Service Data Objects SDO** protocol: used to set and read values from the Object Dictionary of a remote device in the asynchronous phase.

7.15.1 NMT Network Management

POWERLINK Network Management (NMT) is node-oriented and follows a Master/Slave relationship. The function of the NMT Master is carried out by the MN.

- **NMT State Command Services.** The MN uses NMT State Command Services to control the CN state machine(s), see the "7.13 CN Node NMT States" section on page 87.
- **NMT Managing Command Services.** The MN uses NMT Managing Command Services to access NMT data items of the CN(s) in a fast coordinated way.
- **NMT Response Services.** NMT Response Services indicate the current NMT state of a CN to the MN.
- **NMT Info Services.** NMT Info Services are used to transmit NMT information from the MN to a CN.
- **NMT Guard Services.** NMT Guard Services are used by the MN and CNs to detect failures in a POWERLINK network.

A CN may request NMT command and info services to be issued by the MN. NMT services are defined in the Communication Profile Area of the Object Dictionary, refer to the "7.16 Object Dictionary" section on page 94.

7.15.2 PDO objects

The real-time data transfer is performed by means of Process Data Objects (PDO).

PDO communication in POWERLINK is always performed isochronously by PReq and/or PRes frames. The PRes frames are sent as broadcasts following the

producer/consumer scheme. The PReq frames with unicast addresses comply with the Master/Slave relationship.

The transmission type of PDO is continuous. There is no "on event" or "on change" transmission type provided.

7.15.3 PDO Mapping

The PDO Mapping determines the payload of a PDO frame in a POWERLINK network. PDO payload is transported via the frame types PollRequest and PollResponse. Due to the IEEE802.3 standard, any Ethernet frame can have a size between 64 bytes and 1518 bytes. Subtracting the Ethernet- and the POWERLINK-header possible payload sizes of 36 bytes up to 1490 bytes are resulting that can be used for PDO data.

The PDO Mapping is derived from the communication relations between the different nodes in the network. Based on these relations, the payload of the Transmit PDO frames (TPDOs) and the Receive PDO frames (RPDOs) is defined. The PDO mapping can be unchangeable ("static mapping") or configurable ("dynamic mapping").

Mapping tables

The PDO Mapping itself is configured in the form of mapping tables in the Object Dictionary on each node in the network. It consists of PDO Communication Parameters and PDO Mapping Parameters.

The following regions of the Object Dictionary are assigned for the PDO Mapping:

Region	Name	Description
1400h - 14FFh	PDO_RxCommParam_XXh_REC see on page 101 and following	Communication parameters for RPDOs
1600h - 16FFh	PDO_RxMappParam_XXh_AU64 see on page 101	Object mapping for RPDOs
1800h - 18FFh	PDO_TxCommParam_XXh_REC see on page 102 and following	Communication parameters for TPDOs
1A00h - 1AFFh	PDO_TxMappParam_XXh_AU64 see on page 103	Object mapping for TPDOs

Relationship of Communication Parameters and Mapping Parameters

A single PDO is always described by a pair of one communication parameter object and one mapping parameter object. The pairing is based on the lower byte of the object index.

Examples:

Direction	Communication parameter	Mapping parameter
RPDO	1400h	1600h
	1401h	1601h

TPDO	1800h	1A00h
	1801h	1A01h

7.15.4 SDO objects

To access the entries of the Object Dictionary of a device via Ethernet POWERLINK a set of command services is specified.

An SDO client establishes a connection to an SDO server and issues a specific command (read or write from/to an object). This connection is unicast, allows access to all objects of a remote node and is not deterministic (i.e. there is no guarantee on how long the response takes after sending a request). SDO communication takes place in the asynchronous phase and can be embedded in a POWERLINK Asynchronous Send (ASnd) frame or in a UDP/IP packet.

SDO communication attends to the Client / Server model.

7.16 Object Dictionary

The most important part of a device profile is the Object Dictionary. The Object Dictionary is essentially a grouping of objects accessible via the network in an ordered, pre-defined fashion. Each object within the dictionary is addressed using a 16-bit index.

The Object Dictionary can contain a maximum of 65,536 entries.

The user-related objects are grouped in three main areas: the Communication Profile Area, the Manufacturer Specific Profile Area and the Standardised Device Profile Area. The objects are described in the XDD file.

The **Communication Profile Area** at indexes from 1000h to 1FFFh contains the communication specific parameters for the POWERLINK network. These entries are common to all devices. NMT services, PDO objects and SDO objects are described in this section. The Communication Profile Area objects comply with the "CiA Draft Standard Proposal 301 CANopen Application layer and communication profile". Refer to the "7.16.1 Communication Profile Area objects (DS 301)" section on page 96.

The **Manufacturer Specific Profile Area** at indexes from 2000h to 5FFFh is free to add manufacturer-specific functionality. The objects that are specifically intended to be used for configuring Lika's actuators can be found in this group. Refer to the "7.16.2 Manufacturer Specific Profile Area objects" section on page 117.

The **Standardised Device Profile Area** at indexes from 6000h to 9FFFh contains all data objects common to a class of devices that can be read or written via the network. The device profiles may use entries from 6000h to 9FFFh to describe the device parameters and the device functionality. RD1xA rotary actuators have no parameters in this profile area.

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

Index-subIndex Object name

[data types, attribute]

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

7.16.1 Communication Profile Area objects (DS 301)

1000-00 NMT_DeviceType_U32

[Unsigned32, const]

It contains information about the device type. The object describes the type of device and its functionality.

Default = 0000 012Dh = rotary actuator, RD1xA series, in compliance with DS301 profile

1001-00 ERR_ErrorRegister_U8

[Unsigned8, ro]

This object provides error information. The POWERLINK device maps internal errors into this object. The structure of the error register is as follows:

Bit	Meaning
0	Generic error
1	Current
2	Voltage
3	Temperature
4	Communication error
5	Device profile specific
6	Reserved (always 0)
7	Manufacturer specific

If a generic error occurs, the bit 0 will be set to "1".

If a specific error occurs, the corresponding bit (except 6) will be set to "1".

Default = 00h

1006-00 NMT_CycleLen_U32

[Unsigned32, rw]

This object defines the communication cycle time interval expressed in μ s. This period defines the SYNC interval. The object should be set by the system configuration.

Default = 1000000 (min. = 200, max. = 2147483)

1008-00 NMT_ManufactDevName_VS

[String64, const]

It contains the manufacturer device name.

Default = RD1xA-PL-xx = RD1xA series with POWERLINK interface, see the order code

1009-00 NMT_ManufactHwVers_VS

[String64, const]

It shows the manufacturer hardware version description.

Default = device dependent

100A-00 NMT_ManufactSwVers_VS

[String64, const]

It shows the manufacturer software version description.

Default = device dependent

1018-00 NMT_IdentityObject_REC

[Unsigned8, const]

The following objects contain general information about the device. This sub-Index contains the number of entries.

Default = 4

1018-01 NMT_IdentityObject_REC.VendorID_U32

[Unsigned32, const]

It provides the manufacturer-specific vendor ID. The POWERLINK vendor ID is the same as the CANopen vendor ID.

Default = 0000 012Eh = Lika Electronic

1018-02 NMT_IdentityObject_REC.ProductCode_U32

[Unsigned32, const]

The manufacturer-specific product code identifies a specific device version.

Default = 0000 2010h = RD1xA rotary actuator series

1018-03 NMT_IdentityObject_REC.RevisionNo_U32

[Unsigned32, const]

The manufacturer-specific revision number consists of a major revision number and a minor revision number. The major revision number identifies a specific device behaviour. The minor revision number identifies different versions with the same device behaviour.

Default = 0000 0001h

31	16	15	0
Major revision number		Minor revision number	
MSB	LSB

1018-04 NMT_IdentityObject_REC.SerialNo_U32

[Unsigned32, const]

It provides the Serial Number of the device.

Default = FFFF FFFFh (=not used)

1020-00 CFM_VerifyConfiguration_REC

[Unsigned8, const]

The following objects hold device local configuration date and time. This sub-Index contains the number of entries.

Default = 2

1020-01 CFM_VerifyConfiguration_REC.ConfDate_U32

[Unsigned32, rw]

It holds the local configuration date. It contains the number of days since January 1, 1984.

Default = 0000 0000 (min. = 0000 0000, max. = FFFF FFFFh)

1020-02 CFM_VerifyConfiguration_REC.ConfTime_U32

[Unsigned32, rw]

It holds the local configuration time. It contains the number of ms since midnight.

Default = 0000 0000 (min. = 0000 0000, max. = FFFF FFFFh)

1021-00 CFM_StoreDevDescrFile_DOM

[Domain, ro]

It holds the device local Device Description File. The object is read-only if the Device Description File stored by **1021-00 CFM_StoreDevDescrFile_DOM** is unchangeable.

1022-00 CFM_StoreDevDescrFormat_U16

[Unsigned16, ro]

It holds the format of the Device Description File stored by **1021-00 CFM_StoreDevDescrFile_DOM**.

Default = 255 (min. = 0000, max. = FFFFh)

1030-00 NMT_InterfaceGroup_0h_REC

[Unsigned8, const]

The following objects are used to configure and retrieve parameters of the network interfaces (physical or virtual) via SDO. This sub-Index contains the number of entries.

Default = 9

1030-01 NMT_InterfaceGroup_0h_REC.InterfaceIndex_U16

[Unsigned16, ro]

Interface index of the physical interface. This number is the index number subtracted by 102Fh. The POWERLINK node that adds an interface generates the respective value.

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

1030-02 NMT_InterfaceGroup_0h_REC.InterfaceDescription_VSTR

[String, const]

This string provides information about the product family, the product name and the version of the hardware interface.

Default = DRIVECOD RD1xA-PL-xx 1.0 = RD1xA series

1030-03 NMT_InterfaceGroup_0h_REC.InterfaceType_U8

[Unsigned8, const]

The type of interface, distinguished according to the physical/link protocol(s) immediately 'below' the network layer in the protocol stack.

Default = 6 (ethernet-csmacd)

1030-04 NMT_InterfaceGroup_0h_REC.InterfaceMtu_U16

[Unsigned16, const]

It contains the size of the largest datagram that can be sent/received on the interface, specified in bytes.

Default = 1518 (min. = 0, max. = 65535)

1030-05 NMT_InterfaceGroup_0h_REC.InterfacePhysAddress_OSTR

[Octet_String, const]

This object contains the MAC address of the Ethernet device assigned at production.

Default = device dependent

1030-06 NMT_InterfaceGroup_0h_REC.InterfaceName_VSTR

[String, ro]

The user reference name for the interface.

Default = Interface 1

1030-07 NMT_InterfaceGroup_0h_REC.InterfaceOperStatus_U8

[Unsigned8, ro]

It shows the current operational state of the interface.

It can be: 0 = Down

 1 = Up

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

1030-08 NMT_InterfaceGroup_0h_REC.InterfaceAdminState_U8

[Unsigned8, rw]

It shows the current administration state of the interface.

It can be: 0 = Down

 1 = Up

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

1030-09 NMT_InterfaceGroup_0h_REC.Valid_BOOL

[Boolean, rw]

It specifies whether the data of this object is valid. If the value is TRUE (1) the data of this object is valid. If the value is FALSE (0) the data of this object is invalid.

Default = TRUE (1) (min. = FALSE (0), max. = TRUE (1))

1300-00 SDO_SequLayerTimeout_U32

[Unsigned32, rw]

The object provides a timeout value in ms for the connection abort recognition of the SDO sequence layer.

The connection is detected as broken if the opposite node is shut down or disconnected from the network. The connection is considered broken when no acknowledgement is received within the timeout set in this object.

Default = 15000 (min. = 100, max. = FFFF FFFFh)

1400-00 PDO_RxCommParam_00h_REC

[Unsigned8, const]

These indices describe the communication attributes of the RPDO channels. Mapping version and address information are provided. The values in these object entries cannot be changed in POWERLINK states **NMT_CS_READY_TO_OPERATE** or **NMT_CS_OPERATIONAL**. If the network tries to change any of these entries when the device is in one of these states the command will be aborted and an SDO abort code will be returned to the network. This sub-Index contains the number of entries.

Default = 2

1400-01 PDO_RxCommParam_00h_REC.NodeID_U8

[Unsigned8, ro]

Node ID of the node transmitting the corresponding PRes. Node ID entry 0 is reserved for PReq received by a CN.

Default = 0

1400-02 PDO_RxCommParam_00h_REC.MappingVersion_U8

[Unsigned8, ro]

For static mapping, the value is 0. Access is ro if only static mapping is provided by the device.

Default = 0

1600-00 PDO_RxMappParam_00h_AU64

[Unsigned8, ro]

These indices describe the mapping of the objects contained in Receive PDO payload to the Object Dictionary entries. The values in these object entries cannot be changed in POWERLINK states **NMT_CS_READY_TO_OPERATE** or **NMT_CS_OPERATIONAL**. If the network tries to change any of these entries when the device is in one of these states the command will be aborted and an SDO abort code will be returned to the network. This sub-Index contains the number of entries.

Default = 2

1600-01 PDO_RxMappParam_00h_AU64.ObjectMapping

[Unsigned64, ro]

It describes the mapping of the **2200-00 Control Word** object to the according RPDO. The entries are interpreted according to the following table (all values in hex):

	Length (bit)	Offset (bit)	res	Sub-Index	Index
	LLLL	0000	rr	ss	IIII
e.g.	0010	0000	00	00	2200

Map 10h = 16bit starting at offset 0 to **2200-00 Control Word** object

Default = 0010 0000 0000 2200h

1600-02 PDO_RxMappParam_00h_AU64.ObjectMapping

[Unsigned64, ro]

It describes the mapping of the **2201-00 Target position** object to the according RPDO. The entries are interpreted according to the following table (all values in hex):

	Length (bit)	Offset (bit)	res	Sub-Index	Index
	LLLL	0000	rr	ss	IIII
e.g.	0020	0010	00	00	2201

Map 20h = 32bit starting at offset 10h = 32bit to **2201-00 Target position** object

Default = 0020 0010 0000 2201h

1800-00 PDO_TxCommParam_00h_REC

[Unsigned8, const]

These indices describe the communication attributes of the Transmit PDO channels. Mapping version and address information are provided. As a CN has only one TPDO channel, only the first index **PDO_TxCommParam_XXh_REC** is implemented on a CN. The values in these object entries cannot be changed in POWERLINK states **NMT_CS_READY_TO_OPERATE** or **NMT_CS_OPERATIONAL**. If the network tries to change any of these entries when the device is in one of these states the command will be aborted and an SDO abort code will be returned to the network. This sub-Index contains the number of entries.

Default = 2

1800-01 PDO_TxCommParam_00h_REC.NodeID_U8

[Unsigned8, ro]

It contains the Node ID of the PDO target. It is 0 (not used) for CNs.

Default = 0

1800-02 PDO_TxCommParam_00h_REC.MappingVersion_U8

[Unsigned8, ro]

Compatibility of TPDO channels and corresponding RPDO channels may be ensured by PDO mapping version handling. PDO Mapping can be variable (configurable) or static (unchangeable). Variable mapping may be dynamically modified by the application, even under operation. Static mapping is pre-defined and may not be modified in any way. The version info is transmitted by the Master or producer with every PDO transporting PReq and PRes frame. A PDO mapping version value of 0 indicates that there is no mapping version available.

Default = 0

1A00-00 PDO_TxMappParam_00h_AU64

[Unsigned8, ro]

These indices describe the mapping of the objects contained in TPDO payload to the Object Dictionary entries. As a CN has only one TPDO channel, only the first index **PDO_TxMappParam_XXh_AU64** is implemented on a CN. The values in these object entries cannot be changed in POWERLINK states **NMT_CS_READY_TO_OPERATE** or **NMT_CS_OPERATIONAL**. If the network tries to change any of these entries when the device is in one of these states the command will be aborted and an SDO abort code will be returned to the network. This sub-Index contains the number of active entries.

Default = 2

1A00-01 PDO_TxMappParam_00h_AU64.ObjectMapping

[Unsigned64, ro]

It describes the mapping of the **2202-00 Status word** object to the according TPDO. The entries are interpreted according to the following table (all values in hex):

	Length (bit)	Offset (bit)	res	Sub-Index	Index
	LLLL	0000	rr	ss	IIII
e.g.	0010	0000	00	00	2202

Map 10h = 16bit starting at offset 0 to **2202-00 Status word** object

Default = 0010 0000 0000 2202h

1A00-02 PDO_TxMappParam_00h_AU64.ObjectMapping

[Unsigned64, ro]

It describes the mapping of the **2203-00 Position** object to the according TPDO. The entries are interpreted according to the following table (all values in hex):

	Length (bit)	Offset (bit)	res	Sub-Index	Index
	LLLL	0000	rr	ss	IIII
e.g.	0020	0010	00	00	2203

Map 20h = 32bit starting at offset 10h = 32bit to **2203-00 Position** object

Default = 0020 0010 0000 2203h

1C0B-00 DLL_CNLossSoC_REC

[Unsigned8, const]

The following objects are used to monitor "Loss of SoC" (Start of Cycle frame) error symptoms detected by a CN. The record consists of a cumulative counter and a threshold counter data object and its threshold data object. This sub-Index contains the number of entries.

Default = 3

1C0B-01 DLL_CNLossSoC_REC.CumulativeCnt_U32

[Unsigned32, rw]

The cumulative counter is incremented by 1 every time a "Loss of SoC" error symptom occurs. Its value monitors all "Loss of SoC" error symptoms that are detected by the CN.

Default = 0



NOTE

If the unit is reset, this attribute is set to its default value.

1C0B-02 DLL_CNLossSoC_REC.ThresholdCnt_U32

[Unsigned32, ro]

The threshold counter shall be incremented by 8 every time a "Loss of SoC" error symptom occurs and decremented by 1 at every cycle without recurrence of the error. Its value monitors the quality of network in relation to the "Loss of SoC" error occurrence.

Default = 0

1COB-03 DLL_CNLossSoC_REC.Threshold_U32

[Unsigned32, rw]

Every time **1COB-02 DLL_CNLossSoC_REC.ThresholdCnt_U32** reaches the threshold, a defined action proceeds and **1COB-02 DLL_CNLossSoC_REC.ThresholdCnt_U32** is reset to 0.

Threshold Counting can be deactivated by setting **1COB-03 DLL_CNLossSoC_REC.Threshold_U32** to 0. If Threshold Counting is deactivated, no error reaction will occur.

Default = 15 (min. = 0, max. = FFFF FFFFh)

1COF-00 DLL_CNCRCErrror_REC

[Unsigned8, const]

The following objects are used to monitor CRC (Cyclic Redundancy Check) errors. The record consists of a cumulative counter and a threshold counter data object and its threshold data object. This sub-Index contains the number of entries.

Default = 3

1COF-01 DLL_CNCRCErrror_REC.CumulativeCnt_U32

[Unsigned32, rw]

The cumulative counter is incremented by 1 every time a CRC error occurs. Its value monitors all CRC errors that are detected by the CN.

Default = 0



NOTE

If the unit is reset, this attribute is set to its default value.

1COF-02 DLL_CNCRCErrror_REC.ThresholdCnt_U32

[Unsigned32, ro]

The threshold counter is incremented by 8 every time a CRC error occurs on the CN and decremented by 1 at every cycle without recurrence of the error. Its value monitors the quality of network in relation to the CRC error occurrence.

Default = 0

1COF-03 DLL_CNCRCErrror_REC.Threshold_U32

[Unsigned32, rw]

Every time **1COF-02 DLL_CNCRCErrror_REC.ThresholdCnt_U32** reaches the threshold, a defined action proceeds and **1COF-02 DLL_CNCRCErrror_REC.ThresholdCnt_U32** is reset to 0.

Threshold Counting can be deactivated by setting **1C0F-03 DLL_CNCRCErrror_REC.Threshold_U32** to 0. If Threshold Counting is deactivated, no error reaction will occur.

Default = 15 (min. = 0, max. = FFFF FFFFh)

1C14-00 DLL_CNLossOfSocTolerance_U32

[Unsigned32, rw]

This object provides a tolerance interval expressed in ns to be applied by CNs for "Loss of SoC" error recognition.

Default = 100000 (min. = 0, max. = 2147483000)

1F50-00 PDL_DownloadProgData_ADOM

[Unsigned8, const]

It holds downloaded programs. It allows the access to up to 254 programs. This sub-Index contains the number of entries.

Default = 1

1F50-01 PDL_DownloadProgData_ADOM.Program

[Domain, rw]

This 1 is the program access point of a device that is reserved for the firmware of the device holding the device communication profile. Therefore no common program download to this sub-index is allowed.

1F51-00 PDL_ProgCtrl_AU8

[Unsigned8, ro]

It is specified for controlling the execution of stored programs. This sub-Index contains the number of entries.

Default = 1

1F51-01 PDL_ProgCtrl_AU8.ProgCtrl

[Unsigned8, ro]

It controls the program holding the device communication profile to be accessed by **1F50-01 PDL_DownloadProgData_ADOM.Program**.

On write access program execution will be commanded, on read access the current execution state of the program may be queried. The following values are defined:

	Write Access	Read Access
0	stop program	program stopped
1	start program	program running
2	reset program	program stopped

Default = 0

1F52-00 PDL_LocVerApplSw_REC

[Unsigned8, const]

It is defined to support verification of the version of the program holding the device communication profile to be accessed via **1F50-01 PDL_DownloadProgData_ADOM.Program**. This sub-Index contains the number of entries.

Default = 2

1F52-01 PDL_LocVerApplSw_REC.ApplSwDate_U32

[Unsigned32, rw]

It contains the number of days since January 1, 1984.

1F52-02 PDL_LocVerApplSw_REC.ApplSwTime_U32

[Unsigned32, rw]

It contains the number of ms since midnight.

1F81-00 NMT_NodeAssignment_AU32

[Unsigned8, rw]

This object assigns nodes to the NMT Master (MN). Each sub-Index in the array corresponds to the node with the Node ID equal to the sub-Index. This sub-Index contains the number of entries.

Default = 254 (min. = 1, max. = 254)

1F81-01 NMT_NodeAssignment_AU32.NodeAssignment

[Unsigned32, rw]

Bit field, its meaning is according to the following table:

Byte	Bit	Value	Description
0	0	0	Node with this ID does not exist, bits 1 to 30 are not used.
		1	Node with this ID exists.
	1	0	Node with this ID is not a CN, Bits 2 .. 7, 9, 13 .. 30 are not used.
		1	Node with this ID is a CN. After configuration (with Configuration Manager) the Node will be set to state NMT_CS_OPERATIONAL .
	2	0	On detection of a booting CN MN informs the application but does NOT automatically configure and start the node.
		1	On detection of a booting CN MN informs the application and

			continues the process "START_CN".
	3	0	Optional CN.
		1	Mandatory CN.
	4	0	The CN node can be reset by the NMTSwReset , NMTResetNode , NMTResetCommunication or NMTResetConfiguration commands independent of its state. Hence no checking of its state needs to be performed prior to NMT Reset Communication.
		1	MN must not send any of the reset commands listed above to this node if it notices that the CN is in NMT_CS_OPERATIONAL state.
	5	0	Application software version verification for this node is not required.
		1	Application software version verification for this node is required.
	6	0	Automatic application software update (download) is not allowed.
		1	Automatic application software update (download) is allowed.
	7	–	Reserved
1	8	0	Isochronously accessed node.
		1	AsyncOnly node.
	9	0	Continuously accessed CN.
		1	Multiplexed CN.
	10	0	Device is not a Router Type 1.
		1	Device is a Router Type 1.
	11	0	Device is not a Router Type 2.
		1	Device is a Router Type 2.
	12	0	MN does not transmit PRes.
		1	MN transmits PRes.
	13	–	Reserved.
	14	–	Reserved.
	15	–	Reserved.
2	16 ... 23	–	Reserved.
3	24 ... 30	–	Reserved.
	31	0	Bit 0 ... 30 not valid.
		1	Bit 0 ... 30 valid.

Default = 0

1F82-00 NMT_FeatureFlags_U32

[Unsigned32, const]

Feature Flags indicate communication profile specific properties of the device given by its design. The object is set up by the device firmware during system initialisation.

Its meaning is according to the following table (grey background shows the set values, see the default binary value):

Byte	Bit	Name	TRUE (1)	FALSE (0)
0	0	Isochronous	Device can be isochronously accessed via PReq, it can be operated as isochronous CN. D_NMT_Isochronous_BOOL	Device does not support isochronous access via PReq, it may be exclusively used as async-only CN.
	1	SDO by UDP/IP	Device supports SDO communication via UDP/IP frames. D_SDO_SupportUdpIp_BOOL	Device does not support
	2	SDO by Asnd	Device supports SDO communication via POWERLINK Asnd frames. D_SDO_SupportASnd_BOOL	Device does not support
	3	SDO by PDO	Device supports SDO communication via container embedded in PDO communication. D_SDO_SupportPDO_BOOL	Device does not support
	4	NMT Info Services	Device supports NMT Info Services.	Device does not support
	5	Extended NMT State Commands	Device supports extended NMT State Commands. D_NMT_ExtNmtCmds_BOOL	Device does not support
	6	Dynamic PDO Mapping	Device supports dynamic PDO mapping. D_PDO_DynamicMapping_BOOL	Device does not support
	7	NMT Services by UDP/IP	Device supports NMT Services by UDP/IP. D_NMT_ServiceUdpIp_BOOL	Device does not support
1	8	Configuration Manager	Device supports configuration manager functions. D_CFM_ConfigManager_BOOL	Device does not support
	9	Multiplexed Access	Device supports multiplexed isochronous access. D_DLL_CNFeatureMultiplex_BOOL	Device does not support

	10	Node ID setup by SW	Device supports	Device does not support
			Node ID setup via SW. D_NMT_NodeIDBySW_BOOL	
	11	MN Basic Ethernet Mode	MN Device supports	MN Device does not support
			Basic Ethernet Mode. D_NMT_MNBasicEthernet_BOOL	
	12	Routing Type 1 Support	Device supports	Device does not support
			Routing Type 1 functions. D_RT1_RT1Support_BOOL	
	13	Routing Type 1 Support	Device supports	Device does not support
			Routing Type 2 functions. D_RT2_RT2Support_BOOL	
	14	SDO Read/Write All by Index	Device supports	Device does not support
			SDO commands Read and Write All by Index. D_SDO_CmdReadAllByIndex_BOOL D_SDO_CmdWriteAllByIndex_BOOL	
	15	SDO Read/Write Multiple Parameter by Index	Device supports	Device does not support
			SDO commands Read and Write Multiple Parameter by Index. D_SDO_CmdReadMultParam_BOOL D_SDO_CmdWriteMultParam_BOOL	
2	16 ... 23	-	Reserved.	
3	24 ... 31	-	Reserved.	

Default = 0004 8204h = 0100 1000 0010 0000 0101₂

1F83-00 NMT_EPLVersion_U8

[Unsigned8, const]

The index holds the POWERLINK communication profile version that is implemented by the device, according to the following table:

High Nibble	Low Nibble
POWERLINK Main Version	POWERLINK Sub Version

Default = 20h

1F8C-00 NMT_CurrNMTState_U8

[Unsigned8, ro]

The index holds the node's current NMT state, according to the following table. The CN Node specific states are described in the "7.13 CN Node NMT States" section on page 87.

	NMT states	Value	
MN and CN States	NMT_GS_OFF	0000 0000	
	NMT_GS_POWERED	xxxx 1xxx	Super state
	NMT_GS_INITIALISATION	xxxx 1001	Super state
	NMT_GS_INITIALISING	0001 1001	
	NMT_GS_RESET_APPLICATION	0010 1001	
	NMT_GS_RESET_COMMUNICATION	0011 1001	
	NMT_GS_RESET_CONFIGURATION	0111 1001	
	NMT_GS_COMMUNICATING	xxxx 11xx	Super state
CN States	NMT_CS_NOT_ACTIVE	0001 1100	
	NMT_CS_EPL_MODE	xxxx 1101	Super state
	NMT_CS_PRE_OPERATIONAL_1	0001 1101	
	NMT_CS_PRE_OPERATIONAL_2	0101 1101	
	NMT_CS_READY_TO_OPERATE	0110 1101	
	NMT_CS_OPERATIONAL	1111 1101	
	NMT_CS_STOPPED	0100 1101	
	NMT_CS_BASIC_ETHERNET	0001 1110	



EXAMPLE

1F8C-00 NMT_CurrNMTState_U8 = 253₁₀ = 1111 1101₂ = NMT_CS_OPERATIONAL state

1F8D-00 NMT_PResPayloadLimitList_AU16

[Unsigned8, rw]

The following object(s) hold(s) a list of the expected PRes payload data slot size in bytes for each configured node that is isochronously accessed, e.g. via PReq / PRes frame exchange. The payload data slot size is a measure for the configured size of the PRes frame. The data slot can be filled by PDO data up to this limit. This sub-Index contains the number of entries.

Default = 254 (min. = 1, max. = 254)

1F8D-01 NMT_PResPayloadLimitList_AU16.PResPayloadLimit

[Unsigned16, rw]

Each sub-Index in the array corresponds to the node with the Node ID equal to the sub-Index. The sub-Index value is valid only if there is an isochronous node assigned to the Node ID by index [NMT_NodeAssignment_AU32](#)[sub-Index] bits 0 and 8 (see page 107).

Sub-Index [C_ADR_MN_DEF_NODE_ID](#) indicates the payload size of the PRes frame issued by the MN.

Values should be equal on all nodes of the segment.

Default = 36 (C_DLL_ISOCHR_MAX_PAYL)

1F93-00 NMT_EPLNodeID_REC

[Unsigned8, const]

The following objects store the device's POWERLINK Node ID. This sub-Index contains the number of entries.

Default = 2

1F93-01 NMT_EPLNodeID_REC.NodeID_U8

[Unsigned8, ro]

This sub-Index holds the device's current POWERLINK Node ID. It is provided by hardware settings (dip switch), refer to the "4.4.1 Node address (Node ID): DIP A (Figure 8)" section on page 45. By default the POWERLINK Node ID is set to "1".

Default = 1 (min. = 1, max. = 254)

1F93-02 NMT_EPLNodeID_REC.NodeIDByHW_BOOL

[Boolean, ro]

This sub-Index displays the Node ID setup mode of the device. It is set up during system initialisation.

- TRUE (1) = the device's POWERLINK Node ID is set up exclusively by HW.
- FALSE (0) = the device's POWERLINK Node ID is set up exclusively by SW.

Default = TRUE (1) (min. = 0 (FALSE), max. = 1 (TRUE))

1F98-00 NMT_CycleTiming_REC

[Unsigned8, const]

The following objects provides node specific timing parameters, that influence the POWERLINK cycle timing. This sub-Index contains the number of entries.

Default = 8

1F98-01 NMT_CycleTiming_REC.IsochrTxMaxPayload_U16

[Unsigned16, const]

It provides the device specific upper limit for payload data size expressed in bytes of isochronous messages to be transmitted by the device.

On all nodes, the sub-Index limits the size of the PRes frame issued by the node (sub-Index **1F98-05 NMT_CycleTiming_REC.PResActPayloadLimit_U16**, refer below). Additionally on the MN, the size of transmitted PReq messages (object **NMT_MNPREqPayloadLimitList_AU16**) is affected.

The limit is set up by the device firmware during system initialisation.

Default = 1490 (min. = 36, max. = 1490)

1F98-02 NMT_CycleTiming_REC.IsochrRxMaxPayload_U16

[Unsigned16, const]

It provides the device specific upper limit for payload data size expressed in bytes of isochronous messages to be received by the device.

On all nodes, the sub-Index limits the size of the PRes frames received by the node (object **1F8D-00 NMT_PResPayloadLimitList_AU16**). Additionally on the CN, the size of the received PReq message (sub-Index **1F98-04 NMT_CycleTiming_REC.PReqActPayloadLimit_U16**, refer below) is affected.

The limit is set up by the device firmware during system initialisation.

Default = 1490 (min. = 36, max. = 1490)

1F98-03 NMT_CycleTiming_REC.PresMaxLatency_U32

[Unsigned32, const]

It provides the maximum time expressed in ns, that is required by the CN to respond to PReq.

The value is set up by the device firmware during system initialisation.

Default = 1000 (min. = 0, max. = 4294967295)

1F98-04 NMT_CycleTiming_REC.PReqActPayloadLimit_U16

[Unsigned16, rw]

It provides the configured PReq payload data slot size expressed in bytes expected by the CN. The payload data slot size plus headers gives the size of the PReq frame. The data slot can be filled by PDO data up to this limit.

Default = 36 (min. = 36, max. = 1490)



NOTE

This results in a fixed frame size regardless of the size of PDO data used.

1F98-05 NMT_CycleTiming_REC.PResActPayloadLimit_U16

[Unsigned16, rw]

It provides the configured PRes payload data slot size expressed in bytes sent by the CN. The payload data slot size plus headers gives the size of the PRes frame. The data slot can be filled by PDO data up to this limit.

Default = 36 (min. = 36, max. = 1490)


NOTE

This results in a fixed frame size regardless of the size of PDO data used.

1F98-06 NMT_CycleTiming_REC.ASndMaxLatency_U32

[Unsigned32, const]

It provides the maximum time in ns, that is required by the CN to respond to SoA.

The value is set up by the device firmware during system initialisation.

Default = 1000 (min. = 0, max. = 4294967295)

1F98-07 NMT_CycleTiming_REC.MultiplCycleCnt_U8

[Unsigned8, rw]

This sub-Index describes the length of the multiplexed cycle in multiples of the POWERLINK cycle.

The value is upper limited by the MN's device description entry **D_NMT_MNMultiplCycMax_U8**. It is the same in all nodes of the segment.

If **1F98-07 NMT_CycleTiming_REC.MultiplCycleCnt_U8** is zero (0), there is no support of multiplexed cycle on the network.

Default = 0 (min. = 0, max. = 255)

1F98-08 NMT_CycleTiming_REC.AsyncMTU_U16

[Unsigned16, rw]

This sub-Index describes the maximum asynchronous frame size expressed in bytes. The value applies to Asnd frames as well as to UDP/IP and other legacy Ethernet type frames. For this reason the value describes the length of the complete Ethernet frame minus 14 bytes Ethernet header and 4 byte checksum.

It is upper limited by the **1030-04 NMT_InterfaceGroup_0h_REC.InterfaceMtu_U16** values of all devices in the segment. This limit is 18 bytes less than the minimum **InterfaceMTU_U16** value provided by any node in the segment. **AsyncMTU_U16** can grow up to **C_DLL_MAX_ASYNC_MTU**.

AsyncMTU_U16 is equal in all nodes of the segment.

This sub-Index is valid in all NMT states.

Default = 300 (min. = 300, max. = 1500)

1F99-00 NMT_CNBasicEthernetTimeout_U32

[Unsigned32, rw]

It provides the time expressed in μ s to be applied before changing from **NMT_CS_NOT_ACTIVE** state to **NMT_CS_BASIC_ETHERNET** state.

Please note that MN and CN startup timing has to be well balanced. System power up sequence has to be considered.

Value "0" means that the state never changes to **NMT_CS_BASIC_ETHERNET**. If other than "0", the value shall be greater than **1006-00 NMT_CycleLen_U32**.

To avoid erroneous change over to **NMT_CS_BASIC_ETHERNET** state at system startup, **1F99-00 NMT_CNBasicEthernetTimeout_U32** must be greater than the **NMT_BootTime_REC.MNWaitNotAct_U32** of the MN.

Default = 5000000 (min. = 0, max. = 4294967295)

1F9B-00 NMT_MultiplCycleAssign_AU8

[Unsigned8, rw]

These objects assign the nodes to the particular POWERLINK cycles of the multiplexed cycle period defined by **1F98-07 NMT_CycleTiming_REC.MultiplCycleCnt_U8**. The value has to be equal in all nodes of the segment. This sub-Index contains the number of entries.

Default = 254 (min. = 1, max. = 254)

1F9B-01 NMT_MultiplCycleAssign_AU8.CycleNo

[Unsigned8, rw]

Each sub-Index in the array corresponds to the node with the Node ID equal to the sub-Index. The sub-Index value is valid only if there is a multiplexed node assigned to the Node ID by index **NMT_NodeAssignment_AU32[sub-Index]** bits 0, 1 and 9.

This object defines the POWERLINK cycle index in the multiplexed cycle, when the respective nodes are accessed. If the value is "0", the node is accessed continuously.

Default = 0 (min. = 0, max. = 255)

1F9E-00 NMT_ResetCmd_U8

[Unsigned8, rw]

This is used to initiate the reset of a node.

Setting **1F9E-00 NMT_ResetCmd_U8** to the NMT Command ID **NMTResetNode**, **NMTResetConfiguration**, **NMTResetCommunication** or **NMTSwReset** will trigger the node internal generation of a respective NMT command to itself.

1F9E-00 NMT_ResetCmd_U8 is automatically reset to **NMTInvalidService** by the node when the reset has been completed.

On read access, **1F9E-00 NMT_ResetCmd_U8** always shows **NMTInvalidService**.

If applied in **NMT_CS_EPL_MODE** state, resets by **1F9E-00 NMT_ResetCmd_U8** may violate the NMT rules and stimulate DLL and NMT Guarding errors.

Default = 255 (min. = 0, max. = 255)

7.16.2 Manufacturer Specific Profile Area objects

2101-00 HMS Serial Number

[Unsigned32, ro]

It shows the serial number of the HMS module.

Value = device dependent

2102-00 HMS_FW_Major

[Unsigned8, ro]

The HMS firmware revision number consists of a major revision number and a minor revision number. In this object the major revision number is shown.

Value = device dependent

2103-00 HMS_FW_Minor

[Unsigned8, ro]

The HMS firmware revision number consists of a major revision number and a minor revision number. In this object the minor revision number is shown.

Value = device dependent

2104-00 HMS_FW_Build

[Unsigned8, ro]

It shows the firmware build number of the HMS module.

Value = device dependent

2105-00 Position Offset

[Signed32, ro]

This variable defines the difference between the position value transmitted by the device and the real position: $\text{real position} - \text{preset}$. As soon as the preset value **2211-00 Preset-pulse** is entered, the calculated offset is automatically stored on memory. The value is expressed in pulses.

2106-00 Real Speed [rpm]

[Signed32, ro]

This object shows the current speed of the device expressed in revolutions per minute (rpm), updated at every second.

2107-00 Electronics Temperature [°C]

[Signed8, ro]

This variable shows the temperature of the electronics as detected by an internal probe. The value is expressed in degrees Celsius (°C). The minimum detectable temperature is -20°C.

2108-00 Motor Temperature [°C]

[Signed8, ro]

This variable shows the temperature of the motor as detected by an internal probe. The value is expressed in degrees Celsius (°C). The minimum detectable temperature is -20°C.

2109-00 Real Current

[Unsigned32, ro]

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

210A-00 Following error [pulse]

[Unsigned32, 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 **2207-00 Max following error-pulse** parameter, then the **Following error** alarm is triggered and the unit stops. The value is expressed in pulses.

210B-00 Pos. Limit Switch [pulse]

[Signed32, ro]

This is the **SW limit switch +** value (maximum positive limit) calculated according to the values set next to the **2211-00 Preset-pulse** and **220C-00 Max delta pos-pulse** objects. When the maximum forward limit is reached, the condition is signalled through the **SW limit switch +** status bit 3 of the **2202-00 Status word**.

SW limit switch + = 2211-00 Preset-pulse + 220C-00 Max delta pos-pulse.

The value is expressed in pulses.

Refer also to the EXAMPLE 1 in the "6.4 Distance per revolution, Preset, Positive delta and Negative delta" section on page 55.

210C-00 Neg. Limit Switch [pulse]

[Signed32, ro]

This is the **SW limit switch** - value (maximum negative limit) calculated according to the values set next to the **2211-00 Preset-pulse** and **220D-00 Max delta neg-pulse** objects. When the maximum backward limit is reached, the condition is signalled through the **SW limit switch** - status bit 4 of the **2202-00 Status word**.

SW limit switch - = **2211-00 Preset-pulse** - **220D-00 Max delta neg-pulse**.

The value is expressed in pulses.

Refer also to the EXAMPLE 1 in the "6.4 Distance per revolution, Preset, Positive delta and Negative delta" section on page 55.

210D-00 Parameter Error List

[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, 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	2204-00 Distance per revolution-pulse
2	220A-00 Acceleration-rev/s²
3	220B-00 Deceleration-rev/s²
4	220C-00 Max delta pos-pulse
5	220D-00 Max delta neg-pulse
6	220E-00 Jog speed-rpm
7	220F-00 Work speed-rpm
8	2210-00 Count direction 0=CW,1=CCW
9	2211-00 Preset-pulse
10	2212-00 Jog step-pulse
11	2208-00 Proportional gain
12	2209-00 Integral gain
13	2206-00 Settling time-ms
14	2207-00 Max following error-pulse

15 ... 31	Not used
-----------	----------

210E-00 Node ID

[Unsigned8, ro]

This is meant to show the node address set in the RD1xA unit; the node address has to be set through the provided rotary switches inside the actuator's enclosure. For any information on setting the node address refer to the "4.4.1 Node address (Node ID): DIP A (Figure 8)" section on page 45. The default value is "1".

210F-00 Alarms List

[Unsigned16, ro]

Bit	Function	bit = 0	bit = 1
0	Machine data not valid	Alarm not active	Alarm active
1	Flash memory error	Alarm not active	Alarm active
2	Counting error	Alarm not active	Alarm active
3	Following error	Alarm not active	Alarm active
4	Encoder not synchronized	Alarm not active	Alarm active
5	Target not valid	Alarm not active	Alarm active
6	Emergency	Alarm not active	Alarm active
7	Overcurrent	Alarm not active	Alarm active
8	Electronics Overtemperature	Alarm not active	Alarm active
9	Motor Overtemperature	Alarm not active	Alarm active
10	Undervoltage	Alarm not active	Alarm active
11 ... 13	not used		
14	Hall sequence	Alarm not active	Alarm active
15	Overvoltage	Alarm not active	Alarm active

This object provides information on the alarm messages supported by the actuator. An alarm will be set if a malfunction in the actuator or a wrong parametrization could lead to an incorrect operation. If an alarm occurs, the relevant bit is set to logical high (1) until the alarm is cleared and the actuator is able to run properly.

The available alarm error codes are listed hereafter:

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 [210D-00 Parameter Error List](#) object.

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 [2207-00 Max following error-pulse](#) object; we suggest reducing the dynamics of the movements (acceleration, deceleration, velocity).

Encoder 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. Set a proper value next to the [2201-00 Target position](#) object.

Emergency

bit 6 Bit 7 **Emergency** in the [2200-00 Control Word](#) has been forced to low value (0); or alarms are active in the unit.

Overcurrent

bit 7 Motor overcurrent; we suggest reducing the dynamics of the movements (acceleration, deceleration, velocity).

Electronics Overtemperature

bit 8 The temperature of the MOSFETs detected by an internal probe is exceeding the maximum ratings (see [2107-00 Electronics Temperature \[°C\]](#) on page 118). 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 [2108-00 Motor](#)

Temperature [°C] on page 118). 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 and the power of the external power pack is suitable.

bits 11 ... 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 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, bit 3 in the **2200-00 Control Word**. 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 **210D-00 Parameter Error List**), the normal work status can be restored only after having set proper values. The **Flash memory error** and **Encoder not synchronized** alarms cannot be reset.

2200-00 Control Word

[Unsigned16, rw]

It contains the commands to be sent in real time to the CN device in order to manage it. It is mapped in the **1600-01 PDO_RxMappParam_00h_AU64.ObjectMapping** object that contains the mapping parameters for the PDOs the POWERLINK device is able to receive, see on page 102.

Byte 0

Jog +

bit 0

If the bit 4 **Incremental jog** = 0, as long as **Jog +** = 1, the CN 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 **2212-00 Jog step-pulse** 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 **220E-00 Jog speed-rpm**, **220A-00 Acceleration-rev/s²** and **220B-00 Deceleration-rev/s²** objects respectively. For a detailed description of the jog control see on page 52.



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 **2212-00 Jog step-pulse** 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 **220E-00 Jog speed-rpm**, **220A-00 Acceleration-rev/s²** and **220B-00 Deceleration-rev/s²** objects respectively. For a detailed description of the jog control see on page 52.



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 CN device is allowed to execute the movements as commanded. If, while the unit is running, this bit switches to "0", then the CN device must stop and

execute the deceleration procedure set in **220B-00 Deceleration-rev/s2**. 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 **210D-00 Parameter Error List**), the normal work status can be restored only after having set proper values. The **Flash memory error** and **Encoder not synchronized** alarms cannot be reset.

Incremental jog bit 4

If the bit is set to "=0", the activation of the bits **Jog +** and **Jog -** causes the Slave to move as long as **Jog + / Jog -** = 1. If it is set 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 **2212-00 Jog step-pulse** item to be executed at rising edge; then the actuator stops and waits for another command.

bit 5

Not used.

Start bit 6



When the bit value switches from "0" to "1", the device moves in order to reach the set target position (see **2201-00 Target position** on page 126). For a complete description of the position control see on page 51. This bit must be switched back to "0" after the device has started.

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

In normal work condition this bit must be 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 208.

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 **2211-00 Preset-pulse** 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 136.



When you set a new preset value next to the **2211-00 Preset-pulse** parameter, the entered value is not activated automatically, thus the **Setting the preset** bit operation is always required.

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 54.

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 the bit to "=1" causes the brake to be disabled and not operating; setting the bit "=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.

2201-00 Target position

[Signed32, rw]

This object is mapped in the **1600-02 PDO_RxMappParam_00h_AU64.ObjectMapping** object that contains the

mapping parameters for the PDOs the POWERLINK device is able to receive, see on page 102.

It sets the position to be reached, otherwise referred to as commanded position. The value is expressed in pulses. When the **Start** command is sent while **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 **2205-00 Position tolerance** item, the bit 8 **Target position reached** in the **2202-00 Status word** goes high ("=1"). When the position is within the tolerance window limits set next to the **2205-00 Position tolerance** item, after the delay set next to the **2206-00 Settling time-ms** item, the bit 0 **Axis in position** in the **2202-00 Status word** goes high ("=1").

For more information refer also to the "Positioning: position and speed control" section on page 53.

Default = 0 (min. = 0, max. = within **210B-00 Pos. Limit Switch [pulse]** / **210C-00 Neg. Limit Switch [pulse]**)



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 **2201-00 Target position** item. See also on page 53.



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.

Should the device be disconnected from the POWERLINK network while it is moving (for instance because of a broken cable or a faulty wiring), the device will stop moving immediately and enter the emergency state while the PLC will enter the service state; then a warm restart will be required.

2202-00 Status word

[Unsigned16, ro]

This object provides information about the current state of the device. It is mapped in the **1A00-01 PDO_TxMappParam_00h_AU64.ObjectMapping** object that contains the mapping parameters for the PDOs the POWERLINK device is able to transmit, see on page 103.

Byte 0

Axis in position

bit 0

The value is "1" when the device reaches and keeps the commanded position (**2201-00 Target position**) for the time set next to the **2206-00 Settling time-ms** item. It is kept active until the position error is lower than **2205-00 Position tolerance**. For further information please refer to the "Positioning: position and speed control" section on page 53.

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: the PWM is active and the axis is under closed-loop control (for instance, while reaching a target position or using a jog). 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). The maximum positive limit can be read next to the **210B-00 Pos. Limit Switch [pulse]** object. For more information see the **220C-00 Max delta pos-pulse** object.

SW limit switch -

bit 4

The value is "1" should it happen that the device reaches the maximum negative limit (negative limit switch). The maximum negative limit can be read next to the **210C-00 Neg. Limit Switch [pulse]** object. For more information see the **220D-00 Max delta neg-pulse** object.

Alarm

bit 5

The value is "1" when an alarm occurs, see details in the **210F-00 Alarms List** variable on page 120.

Axis running

bit 6

Theoretical state of the axis.

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 **2201-00 Target position** object (it is within the limits set next to the **2205-00 Position tolerance** object). The bit is kept active until a new **2201-00 Target position** or the **Alarm reset** command are sent. For more information refer also to the "Positioning: position and speed control" section on page 53.

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 44 and the "4.5.1 JOG + and JOG - buttons (Figure 9)" section on page 47.

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 44 and the "4.5.1 JOG + and JOG - buttons (Figure 9)" section on page 47.

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 44 and the "4.5.2 PRESET button (Figure 9)" section on page 48.

PWM saturation

bit 12

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

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 54.

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 54.

IN 2 = 0 input 1 low (not active)

IN 2 = 1 input 1 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 54.

IN 3 = 0 input 1 low (not active)

IN 3 = 1 input 1 high (active)

2203-00 Position

[Signed32, ro]

Current position of the device expressed in pulses. This object is mapped in the **1A00-02 PDO_TxMappParam_00h_AU64.ObjectMapping** object that contains the mapping parameters for the PDOs the POWERLINK device is able to transmit, see on page 104.

2204-00 Distance per revolution-pulse

[Unsigned16, rw]

2204-00 Distance per revolution-pulse sets the number of pulses per each complete revolution of the actuator's shaft. This parameter is useful to relate the revolution of the shaft and a linear measurement. For example: the unit is joined to a worm screw having a 5 mm (0.197") pitch; by setting **2204-00 Distance per revolution-pulse** = 500, at each shaft revolution the system performs a 5 mm (0.197") pitch with one-hundredth of a millimetre resolution. Default = 1024 (min. = 1, max. = 1024).



WARNING

After having changed this parameter, then you must set a new value also next to the **2211-00 Preset-pulse** object. For a detailed explanation see on page 55 and the relevant objects.

Please note that the objects listed hereafter are closely related to the **2204-00 Distance per revolution-pulse** object as they are all expressed in pulses; hence when you change the value in **2204-00 Distance per revolution-pulse** also the value in each of them necessarily changes. They are: **2201-00 Target position**, **2205-00 Position tolerance**, **2207-00 Max following error-pulse**, **220C-00 Max delta pos-pulse** and **220D-00 Max delta neg-pulse**. The **2203-00 Position** value is also affected.



NOTE

If **2204-00 Distance per revolution-pulse** 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.

2205-00 Position tolerance

[Unsigned16, rw]

This object defines the tolerance window limits for the **2201-00 Target position** value. As soon as the axis is within the tolerance window limits, the bit 8 **Target position reached** in the **2202-00 Status word** goes high ("=1"). When the axis is within the tolerance window limits for the time set in the **2206-00 Settling time-ms** object, the bit 0 **Axis in position** in the **2202-00 Status word** goes high ("=1"). This parameter is expressed in pulses. See also the "Positioning: position and speed control" section on page 53.

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

2206-00 Settling time-ms

[Unsigned16, rw]

It represents the time for which the axis has to be within the tolerance window limits set in the **2205-00 Position tolerance** object before the state is signalled through the **Axis in position** status bit 0 of the **2202-00 Status word**. The parameter is expressed in milliseconds. See also the "Positioning: position and speed control" section on page 53.

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

2207-00 Max following error-pulse

[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).

2208-00 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).

2209-00 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).

220A-00 Acceleration-rev/s²

[Unsigned16, rw]

This parameter defines the maximum acceleration value that has to be used by the device when reaching both the **220E-00 Jog speed-rpm** and the **220F-00 Work speed-rpm**. The parameter is expressed in revolutions per second² [rev/s²]. See also the "6.2 Movements: jog and positioning" section on page 52.

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

220B-00 Deceleration-rev/s²

[Unsigned16, rw]

This parameter defines the maximum deceleration value that has to be used by the device when stopping. The parameter is expressed in revolutions per second² [rev/s²]. See also the "6.2 Movements: jog and positioning" section on page 52.

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

220C-00 Max delta pos-pulse

[Unsigned32, rw]

This value 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 **2202-00 Status word** (the bit is forced high). The parameter is expressed in pulses.

SW limit switch + = **2211-00 Preset-pulse** + **220C-00 Max delta pos-pulse**. The maximum positive limit can be read next to the **210B-00 Pos. Limit Switch [pulse]** object.

For further information please refer to the "6.4 Distance per revolution, Preset, Positive delta and Negative delta" section on page 55.

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 **2204-00 Distance per revolution-pulse** = 1,024 and **2211-00 Preset-pulse** = 0, the maximum acceptable value for **220C-00 Max delta pos-pulse** 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 (see the default value)

When **2204-00 Distance per revolution-pulse** = 256 and **2211-00 Preset-pulse** = 0, the maximum acceptable value for **220C-00 Max delta pos-pulse** 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, Positive delta and Negative delta" section on page 55.



WARNING

Every time **2204-00 Distance per revolution-pulse** and **2211-00 Preset-pulse** objects are changed, **220C-00 Max delta pos-pulse** and **220D-00 Max delta neg-pulse** values have to be checked carefully. Each time you change the value in **2204-00 Distance per revolution-pulse** you must then update the value in **2211-00 Preset-pulse** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **2211-00 Preset-pulse** 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 **220C-00 Max delta pos-pulse** and **220D-00 Max delta neg-pulse** objects. For a detailed explanation see on page 55.

220D-00 Max delta neg-pulse

[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 **2202-00 Status word** (the bit is forced high). The parameter is expressed in pulses.

SW limit switch - = **2211-00 Preset-pulse** - **220D-00 Max delta neg-pulse**. The maximum negative limit can be read next to the **210C-00 Neg. Limit Switch [pulse]** object.

For further information please refer to the "6.4 Distance per revolution, Preset, Positive delta and Negative delta" section on page 55.

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 **2204-00 Distance per revolution-pulse** = 1,024 and **2211-00 Preset-pulse** = 0, the maximum acceptable value for **220D-00 Max delta neg-pulse** is:

(1,024 steps per revolution * 512 revolutions) - 1 step - 1,024 steps (i.e. 1 revolution for safety reasons) = 523 263 (see the default value)

When **2204-00 Distance per revolution-pulse** = 256 and **2211-00 Preset-pulse** = 0, the maximum acceptable value for **220D-00 Max delta neg-pulse** 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, Positive delta and Negative delta" section on page 55.



WARNING

Every time **2204-00 Distance per revolution-pulse** and **2211-00 Preset-pulse** objects are changed, **220C-00 Max delta pos-pulse** and **220D-00 Max delta neg-pulse** values have to be checked carefully. Each time you change the value in **2204-00 Distance per revolution-pulse** you must then update the value in **2211-00 Preset-pulse** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **2211-00 Preset-pulse** 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 **220C-00 Max delta pos-pulse** and **220D-00 Max delta neg-pulse** objects. For a detailed explanation see on page 55.

220E-00 Jog speed-rpm

[Unsigned16, rw]

This parameter contains the maximum speed the device is allowed to reach when using the **Jog +** and **Jog -** functions (see the **2200-00 Control Word** object). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 52.

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

220F-00 Work speed-rpm

[Unsigned16, rw]

This parameter contains the maximum speed the device is allowed to reach in automatic work mode (movements are controlled using **Start** and **Stop** commands -see the **2200-00 Control Word** object- and are performed in order

to reach the position set in **2201-00 Target position**). The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 53.

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

2210-00 Count direction 0=CW,1=CCW

[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 **2211-00 Preset-pulse** object and then check the values set next to the **220C-00 Max delta pos-pulse** and **220D-00 Max delta neg-pulse** objects.

2211-00 Preset-pulse

[Signed32, rw]

Use this object 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 object 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 preset value is activated. When you set a new preset value next to this **2211-00 Preset-pulse** object, the entered value is not saved and activated automatically. To activate the preset, you must switch the **Setting the preset** bit 11 in the **2200-00 Control Word** from logic level low ("0") to logic level high ("1"). After executing the preset command, you are required to save the parameters in order to store on memory the preset. If

you do not save the new preset value, at next power-on the system will calculate the actuator position using the preset value previously saved. To save the preset value, you must use the **Save parameters** bit 9 in the **2200-00 Control Word**. As soon as the new preset value is entered, the calculated offset (**2105-00 Position Offset**) is automatically stored on memory. The value is expressed in pulses.

To execute the preset you must transmit the value to the **2211-00 Preset-pulse** object in the asynchronous phase via SDO when the actuator is in operational state (**NMT_CS_OPERATIONAL**) and then execute the **Save parameters** function (see the **Save parameters** function on page 125).

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 125.



WARNING

A new value must be set in the **2211-00 Preset-pulse** object every time the **2204-00 Distance per revolution-pulse** value is changed. After having entered a new value in **2211-00 Preset-pulse** 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 **220C-00 Max delta pos-pulse** and **220D-00 Max delta neg-pulse** items. For a detailed explanation see on page 55.

2212-00 Jog step-pulse

[Unsigned16, rw]

When the incremental jog function is enabled (bit 4 **Incremental jog** in the **2200-00 Control Word** = 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).



NOTE

Always save the new values after setting any object in order to store them in the non-volatile memory permanently. Use the **Save parameters** function available in the **2200-00 Control Word** object, see on page 125.

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

7.17 SDO abort codes

Here follows the list and meaning of the SDO abort codes indicated by POWERLINK but not necessarily supported by the manufacturer. They comply with the CANopen SDO abort codes (for CANopen SDO abort codes refer to the "SDO abort transfer protocol" section in the "CiA Draft Standard 301" document). For complete information on the POWERLINK implemented abort codes please refer to the "SDO Abort Codes" section in the "EPSG DSP 301 V1.2.0" document.

Abort code	Description
0503 0000h	Reserved
0504 0000h	SDO protocol timed out
0504 0001h	Client/server Command ID not valid or unknown
0504 0002h	Invalid block size
0504 0003h	Invalid sequence number
0504 0004h	Reserved
0504 0005h	Out of memory
0601 0000h	Unsupported access to an object
0601 0001h	Attempt to read a write-only object
0601 0002h	Attempt to write a read-only object
0602 0000h	Object does not exist in the object dictionary
0604 0041h	Object cannot be mapped to the PDO
0604 0042h E_PDO_MAP_ OVERRUN	The number and length of the objects to be mapped would exceed PDO length
0604 0043h	General parameter incompatibility
0604 0044h E_NMT_INVALID _HEARTBEAT	Invalid heartbeat declaration
0604 0047h	General internal incompatibility in the device
0606 0000h	Access failed due to an hardware error
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-Index does not exist
0609 0030h	Value range of parameter exceeded (only for write access)
0609 0031h	Value of parameter written too high
0609 0032h	Value of parameter written too low
0609 0036h	Maximum value is less than minimum value

0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application
0800 0021h	Data cannot be transferred or stored to the application because of local control
0800 0022h	Data cannot be transferred or stored to the application because of the present device state
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error)
0800 0024h E_CFM_DATA_ SET_EMPTY	EDS, DCF or Concise DCF Data set empty

The abort codes not listed here are reserved.

7.18 Storing parameters

To store the parameters you must set the bit 9 **Save parameters** in the **2200-00 Control Word** object in the asynchronous phase via SDO when the actuator is in operational state (**NMT_CS_OPERATIONAL**).

7.19 Restoring default parameters



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.

Default values are provided to each parameter of the device and are preset at the factory by Lika Electronic engineers. The first time you install the actuator, it will operate using the default values. They allow the operator to run the CN device for standard and safe operation. They are plainly not optimized for specific application yet they provide maximum performance for most systems. To suit the specific application requirements it may be advisable and even necessary to enter new parameters instead of the factory default settings.

There could be exceptional circumstances where it would be necessary for you to restore the default values of the settable parameters. When this is the case, you must set the bit 10 **Load default parameters** in the **2200-00 Control Word** object in the asynchronous phase via SDO when the actuator is in operational state (**NMT_CS_OPERATIONAL**) and then execute the **Save parameters** function (see the **Save parameters** function on page 125).

**NOTE**

When you restore the default values, please always consider that:

- the actuator parameters will be restored to the default values;
- the actuator offset will be reset.

**WARNING**

The execution of this command causes all the values which have been set previously next to each parameter to be overwritten!

**NOTE**

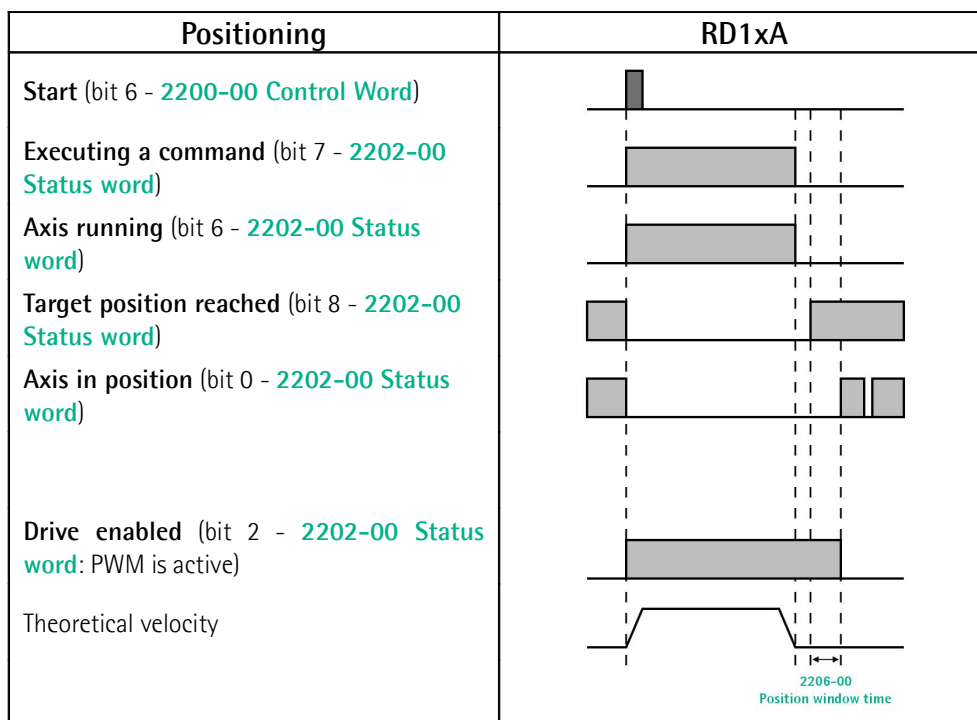
The complete list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 208.

7.20 Executing the preset

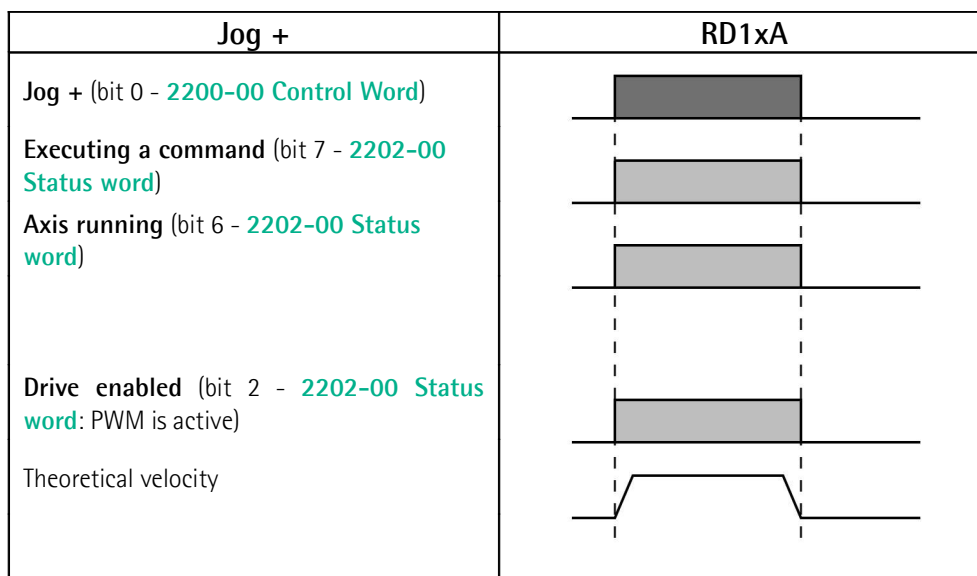
For complete information on executing the preset please refer to the "7.4.9 Preset setting program" section on page 75.



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.

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 launch the program is **SW_RD1xA_LKC742_Vx.x.EXE** and is available in the enclosed documentation or at the address www.lika.biz > **ROTARY ACTUATORS** > **ROTARY ACTUATORS/POSITIONING UNITS (DRIVECOD)**. The program is designed to be installed simply by copying the executable file to the desired location and there is **no installation** process. To launch it just double-click the file icon. To close the program press the **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.

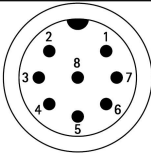
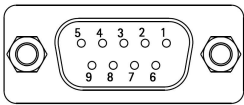
On the DRIVECOD side the cable must be connected to the M12 8-pin male connector service serial port. See the "Electrical connections" section on page 36.

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



NOTE

If you use the EXC-USB4-S54-GN-2-M12MC-S54 USB connection cable, you are required to install the drivers of the USB Serial Converter and the USB Serial Port first. The drivers are available in the Software folder of the actuator and downloadable from Lika's web site.

			
Function	RS-232 M12 8-pin male connector	9-pin D-SUB female connector	Function
TD	6	2	RD
RD	7	3	TD
OVdc	8	5	OVdc

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:

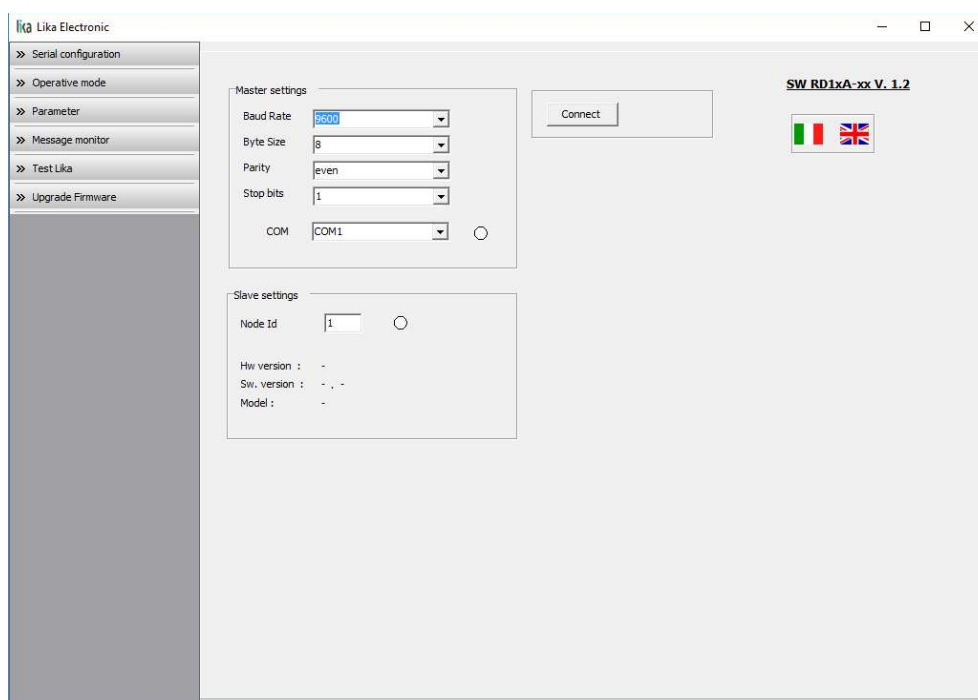
RS-232 Modbus



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

The MODBUS address is according to the rotary switches setting. To set the node address of the RD1xA device refer to the "4.4.1 Node address (Node ID): DIP A (Figure 8)" section on page 45. See also the "8.2 "Serial configuration" page" section right hereafter.

8.2 "Serial configuration" page

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



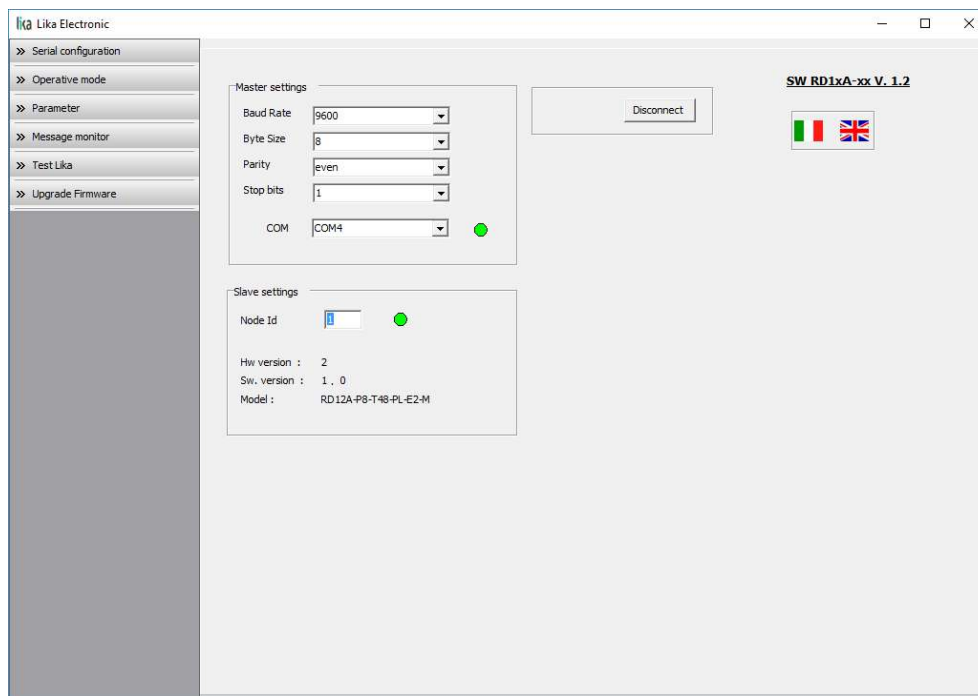
First of all this page allows the operator to choose the language used to display texts and items in the user interface. Click on the **Italian flag**  to choose the Italian language; click on the **UK flag**  to choose the English language.

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 (the MODBUS address is according to the rotary switches setting, see on page 45).

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.

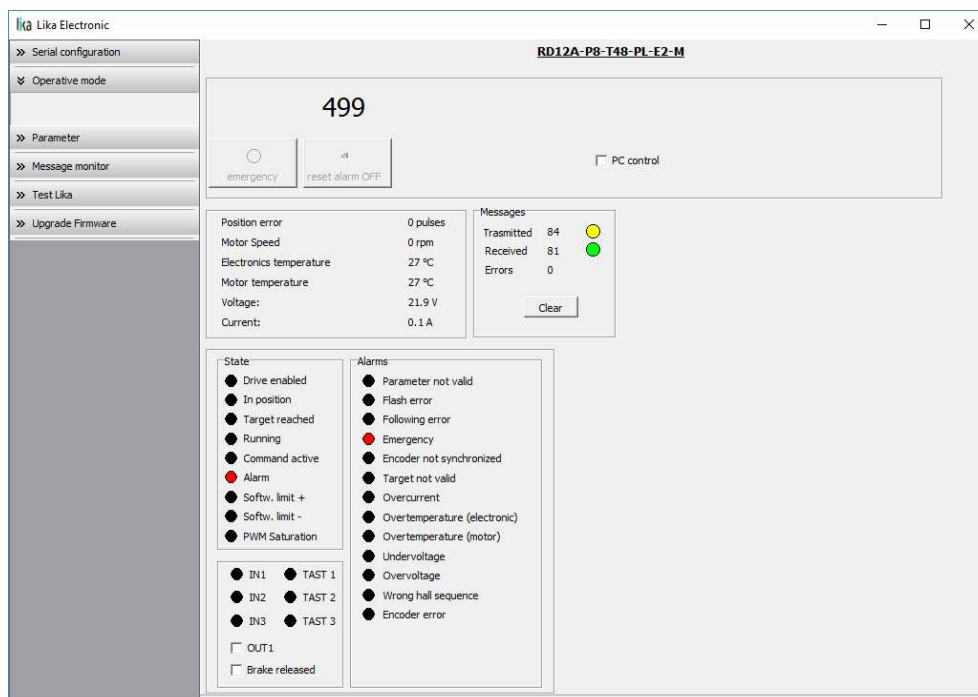
The green light next to the **COM** item indicates that the COM port is open successfully.

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

The actuator's model is shown next to the **Model** item, see the order code.

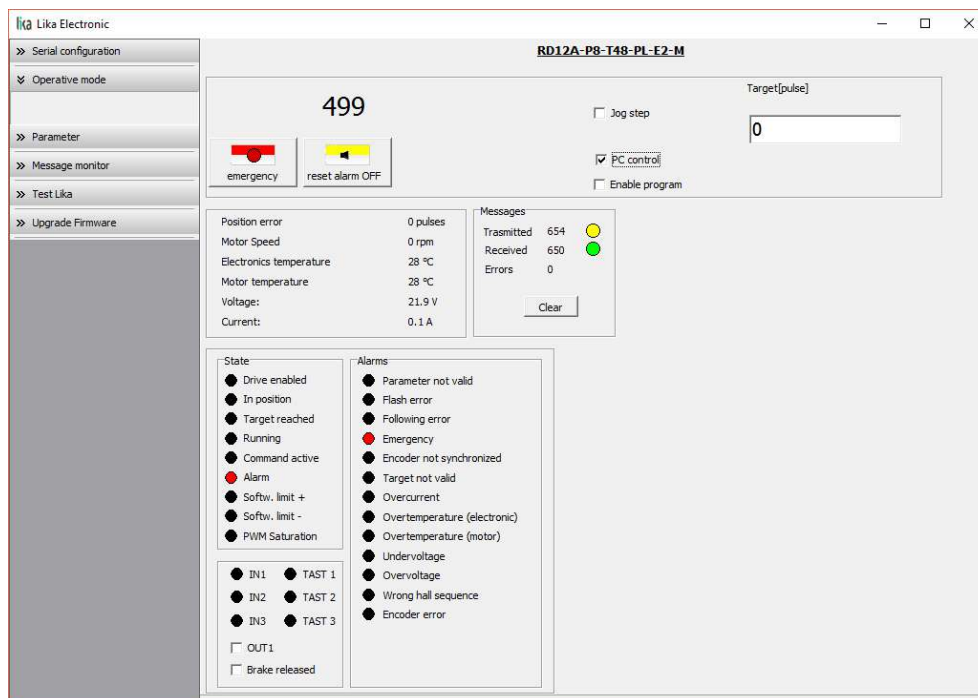
8.3 "Operative mode" page

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 POWERLINK 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 182).

Some commands and functions will become available for use as shown in the following page:



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 and some further buttons will become available.

The following page will appear:



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

Jog -

See the **Jog -** item on page 184.

Jog +

See the **Jog +** item on page 183.

Between the **Jog -** and **Jog +** button the current position of the axis is displayed. See the **Current position [0x02-0x03]** item on page 196.

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 185.

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 184.

Stop

Press this button to force a normal halt of the device, respecting the acceleration and deceleration values. See the **Stop** item on page 184.

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 185.

Jog step

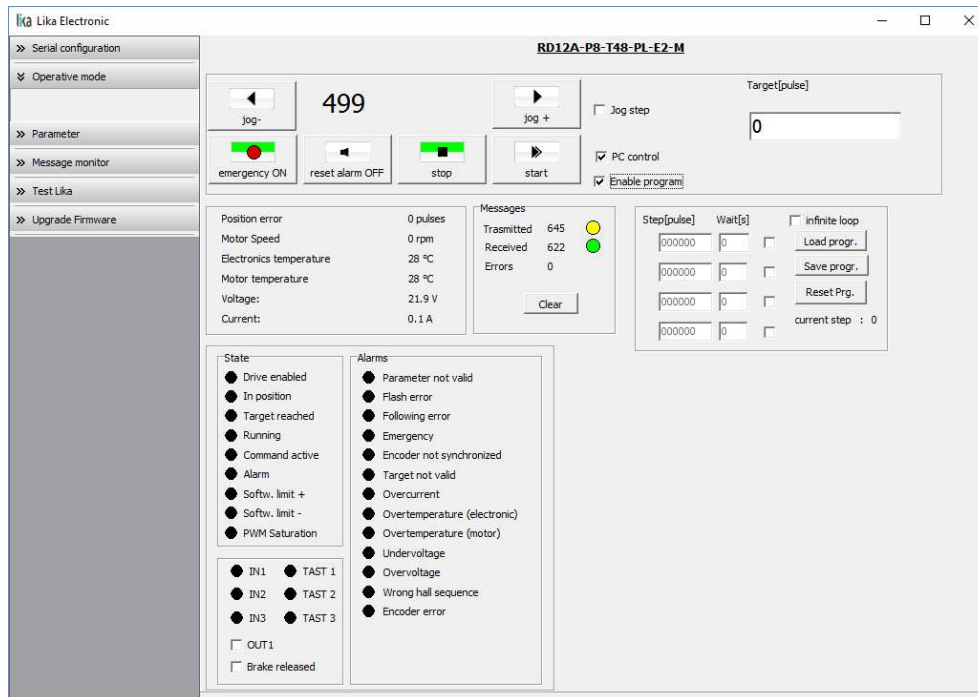
See the **Incremental jog** item on page 184.

Target [pulse]

See the **Target position [0x2B-0x2C]** item on page 188. 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 [P]** 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 the work programs of 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 buttons box the following functions are available.

Position error [pulses]

See the [Position following error \[0x05–0x06\]](#) item on page 196.

Motor Speed [rpm]

See the [Current velocity \[0x04\]](#) item on page 196.

Electronics temperature [°C]

See the [Temperature value \[0x07\]](#) item on page 196.

Motor temperature [°C]

See the [Temperature value \[0x07\]](#) item on page 196.

Voltage [V]

See the [Motor voltage \[0x0A\]](#) item on page 197.

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 197.

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 156).

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 193; for a detailed description of the alarms see the [Alarms register \[0x00\]](#) item on page 190.

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 195 and ff. For complete information on the buttons refer to the **Button 1 Jog +** item on page 194 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 187.

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]:	445030		
Jog speed motor [rpm]:	2000	10	3000
Motor work speed [rpm]:	2000	10	3000
Acceleration [rev/s²]:	10	1	500
Deceleration [rev/s²]:	10	1	500
Code sequence:	0		
Jog step length[pulses]:	1000	1	10000
Max follow error [pulses]:	1024	0	65535
Position window[pulses]:	1	0	65535
Pos. window time[ms]:	0	0	10000
Kp (position loop):	300	0	1000
Ki (position loop):	10	0	1000
SW LS+:	523263		
SW LS-:	-523263		

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 175.

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 186.

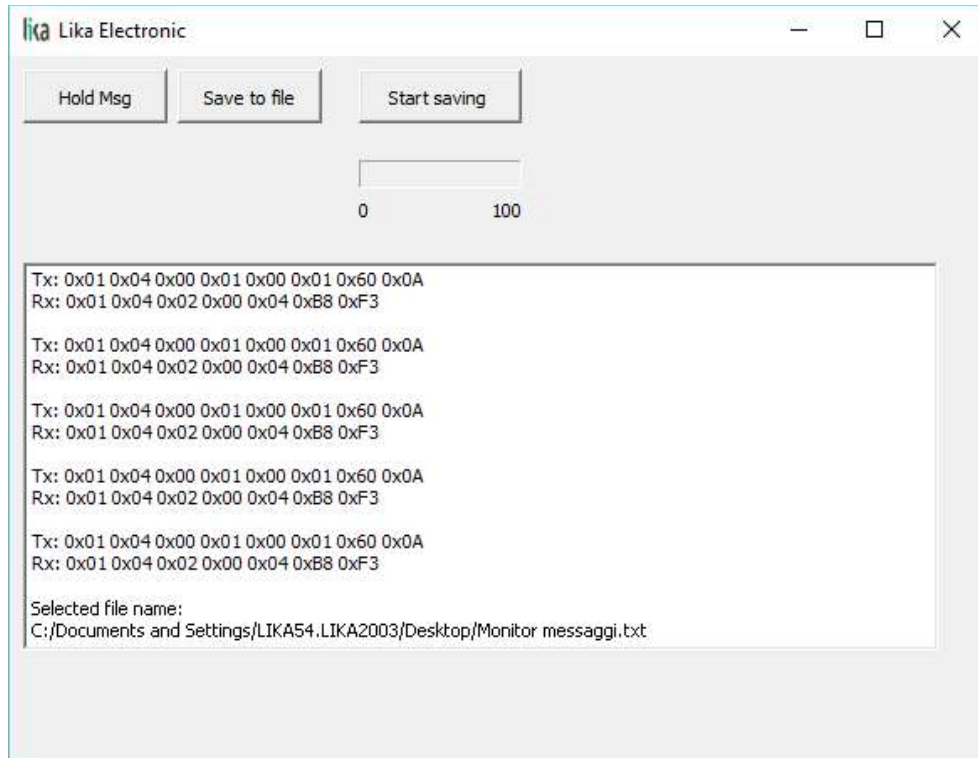
When you need to load the default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) press the **LOAD DEFAULT** button. For any further information on loading the default parameters refer to the **Load default parameters** variable on page 186. The complete list of the machine data parameters and the relevant default values as set by Lika Electronic are available on page 208.

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 178 and the **Negative delta [0x0B-0x0C]** item on page 179.

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.

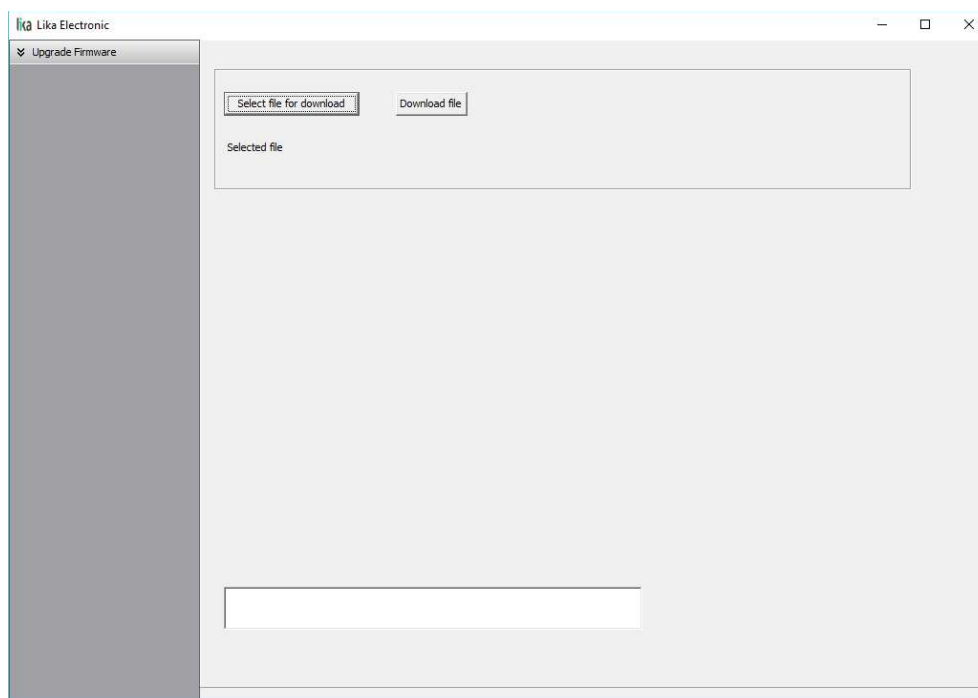
A box in the **Parameter** page offers a brief description of the communication between the Master and the Slave, by displaying the Request PDU (Transmitted) and the Response PDU (Received) messages. The fields in the box are meant to show the number of transmitted messages, the number of received messages 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 145).

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 Modbus RS-232 service port" section on page 40;
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_PL.BIN, where:

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

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.x.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.x.EXE program. It is not possible to connect to the unit through the **Serial configuration** page; the operator must enter the **Upgrade Firmware** page and restart the firmware installation process from point 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.x.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

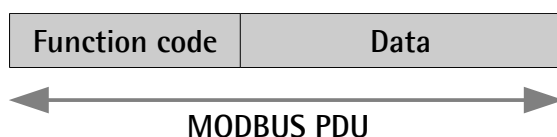
communicate with each other. The Master node initiates only one Modbus transaction at the same time.

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

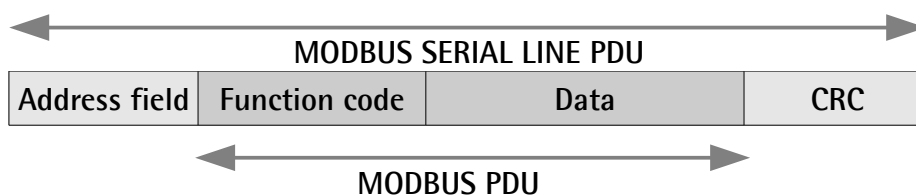
- **UNICAST mode:** in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must 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 160), 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 165.
- **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 165).
- **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 = 9600 bit/s the "silent interval" is 4 ms.

When baud rate = 19200 bit/s the "silent interval" is 2 ms.

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

Coding system: 8-bit binary

Bits per Byte: 1 start bit;

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

1 bit for parity completion (= Even);

1 stop bit.

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

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

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

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

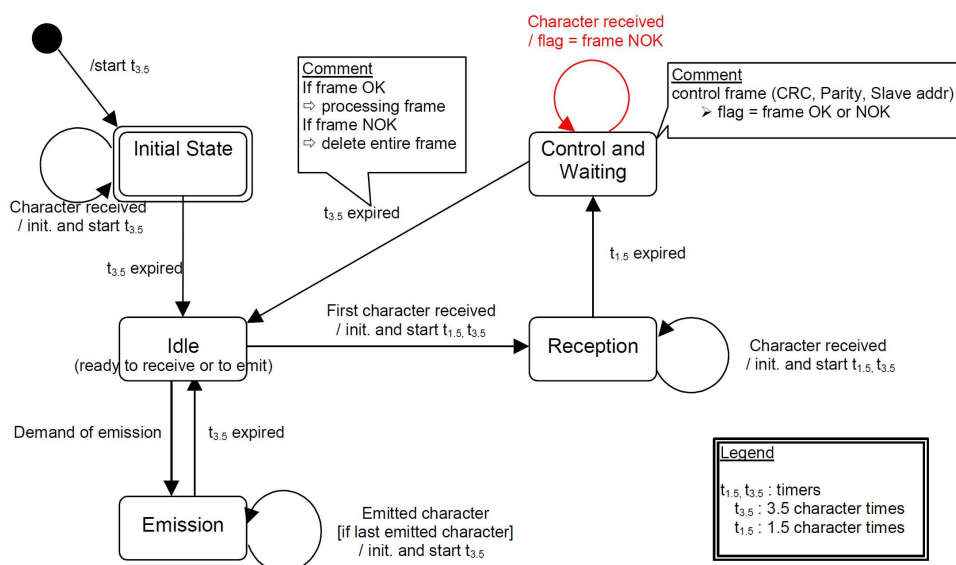
* When "No parity" is activated, the parity bit is replaced by a stop bit.

The default parity mode must be even parity.

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

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

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



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

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

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) r0

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 175.

Request PDU

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

Response PDU

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

*N = Quantity of registers

Exception Response PDU

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



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

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

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 190.

Request PDU

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

Response PDU

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

*N = Quantity of registers

Exception Response PDU

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



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

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

Quantity of Input Reg. Hi	00	Register 3 value Lo	00
Quantity of Input Reg. Lo	02	Register 4 value Hi	2F
		Register 4 value Lo	F0

As you can see in the table, 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 175.

Request PDU

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

Response PDU

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

Exception Response PDU

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



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

Request		Response	
Field name	(Hex)	Field name	(Hex)
Function	06	Function	06
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 175.

Request PDU

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

*N = Quantity of registers

Response PDU

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

Exception Response PDU

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



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

Request		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 202.



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 165.



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 183.

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 (0.197") pitch; by setting **Distance per revolution [0x00]** = 500, at each shaft revolution the system performs a 5 mm (0.197") 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 55 and relevant parameters.

Please note that the parameters listed hereafter are closely related to the **Distance per revolution [0x00]** parameter as they are all expressed in pulses; 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]**, **Jog step length [0x14]** and **Target position [0x2B-0x2C]**. The **Current position [0x02-0x03]** value is also affected.



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 53.

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

Position window time [0x02]

[Register 3, Unsigned16, rw]

It represents the time for which the axis has to be within the tolerance window limits set in the **Position window [0x01]** parameter before the state is signalled through the **Axis in position** status bit of the **Status word [0x01]**. The parameter is expressed in milliseconds. See also the "Positioning: position and speed control" section on page 53.

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 maximum acceleration value that has to be used by the device when reaching both the **Jog speed [0x0D]** and the **Work speed [0x0E]**. The parameter is expressed in revolutions per second² [rev/s²]. See also the "6.2 Movements: jog and positioning" section on page 52.

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

Deceleration [0x08]

[Register 9, Unsigned16, rw]

This parameter defines the maximum deceleration value that has to be used by the device when stopping. The parameter is expressed in revolutions per second² [rev/s²]. See also the "6.2 Movements: jog and positioning" section on page 52.

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. When the maximum forward limit is reached, the condition is signalled 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, Positive delta and Negative delta" section on page 55.

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, Positive delta and Negative delta" section on page 55.



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 55.

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. When the maximum backward limit is reached, the condition is signalled 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, Positive delta and Negative delta" section on page 55.

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:

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

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, Positive delta and Negative delta" section on page 55.



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 55.

Jog speed [0x0D]

[Register 14, Unsigned16, rw]

This parameter contains the maximum speed the device is allowed to reach when using the **Jog +** and **Jog -** functions (see the **Control Word [0x2A]** parameter). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 52.

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

Work speed [0x0E]

[Register 15, Unsigned16, rw]

This parameter contains the maximum speed the device is allowed to reach in automatic work mode (movements are controlled using the **Start** and **Stop** commands -see the **Control Word [0x2A]** parameter- and are performed in

order to reach the position set in **Target position [0x2B-0x2C]**). The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 53.

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

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 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 **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. Following a Preset operation, the Offset value is automatically stored in the memory. The value is expressed in pulses.

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"). Following a Preset operation, the Offset value is automatically stored in the memory. The value is expressed in pulses.

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 186.



WARNING

A new value must 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 55.

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 52.



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 52.



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 and execute 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** and **Encoder not synchronized** alarms 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 the bit value switches from "0" to "1", the device moves in order to reach the set target position (see **Target position [0x2B-0x2C]** on page 188). For a complete description of the position control see on page 53. This bit has to be switched back to "0" after the device has started.



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 208.

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 181.



When you set a new preset value next to the **Preset [0x12-0x13]** parameter, the entered value is not activated automatically, thus the **Setting the preset** bit operation is always required.

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 54.

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. The value is expressed in pulses. 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 53.

Default = 0 (min. = 0, max. = within **Positive delta [0x09-0x0A]** / **Negative delta [0x0B-0x0C]**)



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 53.



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 165.

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

Bit	Function	bit = 0	bit = 1
0	Machine data not valid	Alarm not active	Alarm active
1	Flash memory error	Alarm not active	Alarm active
2	Counting error	Alarm not active	Alarm active
3	Following error	Alarm not active	Alarm active
4	Encoder not synchronized	Alarm not active	Alarm active
5	Target not valid	Alarm not active	Alarm active
6	Emergency	Alarm not active	Alarm active
7	Overcurrent	Alarm not active	Alarm active
8	Electronics Overtemperature	Alarm not active	Alarm active
9	Motor Overtemperature	Alarm not active	Alarm active
10	Undervoltage	Alarm not active	Alarm active
11	Watch dog	Alarm not active	Alarm active
12 and 13	not used		
14	Hall sequence	Alarm not active	Alarm active
15	Overvoltage	Alarm not active	Alarm active

This object provides information on the alarm messages supported by the actuator. An alarm will be set if a malfunction in the actuator or a wrong parametrization could lead to an incorrect operation. If an alarm occurs, the relevant bit is set to logical high (1) until the alarm is cleared and the actuator is able to run properly.

The available alarm error codes are listed hereafter:

Byte 0

Machine data not valid

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

Flash memory error

bit 1 Internal error, it cannot be restored.

Counting error

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

Following error

bit 3 The difference between the real position and the theoretical position is greater than the value set in the [Max following error \[0x03-0x04\]](#) parameter; we suggest reducing the dynamics of the movements (acceleration, deceleration, velocity).

Encoder 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. Set a proper value next to the [Target position \[0x2B-0x2C\]](#) registers.

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; we suggest reducing the dynamics of the movements (acceleration, deceleration, velocity).

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 196). 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 196). 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** and **Encoder not synchronized** alarms 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 commanded 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 53.

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: the PWM is active and the axis is under closed-loop control (for instance, while reaching a target position or using a jog). 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

Theoretical state of the axis.
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 53.

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 44 and the "4.5.1 JOG + and JOG - buttons (Figure 9)" section on page 47.

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 44 and the "4.5.1 JOG + and JOG - buttons (Figure 9)" section on page 47.

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 44 and the "4.5.2 PRESET button (Figure 9)" section on page 48.

PWM saturation

bit 12

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

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 54.

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 54.

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 54.

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]

Current speed of the device expressed in revolutions per minute [rpm], updated at every second.

Position following error [0x05-0x06]

[Registers 6-7, Integer32, ro]

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

Temperature value [0x07]

[Register 8, Integer16, ro]

This variable shows both the temperature of the motor and the temperature of the electronics as detected by internal probes. The value is expressed in Celsius degrees (°C). The minimum detectable temperature is -20°C.

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

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
MSB								LSB							
Major number								Minor number							
Temperature of the motor								Temperature of the electronics							

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

Wrong parameters list [0x08-0x09]

[Registers 9-10, Unsigned32, ro]

The operator has set invalid data and the **Machine data not valid** alarm has been triggered. This variable is meant to show the list of the wrong parameters, respecting the structure shown in the following table.

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

Bit	Parameter
0	Not used
1	Distance per revolution [0x00]
2	Acceleration [0x07]
3	Deceleration [0x08]
4	Positive delta [0x09-0x0A]
5	Negative delta [0x0B-0x0C]
6	Jog speed [0x0D]
7	Work speed [0x0E]
8	Code sequence [0x0F]
9	Preset [0x12-0x13]
10	Jog step length [0x14]
11	Kp position loop [0x05]
12	Ki position loop [0x06]
13	Position window time [0x02]
14	Max following error [0x03-0x04]
15 to 31	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 has fixed value and cannot be changed by the user through DIP switch.

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 rotary switch.

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

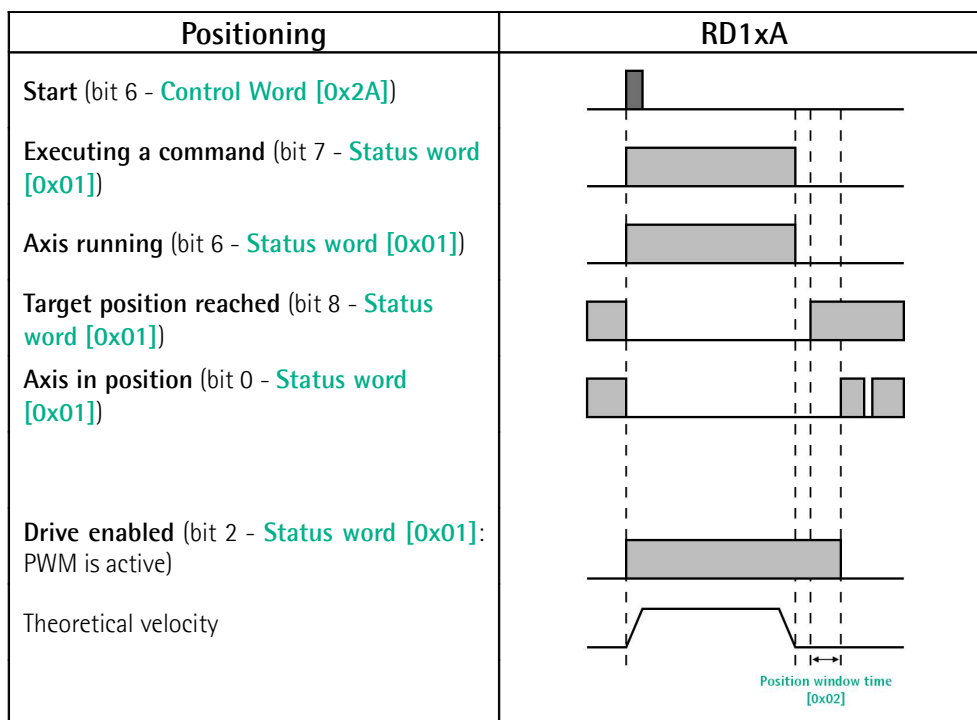
Value 30 B1 hex in hexadecimal notation corresponds to the binary representation 0011 0000 1011 0001 and has to be interpreted as follows: RD1xA model with MODBUS RTU interface, T48 reduction gear, equipped with brake, hardware version 1.


NOTE

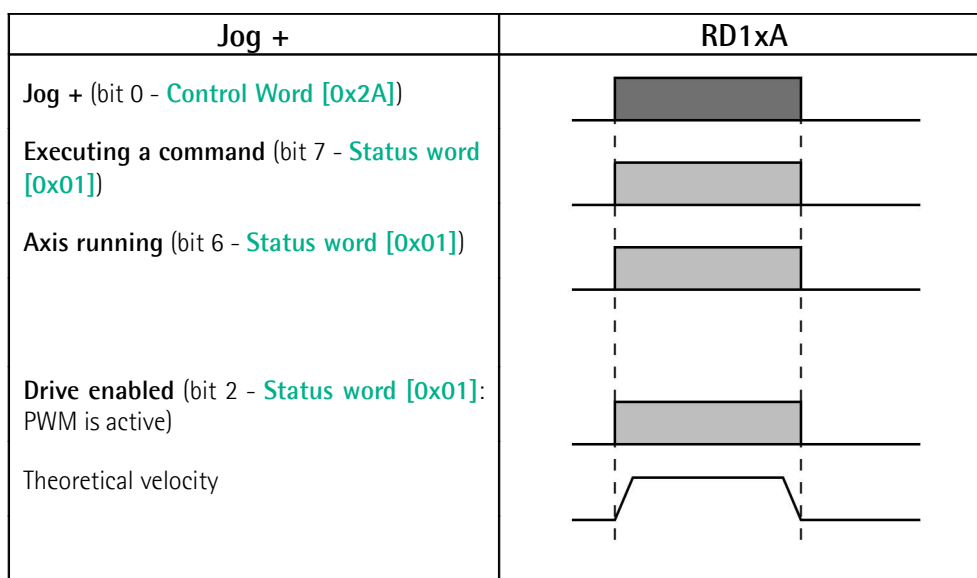
Save the set values using the **Save parameters** function.
Should the power be turned off all data not saved will be lost!



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 190): **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

ETHERNET

POWERLINK

Parameters list	Default value		
2201-00 Target position P	0		
2204-00 Distance per revolution-pulse PPR	1,024		
2205-00 Position tolerance P	1		
2206-00 Settling time-ms ms	0		
2207-00 Max following error-pulse P	1,024		
2208-00 Proportional gain	300		
2209-00 Integral gain	10		
220A-00 Acceleration- rev/s ² rev/s ²	10		
220B-00 Deceleration- rev/s ² rev/s ²	10		
220C-00 Max delta pos-pulse P	523 263		
220D-00 Max delta neg-pulse P	523 263		
220E-00 Jog speed-rpm rpm	2,000		
220F-00 Work speed-rpm rpm	2,000		
2210-00 Count direction 0=CW,1=CCW	0		
2211-00 Preset-pulse P	0		
2212-00 Jog step-pulse 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] rev/s ²	10		
Deceleration [0x08] 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		
Target position [0x2B-0x2C] P	0		

This page intentionally left blank

This page intentionally left blank

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