Smart encoders & actuators

User's guide

RD1A
RD12A

EtherNet/IP™ Modbus

RS-232 version

- Rotary actuator with EtherNet/IP interface in compliance with ODVA specification, edition April 2017
- Class 1 Real Time Ethernet (RTE) according to IEC 61 784-2
- Brushless motor, nominal torque 5 Nm
- 18-bit real multiturn absolute encoder
- RD12A with integrated brake, braking torque 17 Nm
- For change-over operations and automated positioning systems

Suitable for the following models:

- RD1A-P8-Txx-EP-...
- RD12A-P8-Txx-EP-...

General Contents

General contents

Subject Index

Typographic and iconographic conventions

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

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

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

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

Preliminary information

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

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

- RD1xA-P8-Txx-**CB**-... = CANopen DS301 interface;
- RD1xA-P8-Txx-**EC**-... = EtherCAT interface;
- RD1xA-P8-Txx-**EP**-... = EtherNet/IP interface;
- RD1xA-P8-Txx-**MB**-... = MODBUS RTU (RS-485) interface;
- RD1xA-P8-Txx-**PB**-... = Profibus-DP interface;
- RD1xA-P8-Txx-**PL**-... = POWERLINK interface;
- RD1xA-P8-Txx-**PT**-... = Profinet interface.

The present manual is specifically designed to describe the EtherNet/IP 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 EtherNet/IP 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 four 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 information on how to install and configure the actuator under Studio 5000 development environment as well as tips for setting up and running properly and efficiently the unit are provided.
In the third section, entitled **EtherNet/IP Interface** and **Integrated Web Server**, both general and specific information is given on the EtherNet/IP interface. In this section the interface features and the parameters implemented in the unit are fully described.
In the fourth section, entitled **MODBUS Interface**, both general and specific information is given on the Modbus interface. As previously stated, EtherNet/IP 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 EtherNet/IP network. In the **MODBUS Interface** section the interface features and the registers implemented in the unit are fully described.

Glossary of EtherNet/IP terms

EtherNet/IP, 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 Ethernet/IP interface. They are listed in alphabetical order.

Adapter	Devices such as drives, controllers, and computers usually require an adapter to provide a communication interface between them and a network such as EtherNet/IP. An adapter reads data on the network and transmits it to the connected device. It also reads data in the device and transmits it to the network.
Adapter Class Device	An Adapter Class product emulates functions provided by traditional rack-adapter products. This type of node exchanges real-time I/O data with a Scanner Class product. It does not initiate connections on its own (see I/O Adapter).
Application I/O Trigger	The Application Trigger is one of three types of I/O triggers supported by CIP for the exchange of data on I/O connections. It is very similar to the CoS trigger and not common.
Application Objects	A reference to multiple Object Classes that implement product-specific features.
Attribute	Attributes are characteristics of an Object and/or an Object Class. They provide a description of an externally visible characteristic or feature of an object. Typically, Attributes provide status information or govern the operation of an Object. For example: the ASCII name of an object; and the repetition rate of a cyclic object. The Attribute part of an object specification is divided into two sections: • Class attributes; • Instance attributes.
Behavior	The relationship between attribute values and services, i.e. a specification of how an object acts. Actions results from different events the object detects, such as receiving service request, detecting internal faults or elapsing timers. The Behavior of an Object indicates how it responds to particular events. For example, a person can be abstractly viewed as an Instance within the Class Human. Generally speaking, all humans have the same set of attributes: age, gender, etc., yet, because the values of each attribute vary, each of us looks/behaves in a distinct fashion.
BOOTP (Bootstrap Protocol)	BOOTP lets the device configure itself dynamically at boot time if the network has a BOOTP server. The BOOTP server assigns the device a pre-configured IP address, a subnet mask, and a gateway address; therefore, you do not have to

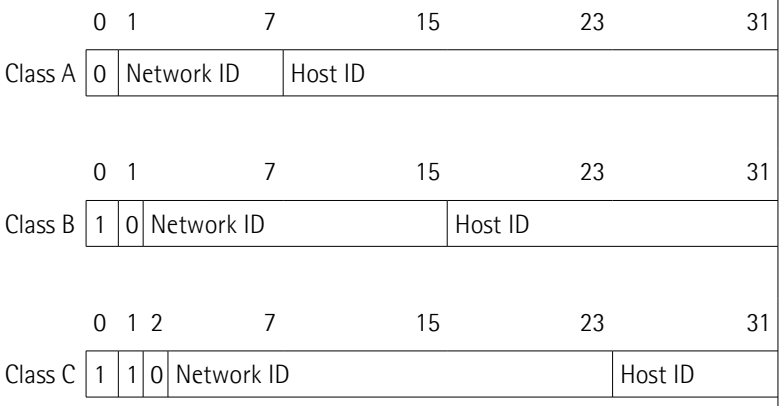
	configure these using the parameters in the device. BOOTP can make it easier to administer an EtherNet/IP network.
Bridge	A bridge refers to a network device that can route messages from one Ethernet network to another.
Broadcast	A broadcast transmission is a packet that all nodes on the network receive.
Change of State I/O Trigger	Change of State (CoS) is one of three types of I/O triggers supported by CIP for the exchange of data on Class 0 or 1 I/O connections. CoS endpoints send their messages when a change occurs. The data is also send at a background cyclic interval if no change occurs to keep the connection from timing out.
CIP (Common Industrial Protocol)	CIP is the transport and application layer protocol used for messaging over EtherNet/IP, ControlNet, and DeviceNet networks. The protocol is used for implicit messaging (real time I/O) and explicit messaging (configuration, data collection, and diagnostics).
Class	A class (of objects) is a set of objects that all represent the same kind of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but may contain different attribute values. A class contains the objects that relate to a device, they are organized in instances. For example, Ethernet/IP encoders from Lika supports the following classes: • Identity Object (Class Code 01h); • Message Router Object (Class Code 02h); • Assembly Object (Class Code 04h); • Connection Manager Object (Class Code 06h); • Position Sensor Object (Class Code 23h); • TCP/IP Interface Object (Class Code F5h); • EtherNet Link Object (Class Code F6h); • Device Level Ring (DLR) Object (Class Code 47h); • Quality of Service (QoS) Object (Class Code 48h).
Class Attribute	A Class Attribute is an attribute whose scope is that of the class as a whole, rather than any one particular instance. Therefore, the list of Class Attributes is different than the list of Instance Attributes. CIP defines the Instance ID value zero (0) to designate the Class level versus a specific Instance within the Class.
Class code	A hexadecimal identifier assigned to each CIP object.
Connected Messaging	A CIP connection is a relationship between two or more application objects on different nodes. The connection establishes a virtual circuit between end points for transfer of data. Node resources are reserved in advance of data transfer and are dedicated and always available. Connected messaging reduces data handling of messages in the node. Connected

	messages can be Implicit (I/O) or Explicit.
Connection Establishment/Close	Connections are established Connection Originators using the ForwardOpen service and closed by using the ForwardClose service. Connection clean-up takes place when either connection end point times out.
Connection Originator	The source node that makes a request to a Connection Target for a connection. It can initiate either an I/O connection or explicit message connection using the ForwardOpen service.
Connection Target	Destination for I/O or explicit message connection requests. Responds to a connection request with a ForwardOpen service response.
Client	Within a client/server model, the client is the device that sends a request to a server. The client expects a response from the server.
Communication Objects	A reference to the Object Classes that manage and provide the run-time exchange of implicit (I/O) and explicit messages.
Consumer	Within the producer/consumer model, the consumer is one of potentially several consuming devices that picks up a message placed on the network by a producing device.
Controller	A controller, also called programmable logic controller, is a solid-state control system that has a user-programmable memory for storage of instructions to implement specific functions such as I/O control, logic, timing, counting, report generation, communication, arithmetic, and data file manipulation. A controller consists of a central processor, input/output interface, and memory.
Cyclic I/O Trigger	Cyclic is one of three types of I/O triggers supported by CIP for the exchange of data on Class 0 or 1 I/O connections. Endpoints send their messages at pre-determined cyclic time intervals.
Data Rate	The data rate is the speed at which data is transferred on the EtherNet/IP network. You can set the device to a data rate of 10 Mbps Full-Duplex, 10 Mbps Half-Duplex, 100 Mbps Full-Duplex, or 100 Mbps Half-Duplex. If another device on the network sets or auto-negotiates the data rate, you can set the device to automatically detect the data rate.
DSI (Drive Serial Interface)	DSI stands for Drive Serial Interface, it is based on the ModBus RTU serial communication protocol.
DSI Peripheral	A device that provides an interface between DSI and a network or user.
DSI Product	A device that uses the DSI communications interface to communicate with one or more peripheral devices. For example, a motor drive is a DSI product.
Duplex	Duplex describes the mode of communication. Full-duplex communications let a device exchange data in both directions at the same time. Half-duplex communications let a device

	exchange data only in one direction at a time. The duplex used by the adapter depends on the type of duplex that other network devices, such as switches, support.
EDS (Electronic Data Sheet) Files	EDS files are simple text files that are used by network configuration tools for EtherNet/IP to describe products so that you can easily commission them on a network. EDS files describe a product device type, revision, and configurable parameters. EDS files can be downloaded from Lika web site.
EDS File	An Electronic Data Sheet (EDS) is an ASCII text file that describes the features of an EtherNet/IP device and is used by software tools for device and network connection configuration.
EEPROM	EEPROM is the permanent memory of a device. Devices such as the encoder store parameters and other information in EEPROM so that they are not lost when the device loses power. EEPROM is sometimes called "NVS (Non-Volatile Storage)".
Encapsulation Protocol	Defines the communication relationship between two nodes known as an Encapsulation Session. The Encapsulation Protocol uses TCP/UDP Port 44818 for several Encapsulation Commands and for CIP Explicit Messaging. An example encapsulation command is the List_Identity Command that performs a "network who". An Encapsulation Session must be established before any CIP communications can take place. Data format for the Encapsulation Protocol is Little-Endian.
EtherNet/IP Network	Ethernet/IP (Industrial Protocol) is an open producer-consumer communication network based on the Ethernet standard (IEEE 802.3), TCP/IP, UDP/IP, and CIP. Designed for industrial communications, both I/O and explicit messages can be transmitted over the network. Each device is assigned a unique IP address and transmits data on the network. The number of devices that an EtherNet/IP network can support depends on the class of IP address. For example, a network with a Class C IP address can have 254 nodes. General information about EtherNet/IP and the EtherNet/IP specification are maintained by the Open DeviceNet Vendor's Association (ODVA). ODVA is online at http://www.odva.org.
Exclusive Owner Connection	This is one of three types of Implicit (I/O) Connections. It is a Class 0 or 1 bidirectional connection to an Output connection point (typically an Assembly Object), where the data of this assembly can only be controlled by one Scanner. There may be a connection to an input assembly; this data is being sent to the scanner. If the input data length is zero, then this direction becomes a Heartbeat connection.
Explicit Message Client	An explicit message client initiates request/response oriented communications with other devices. Examples of explicit message clients are HMI devices, programming tools, or PC or Linux based applications that gather data from control

	devices.
Explicit Message Server	An explicit message server responds to request/response oriented communications initiated by explicit message clients. An example of an explicit message server is a bar code reader.
Explicit Messaging	Explicit Messages are used to transfer data that does not require continuous updates. They are typically used to configure, monitor, and diagnose a device over the network. Explicit Messages can be sent as a connected or unconnected message. CIP defines an Explicit Messaging protocol that states the meaning of the message. This messaging protocol is contained in the message data. Explicit Messaging provide the means by which typical request/response oriented functions are performed (e.g., module configuration). These messages are typically point-to-point. Message rates and latency requirements are typically not as demanding as I/O messaging.
ForwardOpen Service Request	The ForwardOpen Service Request is sent by the Connection Originator and received by the Connection Target to open and establish explicit and I/O connections. The ForwardOpen Service request and associated response contains all of the connection parameters, including transport class, production trigger, timing information, electronic key and connection IDs.
Gateway	A gateway is a device on a network that connects an individual network to a system of networks. When a node needs to communicate with a node on another network, a gateway transfers the data between the two networks.
Hardware Address	Each Ethernet device has a unique hardware address (sometimes called a MAC address) that is 48 bits. The address appears as six digits separated by colons (for example, xx:xx:xx:xx:xx:xx). Each digit has a value between 0 and 255 (0x00 and 0xFF). This address is assigned in the hardware and cannot be changed. It is required to identify the device if you are using a BOOTP utility.
I/O Adapter	An I/O Adapter receives implicit communications requests from an I/O Scanner then produces and consumes its I/O data, typically at the requested cyclic rate. An I/O Adapter can be a simple digital input device, or something more complex such as a modular pneumatic valve system.
I/O Client	Function that uses the I/O messaging services of another (I/O Server) device to perform a task. Initiates a request for an I/O message to the server module. The I/O Client is a Connection Originator of Implicit Message connections
I/O Data	I/O data, sometimes called "implicit messages" or "input/output," transmit time-critical data. The terms "input" and "output" are defined from the controller's point of view. Output is transmitted by the controller and consumed by the device. Input is transmitted by the device and consumed by the controller.

I/O Messaging	Used interchangeably with the term Implicit Messaging.
I/O Scanner	An I/O scanner initiates implicit connections with I/O adapter devices, i.e., it is an I/O Client. A scanner is typically the most complex type of EtherNet/IP device, as it must deal with issues such as configuration of which connections to make, and how to configure the adapter device. Scanners also typically support initiating explicit messages, i.e., it is also an Explicit Message Client. A programmable controller is an example of an I/O scanner (used interchangeably with Scanner Class).
I/O Server	Function that provides I/O messaging services to another (I/O Client) device. Responds to a request from the I/O Client for an I/O connection. An I/O Server is the target of the implicit message connection request.
Implicit Messaging	Implicit Messages are exchanged across I/O Connections with an associated Connection ID. The Connection ID defines the meaning of the data and establishes the regular/repeated transport rate and the transport class. No messaging protocol is contained within the message data as with Explicit Messaging. Implicit Messages can be point to point (unicast) or multicast and are used to transmit application specific I/O data. This term is used interchangeably with the term I/O Messaging. Implicit Messaging on EtherNet/IP uses UDP/IP frames on port 2222. They are typically Class 0 or 1 and of the type Exclusive Owner, Input Only and Listen Only.
Input Only Connection	This is one of three types of Implicit (I/O) Connections. It is a Class 0 or 1 Connection to an Input connection point (typically an assembly object). The scanner receives input data from the target device and produces a Heartbeat to the target device. There is no Output data.
Instance	An object instance is the actual representation of a particular object within a class, i.e. it is a specific and real (physical) occurrence of an object. For example: New Zealand is an instance of the object class Country. Each instance of a class has the same attributes, but also has its own particular set of attribute values. The terms Object, Instance, and Object Instance all refer to a specific Instance.
Instance Attribute	An Instance Attribute is an attribute whose value is unique to an object instance and whose definition is shared by all instances of an object. Each instance need only support the optional attributes that apply to it. If an instance does not support an optional attribute, the Attribute Not Supported (General Status code 0x14) error shall be returned for services targeting that attribute.
IP Address	A unique IP address identifies each node on an EtherNet/IP network. An IP address consists of 32 bits that are divided into four segments of one byte each. It appears as four decimal integers separated by periods (xxx.xxx.xxx.xxx). Each "xxx" can have a decimal value from 0 to 255. For example, an IP address

	could be 192.168.0.1. An IP address has two parts: a network ID and a host ID. The class of network determines the format of the address.  The number of devices on your EtherNet/IP network will vary depending on the number of bytes that are used for the network address. In many cases you are given a network with a Class C address, in which the first three bytes contain the network address (subnet mask = 255.255.255.0). This leaves 8 bits or 256 addresses on your network. Because two addresses are reserved for special uses (0 is an address for the network usually used by the router, and 255 is an address for broadcast messages to all network devices), you have 254 addresses to use on a Class C address block. You must ensure that each device on the Internet has a unique address. You can then set the unique IP address for the device by using a BOOTP server or by manually configuring parameters in the device. The device reads the values of these parameters only at power-up.
Listen Only Connection	This is one of three types of Implicit Connections. It is a Class 0 or 1 Connection to an Input connection point (typically an assembly object). The scanner receives input data from the target device and produces a Heartbeat to the target device. There is no Output data. A Listen Only Connection can only be attached to an existing Exclusive Owner or Input Only Connection. If this underlying connection closes, then the Listen Only connection will also be closed or timed out.
Master	EtherNet/IP does not use Master/Slave technology or terminology.
Message Client	Function that uses the Explicit messaging services of another (Message Server) device to perform a task. Initiates an Explicit Message request to the server device.
Message Server	Function that provides Explicit Messaging services to another (Message Client) device. Responds to an Explicit Message request from the Message Client.
Multicast	Multicast is the single transmission of an I/O data packet that may be consumed by multiple devices using multicast IP and Ethernet destination addresses. See Producer/Consumer Communications Model.

Object	A CIP node is modeled as a collection of Objects. An Object provides an abstract representation of a particular component within a product. The realization of this abstract object model within a product is implementation dependent. In other words, a product internally maps this object model in a fashion specific to its implementation.
Ping	A ping is a message that is sent by a DSI product to its peripheral devices. They use the ping to gather data about the product, including whether it can receive messages and whether they can log in for control.
Point to Point (Unicast)	Point to Point or Unicast is the transmission of data to a single device.
Producer	Within the producer/consumer model, the producing device places a message on the network for consumption by one or several consumers. Generally, the produced message is not directed to a specific consumer.
Producer/Consumer Communications Model	For I/O Connections, CIP supports object-oriented Producer/Consumer communication. Connection identifiers embedded into each message are used by devices to determine which messages they should "consume" from other devices that "produce" messages. This enables efficient use of network bandwidth by transmitting information only once. Less bandwidth equates to greater efficiency and overall speed. EtherNet/IP uses IP multicast and Ethernet multicast destination addressing to implement this capability.
Requested Packet Interval (RPI)	EtherNet/IP devices typically produce or consume data based upon a Requested Packet Interval (RPI) value. Producer devices send data packets at a predetermined time interval based on the RPI, whereas consumer devices will listen for a packet of data at a given RPI.
Scanner Class	A Scanner Class product exchanges real-time I/O data with Adapter Class and Scanner Class products. This type of node can respond to connection requests and can also initiate connections to target devices (see I/O Scanner).
Server	Within a client/server model, the server is the device that receives a request from a client. The server is expected to give a response to the client.
Service (common service)	A list of the common services defined for the object. A function supported by an object and/or object class.
Service (object-specific service)	The full specifications of any services unique to the object.
Service code	Service codes are used to define the action that is requested to take place when an object or parts of an object are addressed through explicit messages. They are used to access classes or the attributes of a class or to generate specific events.

Slave	EtherNet/IP does not use Master/Slave technology or terminology.
Subnet Mask	A subnet mask is an extension to the IP addressing scheme that lets you use a single network ID for multiple physical networks. A bit mask identifies the part of the address that specifies the network and the part of the address that specifies the unique node on the network. A "1" in the subnet mask indicates the bit is used to specify the network. A "0" in the subnet mask indicates that the bit is used to specify the node. For example, a subnet mask on a Class C address may appear as follows: 11111111 11111111 11111111 11000000 (255.255.255.192). This mask indicates that 26 bits are used to identify the network and 6 bits are used to identify devices on each network. Instead of a single physical Class C network with 254 devices, this subnet mask divides it into four networks with up to 62 devices each.
Switches	Switches are network devices that provide virtual connections that help to control collisions and reduce traffic on the network. They are able to reduce network congestion by transmitting packets to an individual port only if they are destined for the connected device. In a control application, in which real time data access is critical, network switches may be required in place of hubs.
TCP (Transmission Control Protocol)	EtherNet/IP uses this protocol to transfer Explicit Messaging packets using IP. TCP guarantees delivery of data through the use of retries.
Transport Classes	CIP defines several Transport Classes for messaging connections. Within EtherNet/IP, I/O data sent on Class 1 connections is pre-pended with a 16-bit sequence count, while data on Class 0 connections is not. Class 3 connections are used for Explicit Messaging Connections.
UDP (User Datagram Protocol)	EtherNet/IP uses this protocol to transfer I/O packets using IP. UDP provides a simple, but fast capability to send I/O messaging packets between devices. This protocol ensures that devices transmit the most recent data because it does not use acknowledgments or retries.
Unconnected Messaging	Provides a means for a node to send message requests without establishing a CIP connection prior to data transfer. More overhead is contained within each message and the message is not guaranteed destination node resources. Unconnected Messaging is used for non-periodic requests (e.g., network "Who" function). Applies to explicit messages only.
Unicast (Point to Point)	Unicast or Point to Point is a connection for the transmission of data to a single device.

List of abbreviations

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

API	Actual Packet Interval
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation
CIP	The Common Industrial Protocol defined in this volume of the CIP Networks Library. CIP includes both connected and unconnected messaging.
CID	Connection Identifier
DLL	Data Link Layer
EPR	Expected Packet Rate
ISO	International Standards Organization
MAC ID	Media Access Control Identifier
PDU	Protocol Data Unit
ODVA	ODVA, Inc.
O ➡ T	Originator to Target (used to describe packets that are sent from the originator to the target)
OSI	Open Systems Interconnection (see ISO 7498)
RPI	Requested Packet Interval
SDU	Service Data Unit
SEM	State Event Matrix
SEMI	Semiconductor Equipment Materials International
STD	State Transition Diagram, used to describe object behaviour
T ➡ O	Target to Originator (used to describe packets that are sent from the target to the originator)
UCMM	Unconnected Message Manager

References

[1] THE CIP NETWORKS LIBRARY, Volume 1, Common Industrial Protocol (CIP™), Edition 3.22, April 2017

[2] THE CIP NETWORKS LIBRARY, Volume 2, EtherNet/IP Adaptation of CIP, Edition 1.23, April 2017

Glossary of MODBUS terms

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

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

	specified by function codes. The function code is sent from a Client to the Server and indicates which kind of action the Server must perform. MODBUS function codes are elements of MODBUS request/reply PDUs. The function code field of a MODBUS data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (the range 128 – 255 is reserved and used for exception responses). Function code "0" is not valid. Lika actuators only implement public function codes.
Holding register	In the MODBUS data model, a Holding register is the output data. A Holding register has a 16-bit quantity, is alterable by an application program, and allows either read-write or read-only access.
IEEE 1588	This standard defines a protocol enabling synchronisation of clocks in distributed networked devices (e.g. connected via Ethernet).
Input register	In the MODBUS data model, an Input register is the input data. An Input register has a 16-bit quantity, is provided by an I/O system, and allows read-only access.
LRC Checking	In ASCII mode, messages include an error-checking field that is based on a Longitudinal Redundancy Checking (LRC) calculation that is performed on the message contents, exclusive of the beginning 'colon' and terminating CRLF pair characters. It is applied regardless of any parity checking method used for the individual characters of the message.
Master	A Master is any network device that sends data requests to Slaves.
Message	The MODBUS messaging service provides a Client/Server communication between devices connected on the network. The Client / Server model is based on four types of messages: • MODBUS Request • MODBUS Confirmation • MODBUS Indication • MODBUS Response The MODBUS messaging services are used for information exchange.
MODBUS Confirmation	A MODBUS Confirmation is the Response Message received on the Client side.
MODBUS Indication	A MODBUS Indication is the Request message received on the Server side.
MODBUS Request	A MODBUS Request is the message sent on the network by the Client to initiate a transaction.
MODBUS Response	A MODBUS Response is the Response message sent by the Server.
Network	Network is a group of computers on a single physical network segment.

PDU	The Protocol Data Unit (PDU) is the MODBUS function code and data field. It is packed together with the Address Field and the CRC (or LRC) to form the Modbus Serial Line PDU. The MODBUS protocol defines three PDUs. They are: • MODBUS Request PDU, mb_req_pdu • MODBUS Response PDU, mb_rsp_pdu • MODBUS Exception Response PDU, mb_excep_rsp_pdu
Read Holding Registers (03, 0003hex)	This function code is used to READ the contents of a contiguous block of holding registers in a remote device; in other words, it allows to read the values set ina group of work parameters placed in order.
Read Input Register (04, 0004hex)	This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some result values and state / alarm messages in a remote device.
Register	MODBUS functions operate on memory registers to configure, monitor, and control device I/O.
RTU transmission mode	Remote Terminal Unit. When devices communicate on a MODBUS serial line using the RTU mode, each 8–bit byte in a message contains two 4–bit hexadecimal characters. The main advantage of this mode is that its greater character density allows better data throughput than ASCII mode for the same baud rate. Each message must be transmitted in a continuous stream of characters.
Server	A Server is any program that awaits data requests to be sent to it. Servers do no initiate contacts with Clients, but only respond to them. MODBUS follows the Client/Server model. MODBUS Masters are referred to as clients, while MODBUS Slaves are servers.
Service request	It is the MODBUS Request, i.e. the message sent on the network by the Client to initiate a transaction.
Slave	A Slave is any program that awaits data requests to be sent to it. Slaves do no initiate contacts with Masters, but only respond to them.
Transmission rate	Data transfer rate (in bps).
Write Multiple Registers (16, 0010hex)	This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
Write Single Register (06, 0006hex)	This function code is used to WRITE a single holding register in a remote device.

1 Safety summary

1.1 Safety

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

1.2 Electrical safety

- Turn OFF power supply before connecting the device;
- connect according to explanation in the "Electrical connections" section on page 39;
- 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 35;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the unit;
- do not tool the unit or its shaft;
- delicate electronic equipment: handle with care; do not subject the device and the shaft to knocks or shocks;
- respect the environmental characteristics of the product.

WARNING

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

WARNING

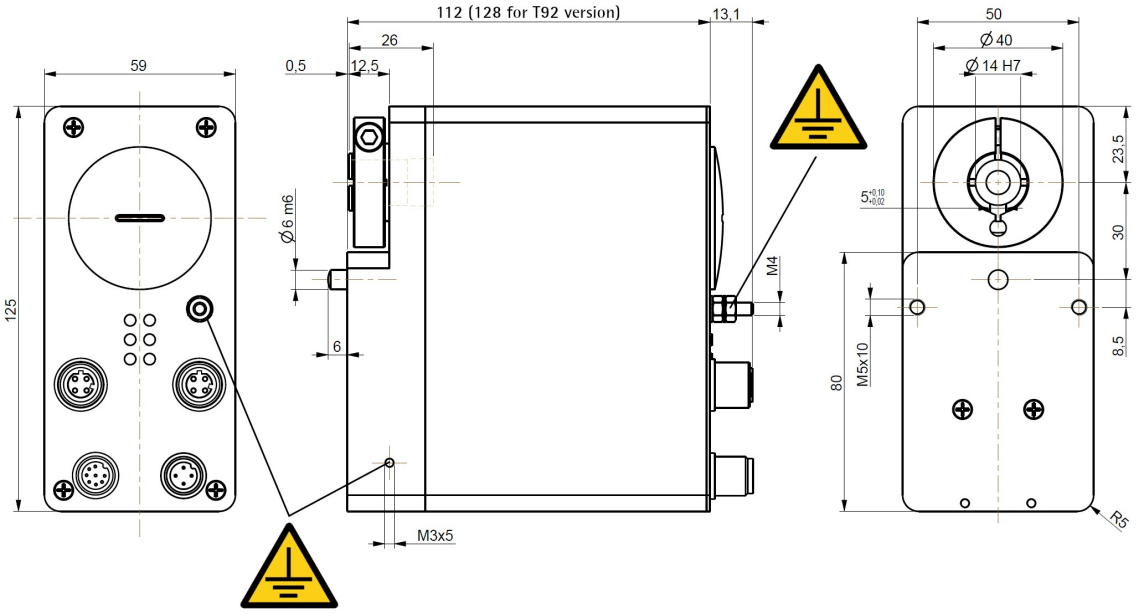
The counter-electromotive force (back EMF) generated by the motor in case the shaft is forced to spin due to a manual external force can cause irreparable damages to the internal circuitry.

2 Identification

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

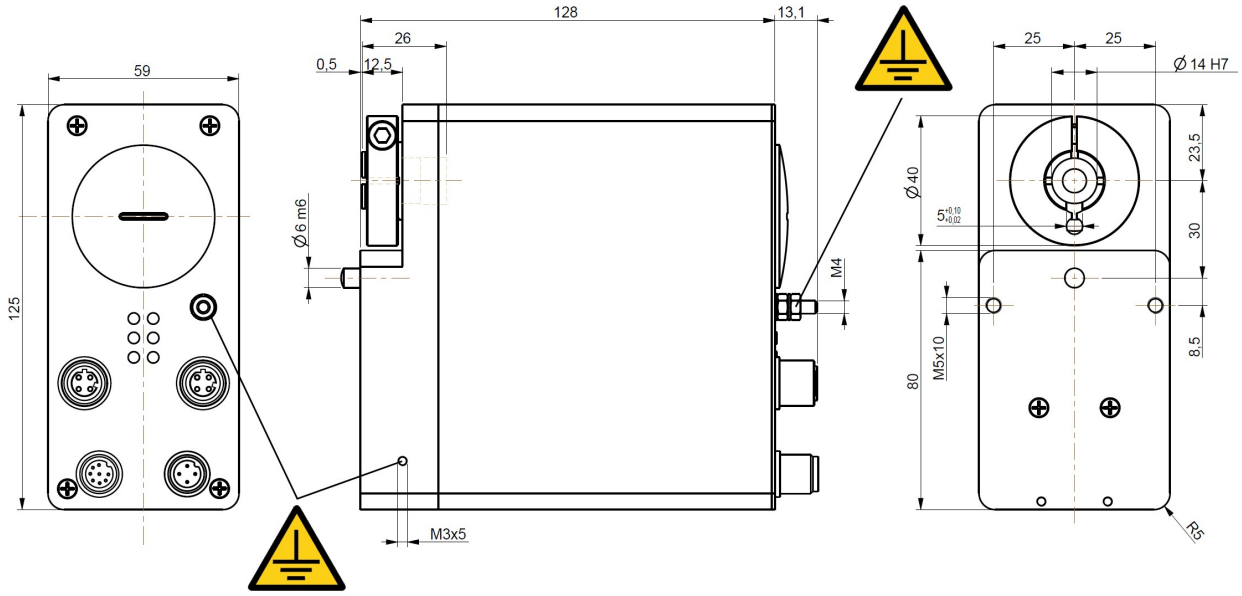
3 Mechanical installation

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



(values are expressed in mm)

Figure 1 - RD1A unit – Overall dimensions



(values are expressed in mm)

Figure 2 - RD12A unit – Overall dimensions

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

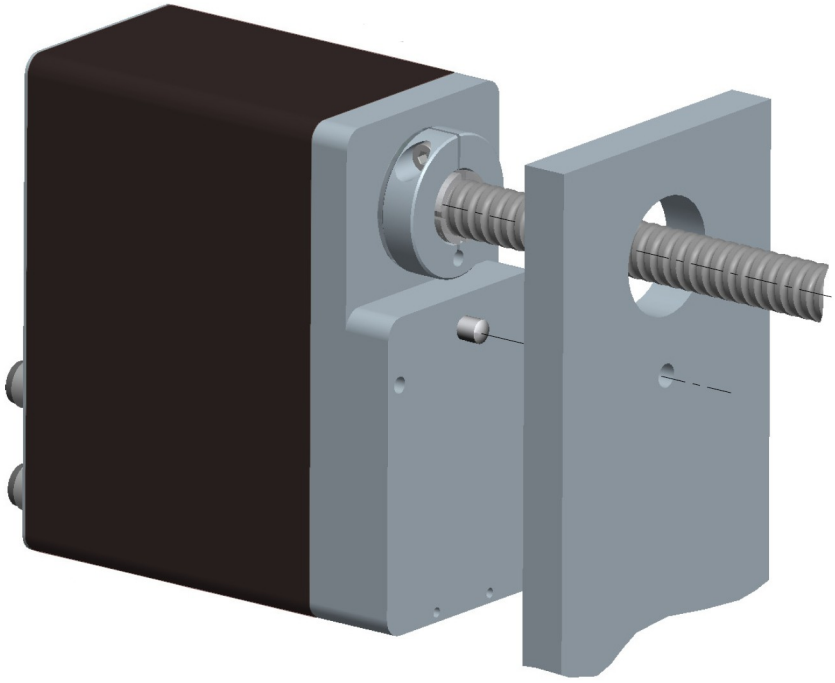
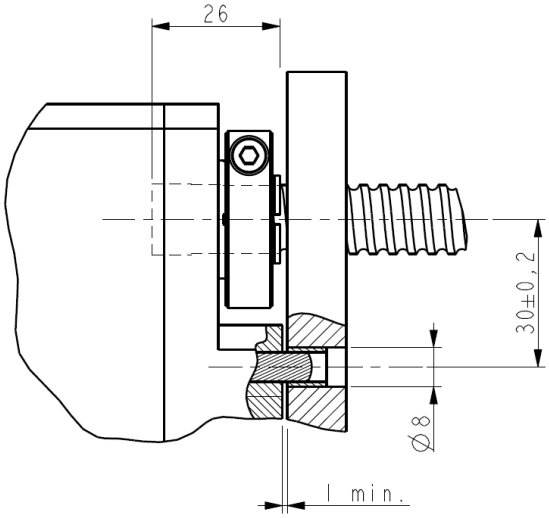


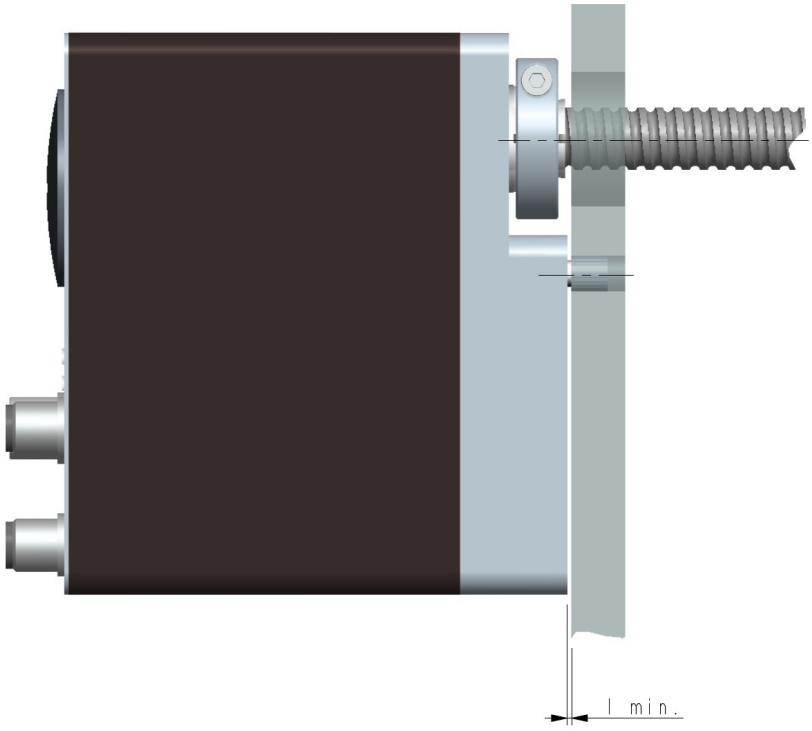
Figure 3 - Example of typical installation of RD1xA unit on worm screw

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 anti-rotation pin. We suggest adding a silicone isolator (not provided). The distance between the axis of the shaft and the axis of the hole must be 30 mm (1.18") ± 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 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 hole (in the silicone isolator, if installed);
- 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
Power supply must be turned off before performing any electrical connection!
When you send the **Start**, **Jog +** or **Jog -** commands, the unit and the shaft start moving! Before operating please make sure that there are no risks of personal injury and mechanical damages.
Each **Start** routine has to be checked carefully in advance!
Never force manually the rotation of the shaft not to cause permanent damages!

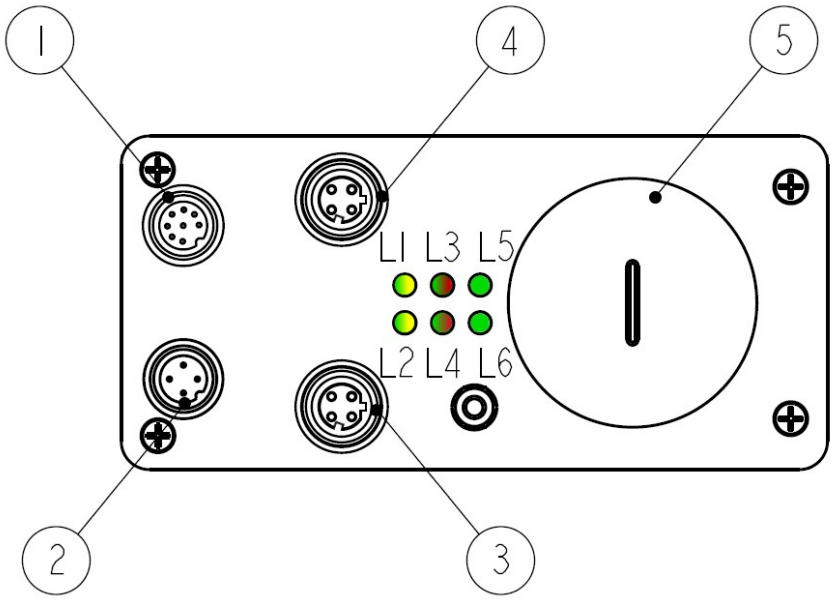


Figure 4: Connectors and diagnostic LEDs

Legend

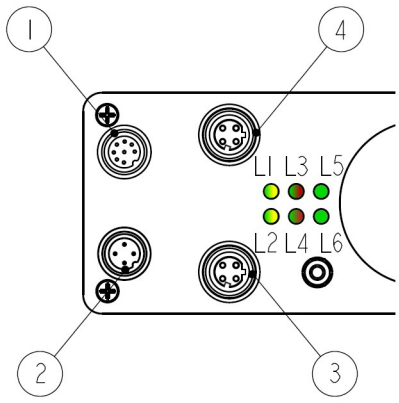
1	M12 8-pin male connector inputs / output + MODBUS RS-232
2	M12 4-pin male connector power supply
3	M12 4-pin female connector PORT 1
4	M12 4-pin female connector PORT 2
5	Internal housing of Preset / Jog buttons
L1	LED 1 Link / Activity in PORT 2
L2	LED 2 Link / Activity in PORT 1
L3	LED 3 MS Module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

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

4.1 Ground connection (Figure 1 and Figure 2)

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

4.2 Connectors (Figure 4 and Figure 5)



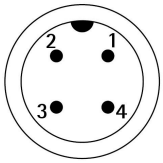
1	M12 8-pin male connector, 3 inputs / 1 output + service serial port MODBUS RS-232
2	M12 4-pin male connector power supply
3	M12 4-pin female connector PORT 1
4	M12 4-pin female connector PORT 2

Figure 5: Connectors

4.2.1 Power supply connector

Power supply
M12 4-pin male connector

(frontal side)

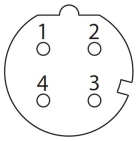


Pin	Description
1	motor +24Vdc ±10% power supply
2	controller +24Vdc ±10% power supply
3	motor and controller 0Vdc supply voltage
4	not connected

4.2.2 EtherNet/IP interface connectors (PORT 1 and PORT 2)

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

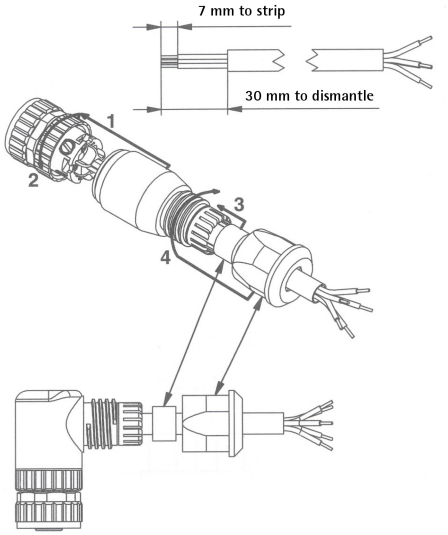
[1] Lika EC- pre-assembled cables only

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

P1 PORT 1 and P2 PORT 2 M12 connectors have pin-out in compliance with the Ethernet standard. Therefore you can use standard Ethernet cables commercially available, for more information see later.
P1 PORT 1 and P2 PORT 2 connectors are interchangeable.

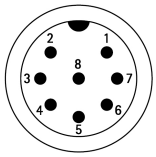
NOTE

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



4.2.3 Inputs / output + MODBUS RS-232 service port

Inputs / output + MODBUS RS-232 service port
M12 8-pin connector
A coding, male
(frontal side)



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

For any information on the three digital inputs and the digital output please refer to the "6.3 Digital inputs and output" section on page 85.

The configuration parameters of the MODBUS service serial port have fixed values so the user cannot change them.
They are:

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

The MODBUS address is "1". It cannot be changed. See the "9.2 "Serial configuration" page" section on page 174.

For any further information on configuring and using the RS-232 service serial port refer to the "MODBUS® interface" section on page 172.

4.3 Network configuration: topologies, cables, hubs, switches - Recommendations

Cables and connectors comply with the Ethernet specifications.
Standard Ethernet cables type CAT-5, CAT-5e and CAT-6 commercially available can be used.
The minimum cabling performance that will support EtherNet/IP is Category 5 as defined by ANSI/TIA/EIA-568-B.2 Annex N. There are reasons to select one category of cabling over another. In general, the higher the category, the better the cabling performance. Another consideration is balance. Category 5e, 6 and the newest proposed category, known as augmented 6 or Category 6a, will support current applications such as 1Gb/s and 10 Gb/s. Generally speaking, the greater the cabling category, the less EMC protection that is needed. Consult your cable supplier for guidance on EMC protection for the specific cable being used.
For complete information please refer to IEC 61918, IEC 61784-5-13 and IEC 61076-2-101.

The maximum cable length (100 meters) predefined by Ethernet 100Base-TX must be compulsorily fulfilled.
Regarding wiring and EMC measures, the IEC 61918 and IEC 61784-5-13 must be considered.
Compliance with IEEE Ethernet standards provides users with a choice of network interface speeds – e.g., 10, 100 Mbps, 1 Gbps and beyond – and a flexible network architecture compatible with commercially available Ethernet installation options including copper, fiber, fiber ring and wireless, and topologies including star, linear and ring.
A hub is an inexpensive connectivity method that provides an easy method of connecting devices on information networks (shared Ethernet). A switch reduces

collisions and is recommended for real-time control installations (switched Ethernet). Routers are used to isolate control data traffic from other types of office data traffic, to isolate information traffic on the plant floor from control traffic on the plant floor, and for security purposes, i.e., firewalls. Repeaters extend the overall network cable length. They can also connect networks with different media types.

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

4.4 MAC 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 as well as the subnet mask.
For additional information on the MAC address refer to the "5.4 MAC address" section on page 59.
For additional information on the IP address refer to the "4.5 EtherNet/IP Node ID" section below.

4.5 EtherNet/IP Node ID

By default, the rotary actuator is configured so that it uses the IP address, Subnet mask, and Gateway address that are saved internally. The use of a DHCP Server to allocate the IP address is disabled.
The IP address, the Subnet mask and the Gateway address are set next to the **IP Address**, **Network Mask** and **Gateway Address** parameters in the **F5-01-05 Interface Configuration** attribute, see the "7.12.6 Class F5h: TCP/IP Interface Object" section on page 136. For more information on setting the node ID *via software* refer to the "4.5.1 Setting the node ID via software" section hereafter.
The following table summarizes the default software IP parameters.

IP Parameter	IP address
IP address	192.168.1.10
Subnet mask	255.255.255.0
Gateway address	0.0.0.0
DHCP	Disabled

The network parameters can be set also in the Integrated Web Server, see the "8.7 Network configuration" section on page 166.
As an alternative, the node address can be set *via hardware* by using the rotary switches located inside the enclosure. For more information on setting the node ID via hardware refer to the "4.5.3 Setting the node ID via hardware (DIP A rotary switches)" section below.

4.5.1 Setting the node ID via software

As stated, by default, the rotary actuator is configured so that it uses the IP address saved internally. The DIP A rotary switches located inside the enclosure are set to OFF (high rotary switch = 0; low rotary switch = 0) so meaning that the software values saved internally are used, see the next section.
The software values can be changed by using a software tool such as Studio 5000 or by means of the integrated web server (see the "8.7 Network configuration" section on page 166) or by enabling a DHCP server (see the "4.5.3 Setting the node ID via hardware (DIP A rotary switches)" section hereafter).
Any Net ID value and Host ID value can be set via software.

4.5.2 Screw plug for internal access (Figure 4 and Figure 6)

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.

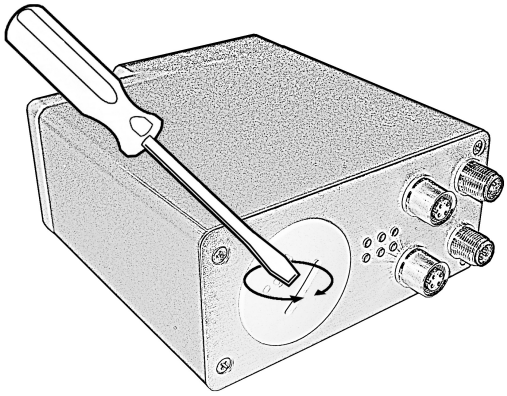


Figure 6: Screw plug for internal access

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

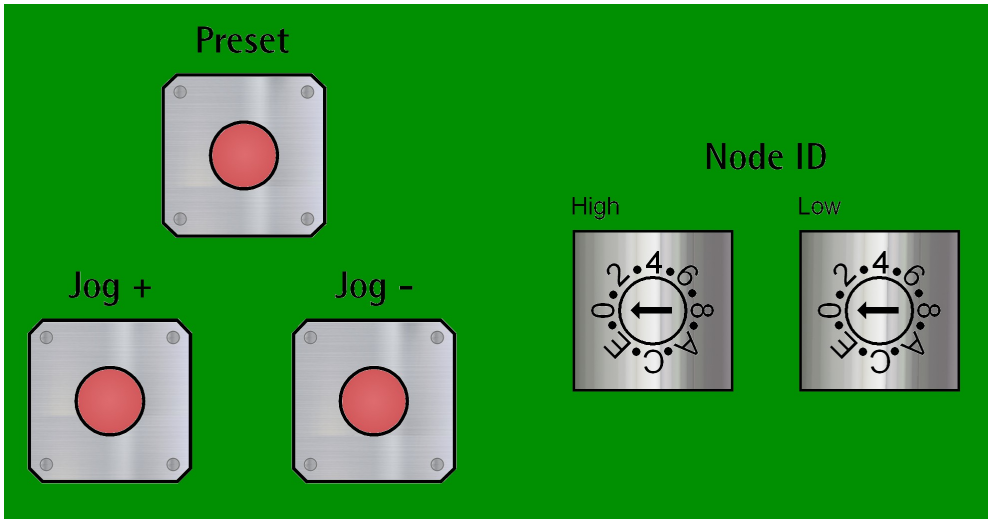


Figure 7: Preset / Jog buttons and rotary switches

4.5.3 Setting the node ID via hardware (DIP A rotary switches)

WARNING
Power supply must be turned off before setting the rotary switches!

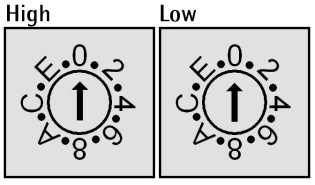
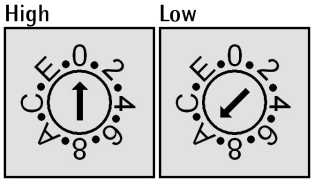
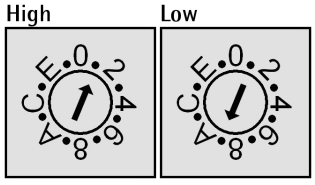
The EtherNet/IP node ID can be set *via hardware* using the DIP A rotary switches located inside the enclosure. To access the DIP A rotary switches please refer to the "4.5.2 Screw plug for internal access (Figure 4 and Figure 6)" section on page 45. Set the Node ID in hexadecimal notation.

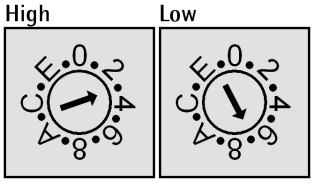
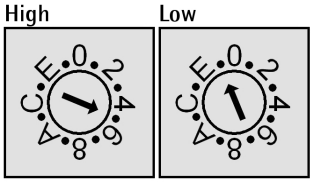
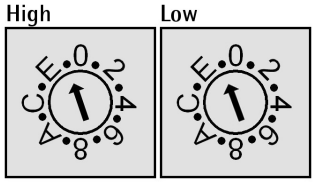
The DIP A rotary switches allow to set the Host ID; the Net ID is fixed, as defined in the following table:

192.168.1.	EtherNet/IP Node
Net ID	Host ID

Allowed node addresses range between 1_{10} (01 hex) and 254_{10} (FE hex). The subnet mask is 255.255.255.0.
Value 0_{10} (01 hex) means that the system uses the software IP address, Subnet mask, and Gateway address that are saved internally (default value, see the "4.5.1 Setting the node ID via software" section on page 45).
Value 255_{10} (FF hex) enables the use of a DHCP Server. The IP address and the Subnet mask are assigned by a DHCP Server.
The rotary switches are evaluated only during switching the operating voltage on or when resetting the rotary actuator.
Changes in the position of the switches when the rotary actuator is switched on are taken into consideration only after switching the rotary actuator off and then on again.

EXAMPLE

Address 0 = 00 hex:	Address 10 = 0A hex:	Address 25 = 19 hex:
High Low 	High Low 	High Low 

Address 55 = 37 hex:	Address 95 = 5F hex:	Address 255 = FF hex:
High Low 	High Low 	High Low 

4.6 Line Termination

EtherNet/IP network needs no line termination because the line is terminated automatically; in fact every device is able to detect the presence of the downstream devices.

4.7 Diagnostic LEDs (Figure 4 and Figure 8)

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

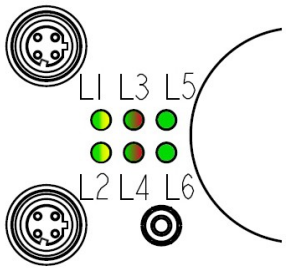


Figure 8: Diagnostic LEDs

L1	Link / Activity in PORT 1	**L4**	NS Network Status
L2	Link / Activity in PORT 2	**L5**	Controller power supply information
L3	MS Module Status	**L6**	Motor power supply information

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

EtherNet/IP™

LED L1 GREEN / YELLOW	Description
It shows the state of the physical link and the activity in PORT 1 (connector 4)	
OFF	There is neither network link nor activity on port P1
ON GREEN	The network link has been established (100 Mbit/s), but there is no activity
FLICKERING GREEN	Activity (100 Mbit/s) on port P1
ON YELLOW	The network link has been established (10 Mbit/s), but there is no activity
FLICKERING YELLOW	Activity (10 Mbit/s) on port P1

LED L2 GREEN	Description
It shows the state of the physical link and the activity in PORT 2 (connector 3)	
OFF	There is neither network link nor activity on port P2
ON GREEN	The network link has been established (100 Mbit/s),

	but there is no activity
FLICKERING GREEN	Activity (100 Mbit/s) on port P2
ON YELLOW	The network link has been established (10 Mbit/s), but there is no activity
FLICKERING YELLOW	Activity (10 Mbit/s) on port P2

LED L3 RED	**Description**
MS Module Status LED, it shows the state of the EtherNet/IP device	
OFF	The power supply is switched off
ON GREEN	The device is controlled by a Scanner in **Run** state
Flashing GREEN	• The device is not configured • The Scanner is in **Idle** state
ON RED	A major fault, i.e. an unexpected error has occurred (EXCEPTION state, FATAL error, etc.). See the **01-01-05 Status** attribute on page 105
Flashing RED	One or more recoverable faults have occurred. The module is configured, but stored parameters differ from currently used parameters. See the **01-01-05 Status** attribute on page 105

LED L4 GREEN / RED	**Description**
NS Network Status LED, it shows the current state of the EtherNet/IP network	
OFF	• The device is switched off • No IP address has been set
ON GREEN	The device is online, one or more CIP connections have been established (Class 1 or Class 3 communications)
Flashing GREEN	The device is online, but no CIP connection has been established
ON RED	• Duplicate IP address conflict has occurred, two devices on the network have been assigned the same IP address • A fatal error has occurred. If such a condition arises, please contact Lika Electronic After Sales Service
Flashing RED	One or more CIP connections have been expired (Class 1 or Class 3 communications)

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 whether the power supply of the motor is switched on	

ON	It indicates that the power supply of the motor is turned on
OFF	It indicates that the power supply of the motor is turned off

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

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

4.8 Preset / Jog buttons (Figure 9)

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

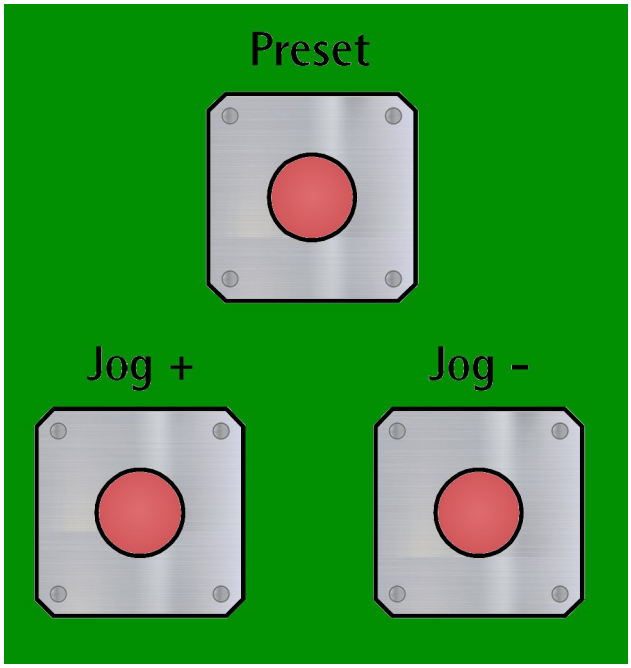


Figure 9: Preset / Jog buttons

4.8.1 JOG + and JOG – buttons (Figure 9)

Press the 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 114 ff (EtherNet/IP interface) or on page 212 ff (Modbus interface).

WARNING

JOG and PRESET buttons can be active and available for use to the operator, even when the DRIVECOD unit is in an alarm or emergency condition. Before pressing the 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 64-01-01 Control Word on page 115 -EtherNet/IP version; **Incremental jog** in Control Word [0x2A] on page 213 -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.

4.8.2 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 has to be kept pressed for 3 seconds at least. We suggest setting the preset when the actuator is in stop. For any further information on the preset function see the **Preset** item on page 127 (EtherNet/IP interface) or on page 210 (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 the 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

5.1 Quick setting and main functions

The following instructions allow the operator to quickly and safely set up the rotary actuator in a standard operational mode and to execute its main functions.
Sometimes a function or a procedure can be accomplished by using alternative ways:

- by means of a software tool such as Studio 5000 from Rockwell Automation (see the "5.5 Actuator installation under Studio 5000 design environment" section on page 59 ff);
- by means of the Integrated Web Server (see the "Integrated Web Server" section on page 153);
- or via hardware by means of the internal rotary switches (see the "4.5.3 Setting the node ID via hardware (DIP A rotary switches)" section on page 46).

They are all mentioned whenever available.
For complete and detailed information please read the mentioned pages thoroughly.

- Mechanically install the device, see on page 35 ff;
- execute the electrical and network connections, see on page 39 ff;
- 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 116 -EtherNet/IP interface; see on page 213 -MODBUS interface); reset the active alarms: switch high ("=1") the **Alarm reset** bit 3 of the **Control Word** (see on page 115 -EtherNet/IP interface; see on page 213 -MODBUS interface). Check the operating condition shown through the LEDs;
- in the software tool install the EDS file, see on page 66 ff;
- in the software tool insert the Lika module and select the actuator type, see on page 69 ff;
- in the software tool set the device name, see on page 69 ff;
- if required, set the IP address and the subnet mask to the node, see here later for alternatives; the default address (software address) set by Lika is **192.168.1.10**;
- the attributes used to specifically configure the rotary actuator are grouped in the Application Object, see the "7.12.5 Class 64h: Application Object" section on page 113; the complete list of the default parameters is available on page 237;

- set a proper value next to the **Distance per revolution** item (see on page 122 -EtherNet/IP interface; see on page 205 -MODBUS interface);
- set a proper value next to the **Jog speed** item (see on page 126 -EtherNet/IP interface; see on page 209 -MODBUS interface);
- set a proper value next to the **Work speed** item (see on page 126 -EtherNet/IP interface; see on page 209 -MODBUS interface);
- if required, set a proper value next to the **Preset** item (see on page 127 -EtherNet/IP interface; see on page 210 -MODBUS interface);
- set the limit switch values next to the **Max delta pos** / **Positive delta** and **Max delta neg** / **Negative delta** items (see on page 124 -EtherNet/IP interface; see on page 207 -MODBUS interface);
- set the commanded position next to the **Target position** item (see on page 118 -EtherNet/IP interface; see on page 216 -MODBUS interface);
- save the new setting values (**Save parameters** command; see on page 116 -EtherNet/IP interface; see on page 214 -MODBUS interface).

Use the **Jog +**, **Jog -**, **Start** and **Stop** commands in the **Control Word** (see on page 114 -EtherNet/IP interface; see on page 212 -MODBUS interface) to move the axis and reach the commanded position.

NOTE
The parameters **Distance per revolution**, **Jog speed**, **Work speed**, **Preset**, **Max delta pos** / **Positive delta** and **Max delta neg** / **Negative delta** are closely related, hence you have to be very attentive when you need to change the value in any of them. For any further information please refer to page 86.

5.1.1 Setting the node address

The node address and the network-related parameters can be set either via software or via hardware.
Software configuration:

- set the **IP Address**, **Network Mask** and **Gateway Address** parameters in the **F5-01-05 Interface Configuration** attribute, see the "7.12.6 Class F5h: TCP/IP Interface Object" section on page 136; both DIP A rotary switches are set to 00h, see the "4.5 EtherNet/IP Node ID" section on page 44);
- set the parameters in the Integrated Web Server, see the "8.7 Network configuration" section on page 166; both DIP A rotary switches are set to 00h, see the "4.5 EtherNet/IP Node ID" section on page 44);
- enable a DHCP Server as follows (both DIP A rotary switches are set to 00h; or both are set to FFh,value 255_{10}):
  - see the **F5-01-03 Configuration Control** attribute, see the "7.12.6 Class F5h: TCP/IP Interface Object" section on page 136;

- enable the DHCP Server in the Integrated Web Server, see the "8.7 Network configuration" section on page 166.

Hardware configuration:

- set the DIP A rotary switches to 00h to enable the software IP address, Subnet mask, and Gateway address that are saved internally, see the software configuration above;
- set the DIP A rotary switches to any value in the range between 01h and FEh (254_{10}). The Subnet mask is 255.255.255.0;
- set the DIP A rotary switches to FFh (255_{10}) to enable the use of a DHCP Server.

5.1.2 Setting a custom resolution

To set a custom resolution you can choose among the following methods.

- Set a proper value next to the **64-01-05 Distance per Revolution** attribute, see on page 122.
- you can also use the Integrated Web Server, see the "8.6 Setting the attributes" section on page 163;
- otherwise you can use a software tool, see the "5.5.11 Configuring the actuator" section on page 72.

5.1.3 Reading the absolute position

To read the position value you can choose among the following methods.

- To read the current position of the actuator see the **64-01-04 Real Position** attribute on page 121;
- open the Integrated Web Server, see the "8.3 Actuator position and Status Word information page" section on page 156; see the "8.4 RD1xA-EP Information (EtherNet/IP attributes)" section on page 158;
- open the **Monitor Tags** tabbed page in your project, see the "5.5.9 Checking the communication" section on page 71.

5.1.4 Setting and executing the preset

To set and execute the preset you can choose among the following methods.

- Enter a suitable value next to the **64-01-12 Preset** attribute, see on page 127; the preset value is activated as soon as the value is confirmed;
- if you need to activate in a different physical position of the actuator shaft the value that has been already set next to the **64-01-12 Preset** attribute, you can use the bit 11 **Setting the preset** in the **64-01-01 Control Word** attribute, see on page 117;
- open the **Set RD1xA-EP Preset** page in the Integrated Web Server, see the "8.5 Setting the Preset value" section on page 160;

- use the Test_RD1xA_EP.acd sample program, you can find it in the **Test_RD1xA_EP.zip** compressed file. Refer also to the "5.5.12 How to create a sample program and send parameters" section on page 73.

5.1.5 Saving data

To save the parameters permanently you can choose among the following methods.

- Use the Class Service 16h available for the Application Object, see on page 113;
- set the bit 9 **Save parameters** in the **64-01-01 Control Word** attribute to 1 and then back to 0, see on page 116;
- use the **Save Parameters** function in the **Set RD1xA-EP Registers** page of the Integrated Web Server, see the "8.6 Setting the attributes" section on page 163.

5.1.6 Restoring defaults

To restore the default parameters you can choose among the following methods.

- Use the Class Service 15h available for the Application Object, see on page 113;
- set the bit 10 **Load default parameters** in the **64-01-01 Control Word** attribute to 1 and then back to 0, see on page 116;
- use the **Load Default Param.** function in the **Set RD1xA-EP Registers** page of the Integrated Web Server, see the "8.6 Setting the attributes" section on page 163.

5.2 About Lika actuators

Lika rotary actuators are **2B hex type devices** and comply with the specifications reported in the Chapter 6 "Device Profiles, Generic Device Type 2B hex" of the publication "THE CIP NETWORKS LIBRARY, Volume 1, Common Industrial Protocol (CIP™)".
The Object Model of a generic device is represented in the following picture:

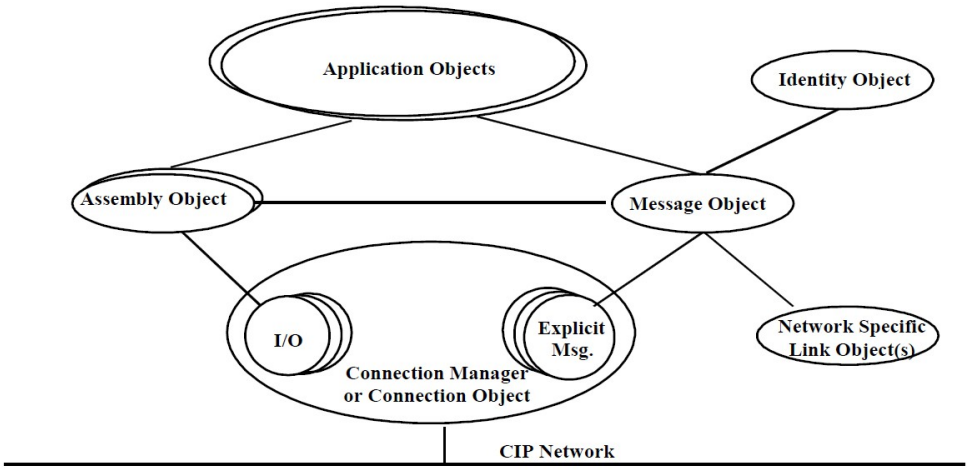


Figure 1 - Object model

The attributes that are used to specifically configure the actuator and make it operational are all grouped in the Application Object, refer to the "7.12.5 Class 64h: Application Object" section on page 113.

5.2.1 Network identity

Lika EtherNet/IP rotary actuators use the following identity settings available in the Identity Object, see the "7.12.1 Class 01h: Identity Object" section on page 103:

Identity Name: **Vendor ID**
Attribute: **01-01-01 Vendor ID**
Setting: **0299h = 665dec = Lika Electronic Srl**

Identity Name: **Device Type**
Attribute: **01-01-02 Device type**
Setting: **002Bh: Generic Device Profile**

Identity Name: **Product Code**
Attribute: **01-01-03 Product code**
Setting: **3000h RD1xA rotary actuator**

Identity Name: **Revision**
Attribute: **01-01-04 Revision**
Setting: **device dependent**

Identity Name: **Serial Number**
Attribute: **01-01-06 Serial number**
Setting: **device dependent**

Identity Name: **Product Name**
Attribute: **01-01-07 Product name**
Setting: **RD1xA Rotary Actuator**

5.2.2 Network and communication settings

The **MAC address** of the device is always reported in the label applied to the actuator enclosure. See on page 34.
The **EtherNet/IP Node ID** can be set both via software and via hardware using the DIP A rotary switches located inside the actuator enclosure. By default it is set via software and its value is 192.168.1.10. See on page 44.

5.3 Configuring the actuator with Studio 5000 V30.00 from Rockwell Automation

In this manual some screenshots are shown to explain how to install and configure the actuator in a supervisor. In the specific example the development environment is Studio 5000 V30.00 from Rockwell Automation; it is used in combination with CompactLogix 5370 L1 Controller "1769-L16ER-BB1B/B" series from Allen Bradley. Therefore, the information on the installation of the EDS 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 design environment. 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.
In the following pages the Controller is assumed to have 192.168.1.20 IP address and 255.255.255.0 Subnet mask.

Lika Electronic EtherNet/IP actuator documentation is complete with a **sample project** supplied free of charge. This program is designed to make your own project planning, programming, communication and diagnostics with Studio 5000 V30.00 design environment user-friendly and reliable. You can find it in the **Test_RD1xA_EP.zip** compressed file.

NOTE
Lika Electronic does not accept responsibility for any loss or damage that you could suffer as a result of using the example project. It is provided "as is", without warranty of any kind, express or implied. In no event shall Lika Electronic be liable for any claim, damages or other liability arising from, out of or in connection with the program or the use or other dealings in the program.

5.4 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 also be read next to the **F6-01-03 Physical Address** attribute. Refer to the "7.12.7 Class F6h: Ethernet Link Object" section on page 141.
It is further shown in the **RD1xA-EP Information** page of the web server under the title of the page. Refer to the "8.4 RD1xA-EP Information (EtherNet/IP attributes)" section on page 158.

5.5 Actuator installation under Studio 5000 design environment

5.5.1 Description of the EDS file

The functionality of an EtherNet/IP device is always described in an EDS file (Electronic Data Sheet file). The Electronic Data Sheet file provides information about the device basic communication and functional properties. It must be installed in the Controller.

EtherNet/IP rotary actuators from Lika Electronic are supplied with their own EDS file.
Specific EDS files are provided to each RD actuator series and specific models, please refer to the order code.
RD1xA actuator's EDS files are:

- **RD1xA_EP_HxSx.eds**: it is intended for installation of **both RD1A and RD12A series actuators** ("RD1xA" is the actuator series; "EP" is the Lika code that identifies the EtherNet/IP protocol; "Hx" is the hardware version of the actuator; "Sx" is the software version of the actuator); see the order code, for instance: RD1A-P8-T48-EP-... .

The version of the EDS file is reported under the "Version" item inside the file.
EDS files can be paired with the **RD1xA_48x48.ico** picture file available inside the file folder (the picture is also integrated into the EDS file).

Follow the path **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS** to download the EDS files from Lika's corporate web site.

5.5.2 Configuring the network interface controller (NIC) of the computer

To set the computer's IP address in Windows, type *network and sharing* into the **Search** box in the **Start** menu and select **Network and Sharing Center** when the **Control Panel** comes up.

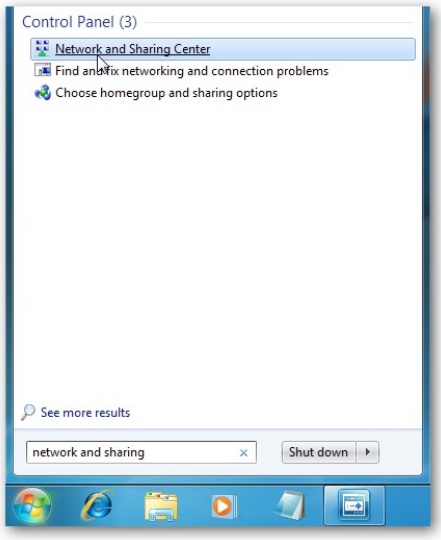


Figure 2 - Network and Sharing Center

Then when the **Network and Sharing Center** opens, click on **Change adapter settings**.

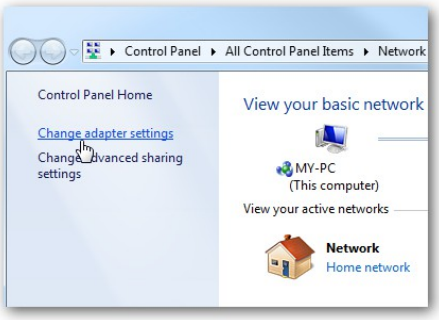


Figure 3 - Change adapter settings

Right-click on your local adapter and select **Properties**.

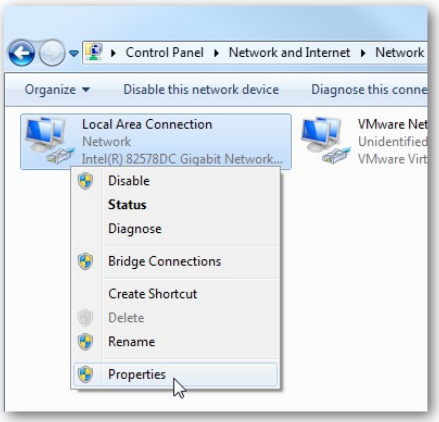


Figure 4 - Local Area Connection properties

In the **Local Area Connection Properties** window highlight *Internet Protocol Version 4 (TCP/IPv4)*, then click the **Properties** button.

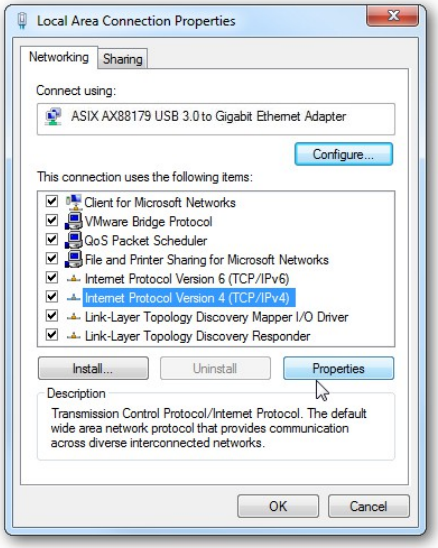


Figure 5 - Internet Protocol Version 4 properties

Now select the **Use the following IP address** radio button and enter in the correct IP, Subnet mask, and Default gateway that corresponds with your network setup. Then, if required, enter your Preferred and Alternate DNS Server addresses. We suggest setting a simple Class C network configuration such as 192.168.1.xx as the default software IP address of the actuator has this NET ID.
Check **Validate settings upon exit** so Windows can find any problems with the addresses you entered. When you are finished click **OK**.

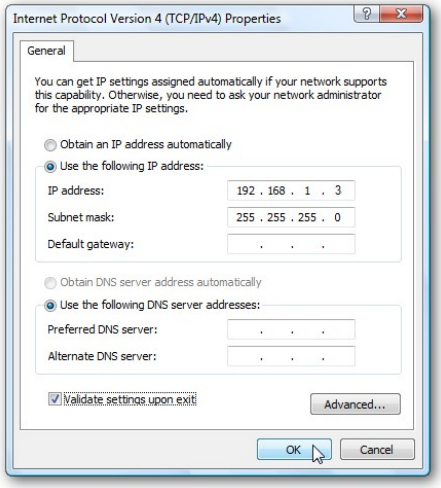


Figure 6 - Setting the IP Address

5.5.3 Networking the PC and the Controller

Use a Category 5 minimum cable to network the Ethernet port of the PC to the Ethernet port of the Controller.

5.5.4 Configuring the driver

Launch the **RSLinx Classic** communication software and then open **RSWho** by pressing **Communication** and then the **RSWho** command.
Again in the menu bar of the main page press **Communication** and then the **Configure Drivers** command.
The **Configure Drivers** dialog box will appear.

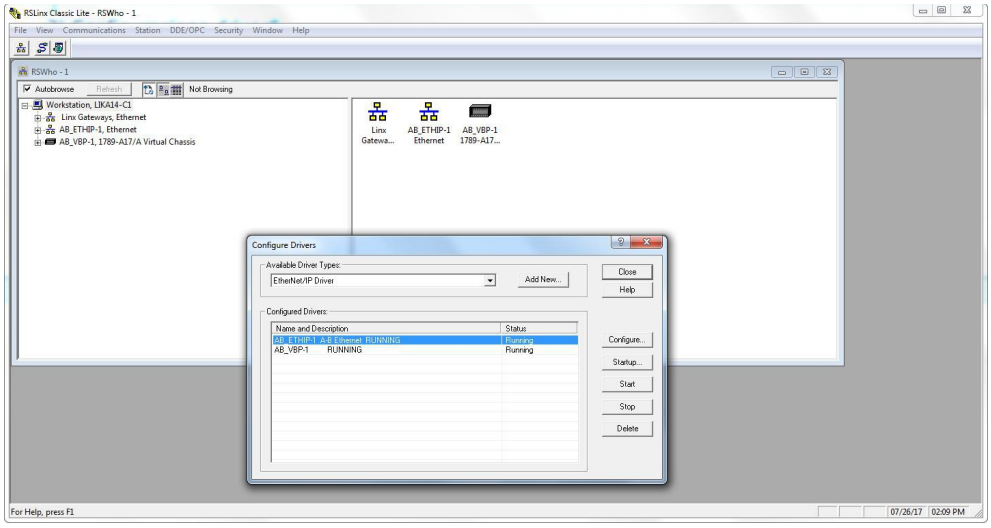


Figure 7 - Configure Drivers

From the **Configure Drivers** dialog box, select the desired driver from the **Available Driver Types** list.
Click **Add New**. The **Add New RSLinx Classic Driver** dialog box opens.
Enter a name for the selected driver (15 characters at maximum), and click **OK**.
The **Configuration** dialog box for that driver shows.
In the **Configuration** dialog box, enter the appropriate parameters for the desired driver.
Click **OK** to close the **Configuration** dialog box. The new driver now appears in the **Configured Drivers** list.
Press **Close** to close the dialog box.

Now right-click the driver you have just installed and press **Configure Driver**.

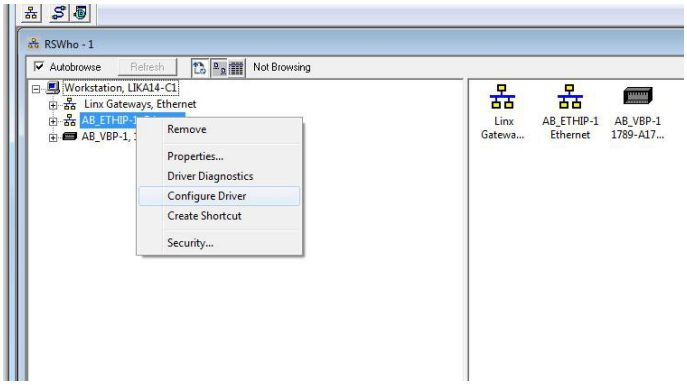


Figure 8 - Configure Driver

In the **Configure Driver** dialog box, select the network interface controller you configured and connected to the PLC; finally press **OK** to confirm.

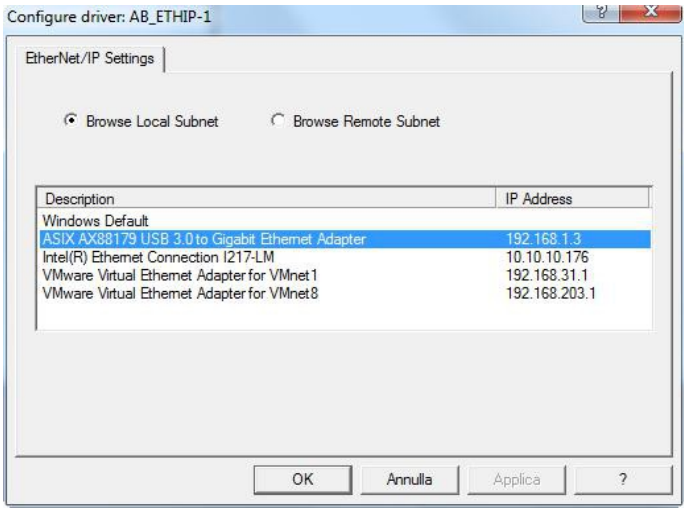


Figure 9 - Browse Local Subnet

5.5.5 Starting a new project

Double-click on the **Studio 5000** icon on your Desktop to launch Studio 5000 software. The Studio 5000 Splash Screen appears.
Select **New Project** under the **Create** section.

Figure 10 - Studio 5000 New Project

When the **New Project** pop-up is displayed, select **Logix** and the type of controller (such as "1769-L16ER-BB1B", in the example). Enter the name of the project and the path where the file has to be saved.

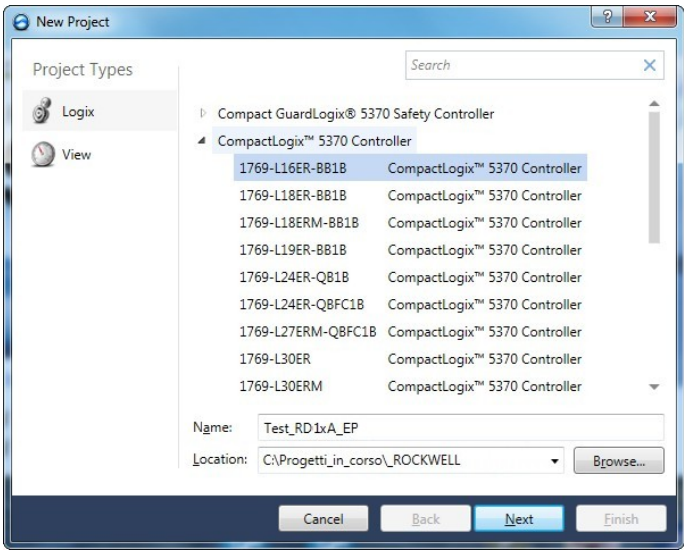


Figure 11 - New Project

Press the **Next** button and then set the **Revision** and the **Expansion I/O** settings. Finalize by pressing the **Finish** button.

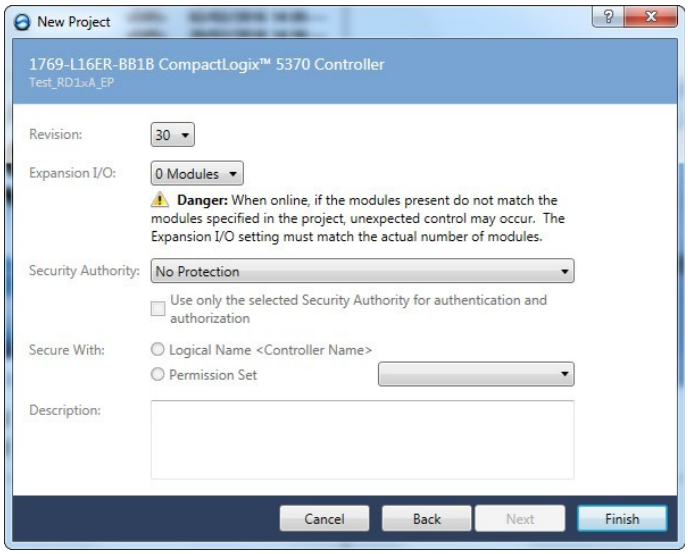


Figure 12 - Controller's settings

5.5.6 Installing the EDS file

To manually register the EDS files of the actuator in the **EDS Hardware Installation Tool**, perform the following steps.

Launch the **EDS Hardware Installation Tool** by pressing **Tools** and then the **EDS Hardware Installation Tool** command.
The **Rockwell Automation's EDS Wizard** dialog box opens.
On the **Options** screen select **Register an EDS file(s)**, then press **Next**.

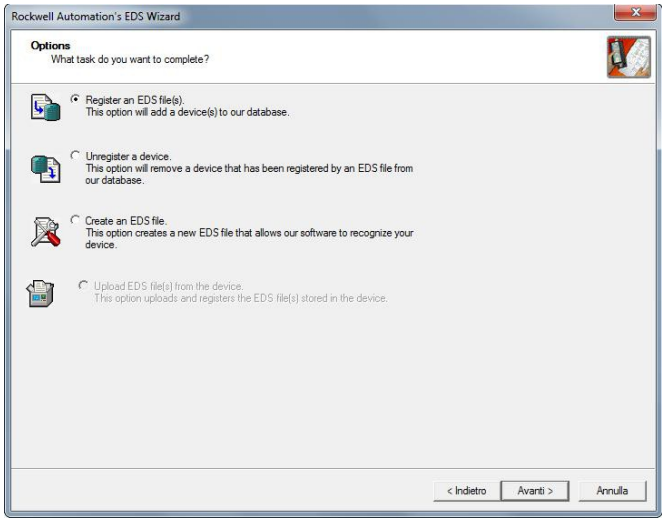


Figure 13 - EDS Wizard

On the **Registration** screen select **Register a single file** to register one EDS file at a time, and click **Browse** to select the EDS file corresponding to the actuator to be installed (RD1xA_EP_H2S1.eds in the screenshot Figure 14, please check the order code) and press **Next** button until the registration is finalized.

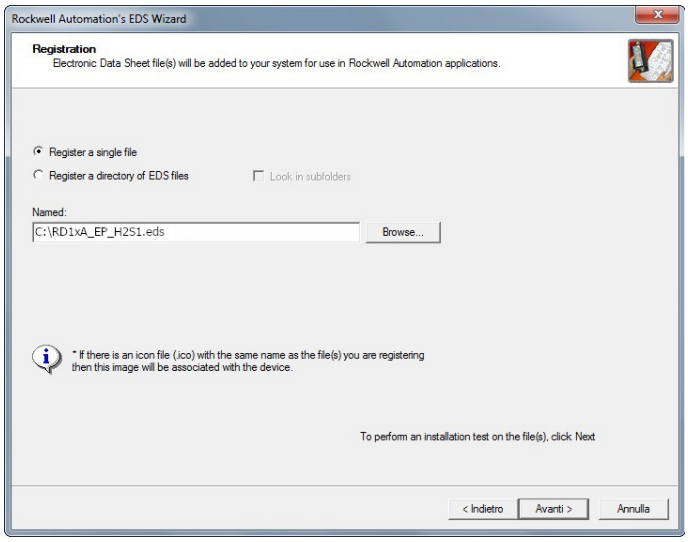


Figure 14 - EDS Wizard

5.5.7 Defining the communication path

To define a path to the controller click on the icon shown in Figure 15.

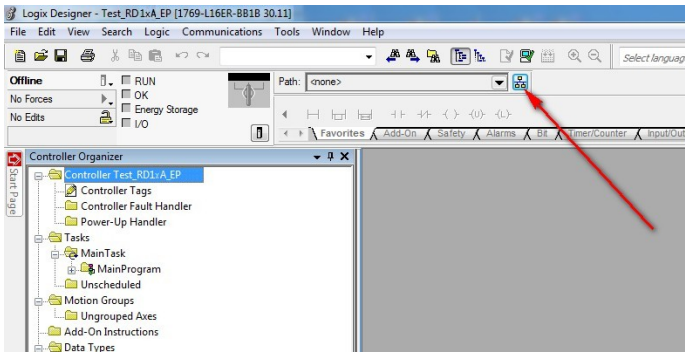


Figure 15 - Path to Controller

Browse to the Controller, select it and click the **Set Project Path** button.

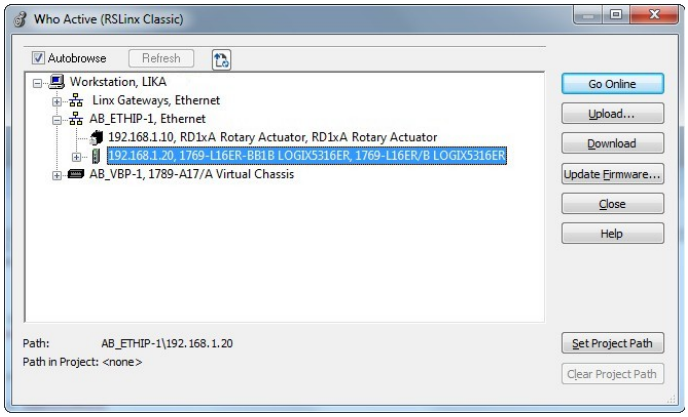


Figure 16 - Set Project Path

Close the dialog box: the selected path will appear on the main page.

Figure 17 - Project Path set

5.5.8 Adding the actuator to the project

On the **Controller Organizer**, right-click on **Ethernet** and select **New Module ...** from the pull-down menu.

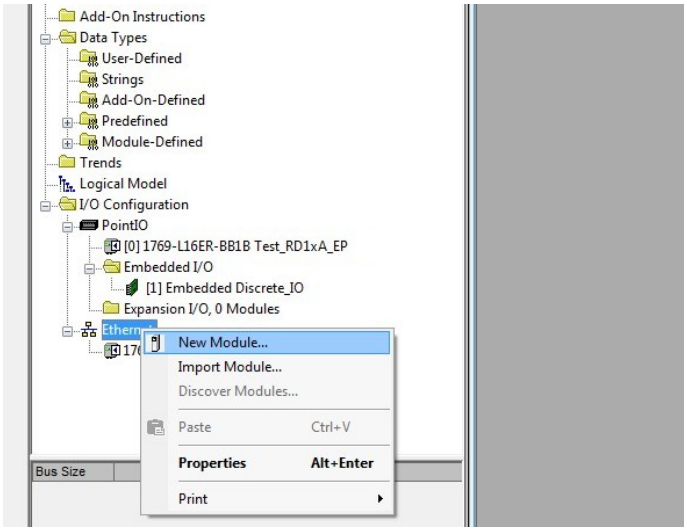


Figure 18 - New Module

On the **Select Module Type** dialog box select the installed actuator module (RD1xA-Txx-... in the screenshot, Figure 19). Click **Create**.

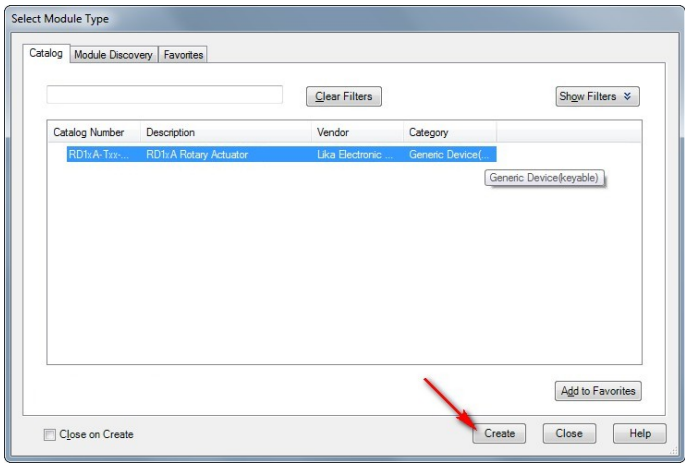


Figure 19 - Select Module Type

Configure the actuator module by setting the required parameters **Name** and **Ethernet Address**. Then press **Change...** button to select the connection type.

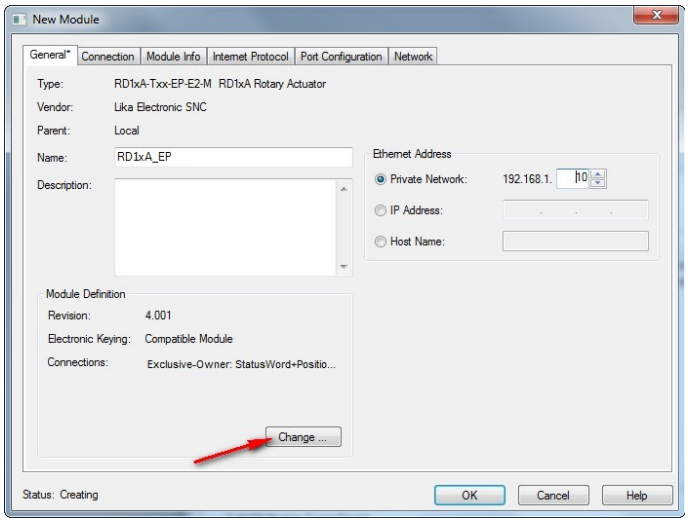


Figure 20 - New module configuration

Select the required connection type and then click **OK**. For more information on the available connection types refer to the "7.12.3.4 Supported connection types" section on page 108.

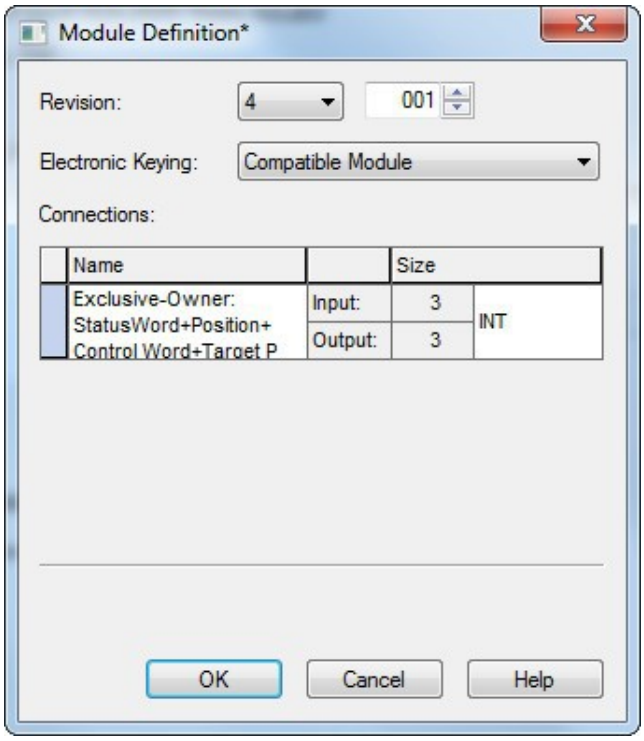


Figure 21 - Select connection type

In the example an Exclusive Owner connection has been set: the actuator will send the position value and Status Word information (Connection point T → 0: Assembly Object, instance 01h, Producing Instance), while the Controller will send both commands cyclically (Connection point 0 → T: Assembly Object, instance 96h, Consuming Instance) and parameters configuration at switching on (the actuator will receive configuration data).
Press **OK** to finalize and **YES** in the next dialog box.

Close the **New Module** and **Select Module Type** dialog boxes.

Connection point 0 → T	Assembly Object, instance 96h (Consuming Instance)
Connection point T → 0	Assembly Object, instance 01h (Producing Instance)

5.5.9 Checking the communication

You can check whether the communication between the Controller and the actuator is established properly by displaying the actuator parameters.
On the **Controller Organizer**, double-click on **Controller Tags** in the **Controller Test_RD1xA_EP** folder: the actuator parameters will be displayed in the **Monitor Tags** tabbed page. The **Monitor Tags** page displays the tags.

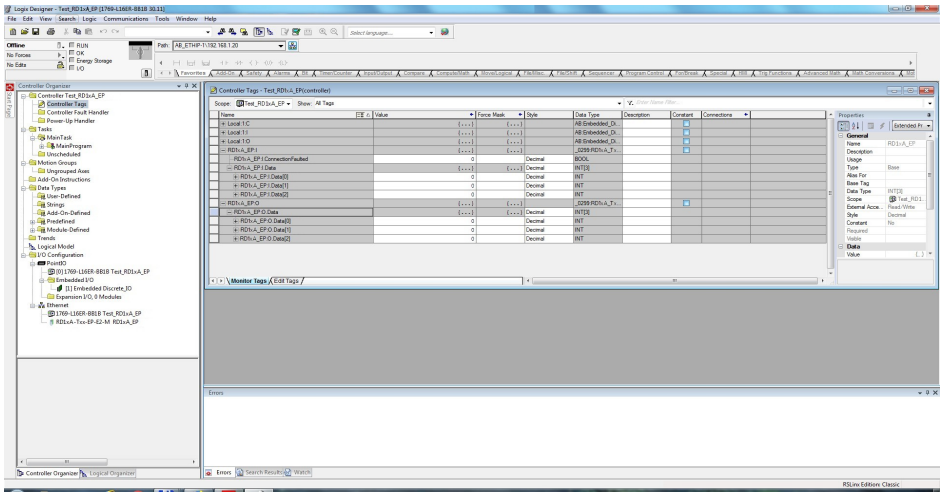


Figure 22 - Monitor Tags

5.5.10 Downloading the configuration to the Controller

To download the configuration to the Controller you must go online first.
Press the drop-down box between the **Offline** and **RUN** items and select **Go Online** in the pull-down menu.

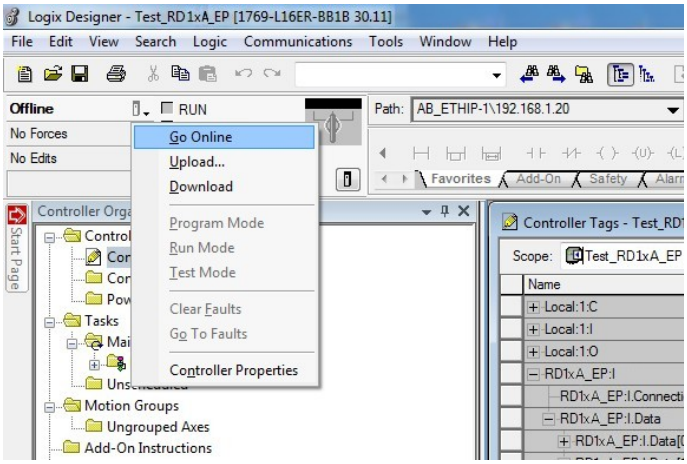


Figure 23 - Going online

Press **Download** in the **Who Active** window to start the download process; the **Download** window will be displayed. Before pressing the **Download** button once more please note the cautionary messages. Click **Download** to continue the download process.
When the download process is completed, the Controller may return to Remote Program mode or ask whether you want to return to Run mode. The message you see is determined by the state the Controller was in at the beginning of the download process.
If everything went well, the NS LED of the actuator lights up green (the actuator is online) while the MS LED blinks green (the Scanner is in **Idle** state). Refer to the "4.7 Diagnostic LEDs (Figure 4 and Figure 8)" section on page 48).

5.5.11 Configuring the actuator

Before executing the download process, you can set the configuration parameters of the actuator.
On the **Controller Organizer**, right-click **Controller Tags** and choose **Monitor Tags**: the Tag Monitor displays the tags.
A blue arrow indicates that when you change the value, it immediately takes effect.
To see a value in a different style, select the desired style.
To change a value, click the **Value** cell, type the new value, and click **ENTER**.
To expand a tag and show its members, click the **+** sign.

WARNING
Parameters are not saved on the non-volatile memory. At next power-on you are required to send them again.
To save the parameters permanently you can choose among the following methods: by means of the Class Service 16h, see on page 113; or by setting the bit 9 **Save parameters** in the **64-01-01 Control Word** attribute to 1 and then back to 0, see on page 116; or by using the **Save Parameters** function in the **Set RD1xA-EP Registers** page of the Integrated Web Server, see the "8.6 Setting the attributes" section on page 163.

5.5.12 How to create a sample program and send parameters

Here follows a description of a simple program created using "Structured Text" programming language. The program allows to send a preset "1000" to the actuator by means of EtherNet/IP explicit messages with CIP protocol. See also the **64-01-12 Preset** attribute on page 127.
Described program can be used as a base to build further programs: the procedure is exactly the same, you have just to change the Attribute value and the type of variable of the parameter.

NOTE
As previously stated, Lika Electronic EtherNet/IP actuator documentation is complete with a **sample project Test_RD1xA_EP.acd** supplied free of charge. This program is designed to make your own project planning, programming, communication and diagnostics with Studio 5000 V30.00 design environment user-friendly and reliable. You can find it in the **Test_RD1xA_EP.zip** compressed file.

- **Test_RD1xA_EP.acd** program mainly allows the user to set and execute the preset. It uses 4 inputs of the PLC: input 0 is used to execute a JOG + command (see the **Jog +** bit); input 1 is used to execute a JOG - command (see the **Jog -** bit); input 2 is used to enter the preset value (see the **64-01-12 Preset** attribute); input 4 is used to execute the preset command (see the **Setting the preset** bit). When you force high (+24V) the input, the matched function is activated. It is summarily described in the following pages.

Each program requires a main routine. Once you create your routines, assign a main routine for each program.
On the **Controller Organizer**, expand the **Tasks**, **MainTask** and **MainProgram** folders and right-click on **MainProgram**. Select **Add** and **New Routine...** in the pull-down menu.

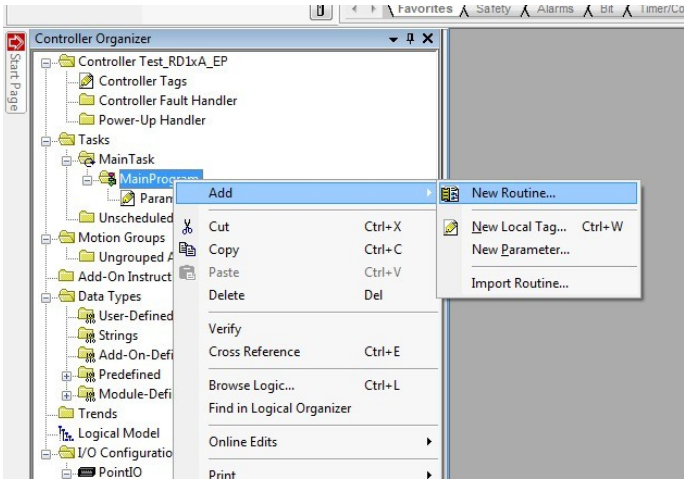


Figure 24 - Adding a main routine

In the **New Routine** screen enter the name of the new routine next to the **Name** item and select the "Structured Text" language in the **Type** drop-down box. Confirm pressing the **OK** button.

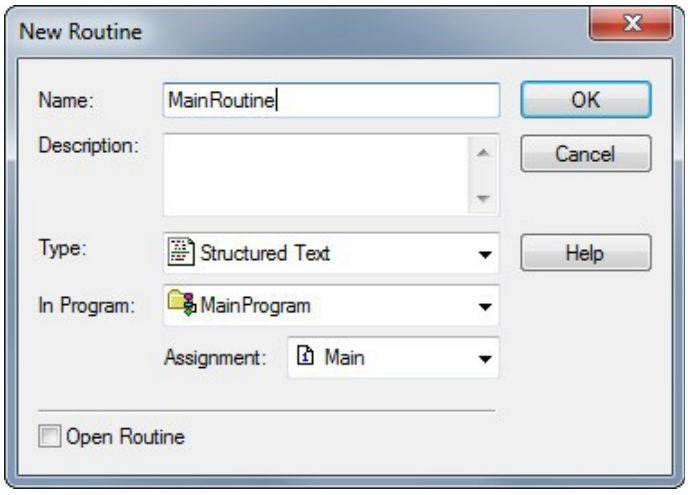


Figure 25 - Assigning the main routine features

If you need to delete an existing main routing, right-click on **MainRoutine** and then select **Delete** command in the pull-down menu.

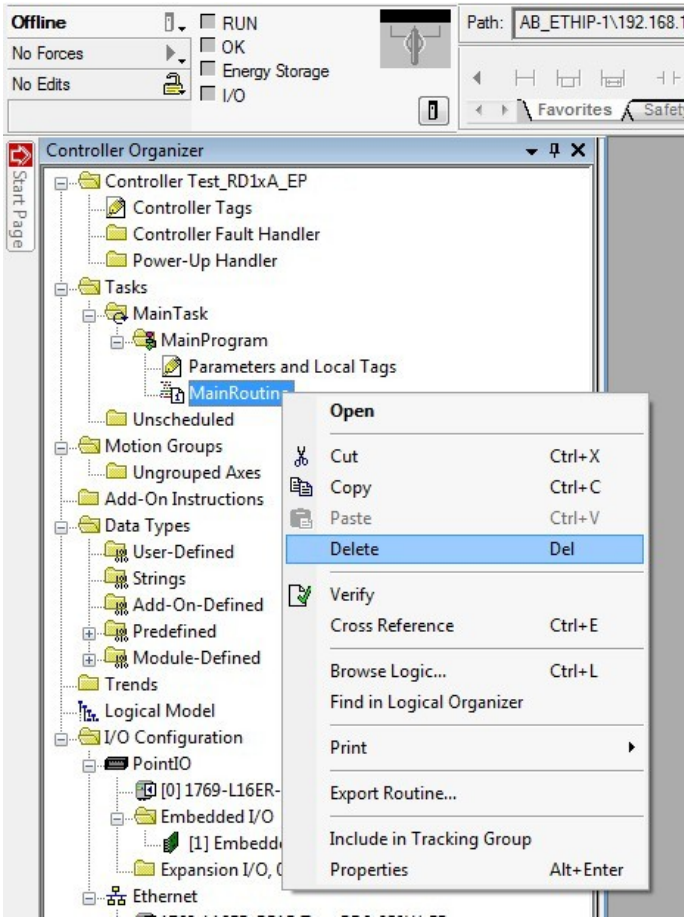


Figure 26 - Deleting a main routine

We need to create some tags (variables) that are needful for the program.
On the **Controller Organizer**, right-click on **Controller Tags** and select **New Tag...** from the pull-down menu.

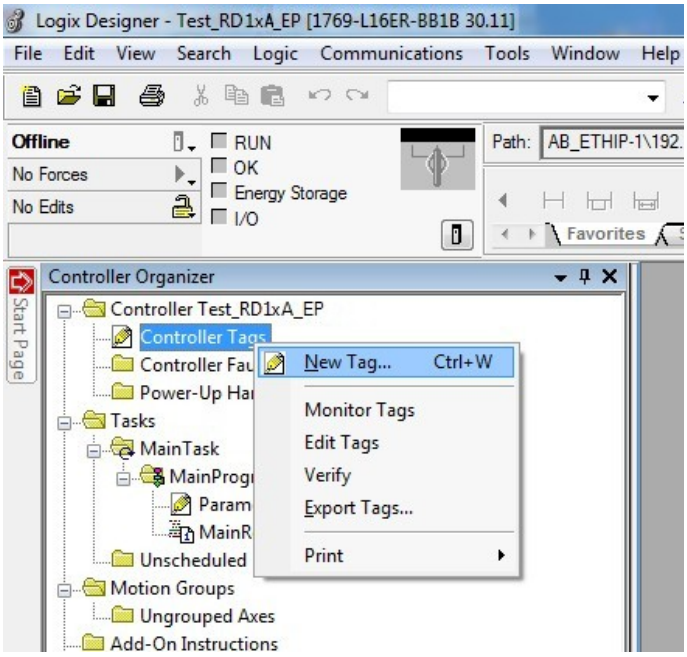
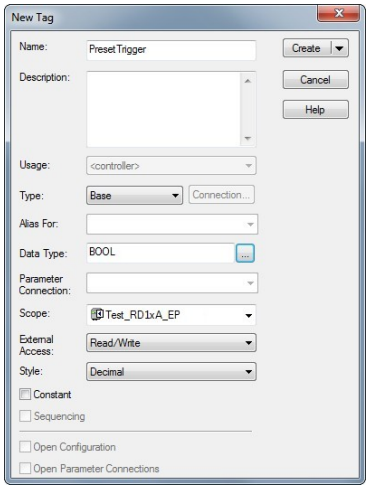


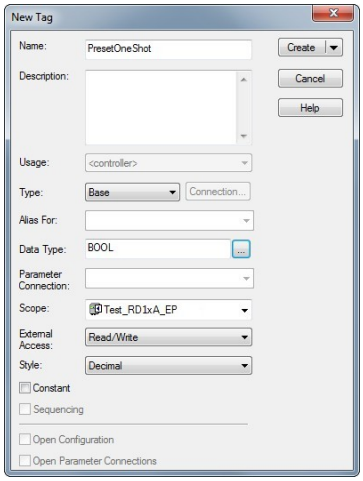
Figure 27 - New Tag

The following tags need to be created:

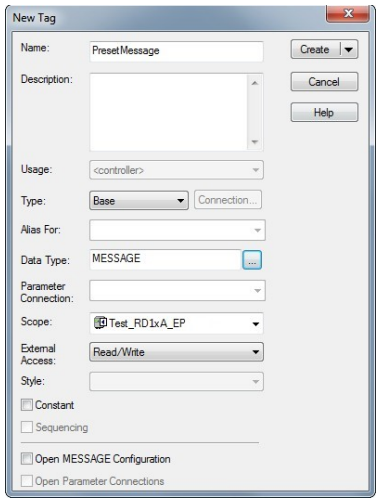
PresetTrigger tag, data type BOOL



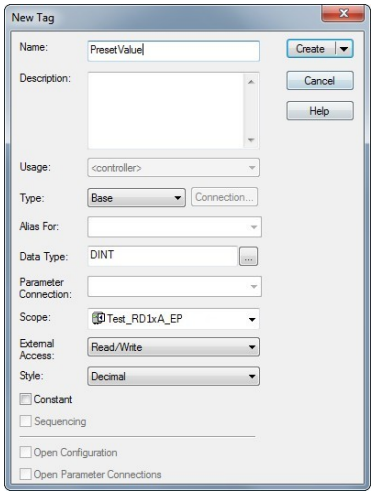
PresetOneShot tag, data type BOOL



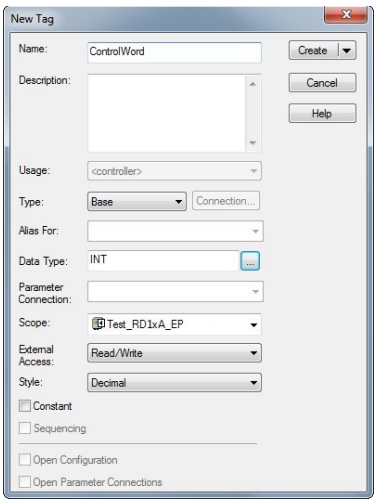
PresetMessage tag, data type MESSAGE



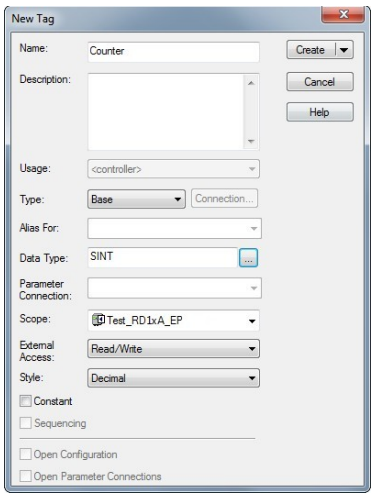
PresetValue tag, data type DINT



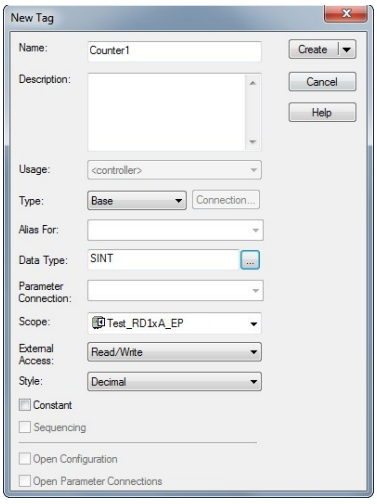
ControlWord tag, data type INT



Counter tag, data type SINT



Counter1 tag, data type SINT



NOTE
You can type any name for the tags.

Now double-click the **Controller Tags** item on the **Controller Organizer**, the list of the tags just created will appear. In the **Value** field of the **PresetValue** tag enter the desired preset value, e.g. "1000".

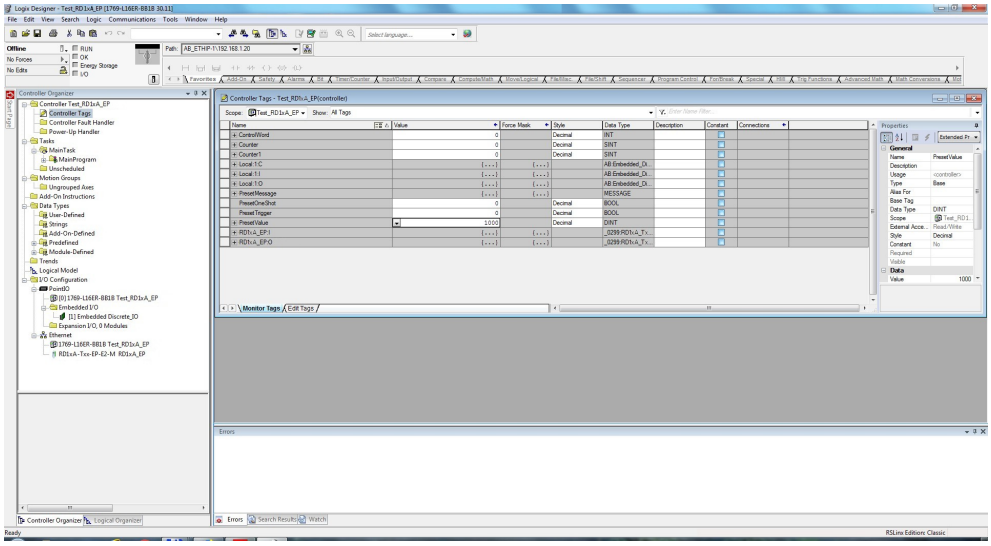


Figure 28 - Controller tags and PresetValue

Then right-click on the **PresetMessage** tag and press the **Configure "PresetMessage"** command.

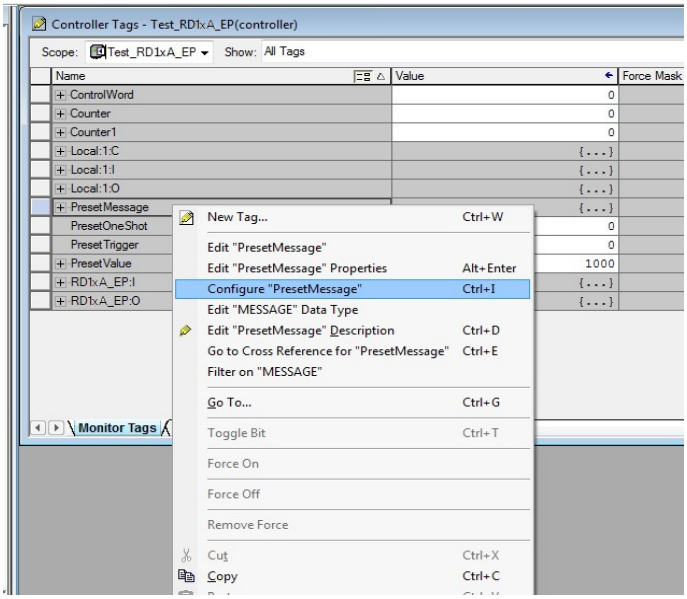


Figure 29 - Configuring the PresetMessage

The **Message Configuration – PresetMessage** property sheet will appear.

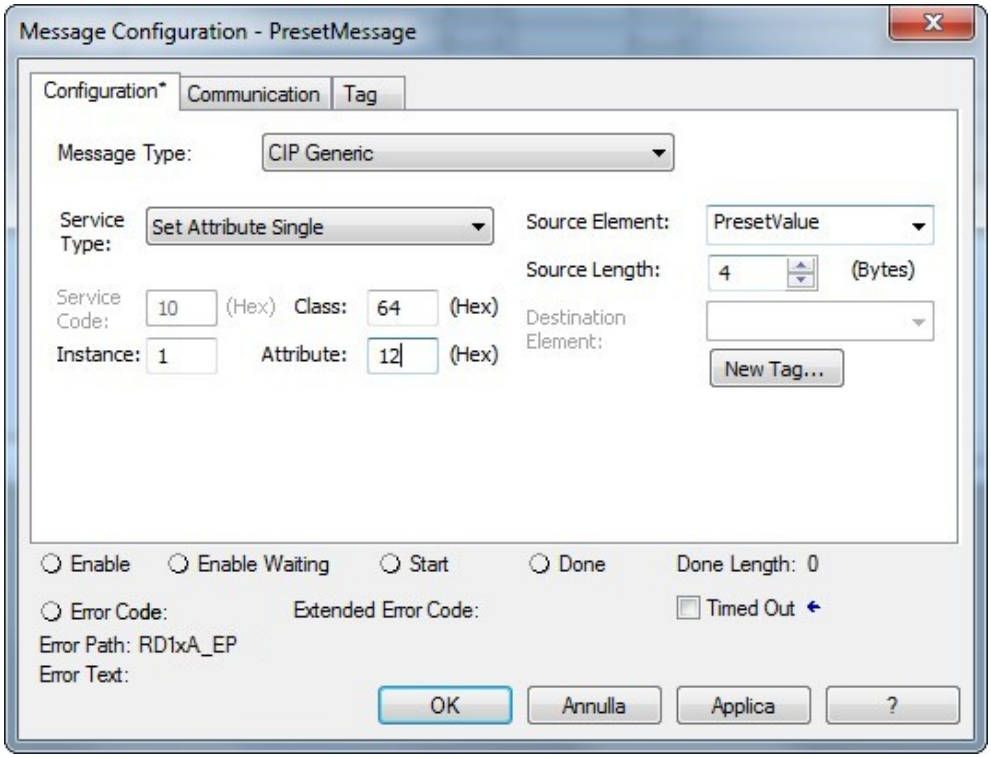


Figure 30 - Message configuration property sheet

You need to configure the **PresetMessage**. Configure both the **Configuration** and the **Communication** tabbed pages as shown in the following screenshots, Figure 31.

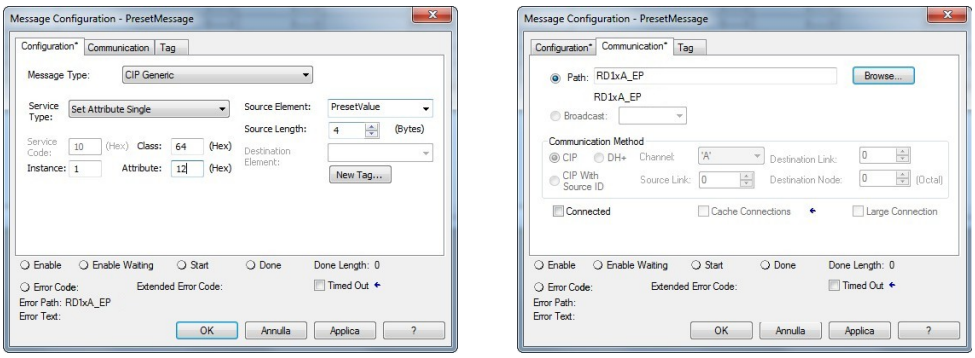
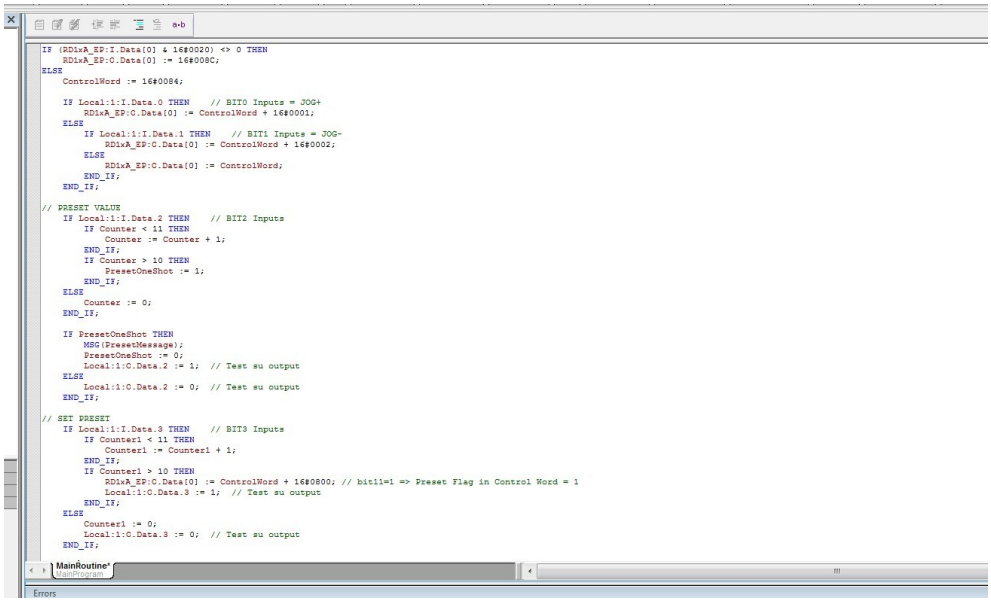


Figure 31 - Message Configuration

In the **Configuration** tabbed page set the **Service Type** to "Set Attribute Single", the **Source Element** to "PresetValue" and the **Source Length** to "4" bytes. Enter Class ("64"), Instance ("1") and Attribute ("12") of the Preset value.
See the **64-01-12 Preset** attribute in the "7.12.5 Class 64h: Application Object" section on page 127.

In the **Communication** tabbed page select the network path to the RD1xA actuator by means of the **BROWSE...** button. Press **OK** button to confirm.

Write the Main Routine program instructions in "Structured Text" language.



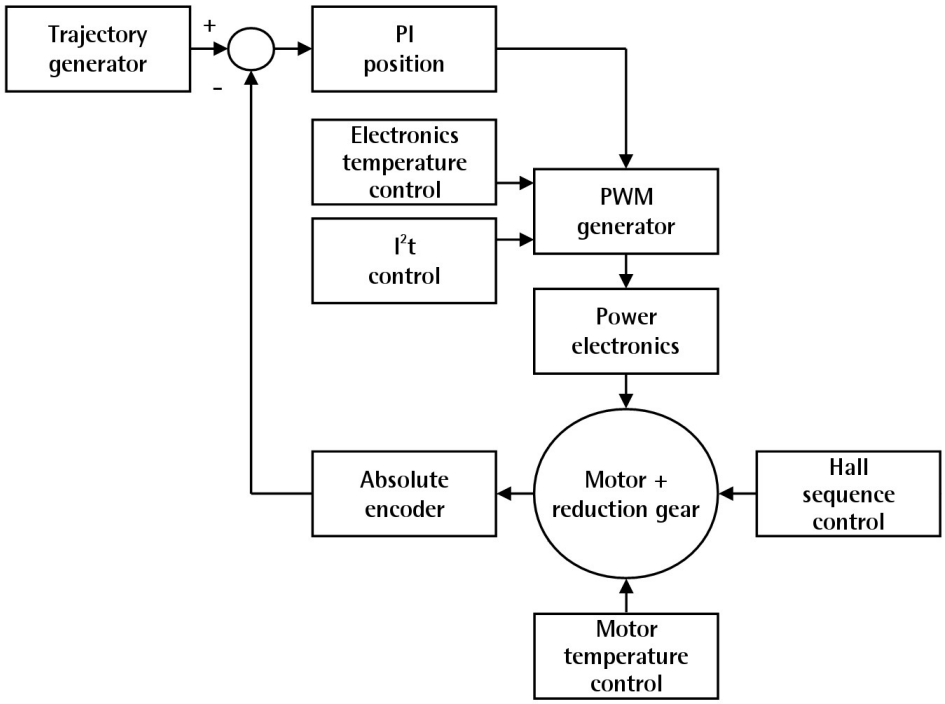
```
IF (RD1xA_EP:I.Data[0] & 16#0020) <> 0 THEN
    RD1xA_EP:O.Data[0] := 16#008C;
ELSE
    ControlWord := 16#0084;

    IF Local:1:I.Data.0 THEN    // BIT0 Inputs = JOG+
        RD1xA_EP:O.Data[0] := ControlWord + 16#0001;
    ELSE
        IF Local:1:I.Data.1 THEN    // BIT1 Inputs = JOG-
            RD1xA_EP:O.Data[0] := ControlWord + 16#0002;
        ELSE
            RD1xA_EP:O.Data[0] := ControlWord;
        END_IF;
    END_IF;

// PRESET VALUE
    IF Local:1:I.Data.2 THEN    // BIT2 Inputs
        IF Counter < 11 THEN
            Counter := Counter + 1;
        END_IF;
        IF Counter > 10 THEN
            PresetOneShot := 1;
        END_IF;
    ELSE
        Counter := 0;
    END_IF;

    IF PresetOneShot THEN
        MSG(PresetMessage);
        PresetOneShot := 0;
        Local:1:O.Data.2 := 1;  // Test su output
    ELSE
        Local:1:O.Data.2 := 0;  // Test su output
    END_IF;

// SET PRESET
    IF Local:1:I.Data.3 THEN    // BIT3 Inputs
        IF Counter1 < 11 THEN
            Counter1 := Counter1 + 1;
        END_IF;
        IF Counter1 > 10 THEN
            RD1xA_EP:O.Data[0] := ControlWord + 16#0800; // bit11=1 => Preset Flag in Control Word = 1
            Local:1:O.Data.3 := 1;  // Test su output
        END_IF;
    ELSE
        Counter1 := 0;
        Local:1:O.Data.3 := 0;  // Test su output
    END_IF;
```

Figure 32 - Main Routine program instructions

6 Functions

6.1 Working principle

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



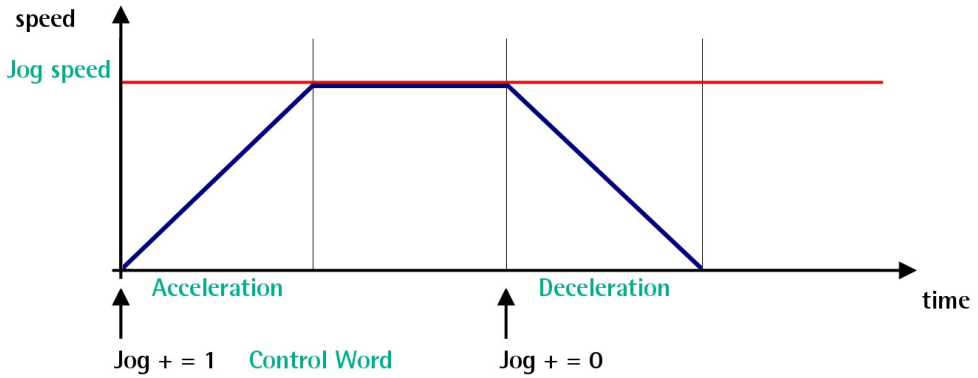
6.2 Movements: jog and positioning

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 **64-01-0F Jog speed** / **Jog speed [0x0D]** parameter.

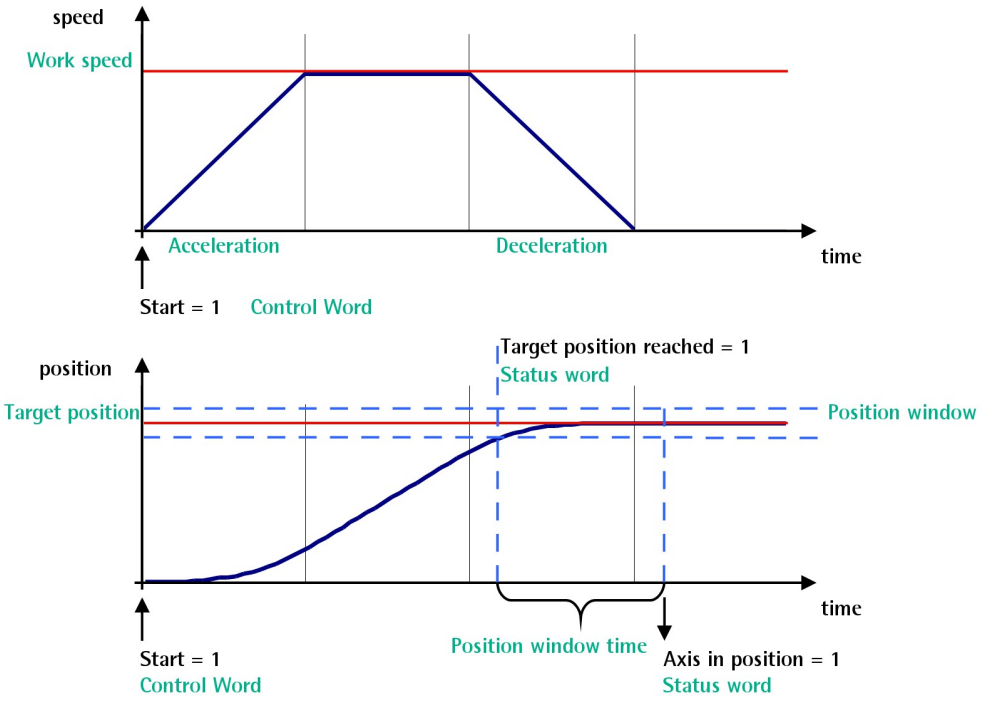


When the bit 0 **Jog +** in the **64-01-01 Control Word** / **Control Word [0x2A]** is "=1", the motor accelerates toward the positive direction according to the value set next to the **64-01-0B Acceleration** / **Acceleration [0x07]** item; if the available travel is long enough it reaches the speed set next to the **64-01-0F Jog speed** / **Jog speed [0x0D]** item. As soon as the bit 0 **Jog +** in the **64-01-01 Control Word** / **Control Word [0x2A]** goes low ("=0"), the motor decelerates according to the value set next to the **64-01-0C Deceleration** / **Deceleration [0x08]** item until it stops.
Setting the bit 1 **Jog -** in the **64-01-01 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 **64-01-0F Jog speed** / **Jog speed [0x0D]** parameter is the speed of the motor, not the speed of the output shaft after the reduction gears.

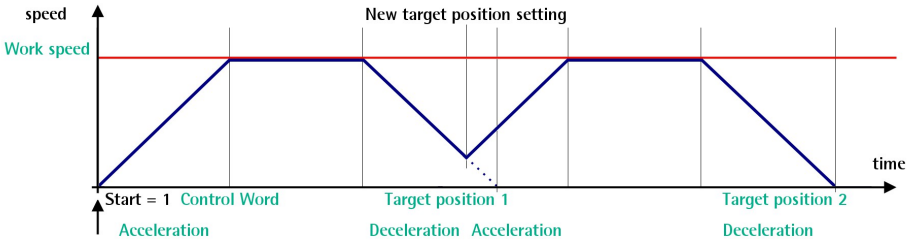
Positioning: position and speed control

This kind of control is a point-to-point movement and the maximum reachable speed is equal to the value set in the 64-01-10 Work speed / Work speed [0x0E] parameter; the set speed can be reached only if the available travel is long enough.



When the bit 6 **Start** in the 64-01-01 Control Word / Control Word [0x2A] is "=1", the motor starts moving and accelerates according to the value set next to the 64-01-0B Acceleration / Acceleration [0x07] item in order to reach the target position as set next to the 64-01-02 Target Position / Target position [0x2B-0x2C] item. If the available travel is long enough it reaches the speed set next to the 64-01-10 Work speed / Work speed [0x0E] item. The movement direction can be either positive or negative according to the target position to reach. As soon as the axis is within the tolerance window limits set next to the 64-01-06 Position Tolerance / Position window [0x01] item, the bit 8 **Target position reached** in the 64-01-03 Status Word / Status word [0x01] goes high ("=1"). When the position is within the tolerance window limits set next to the 64-01-06 Position Tolerance / Position window [0x01] item, after the delay set next to the 64-01-07 Settling time / Position window time [0x02] item, the bit 0 **Axis in position** in the 64-01-03 Status Word / Status word [0x01] goes high ("=1"). The motor decelerates according to the value set next to the 64-01-0C Deceleration / Deceleration [0x08] item in order to reach the halt position according to the set target position.

NOTE
Position override function
It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the **64-01-02 Target Position** / **Target position [0x2B-0x2C]** item.



NOTE
Please note that the value in the **64-01-10 Work speed** / **Work speed [0x0E]** parameter is the speed of the motor, not the speed of the output shaft after the reduction gears.

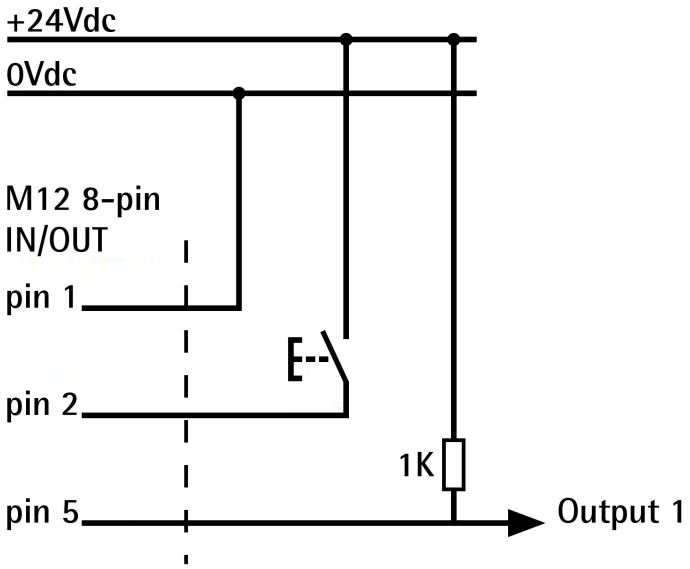
6.3 Digital inputs and output

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 121 ff -EtherNet/IP interface- or page 222 ff -Modbus interface) when the device is running in **Idle** state.
"High" logic value is read when the voltage is equal to +24Vdc ±10%.

The Slave **output** 1 is operated by the Master through the **Control word** (bit 13; see on page 117 -EtherNet/IP interface- or page 212 -Modbus interface) when the device is running in **Idle** state.
It is an "open collector" type output having Imax = 150mA.

Example of connection scheme:



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

The variables Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta are closely related, hence you have to be very attentive every time you need to change the value in any of them.
Should a new setting be necessary, please comply with the following procedure:

- set a proper value next to the Distance per revolution item (see on page 122 -EtherNet/IP interface; see on page 205 -MODBUS interface);
- set a proper value next to the Jog speed item (see on page 126 -EtherNet/IP interface; see on page 209 -MODBUS interface);
- set a proper value next to the Work speed item (see on page 126 -EtherNet/IP interface; see on page 209 -MODBUS interface);
- set a proper value next to the Preset item (see on page 127 -EtherNet/IP interface; see on page 210 -MODBUS interface);
- check the value next to the Max delta pos / Positive delta item (see on page 124 -EtherNet/IP interface; see on page 207 -MODBUS interface);
- check the value next to the Max delta neg / Negative delta item (see on page 125 -EtherNet/IP interface; see on page 208 -MODBUS interface);
- save the new values (**Save parameters** command, bit 9 in the 64-01-01 Control Word / Control Word [0x2A] item, see on page 114 -EtherNet/IP interface; see on page 214 -MODBUS interface).

Each time you change the value in **Distance per revolution** then you must update the value in **Preset** in order to define the zero of the axis as the system reference has changed.
After having changed the parameter in the **Preset** item it is not necessary to set new values for the travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Max delta pos** / **Positive delta** and **Max delta neg** / **Negative delta**.

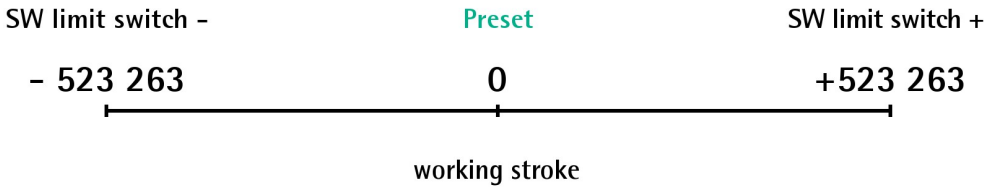
The number of revolutions managed by the system is 512 in negative direction and 512 in positive direction assuming the **Preset** value as a reference (the max. number of revolutions is 1,024).
The value set next to the **Max delta pos** / **Positive delta** item plus the value set in the **Preset** parameter is the maximum forward travel (positive travel) starting from the preset (the value is expressed in pulses).
The value set next to the **Max delta neg** / **Negative delta** item subtracted from the value set in the **Preset** parameter is the maximum backward travel (negative travel) starting from the preset (the value is expressed in pulses).

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

EXAMPLE 1
Default values:
Distance per revolution = 1,024 steps per revolution
Work speed: 2,000 rpm
Preset = 0

Max delta pos / **Positive delta** and **Max delta neg** / **Negative delta** max. values = 523 263 = (1,024 steps per revolution x 512 revolutions) - 1 – 1,024 steps (i.e. 1 revolution for safety reasons) when **Preset** = 0
Max. **SW limit switch +** = 0 + 523 263 = + 523 263 pulses (forward travel)
Max. **SW limit switch -** = 0 – 523 263 = - 523 263 pulses (backward travel)

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

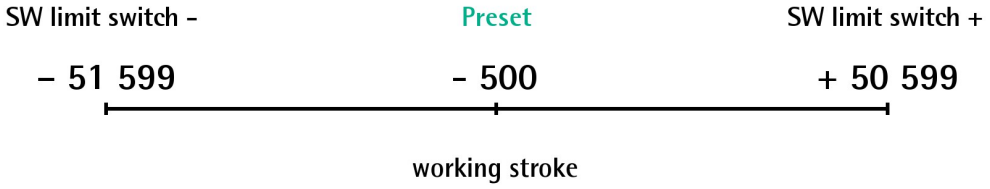


EXAMPLE 2

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

Distance per revolution = 100 steps per revolution
Max. **Work speed** = 293 rpm (100 * 3000 / 1024 = 293)
Preset = -500 (ex. thickness of the tool)
Max. delta pos / **Positive delta** and **Max delta neg** / **Negative delta** = (100 steps per revolution * 512 revolutions) - 1 – 100 steps (i.e. 1 revolution for safety reasons) = 51 099 pulses
Max. **SW limit switch +** = (-500) + 51 099 = + 50 599 pulses (forward travel)
Max. **SW limit switch -** = (-500) – 51 099 = - 51 599 pulses (backward travel)

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



7 EtherNet/IP interface

7.1 Introduction to EtherNet/IP

EtherNet/IP is the name given to the Common Industrial Protocol (CIP), as implemented over standard Ethernet (IEEE 802.3 and the TCP/IP protocol suite). EtherNet/IP was introduced in 2001 and today is the most developed, proven and complete industrial Ethernet network solution available for manufacturing automation, with rapid growth as users seek to harness the advantages of open technologies and the Internet. EtherNet/IP is a member of a family of networks that implements CIP at its upper layers.
EtherNet/IP and CIP are managed by ODVA, see later. ODVA publishes "The EtherNet/IP™ Specification" and helps ensure compliance through conformance testing.

7.2 CIP protocol

The Common Industrial Protocol (CIP) is a media independent, connection-based, object-oriented protocol designed for automation applications. It encompasses a comprehensive set of communication services for automation applications: control, safety, synchronization, motion, configuration and information. It allows users to integrate these applications with enterprise-level Ethernet networks and the Internet. CIP provides users with a unified communication architecture throughout the manufacturing enterprise. CIP allows users to benefit from the many advantages of open networks while protecting their existing automation investments when upgrading in the future.
CIP brings:

- Coherent integration of I/O control, device configuration and data collection.
- Seamless flow of information across multiple networks.
- Ability to implement multi-layer networks without the added cost and complexity of bridges and proxies.
- Minimized investment in system engineering, installation and commissioning.

The "IP" in "EtherNet/IP" refers to "Industrial Protocol". EtherNet/IP utilizes CIP over standard IEEE 802.3 and the TCP/IP protocol suite. Since EtherNet/IP uses standard Ethernet and TCP/IP technologies, it allows compatibility and coexistence with other applications and protocols.

7.3 CIP and International Standards

CIP technologies are compliant with a number of fieldbus-related international standards, and are generally referred to as members of CPF 2 (Communication Profile Family 2) of IEC 61158.

- IEC 61158: Specifies various fieldbus protocols for applications ranging from discrete manufacturing to process control. It includes the specifications for CIP, as well as EtherNet/IP and ControlNet-specific protocol elements, as Type 2.
- IEC 61784-1 and IEC 61784-2: Specify general-purpose and real time Ethernet fieldbus Communication Profiles (CPs) (i.e., how to build a specific communication network using IEC 61158 and other standards). ControlNet, EtherNet/IP and DeviceNet are defined respectively as CP 2/1, CP 2/2 (CP 2/2.1 with CIP Sync), and CP 2/3.
- IEC 61784-3: Specifies Functional Safety Communication Profiles (FSCPs), i.e., extensions of fieldbusses for use in safety related applications. CIP Safety is included as FSCP 2/1.
- IEC 61918 & IEC 61784-5: Specify general and fieldbus-specific cabling installation guidelines. IEC 61784-5 includes specific guidelines for ControlNet, EtherNet/IP and DeviceNet.
- IEC 61800-7: Specifies profiles for power drive systems and their mapping to existing communication systems by use of a generic interface. It includes CIP Motion and its mapping on ControlNet, EtherNet/IP and DeviceNet.
- ISO 15745: Defines elements and rules for application integration, including communication network profiles and the communication aspects of device profiles for some fieldbus technologies. EDS files used for device and network integration of DeviceNet, ControlNet or EtherNet/IP applications are compliant with the relevant parts of ISO 15745 (respectively Parts 2, 3 and 4).

Also:

- The lower layers of EtherNet/IP are based on the various RFC Internet standards for the TCP/UDP/IP suite, on the IEEE 802.3 and ISO Ethernet standards (ISO/IEC 8802-3), without modification or extension.
- CIP Safety (on EtherNet/IP) has been certified for use in applications in systems needing to meet the requirements of IEC 61508 up to and including SIL3.

7.4 EtherNet/IP adaptation to CIP

EtherNet/IP, like other CIP Networks, follows the Open Systems Interconnection (OSI) model, which defines a framework for implementing network protocols in seven layers: physical, data link, network, transport, session, presentation and application. Networks that follow this model define a complete suite of network functionality from the physical implementation through the application or user interface layer. As with all CIP Networks, EtherNet/IP implements CIP at the Session layer and above and adapts CIP to the specific EtherNet/IP technology at the Transport layer and below. This network architecture is shown in Figure 33.
Ethernet has the unique characteristic of being a network with an active infrastructure. Therefore, unlike typical device or control level networks—which generally have a passive infrastructure that limits the number of devices that

can be connected and the way they can be connected—the EtherNet/IP network infrastructure can accommodate a virtually unlimited number of point-to-point nodes, providing users with unsurpassed flexibility in designing networks that accommodate their current requirements while enabling easy, cost-effective expansion in the future.

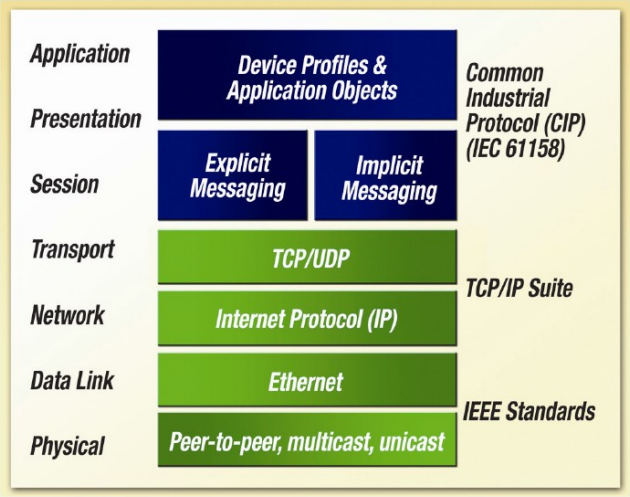


Figure 33 - EtherNet/IP adaptation to CIP

To further decrease complexity, EtherNet/IP systems require only a single point of connection for both configuration and control, because EtherNet/IP supports both I/O (or implicit) messages—those that typically contain time-critical control data—and explicit messages—those in which the data field carries both protocol information and instructions for service performance (see the "7.9.4 Types of EtherNet/IP communications" section on page 98). And, as a producer-consumer network that supports multiple communication hierarchies and message prioritization, EtherNet/IP provides more efficient use of bandwidth than a device network based on a source-destination model. EtherNet/IP systems can be configured to operate either in a Master/Slave or distributed control architecture using peer-to-peer communication.

7.5 The Physical Layer

EtherNet/IP uses standard IEEE 802.3 technology at the Physical and Data Link Layers. This standard provides a specification for physical media, defines a simple frame format for moving packets of data between devices and supplies a set of rules for determining how network devices respond when two devices attempt to use a data channel simultaneously. This is known as CSMA/CD (Carrier Sense Multiple Access/Collision Detection).

As a network with an active infrastructure, EtherNet/IP is typically configured using a series of network segments constructed of point-to-point connections in a star configuration. The core of this network topology is an interconnection

of Ethernet Layer 2 and Layer 3 switches that can accommodate an unlimited number of point-to-point nodes.

7.6 The Data Link Layer

IEEE's 802.3 specification is also the standard used for transmitting packets of data from device to device on the EtherNet/IP Data Link Layer. Ethernet employs a CSMA/CD media access mechanism that determines how networked devices share a common bus (i.e., cable), and how they detect and respond to data collisions.

Originally, Ethernet worked in a half-duplex mode of operation, meaning that a node could send or receive data, but it could not do both at the same time. This caused data traffic jams, which are unacceptable in time-critical control applications. With full-duplex Ethernet, networked devices can both send and receive packets of Ethernet data at the same time. This is one of several advances in Ethernet technology that has increased its level of determinism to the point where Ethernet can be used in an ever-increasing number of manufacturing applications.

The Media Access Control (MAC) protocol of the IEEE 802.3 specification is what actually allows devices to "talk" on the Ethernet network. Each device has a unique MAC address comprised of a 6-byte number that is regulated by IEEE and the product manufacturer to maintain uniqueness (refer also to the "5.4 MAC address" section on page 59). This MAC address is used in the source address (SA) field of the frame to indicate what node sent the frame, and it is used in the destination address (DA) field to indicate the destination of the frame. Setting the first bit to a "1" in the DA field indicates a packet of data for multiple destinations, and enables an Ethernet node to transmit a single data packet to broadcast to the various destinations.

A single frame of industrial EtherNet/IP can contain up to 1,500 bytes of data, depending on the application requirements. The combination of real-time control with high-data capacity makes industrial Ethernet increasingly attractive, as more intelligence is embedded into smaller and less-expensive devices.

7.7 Ethernet data packets

Ethernet data packets are sent in the format shown in Figure 34.

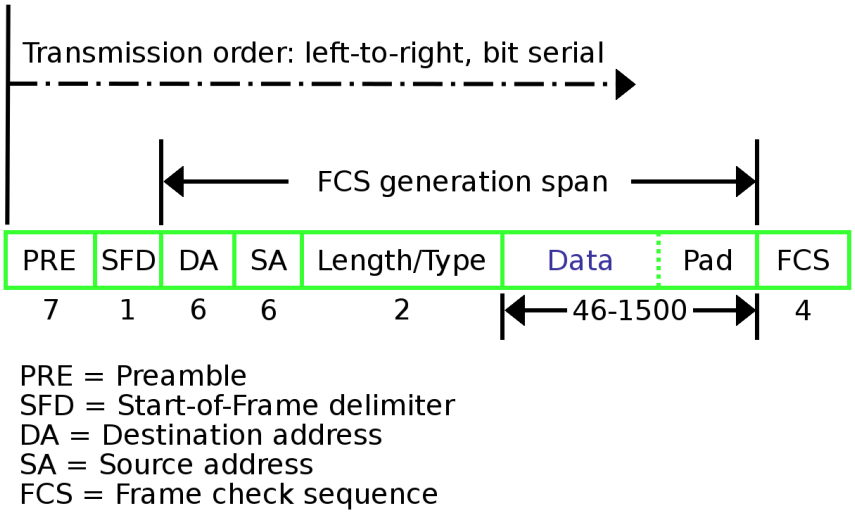


Figure 34 - Typical layout of an Ethernet Data Packet

This data format is used to implement the **Media Access Control (MAC)** protocol that allows a device to "talk" on the Ethernet network. Each MAC device has a unique **Source Address (SA)** comprised of a 6-byte number (48 bits or 12 hexadecimal digits) that was assigned to it at the time of manufacture. The **Destination Address (DA)** is the target MAC address for which the packet of data is intended. Setting the first bit to a "1" in the DA field, indicates a packet of data for multiple destinations. This enables an Ethernet device to transmit one packet that can be received by multiple other devices.
There are a number of different types of Ethernet packets that can be sent and received on an Ethernet network. Some of these protocols are Novell's IPX/SPX, DECNET, UDP, TCP/IP, FTP, TELNET, and so on. All of these unique protocols use the MAC to do the physical sending and receiving of data packets. However, by defining how the "DATA" portion of the data packet is organized, different protocols and functions are created.

7.8 The Network and Transport Layers

At the Network and Transport Layers, EtherNet/IP utilizes the Internet standard known as the Transmission Control Protocol/Internet Protocol (TCP/IP) Suite to send messages between one or more devices. TCP/IP provides the necessary communication protocol features needed to implement fully functional networks (i.e., an addressing scheme and mechanisms for establishing a connection with a device and exchanging data) that the IEEE specification in and of itself lacks.
Also, at these layers, the standard CIP messages used by all CIP Networks are encapsulated. TCP/IP encapsulation allows a node on the network to embed a

message as the data portion in an Ethernet message. The node then sends the message—TCP/IP protocol with the message inside—to an Ethernet communication chip (the Data Link Layer). By using TCP/IP, EtherNet/IP is able to send **explicit messages**, which are used to perform Client-Server type transactions between nodes.

The TCP/IP Suite consists of the following:

- The TCP portion of the TCP/IP protocol is a connection-oriented, unicast transport mechanism that provides data flow control, fragmentation reassembly and message acknowledgments. Nodes must interpret each message, execute the requested task and generate responses. Since TCP is ideal for the reliable transmission of large quantities of data, EtherNet/IP uses TCP/IP to encapsulate CIP explicit messages, which are generally used to transmit configuration, diagnostic and event data.
- The IP portion of the TCP/IP protocol is the mechanism that enables packet routing through multiple possible paths. The ability to send messages to their destinations even when the primary path is disrupted is the basis of the Internet. This same type of routing is used in industrial networks to maintain proper separation of control elements and other factory infrastructure through the use of managed switches and Layer 3 routers. All devices and infrastructure components with added diagnostic capabilities (managed switches and routers) on an industrial Ethernet-based system must be assigned an IP address. This is most commonly identified by the four-byte address listed in the "network properties" on personal computers that use TCP/IP as their Ethernet network connection (e.g., 192.168.1.10). IP addresses must be unique on a given network (see also the "4.5 EtherNet/IP Node ID" section on page 44).

For real-time messaging, EtherNet/IP also employs UDP over IP, which allows messages to be multicast to a group of destination addresses. This is how CIP I/O data transfers (**implicit messaging**, see the "7.9.4 Types of EtherNet/IP communications" section later) are sent on EtherNet/IP. With implicit messaging, the data field contains no protocol information, only real-time I/O data. Since the meaning of the data is pre-defined at the time the connection is established, processing time is minimized during runtime. UDP is connectionless and makes no guarantee that data will get from one device to another; however, UDP messages are smaller and can be processed more quickly than explicit messages. As a result, EtherNet/IP uses UDP/IP to transport I/O messages that typically contain time-critical control data. The CIP Connection mechanism provides timeout mechanisms that can detect data delivery problems, a capability that is essential for reliable control system performance.

7.9 Upper Layers: Objects, Services, and Application Data

7.9.1 EtherNet/IP services

The CIP application layer defines a set of **application objects** and **device profiles** that define common interfaces and behaviors. In addition, CIP communication services enable end-to-end communication between devices on the different CIP networks. EtherNet/IP maps the CIP communication services to Ethernet and TCP/IP, enabling multi-vendor interoperability between devices on Ethernet as well as with the other CIP networks.

7.9.2 Simplified EtherNet/IP Object Model Overview

Within the CIP application layer, devices are represented using an object model (Figure 35). **Application objects** define how device data is represented and accessed in a common way. **Network-specific objects** define how parameters such as IP addresses are configured and EtherNet/IP specific functions.
Communication objects and services provide the means to establish communication associations and access device data and services over the network.

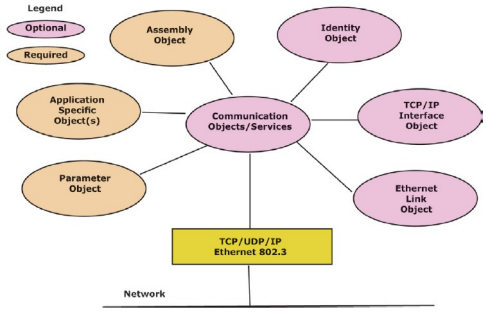


Figure 35 - EtherNet/IP Object Model

7.9.3 Exposing Application Data with CIP

Objects within a device are groups of related data and behavior associated with this data. CIP requires certain objects to describe a device, how it functions, communicates and its unique identity. The **Identity Object** (see on page 103), for example, contains identity data values called **attributes** that are used to store the identity information of a device. Attributes for the Identity Object include the Vendor ID, Device Type, device serial number and other identity data. CIP does not specify how object data is implemented, rather, which data values or attributes must be supported and made available to other CIP devices.

There are three types of objects defined by CIP:

- **Required Objects** must be included in all CIP devices. These objects include the Identity Object (page 103), the Message Router Object (page

107) and network-specific objects such as TCP/IP Interface Object (page 136) and Ethernet Link Object (page 141) for EtherNet/IP protocol.

- **Application Objects** describe how data is encapsulated by a device. These objects are specific to the Device Type and function. For example, an input device would have an input object with attributes that describe the value and fault status of a particular input point. See Application Object (page 113).
- **Vendor-specific Objects** describe services that are specific to a particular vendor; they are optional and not described in a predefined Device Profile.

Addressing data within a CIP device utilizes the same object-oriented view. A **class** (of objects) is a set of objects that represent the same type of system component (Figure 36). Sometimes it is necessary to have more than one 'copy' of an object, called **object instances**, within a device. This set of objects is called an **object class**. Each instance of the object class will have the same set of attributes, but will have a unique set of values. An object instance or an object class has **attributes**, providing services and implementing behavior.

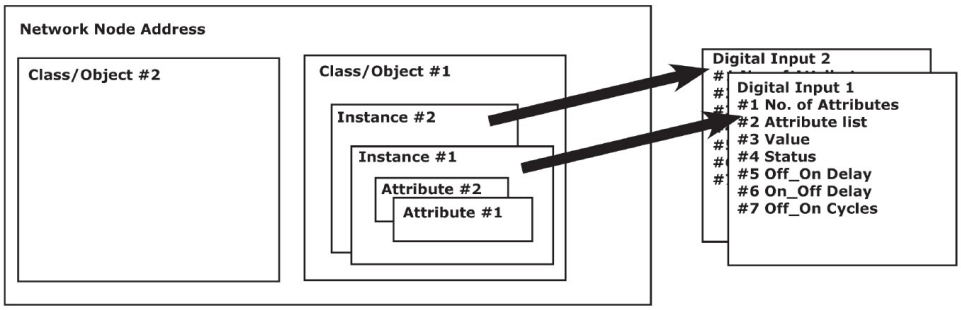


Figure 36 - CIP Object-oriented view of application data

The following Object Modeling related terms are used when describing CIP services and protocol.

- **Object** – An abstract representation of a particular component within a product.
- **Class** – A set of objects that all represent the same kind of system component. A class is a generalization of an object. All objects in a class are identical in form and behavior, but may contain different attribute values.
- **Instance** – A specific and real (physical) occurrence of an object. For example: New Zealand is an instance of the object class Country. The terms Object, Instance, and Object Instance all refer to a specific Instance.
- **Attribute** – A description of an externally visible characteristic or feature of an object. Typically, attributes provide status information or

govern the operation of an Object. For example: the ASCII name of an object; and the repetition rate of a cyclic object.

- **Instantiate** - To create an instance of an object with all instance attributes initialized to zero unless default values are specified in the object definition.
- **Behavior** – A specification of how an object acts. Actions result from different events the object detects, such as receiving service requests, detecting internal faults or elapsing timers.
- **Service** – A function supported by an object and/or object class. CIP defines a set of common services and provides for the definition of Object Class and/or Vendor Specific services.
- **Communication Objects** - A reference to the Object Classes that manage and provide the runtime exchange of implicit (I/O) and explicit messages.
- **Application Objects** - A reference to multiple Object Classes that implement product-specific features.

Accessing data within a device using a non-time critical message (an explicit message – see "7.9.4 Types of EtherNet/IP communications" section on page 98) typically contains the following address information:

- Device network address
- Class ID
- Instance ID
- Attribute ID
- Service code (describing the action/service required)

The Class/Instance/Attribute ID form of addressing is also used in Electronic Data Sheets (EDS) to identify configurable parameters within a device.
In addition to specifying how device data is represented, CIP also specifies methods by which I/O data can be accessed, using triggers, such as cyclic or change-of-state. Vendors can also describe how data from different objects can be combined in an I/O or configuration message using the Assembly Object, refer to the "7.12.3 Class 04h: Assembly Object" section on page 108.

7.9.4 Types of EtherNet/IP communications

EtherNet/IP defines two primary types of communications: **explicit** and **implicit**.

CIP Message Type	CIP Communication Relationship	Transport Protocol	Communication Type	Typical Use	Example
Explicit	Connected or Unconnected	TCP/IP	Request/reply transactions	Non time-critical information data	Read/Write configuration parameters
Implicit	Connected	UDP/IP	I/O data transfers	Real-time I/O data	Real-time control data from a remote I/O device

- **Explicit Messaging** in general has a request/reply (or Client/Server) nature. This type of communication is **used for non-real-time data**, normally for information. Explicit messages include a description of their meaning (expressed explicitly), so the transmission is less efficient, but very flexible. It may be used by an HMI to collect data, or by a device programming tool. In CIP terms, with Explicit Messaging you request a service of a particular object, e.g., a read or a write service. For EtherNet/IP, Explicit Messaging uses TCP. Explicit Messaging can be done with or without prior establishment of a CIP connection.
- **Implicit Messaging** is also often referred to as **"I/O" and is time-critical** in nature. Typically this type of communication is used for **real-time data exchange**, where speed and low latency are important. Implicit messages include very little information about their meaning, so the transmission is more efficient, but less flexible than explicit. The interpretation of the transmitted data is fast. With Implicit Messaging you establish an association (a "CIP connection") between two devices and produce the Implicit Messages according to a predetermined trigger mechanism, typically at a specified packet rate. The devices both know and agree on the data formats they will use (i.e., the format is "implied"). For EtherNet/IP, Implicit Messaging uses UDP and can be multicast or unicast.

Connections are established using the ForwardOpen Request service of the Connection Manager Object, see the "7.12.4 Class 06h: Connection Manager Object" section on page 112. The ForwardOpen Request contains all of the connection parameters, including transport class, production trigger, timing information, electronic key and connection IDs. Connection clean-up takes place when a ForwardClose Request service request is issued or when either connection end point times out.
Implicit messaging can make use of the CIP Producer/Consumer communication model. With **Producer/Consumer**, the producing device transmits data once, regardless of the number of consumers. All interested consuming devices receive

the same data. For EtherNet/IP the produced data is identified by the IP multicast address and the CIP Connection ID. The Producer/Consumer model leads to greater network efficiency when multiple consumers need to receive the same data from a producer. For I/O connections, once the connection is established there is no request/response, the data with the ConnectionID is just produced and consumed at intervals determined by the Production Trigger which was specified at connection establishment. Triggers can be Cyclic (most common), Change of State (CoS) or Application.

7.9.5 Types of EtherNet/IP devices

Several device classifications, based on their general behavior and types of EtherNet/IP communications they support, have been defined:

- **Explicit Message Server**: An explicit message server responds to request/response oriented communications initiated by explicit message clients. An example of an explicit message server is a bar code reader.
- **Explicit Message Client**: An explicit message client initiates request/response oriented communications with other devices. Message rates and latency requirements are typically not too demanding. Examples of explicit message clients are HMI devices, programming tools, or PC or Linux based applications that gather data from control devices.
- **I/O Adapter**: An I/O adapter receives implicit communication connection requests from an I/O scanner then produces its I/O data at the requested rate. An I/O adapter is also an explicit message server. An I/O adapter can be a simple digital input device, or something more complex such as a modular pneumatic valve system.
- **I/O Scanner**: An I/O scanner initiates implicit communications with I/O adapter devices. A scanner is typically the most complex type of EtherNet/IP device, as it must deal with issues such as configuration of which connections to make, and how to configure the adapter device. Scanners also typically support initiating explicit messages. A programmable controller is an example of an I/O scanner.

7.10 ODVA

ODVA is an international association comprising members from the world's leading automation companies. Collectively, ODVA and its members support network technologies based on the Common Industrial Protocol (CIP™). These currently include DeviceNet™, EtherNet/IP™, CompoNet™, and ControlNet™, along with the major extensions to CIP – CIP Safety™, CIP Sync™ and CIP Motion™. ODVA manages the development of these open technologies, and assists manufacturers and users of CIP Networks through its activities in standards development, certification, vendor education and industry awareness.

For further information on ODVA, see the ODVA website: www.odva.org.

7.11 EDS file

The functionality of an EtherNet/IP device is always described in an EDS file (Electronic Data Sheet file). The Electronic Data Sheet file provides information about the device basic communication and functional properties. It must be installed in the Controller.

EtherNet/IP rotary actuators from Lika Electronic are supplied with their own EDS file.
Specific EDS files are provided to each RD actuator series and specific models, please refer to the order code.
RD1xA actuator's EDS files are:

- **RD1xA_EP_HxSx.eds**: it is intended for installation of **both RD1A and RD12A series actuators** ("RD1xA" is the actuator series; "EP" is the Lika code that identifies the EtherNet/IP protocol; "Hx" is the hardware version of the actuator; "Sx" is the software version of the actuator); see the order code, for instance: RD1A-P8-T48-EP-... .

The version of the EDS file is reported under the "Version" item inside the file.
EDS files can be paired with the **RD1xA_48x48.ico** picture file available inside the file folder (the picture is also integrated into the EDS file).

Follow the path **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS** to download the EDS files from Lika's corporate web site.

7.12 Object Library

As previously stated, object modeling is used to represent the network visible behavior of devices (i.e. the actuator). Devices are modeled as a collection of objects. Each class of objects is a collection of related services, attributes and behaviors. Services are the procedures that an object performs. Attributes are characteristics of objects represented by values, which can vary. An object's behavior is an indication of how the object responds to particular events. For more information refer to the "7.9.3 Exposing Application Data with CIP" section on page 95.
This section contains the description of the objects specific to Lika actuators, including services and attributes.

In the following pages the Class Attributes are listed and described as follows:

Class-Attribute ID Attribute name
[Data type, Access Rule, NV]

While the Instance Attributes are listed and described as follows:
Class-Instance-Attribute ID Attribute name
[Data type, Access Rule, NV]

- Class, instance and attribute are expressed in hexadecimal notation.
- Data types are as shown in the following table:

Data type	Code	Name	Range
BOOL	C1h	Boolean	0 (FALSE) and 1 (TRUE)
SINT	C2h	Signed 8-bit integer	-128 to 127
INT	C3h	Signed 16-bit integer	-32,768 to 32,767
DINT	C4h	Signed 32-bit integer	-2^{31} to $2^{31}-1$
LINT	C5h	Signed 64-bit integer	-2^{63} to $2^{63}-1$
USINT	C6h	Unsigned 8-bit integer	0 to 255
UINT	C7h	Unsigned 16-bit integer	0 to 65,535
UDINT	C8h	Unsigned 32-bit integer	0 to $2^{31}-1$
ULINT	C9h	Unsigned 64-bit integer	0 to $2^{63}-1$
STRING	D0h	Character string	1 byte per character
BYTE	D1h	Bit string – 8 bits	$2\#b_{N-1}b_{N-2}...b_2b_1b_0$, where N is the number of bits in the bit string, b_{N-1} is the "most significant bit", and b_0 is the "least significant bit"
WORD	D2h	Bit string - 16 bits	
SHORT_STRING	DAh	Character string	1 byte per character, 1 byte length indicator
ENGUNIT	DDh	Engineering unit	0 to 65,535

- Access rule can be:

Get (Gettable): the same as "ro" = read only access. The attribute can be accessed by at least one of the get services.

Set (Settable): the same as "rw" = read and write access. The attribute shall be accessed by at least one of the set services. Settable attributes, unless otherwise specified by the object definition, shall also be accessed by get services.

- NV

It indicates whether an attribute value is maintained through power cycles. An entry of 'NV' indicates value shall be saved, 'V' means not saved.

- Default, Min. and Max. values

Default, Min. and Max. values are expressed in hexadecimal notation, unless otherwise indicated.

NOTE

All data bytes are sent from least significant byte (LSB) to most significant byte (MSB).

7.12.1 Class 01h: Identity Object

Class Code	Object Class	Access	Nr. of Instances
01h	Identity Object	Get	1

The Identity Object provides identification of and general information about the actuator (e.g. Vendor ID, device type, product code, etc.). Instance 1, which is the only mandatory instance, describes the whole product. It is used by applications to determine what nodes are on the network and to match an EDS file with a product on the network.

7.12.1.1 Supported Class Services

The supported **Class Services** of the Identity Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.1.2 Class Attributes

01-01 Revision
[UINT, Get, NV]
Object revision. The current value assigned to this attribute is 0001h.
Default = 0001h

01-02 Max Instance
[UINT, Get, NV]
The largest instance number of a created object in this class.
Default = 0001h

01-03 Number of Instances
[UINT, Get, NV]
The number of object instances in this class.
Default = 0001h

7.12.1.3 Supported Instance Services

The supported **Instance Services** of the Identity Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
0Eh = Get_Attribute_Single: used to read the value of an attribute.
05h = Reset: the following types of reset are defined:

0 = Power Cycle Reset	It emulates a power cycling of the actuator.
1 = Return to Factory Defaults Reset	It returns to the factory default configuration of the actuator parameters and communication link parameters and emulates a power cycling of the actuator.

NOTE
After executing a Return to Factory Defaults reset (type 1), if the DIP A rotary switches are set to 00h, the actuator restarts using the IP address saved internally. If the DIP A rotary switches are set to any value between 01h and FEh, then the actuator restarts using the address 192.168.1."rotary switch setting". For more information refer to page 46.

7.12.1.4 Instance Attributes

01-01-01 Vendor ID
[UINT, Get, NV]
Identification of the vendor by its own number. Lika Vendor ID is 0299h = 665. Vendor IDs are managed by ODVA.
Default = 0299h = Lika Electronic Srl

01-01-02 Device type
[UINT, Get, NV]
The Device Type value is used to identify the device profile that a particular product is using. Device profiles are managed by ODVA and define minimum requirements a device must implement, as well as common options.
Default = 002Bh: Generic Device Profile.

01-01-03 Product code
[UINT, Get, NV]
Product Code identifies a particular product within the actuator device type.
The available product codes are:

- 3000h = RD1xA rotary actuator

01-01-04 Revision
[USINT, Get, NV]
The Revision attribute, which consists of Major and Minor Revisions, identifies the Revision of the item the Identity Object is representing. It is displayed as majorXX.minorYY, so representing the hardware (XX) and software (YY) revisions.

LSByte XX	MSByte YY
Major revision	Minor revision

Default = device dependent

01-01-05 Status

[WORD, Get, V]

This attribute represents the current status of the device. Its value changes as the state of the device changes. The Status attribute is a WORD, with the following bit definitions:

Bit(s)	Called	Definition
0	Owned	TRUE indicates the device (or an object within the device) has an owner. Within the Master/Slave paradigm the setting of this bit means that the Predefined Master/Slave Connection Set has been allocated to a Master. Outside the Master/Slave paradigm the meaning of this bit is TBD. 0 = no connection to the Master 1 = connection to the Master established
1	Reserved	Reserved, shall be 0
2	Configured	TRUE indicates the application of the device has been configured to do something different than the "out-of-box" default. This shall not include configuration of the communications. 0 = actuator is set to default parameters 1 = actuator is not set to default parameters
3	Reserved	Reserved, shall be 0
4-7	Extended device status	Bits are defined as follows: 0000 Unknown 0001 Reserved 0010 At least one faulted I/O connection 0011 No I/O connection established 0100 Non-Volatile Configuration bad (EEPROM) 0101 Major Fault – either bit 10 or bit 11 is TRUE (1) 0110 At least one I/O connection in run mode 0111 At least one I/O connection established, all in idle mode 1000 ... 1111 Reserved
8	Minor recoverable fault	TRUE indicates that the device detected a problem with itself, which is thought to be recoverable. The problem does not cause the device to go into one of the faulted states. Not implemented. For Alarms list refer to page 131
9	Minor unrecoverable fault	TRUE indicates that the device detected a problem with itself, which is thought to be unrecoverable. The problem does not cause the device to go into one of the faulted states.

		Not implemented. For Alarms list refer to page 131
10	Major recoverable fault	TRUE indicates that the device detected a problem with itself, which caused the device to go into the "Major Recoverable Fault" state. Not implemented. For Alarms list refer to page 131
11	Major unrecoverable fault	TRUE indicates that the device detected a problem with itself, which caused the device to go into the "Major Unrecoverable Fault" state. Not implemented. For Alarms list refer to page 131
12 ...15	Reserved	Reserved, shall be 0

For any further information on status instance attribute refer to the publication "The CIP Networks Library. Volume I. Common Industrial Protocol (CIP™)".

01-01-06 Serial number

[UDINT, Get, NV]
This attribute is a number used in conjunction with the Vendor ID to form a unique identifier for each device on any CIP network.
The Serial Number is shown in the following format: YYwwnnnnn.
YY = Year
ww = week
nnnnn = unique number in ascending order assigned by Lika Electronic
Default = device dependent

EXAMPLE
182100123 has to be intended as follows:
18 = Year of production = 2018
21 = Week of production = week 21
00123 = unique number in ascending order assigned by Lika Electronic

01-01-07 Product name

[SHORT_STRING, Get, NV]
This text string represents a short description of the product represented by the Product Code in attribute 01-01-03 Product code.
Default = RD1xA Rotary Actuator

7.12.2 Class 02h: Message Router Object

Class Code	Object Class	Access	Nr. of Instances
02h	Message Router Object	Get	1

This object provides a messaging connection point through which a Client may address a service to any object class or instance residing in the actuator.

In Lika actuators it is used internally to direct object requests.

7.12.3 Class 04h: Assembly Object

Class Code	Object Class	Access	Nr. of Instances
04h	Assembly Object	Get	4

The Assembly Object binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection. Assembly objects can be used to bind input data or output data. The terms "input" and "output" are defined from the network's point of view. An input will produce data on the network and an output will consume data from the network.
Assembly objects instances are static: assemblies with member lists defined by the open device profile or vendor specific device profile. The Instance number, number of members, and member list are fixed.

7.12.3.1 Supported Class Services
The supported **Class Services** of the Assembly Object are:
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.3.2 Class Attributes
04-01 Revision
[UINT, Get, NV]
Object revision. The current value assigned to this attribute is 0002h.
Default = 0002h

04-02 Max Instance
[UINT, Get, NV]
The largest instance number of a created object in this class.
Default = 0096h

7.12.3.3 Supported Instance Services
The supported **Instance Services** of the Assembly Object are:
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.3.4 Supported connection types
Lika RD1xA EtherNet/IP actuators support "Exclusive Owner" connection.

Exclusive Owner connection
An Exclusive Owner connection type is one that may provide application data in both the T → O direction and the O → T direction. If a connection has an

application type of Exclusive Owner, it shall not be dependent on any other connection for its existence.
When an Exclusive Owner connection timeout occurs in a target device, the target device shall stop sending the associated T → O data.
Connection point O → T Assembly Object, instance 96h (Consuming Instance)
Connection point T → O Assembly Object, instance 01h (Producing Instance)
T is the Target, i.e. the RD1xA actuator
O is the Origin, i.e. the Master

Refer also to the "7.12.4 Class 06h: Connection Manager Object" section on page 112.

7.12.3.5 Instance Attributes

The following table identifies the I/O Assembly instances, which are supported by the RD1xA actuator device.

Producing Instance (01h)

Instance ID	Attribute	Access	Description	Bits	Bytes
01h	03h	Get	64-01-03 Status Word 64-01-04 Real Position	16 32	6

Consuming Instance (96h)

Instance ID	Attribute	Access	Description	Bits	Bytes
96h	03h	Set	64-01-01 Control Word 64-01-02 Target Position	16 32	6

Configuration Assembly (6Ah)

Instance ID	Attribute	Access	Description	Bits	Bytes
6Ah	03h	Get/Set	**Configuration Assembly**	272	34

7.12.3.6 Configuration Assembly

Instance	Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Attribute ID
6Ah	0	64-01-05 Distance per Revolution							(low byte)	05h
	1								(high byte)	
	2	64-01-06 Position Tolerance							(low byte)	06h
	3								(high byte)	
	4	64-01-07 Settling time							(low byte)	07h
	5								(high byte)	
	6	64-01-08 Max following error							(low byte)	08h
	7									
	8									
	9								(high byte)	
	10	64-01-09 Proportional gain							(low byte)	09h
	11								(high byte)	
	12	64-01-0A Integral gain							(low byte)	0Ah
	13								(high byte)	
	14	64-01-0B Acceleration							(low byte)	0Bh
	15								(high byte)	
	16	64-01-0C Deceleration							(low byte)	0Ch
	17								(high byte)	
	18	64-01-0D Max delta pos							(low byte)	0Dh
	19									
	20									
	21								(high byte)	
	22	64-01-0E Max delta neg							(low byte)	0Eh
	23									
	24									
	25								(high byte)	
	26	64-01-0F Jog speed							(low byte)	0Fh
	27								(high byte)	
	28	64-01-10 Work speed							(low byte)	10h
	29								(high byte)	
	30								(low byte)	11h

	31	64-01-11 Count direction	(high byte)	
	32	64-01-13 Step jog	(low byte)	13h
	33		(high byte)	

Mentioned attributes are fully described in the "7.12.5 Class 64h: Application Object" section on page 113.

7.12.4 Class 06h: Connection Manager Object

Class Code	Object Class	Access	Nr. of Instances
06h	Connection Manager Object	Get	1

The Connection Manager Class allocates and manages the internal resources associated to both "I/O Messages" and "Explicit Messaging Connections".

For complete information refer to "THE CIP NETWORKS LIBRARY, Volume 1, Common Industrial Protocol (CIP™), Chapter 3: Communication Object Classes".

7.12.5 Class 64h: Application Object

Class Code	Object Class	Access	Nr. of Instances
64h	Application Object	Set/Get	1

This class is meant to describe the attributes used by the device, they are customer-specific.

7.12.5.1 Supported Class Services

The supported **Class Services** of the Application Object are:

05h = Reset: resets all parameter values to the factory default values and saves them on flash memory. The following types of reset are defined:

0 = Power Cycle Reset	It emulates a power cycling of the actuator.
1 = Return to Factory Defaults Reset	It returns to the factory default configuration of the actuator parameters and communication link parameters and emulates a power cycling of the actuator.

NOTE

After executing a Return to Factory Defaults reset (type 1), if the DIP A rotary switches are all set to 0, the rotary actuator restarts using the IP address saved internally. If the DIP A rotary switches are set to any value between 1 and 254, then the rotary actuator restarts using the address 192.168.1."rotary switch setting". For more information refer to page 46.

0Eh = Get_Attribute_Single: used to read connection class attribute value.

15h = Restore: restores all parameter values from flash memory and saves them.

16h = Save: saves all parameters to non-volatile memory.

7.12.5.2 Class Attributes

64-01 Revision

[UINT, Get, NV]

Object revision. The current value assigned to this attribute is 0002h.

Default = 0002h

7.12.5.3 Supported Instance Services

The supported **Instance Services** of the Application Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
10h = Set_Attribute_Single: used to write connection class attribute value.

7.12.5.4 Instance Attributes

64-01-01 Control Word

[UINT, Set, V]
This attribute contains the commands to be sent in real time to the actuator in order to manage it. It is updated every 1 msec.
Default = 0 (min. value 0, max. value 65535)

Byte 0

Jog +

bit 0

If the bit 4 **Incremental jog** = 0, as long as **Jog +** = 1, the actuator 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 **64-01-13 Step jog** attribute 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 **64-01-0F Jog speed**, **64-01-0B Acceleration** and **64-01-0C Deceleration** attributes respectively. For a detailed description of the jog control see on page 83.

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the actuator 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 actuator 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 **64-01-13 Step jog** attribute 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 **64-01-0F Jog speed**, **64-01-0B Acceleration** and **64-01-0C Deceleration** attributes

respectively. For a detailed description of the jog control see on page 83.

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the actuator 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 actuator is allowed to execute the movements as commanded. If, while the unit is running, this bit switches to "=0" then the actuator must stop executing the deceleration procedure set in **64-01-0C Deceleration**. For an immediate halt in the movement, use the bit 7 **Emergency**.

Alarm reset

bit 3

This command is used to reset an alarm condition of the actuator 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 **64-01-20 Parameter Error List**), the normal work status can be restored only after having set proper values. The **Flash memory error** alarm cannot be reset.

Incremental jog

bit 4

If set to "=0", the activation of the bits **Jog +** and **Jog -** causes the actuator 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 **64-01-13 Step jog** attribute to be executed at rising edge; then the actuator stops and waits for another command.

bit 5

Not used.

Start

bit 6

When it is set to "=1" the device moves in order to reach the set target position (see **64-01-02 Target Position** on page 118). For a complete description of the position control see on page 84.

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the actuator 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

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 237.
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 **64-01-12 Preset** attribute. 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"). Then the bit must be switched back to logic level low ("0") to finalize the command. When the command is sent, the current actuator position is saved temporarily in the **64-01-18 Position Offset** attribute. For any further information on the preset function and the meaning and use of the related attributes and commands **64-01-12 Preset**, **64-01-18 Position Offset** and **Setting the preset** refer to page 127.

NOTE

Please note that as soon as the preset value is entered next to the **64-01-12 Preset** attribute, it is also automatically activated, so you do not need to use this command. Use the **Setting the preset** command to activate a preset value that has been already set next to the **64-01-12 Preset** attribute and you want to set for a different shaft position.

WARNING

To save permanently the current position in the **64-01-18 Position Offset** attribute, please execute the **Save parameters** command. Should the power supply be turned off without saving data, the **64-01-18 Position Offset** that has not been saved will be lost!

Release axis torque

bit 12

This function is available only in the RD1A version (model without brake); in the RD12A version (model fitted with brake) the bit 12 is not used. When the axis has reached the commanded position, it maintains the torque.
If set to "=1", when the axis is in position, the PWM is active (the torque is applied to the axis when the position is reached).
If set to "=0", when the axis is in position, the PWM is kept 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 85.

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 operating; setting this bit to "=0" causes the brake to be enabled and managed automatically by the system.

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

bit 15 Not used.

64-01-02 Target Position

[DINT, Set, V]

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

As soon as the axis is within the tolerance window limits set next to the 64-01-06 Position Tolerance attribute, the bit 8 **Target position reached** in the 64-01-03 Status Word goes high ("=1"). When the position is within the tolerance window limits set next to the 64-01-06 Position Tolerance attribute, after the delay set next to the 64-01-07 Settling time attribute, the bit 0 **Axis in position** in the 64-01-03 Status Word goes high ("=1"). It is updated every 1 msec.

For more information refer also to the "Positioning: position and speed control" section on page 84.

Default = 0 (min. = -2147483647, max. = +2147483647 = within 64-01-1E Positive Limit Switch [pulse] / 64-01-1F Negative Limit Switch [pulse])

NOTE

Position override function

It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the 64-01-02 Target Position attribute. See also on page 84.

NOTE

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the actuator 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 network while it is moving (for instance because of a broken cable or a faulty wiring), the device stops moving immediately and activates the **Network timeout** alarm bit.

64-01-03 Status Word

[UINT, Get, V]
This attribute provides information about the current state of the device. It is updated every 1 msec.

Byte 0

Axis in position

bit 0 — The value is "=1" when the device reaches and keeps the commanded position (**64-01-02 Target Position**) for the time set next to the **64-01-07 Settling time** attribute. It is kept active until the position error is lower than **64-01-06 Position Tolerance**. For further information please refer to the "Positioning: position and speed control" section on page 84.

bit 1 — Not used.

Axis enabled

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

SW limit switch +

bit 3 — The value is "=1" should it happen that the device reaches the maximum positive limit (positive limit switch). For more information see the **64-01-0D Max delta pos** attribute on page 124.

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 **64-01-0E Max delta neg** attribute on page 125.

Alarm

bit 5 — The value is "=1" when an alarm occurs, see details in the **64-01-21 Alarms List** attribute.

Axis running

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

Executing a command

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

Byte 1

Target position reached

bit 8 — The value is "=1" when the device reaches the target position set next to the **64-01-02 Target Position** attribute (it is within the limits set next to **64-01-06 Position Tolerance**). The bit is kept active until a new **64-01-02 Target Position** value or the **Alarm reset** command are sent. For more information refer also to the "Positioning: position and speed control" section on page 84.

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 "4.5.2 Screw plug for internal access (Figure 4 and Figure 6)" section on page 45 and the "4.8.1 JOG + and JOG – buttons (Figure 9)" section on page 51.

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 "4.5.2 Screw plug for internal access (Figure 4 and Figure 6)" section on page 45 and the "4.8.1 JOG + and JOG – buttons (Figure 9)" section on page 51.

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 "4.5.2 Screw plug for internal access (Figure 4 and Figure 6)" section on page 45 and the "4.8.2 PRESET button (Figure 9)" section on page 52.

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 85.

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 85.

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 85.

IN 3 = 0 input 3 low (not active)
IN 3 = 1 input 3 high (active)

64-01-04 Real Position

[DINT, Get, V]

This attribute represents the current position of the device detected by the built-in encoder and expressed in pulses. It is updated every 1 msec.

64-01-05 Distance per Revolution

[UINT, Set, NV]

This attribute sets the number of pulses per each complete revolution of the shaft. It is useful to relate the revolution of the shaft and a linear measurement. For example: the unit is joined to a warm screw having 5 mm (0.197") pitch; by setting **64-01-05 Distance per Revolution** = 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 attribute then you must set new values also next to the **64-01-12 Preset** attribute. For a detailed explanation see on page 86 and the relevant attributes.

Please note that the attributes listed hereafter are closely related to the **64-01-05 Distance per Revolution** attribute; hence when you change the value in **64-01-05 Distance per Revolution** also the value in each of them necessarily changes. They are: **64-01-06 Position Tolerance**, **64-01-08 Max following error**, **64-01-0D Max delta pos**, **64-01-0E Max delta neg**, **64-01-02 Target Position**, **64-01-04 Real Position** and **64-01-1D Following error [pulse]**.

NOTE

If **64-01-05 Distance per Revolution** is not a power of 2 (2, 4, ..., 512, 1024), at position control a positioning error could occur having a value equal to one count.

64-01-06 Position Tolerance

[UINT, Set, NV]

This attribute defines the tolerance window limits for the **64-01-02 Target Position** value. As soon as the axis is within the tolerance window limits, the bit 8 **Target position reached** in the **64-01-03 Status Word** goes high ("=1"). When the axis is within the tolerance window limits for the time set in the **64-01-07 Settling time** attribute, the bit 0 **Axis in position** in the **64-01-03 Status Word** goes high ("=1"). The attribute is expressed in pulses. See also the "Positioning: position and speed control" section on page 84.

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

64-01-07 Settling time

[UINT, Set, NV]

It represents the time for which the axis has to be within the tolerance window limits set in the **64-01-06 Position Tolerance** attribute before the state is signalled through the **Axis in position** status bit of the **64-01-03 Status**

Word. The attribute is expressed in milliseconds (ms). See also the "Positioning: position and speed control" section on page 84.
Default = 0 (min. = 0, max. = 10000)

64-01-08 Max following error

[UDINT, Set, NV]
This attribute 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 attribute, the **Following error** alarm is triggered and the unit stops. The attribute is expressed in pulses.
Default = 1024 (min. = 0, max. = 65535)

64-01-09 Proportional gain

[UINT, Set, NV]
This attribute 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)

64-01-0A Integral gain

[UINT, Set, NV]
This attribute 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)

64-01-0B Acceleration

[UINT, Set, NV]
This attribute defines the acceleration value that has to be used by the device when reaching either the **64-01-0F Jog speed** or the **64-01-10 Work speed**. The attribute is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 83.
Default = 10 (min. = 1, max. = 500)

64-01-0C Deceleration

[UINT, Set, NV]
This attribute defines the deceleration value that has to be used by the device when stopping. The attribute is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 83.
Default = 10 (min. = 1, max. = 500)

64-01-0D Max delta pos

[UDINT, Set, NV]

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 **64-01-03 Status Word** (the bit is forced high). The attribute is expressed in pulses.

SW limit switch + = **64-01-12 Preset** + **64-01-0D Max delta pos**. The maximum positive limit can be read next to the **64-01-1E Positive Limit Switch [pulse]** attribute.

For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

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 **64-01-05 Distance per Revolution** = 1,024 and **64-01-12 Preset** = 0, the maximum acceptable value for **64-01-0D Max delta pos** 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 **64-01-05 Distance per Revolution** = 256 and **64-01-12 Preset** = 0, the maximum acceptable value for **64-01-0D Max delta pos** is:
(256 steps per revolution * 512 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 130 815
See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

WARNING

Every time **64-01-05 Distance per Revolution** and **64-01-12 Preset** attributes are changed, **64-01-0D Max delta pos** and **64-01-0E Max delta neg** values have to be checked carefully. Each time you change the value in **64-01-05 Distance per Revolution**, then you must update the value in **64-01-12 Preset** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **64-01-12 Preset** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **64-01-0D Max delta pos** and **64-01-0E Max delta neg** attributes. For a detailed explanation see on page 86.

64-01-0E Max delta neg

[UDINT, Set, NV]

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 **64-01-03 Status Word** (the bit is forced high). The attribute is expressed in pulses.

SW limit switch - = **64-01-12 Preset** - **64-01-0E Max delta neg**. The maximum negative limit can be read next to the **64-01-1F Negative Limit Switch [pulse]** attribute.

For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

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 **64-01-05 Distance per Revolution** = 1,024 and **64-01-12 Preset** = 0, the maximum acceptable value for **64-01-0E Max delta neg** is:
(1,024 steps per revolution * 512 revolutions) - 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 523 263

When **64-01-05 Distance per Revolution** = 256 and **64-01-12 Preset** = 0, the maximum acceptable value for **64-01-0E Max delta neg** is:
(256 steps per revolution * 512 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 130 815

See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

WARNING

Every time **64-01-05 Distance per Revolution** and **64-01-12 Preset** attributes are changed, **64-01-0D Max delta pos** and **64-01-0E Max delta neg** values have to be checked carefully. Each time you change the value in **64-01-05 Distance per Revolution**, then you must update the value in **64-01-12 Preset** in order to define the zero of the shaft as the system reference has now changed.

After having changed the parameter in **64-01-12 Preset** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **64-01-0D Max delta pos** and **64-01-0E Max delta neg** attributes. For a detailed explanation see on page 86.

64-01-0F Jog speed

[UINT, Set, NV]
This attribute contains the maximum speed the motor is allowed to reach when using the **Jog +** and **Jog -** functions (see the **64-01-01 Control Word** attribute). The attribute is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 83.
Default = 2000 (min. = 1, max. = 3000)

NOTE
Please note that this is the speed of the motor, not the speed of the output shaft after the reduction gears.
The speed at output will be as follows:
Motor speed = 2000 rpm
Speed at output: T12 = 166 rpm
T24 = 83 rpm
T48 = 41 rpm
T92 = 21 rpm

64-01-10 Work speed

[UINT, Set, NV]
This attribute 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 **64-01-01 Control Word** attribute- and are performed in order to reach the position set in **64-01-02 Target Position**). The attribute is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 84.
Default = 2000 (min. = 1, max. = 3000)

NOTE
Please note that this is the speed of the motor, not the speed of the output shaft after the reduction gears.
The speed at output will be as follows:
Motor speed = 2000 rpm
Speed at output: T12 = 166 rpm
T24 = 83 rpm
T48 = 41 rpm
T92 = 21 rpm

64-01-11 Count direction

[UINT, Set, NV]
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
1 = count up information with counter-clockwise rotation

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

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 **64-01-12 Preset** attribute and then check the values set next to the **64-01-0D Max delta pos** and **64-01-0E Max delta neg** attributes.

64-01-12 Preset

[DINT, Set, NV]
Use this attribute 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 saved and activated as soon as the value is set.
If you need to activate a value already set next to the **64-01-12 Preset** in a different physical position of the actuator shaft, you can use the bit 11 **Setting the preset** in the **64-01-01 Control Word** attribute, see on page 117.
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 117.

WARNING
A new value has to be set in the **64-01-12 Preset** attribute every time the **64-01-05 Distance per Revolution** value is changed. After having entered a new value in **64-01-12 Preset** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in the **64-01-0D Max delta pos** and **64-01-0E Max delta neg** items. For a detailed explanation see on page 86.

64-01-13 Step jog

[UINT, Set, NV]
If the incremental jog function is enabled (bit 4 **Incremental jog** in the **64-01-01 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).

64-01-14 Network controller Serial Number

[UDINT, Get, NV]
It shows the serial number of the network controller module.
Value = device dependent

64-01-15 Network controller Firmware Major

[USINT, Get, NV]
The firmware revision number of the network controller consists of a major revision number and a minor revision number. In this attribute the major revision number is shown.
Value = device dependent

64-01-16 Network controller Firmware Minor

[USINT, Get, NV]
The firmware revision number of the network controller consists of a major revision number and a minor revision number. In this attribute the minor revision number is shown.
Value = device dependent

64-01-17 Network controller Firmware Build

[USINT, Get, NV]
It shows the firmware build number of the network controller module.
Value = device dependent

64-01-18 Position Offset

[DINT, Get, NV]
This attribute defines the difference between the position value transmitted by the device and the real position: real position – preset. The value is expressed in pulses.

64-01-19 Real Speed [rpm]

[DINT, Get, V]
Speed of the device expressed in revolutions per minute [rpm], updated at every second. This is the speed of the motor, not the speed of the output shaft after the reduction gears. The speed at output will be as follows:
Motor speed = 2000 rpm
Speed at output: T12 = 166 rpm
T24 = 83 rpm
T48 = 41 rpm

T92 = 21 rpm

64-01-1A Electronics Temperature [°C]

[SINT, Get, V]

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

64-01-1B Motor Temperature [°C]

[SINT, Get, V]

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

64-01-1C Real Current [mA]

[DINT, Get, V]

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

64-01-1D Following error [pulse]

[DINT, Get, V]

This attribute 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 **64-01-08 Max following error** attribute, then the **Following error** alarm is triggered and the unit stops. The value is expressed in pulses.

64-01-1E Positive Limit Switch [pulse]

[DINT, Get, V]

This is the **SW limit switch +** value (maximum positive limit) calculated according to values set next to the **64-01-12 Preset** and **64-01-0D Max delta pos** attributes. When the maximum forward limit is reached, the condition is signalled through the **SW limit switch +** status bit 3 of the **64-01-03 Status Word**.

SW limit switch + = **64-01-12 Preset** + **64-01-0D Max delta pos**.

The value is expressed in pulses.

Refer also to the EXAMPLE 1 in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

Default = 523 263

64-01-1F Negative Limit Switch [pulse]

[DINT, Get, V]

This is the **SW limit switch -** value (maximum negative limit) calculated according to values set next to the 64-01-12 Preset and 64-01-0E Max delta neg attributes. When the maximum backward limit is reached, the condition is signalled through the **SW limit switch -** status bit 4 of the 64-01-03 Status Word.

SW limit switch - = 64-01-12 Preset – 64-01-0E Max delta neg.

The value is expressed in pulses.

Refer also to the EXAMPLE 1 in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

Default = - 523 263

64-01-20 Parameter Error List

[UDINT, Get, V]

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	64-01-05 Distance per Revolution
2	64-01-0B Acceleration
3	64-01-0C Deceleration
4	64-01-0D Max delta pos
5	64-01-0E Max delta neg
6	64-01-0F Jog speed
7	64-01-10 Work speed
8	64-01-11 Count direction
9	64-01-12 Preset
10	64-01-13 Step jog
11	64-01-09 Proportional gain
12	64-01-0A Integral gain
13	64-01-07 Settling time
14	64-01-08 Max following error
15	Not used

64-01-21 Alarms List

[UINT, Get, V]
This attribute is meant to show the alarms currently active in the device.

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 **64-01-20 Parameter Error List** attribute.

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 **64-01-08 Max following error** attribute; we suggest reducing the work speed.

Axis not synchronized
bit 4 — Internal error, it cannot be restored.

Target not valid
bit 5 — The set target position is over the maximum travel limits.

Emergency
bit 6 — Bit 7 **Emergency** in **64-01-01 Control Word** has been forced to low value (0); or alarms are active in the unit.

Overcurrent
bit 7 — Motor overcurrent.

Byte 1

Electronics Overtemperature
bit 8 — The temperature of the MOSFETs detected by an internal probe is exceeding the maximum ratings (see **64-01-1A Electronics Temperature [°C]** on page 129). 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 64-01-1B Motor Temperature [°C] on page 129). 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.

Network timeout

bit 11 — This is a safety feature that provides information in the event of an interruption of the network connection (for example because the Ethernet cable is disconnected or the link to the Ethernet port is missing). When no Ethernet communication is detected, the system forces an alarm condition (the **Network timeout** alarm bit is activated).

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 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 **64-01-01 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 **64-01-20 Parameter Error List**), the normal work status can be restored only after having set proper values. The **Flash memory error** alarm cannot be reset.

64-01-22 Node ID

[USINT, Get, NV]
This is meant to show the node address for MODBUS serial communication. The value is forced to "1". See also the "4.2.3 Inputs / output + MODBUS RS-232 service port" section on page 42.

64-01-23 Application controller Firmware Version

[UINT, Get, NV]
This attribute contains the firmware version of the Application DSC (Digital Signal Controller), i.e. the software version of the DRIVECOD unit.
The meaning of the 16 bits in the attribute is as follows:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Major number								Minor number							

EXAMPLE
Value 04 01 hex in hexadecimal notation corresponds to the binary representation 0000 0100 0000 0001 and has to be interpreted as: version 4.1.

Default = Device dependent

64-01-24 Hardware Version

[UINT, Get, NV]
This is meant to show the hardware version and model of the DRIVECOD unit.
The meaning of the 16 bits in the attribute 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.

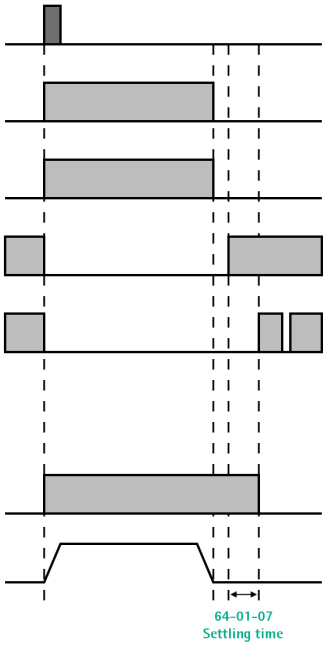
Default = Device dependent

NOTE

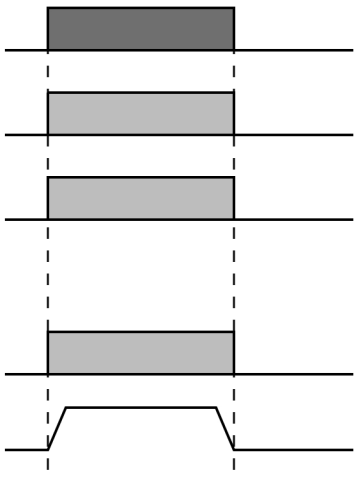
Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the **Save parameters** function available in the **64-01-01 Control Word** attribute, see on page 116.

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

EXAMPLE 1

Positioning	RD1xA
Start (bit 6 - 64-01-01 Control Word) **Executing a command** (bit 7 - 64-01-03 Status Word) **Axis running** (bit 6 - 64-01-03 Status Word) **Target position reached** (bit 8 - 64-01-03 Status Word) **Axis in position** (bit 0 - 64-01-03 Status Word) **Axis enabled** (bit 2 - 64-01-03 Status Word: PWM is active) Theoretical velocity	

EXAMPLE 2

Jog +	RD1xA
Jog + (bit 0 - 64-01-01 Control Word) **Executing a command** (bit 7 - 64-01-03 Status Word) **Axis running** (bit 6 - 64-01-03 Status Word) **Axis enabled** (bit 2 - 64-01-03 Status Word: PWM is active) Theoretical velocity	

7.12.6 Class F5h: TCP/IP Interface Object

Class Code	Object Class	Access	Nr. of Instances
F5h	TCP/IP Interface Object	Get	1

The TCP/IP Interface Object provides the mechanism to configure the TCP/IP network interface of a device. Examples of configurable items include the device's IP Address, Network Mask, and Gateway Address.
For complete information on TCP/IP Interface Object attributes refer to the publication "The CIP Networks Library. Volume 2. EtherNet/IP Adaptation of CIP".

7.12.6.1 Supported Class Services

The supported **Class Services** of the TCP/IP Interface Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.6.2 Class Attributes

F5-01 Revision

[UINT, Get, NV]
Object revision. The current value assigned to this attribute is 0004h.
Default = 0004h

F5-02 Max Instance

[UINT, Get, NV]
The largest instance number of a created object in this class.
Default = 0001h

F5-03 Number of Instances

[UINT, Get, NV]
The number of object instances in this class.
Default = 0001h

7.12.6.3 Supported Instance Services

The supported **Instance Services** of the TCP/IP Interface Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
0Eh = Get_Attribute_Single: used to read the value of an attribute.
10h = Set_Attribute_Single: used to write connection class attribute value.

7.12.6.4 Instance Attributes

F5-01-01 Status

[DWORD, Get, V]

This attribute represents the current status of the interface. Its value changes as the state of the interface changes. The Status attribute is a DWORD, with the following bit definitions:

Bit(s)	Called	Definition
0 ... 3	Interface Configuration Status	It indicates the status of the F5-01-05 Interface Configuration attribute. 0 = the F5-01-05 Interface Configuration attribute has not been configured. 1 = the F5-01-05 Interface Configuration attribute contains configuration obtained from DHCP or non-volatile storage. 2 = the F5-01-05 Interface Configuration attribute contains configuration from hardware settings. 3 ... 15 = reserved for future use
4	Mcast Pending	If set to 1 it indicates a multicast pending configuration.
5	Interface Configuration Pending	If set to 1 it indicates an interface pending configuration. A new configuration will be loaded at next reset.
6	AcdStatus	It indicates when an IP address conflict has been detected by ACD. To enable/disable the ACD refer to F5-01-0A SelectAcd attribute on page 140.
7	AcdFault	It indicates when an IP address conflict has been detected by ACD or the defense failed, and that the current Interface Configuration cannot be used due to this conflict.
8 ... 31	Reserved	Reserved, shall be 0

F5-01-02 Configuration Capability

[DWORD, Get, NV]

It indicates the method of obtaining an initial IP address.

Bit(s)	Called	Definition
0 1 2 3	**BOOTP Client** **DNS Client** **DHCP Client** **DHCP-DNS Update**	It is set to 4 (0010_2): the actuator is able of obtaining its network configuration via DHCP.
4	**Configuration Settable**	If set to 1, it indicates that the F5-01-05

		Interface Configuration attribute is settable.
5	**Hardware Configurable**	The actuator is hardware configurable when the bit is set to 1.
6	**Reset Required at change**	It is always set to 0.
7	**AcdCapable**	If set to 1, the actuator is capable of detecting address conflicts (ACD capable). See the F5-01-0A SelectAcd attribute on page 140.
8 ... 31	Reserved	Reserved, shall be 0

F5-01-03 Configuration Control

[DWORD, Get/Set, NV]

It is used to control network configuration options.

When its value is **0**, the device shall use statically-assigned IP configuration values from non-volatile memory.

When its value is **2**, the device shall obtain the interface configuration values from DHCP.

F5-01-04 Physical Link Object

[Struct of, Get, NV]

This attribute identifies the object associated with the underlying physical communications interface.

Path size

[UINT] Size of path (0002h).

Path

[Padded EPATH] Path to Ethernet Link Object, F6-01-03 Physical Address instance, see on page 143 (20 F6 24 03h).

F5-01-05 Interface Configuration

[Struct of, Get/Set, V/NV]

IP Address

[UDINT] The device's IP address (192.168.1.10).

Network Mask

[UDINT] The device's network mask (255.255.255.0).

Gateway Address

[UDINT] The IP address of the device's default gateway (0.0.0.0).

Name Server

[UDINT] Primary DNS.

Name Server 2
[UDINT] Secondary DNS.

Domain Name
[STRING] The default domain name.

F5-01-06 Host Name

[STRING, Get/Set, NV]
It contains the device's host name, which can be used for informational purposes.

F5-01-08 TTL Value

[USINT, Get/Set, NV]
The device shall use the TTL value for the IP header Time-to-live field when sending EtherNet/IP packets via IP multicast.
Default = 1

F5-01-09 Mcast Config

[Struct of, Set, NV]
It contains the configuration of the device's IP multicast addresses to be used for EtherNet/IP multicast packets.

Alloc Control
[USINT] 0 = multicast addresses shall be generated using the default allocation algorithm according to specifications. 1 = multicast addresses shall be allocated according to the values specified in **Num Mcast** and **Mcast Start Addr** parameters.

(reserved)
[USINT] set to 0, do not change.

Num Mcast
[UINT] Number of IP multicast addresses allocated (1).

Mcast Start Addr
[UDINT] Starting multicast address from which **Num Mcast** addresses are allocated.

F5-01-0A SelectAcd

[BOOL, Set, NV]
It allows to enable / disable Address Conflict Detection (ACD). If ACD is enabled, as soon as an address conflict is detected, the bit 6 **AcdStatus** in the **F5-01-01 Status** attribute will be set to 1 and NS Network State Error LED will light on red.
0 = Disable ACD
1 = Enable ACD
Default = 1

F5-01-0B LastConflictDetected

[Struct of, Set, NV]
It is a diagnostic attribute presenting information about the ACD state when the last IP address conflict was detected.

AcdActivity
[USINT] State of the ACD algorithm when the last IP address conflict was detected.

RemoteMAC
[Array of 6 USINTs] The IEEE 802.3 source MAC address from the header of the received Ethernet packet sent by the device when reporting the conflict.

ArpPDU
[Array of 28 USINTs] The ARP Response PDU in binary format.

F5-01-0C EtheNet/IP QuickConnect

[BOOL, Set, NV]
It shall enable (1) or disable (0) the EtherNet/IP QuickConnect feature. If EtherNet/IP QuickConnect is enabled, it will direct EtherNet/IP target devices to quickly power up and join an EtherNet/IP network.
Default = 0

F5-01-0D Encapsulation Inactivity Timeout

[UINT, Set, NV]
Number of seconds with no Encapsulation activity before the TCP connection is closed. It is disabled (0).
Default = 0 (min. value 0, max. value 3600)

7.12.7 Class F6h: Ethernet Link Object

Class Code	Object Class	Access	Nr. of Instances
F6h	Ethernet Link Object	Get	1

The EtherNet Link Object maintains link-specific counters and status information for an IEEE 802.3 communications interface such as transmission speed, interface status and the MAC address.

7.12.7.1 Supported Class Services

The supported **Class Services** of the Ethernet Link Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.7.2 Class Attributes

F6-01 Revision

[UINT, Get, NV]
Object revision. The current value assigned to this attribute is 0004h.
Default = 0004h

F6-02 Max Instance

[UINT, Get, NV]
The largest instance number of a created object in this class (1 or 3).
Default = 0003h

F6-03 Number of Instances

[UINT, Get, NV]
The number of object instances in this class (1 or 3).
Default = 0003h

7.12.7.3 Supported Instance Services

The supported **Instance Services** of the Ethernet Link Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
0Eh = Get_Attribute_Single: used to read the value of an attribute.
10h = Set_Attribute_Single: used to write connection class attribute value.
4Ch = Get_And_Clear: used to get and then clear the specified attribute.

7.12.7.4 Instance Attributes

F6-01-01 Interface Speed

[UDINT, Get, V]
Interface speed currently in use, expressed in Mbps (10 or 100).

F6-01-02 Interface Flags

[DWORD, Get, V]
Interface status flags, according to the following table.

Bit(s)	Called	Definition
0	Link Status	It indicates whether or not the IEEE 802.3 communications interface is connected to an active network. 0 indicates an inactive link; 1 indicates an active link.
1	Half/Full Duplex	It indicates the duplex mode currently in use. 0 indicates the interface is running half duplex; 1 indicates full duplex. If the Link Status flag is 0, then the value of the Half/Full Duplex flag is indeterminate.
2 ... 4	Negotiation Status	It indicates the status of link auto-negotiation. 0 = Auto-negotiation in progress 1 = Auto-negotiation and speed detection failed. Using default values. Recommended defaults are 10 Mbps and half duplex. 2 = Auto-negotiation failed but detected speed. Duplex was defaulted. 3 = Successfully negotiated speed and duplex. 4 = Auto-negotiation not attempted. Forced speed and duplex.
5	Manual Setting Requires Reset	It is 0 when the interface can activate changes to link parameters during runtime. It is 1 when reset is required in order for changes to take effect.
6	Local Hardware Fault	0 indicates the interface detects no local hardware fault; 1 indicates a local hardware fault is detected.
7 ... 31	Reserved	Reserved, shall be 0

F6-01-03 Physical Address

[Array of 6 UINTs, Get, NV]
MAC ID. This attribute contains the physical network address, i.e. the assigned MAC address.

F6-01-04 Interface Counters

[Struct of, Get, V]
This attribute contains counters relevant to the receipt of packets on the interface.

In Octets
[UDINT] Octets received on the interface.

In Ucast Packets
[UDINT] Unicast packets received on the interface.

In NUcast Packets
[UDINT] Non-unicast packets received on the interface.

In Discards
[UDINT] Inbound packets received on the interface but discarded.

In Errors
[UDINT] Inbound packets that contain errors (does not include **In Discards**).

In Unknown Protos
[UDINT] Inbound packets with unknown protocol.

Out Octets
[UDINT] Octets sent on the interface.

Out Ucast Packets
[UDINT] Unicast packets sent on the interface.

Out NUcast Packets
[UDINT] Non-unicast packets sent on the interface.

Out Discards
[UDINT] Outbound packets discarded.

Out Errors
[UDINT] Outbound packets that contain errors (does not include **Out Discards**).

F6-01-05 Media Counters

[Struct of, Get, V]
This attribute contains counters specific to Ethernet media.

Alignment Errors
[UDINT] Frames received that are not integral number of octets in length.

FCS Errors
[UDINT] Frames received that do not pass the FCS check.

Single Collisions
[UDINT] Successfully transmitted frames which experienced exactly one collision.

Multiple Collisions
[UDINT] Successfully transmitted frames which experienced more than one collision.

SQE Test Errors
[UDINT] Number of times SQE test error message is generated.

Deferred Transmissions
[UDINT] Frames for which first transmission attempt is delayed because the medium is busy.

Late Collisions
[UDINT] Number of times a collision is detected later than 512 bit-times into the transmission of a packet.

Excessive Collisions
[UDINT] Frames for which transmission fails due to excessive collisions.

MAC Transmit Errors
[UDINT] Frames for which transmission fails due to an internal MAC sublayer transmit error.

Carrier Sense Errors
[UDINT] Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame.

Frame Too Long
[UDINT] Frames received that exceed the maximum permitted frame size.

MAC Receive Errors
[UDINT] Frames for which reception on an interface fails due to an internal MAC sublayer receive error.

F6-01-06 Interface Control

[Struct of, Get/Set, NV]
This attribute is a structure consisting of the following parameters.

Control Bits

[WORD] Interface control bits.

Bit(s)	Called	Definition
0	**Auto-negotiate**	0 indicates that 802.3 link auto-negotiation is disabled. 1 indicates that auto-negotiation is enabled. If auto-negotiation is disabled, then the device shall use the settings indicated by the Forced Duplex Mode and Forced Interface Speed bits.
1	**Forced Duplex Mode**	If the **Auto-negotiate** bit is 0, the **Forced Duplex Mode** bit indicates whether the interface shall operate in full or half duplex mode. 0 indicates that the interface duplex should be half duplex. 1 indicates that the interface duplex should be full duplex. Interfaces not supporting the requested duplex shall return status code 0x09 (**Invalid Attribute Value**). If auto-negotiation is enabled, attempting to set the **Forced Duplex Mode** bit shall result in status code 0x0C (**Object State Conflict**).
2 ... 15	Reserved	Reserved, shall be 0

Forced Interface Speed

[UINT] If the **Auto-negotiate** bit is 0, the **Forced Interface Speed** bits indicate the speed at which the interface shall operate. Speed is specified in megabits per second (e.g., for 10 Mbps Ethernet, the Interface Speed shall be 10).

F6-01-07 Interface Type

[USINT, Get, NV]
This attribute indicates the type of the physical interface according to the following table.

Instance	Value	Type of interface
1	2	Twisted-pair
2	2	Twisted-pair
3	1	The interface is internal to the device

F6-01-08 Interface State

[USINT, Get, V]
This attribute indicates the current operational state of the interface according to the following table.

Value	Interface State
0	Unknown interface state
1	The interface is enabled and is ready to send and receive data
2	The interface is disabled
3	The interface is testing
4 ... 255	Reserved

F6-01-09 Admin State

[USINT, Set, V]
This attribute allows administrative setting of the interface state according to the following table.

Value	Admin State
0	Reserved
1	Enable the interface
2	Disable the interface
3 ... 255	Reserved

F6-01-0A Interface Label

[SHORT_STRING, Get, NV]
This attribute is a string that describes the interface according to the following table.

Instance	Value
1	Port 1
2	Port 2
3	Internal

F6-01-0B Interface Capability

[Struct of, Get, NV]

This attribute indicates the set of capabilities for the interface according to the following table.

Bit(s)	Called	Definition
0	**Manual Setting Requires Reset**	It indicates whether or not the device requires a reset to apply changes made to the **F6-01-06 Interface Control** attribute. 0 = It indicates that the device automatically applies changes made to the **F6-01-06 Interface Control** attribute and, therefore, does not require a reset in order for changes to take effect. This is the value this bit shall have when the **F6-01-06 Interface Control** attribute is not implemented. 1 = It indicates that the device does not automatically apply changes made to the **F6-01-06 Interface Control** attribute and, therefore, will require a reset in order for changes to take effect. Note: this bit shall also be replicated in the **F6-01-02 Interface Flags** attribute in order to retain backwards compatibility with previous object revisions.
1	**Auto-negotiate**	0 = It indicates that the interface does not support link auto-negotiation (internal interface) 1 = It indicates that the interface supports link auto-negotiation (external interface)
2	**Auto-MDIX**	0 = It indicates that the interface does not support auto MDIX operation (internal interface) 1 = It indicates that the interface supports auto MDIX operation (external interface)
3	**Manual Speed/Duplex**	0 = It indicates that the interface does not support manual setting of speed/duplex. The **F6-01-06 Interface Control** attribute shall not be supported (internal interface) 1 = It indicates that the interface supports manual setting of speed/duplex via the **F6-01-06 Interface Control** attribute (external interface)
4 ... 31	Reserved	Reserved, shall be 0

7.12.8 Class 47h: Device Level Ring (DLR) Object

Class Code	Object Class	Access	Nr. of Instances
47h	Device Level Ring (DLR) Object	Get	1

The Device Level Ring (DLR) Object provides the configuration and status information interface for the DLR protocol. The DLR protocol is a layer 2 protocol that enables the use of an Ethernet ring topology. The DLR Object provides the CIP application-level interface to the protocol. The DLR protocol is fully specified in Chapter 9 of the publication "THE CIP NETWORKS LIBRARY, Volume 2, EtherNet/IP Adaptation of CIP".

7.12.8.1 Supported Class Services

The supported **Class Services** of the Device Level Ring (DLR) Object are:
01h = Get_Attribute_All: used to read the value of all attributes.
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.8.2 Class Attributes

47-01 Revision

[UINT, Get, NV]
Object revision. The current value assigned to this attribute is 0003h.
Default = 0003h

7.12.8.3 Supported Instance Services

The supported **Instance Services** of the Device Level Ring (DLR) Object are:
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.8.4 Instance Attributes

47-01-01 Network Topology

[USINT, Get, V]
It indicates the current network topology mode. A value of "0" indicates "Linear" topology; a value of "1" indicates "Ring" topology.

47-01-02 Network Status

[USINT, Get, V]

This attribute provides current status of the network based on the device's view of the network, according to the following table.

Network Status value	Description
0	Normal operation in both Ring and Linear Network Topology modes.
1	Ring Fault. A ring fault has been detected. Valid only when **47-01-01 Network Topology** is "1" = Ring.
2	Unexpected Loop Detected. A loop has been detected in the network. Valid only when **47-01-01 Network Topology** is "0" = Linear.
3	Partial Network Fault. A network fault has been detected in one direction only. Valid only when **47-01-01 Network Topology** is "1" = Ring and the node is the active ring supervisor.
4	Rapid Fault/Restore Cycle. A series of rapid ring fault/restore cycles has been detected. Similar to the Partial Network Fault status (3), the supervisor remains in a state with forwarding blocked on its ring ports. The condition must be cleared explicitly via the "Clear Rapid Faults" service.

47-01-0A Active Supervisor Address

[Struct of, Get, V]

This attribute contains the IP address (IPv4) and/or Ethernet MAC address of the active ring supervisor. The initial values of IP address and Ethernet MAC address shall be 0, until the active ring supervisor is determined.

47-01-0C Capability Flags

[DWORD, Get, NV]

The Capability Flags describe the DLR capabilities of the device, according to the following table.

Bit(s)	Called	Definition
0	Announce-based Ring Node	It sets if device's ring node implementation is based on processing of Announce frames.
1	Beacon-based Ring Node	It sets if device's ring node implementation is based on processing of Beacon frames.
2 ... 4	Reserved	Reserved, shall be 0
5	Supervisor Capable	It sets if device is capable of providing the supervisor function.
6	Redundant Gateway Capable	It sets if device is capable of providing the redundant gateway function.
7	Flush_Table Frame	It sets if device is capable of supporting the

	Capable	Flush_Tables frame.
8 ... 31	Reserved	Reserved, shall be 0

Default = 0082h = Beacon-based Ring Node + Flush_Table Frame Capable

7.12.9 Class 48h: Quality of Service (QoS) Object

Class Code	Object Class	Access	Nr. of Instances
48h	Quality of Service (QoS) Object	Get	1

The Quality of Service (QoS) Object is used to treat traffic streams with different relative priorities or other delivery characteristics. Standard QoS mechanisms include IEEE 802.1D/Q (Ethernet frame priority) and Differentiated Services (DiffServ) in the TCP/IP protocol suite.
The QoS Object provides a means to configure certain QoS-related behaviors in EtherNet/IP devices.
The QoS Object is required for devices that support sending EtherNet/IP messages with non-zero DiffServ code points (DSCP), or sending EtherNet/IP messages in 802.1Q tagged frames.

7.12.9.1 Supported Class Services

The supported **Class Services** of the Quality of Service (QoS) Object are:
0Eh = Get_Attribute_Single: used to read the value of an attribute.

7.12.9.2 Class Attributes

48-01 Revision

[UINT, Get, NV]
Object revision. The current value assigned to this attribute is 0001h.
Default = 0001h

7.12.9.3 Supported Instance Services

The supported **Instance Services** of the Quality of Service (QoS) Object are:
0Eh = Get_Attribute_Single: used to read the value of an attribute.
10h = Set_Attribute_Single: used to write connection class attribute value.

7.12.9.4 Instance Attributes

48-01-01 802.1Q Tag Enable

[USINT, Set, NV]
This attribute enables (1) or disables (0) sending 802.1Q frames on CIP and IEEE 1588 messages. When the attribute is enabled, the device shall send 802.1Q frames for all CIP and IEEE 1588 messages.

48-01-04 DSCP Urgent

[USINT, Set, NV]
DSCP value for CIP transport class 1 Urgent priority messages.
Default = 55

48-01-05 DSCP Scheduled

[USINT, Set, NV]
DSCP value for CIP transport class 1 Scheduled priority messages.
Default = 47

48-01-06 DSCP High

[USINT, Set, NV]
DSCP value for CIP transport class 1 High priority messages.
Default = 43

48-01-07 DSCP Low

[USINT, Set, NV]
DSCP value for CIP transport class 1 Low priority messages.
Default = 31

48-01-08 DSCP Explicit

[USINT, Set, NV]
DSCP value for CIP explicit messages (transport class 3 and UCMM) and all other EtherNet/IP encapsulation messages.
Default = 27

8 Integrated Web Server

8.1 Integrated web server – Preliminary information

EtherNet/IP actuator from Lika Electronic integrate a web server. This web-based user interface is designed to offer helpful functions and deliver complete information on the device that can be accessed through the Internet.
In particular it allows:

- to display and check the currently set parameters;
- to set the network communication parameters;
- to set some parameters such as the preset;
- to upgrade the firmware;
- to monitor the actuator and access some advanced maintenance functions.

The web server can be accessed from any PC running a web browser. Since its only requirement is a HTTP connection between the web browser and the web server running on the device, it is perfectly fitted also for remote access scenarios.

Before opening the EtherNet/IP actuator web server please ascertain that the following requirements are fully satisfied:

- the actuator is connected to the network;
- the actuator has valid IP address;
- the PC is connected to the network;
- a web browser (Internet Explorer, Mozilla Firefox, Google Chrome, Opera, ...) is installed in the PC or in the device used for connection.

NOTE
This web server has been tested and verified using the following web browsers:

- Internet Explorer IE11 version 11.1246.17134.0
- Mozilla Firefox Quantum version 73.0.1
- Google Chrome version 80.0.3987.116
- Opera version 66.0.3515.103

NOTE
Please note that the snapshot look may vary depending on the used web browser. The following snapshots have been taken from Mozilla Firefox.

8.2 Web server Home page

To open the EtherNet/IP actuator web server proceed as follows:

1. type the IP address of the actuator you want to connect to (in the example: 192.168.1.10, this is the default software IP address set at Lika, see on page 44) in the address bar of your web browser and confirm by pressing **ENTER**;

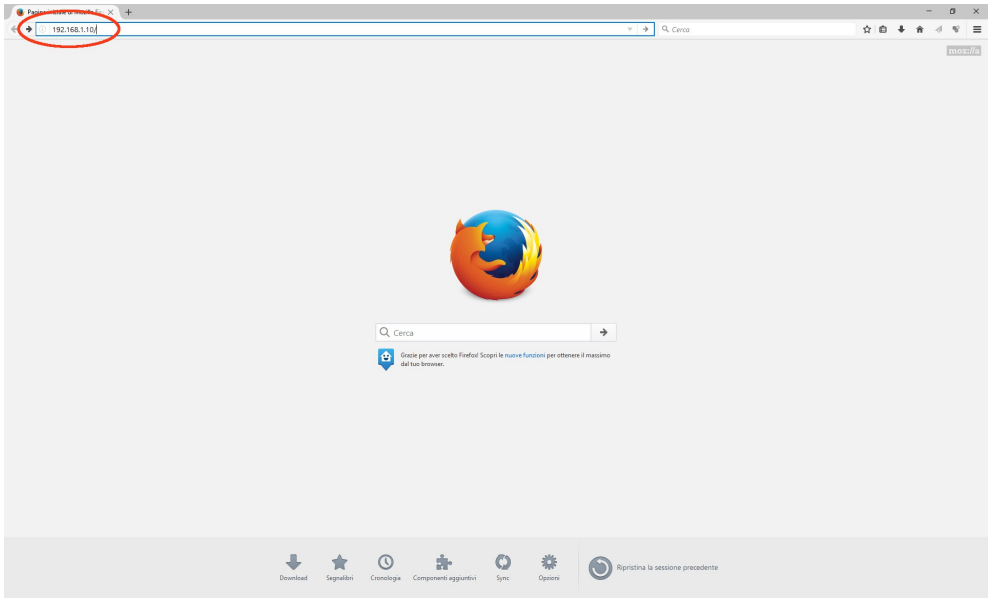


Figure 37 - Opening the web server

2. as soon as the connection is established, the web server **Home** page will appear on the screen;

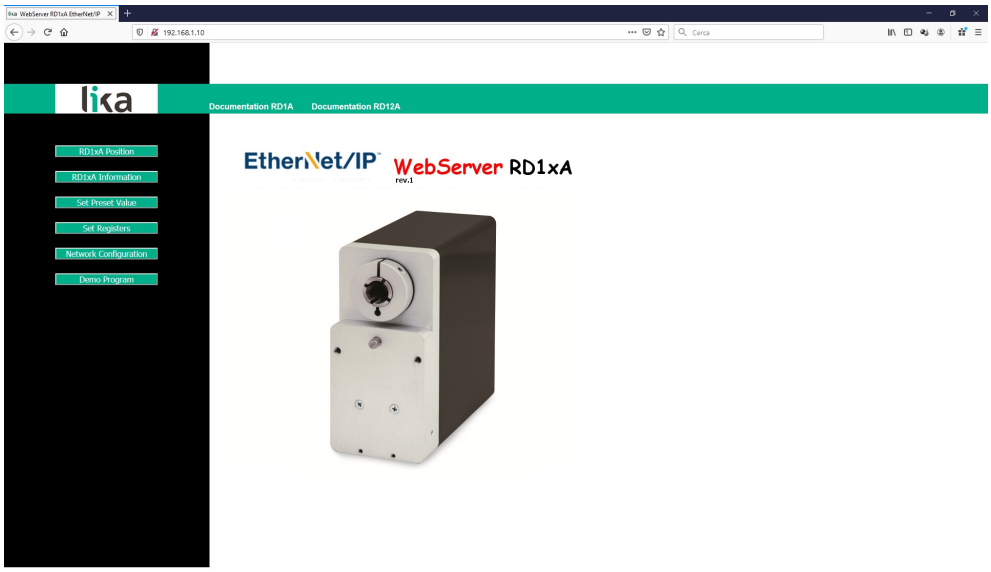


Figure 38 - Web server Home page

Some commands are available in the menu bar of the **Home** page.

Press on the **Lika logo** to enter Lika's web site (www.lika.biz).

Press the **Documentation** command to enter the EtherNet/IP actuator technical documentation page available on Lika's web site (http://www.lika.it/eng/products/rotary-actuators/rotary-actuators) where specific technical information and documentation concerning the EtherNet/IP actuator can be found.

Furthermore some commands are available in the left navigation bar. All the pages that can be entered through the commands in the bar are freely accessible.
These commands allow to enter specific pages where information and diagnostics on the connected actuator as well as useful functions can be achieved.
They are described in the following sections.

8.3 Actuator position and Status Word information page

Press the **RD1xA Position** command in the left navigation bar of the Web server **Home** page to enter the page where the current position of the actuator and the Status Word information are displayed.

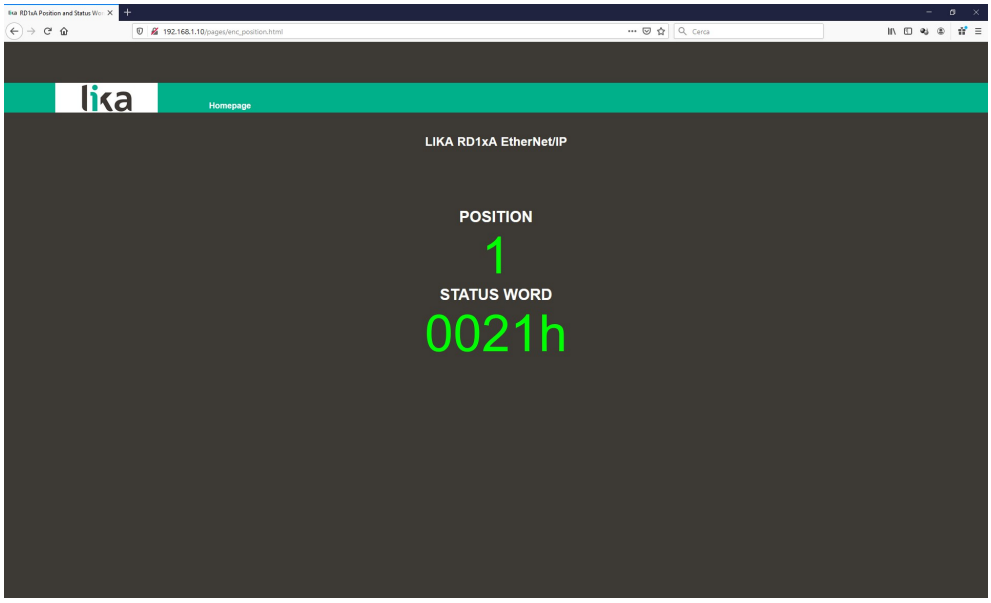


Figure 39 - Actuator position and Status Word information page

The current position of the actuator is expressed in pulses. For any information refer to the **64-01-04 Real Position** attribute on page 121.

The current Status Word information is expressed according to the value of the bits in the **64-01-03 Status Word** attribute. For any information refer to the **64-01-03 Status Word** attribute on page 119.

NOTE
The current actuator position and Status Word values are real-time processed and continuously updated (every 200 msec. on the screen).

Press the **Homepage** command to move back to the Web server **Home** page.

8.3.1 Specific notes on using Internet Explorer

The following options must be set properly on Internet Explorer in order to get the **Actuator position and Status Word information** page to be continuously updated.

- Open the **Settings** menu;
- open the **Internet Options** property sheet;
- in the **General** tabbed page, press the **Setting** button available in the **History Browsing** section;
- under **Check for newer versions of stored pages**, click **Every time I visit the webpage**;
- press the **OK** button to confirm whenever requested.

8.4 RD1xA-EP Information (EtherNet/IP attributes)

Press the **RD1xA Information** command in the left navigation bar of the Web server **Home** page to enter the **RD1xA-EP Information** page. In this page the complete list of the available EtherNet/IP attributes implemented by the manufacturer is displayed. Attributes are expressed in decimal notation, values are expressed in either hexadecimal or decimal notation. The MAC address of the connected actuator is shown under the page name.

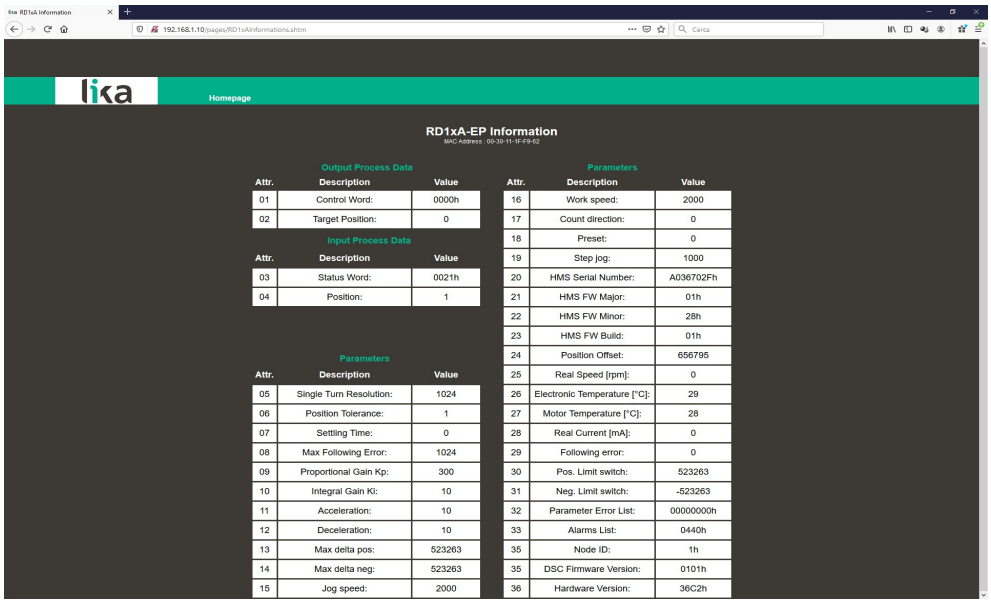


Figure 40 - RD1xA-EP Information page

The attributes listed under the **Output Process Data** section are output process data and read-write (Set) access values.
The attributes listed under the **Input Process Data** section are input process data and read-only (Get) access values.
The attributes listed under the **Parameters** section are the actuator configuration parameters; they can be either read-write (Set) or read-only (Get) access parameters.
For a complete description of the available actuator attributes please refer to the "7.12.5 Class 64h: Application Object" section on page 113.

NOTE
Please note that the values shown in the **RD1xA-EP Information** page are "frozen" in the moment when the page is displayed. To update the values you must refresh the web page.

NOTE
The attributes in the **RD1xA-EP Information** page cannot be changed even though they are read-write access attributes. To change the set values please enter the **Set RD1xA-EP Registers** page (see on page 163).

Press the **Homepage** command to move back to the Web server **Home** page.

8.5 Setting the Preset value

Press the **Set Preset Value** command in the left navigation bar of the Web server **Home** page to enter the **Set RD1xA-EP Preset** page and set/activate a Preset value. For complete information on the preset function please refer to the 64-01-12 Preset attribute on page 127.

A soon as you press the **Set Preset Value** command a warning message (**Are you sure you want to change Preset Value?**) appears on the screen: it warns the operator about the awkwardness of the operation, thus he is required to confirm the procedure before continuing.

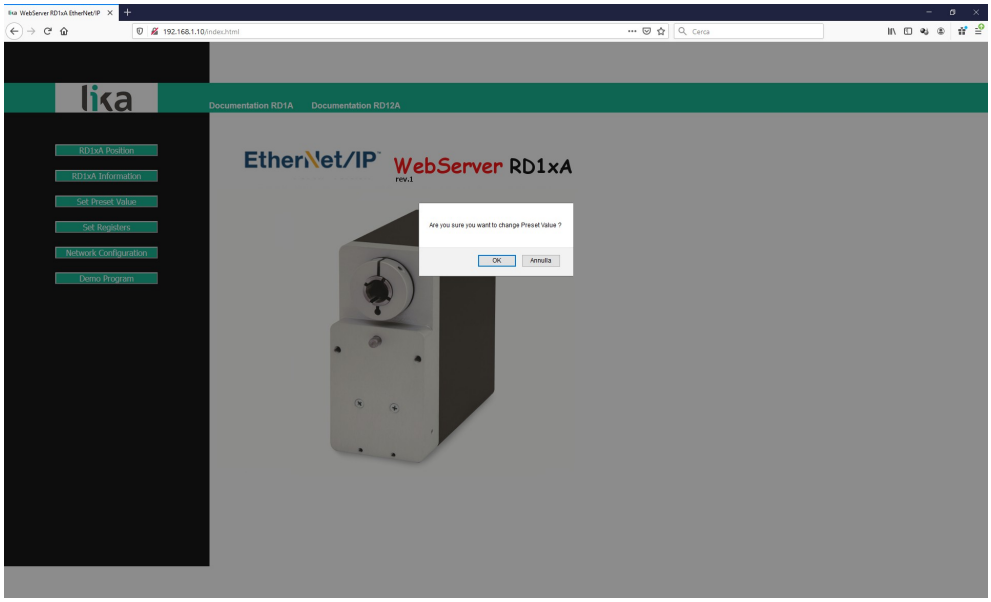


Figure 41 - Entering the Set RD1xA-EP Preset page

Press the **OK** button to proceed.

Otherwise press the **EXIT** button to abort the procedure. The **Set Preset cancelled!** message will appear on the screen. Press the **OK** button to move back to the Web server **Home** page.

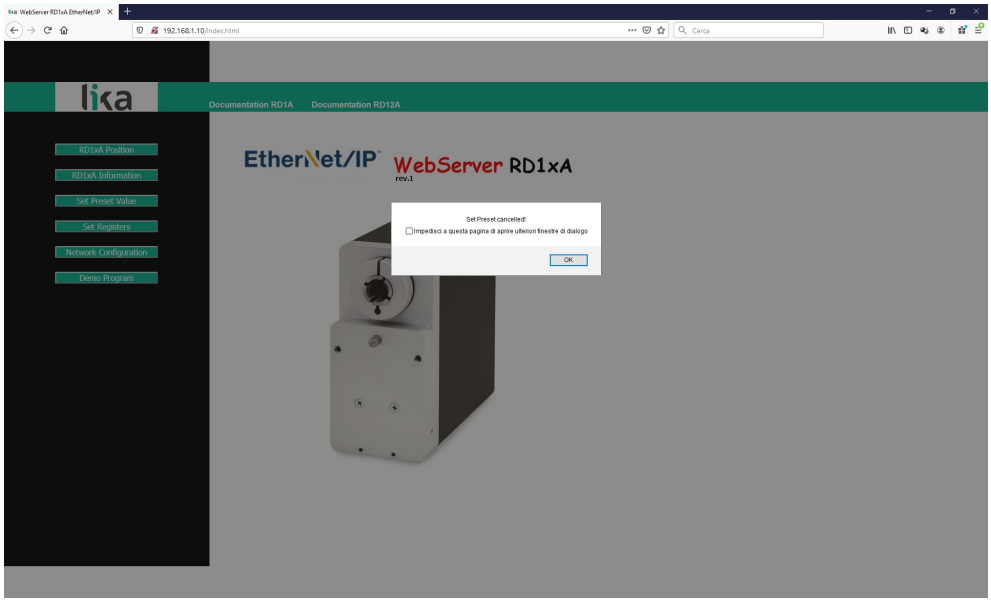


Figure 42 - Preset operation aborted

If you confirm the procedure, the **Set RD1xA-EP Preset** page will appear on the screen:

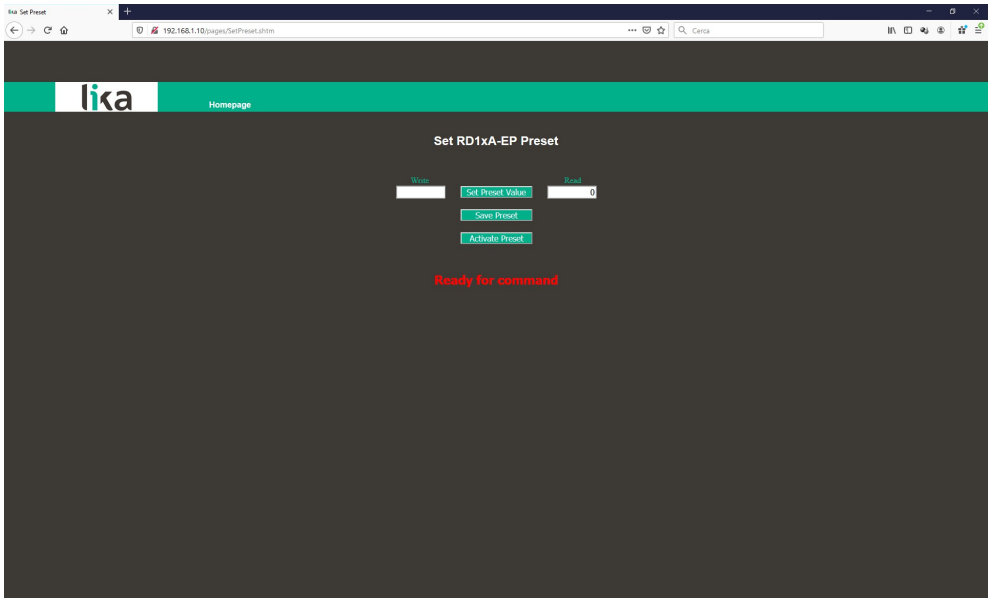


Figure 43 - Set RD1xA-EP Preset page

The Preset value that is currently set in the actuator (see the **64-01-12 Preset** attribute on page 127) will be displayed in the **READ** box.

To change the Preset enter a suitable value in the **WRITE** box and then press the **Set Preset Value** button to confirm. The value has to be set in decimal notation.

NOTE
Please note that the Preset value is now saved temporarily in the **64-01-12 Preset** attribute. To save permanently the set Preset value in the **64-01-12 Preset** attribute, please press the **Save Preset** button. Should the power supply be turned off without saving data, the Preset value that has not been saved on the Flash EEProm will be lost!

The preset value is set and activated for the position of the actuator in the moment when the preset value is transmitted. It is activated as soon as the value is confirmed by pressing the **Set Preset Value** button. We suggest activating the preset value when the actuator is in stop.
If you need to activate a value already set next to the **64-01-12 Preset** and displayed in the **READ** box in a different physical position of the actuator shaft, press the **Activate Preset** button, refer to the bit 11 **Setting the preset** in the **64-01-01 Control Word** attribute, see on page 117.

NOTE
At each confirmation and/or activation of the Preset setting, a message will appear under the buttons (see **No Command sent** message). It informs whether the operation has been accomplished properly or an error occurred (for example **Command was set correctly** if everything went well; or **Command Error!** if something went wrong).

Press the **Homepage** command to move back to the Web server **Home** page.

8.6 Setting the attributes

Press the **Set Registers** command in the left navigation bar of the Web server **Home** page to enter the **Set RD1xA-EP Registers** page. In this page the read-write (Set) access EtherNet/IP actuator attributes available in the Application Object (Class 64h) are displayed and their value can be changed.
For complete information on the actuator attributes please refer to the "7.12.5 Class 64h: Application Object" section on page 113.

A soon as you press the **Set Registers** command a warning message (**Are you sure you want to change Registers Values?**) appears on the screen: it warns the operator about the awkwardness of the operation, thus he is required to confirm the procedure before continuing.

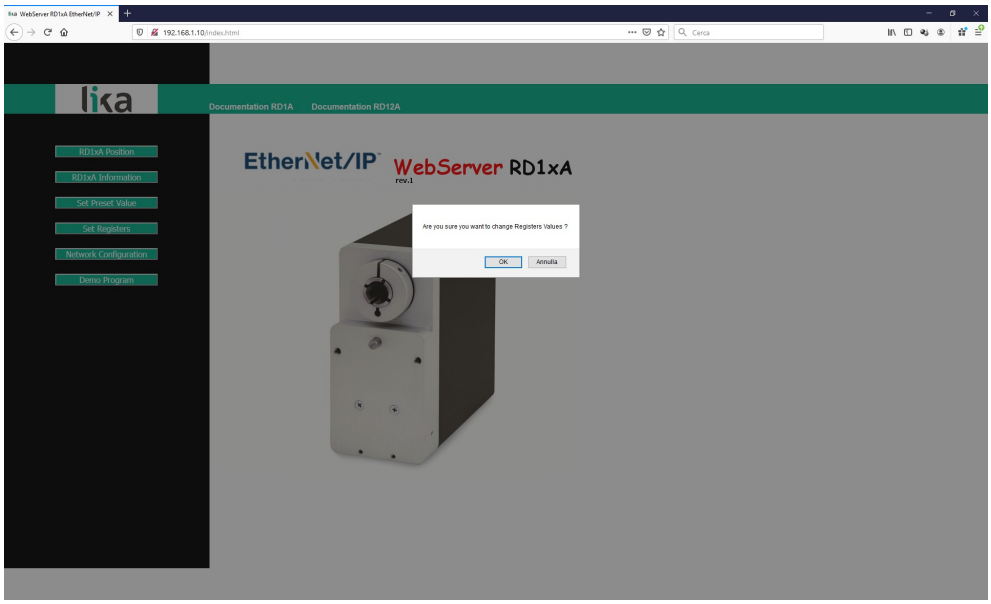


Figure 44 - Entering the Set RD1xA-EP Registers page

Press the **OK** button to proceed, otherwise press the **EXIT** button to abort the procedure. The **Set Registers cancelled!** message will appear on the screen. Press the **OK** button to move back to the Web server **Home** page.

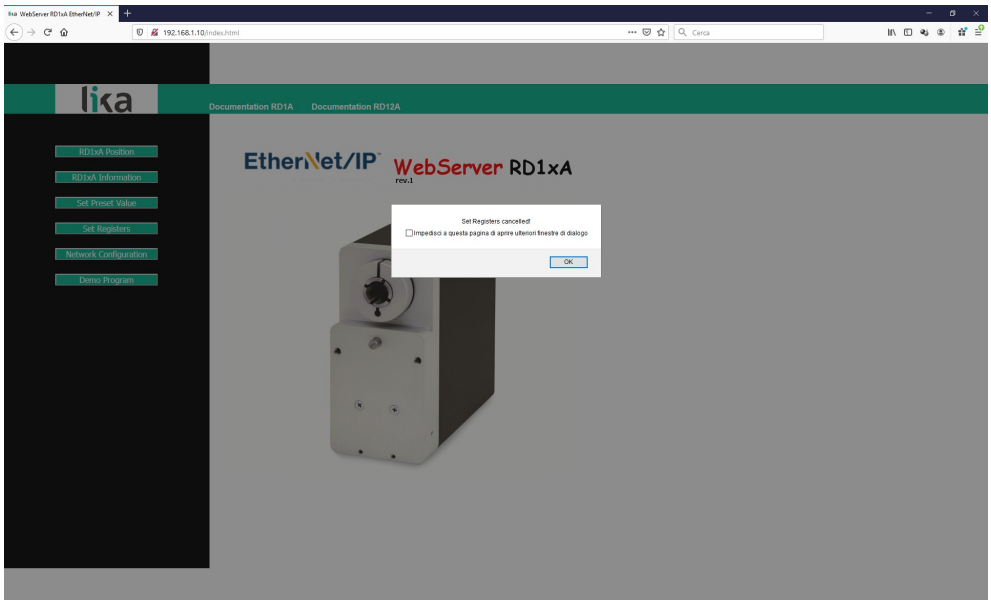


Figure 45 - Register setting operation aborted

If you confirm the procedure, the **Set RD1xA-EP Registers** page will appear on the screen:

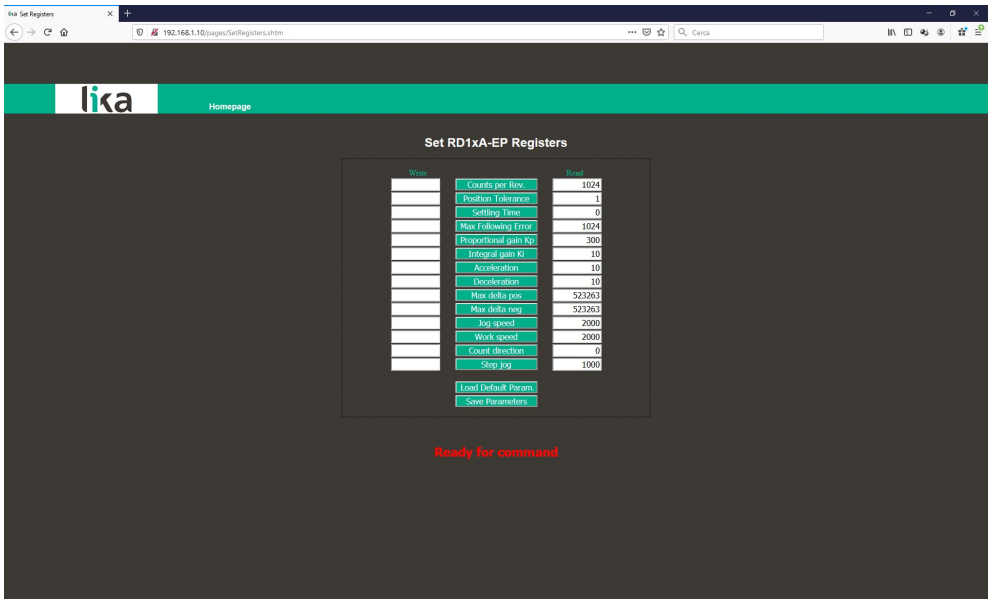


Figure 46 - Set RD1xA-EP Registers page

The values that are currently set in the actuator are displayed in the **READ** box.
To change any value enter a suitable value in the **WRITE** box next to the desired parameter and then press the button between the boxes to confirm. The values have to be set either in decimal notation or by using the drop-down menu (when available).
For complete information on the available attributes please refer to the "7.12.5 Class 64h: Application Object" section on page 113.

EXAMPLE
The **64-01-05 Distance per Revolution** attribute (**Counts per Rev.** in the web server) is currently set to "**1024**" (see the **READ** box in the first line of the Figure above). To change the set value enter a suitable value in the corresponding **WRITE** box of the same line and then press the **COUNTS PER REV.** button to confirm.

NOTE
Please note that, after pressing the button between the boxes, the set value is saved temporarily in the attributes. To save it permanently, please press the **Save Parameters** button. Should the power supply be turned off without saving data, the values that have not been saved on the Flash EEProm will be lost! For more information refer to the "5.1.5 Saving data" section on page 56.

Press the **Load Default Param.** button to restore all parameters to default values. Default values are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode. This function can be useful, for instance, to restore the factory values in case the actuator is set incorrectly and you are not able to resume the proper operation. For more information refer to the "5.1.6 Restoring defaults" section on page 56.

WARNING
The execution of this command causes all parameters which have been set previously to be overwritten!

NOTE
At each confirmation of the set parameters, a message will appear under the buttons (see **No Command sent** message). It informs whether the operation has been accomplished properly or an error occurred (for example **Command was set correctly** if everything went well; or **Command Error!** if something went wrong).

Press the **Homepage** command to move back to the Web server **Home** page.

8.7 Network configuration

Press the **Network Configuration** command in the left navigation bar of the Web server **Home** page to enter the **RD1xA-EP Network Configuration** page. This page allows the operator to configure the TCP/IP properties, that is how the actuator communicates with other devices in the network.
For further information on the network communication parameters please refer to the "4.5 EtherNet/IP Node ID" section on page 44.

WARNING
The network configuration must be accomplished by skilled and competent personnel.

A soon as you press the **Network Configuration** command a warning message (**Are you sure you want to change Network Parameters?**) appears on the screen: it warns the operator about the awkwardness of the operation, thus he is required to confirm the procedure before continuing.

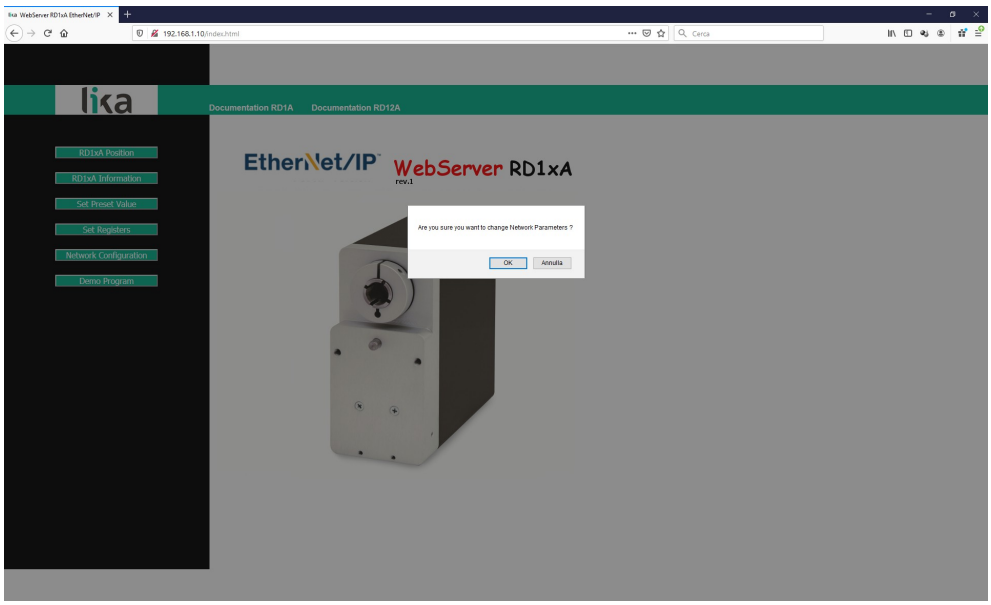


Figure 47 - Entering the RD1xA-EP Network Configuration page

Press the **OK** button to proceed, otherwise press the **EXIT** button to abort the procedure. The **Set Network parameters cancelled!** message will appear on the screen. Press the **OK** button to move back to the Web server **Home** page.

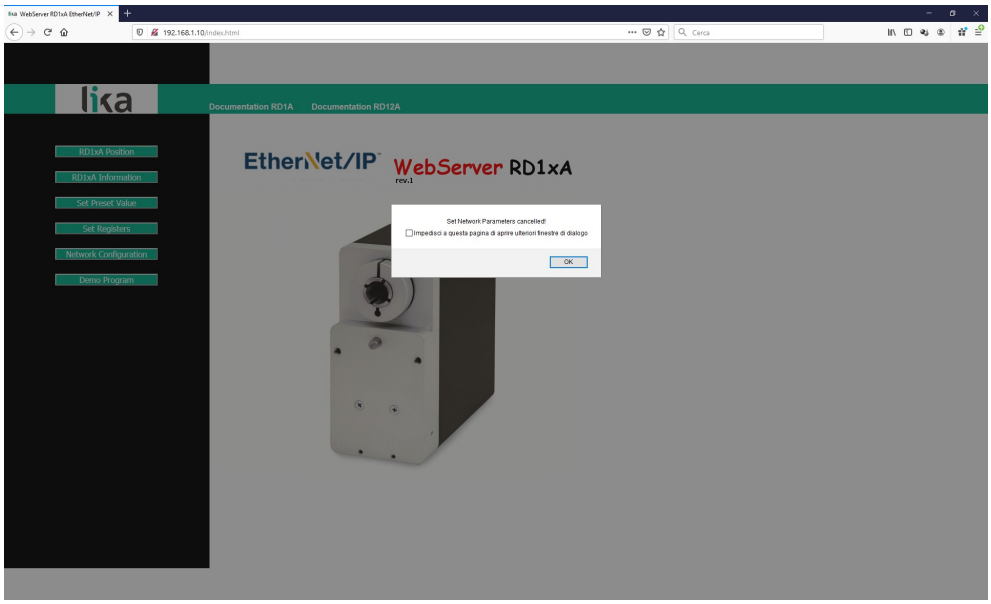


Figure 48 - RD1xA-EP Network configuration aborted

If you confirm the procedure, the **RD1xA-EP Network Configuration** page will appear on the screen:

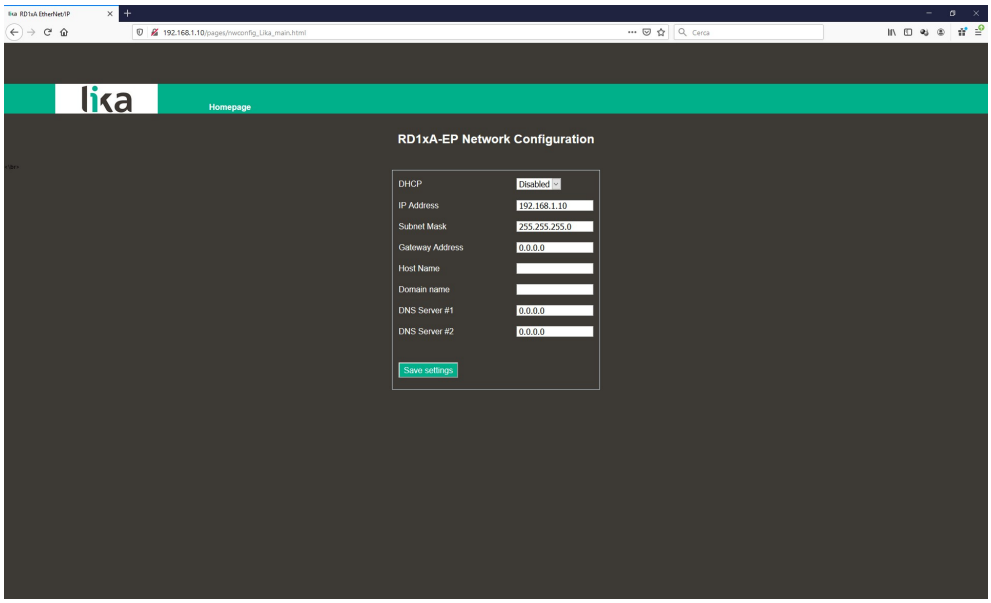


Figure 49 - Network Configuration page

WARNING
Only competent technicians, who are properly trained, have adequate experience and are familiar with computer architecture, network design and operating systems should configure the network communication parameters.
The inappropriate setting of the network parameters results in an incorrect operation of the system.

In this page it is possible to set the parameters that affect the proper communication of the actuator in the TCP/IP network: IP address, Subnet mask, DHCP, DNS, etc.

The following table summarizes the default software IP address and the network configuration parameters.

IP Parameter	Value
IP address	192.168.1.10
Subnet mask	255.255.255.0
Default Gateway	0.0.0.0

To save the set values permanently, please press the **Save Settings** button. Should the power supply be turned off without saving data, the values that have not been saved on the Flash EEProm will be lost!

WARNING
After any setting please note down the configuration values to have access to the actuator and the Web server pages in the future.

WARNING
If you enable the DHCP network protocol (DHCP = ENABLED), then the following default parameters are set for the actuator:
IP ADDRESS = 0.0.0.0
SUBNET MASK = 0.0.0.0
Please check that these settings are allowed by the DHCP server and they are valid address values.

Press the **Homepage** command to move back to the Web server **Home** page.

8.8 Demo Program

Press the **Demo Program** command in the left navigation bar of the Web server **Home** page to enter the **RD1xA-EP Demo Program** page. This page allows the operator to check the operation of the actuator, i.e. to set and command a target position operation and to try the manual movements by forcing the JOG + and JOG - commands.

A soon as you press the **Demo Program** command a safety information message (**To activate Jog+ or Jog- , press only once the relative button and then press 'STOP' to release the command !!**) appears on the screen: it warns the operator about the correct use of the JOG + and JOG - commands, he is required to confirm before continuing.

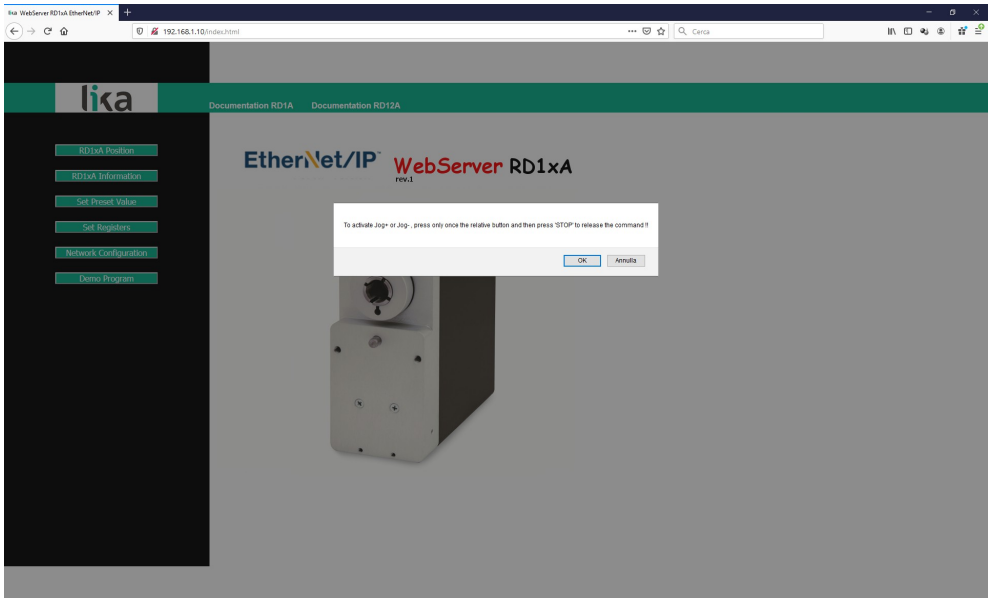


Figure 50 - Entering the RD1xA-EP Demo Program page

Press the **OK** button to proceed, otherwise press the **EXIT** button to abort the procedure. The **Demo Program start cancelled!** message will appear on the screen. Press the **OK** button to move back to the Web server **Home** page.

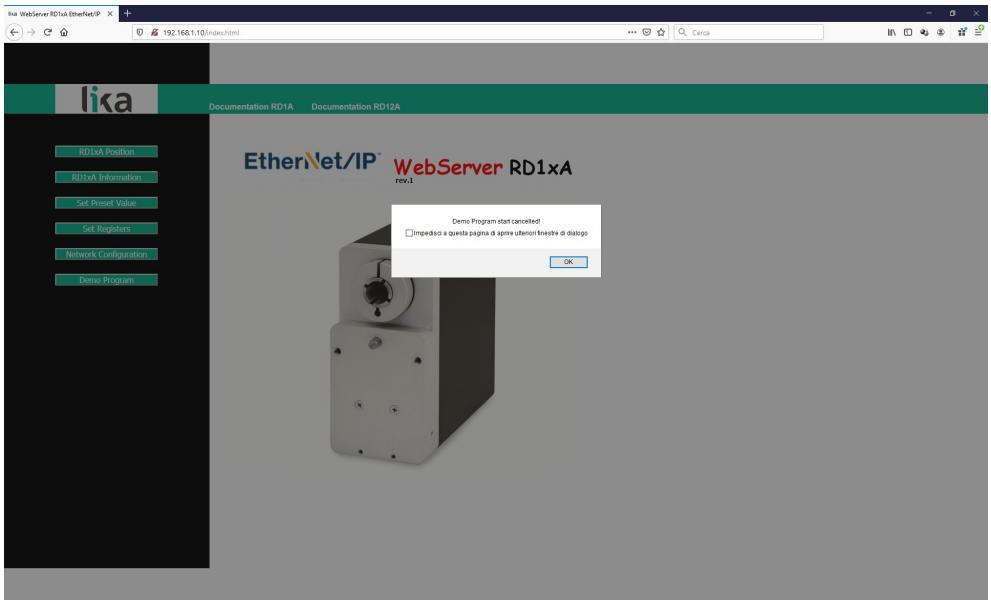


Figure 51 - RD1xA-EP Demo Program aborted

If you confirm the procedure, the **RD1xA-EP Demo Program** page will appear on the screen:

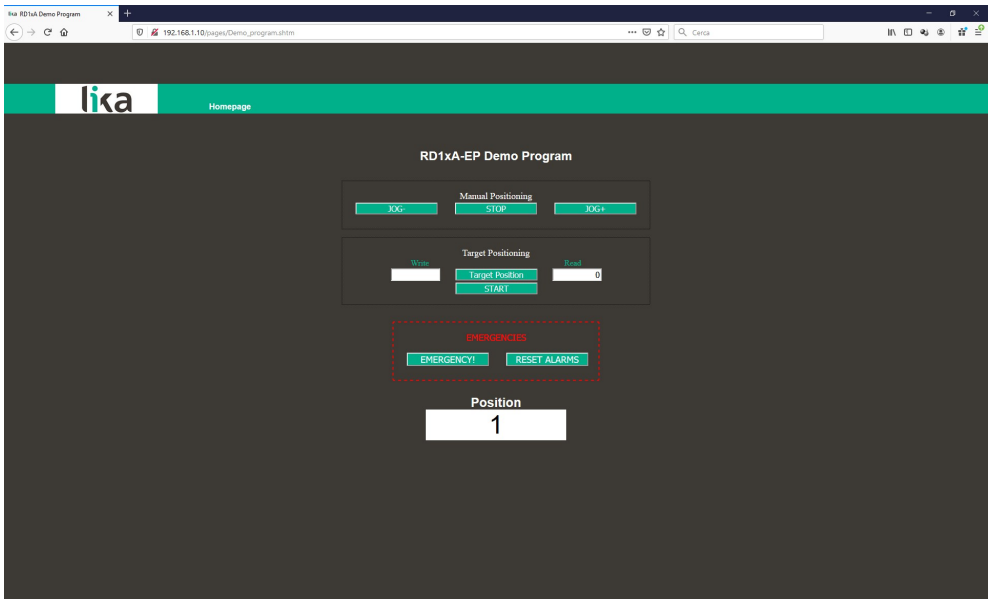


Figure 52 - Demo Program page

NOTE
Before using the manual commands available in the page, press the **RESET ALARMS** button available in the **Emergencies** section to reset the alarm conditions of the actuator and resume the normal work status.

Press the **JOG +** button in the **Manual Positioning** section to force the movement of the motor toward the positive direction; press the **JOG -** button to force the movement of the motor toward the negative direction. Velocity, acceleration, and deceleration are performed according to the values set next to the **64-01-0F Jog speed**, **64-01-0B Acceleration** and **64-01-0C Deceleration** attributes respectively. Press the **STOP** button to stop the movement. For an immediate halt in the movement, press the **EMERGENCY** button available in the **Emergencies** section. For more information please refer to the "Jog: speed control" section on page 83.

WARNING
Before pressing the **JOG +** and **JOG -** 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.

In the **Target Positioning** section the items needed to set the target position and execute a complete positioning cycle are available.
In the **Write** field enter the target position, i.e. the position you want the actuator to reach. Press the **TARGET POSITION** button to confirm the setting, the entered value will be transferred to the **Read** field. Press the **START** button to execute a complete positioning cycle: the device will move in order to reach the set target position. Velocity, acceleration, and deceleration are performed according to the values set next to the **64-01-0F Jog speed**, **64-01-0B Acceleration** and **64-01-0C Deceleration** attributes respectively. For a complete description of the position control see on page 84.

Press the **EMERGENCY** button for an immediate halt in the movements.
Press the **RESET ALARMS** button to reset the alarm conditions of the actuator and resume the normal work status.

Under the **Position** item below in the page the current position of the device is displayed. It is expressed in pulses.

Press the **Homepage** command to move back to the Web server **Home** page.

9 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.

9.1 Configuring the device using Lika's setting up software

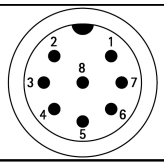
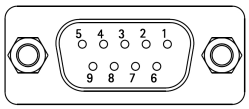
RD1xA DRIVECOD positioning units can be equipped with several communication interfaces such as EtherNet/IP, EtherCAT, POWERLINK, MODBUS RTU, Profibus-DP, CANopen DS 301 etc. All versions except the MODBUS RTU one are equipped with an RS-232 service serial port in compliance with the MODBUS protocol. 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 **ROTADRIVE_INTERFACE.EXE** and is available in the enclosed documentation or at the address **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS**. 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.

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 39.

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

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

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

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

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

RS-232 MODBUS

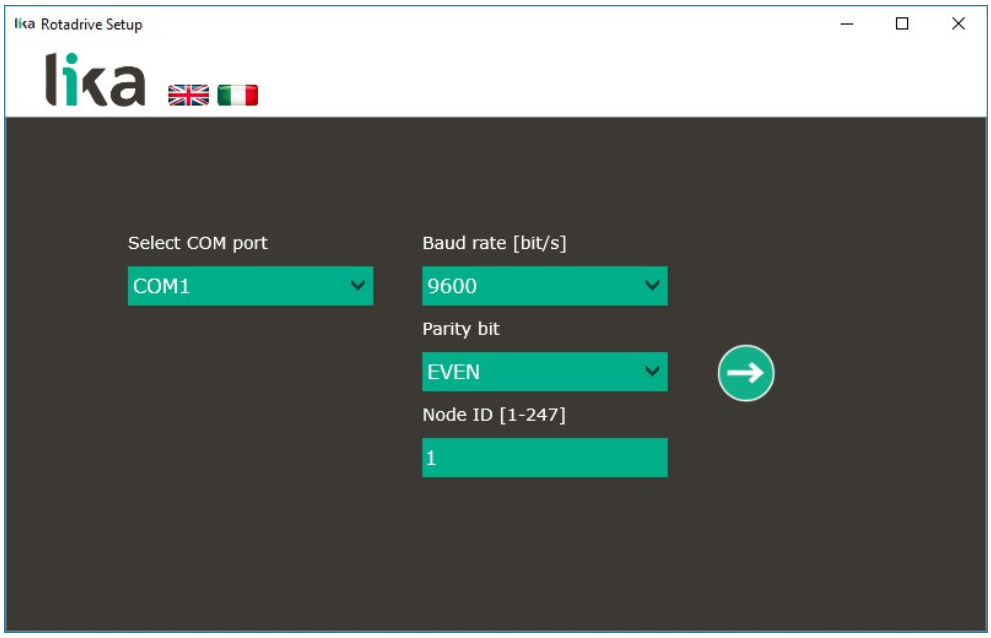
Serial port settings	**Default value**
Baud rate	9600
Byte size	8
Parity	Even
Stop bits	1

The MODBUS address is "1". It cannot be changed. See the "9.2 "Serial configuration" page" section hereafter.

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

9.2 "Serial configuration" page

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



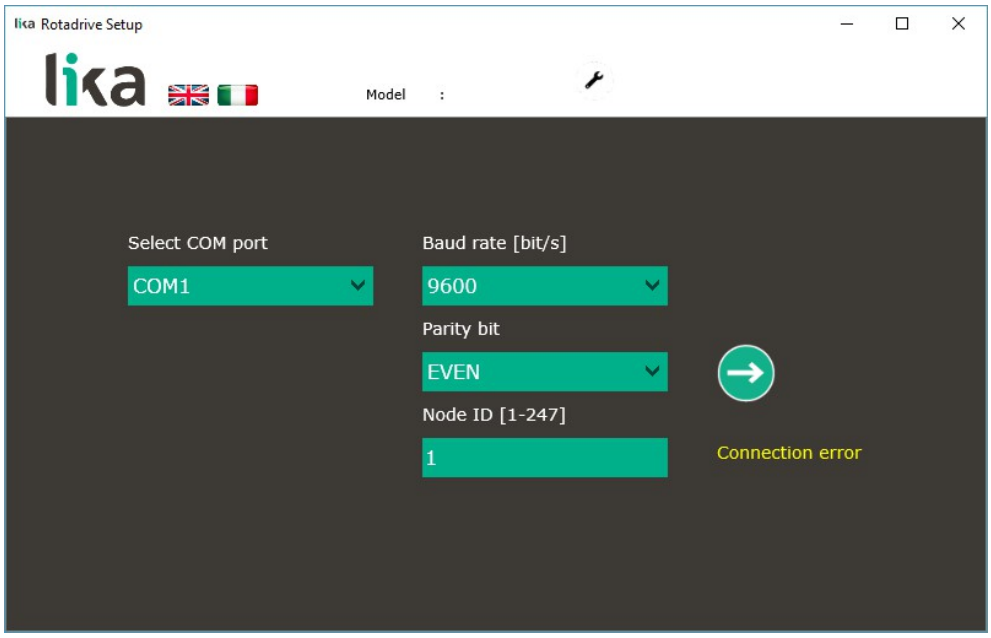
First of all this page allows the operator to choose the language used to display texts and items in the user interface. Click on the **Italian flag** icon to choose the Italian language; click on the **UK flag** icon to choose the English language. The default language is according to the language of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

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

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

For serial port settings see the previous section.
Lastly set the node address of the device the personal computer is connected to through the **Node ID [1-247]** drop-down box (for RD1xA positioning units with EtherNet/IP interface = "**1**"). See on page 42.

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



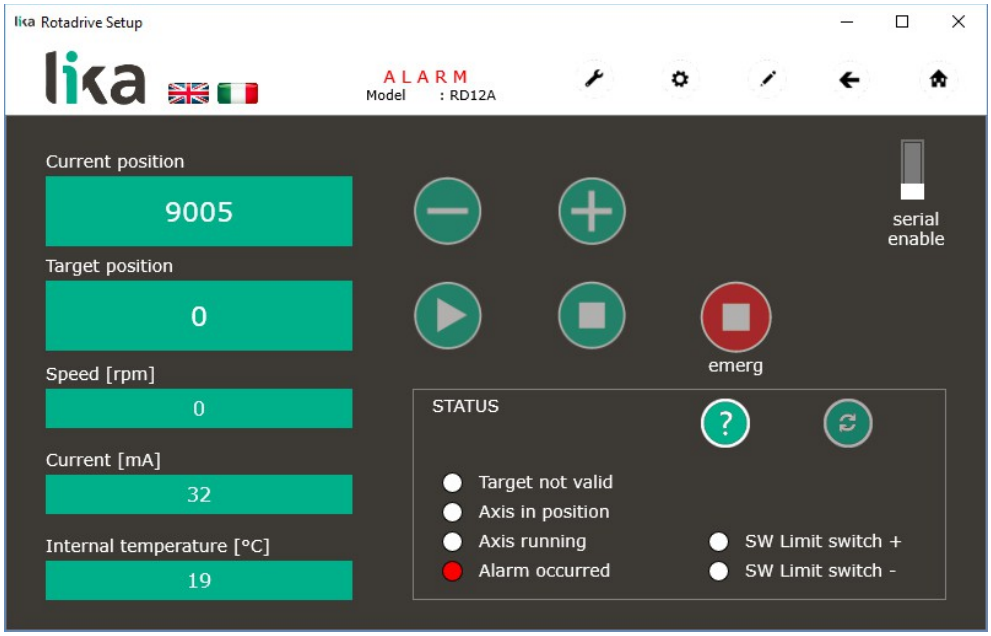
If the connection cannot be established (for instance, because of a wrong COM port setting or a wrong Node ID), the messages **Connection error**, **Device not responding**, **Error opening COM**, **Select COM port** or **Node ID error** may appear under the confirmation button . Please check the settings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

NOTE

Please note that the **FIRMWARE** button appears in the white bar at the top of the page (toolbar). It allows to enter the **Programming firmware** page. For complete information refer to the "9.7 "Programming firmware" page" section on page 182.

9.3 "Main" page

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



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

FIRMWARE

The **FIRMWARE** button allows to enter the **Programming firmware** page. This page allows the operator to upgrade the firmware of the DRIVECOD unit by downloading upgrading data to the flash memory. For complete information refer to the "9.7 "Programming firmware" page" section on page 182.

SETUP

The **SETUP** button allows to enter the **Parameters** page. In this page the list of the parameters available to set the RD1xA positioning units (machine data) is displayed. For complete information refer to the "9.8 "Parameters" page" section on page 185.

SCHEDULE

The **SCHEDULE** button allows to enter the **Program** page. The functions available in the **Program** page allow the operator to create and save work programs in order to test the operation of the RD1xA unit. For complete information refer to the "9.9 "Program" page" section on page 187.

ARROW BACK

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

HOME

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

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

Current position
See the Current position [0x02-0x03] registers on page 223.

Target position
See the Target position [0x2B-0x2C] registers on page 216. Set the position you need the unit to reach and then press the **ENTER** key in the keyboard to confirm it. As soon as you press the **START** button the device starts moving in order to reach the commanded position set next to this **Target position** item, then it stops and activates the **Axis in position** and **Target position reached** status bits (see the "9.5 "Status" box" section on page 180). For detailed information on creating a complete cycle of movements and test the operation of the RD1xA actuator (speed, acceleration, deceleration, etc.) refer to the "9.9 "Program" page" section on page 187.

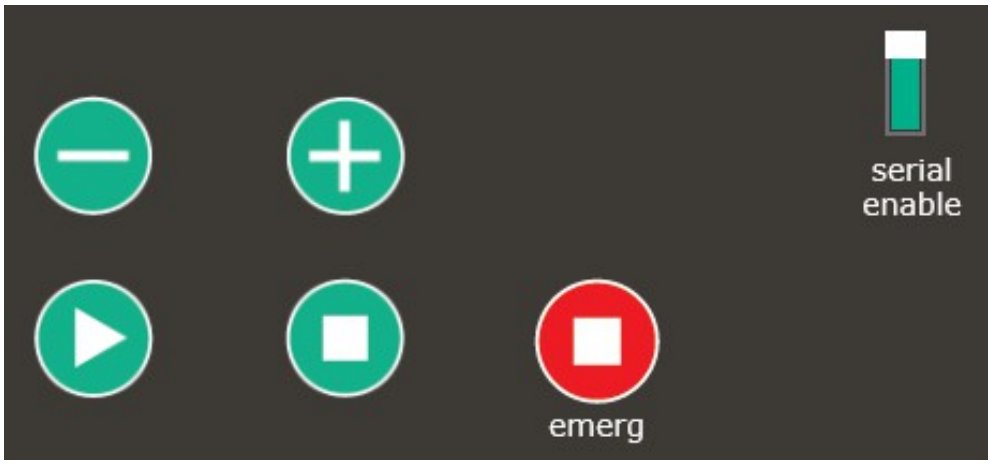
Speed [rpm]
See the Current velocity [0x04] register on page 223.

Current [mA]
See the Current value [0x0B] register on page 225.

Internal temperature [°C]
See the Temperature value [0x07] register on page 224.

9.4 MODBUS commands

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



They allow to control manually and monitor the connected device. When you first enter the **Main** page, all commands are disabled as the unit is still under EtherNet/IP network control. To start programming, controlling manually and monitoring the device through the RS-232 service serial interface and 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, set the **SERIAL ENABLE** slider  (see Extra commands register [0x29] on page 211).
All commands become immediately available for use.

JOG -

See the **Jog -** item on page 212. If the **Incremental JOG** item is enabled = ON, the **Jog step length** value that is set currently appears between the **JOG –** button and the **JOG +** button. This button is available only if the MODBUS commands are enabled (see the **SERIAL ENABLE** slider). See the "9.8 "Parameters" page" section on page 185.

JOG +

See the **Jog +** item on page 212. If the **Incremental JOG** item is enabled = ON, the **Jog step length** value that is set currently appears between the **JOG –** button and the **JOG +** button. This button is available only if the MODBUS commands are enabled (see the **SERIAL ENABLE** slider). See the "9.8 "Parameters" page" section on page 185.

START

Pressing the **START** button causes the unit to start running in order to reach the position set next to the **Target position** item. As soon as the commanded position is reached, the device stops and activates the **Axis in position** and **Target position reached** status bits (see the "9.5 "Status" box" section on page 180). For a normal halt of the device press the **STOP** button; for an immediate emergency halt press the **EMERGENCY** button. This button is available only if the MODBUS commands are enabled (see the **SERIAL ENABLE** slider). See the **Start** item on page 213.

STOP

Press the **STOP** button to force a normal halt of the device, according to the deceleration values. This button is available only if the MODBUS commands are enabled (see the **SERIAL ENABLE** slider). See the **Stop** item on page 213.

EMERGENCY

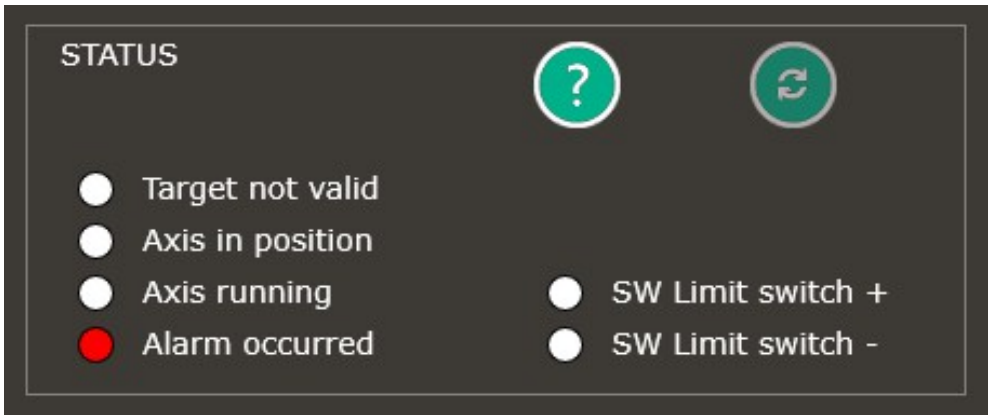
When the unit is running, press the **EMERGENCY** button to force an immediate halt in emergency condition. Press the **RESET** button to restore the normal work condition of the device (see the "9.5 "Status" box" section on page 180). This button is available only if the MODBUS commands are enabled (see the **SERIAL ENABLE** slider). See the **Emergency** item on page 213.

Serial enable

As previously stated, when you first enter the **Main** page, all commands are disabled as the unit is still under EtherNet/IP network control. To start programming, controlling manually and monitoring the device through the RS-232 service serial interface and 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, set the **SERIAL ENABLE** slider (see **Extra commands register [0x29]** on page 211).

9.5 "Status" box

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



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

Please note that at power-on for safety reasons the RD1xA unit is necessarily in an emergency condition: therefore when you first connect and enter the **Main** page the **Alarm occurred** warning is lit red. To know more about the specific active alarm enter the **Alarm and status** page by pressing the **STATUS & INFO** button , refer to the "9.6 "Alarm and status" page" section on page 181. To restore the **Idle** state of the device, press the **RESET** button in the box. Alarm warnings and emergencies will be reset and the normal work condition of the device will be restored.

For complete information on the states and alarms that appear in this box please refer to the **Status word [0x01]** register on page 220; and to the **Alarms register [0x00]** on page 218.

STATUS & INFO

Press this button to enter the **Alarm and status** page where specific information on the states and alarms that are active in the rotary actuator can be found. Refer to the "9.6 "Alarm and status" page" section on page 181.

RESET

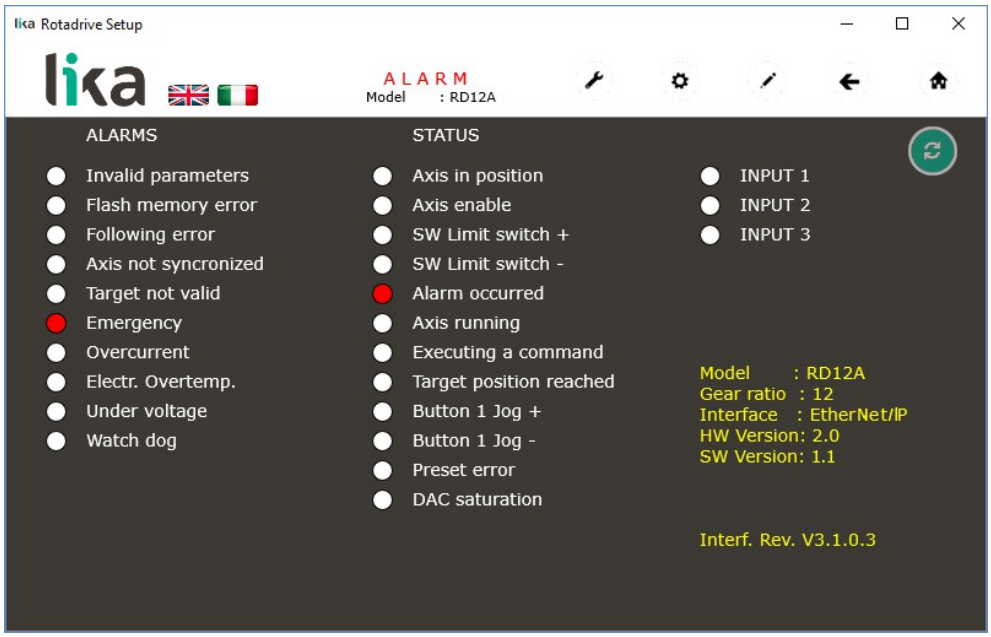
If an alarm is active, it is signalled through the generic warnings in the **Status** box. To know more about the specific active alarm enter the **Alarm and status** page by pressing the **STATUS & INFO** button , refer to the "9.6 "Alarm and status" page" section on page 181. Press this button to reset the alarm and restore the normal work condition of the device. This button is available only if

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

). See the **Alarm reset** item on page 213.

9.6 "Alarm and status" page

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



As previously stated the **Status** box provides short information about the states and the alarms that are active in the rotary actuator. This **Alarm and status** page provides more detailed information on the active statuses and alarms. The active alarms and serious states are lit red. The active ordinary states are lit green.
For complete information on the states and alarms that appear in this page please refer to the Status word [0x01] register on page 220; and to the Alarms register [0x00] on page 218.

Furthermore the page provides detailed information on the connected actuator and the software tool. Data is listed in yellow at the bottom right of the page.
In particular you can found:

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

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

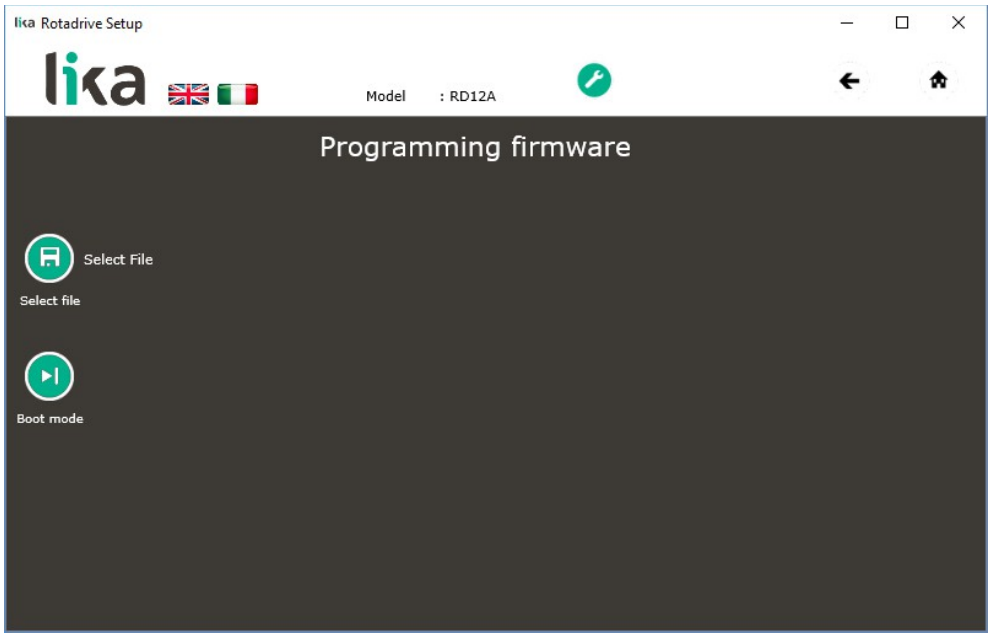
RESET

RESET button . See the previous page.

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

9.7 "Programming firmware" page

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



The functions available in this page allow the operator to upgrade the firmware of the DRIVECOD unit by downloading upgrading data to the flash memory.
The firmware is a software program which controls the functions and operation of a device; the firmware program, sometimes referred to as "user program", is stored in the flash memory integrated inside the DRIVECOD unit. DRIVECOD units are designed so that the firmware can be easily updated by the user himself. This allows Lika Electronic to make new improved firmware programs available during the lifetime of the product.
Typical reasons for the release of new firmware programs are the necessity to make corrections, improve and even add new functionalities to the device.

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

WARNING
The firmware upgrading process for any DRIVECOD unit has to be accomplished by skilled and competent personnel. If the upgrade is not performed according to the instructions provided or a wrong or incompatible firmware program is installed, then the unit may not be updated correctly, in some cases preventing the DRIVECOD unit from working.
If the latest firmware version is already installed in the DRIVECOD unit, you do not need to proceed with any new firmware installation. Current firmware version can be verified from the **SW Version** item in the **Alarm and status** page after connecting properly to the unit (see the previous page).
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 "4.2.3 Inputs / output + MODBUS RS-232 service port" section on page 42;
2. make sure that the DRIVECOD unit you need to update is the only node connected to the personal computer;
3. connect to the unit, go online and then enter the **Programming firmware** page;
4. when you switch on the power supply, if user program is not present in the flash memory, you are not able to connect to the unit through the **Serial configuration** page; when this happens you need to enter directly the **Programming firmware** page; after the attempt to connect has failed the **FIRMWARE** button 

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

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

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

For instance: RD12A_PL_V2_H2S1.BIN, where:

- RD12A = DRIVECOD unit model;
- PL = bus interface of the DRIVECOD unit (CB = CANopen; EC = EtherCAT; EP = EtherNet/IP; MB = MODBUS RTU; MT = MODBUS TCP; PB = Profibus; PL = POWERLINK; PT = Profinet);
- V2 = file / firmware version;
- H2 = hardware version;
- S1 = firmware version.

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

WARNING

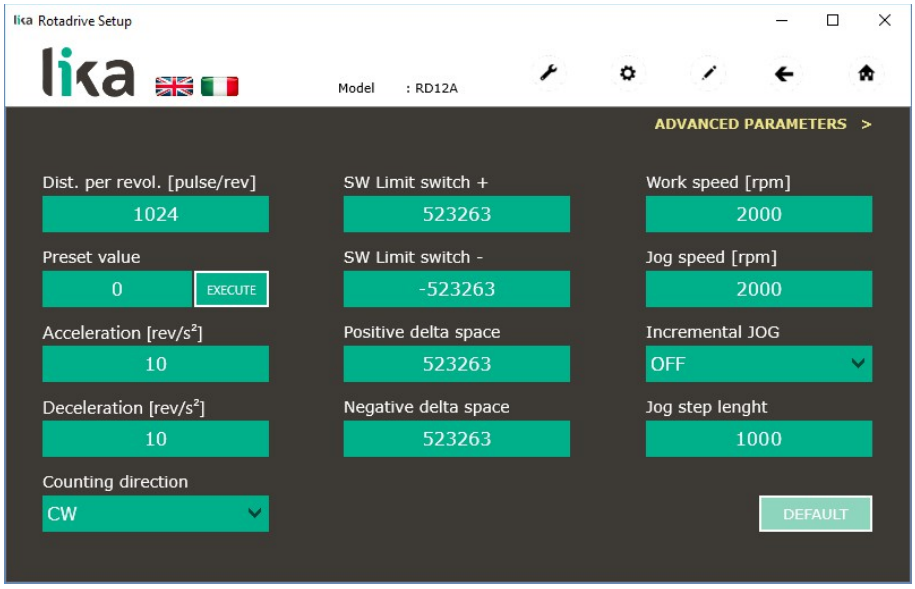
Do not exit the **Programming firmware** page during installation, the process will be aborted!

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

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

9.8 "Parameters" page

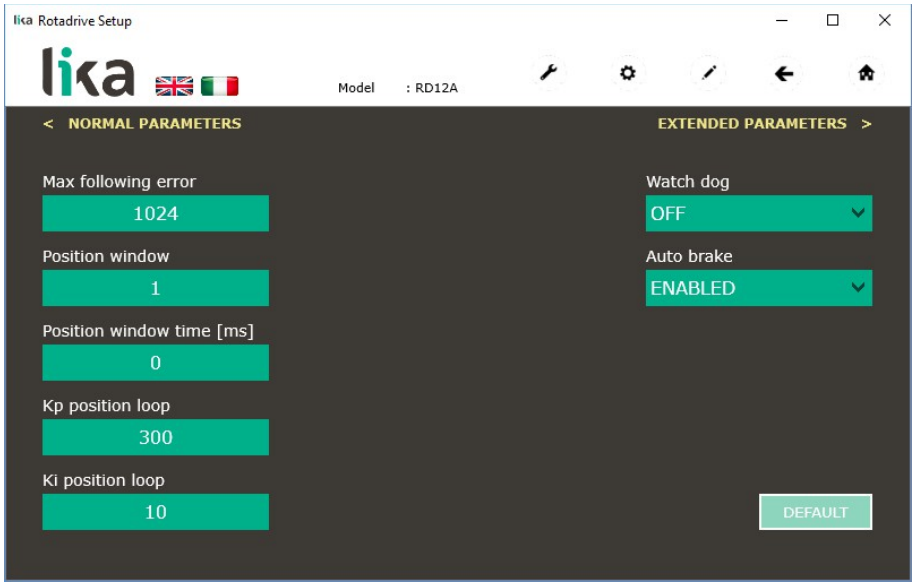
Press the **SETUP** button in the toolbar to configure the device and set the parameters. The page below will appear.



In this page the list of the "normal parameters" available to set the RD1xA positioning units is displayed. Parameters in the page need to be set more usually when you configure and test a new program of the actuator.

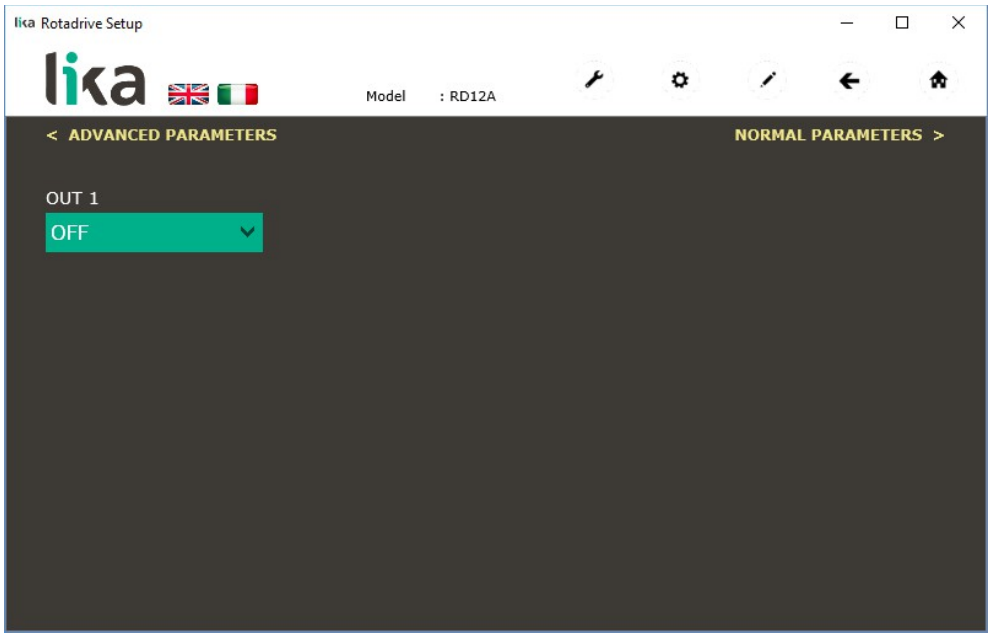
ADVANCED PARAMETERS > ADVANCED PARAMETERS >

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



EXTENDED PARAMETERS EXTENDED PARAMETERS >

Press the **EXTENDED PARAMETERS >** button EXTENDED PARAMETERS > to enter the page where the outputs can be enabled. For more information please refer to the "6.3 Digital inputs and output" section on page 85.



NORMAL PARAMETERS NORMAL PARAMETERS

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

For detailed information on the function and the setting of the parameters refer to the "9.14.1 Holding Register parameters" section on page 204.

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

DEFAULT DEFAULT

When you need to load the default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) press the **DEFAULT** button DEFAULT. For any further information on loading the default parameters refer to the **Load default**

parameters variable on page 214. The complete list of the machine data parameters and the relevant default values as set by Lika Electronic are available on page 237.

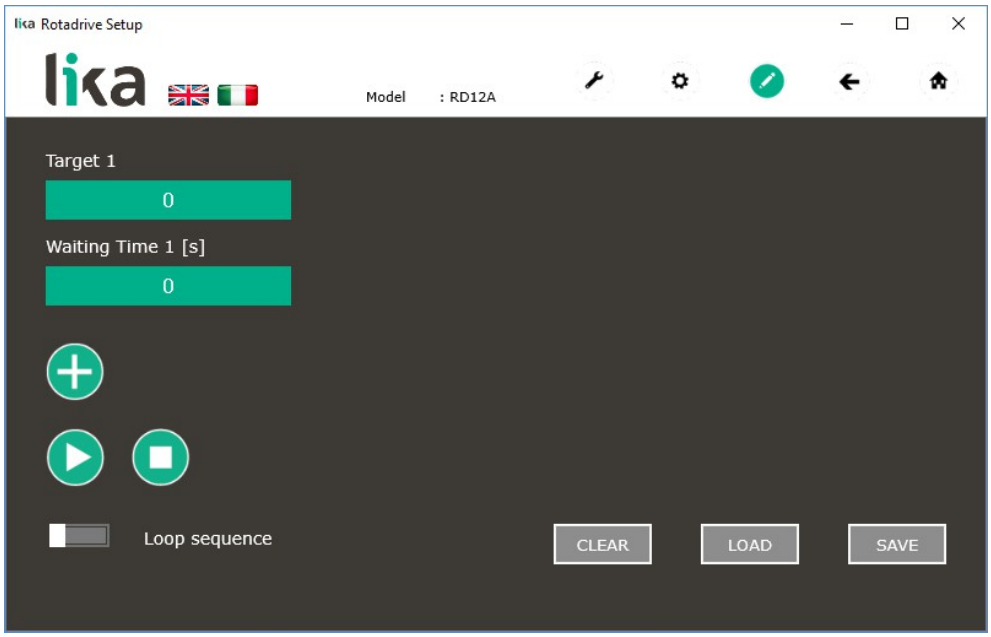
In the **Parameters** pages the following functions are also available.

EXECUTE

The **EXECUTE** button is placed next to the **Preset** item in the first **Parameters** page. When you enter a new value next to the **Preset** item, the Preset is saved but not activated. Press the **EXECUTE** button to activate the Preset value for the current position of the actuator's shaft. For any further information on activating the Preset value refer to the bit 11 **Setting the preset** in the Control Word [0x2A] register on page 215. Refer also to the Preset [0x12-0x13] registers on page 210.

9.9 "Program" page

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

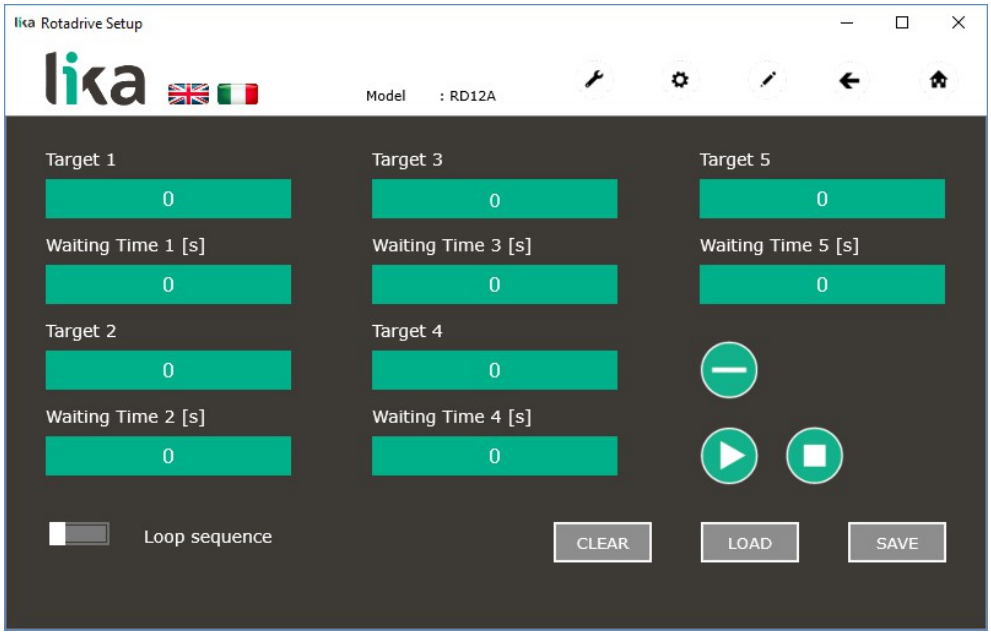


The functions available in the **Program** page allow the operator to create and save work programs for the RD1xA unit. In this way it is possible to test the operation of the actuator (speed, acceleration, deceleration, etc.) by setting up to five targets and commanding their execution.

Target 1 ... 5

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

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

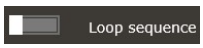


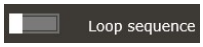
You can press the **MINUS** button to delete the last target.

Waiting Time 1 [s]

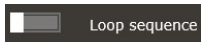
Next to the **Waiting Time 1 [s]** item you must set the interval between the first step (commanded movement) and the second. Press the **ENTER** key in the keyboard to confirm.

Loop sequence

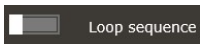


The **Loop sequence** slider 

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

If the **Loop sequence** slider 

is activated, when you press the **START** button , the device starts moving in order to reach the first commanded position **Target 1**; a green light appears next to the field and a green progress bar is shown at the top of the page; as soon as the commanded position set next to the **Target 1** item is reached, the device stops. After the set interval **Waiting Time 1 [s]** has expired, a green light appears next to the **Target 2** item and the RD1xA unit restarts running in order to reach the second commanded position **Target 2**; again a green progress bar is shown at the top of the page; and so on, from the first to the fifth commanded position (if

enabled) and then again from the first to the fifth commanded position without interruption, until you press the **STOP** button .
If the **Loop sequence** slider 

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

The following buttons are available in the page:

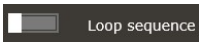
PLUS

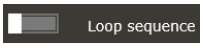
Press the **PLUS** button to add more target positions for the actuator to reach.

MINUS

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

START

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

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

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

STOP

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

CLEAR 

Press the **CLEAR** button 

to zero-set the counter designed to count the steps during the execution of the running program: when the operator presses the **START** button the device will start running from step 1, i.e. in order to reach the first commanded position **Target 1**, whatever the position reached before the counter was zero-set.

LOAD 

Press the **LOAD** button 

to load a work program that has been saved previously. Once you press the button, the **Open** dialog box appears on the screen: the operator must open the folder where the previously saved .dat file is

stored, then select it and finally confirm the choice by pressing the **OPEN** button, the dialog box closes and the work values are loaded automatically.

SAVE 

Press the **SAVE** button 

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

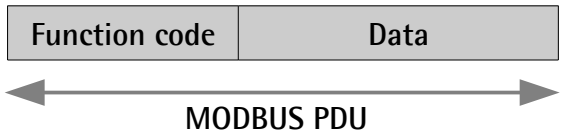
9.10 MODBUS Master / Slaves protocol principle

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.

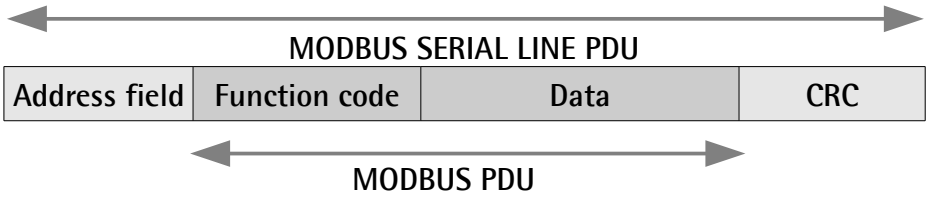
9.11 MODBUS frame description

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 "9.10 MODBUS Master / Slaves protocol principle" section on page 190), 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 "9.13 Function codes" section on page 194.
- **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 "9.13 Function codes" section on page 194).
- **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".

9.12 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.

9.12.1 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

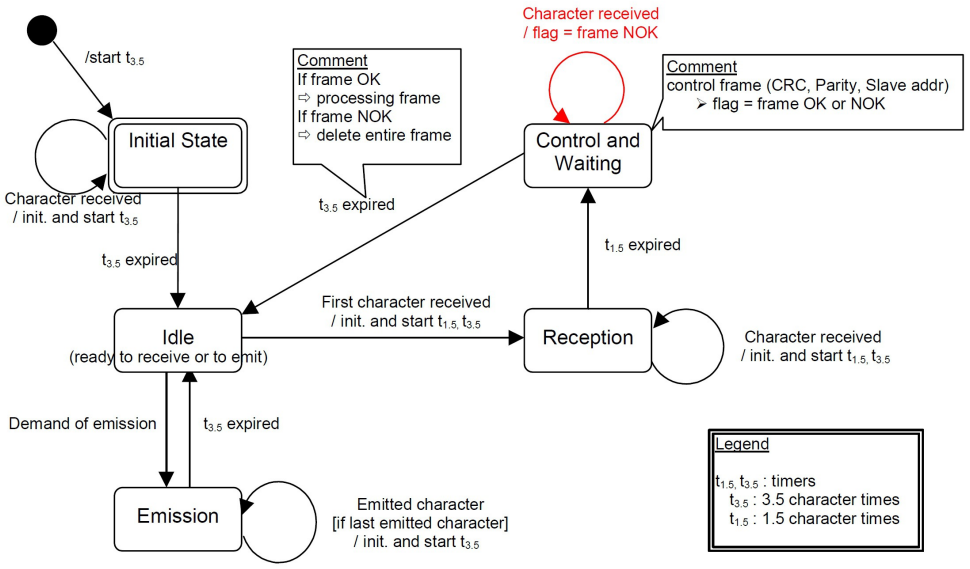
* When "No parity" is activated, the parity bit is replaced by a stop bit.

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.

9.13 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.

9.13.1 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) ro
This function code is used to READ the contents of a contiguous block of holding registers in a remote device; in other words, it allows to read the values set in a group of work parameters placed in order. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore registers numbered 1-16 are addressed as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).
For the complete list of holding registers accessible using **03 Read Holding Registers** function code please refer to the "9.14.1 Holding Register parameters" section on page 204.

Request PDU

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

Response PDU

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

*N = Quantity of registers

Exception Response PDU

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

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

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

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

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

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

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

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

Response PDU (in hexadecimal format)
[01][03][04][00][0A][00][0A][5A][36]

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

04 Read Input Register

FC = 04 (Hex = 0x04)

This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some results values and state / alarm messages in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).
For the complete list of input registers accessible using **04 Read Input Register** function code please refer to the "9.14.2 Input Register parameters" section on page 218.

Request PDU

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

Response PDU

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

*N = Quantity of registers

Exception Response PDU

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

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

Request	
Field name	(Hex)
Function	**04**
Starting address Hi	**00**
Starting address Lo	**02**
Quantity of Input Reg. Hi	**00**
Quantity of Input Reg. Lo	**02**

Response	
Field name	(Hex)
Function	**04**
Byte count	**04**
Register 3 value Hi	**00**
Register 3 value Lo	**00**
Register 4 value Hi	**2F**
Register 4 value Lo	**F0**

As you can see in the table, the **Current position [0x02-0x03]** parameter (input registers 3 and 4) contains the value 00 00 2F F0 hex, i.e. 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 "9.14.1 Holding Register parameters" section on page 204.

Request PDU

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

Response PDU

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

Exception Response PDU

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

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

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

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

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

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

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

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

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

16 Write Multiple Registers

FC = 16 (Hex = 0x10)

This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
The values to be written are specified in the request data field. Data is packed as two bytes per register.
The normal response returns the function code, starting address and quantity of written registers.
For the complete list of registers accessible using **16 Write Multiple Registers** function code please refer to the "9.14.1 Holding Register parameters" section on page 204.

Request PDU

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

*N = Quantity of registers

Response PDU

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

Exception Response PDU

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

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

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

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

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 "9.16 Programming examples" section on page 231.

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).

9.14 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

9.14.1 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 "9.13.1 Implemented function codes" section on page 195.

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 212.
Should the power supply be turned off all data that has not been saved previously will be lost!

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

Distance per revolution [0x00]

[Register 1, Unsigned16, rw]
This parameter sets the number of pulses per each complete revolution of the shaft. It is useful to relate the revolution of the shaft and a linear measurement. For example: the unit is joined to a warm screw having 5 mm (0.196") pitch; by setting Distance per revolution [0x00] = 500, at each shaft revolution the system performs a 5 mm (0.196") pitch with one-hundredth of a millimetre resolution.
Default = 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 86 and relevant parameters.
Please note that the parameters listed hereafter are closely related to the Distance per revolution [0x00] parameter; hence when you change the value in Distance per revolution [0x00] also the value in each of them necessarily changes. They are: Position window [0x01], Max following error [0x03-0x04], Positive delta [0x09-0x0A], Negative delta [0x0B-0x0C], Target position [0x2B-0x2C], Current position [0x02-0x03] and Position following error [0x05-0x06]. See also on page 209.

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 84.
Default = 1 (min. = 0, max. = 65535)

Position window time [0x02]

[Register 3, Unsigned16, rw]
It represents the time for which the axis has to be within the tolerance window limits set in the **Position window [0x01]** parameter before the state is signalled through the **Axis in position** status bit of the **Status word [0x01]**. The parameter is expressed in milliseconds (ms). See also the "Positioning: position and speed control" section on page 84.
Default = 0 (min. = 0, max. = 10000)

Max following error [0x03-0x04]

[Registers 4-5, Unsigned32, rw]
This parameter defines the maximum allowable difference between the real position and the theoretical position of the device. If the device detects a value higher than the one set in this parameter, the **Following error** alarm is triggered and the unit stops. The parameter is expressed in pulses.
Default = 1024 (min. = 0, max. = 65535)

Kp position loop [0x05]

[Register 6, Unsigned16, rw]
This parameter contains the proportional gain used by the PI controller for the position loop. The value has been optimized by Lika Electronic according to the technical characteristics of the device.
Default = 300 (min. = 0, max. = 1000)

Ki position loop [0x06]

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

Acceleration [0x07]

[Register 8, Unsigned16, rw]
This parameter defines the acceleration value that has to be used by the motor when reaching both the **Jog speed [0x0D]** and the **Work speed [0x0E]**. The parameter is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 83.
Default = 10 (min. = 1, max. = 500)

Deceleration [0x08]

[Register 9, Unsigned16, rw]
This parameter defines the deceleration value that has to be used by the motor when stopping. The parameter is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 83.
Default = 10 (min. = 1, max. = 500)

Positive delta [0x09-0x0A]

[Registers 10-11, Unsigned32, rw]
This value is used to calculate the maximum forward (positive) limit the device is allowed to reach starting from the preset value. Should it happen that the maximum forward limit is reached, a signalling is activated through the **SW limit switch +** status bit of the **Status word [0x01]** (the bit is forced high). The parameter is expressed in pulses.
SW limit switch + = **Preset [0x12-0x13]** + **Positive delta [0x09-0x0A]**.
For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.
Default = 523 263 (min. = 0, max. = 523 263)

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

EXAMPLE
When **Distance per revolution [0x00]** = 1,024 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Positive delta [0x09-0x0A]** is:
(1,024 steps per revolution * 512 revolutions) - 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 523 263
When **Distance per revolution [0x00]** = 256 and **Preset [0x12-0x13]** = 0, the maximum acceptable value for **Positive delta [0x09-0x0A]** is:
(256 steps per revolution * 512 revolutions) - 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 130 815
See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

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 86.

Negative delta [0x0B-0x0C]

[Registers 12-13, Unsigned32, rw]

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

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

For further information please refer to the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

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

WARNING

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

EXAMPLE

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

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

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

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

See further examples in the "6.4 Distance per revolution, Preset, Max delta pos / Positive delta and Max delta neg / Negative delta" section on page 86.

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 86.

Jog speed [0x0D]

[Register 14, Unsigned16, rw]
This parameter contains the maximum speed the motor is allowed to reach when using the **Jog +** and **Jog -** functions (see the **Control Word [0x2A]** parameter). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 83.
Default = 2000 (min. = 1, max. = 3000)

NOTE

Please note that this is the speed of the motor, not the speed of the output shaft after the reduction gears.
The speed at output will be as follows:
Motor speed = 2000 rpm
Speed at output: T12 = 166 rpm
T24 = 83 rpm
T48 = 41 rpm
T92 = 21 rpm

Work speed [0x0E]

[Register 15, Unsigned16, rw]
This parameter contains the maximum speed the motor is allowed to reach in automatic work mode (movements are controlled using the **Start** and **Stop** commands -see the **Control Word [0x2A]** parameter- and are performed in order to reach the position set in **Target position [0x2B-0x2C]**). The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 84.
Default = 2000 (min. = 1, max. = 3000)

NOTE

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

The speed at output will be as follows:
Motor speed = 2000 rpm
Speed at output: T12 = 166 rpm
T24 = 83 rpm
T48 = 41 rpm
T92 = 21 rpm

Code sequence [0x0F]

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

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

Offset [0x10-0x11]

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

Preset [0x12-0x13]

[Registers 19-20, Integer32, rw]
Use this parameter to set the Preset value. The Preset function is meant to assign a desired value to a physical position of the axis. The chosen physical position will get the value set next to this item and all the previous and the following positions will get a value according to it. The preset value will be set for the position of the axis in the moment when the value is entered. The preset value is activated when the bit 11 **Setting the preset** in the **Control Word [0x2A]** register is switched from logic level low ("0") to logic level high ("1").
Default = 0 (min. = -1 048 576, max. = +1 048 576)

NOTE
We suggest activating the preset when the actuator is in stop. See the **Setting the preset** command on page 215.

WARNING
A new value has to be set in the **Preset [0x12-0x13]** registers every time the **Distance per revolution [0x00]** value is changed. After having entered a new value in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in the **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 86.

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 83.

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 83.

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

sent simultaneously, the device will not move or, if already moving, it will stop its movement.

Stop

bit 2

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

Alarm reset

bit 3

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

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

Incremental jog

bit 4

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

bit 5

Not used.

Start

bit 6

When it is set to "=1" the device moves in order to reach the set target position (see **Target position [0x2B-0x2C]** on page 216). For a complete description of the position control see on page 84.

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 237.

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 210.

Axis torque

bit 12 — This function is available only in the RD1A version (model without brake); in the RD12A version (model fitted with brake) the bit 12 is not used. When the axis has reached the commanded position, it maintains the torque.
If set to "=1", when the axis is in position, the PWM is kept active (the torque is applied to the axis when the position is reached).
If set to "=0", 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 85.

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

Brake disabled

bit 14 — This function is available only in the RD12A version (model fitted with brake); in the RD1A version (model without brake) the bit 14 is not used. RD12A model is fitted with a brake designed to activate as soon as the motor comes to a stop in order to prevent it from moving. Setting this bit to "=1" causes the brake to be disabled and not operational; setting this bit to "=0" causes the brake to be enabled and managed automatically by the system.
Please note that you can disengage the brake only when no alarm is active.

bit 15 — Not used.

WARNING

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

Target position [0x2B-0x2C]

[Registers 44-45, Integer32, rw]

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

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

For more information refer also to the "Positioning: position and speed control" section on page 84.

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

NOTE

Position override function

It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the Target position [0x2B-0x2C] registers. See also on page 84.

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!

9.14.2 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 "9.13.1 Implemented function codes" section on page 195.

Alarms register [0x00]

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

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

The available alarm error codes are listed hereafter:

Byte 0

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

Flash memory error
bit 1 — Internal error, it cannot be restored.

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

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

Axis not synchronized
bit 4 — Internal error, it cannot be restored.

Target not valid
bit 5 — The set target position is over the maximum travel limits.

Emergency

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

Overcurrent

bit 7 — Motor overcurrent.

Byte 1

Electronics Overtemperature

bit 8 — The temperature of the MOSFETs detected by an internal probe is exceeding the maximum ratings (see **Temperature value [0x07]** on page 224). 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 224). 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 speed ramp or evaluate attentively the characteristics of the 24V power supply pack (capacitor module).

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

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

Status word [0x01]

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

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

Byte 0

Axis in position

bit 0 — The value is "=1" when the device reaches and keeps the set position (**Target position [0x2B-0x2C]**) for the time set next to the **Position window time [0x02]** register. It is kept active until the position error is lower than **Position window [0x01]**. For further information please refer to the "Positioning: position and speed control" section on page 84.

bit 1 — Not used.

Drive enabled

bit 2

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

SW limit switch +

bit 3

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

SW limit switch -

bit 4

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

Alarm

bit 5

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

Axis running

bit 6

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

Executing a command

bit 7

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

Byte 1

Target position reached

bit 8

The value is "=1" when the device reaches the target position set next to the **Target position [0x2B-0x2C]** item (it is within the limits set next to the **Position window [0x01]**). The bit is kept active until a new **Target position [0x2B-0x2C]** value or the **Alarm reset** command are sent. For more information refer also to the "Positioning: position and speed control" section on page 84.

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 "4.5.2 Screw plug for internal access (Figure 4 and Figure 6)" section on page 45 and the "4.8.1 JOG + and JOG – buttons (Figure 9)" section on page 51.

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 "4.5.2 Screw plug for internal access (Figure 4 and Figure 6)" section on page 45 and the "4.8.1 JOG + and JOG – buttons (Figure 9)" section on page 51.

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 "4.5.2 Screw plug for internal access (Figure 4 and Figure 6)" section on page 45 and the "4.8.2 PRESET button (Figure 9)" section on page 52.

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 85.

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 85.

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 85.

IN 3 = 0 input 3 low (not active)
IN 3 = 1 input 3 high (active)

Current position [0x02-0x03]

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

Current velocity [0x04]

[Register 5, Integer16, ro]
Speed of the device expressed in revolutions per minute [rpm], updated at every second. This is the speed of the motor, not the speed of the output shaft after the reduction gears. The speed at output will be as follows:
Motor speed = 2000 rpm
Speed at output: T12 = 166 rpm
T24 = 83 rpm
T48 = 41 rpm
T92 = 21 rpm

Position following error [0x05-0x06]

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

Temperature value [0x07]

[Register 8, Integer16, ro]

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

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

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

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

Wrong parameters list [0x08-0x09]

[Registers 9-10, Unsigned32, ro]

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

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

Bit	Parameter
0	Not used
1	Distance per revolution [0x00]
2	Acceleration [0x07]
3	Deceleration [0x08]
4	Positive delta [0x09-0x0A]
5	Negative delta [0x0B-0x0C]
6	Jog speed [0x0D]
7	Work speed [0x0E]
8	Code sequence [0x0F]
9	Preset [0x12-0x13]
10	Jog step length [0x14]
11	Kp position loop [0x05]
12	Ki position loop [0x06]

13	Position window time [0x02]
14	Max following error [0x03-0x04]
15	Not used

Motor voltage [0x0A]

[Register 11, Unsigned16, ro]
It shows the motor voltage expressed in millivolts (mV).

Current value [0x0B]

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

Hall [0x0C]

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

Duty cycle [0x0D]

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

DIP switch baud rate [0x0E]

[Register 15, Unsigned16, ro]
This is meant to show the data transmission rate (baud rate) of the serial port the RD1xA unit is equipped with; the data transmission rate has to be set through the provided DIP switch. In this model the baud rate DIP switch has fixed value and is not accessible to the user.

DIP switch node ID [0x0F]

[Register 16, Unsigned16, ro]
This is meant to show the node address set in the RD1xA unit; the node address has to be set through the provided DIP switch. In this model the node ID DIP switch has fixed value and is not accessible to the user.

SW Version [0x10]

[Register 17, Unsigned16, ro]
This is meant to show the software version of the DRIVECOD unit.
The meaning of the 16 bits in the register is as follows:

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

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

HW Version [0x11]

[Register 18, Unsigned16, ro]
This is meant to show the hardware version and model of the DRIVECOD unit.
The meaning of the 16 bits in the register is as follows:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DRIVECOD model								Brake	Gear ratio			Hardware version			

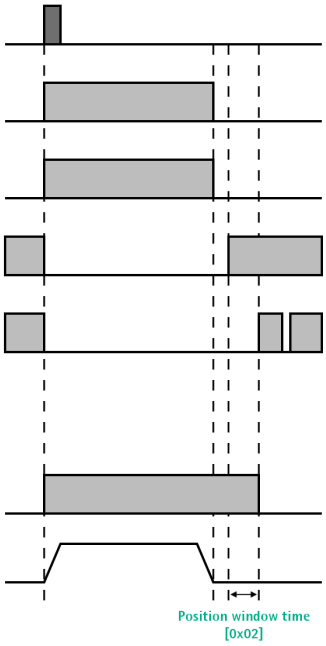
where:

00 ... 03	= hardware version
04 ... 06	= gear ratio (1 = T12; 2 = T24; 3 = T48; 4 = T92)
07	= brake: 0 = RD1A model without brake; 1 = RD12A model with brake
08 ... 15	= RD1xA model equipped with the following interface: 0x30 = MODBUS RTU; 0x31 = Profibus; 0x32 = CANopen; 0x33 = POWERLINK; 0x34 = EtherCAT; 0x35 = MODBUS TCP; 0x36 = EtherNet/IP; 0x37 = Profinet

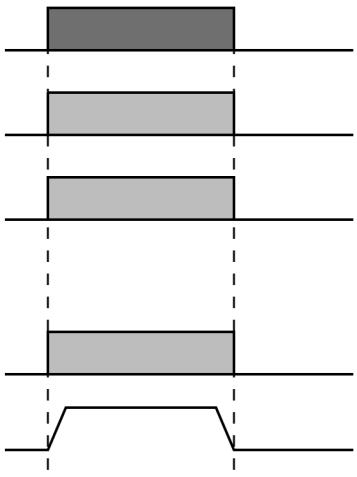
Value 30 B1 hex in hexadecimal notation corresponds to the binary representation 0011 0000 1011 0001 and has to be interpreted as follows: RD1xA model with MODBUS RTU interface, T48 reduction gear, equipped with brake, hardware version 1.

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

Positioning	RD1xA
Start (bit 6 - Control Word [0x2A]) **Executing a command** (bit 7 - Status word [0x01]) **Axis running** (bit 6 - Status word [0x01]) **Target position reached** (bit 8 - Status word [0x01]) **Axis in position** (bit 0 - Status word [0x01]) **Drive enabled** (bit 2 - Status word [0x01]: PWM is active) Theoretical velocity	

EXAMPLE 2

Jog +	RD1xA
Jog + (bit 0 - Control Word [0x2A]) **Executing a command** (bit 7 - Status word [0x01]) **Axis running** (bit 6 - Status word [0x01]) **Drive enabled** (bit 2 - Status word [0x01]: PWM is active) Theoretical velocity	

9.15 Exception codes

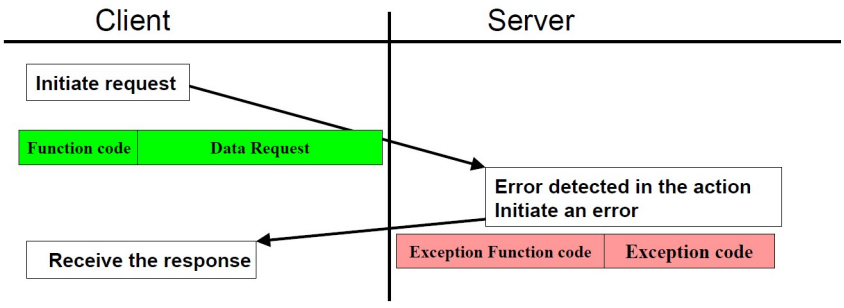
When a Client device sends a request to a Server device it expects a normal response. One of four possible events can occur from the Master's query.

- 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, 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.



NOTE
Please note that here follows the list of the exception codes indicated by MODBUS but not necessarily supported by the manufacturer.

MODBUS Exception codes		
Code	**Name**	**Meaning**
01	ILLEGAL FUNCTION	The function code received in the query is not an allowable action for the server. This may be because the function code is only applicable to newer devices, and was not implemented in the unit selected. It could also indicate that the server is in the wrong state to process a request of this type, for example because it is not configured and is being asked to return register values.
02	ILLEGAL DATA ADDRESS	The data address received in the query is not an allowable address for the server. More specifically, the combination of reference number and transfer length is invalid. For a controller with 100 registers, the PDU addresses the first register as 0, and the last one as 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 4, then this request will successfully operate (address-wise at least) on registers 96, 97, 98, 99. If a request is submitted with a starting register address of 96 and a quantity of registers of 5, then this request will fail with Exception Code 0x02 "Illegal Data Address" since it attempts to operate on registers 96, 97, 98, 99 and 100, and there is no register with address 100.
03	ILLEGAL DATA VALUE	A value contained in the query data field is not an allowable value for server. This indicates a fault in the structure of the remainder of a complex request, such as that the implied length is incorrect. It specifically does NOT mean that a data item submitted for storage in a register has a value outside the expectation of the application program, since the MODBUS protocol is unaware of the significance of any particular value of any particular register.
04	SERVER DEVICE FAILURE	An unrecoverable error occurred while the server was attempting to perform the requested action.
05	ACKNOWLEDGE	Specialized use in conjunction with programming commands. The server has accepted the request and is processing it, but a long duration of time

		will be required to do so. This response is returned to prevent a timeout error from occurring in the client. The client can next issue a Poll Program Complete message to determine if processing is completed.
06	SERVER DEVICE BUSY	Specialized use in conjunction with programming commands. The server is engaged in processing a long-duration program command. The client should retransmit the message later when the server is free.
08	MEMORY PARITY ERROR	Specialized use in conjunction with function codes 20 and 21 and reference type 6, to indicate that the extended file area failed to pass a consistency check. The server attempted to read record file, but detected a parity error in the memory. The client can retry the request, but service may be required on the server device.
0A	GATEWAY PATH UNAVAILABLE	Specialized use in conjunction with gateways, indicates that the gateway was unable to allocate an internal communication path from the input port to the output port for processing the request. Usually means that the gateway is misconfigured or overloaded.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Specialized use in conjunction with gateways, indicates that no response was obtained from the target device. Usually means that the device is not present on the network.

For any information on the available exception codes and their meaning refer to the "MODBUS Exception Responses" section on page 47 of the "MODBUS Application Protocol Specification V1.1b3" document.

9.16 Programming examples

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

9.16.1 Using the 03 Read Holding Registers function code

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

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

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

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

9.16.2 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 218): **Machine data not valid** and **Emergency**.

9.16.3 Using the 06 Write Single Register function code

EXAMPLE 1

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

Request PDU

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

Response PDU

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

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

EXAMPLE 2

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

Request PDU

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

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

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

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

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

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

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

9.16.4 Using the 16 Write Multiple Registers function code

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

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

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

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

10 Default parameters list

10.1 EtherNet/IP attributes of the Class 01h Identity Object

Parameters list	Default values		
01-01-01 Vendor ID	0299h = Lika Electronic		
01-01-02 Device type	002Bh = Generic Device		
01-01-03 Product code	3000h = RD1xA rotary actuator		
01-01-04 Revision	Device dependent		
01-01-06 Serial number	Device dependent		
01-01-07 Product name	RD1xA rotary actuator		

10.2 EtherNet/IP attributes of the Class 64h Application Object

Parameters list	Default values		
64-01-01 Control Word	0		
64-01-02 Target Position	0		
64-01-05 Distance per Revolution	1,024		
64-01-06 Position Tolerance	1		
64-01-07 Settling time	0		
64-01-08 Max following error	1,024		
64-01-09 Proportional gain	300		
64-01-0A Integral gain	10		
64-01-0B Acceleration	10		
64-01-0C Deceleration	10		
64-01-0D Max delta pos	523 263		
64-01-0E Max delta neg	523 263		
64-01-0F Jog speed	2,000		
64-01-10 Work speed	2,000		
64-01-11 Count direction	0		
64-01-12 Preset	0		
64-01-13 Step jog	1,000		

10.3 MODBUS registers

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		
Control Word [0x2A]	0		
Target position [0x2B-0x2C]	0		

This page intentionally left blank

Document release	Release date	Description	HW	SW	EDS file	Interface
1.0	20.02.2020	First issue	2.0	1.1	1.0	3.1

Dispose separately

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

Tel. +39 0445 806600
Fax +39 0445 806699

info@lika.biz • www.lika.biz