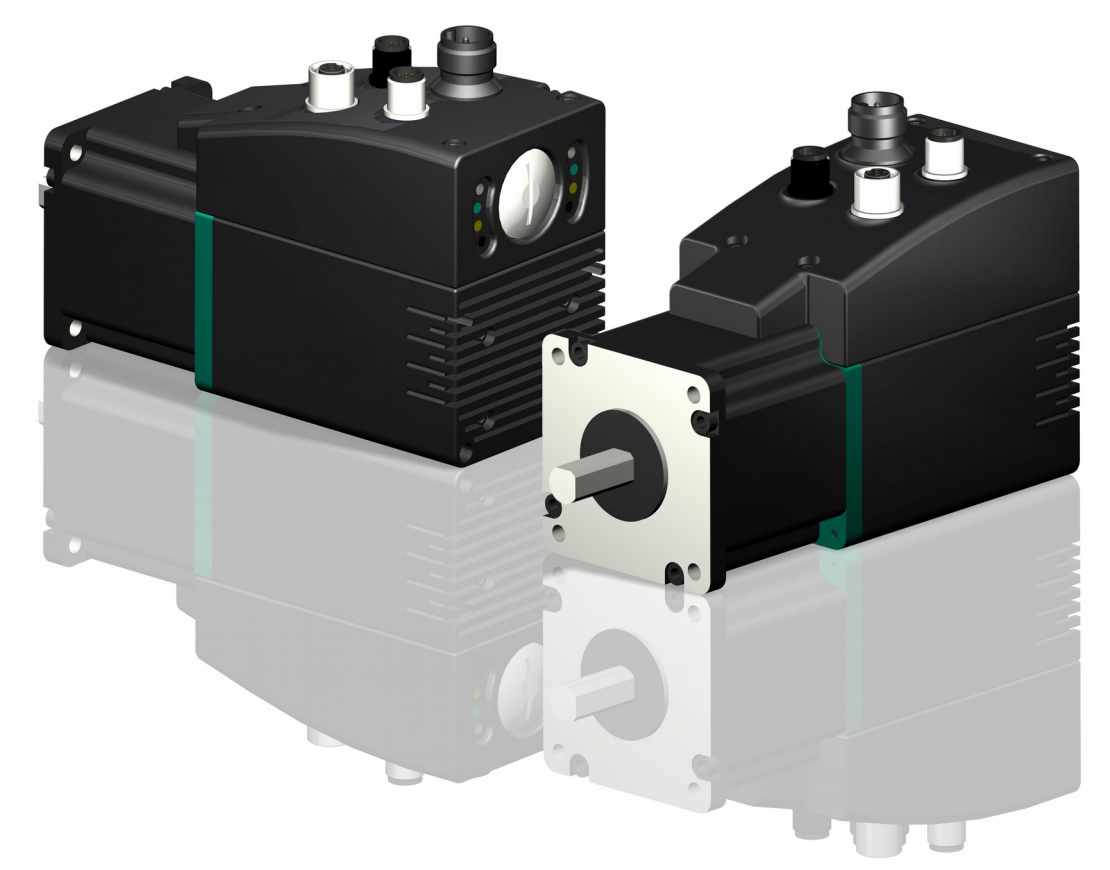
User's guide

RD6

RS-232 version



Smart encoders & actuators

lika

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 RD6 model with Profibus-DP interface**.

RD6 units are positioning devices which integrate into one system a brushless motor, a drive, a multiturn absolute encoder and a position controller. They are not equipped with gears. The 70 mm (2.76") size square flange and the 14 mm (0.55") shaft are designed to be coupled with planetary gearboxes available in the market. Thus the units can be easily integrated into custom applications to meet specific torque and speed requirements. RD6 are designed to drive positioning systems, change-over applications and linear guides. 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.

RD6 rotary actuators can be equipped with the following interfaces:

- RD6-x-xxx-**CB**-... = CANopen DS301 interface;
- RD6-x-xxx-**EC**-... = EtherCAT interface;
- RD6-x-xxx-**MB**-... = Modbus RTU (RS-485) interface;
- RD6-x-xxx-**PB**-... = Profibus-DP interface;
- RD6-x-xxx-**PL**-... = POWERLINK interface.

The present manual is specifically designed to describe the Profibus-DP 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 Profibus version configuration can be done using the same program through a **service RS-232 serial interface, in compliance with Modbus protocol**.

To make it easier to read the text, this guide can be divided into two main sections.
In the first section general information concerning the safety, the mechanical installation and the electrical connection as well as tips for setting up and running properly and efficiently the unit are provided.
In the second section, entitled **Profibus Interface**, both general and specific information is given on the Profibus-DP interface. In this section the interface features and the objects implemented in the unit are fully described.
In the third section, entitled **Modbus Interface**, both general and specific information is given on the Modbus interface. As previously stated, Profibus 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 Profibus network. In the **Modbus Interface** section the interface features and the registers implemented in the unit are fully described.

WARNING

When you install **Lika RD6** module, the value of each parameter is uploaded at power-on from the GSD file which has been loaded in the PLC. Thus any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message will be temporary: when you turn off the power supply, the set value is lost (except the preset value which is the only variable not included in the GSD file; or unless you set the preset value previously which causes all parameters values to be saved automatically, see on page 91) and the value saved in the PLC will be uploaded at next power-on (thus all values, even if previously saved because of a preset setting, will be overwritten anyway).

This unit also allows to install the **Lika RD6-no param** module. Using this module, the unit does not read the parameters from the PLC when the power is switched on (they are read from the flash memory) and allows the operator to save any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message on the non-volatile memory (by means of the bit 9 **Save parameters** in the **Control Word (Bytes 0 and 1)**). When **Lika RD6-no param** module is installed, it is NOT possible to read and change the values of the **configuration data** parameters in the **Device-specific parameters** page of the TIA **Properties** window (see the "7.1.4 Device view" section on page 52). Thus you are allowed to enter new parameter values only through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message or by using the Modbus interface.

Using the RS-232 Modbus service serial interface it is possible to alter and then save the parameter values; in this way altered values are available again at next power-on. But this is true only until the serial interface is kept active: as soon as the unit is connected to the Profibus network, data in the PLC will be uploaded automatically and therefore the values set through the serial interface will be overwritten.

On the other hand using Siemens TIA PORTAL it is possible to alter any value and then save it permanently in the PLC (except in the -no param module). To save values permanently the operator has to enter the **Device-specific parameters** page of TIA and then alter the desired item (see the "7.1.4 Device view" section on page 52). Values altered in the Control Table of TIA (see the "7.1.13 Control Table" section on page 61) are temporary only.

Glossary of Profibus terms

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

Address (Station)	IEC 61158-2: Medium attachment unit identification - unique number of a station connected to a segment (participant).
Address Space	Within PROFIBUS DP the maximum possible number of addressable network nodes per segment, e.g. 127.
Alarm	Notification of an abnormal or unexpected event within a system. Alarms in PROFIBUS DP require in addition to the standard diagnosis event mechanism within the cyclic data exchange a separate acyclic acknowledgement procedure between a host and a Slave application. Since DP-V1, "Device related diagnosis" is the basis for the "Alarm" and "Status" types of diagnosis events (GSD: "DPV1"=1). PROFIBUS DP defines the following alarm types: Diagnosis, Status, Process, Update, Pull and Plug Alarm. See "Device Related Diagnosis". The PNO maintains a Profile Guideline, Part3: Diagnosis, Alarms and Time Stamping, order no. 3.522.
Alert	Alert is a generic term for two different types of notifications within a PROFIBUS DP/PA network especially arranged but not exclusively for the process automation: • alarm; • event. Both alert types may be used with or without a user acknowledgement mechanism. The PNO maintains a PROFIBUS guideline "Time Stamp", order no. 2.192.
Application Profile	Within PROFIBUS a specified agreement within families of field devices on how to use the general PROFIBUS communication platform and its subsystems (e.g. device integration via GSD, EDD, FDT/DTM and Communication Function Blocks). Communication profiles are not a part of the PROFIBUS DP application profiles. See "Profile".
Baud rate (Data Rate)	Other common terms are "data transfer rate" and "transmission rate". Within PROFIBUS DP this is the amount of data transferred across a fieldbus segment per second. A data rate is measured in units of bits per second ("b/s" or "bps"), or baud.
Bus Cycle	The period of time the bus Master needs to poll every participant (Slave) once. More bus Masters can be activated by using the token principle which consequently prolong the bus cycle.

Class	See "DP Master", "DP Master Class 1 (DPM1)" and "DP Master Class 2 (DPM2)".
Class 1 encoder	Encoder class must be set when you configure the device. Mandatory Class 1 provides the basic functions of the device and can be used for: • sending the position value (see **Position value** parameter); • changing the counting direction (see **Code sequence** parameter); • setting the preset value (see **Preset value** parameter); • acquiring reduced diagnostic information (see **Diagnostic type** parameter = "16 bytes fixed (6+10)").
Class 2 (+VEL) encoder	Encoder class must be set when you configure the device. Class 2 (+VEL) provides all the Class 1 and Class 2 functions and additional velocity-related functions: • transmission of the velocity value (see **Position and velocity values** parameter); • setting of the velocity measuring unit (see **Velocity measure unit** parameter).
Class 2 encoder	Encoder class must be set when you configure the device. Class 2 provides all the Class 1 functions and additional advanced functions such as: • scaling function (see **Scaling function control**, **Counts per revolution** and **Total resolution** parameters); • extended diagnostic information (see **Diagnostic type** parameter = "16 bytes (6+10)" or "63 bytes (6+57)").
Communication Function Block (Comm FB)	A basic function block defined for PROFIBUS DP and supplied by the PLC manufacturer for the standardized access of user programs to field devices. The standardization is based on IEC 61131-3. The PNO maintains a guideline "PROFIBUS Communication and Proxy Function Blocks acc. to IEC 61131-3", order no. 2.182.
Communication Parameter	Communication parameters are parameters, which adjust the communication protocol function to the current net configuration. Communication parameters exist for all phases of the communication protocols. Examples are bus address, token rotation time, idle time. See "Slave parametrization" and "Device parametrization".
Communication Profile	IEC 61158 comprises a summary of layer stacks of several different fieldbusses. IEC 61784 defines the useful combinations of these stacks via communication profiles CPF3/1 up to CPF3/3 (PROFINET). One of these is PROFIBUS DP. Within this communication profile three different physical profiles are defined: • RS 485 (RS 485-IS); • MBP-IS (MBP-LP, MBP);

	• Fibre Optics.
Cyclic Data Exchange	IEC 61158-3: Term used to describe events which repeat in a regular and repetitive manner. The MS0 services of PROFIBUS DP are based on cyclic data exchange. See "State machine".
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 Rate (Baud rate)	Other common terms are "data transfer rate" and "transmission rate". Within PROFIBUS DP this is the amount of data transferred across a fieldbus segment per second. A data rate is measured in units of bits per second ("b/s" or "bps"), or baud.
Decentralized Peripherals (DP)	The term "Decentralized Peripherals" and the acronym "DP" stand for the simple, fast, cyclic and deterministic I/O data exchange between a bus Master and its assigned Slave devices. The corresponding PROFIBUS communication protocol is called PROFIBUS DP.
Device Identifier	Ident number: The primary device identification is an ident number of data type Unsigned16. This number is unique and assigned by the PNO business office upon application. It is stored within the device and defined in the corresponding GSD file via keyword. In addition it is part of the GSD file name. At runtime the ident number is used within: • the set Slave address procedure; • the parametrization telegram (octet 5 + 6); • the standard part of a diagnosis message (octet 5 + 6). The ident number explicitly cannot be retrieved from a device. Its main purpose is to make sure that a GSD file and configuration/parametrization data between Master Class 1 and its Slave are matching. The PNO maintains a technical guideline "Specification for PROFIBUS device description and device integration, Volume 1: GSD", Version 5.0, order no. 2.122. For a secondary identification possibility see the identification & maintenance functions (I&M). See "Ident Number".
Device Parametrization	The device parametrization within PROFIBUS DP consists of three phases. The first phase takes place during start-up of the communication system and provides basic communication parametrization and simple additional device parameters. Both are defined within the GSD file of a device, stored within a Master Class 1 after configuration in an engineering tool, and transmitted to the Slave at start-up time. Most of the automation cases in factory automation are covered by this method. More complex devices such as drives, laser scanners, scales, robots, transmitters, etc. require further individual parametrization before final production start. This is done in a second phase. In process automation certain device

	parameters such as value limits, value range, gain, etc. need to be adjusted even at run-time. For this second and third phase PROFIBUS DP provides two ways to accomplish the task: DTM/FDT and EDD. See "Slave parametrization" and "Communication parameter".
Device Profile	See "Profile".
DP Master	IEC 61158-5: Within PROFIBUS DP a fieldbus node that can be either Master Class 1 or Master Class 2. A Master Class 1 is a controlling device which controls several DP Slaves (field devices). NOTE: This is usually hosted by a programmable controller or a process controller. A Master Class 2 is a controlling device which manages configuration data (parameter sets) and diagnosis data of a DP Master Class 1, and that additionally can perform all communication capabilities of a DP Master Class 1.
DP Master Class1 (DPM1)	IEC 61158-5: A controlling device which controls several DP-Slaves (field devices). Usually programmable (logic) controllers or process control systems are hosts for Master Class 1.
DP Master Class2 (DPM2)	IEC 61158-5: A controlling device which manages configuration data (parameter sets) and diagnosis data of a DP-Master (Class 1). Additionally the DP-Master (Class 2) can perform all communication capabilities of a DP-Master (Class 1). Usually personal computers are hosts for DP Master Class 2 for programming, parametrizing, diagnosing and monitoring purposes.
DP Slave	IEC 61158-5: A field device that is assigned to one DP Master Class 1 as a provider for cyclic I/O data exchange. In addition acyclic functions and alarms could be supported.
Event	Within PROFIBUS DP/PA this is a signal or I/O data or process value within a certain field device at that point in time where a trigger condition arises. The values are associated with a time stamp and stored in a buffer. The time-stamped sample values are used to archive and visualize significant changes over the course of the production process. Such an event mechanism does not prevent from the cyclic transmission of these signals. A separate event alarm is requesting the transfer of the events to the main system.
Frame	A single set of data transmission from a device.
General Station Description (GSD)	A GSD is an electronically readable ASCII text file and contains both general and device-specific parameters for communication and network configuration. By means of keywords, a configuration tool allows to: • read device information (manufacturer, type, versions, bitmaps, etc.); • read texts for comfortable and easy to use configuration; • select transmission rates;

	• select modules and their I/O data length (configuration identifier); • read texts to assign diagnosis IDs to HMI displays; • select supported services (freeze, sync, etc.); from the GSD for the configuration of the device. A GSD replaces the previously conventional manuals or data sheets and thus already supports plausibility checks during the configuration phase. Distinction is made between a device GSD (for an individual device only) and a profile GSD, which may be used for devices that comply exactly with a profile such as a "PA device". GSDs for different languages may be provided in separate files with corresponding file extensions (*.gse for English, *.gsg for German, etc.) or altogether in one file (*.gsd). The device manufacturers are responsible for the scope and quality of the GSD of their devices.
Ident Number	See "Device Identifier". Notes: • the ident number is necessary for all DP devices except for Master Class 2; • the same ident number may be used for modular devices as long as the device can be described in the GSD file as a modular device.
Identifier	In general: a symbol that establishes the identity of the one bearing it. Within this context here it represents an absolute value of a parameter such as a physical address. It is intended for systematic and performance handling capabilities within computer systems, e.g. sorting, consistency checking, physical localization and alike. Usually an absolute value is associated with a logical value to represent the particular deployment of the identifier. Typical abbreviation for identifier is ID. IEC 61131-3: A combination of letters, numbers and underline characters, which begins with a letter or underline and which names a language element. Some of the major identifiers within PROFIBUS DP are: - Data type numeric identifier; - Configuration identifier (Cfg); - Device identifier (ident number); - Manufacturer identifier (MANUFACTURER ID); - Profile ident number (PROFILE ID).
Index	IEC 61158-5: Address of an object within an application process. The permitted range in PROFIBUS DP is 0 - 255. Indexes are used to address records of data (parameters, variables, state information, commands, etc.) within modules of a field device.
PDU (Protocol Data Unit)	A packet of data passed across a network via telegrams. The term implies a specific layer of the OSI seven layer model and a specific protocol. Each layer has its own PDU that is extended subsequently from the physical layer up to the application layer:

	• Physical layer protocol data unit (PhPDU); • Data link protocol data unit (DLPDU); • Application protocol data unit (APDU).
PI	The *PROFIBUS Nutzerorganisation e.V.* (PROFIBUS User Organisation, or PNO) was created in 1989. This group was composed mainly of manufacturers and users from Europe. In 1992, the first regional PROFIBUS organization was founded (PROFIBUS Schweiz in Switzerland). In the following years, additional Regional PROFIBUS & PROFINET Associations (RPAs) were added. In 1995, all the RPAs joined together under the international umbrella association PROFIBUS & PROFINET International (PI). Today, PROFIBUS is represented by 25 RPAs around the world (including PNO) with over 1400 members, including most if not all major automation vendors and service suppliers, along with many end users.
PNO	The *PROFIBUS Nutzerorganisation e.V.* (PROFIBUS User Organisation, or PNO) was created in 1989. This group was composed mainly of manufacturers and users from Europe. In 1992, the first regional PROFIBUS organization was founded (PROFIBUS Schweiz in Switzerland). In the following years, additional Regional PROFIBUS & PROFINET Associations (RPAs) were added. In 1995, all the RPAs joined together under the international umbrella association PROFIBUS & PROFINET International (PI). Today, PROFIBUS is represented by 25 RPAs around the world (including PNO) with over 1400 members, including most if not all major automation vendors and service suppliers, along with many end users.
PROFIBUS	PROcess FIeldBUS. PROFIBUS is a manufacturer independent fieldbus standard for applications in manufacturing, process and building automation. The PROFIBUS family is composed of three types of protocol, each of which is used for different tasks. The three types of protocols are: PROFIBUS FMS, DP and PA. IEC 61784-1: Communication network according to communication profile family 3 (CPF3); incorporating application profiles and system integration aspects like interfaces and languages for engineering tools and HMI. PROFIBUS is an open, digital communication system with a wide range of applications, particularly in the fields of factory and process automation. PROFIBUS is suitable for both fast, time-critical applications and complex communication tasks. The PROFIBUS logo is a registered trademark.
PROFIBUS DP	Acronym for "PROFIBUS for Decentralized Peripherals". Specification of an open fieldbus system with the following characteristics: • polling Master-Slave-system (cyclic communications, MS0); • flying Masters with robin round token passing coordination (MM);

	• connection based (MS1) and connectionless (MS2, MS3) acyclic communication between Masters and Slaves. Options (e.g.): • Data exchange broadcast (DXB), i.e. Slave to Slaves communication; • isochronous mode of Slaves; • clock synchronization; • redundancy. PROFIBUS DP is standardized within IEC 61158 and IEC 61784, communication profile families 3/1 and 3/2. The term "PROFIBUS DP" is also a synonym for the RS-485 based deployments within factory automation.
PROFIdrive	Communication technology especially adopted to the requirements of position and speed controlled drive applications (e.g. speed synchronized axis). Within the scope of PROFIBUS, "PROFIdrive" is used for the application of the PROFIBUS DP protocol (DP-V2) in motion control automation together with the corresponding application profiles ("PROFIdrive - Profile for variable speed drives" and "PROFIdrive - Profile drive technology") for the transmission technology RS-485.
Profile	Besides other things profiles in common define agreements on how to use communication means in a standardized manner. Within the context of fieldbusses several types of profiles are known: • communication profiles (e.g. IEC 61784); • physical profiles (MBP-IS, RS-485); • application profiles (see PROFIBUS TC3); • device profiles (e.g. robots); • branch profiles (e.g. extruder).
Profile Ident Number	Identifier of a particular profile definition. The profile ident number is taken from the pool of ident numbers handled by the PNO. It plays a role within the following scenarios. (1) In cases where the device of a manufacturer A should be replaceable by an equivalent device, the PNO is assigning number ranges to dedicated device types (Profile specific IDs) in combination with certain "Profile GSDs". Profiles using this methodology are e.g. "PA Devices" and "PROFIdrive". (2) Usually these Slave devices are designed to communicate with a Master Class 2 application (e.g. profile application or profile DTM). In order to ensure a Master application is communicating with an appropriate Slave, it is sending a profile specific ID during the establishment of the connection (MS2 Initiate Service). The Slave may answer with the same profile specific ID (if it is supporting this profile), with another ID (if it is supporting another profile) or with "0000h" if it is not supporting any profile. (3) I&M functions: Besides its basic I&M information devices -

	following a certain profile - are enabled to provide more detailed profile specific information.
Protocol Data Unit (PDU)	A packet of data passed across a network via telegrams. The term implies a specific layer of the OSI seven layer model and a specific protocol. Each layer has its own PDU that is extended subsequently from the physical layer up to the application layer: • Physical layer protocol data unit (PhPDU); • Data link protocol data unit (DLPDU); • Application protocol data unit (APDU).
Slave Parametrization	For a DP Slave several levels of parametrization exist. (1) The parameters on the DP communication level can be defined via a GSD file and comprise features such as baud rates, timing constraints, identification, options, transferable data structures, publisher subscriber links, etc. This level supports parametrization of simple modular Slaves and also special common additional communication layers such as PROFIsafe. This parametrization is fixed for a given operational life cycle after start-up. (2) More complex devices may be parametrized via EDD and/or FDT/DTM technology via an acyclic communication service (MS2). (3) For parameter changes at run-time such as batch operation (recipes) or motion control, special "parameter channels" associated with the cyclic data structures may be added or the MS1 services together with proxy function blocks may be used.
State Machine (DP)	An abstract machine consisting of a set of states (including the initial state), a set of input events, a set of output events, and a state transition function. A state machine describes the behaviour of a field device how to react in different situations. The state machine for DP Slaves comprises the following states/actions: - Power_On_Reset --> Set Slave address --> if successful, a transition follows to: - Wait_Prm --> Parametrization, diagnosis (optional) --> if successful, a transition follows to: - Wait_Cfg --> Configuration, diagnosis (optional) --> if successful, a transition follows to: - Data_Exch --> Normal operation: cyclic data exchange. On top of this basic communication layer state machine application profiles are defining their own additional state machines, e.g. PA devices, PROFIdrive, PROFIsafe, Ident Systems, Weighing and Dosage Systems. State machines are best modelled and documented with the help of the "Unified Modelling Language (UML)".
Station Address	Within PROFIBUS DP the address of a communication participant (Master or Slave). The permitted range is 0 to 127, with:

	- 126 intended to be used for the "soft" addressing of Slave devices; - 127 intended to be used for broadcast messages to all the Slaves.
Topology	In a communication network, the pattern of interconnection between network nodes; e.g. bus, ring, star configuration.
Transmission Rate (Baud rate)	The signalling rate of a digital communication line. It is the switching speed, or number of transitions (voltage or frequency changes) that are made per second. Within PROFIBUS DP the possible transmission rates depend on the MAU (Medium Attachment Unit) in use.
Watchdog Control	IEC 61158-6: This timer is part of the DP layer within a Slave. It is restarted by received requests from the bus Master and will set the outputs of a Slave to a fail-safe state after the expiration of the timer.
Watchdog Time (Twd)	IEC 61158-5: The watchdog timer is part of the DP layer within a Slave. The watchdog time is set by parametrization at run-up and consists of a watchdog time base (1 or 10 ms) and 2 factors. A selection can be made during configuration via the GSD file of a Slave. This is a Slave parameter. See "Watchdog control".

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;
- 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;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the unit;
- do not tool the unit or its shaft;
- delicate electronic equipment: handle with care; do not subject the device and the shaft to knocks or shocks;
- respect the environmental characteristics of the product;
- the actuator can be mounted directly on the drive shaft or coupled with planetary gearboxes. An adapting flange can be further interposed.

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

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

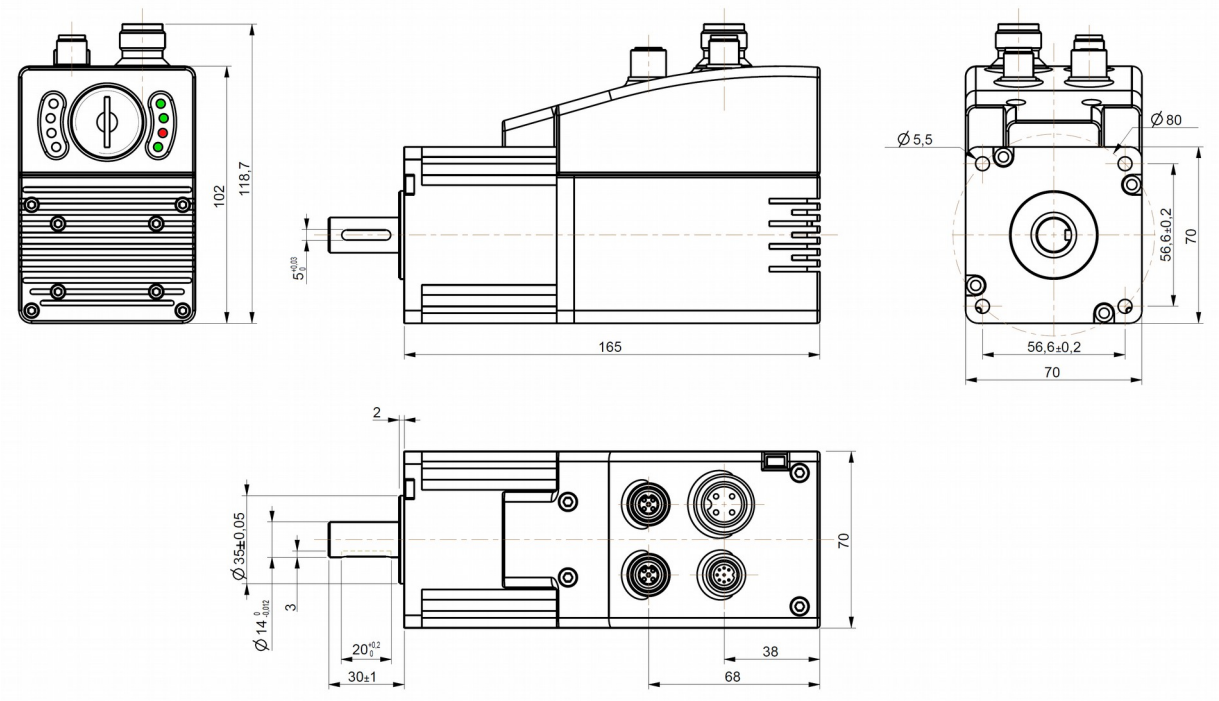
WARNING
Please evaluate attentively the characteristics of the 24V power supply pack as the counter-electromotive force (back EMF) generated by the motor flows back and directly discharge through the capacitor module of the 24V power supply pack.

2 Identification

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

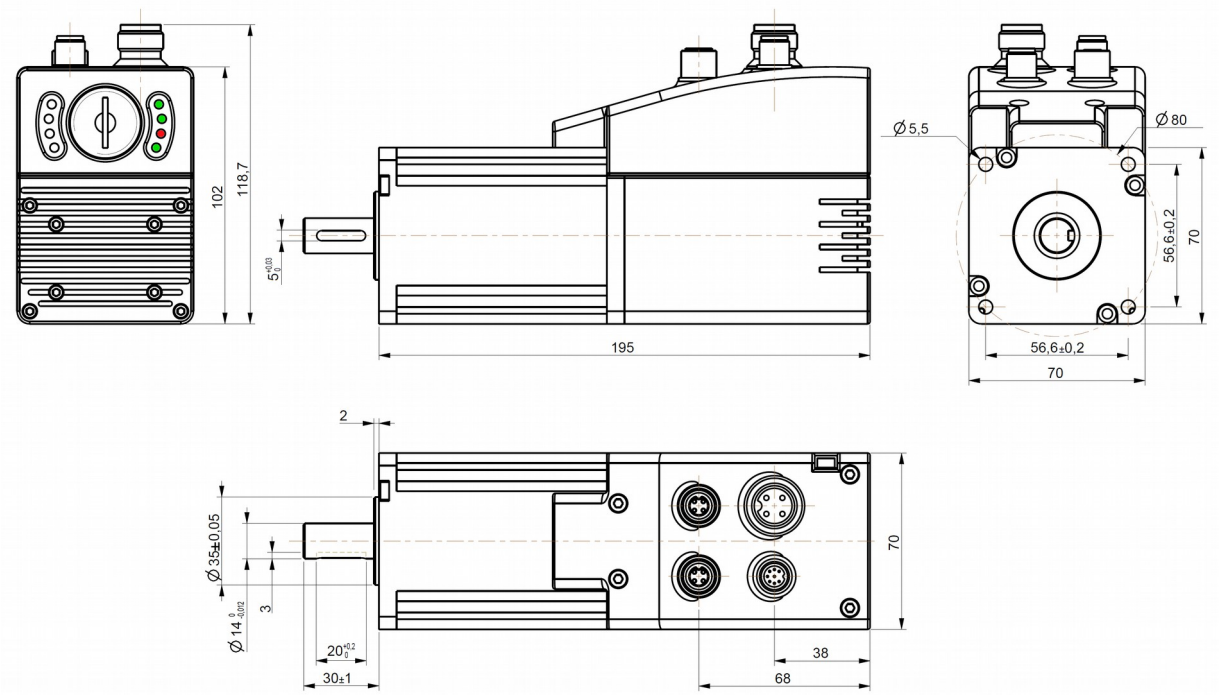
3 Mechanical installation

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 - RD6-P8-157-... unit – Overall dimensions



(values are expressed in mm)

Figure 2 - RD6-P8-250-... unit – Overall dimensions

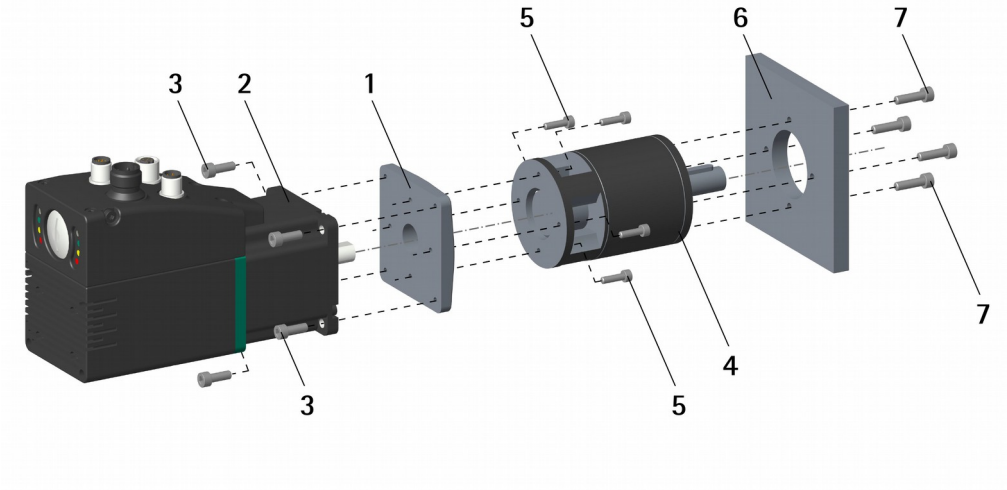
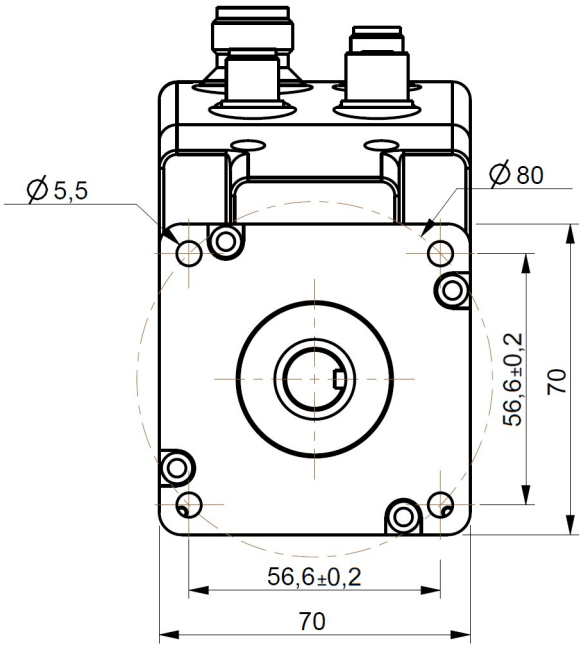


Figure 3 - Installation example of an RD6 unit

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

- If required, mount an adapting flange **1**; the adapting flange **1** can be either fixed to the actuator's flange **2** first and then to the planetary gearbox **4**; or on the contrary it can be fixed to the planetary gearbox **4** first and then to the actuator's flange **2**, according to the mounting configuration;



- use M5 type screws **3** to fix the adapting flange **1** to the actuator's flange **2**;
- use the screws **5** to fix the planetary gearbox **4** to the adapting flange **1**;
- couple and fasten the actuator and the planetary gearbox **4** together using the screws **3** or **5** according to the mounting configuration; properly secure the shaft of the actuator and the shaft of the planetary gearbox **4**;
- it could be advisable to install a coupling between the actuator and the planetary gearbox **4**;
- mount the shaft of the planetary gearbox **4** on the drive's shaft and properly secure them together; then fasten the planetary gearbox **4** to the flange or the mounting support **6** by means of the screws **7**.

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 spin due to a manual external force may cause irreparable damages to the internal circuitry.

4 Electrical connections

WARNING

When you send the **Start**, **Jog +** or **Jog -** commands, the unit and the shaft start moving! Before operating please make sure that there are no risks of personal injury and mechanical damages.
Each **Start** routine has to be checked carefully in advance!
Never force manually the rotation of the shaft not to cause permanent damages!

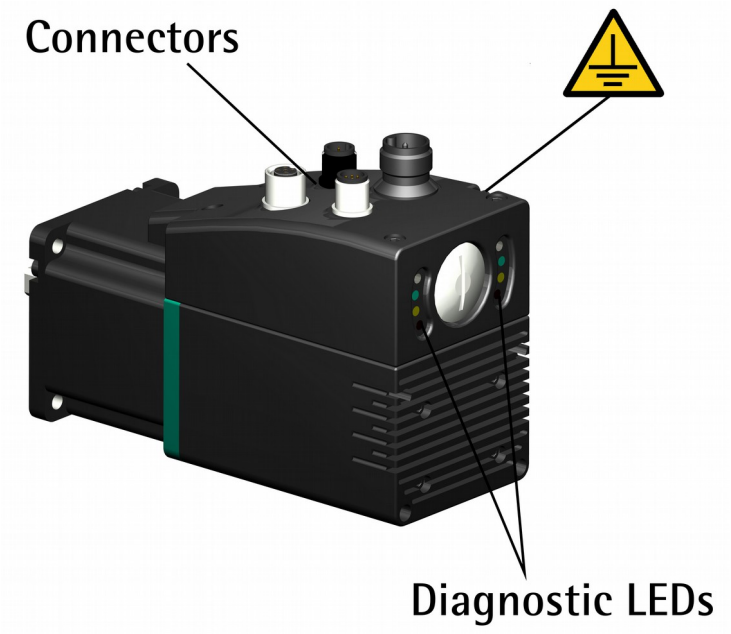
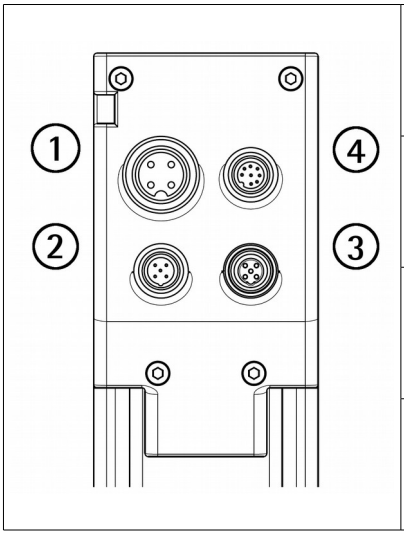


Figure 4: Connectors and diagnostic LEDs

4.1 Ground connection (Figure 5)

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

4.2 Connectors (Figure 4 and Figure 5)



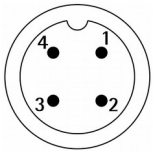
1	M18 4-pin male connector power supply
2	M12 5-pin male connector BUS IN
3	M12 5-pin female connector BUS OUT
4	M12 8-pin male connector, service serial port Modbus RS-232

Figure 5: Connectors

4.2.1 Power supply connector

Power supply
M18 4-pin male connector

(frontal side)



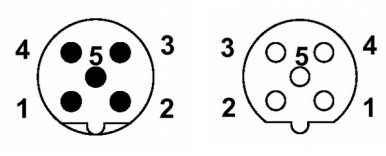
Pin	Description
1	**motor +24Vdc ±10% power supply**
2	**controller +24Vdc ±10% power supply**
3	**motor 0Vdc supply voltage**
4	**controller 0Vdc supply voltage**

NOTE
Wire gauge of the mating connector cable: 1.50 mm^2 max. Please consider the consumption of both the motor and the controller to evaluate the better configuration, see the datasheet.

4.2.2 Profibus-DP interface connectors (BUS IN and BUS OUT)

Two M12 5-pin connectors with B coding are used for Profibus-DP connection through BUS IN and BUS OUT.

Interface
M12 5-pin connectors
B coding
(frontal side)



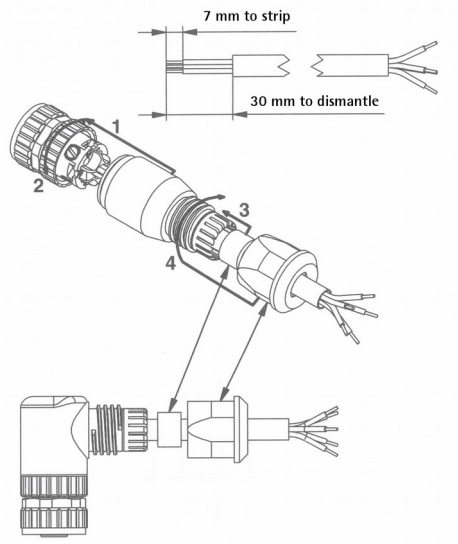
male (BUS IN) female (BUS OUT)

Pin	Description
1	**n.c.**
2	**Profibus A (Green)**
3	**n.c.**
4	**Profibus B (Red)**
5	**n.c.**
Case	**Shielding** [1]

n.c. = not connected
1 Connect the cable shield to the cable gland of the metal connector

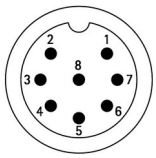
We recommend Profibus-DP certified connectors and cables to be used.

NOTE
We suggest always connecting the cable shield to ground on user's side.
Lika EC- pre-assembled cables are fitted with shield connection to the connector ring nut in order to allow grounding through the body of the device. Lika E-connectors have a plastic gland, thus grounding is not possible (see Figure below). If metal connectors are used, connect the cable shield properly as recommended by the manufacturer.



4.2.3 Modbus RS-232 service port

Modbus RS-232 service port
M12 8-pin connector
A coding, male
(frontal side)



Pin	Description
1	**n.c.**
2	**n.c.**
3	**n.c.**
4	**n.c.**
5	**n.c.**
6	**TD (RS-232)**
7	**RD (RS-232)**
8	**0Vdc (RS-232)**

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 according to the DIP switch setting. To set the node address of the RD6 device refer to the "4.4.1 Setting the node address (Figure 7)" section on page 38. See also the "8.2 "Serial configuration" page" section on page 99.

For any further information on configuring and using the RS-232 service serial port refer to the "Modbus® interface" section on page 97.

4.3 Diagnostic LEDs (Figure 4 and Figure 6)

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

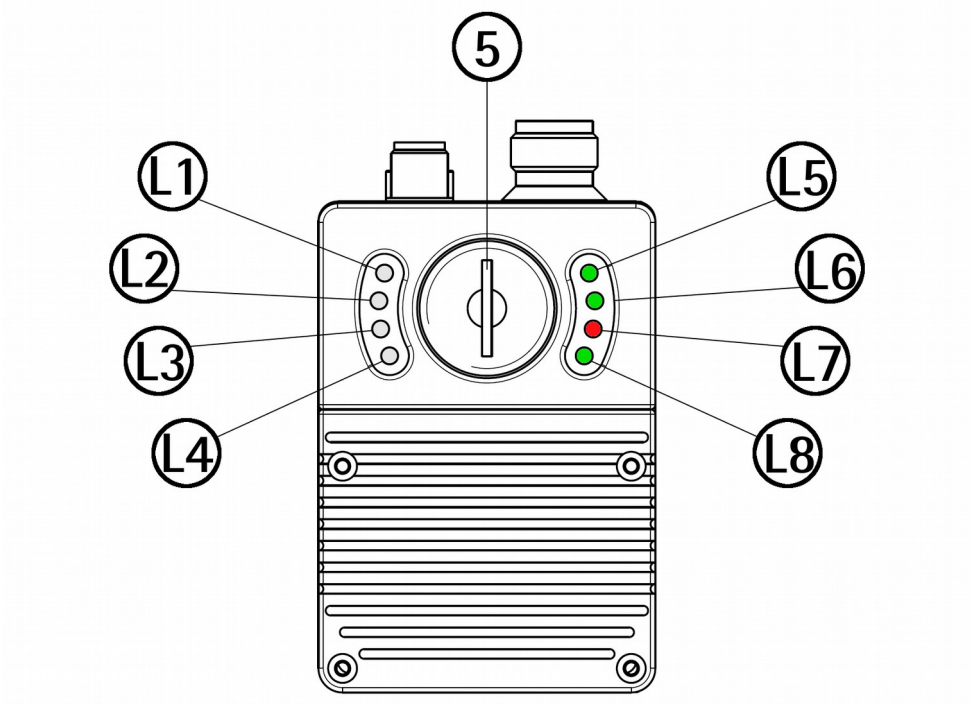


Figure 6: Diagnostic LEDs

	5	Internal housing for DIP switches	
L1	Not used	**L5**	Controller power supply information
L2	Not used	**L6**	Fieldbus interface status information
L3	Not used	**L7**	Active errors / faults information
L4	Not used	**L8**	Motor enabling information

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

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	LED L7 RED	Description
STATUS	FAULT	Profibus network diagnostic LEDs
OFF	OFF	Power supply is turned off or hardware breakdown not recognized
ON	OFF	Normal operation in Data_Exchange mode
Blinking	ON	It indicates that an alarm is active (for the complete list of alarm messages refer to page 80); or configuration parameters are not valid
ON	Blinking	Bus communication is cut off
Blinking	Blinking	Flash memory error, it cannot be restored

LED L8 GREEN	Description
It shows the enabling state of the motor	
ON	It indicates that the motor is enabled (control loop activated)
OFF	It indicates that the motor is disabled (control loop deactivated)

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 L7 RED	Description
ON	Active alarms, internal error. See the Alarms register [0x00] on page 142

OFF	No alarms active
Blinking at 2 Hz	While downloading data to the flash memory for upgrading the firmware of the unit (see the "8.7 "Upgrade Firmware" page" section on page 111), if an error occurs which stops the upgrading process (for instance: a voltage drop and/or the switching off of the ROTADRIVE unit), as soon as the power is turned on again the LED starts blinking red at 2 Hz as the user program is not installed in the flash memory (it has been deleted previously). For any information on restoring the unit please refer to the "8.7 "Upgrade Firmware" page" section on page 111 and the "8.7.1 If an installation issue occurs" section on page 113.
Blinking at 10 Hz	While downloading data to the flash memory for upgrading the firmware of the unit (see the "8.7 "Upgrade Firmware" page" section on page 111), the LED blinks red at 10 Hz.

LED L8 GREEN	**Description**
It shows the enabling state of the motor	
ON	It indicates that the motor is enabled (control loop activated)
OFF	It indicates that the motor is disabled (control loop deactivated)

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

4.4 DIP switches (Figure 4 and Figure 6)

WARNING
The power supply must be turned off before performing this operation!

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

To access the DIP switches unscrew and pull out the screw plug **5** (Figure 6) in the back of the enclosure. Be careful to replace the screw plug at the end of the operation.

The DIP switches are located just beneath the screw plug.

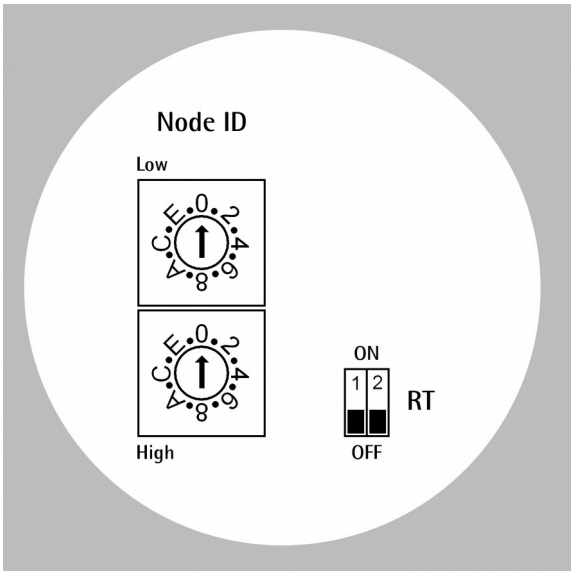


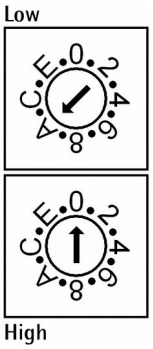
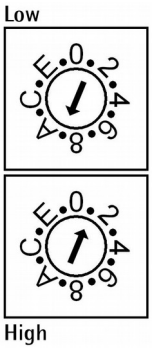
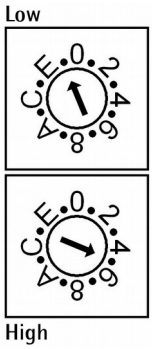
Figure 7: DIP switches

4.4.1 Setting the node address (Figure 7)

WARNING
The power supply must be turned off before performing this operation!

Set the node address (Node ID) expressed in hexadecimal notation.
The range of the node addresses is between 1 and 125 (125 = 7D hex).
The node address which is set using this selector affects both the Profibus and the MODBUS interfaces.

EXAMPLE

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

NOTE

The node address which is set using this selector affects both the Profibus and the MODBUS interfaces.
The address set at Lika is 1.
If you set the address to 0, the device will be set to 1 automatically.
If you set an address higher than 125, the device will be set to 125 automatically.

4.4.2 Setting the Baud rate of the Profibus network

The baud rate of the Profibus network is set by the Master via software at configuration of the node (Slave).
This device supports the following baud rates (they are listed in the .GSD file too):
9.6kbit/s, 19.2 kbit/s, 93.75 kbit/s, 187.5 kbit/s, 500 kbit/s, 1.5 Mbit/s, 3 Mbit/s, 6 Mbit/s, 12 Mbit/s.

The following table shows the maximum transmission rates in relation to permissible line length:

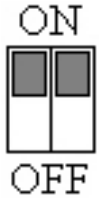
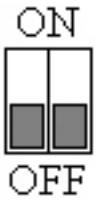
Baud rate [Kbit/s]	9.6	19.2	93.75	187.5	500	1500	12000
Max. cable length	1200 m 4000 ft	1200 m 4000 ft	1200 m 4000 ft	1000 m 3300 ft	400 m 1300 ft	200 m 660 ft	100 m 330 ft

NOTE
The baud rate which is set using this selector does not affect the Modbus fieldbus interface. For information on the baud rate of the Modbus interface, refer to the "4.2.3 Modbus RS-232 service port" section on page 34.

4.4.3 RT bus termination (Figure 7)

WARNING
The power supply must be turned off before performing this operation!

A bus termination resistor is provided and must be activated as line termination if the actuator is at the ends of the transmission line (i.e. it is either the first or the last device in the transmission line).
Use the RT switch to activate or deactivate the bus termination.

RT	Description
 1 = 2 = ON	Activated: when the actuator is either the first or the last device in the transmission line
 1 = 2 = OFF	Deactivated: when the actuator is neither the first nor the last device in the transmission line

5 Quick reference

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

- Mechanically install the device;
- execute the electrical connections;
- if required, set the node address (node ID; see on page 38); the value set by Lika Electronic at factory set-up is "1";
- switch on the +24Vdc power supply (in both the motor and the controller);
- check the operating condition shown through the LEDs;
- to resume the normal work condition reset the active emergency: switch high ("=1") the **Emergency** bit 7 of the **Control Word** (see on page 71 -Profibus interface; see on page 137 -MODBUS interface); reset the active alarms: switch high ("=1") the **Alarm reset** bit 3 of the **Control Word** (see on page 71 -Profibus interface; see on page 137 -MODBUS interface). Check the operating condition shown through the LEDs;
- set a proper value next to the **Distance per revolution** item (see on page 85 -Profibus interface; see on page 130 -MODBUS interface);
- set a proper value next to the **Jog speed** item (see on page 89 -Profibus interface; see on page 134 - MODBUS interface);
- set a proper value next to the **Work speed** item (see on page 89 -Profibus interface; see on page 134 -MODBUS interface);
- if required, set a proper value next to the **Preset** item (see on page 91 -Profibus interface; see on page 135 -MODBUS interface);
- set the limit switch values next to the **Positive delta** and **Negative delta** items (see on page 87 ff -Profibus interface; see on page 132 ff -MODBUS interface);
- set the commanded position next to the **Target position** item (see on page 74 -Profibus interface; see on page 140 -MODBUS interface);
- save the new setting values (**Save parameters** command; see on page 73 -Profibus interface; see on page 139 -MODBUS interface).

Use the **Jog+**, **Jog-**, **Start** and **Stop** commands in the **Control Word** (see on page 71 -Profibus interface; see on page 137 -MODBUS interface) to move the axis and reach the commanded position.

WARNING

When you install **Lika RD6** module, the value of each parameter is uploaded at power-on from the GSD file which has been loaded in the PLC. Thus any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message will be temporary:

when you turn off the power supply, the set value is lost (except the preset value which is the only variable not included in the GSD file; or unless you set the preset value previously which causes all parameters values to be saved automatically, see on page 91) and the value saved in the PLC will be uploaded at next power-on (thus all values, even if previously saved because of a preset setting, will be overwritten anyway).

This unit also allows to install the **Lika RD6-no param** module. Using this module, the unit does not read the parameters from the PLC when the power is switched on (they are read from the flash memory) and allows the operator to save any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message on the non-volatile memory (by means of the bit 9 **Save parameters** in the **Control Word (Bytes 0 and 1)**). When **Lika RD6-no param** module is installed, it is NOT possible to read and change the values of the **configuration data** parameters in the **Device-specific parameters** page of the TIA **Properties** window (see the "7.1.4 Device view" section on page 52). Thus you are allowed to enter new parameter values only through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message or by using the Modbus interface.

Using the RS-232 Modbus service serial interface it is possible to alter and then save the parameter values; in this way altered values are available again at next power-on. But this is true only until the serial interface is kept active: as soon as the unit is connected to the Profibus network, data in the PLC will be uploaded automatically and therefore the values set through the serial interface will be overwritten.

On the other hand using Siemens TIA PORTAL it is possible to alter any value and then save it permanently in the PLC (except in the -no param module). To save values permanently the operator has to enter the **Device-specific parameters** page of TIA and then alter the desired item (see the "7.1.4 Device view" section on page 52). Values altered in the Control Table of TIA (see the "7.1.13 Control Table" section on page 61) are temporary only.

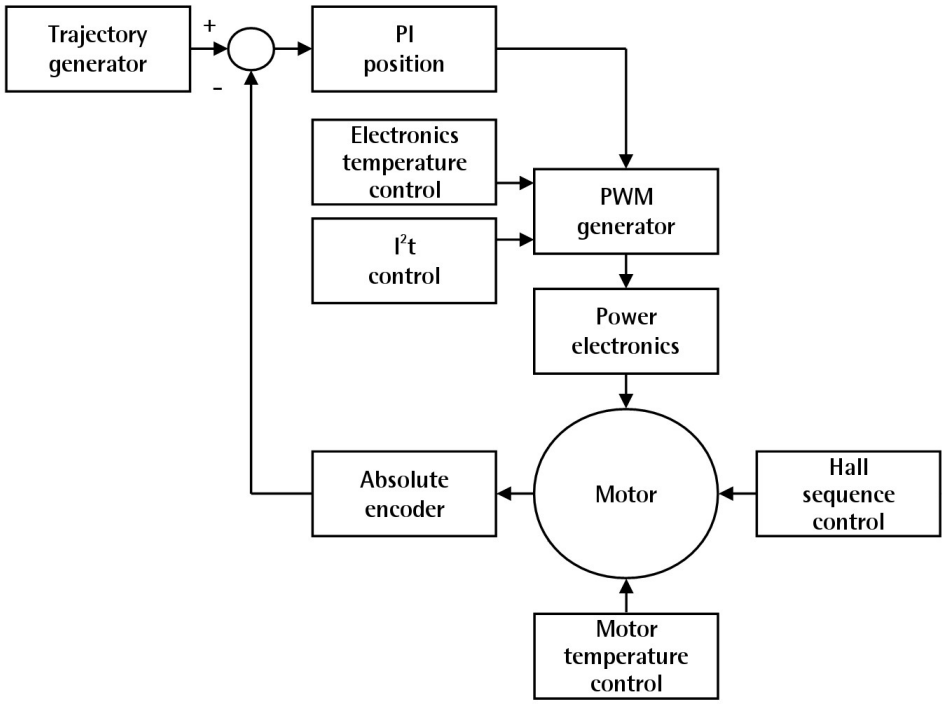
NOTE

The parameters **Distance per revolution**, **Position window**, **Max following error**, **Positive delta**, **Negative delta**, **Jog step length** and **Preset** are closely related, hence you have to be very attentive when you need to change the value in any of them. For any further information please refer to page 46.

6 Functions

6.1 Working principle

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



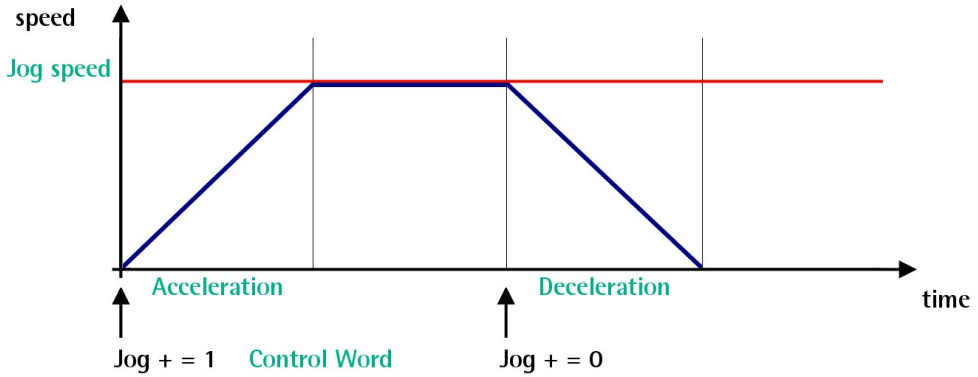
6.2 Movements: jog and positioning

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

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

Jog: speed control

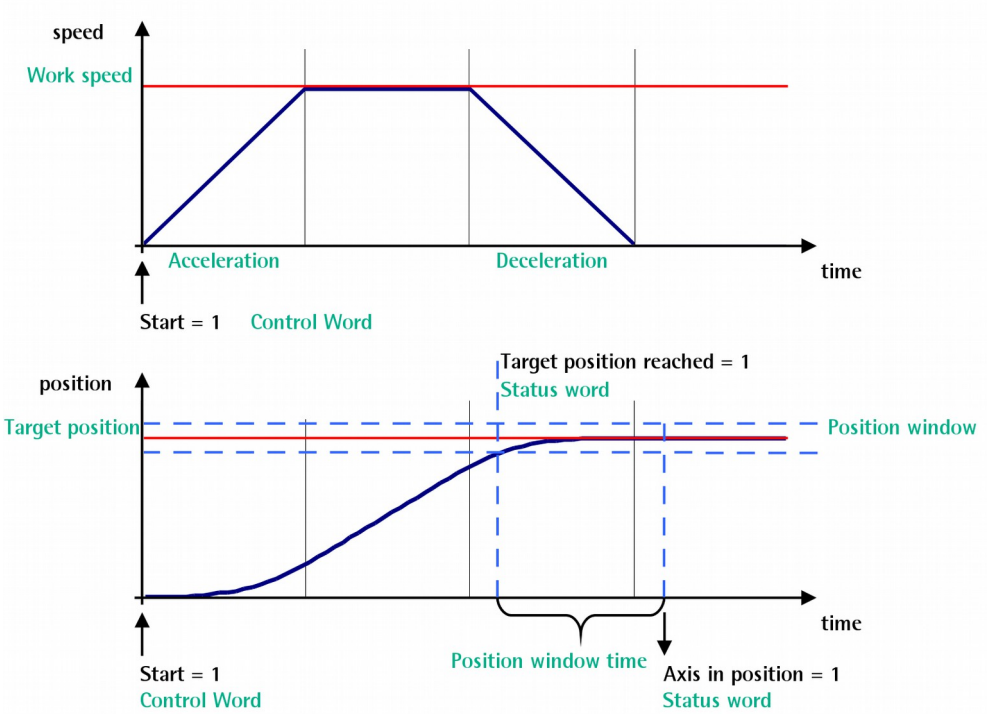
This kind of control is intended to generate a speed trajectory which allows the rotation speed of the DRIVECOD unit shaft to be equal to the value set in the **0A Jog speed** / **Jog speed [0x0D]** parameter.



When the bit 0 **Jog +** in the **Control Word (Bytes 0 and 1)** / **Control Word [0x2A]** is "=1", the motor accelerates toward the positive direction according to the value set next to the **06 Acceleration** / **Acceleration [0x07]** item; if the available travel is long enough it reaches the speed set next to the **0A Jog speed** / **Jog speed [0x0D]** item. As soon as the bit 0 **Jog +** in the **Control Word (Bytes 0 and 1)** / **Control Word [0x2A]** goes low ("=0"), the motor decelerates according to the value set next to the **07 Deceleration** / **Deceleration [0x08]** item until it stops.
Setting the bit 1 **Jog -** in the **Control Word (Bytes 0 and 1)** / **Control Word [0x2A]** to "=1" causes the motor to run in the opposite direction (negative direction) respecting the work phases already described above.

Positioning: position and speed control

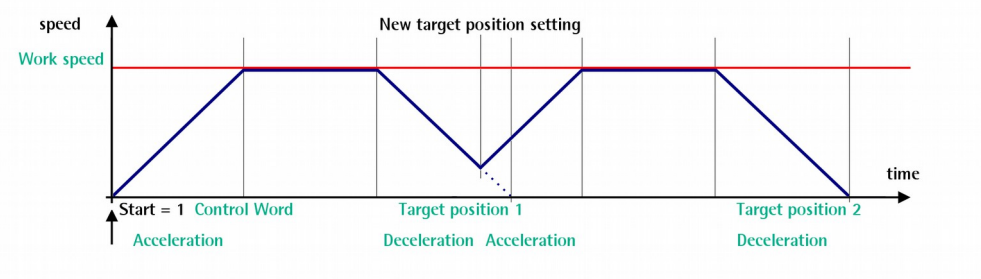
This kind of control is a point-to-point movement and the maximum reachable speed is equal to the value set in the **0B Work speed** / **Work speed [0x0E]** parameter; the set speed can be reached only if the available travel is long enough.



When the bit 6 **Start** in the **Control Word (Bytes 0 and 1)** / **Control Word [0x2A]** is "=1", the motor starts moving and accelerates according to the value set next to the **06 Acceleration** / **Acceleration [0x07]** item in order to reach the target position as set next to the **Target position (Bytes 4 ... 7)** / **Target position [0x2B-0x2C]** item. If the available travel is long enough it reaches the speed set next to the **0B 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 **01 Position window** / **Position window [0x01]** item, the bit 8 **Target position reached** in the **Status word (Bytes 0 and 1)** / **Status word [0x01]** goes high ("=1"). When the position is within the tolerance window limits set next to the **01 Position window** / **Position window [0x01]** item, after the delay set next to the **02 Position window time** / **Position window time [0x02]** item, the bit 0 **Axis in position** in the **Status word (Bytes 0 and 1)** / **Status word [0x01]** goes high ("=1"). The motor decelerates according to

the value set next to the **07 Deceleration** / **Deceleration [0x08]** item in order to reach the halt position according to the set target position.

NOTE
Position override function
It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the **Target position (Bytes 4 ... 7)** / **Target position [0x2B-0x2C]** item.



6.3 Distance per revolution, Preset, Positive delta and Negative delta

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

- set a proper value next to the **Distance per revolution** item (see on page 85 -Profibus interface; see on page 130 -MODBUS interface);
- set a proper value next to the **Preset** item (see on page 91 -Profibus interface; see on page 135 -MODBUS interface);
- check the value next to the **Positive delta** item (see on page 87 -Profibus interface; see on page 132 -MODBUS interface);
- check the value next to the **Negative delta** item (see on page 88 -Profibus interface; see on page 133 -MODBUS interface);
- save the new values (**Save parameters** command, bit 9 in the **Control Word (Bytes 0 and 1)** / **Control Word [0x2A]** item, see on page 71 -Profibus interface; see on page 139 -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.

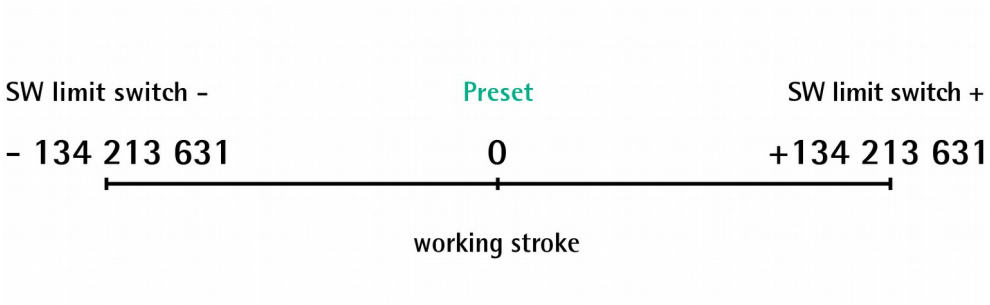
After having changed the parameter in the **Preset** item it is not necessary to set new values for the travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in **Positive delta** and **Negative delta**.

The number of revolutions managed by the system is 32,768 in the negative direction and 32,768 in the positive direction assuming the **Preset** value as a reference.
The value set next to the **Positive delta** item plus the value set in the **Preset** parameter is the maximum forward travel (positive travel) starting from the preset (the value is expressed in pulses).
The value set next to the **Negative delta** item subtracted from the value set in the **Preset** parameter is the maximum backward travel (negative travel) starting from the preset (the value is expressed in pulses).

WARNING
Please note that the parameters listed hereafter are closely related to the **Distance per revolution** parameter; hence when you change the value in **Distance per revolution** also the value in each of them necessarily changes. They are: **Position window**, **Max following error**, **Positive delta**, **Negative delta** and **Target position**. The **Current position** value is also affected.

EXAMPLE 1
Default values:
Distance per revolution = 4096 steps per revolution
Max. **Work speed**: 2000 rpm
Preset = 0

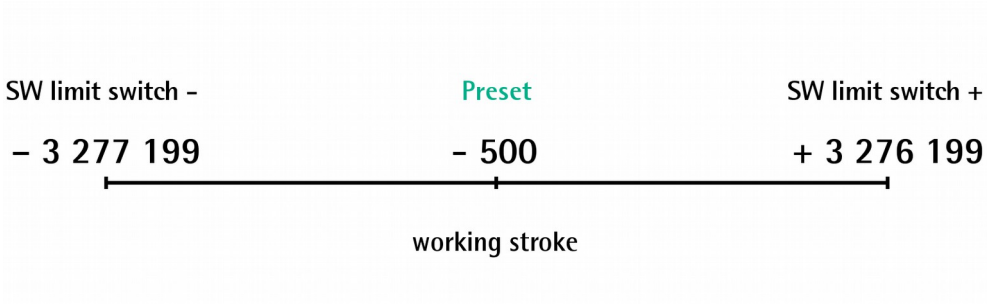
Positive delta and **Negative delta** max. values = 134 213 631 = (4,096 steps per revolution * 32,768 revolutions) - 1 step – 4,096 steps (i.e. 1 revolution for safety reasons) when **Preset** = 0
Max. **SW limit switch +** = 0 + 134 213 631 = + 134 213 631 pulses (forward travel)
Max. **SW limit switch -** = 0 - 134 213 631 = - 134 213 631 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 +** = + 134 213 631 and max. **SW limit switch -** = – 134 213 631.



EXAMPLE 2

DRIVECOD RD6 positioning unit is joined to a worm screw having 1 mm pitch and you need to have a hundredth of a millimetre resolution.

Distance per revolution = 100 steps per revolution
Max. **Work speed** = 73 rpm (100 * 3000 / 4096 = 73)
Preset = -500 (ex. thickness of the tool)
Positive delta and **Negative delta** = (100 steps per revolution * 32,768 revolutions) - 1 step – 100 steps (i.e. 1 revolution for safety reasons) = 3 276 699 pulses
Max. **SW limit switch +** = (-500) + 3 276 699 = +3 276 199 pulses (forward travel)
Max. **SW limit switch -** = (-500) – 3 276 699 = -3 277 199 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 +** = + 3 276 199 and max. **SW limit switch -** = – 3 277 199.



7 Profibus® interface

Lika ROTADRIVE positioning units with Profibus interface are Slave devices and are designed in compliance with "PROFIBUS-DP Profile".
For any further information or omitted specifications please refer to documentation issued by Profibus International and available at the address www.profibus.com.

7.1 Configuring the unit using TIA PORTAL

7.1.1 Preliminary information

In this manual some screenshots are shown to explain how to install and configure the encoder in a supervisor. In the specific example the development environment is TIA PORTAL V13 + SP1 with SIEMENS PLC CPU 315-2 PN / DP. If you need to install the encoder using a different configuration tool, please read and follow carefully the instructions given in the documentation provided by the manufacturer.

Lika Electronic Profibus rotary actuator documentation is complete with an **example project** provided free of charge. This program is designed to make your own project planning, programming, communication and diagnostics with the TIA PORTAL V13 + SP1 development environment user-friendly and reliable. You can find it in the **Lika TIA Portal V13 Profibus example project.zip** compressed file contained in the **Software RD6.zip** file.

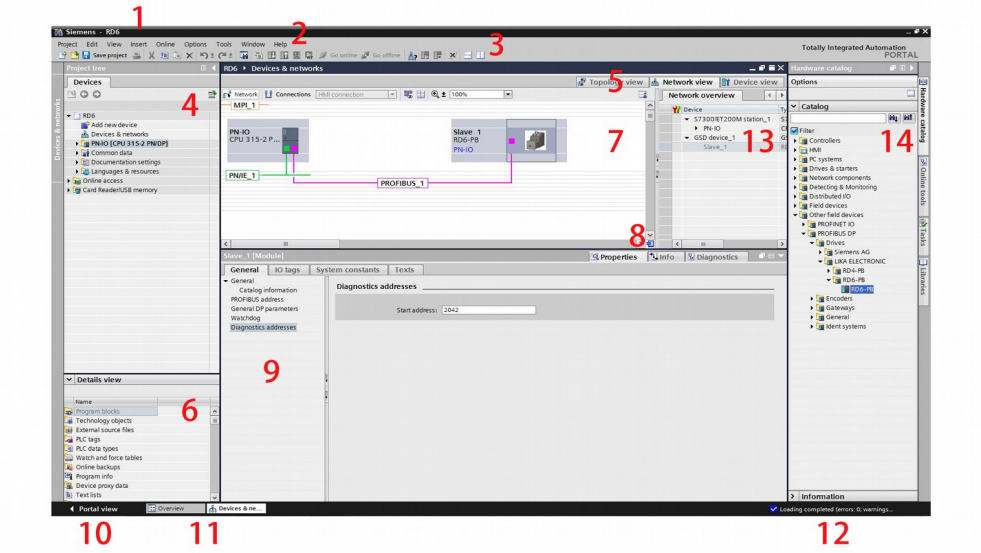
NOTE
Lika Electronic does not accept responsibility for any loss or damage that you could suffer as a result of using the example project. It is provided "as is", without warranty of any kind, express or implied. In no event shall Lika Electronic be liable for any claim, damages or other liability arising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

7.1.2 About TIA Portal

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

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

7.1.3 Project overview



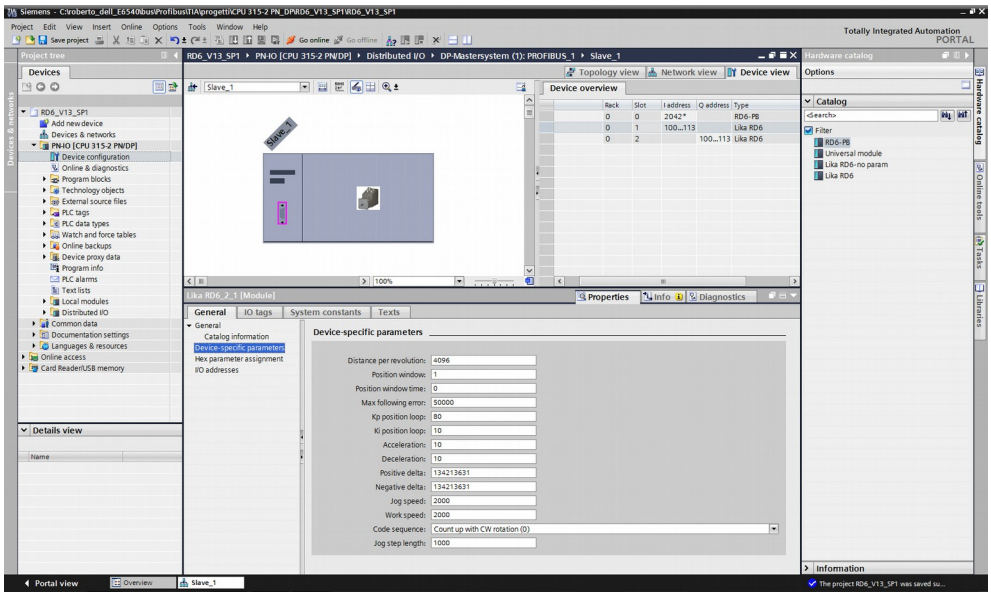
1. **Title bar**: the name of the project is displayed in the title bar.
2. **Menu bar**: the menu bar contains all the commands that you require for your work.
3. **Toolbar**: the toolbar provides you with buttons for commands you will use frequently. This gives you faster access to these commands.
4. **Project Tree**: using the Project Tree features gives you access to all components and project data. You can perform the following tasks in the Project Tree:
    - add new components
    - edit existing components
    - scan and modify the properties of existing components
5. **Changeover switches**: they allow to switch among the three working areas of the **Hardware and network editor**: Topology view, Network view and Device view. See point 7 for more information.
6. **Details view**: it shows certain content of the selected object in the **Overview Window** or in the **Project Tree**. This might include text lists or tags. The content of the folders is not shown, however. To display the content of the folders, use the **Project Tree** or the **Inspector Window**.

7. **Graphic Area** of the **Hardware and network editor**. The **Hardware and network editor** opens when you double-click on the **Devices and Networks** entry in the **Project Tree**. The **Hardware and network editor** is the integrated development environment for configuring, networking and assigning parameters to devices and modules. It provides maximum support for the realization of the automation project. This pane is the graphic area where the current configuration of the installed devices with information on the topology and the network can be found. The **Hardware and network editor** provides you with three views of your project. You can switch between these three views at any time depending on whether you want to produce and edit individual devices and modules, entire networks and device configurations or the topological structure of your project.
   See the **Changeover switches**, point 5: **Device view** for parametrisation and configuration of the individual devices, it allows to configure and assign both device and module parameters, see on page 52; **Network view** for graphical connections between devices, it allows to configure and assign device parameters and to network the devices with one another, see on page 53; and **Topology view** for current interconnection of Profibus devices, it allows to display and configure the Ethernet topology as well as to identify and minimize differences between the desired and actual topology, see on page 54. In the Figure above the SIEMENS PLC CPU 315-2 PN / DP is the Master device and is connected to the RD6-PB rotary actuator, i.e. the Slave device, through the PROFIBUS_1 connection.
8. **Overview Navigation**, it allows to quickly scroll through the objects available in the **Work Area** by pressing the left button of the mouse.
9. **Inspector window**: additional information on an object selected or on actions executed are displayed in the **Inspector window**, the available properties and parameters shown for the object selected can be edited in the Inspector window using the **Properties** tab.
10. It allows to enter the **Portal view**. The Portal view provides you with a task-oriented view of the tools.
11. **Editor bar**: it displays the open editors. If you have opened a lot of editors, they are shown grouped together. You can use the Editor bar to change quickly between the open elements.
12. **Status bar with progress display**. In the status bar, you will find the progress display for processes that are currently running in the background. This also includes a progress bar that shows the progress graphically. Hover the mouse pointer over the progress bar to display a tooltip providing additional information on the active background process. You can cancel the background processes by clicking the button next to the progress bar. If no background processes are currently running, the status bar displays the last generated alarm.

13. **Table Area** of the **Hardware and network editor**: it offers a general overview of the characteristics of the Device (when **Device view** is selected), of the Network (when **Network view** is selected) and of the Topology (when **Topology view** is selected).
14. **Task Cards**: depending on the edited or selected object, task cards that allow you to perform additional actions are available. These actions include:
    - selecting objects from a library or from the hardware catalog
    - searching for and replacing objects in the project
    - dragging predefined objects to the work area

    The task cards available can be found in a bar on the right-hand side of the screen. You can collapse and reopen them at any time. Which task cards are available depends on the products installed. More complex task cards are divided into panes that you can also collapse and reopen.
    The **Hardware catalog** can be selected in the **Task Cards**; it allows to install the available components just dragging and dropping them onto the **Work Area**. Customarily the field devices that have been integrated into the TIA Portal via GSD files are listed under **Other field devices > Profibus DP**.

7.1.4 Device view



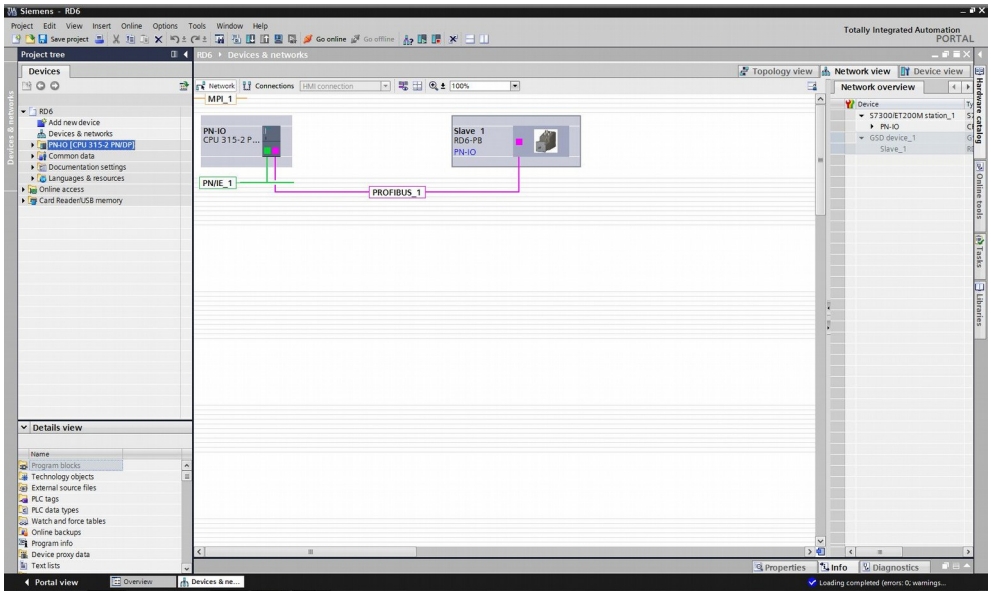
Press the **Device view** changeover switch in the **Hardware and network editor** to enter the **Device view**.
The configuration of devices and assigning of addresses etc. is performed in the Device view. All devices are represented in a photo-realistic way.

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

NOTE

The **Device-specific parameters** tabbed page is hidden and thus not available when you install the **Lika RD6-no param** module. Using this module, the unit does not read the parameters from the PLC when the power is switched on (they are read from the flash memory) and allows the operator to save any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message on the non-volatile memory (by means of the bit 9 **Save parameters** in the **Control Word (Bytes 0 and 1)**). When the **Lika RD6-no param** module is installed, parameters can be saved either through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message or by using the Modbus interface.

7.1.5 Network view



Press the **Network view** changeover switch in the **Hardware and network editor** to enter the **Network view**.
The Network view enables the configuration of plant communication. The communication links between individual stations are displayed here graphically and very clearly.

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

A subnet (PROFIBUS_1) is added to the operator panel. Click the subnet (PROFIBUS_1) to apply the network settings. Specify the required network settings under **Properties > Network Settings** in the **Properties** area (see point 9 on page 50). Make sure that you use the same settings throughout the entire network.

7.1.6 Topology view

Press the **Topology view** changeover switch in the **Hardware and network editor** to enter the **Topology view**.
Decentralised peripherals on Profibus are configured in the Network view. The controllers and the decentralised peripherals assigned to them can be shown graphically. During ongoing operation, however, it is not possible to see which ports are actually connected and communicating with each other.
Yet this is precisely what is often important for diagnostics. For Profibus networks, the Topology view enables this information to be displayed quickly and easily. An offline/online comparison identifies the communicating ports. By detecting, presenting and monitoring the physical connections between devices on Profibus, the administrator can easily monitor and maintain even complex networks.

7.1.7 Installing the GSD file

RD6 rotary actuators equipped with Profibus interface are supplied with their own GSD file **RD6Vx.GSD**, Vx is intended to indicate the file version. To download the file please enter **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS / POSITIONING UNITS (DRIVECOD) > RD6**).

The GSD file is available in both English version (**RD6Vx.GSE**) and Italian version (**RD6Vx.GSI**).
The GSD file has to be installed in the Profibus Master device.

WARNING
RD6 with Profibus interface provides the **-no param** module. Using this module, the unit does not read the parameters from the PLC when the power is switched on (they are read from the flash memory) and allows the operator to save any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message on the non-volatile memory (by means of the bit 9 **Save parameters** in the **Control Word (Bytes 0 and 1)**).

WARNING
Please be aware that you compulsorily comply with the following compatibilities between a release of the GSD file and the release of the Modbus executable file.

Compatibility	GSD file	Modbus EXE
	V1	From 1.2 up to ...

To install the RD6 unit on TIA Portal proceed as follows.

1. In the TIA Portal, select the **Options** > **Install device description file (GSD)** menu.
2. In the **Install general station description file (GSD)** dialog box, select the directory containing the GSD files.
3. Select the GSD file specific to the unit you need to install.
4. Click the **INSTALL** button.

Customarily the field devices that have been integrated into the TIA Portal via GSD files are listed under **Other field devices > Profibus DP**, see the "7.1.3 Project overview" section on page 50.

7.1.8 Adding a node to the project

On the right side of the TIA Portal, open the **Hardware catalog** task card. The field devices integrated into the TIA Portal via the device Master file (GSD file) can then be found in the **Hardware catalog** under **Other field devices**.
Select the desired device from the **Hardware catalog** (for instance, **RD6-PB**) and use drag-and-drop to move the item from the editing window to the

Network view. This creates the device in the project. Detailed information on the selected device is available at the bottom of the Hardware catalog in **Information**.

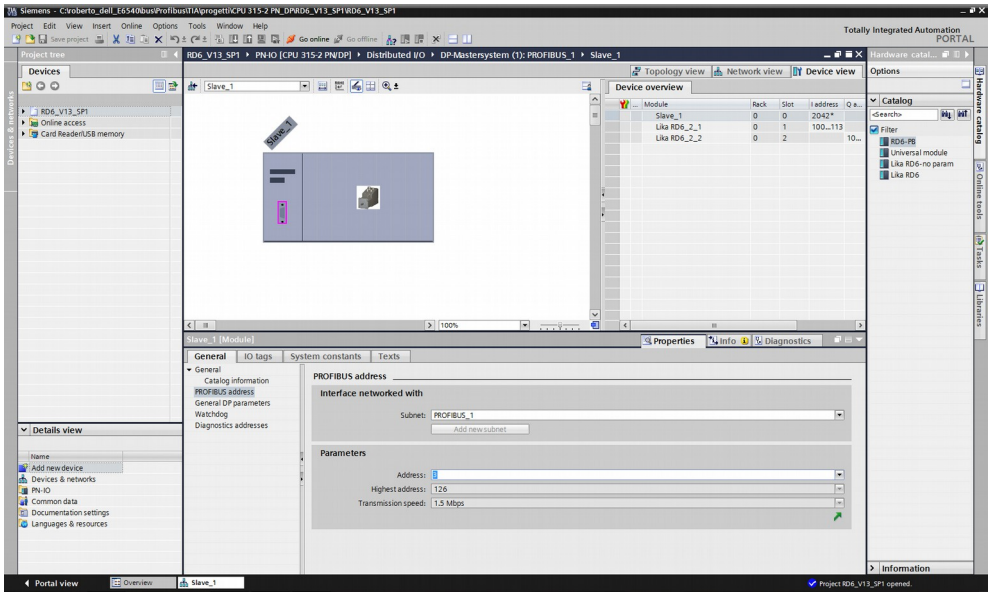
7.1.9 Establishing the bus connection

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

To establish the connection between the PLC and the Slave device via the PROFIBUS DP fieldbus, click on the PROFIBUS port of the Slave device in the TIA Portal and connect this port to the PROFIBUS port of the PLC while keeping the left mouse button pressed. When doing so, make sure that you are in the **Network** function mode in the **Network view**.

After configuring the networking, the device is connected to the PLC via a PROFIBUS DP Master system.

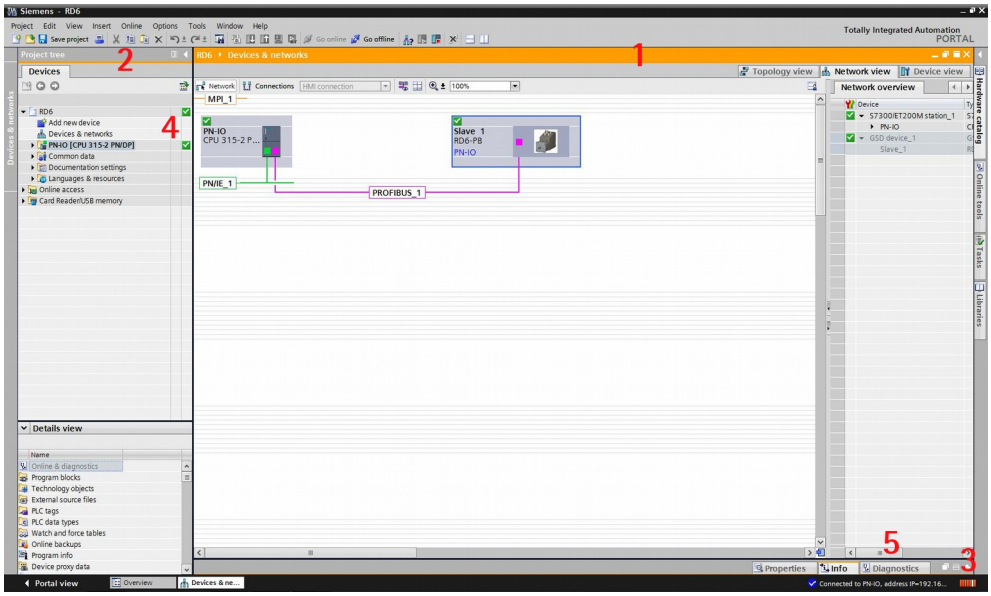
Finally, to completely establish the connection, you have to assign the PROFIBUS address of the Slave device. To do so, click on the image of the appropriate module in the **Network view**. In the **Properties** inspector window, **General** tab, you can now use the **PROFIBUS address** menu option to set the PROFIBUS address of the Slave device.



NOTE
At this point, it is not possible to change the baud rate of the PROFIBUS connection as the **Transmission speed** is displayed for information only.

If you also want to change the baud rate of the PROFIBUS connection, e.g. for faster data exchange between the PLC and the Slave device, click on the appropriate PROFIBUS DP Master system in the **Network view** and in the **General** tab, select the **PROFIBUS > Network settings** menu option. Then you can set the baud rate.

7.1.10 Establishing an online connection (Online mode)



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

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

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

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

To establish an online connection between the PLC (DP Master) and the device (DP Slave) proceed as follows.

- In the **Project Tree** (see point 4 in the "7.1.3 Project overview" section on page 50) mark the folder of the PLC that is configured as DP Master.
- Select the **Go online** command in the **Online** menu bar to establish an online connection to the PLC (DP Master) and to the device (DP Slave).
- If the device has already been connected online, the online connection is automatically established using the previously specified connection path.
- If there was no previous connection, the **Go online** dialog opens.
- Select the connection path:
  - select the type of interface;
  - select the interface of the PLC;
  - select the interface or the subnet for the connection.
- Click the **START SEARCH** button. Devices which can be reached by the set connection path are displayed in the **Compatible devices in target subnet**. The connection line in the graphic is displayed as solid.
- Select the device in the **Compatible devices in target subnet table** and confirm the selection with **Go online**. The online connection to the selected target device is established.

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

1. The title bar of the active window gets an orange background as soon as at least one of the devices currently displayed in the editor has been successfully connected online. If one or more devices are unavailable, a symbol for a broken connection appears in the title bar of the editor.
2. The title bars of inactive windows for the relevant station now have an orange line below them.
3. An orange, pulsing bar appears at the right-hand edge of the status bar. If the connection has been established but it is not working properly, an icon for an interrupted connection is displayed instead of the bar. You will find more information on the error in **Diagnostics** in the **Inspector window**.
4. Operating mode symbols or diagnostics symbols for the stations connected online and their underlying objects are shown in the **Project Tree**. A comparison of the online and offline status is also made

automatically. Differences between online and offline objects are also displayed in the form of symbols.

5. The **Diagnostics > Device information** area is brought to the foreground in the **Inspector window**.

7.1.11 Closing an online connection

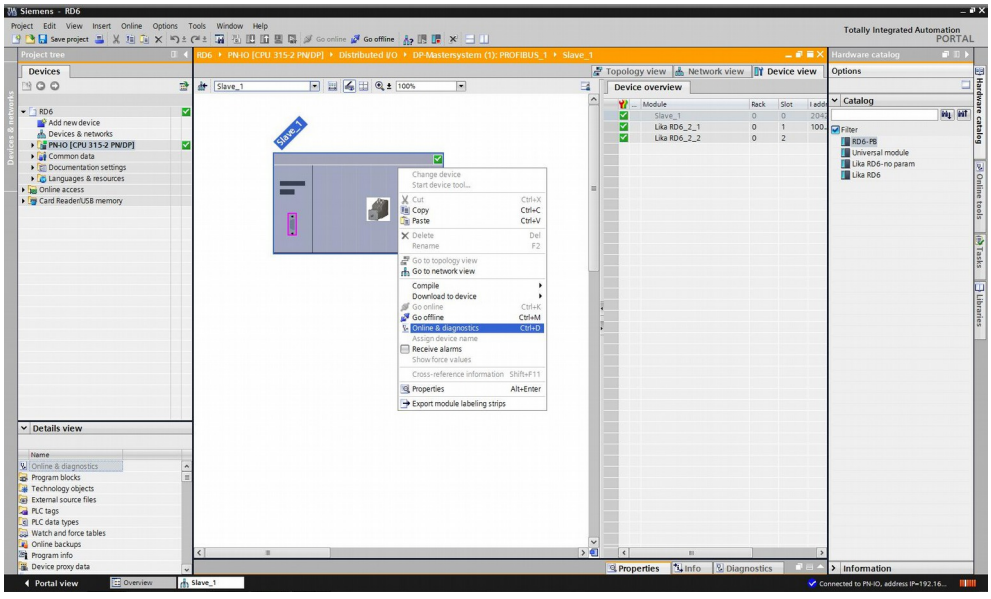
To close the existing online connection, follow these steps.

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

7.1.12 Diagnostics

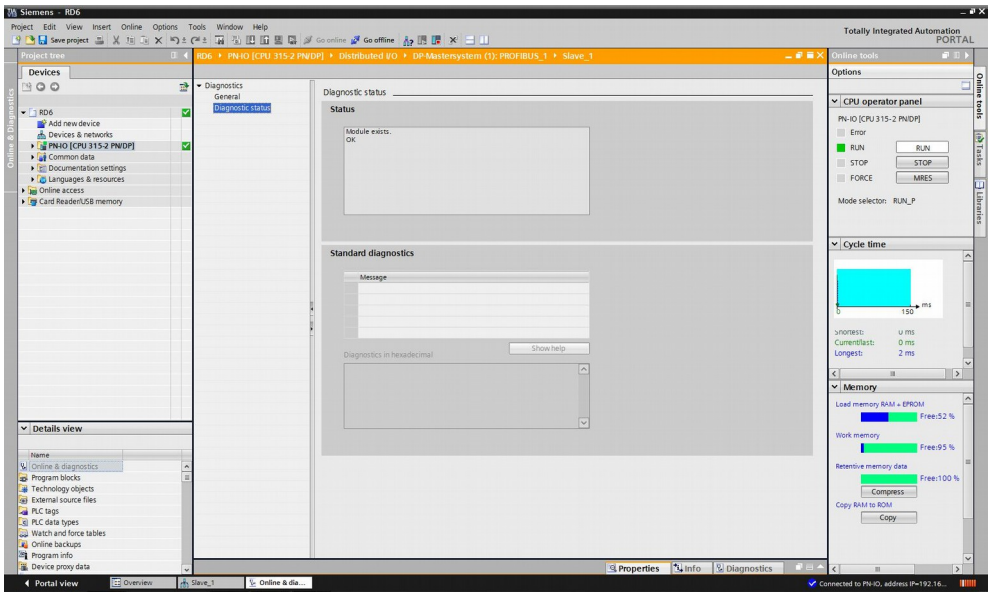
Configuration of the diagnostics is integrated in the system in a user-friendly way and activated with just one click. When new hardware components are introduced, the diagnostic information is updated automatically via the engineering system (HWCN). System diagnostics outputs all relevant information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

- Module
- Message text
- Message status



To access the diagnostics function please proceed as follows.

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



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

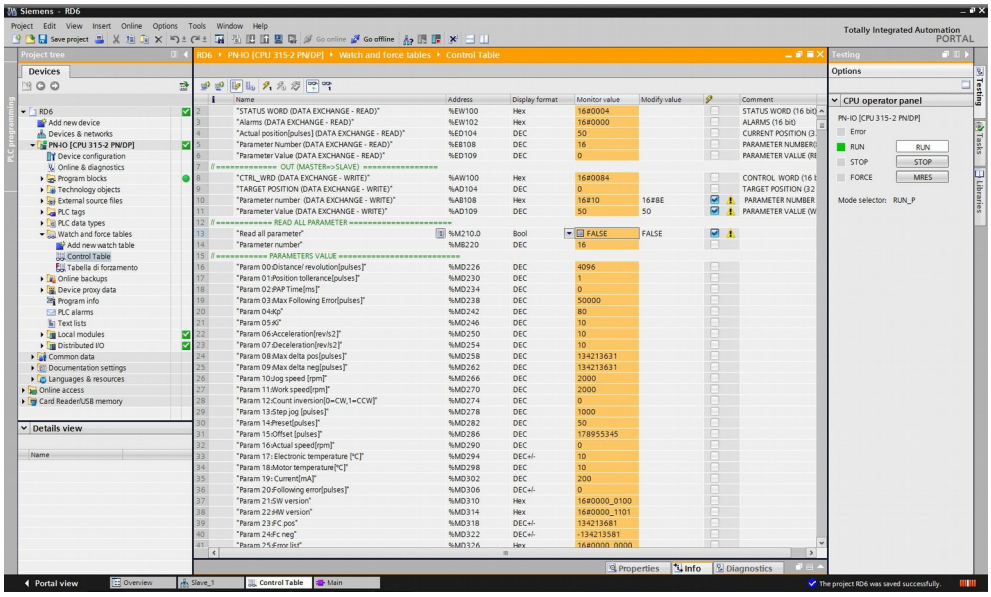
- Status of the module as viewed by the CPU, for example:
  - Module available and OK.
  - Module defective.
    If the module experiences a fault and you have enabled the diagnostic error interrupt during configuration, the "Module defective" status is displayed.
  - Module configured, but not available.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

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

7.1.13 Control Table

The Control Table is available in the **example project** provided free of charge. This program is designed to make your own project planning, programming, communication and diagnostics with the TIA PORTAL V13 + SP1 development environment user-friendly and reliable. You can find it in the **Lika TIA Portal V13 Profibus example project.zip** compressed file contained in the **Software RD6.zip** 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.



Press **Control Table** in the **Watch and force tables** folder to access the Control Table.
The Control Table is only available for testing and sample purposes. It allows you to monitor and modify the current values of individual parameters. You can assign values to individual parameters for testing and run the program in a variety of different situations. The values are not saved permanently.

The page is divided into eight sections:

1. **IN (SLAVE => MASTER)**
   In this section data sent by the Slave to the Master is displayed. In particular the **Status word (Bytes 0 and 1)** -"STATUS WORD (DATA EXCHANGE - READ)"-, the **Alarms (Bytes 2 and 3)** -"ALARMS (DATA EXCHANGE - READ)"-, the **Current position (Bytes 4 ... 7)** -"ACTUAL POSITION [PULSES] (DATA EXCHANGE - READ)"-, the requested **Parameter number (Byte 8)** -"PARAMETER NUMBER (DATA EXCHANGE - READ)"- and **Parameter value (Bytes 9 ... 12)** -"PARAMETER VALUE (DATA EXCHANGE – READ)"- are found.

2. **OUT (MASTER => SLAVE)**
   In this section data sent by the Master to the Slave is displayed. In particular the **Control Word (Bytes 0 and 1)** -"CONTROL WORD (DATA EXCHANGE - WRITE)"-, the **Target position (Bytes 4 ... 7)** -"TARGET POSITION (DATA EXCHANGE - WRITE)"-, the requested **Parameter number (Byte 8)** -"PARAMETER NUMBER (DATA EXCHANGE - WRITE)"- and **Parameter value (Bytes 9 ... 12)** -"PARAMETER VALUE (DATA EXCHANGE – WRITE)"- are found.

3. **READ ALL PARAMETER**
   In this section you can choose to read all parameters ("READ ALL PARAMETER") or a specific parameter whose index is entered next to the "PARAMETER NUMBER" item.

4. **PARAMETERS VALUE**
   In this section the complete list of the available parameters is found. For more information refer to the "7.9 Profibus® programming parameters" section on page 84.

5. **STATUS WORD**
   In this section the value of the single bits of the **Status word (Bytes 0 and 1)** item is shown. See on page 79.

6. **CONTROL WORD**
   In this section the value of the single bits of the **Control Word (Bytes 0 and 1)** item is shown. See on page 71.

7. **ALARMS**
   In this section the value of the single bits of the **Alarms (Bytes 2 and 3)** item is shown. See on page 80.

8. **WRONG PARAMETERS**
   In this section the value of the single bits of the **19 Wrong parameters list** item is shown. See on page 94.

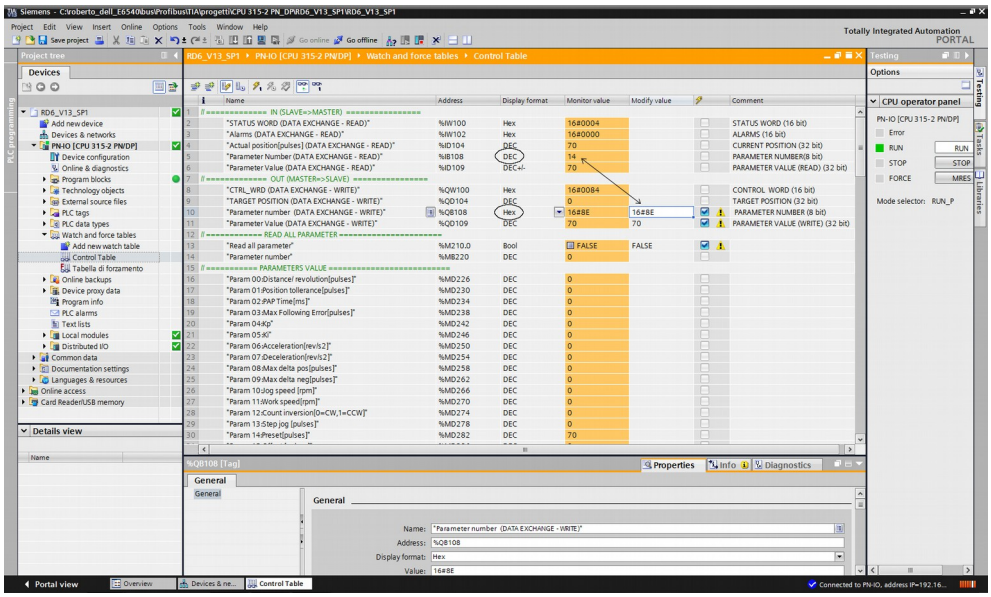
It is possible to choose the display format of each item.

For a comprehensive description of the parameters and how to set them properly refer to the specific explanation in the "7.9 Profibus® programming parameters" section on page 84.

7.1.14 Setting and reading parameter values

When the device is online and in **Data_Exchange** mode, setting and reading of both **configuration data** and **operating parameters** are allowed; the meaning of the messages transmitted between the Master and the Slave is explained in the "7.7 DDLM_Data_Exchange" section on page 70.
To set a new parameter value please refer to the example described in the "7.1.14.1 Setting the 0E Preset parameter" section just below.
To read a parameter value please refer to the example described in the "7.1.14.2 Reading the 10 Current velocity parameter" section further below.

7.1.14.1 Setting the 0E Preset parameter



In the example the preset value 70 dec is set in the device.
You must first enter the preset value you need to set in the device. In the example: Preset = 70 dec. The Preset value must be entered next to the "PARAMETER VALUE (DATA EXCHANGE – WRITE)" item in the OUT (MASTER => SLAVE) section.
Then you must send a request to WRITE in the **0E Preset** parameter the value previously set next to the "PARAMETER VALUE (DATA EXCHANGE – WRITE)" item. To do this you must enter both the index of the **0E Preset** parameter (=0Eh, see on page 91) and the command 80h for writing values (so, 0Eh + 80h = 8Eh) next to the "PARAMETER NUMBER (DATA EXCHANGE – WRITE)" item in the OUT (MASTER => SLAVE) section. Find more information next to the **Parameter number (Byte 8)** item on page 76.

Parameter value = 70	(Preset value, by way of example).
Parameter number = 8Ehex	index of the **0E Preset** parameter is 0Eh (see on page 91); then command 80h for writing values must be added (0Eh + 80h = 8Eh).

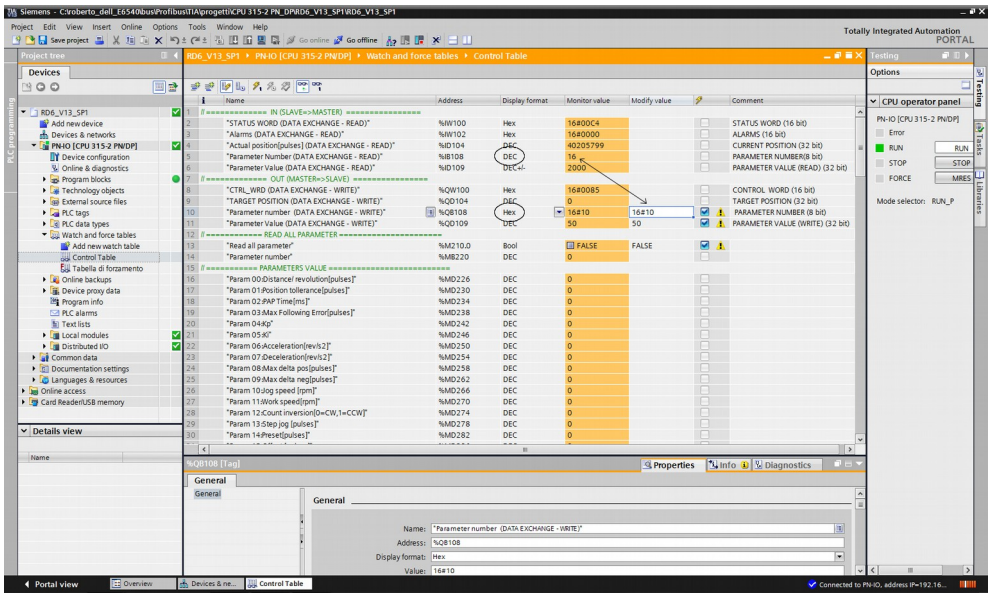
Finally press the **MODIFY ALL SELECTED PARAMETERS IMMEDIATELY AND ONCE**  button in the toolbar to confirm the setting. This command is executed once and as quickly as possible without reference to any particular point in the user program. The operation of the button is available only in online mode.

As soon as you set a new preset value next to the **0E Preset** parameter, the entered value is automatically activated (the **Setting the preset** bit operation is not required).

The set value can be read next to the "PARAMETER VALUE (DATA EXCHANGE – READ)" item in the IN (SLAVE => MASTER) section.
Furthermore, as **Current position (Bytes 4 ... 7)** = **0E Preset** (the actuator should be in stop, as suggested), the same value can be read also next to the "ACTUAL POSITION [PULSES] (DATA EXCHANGE - READ)" item in the IN (SLAVE => MASTER) section.

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

7.1.14.2 Reading the 10 Current velocity parameter



In the following example the device transmits to the Master its current speed value through the "PARAMETER VALUE (DATA EXCHANGE – READ)" item in the IN (SLAVE => MASTER) section.
To do this you must send a request to READ the **10 Current velocity** parameter, so enter both the index of the **10 Current velocity** parameter (=10h, see on page 92) and the command 00h for reading values (so, 10h + 00h = 10h) next to the "PARAMETER NUMBER (DATA EXCHANGE – WRITE)" item in the OUT (MASTER => SLAVE) section. The "PARAMETER VALUE (DATA EXCHANGE – WRITE)" item in the same section is ignored. Find more information next to the **Parameter number (Byte 8)** item on page 76.

Parameter number = 10hex — index of the **10 Current velocity** parameter is 10h (see on page 92); then command 00h for reading values must be added (10h + 00h = 10h).

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

Now the device transmits the speed value: "2000" revolutions per minute.
Parameter value = 2000 — speed = 2000 revolutions per minute (rpm), see the "PARAMETER VALUE (DATA EXCHANGE –

READ)" item in the IN (SLAVE => MASTER) section.

7.2 GSD file

RD6 rotary actuators equipped with Profibus interface are supplied with their own GSD file **RD6Vx.GSD**, Vx is intended to indicate the file version. To download the file please enter **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS / POSITIONING UNITS (DRIVECOD) > RD6**).
The GSD file is available in both English version (**RD6Vx.GSE**) and Italian version (**RD6Vx.GSI**).
The GSD file has to be installed in the Profibus Master device.

WARNING
Please be aware that you compulsorily comply with the following compatibilities between a release of the GSD file and the release of the Modbus executable file.

Compatibility	File GSD	Modbus EXE
	V1	From 1.2 up to ...

WARNING
When you install **Lika RD6** module, the value of each parameter is uploaded at power-on from the GSD file which has been loaded in the PLC. Thus any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message will be temporary: when you turn off the power supply, the set value is lost (except the preset value which is the only variable not included in the GSD file; or unless you set the preset value previously which causes all parameters values to be saved automatically, see on page 91) and the value saved in the PLC will be uploaded at next power-on (thus all values, even if previously saved because of a preset setting, will be overwritten anyway).

This unit also allows to install the **Lika RD6-no param** module. Using this module, the unit does not read the parameters from the PLC when the power is switched on (they are read from the flash memory) and allows the operator to save any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message on the non-volatile memory (by means of the bit 9 **Save parameters** in the **Control Word (Bytes 0 and 1)**). When **Lika RD6-no param** module is installed, it is NOT possible to read and change the values of the **configuration data**

parameters in the **Device-specific parameters** page of the TIA **Properties** window (see the "7.1.4 Device view" section on page 52). Thus you are allowed to enter new parameter values only through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message or by using the Modbus interface.

Using the RS-232 Modbus service serial interface it is possible to alter and then save the parameter values; in this way altered values are available again at next power-on. But this is true only until the serial interface is kept active: as soon as the unit is connected to the Profibus network, data in the PLC will be uploaded automatically and therefore the values set through the serial interface will be overwritten.

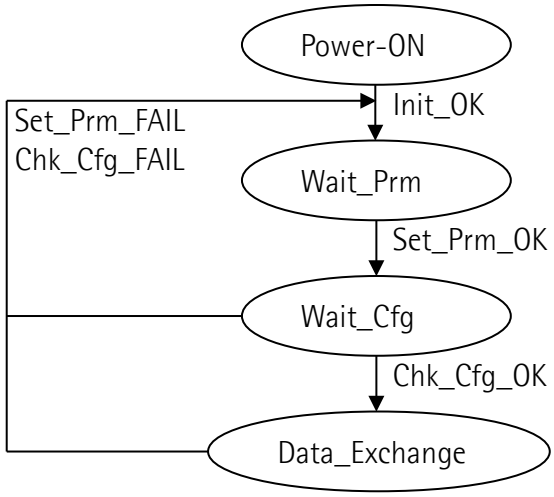
On the other hand using Siemens TIA PORTAL it is possible to alter any value and then save it permanently in the PLC (except in the -no param module). To save values permanently the operator has to enter the **Device-specific parameters** page of TIA and then alter the desired item (see the "7.1.4 Device view" section on page 52). Values altered in the Control Table of TIA (see the "7.1.13 Control Table" section on page 61) are temporary only.

7.3 Baud rate

RD6 ROTADRIVE positioning units with Profibus interface are able to recognize automatically the data transmission rate of the Profibus-DP Master. The same baud rate is set by the Master for all devices in the network. The supported baud rates are as follows:
9.6Kbps – 19.2Kbps – 93.75Kbps – 187.5Kbps – 500Kbps – 1.5Mbps – 3Mbps – 6Mbps – 12Mbps.

7.4 Operating states

Profibus devices are designed to operate using different communication states, see the Figure below.



NOTE
Configuration data parameters are sent when Set_Prm phase is active and the device is in Data_Exchange mode; operational parameters are sent when the device is in Data_Exchange mode.

7.4.1 Communication messages

A basic differentiation is made among the following statuses when exchanging data between the Master and the Slave:

- **DDLM_Set_Prm**: configuring and parametrizing phase. This is active when the system is turned on; in this phase configuration data is sent to the Slave device. For any further information refer to the "7.5 DDLM_Set_Prm" section on page 69.
- **DDLM_Chk_Cfg**: using this function the Master defines the number of bytes needed for data transmission in Data_Exchange mode. For any further information refer to the "7.6 DDLM_Chk_Cfg" section on page 70.
- **DDLM_Data_Exchange**: "Standard operation". In this work mode the Master can send the preset value or the target position to the Slave, while the Slave can respond sending the current position (and velocity, as well). For any further information refer to the "7.7 DDLM_Data_Exchange" section on page 70.

- **DDLM_Slave_Diag**: This is used when the system is turned on and every time the Master needs to acquire diagnostic information from the Slave: in this status the Slave sends diagnostic data to the Master. For any further information refer to the "7.8 DDLM_Slave_Diag" section on page 83.

7.5 DDLM_Set_Prm

When the system is turned on, configuration data set by the operator is sent to the Slave by the controller. Parameters transmission depends on the configuration chosen by the operator. Customarily data is sent automatically while data setting is carried out via the user's interface available in the controller's software (for instance, TIA PORTAL, see on page 49).
A detailed description of the available parameters can be found in the "7.9 Profibus® programming parameters" section on page 84.
Data transmission is carried out respecting the structure shown in the following table.

Bytes	Parameter
0 ... 6	Reserved for PROFIBUS network
7 ... 9	Reserved for PROFIBUS network
10 - 11	00 Distance per revolution
12 - 13	01 Position window
14 - 15	02 Position window time
16 ... 19	03 Max following error
20 - 21	04 Kp position loop
22 - 23	05 Ki position loop
24 - 25	06 Acceleration
26 - 27	07 Deceleration
28 ... 31	08 Positive delta
32 ... 35	09 Negative delta
36 - 37	0A Jog speed
38 - 39	0B Work speed
40	0C Code sequence
41 - 42	0D Jog step length

7.6 DDLM_Chk_Cfg

Configuration function allows the Master to define the number of bytes used for data transmission in **Data_Exchange** mode; transmission and receipt are viewed from the Master device.

Chk_Cfg message structure (1 byte):
bit 7 = Consistency (="1")
bit 6 = Word format ("0"=byte, "1"=word=2bytes)
bits 5 and 4 = IN/OUT data ("01"=Input, "10"=output)
bits 3 ... 0 = Code length

Allowed values:

bit	7	6	5	4	3	2	1	0	
Data	1	1	0	1	0	1	1	0	D6h
	1	1	1	0	0	1	1	0	E6h

D6hex = 14 byte input E6hex = 14 byte output

7.7 DDLM_Data_Exchange

This is the normal operation status of the system.
Using **Data_Exchange messages** system manages all available parameters and operates the movements of the axis.

Data_Exchange messages structure:

Byte	Master → Slave functions	Slave → Master functions
0	Control Word (Bytes 0 and 1)	Status word (Bytes 0 and 1)
1		
2	Not used	Alarms (Bytes 2 and 3)
3		
4	Target position (Bytes 4 ... 7)	Current position (Bytes 4 ... 7)
5		
6		
7		
8	Parameter number (Byte 8)	Parameter number (Byte 8)
9	Parameter value (Bytes 9 ... 12)	Parameter value (Bytes 9 ...
10		

11		
12		
13	Not used	Not used
	See the "7.7.1 Master →Slave functions" section on page 71	See the "7.7.2 Slave →Master functions" section on page 79

7.7.1 Master →Slave functions

In this section Data_Exchange messages sent by the Master to the Slave are described.

Control Word (Bytes 0 and 1)

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

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 0D Jog step length 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 0A Jog speed, 06 Acceleration and 07 Deceleration parameters respectively. For a detailed description of the jog control see on page 44.

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the 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 0D Jog step length 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 **0A Jog speed**, **06 Acceleration** and **07 Deceleration** parameters respectively. For a detailed description of the jog control see on page 44.

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

Stop

bit 2

If set to "=1" the Slave is allowed to execute the movements as commanded. If, while the unit is running, this bit switches to "=0" then the Slave must stop and execute the deceleration procedure set in **07 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 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 **19 Wrong parameters list**), the normal work status can be restored only after having set proper values. The **Flash memory error** and **Axis not synchronized** alarms cannot be reset.

Incremental jog

bit 4

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

bit 5

Not used.

Start

bit 6

When the bit value switches from "0" to "1", the device moves in order to reach the set target position (see **Target**

position (Bytes 4 ... 7) on page 74). For a complete description of the position control see on page 43. This bit has to be switched back to "0" after the device has started.

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

Emergency

bit 7

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

Byte 1

bit 8

Not used.

Save parameters

bit 9

The function of this bit is available only when you install the **Lika RD6-no param** module (see the "Preliminary information" section on page 11). You must NOT set this bit via Profibus if you do not use the -no param module. 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"). For detailed information on the **Lika RD6-no param** module please refer to the "Preliminary information" section on page 11.

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 function of this bit is available only when you install the **Lika RD6-no param** module (see the "Preliminary information" section on page 11). You must NOT set this bit via Profibus if you do not use the -no param module. 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 158. When the default parameters loading operation is carried out, system also triggers an automatic save of all settings on the flash memory. For detailed information on the **Lika RD6-no param** module please refer to the "Preliminary information" section on page 11.

Setting the preset

bit 11 — It sets the current position to the value set next to the **0E Preset** parameter. The operation is performed at each rising edge of the bit, i.e. each time this bit is switched from logic level low ("0") to logic level high ("1"). We suggest activating the preset when the actuator is in stop. For more information refer to page 91.

As soon as you set a new preset value next to the **0E Preset** parameter, the entered value is automatically activated, thus the **Setting the preset** bit operation is not required.

Release axis torque

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

Bits 13 ... 15 — Not used.

Bytes 2 and 3 — Not used.

Target position (Bytes 4 ... 7)

It sets the position to be reached, otherwise referred to as commanded position. The value is expressed in pulses. When the **Start** command is sent while **Stop** and **Emergency** bits are "=1" and the alarm condition is off, the device moves in order to reach the target position set next to this item.
As soon as the axis is within the tolerance window limits set next to the **01 Position window** item, the bit 8 **Target position reached** in the **Status word (Bytes 0 and 1)** goes high ("=1"). When the position is within the tolerance window limits set next to the **01 Position window** item, after the delay set next

to the **02 Position window time** item, the bit 0 **Axis in position** in the **Status word (Bytes 0 and 1)** goes high ("=1").
For more information refer also to the "Positioning: position and speed control" section on page 45.
Default = 0 (min. = 0, max. = within the maximum positive limit / maximum negative limit)

Byte 4	byte 5	byte 6	byte 7
Low	...	...	High

Position override function

It is possible to change the target position value even on the fly, while the device is still reaching a previously commanded target position and without sending a new **Start** command. To do this, just set a new target value in the **Target position (Bytes 4 ... 7)** item. See also on page 45.

NOTE

Jog +, **Jog -** and **Start** functions cannot be enabled simultaneously. For instance: if a **Jog +** command is sent to the Slave while it is moving to the target position, the jog command will be ignored; if **Jog +** and **Jog -** commands are sent simultaneously, the device will not move or, if already moving, it will stop its movement.
Should the device be disconnected from the Profibus network while it is moving (for instance because of a broken cable or a faulty wiring), the device will stop moving immediately and will enter the emergency status.

Parameter number (Byte 8)

This is reserved for setting the index of the parameter you need either to write or to read. If you need to set, i.e. to write, a value, it must be entered in the next four bytes.
For the complete list of the available parameters and their meaning please refer to the "7.9 Profibus® programming parameters" section on page 84.

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

Bit 6: always set to 0

Bit 7: R/W = 0 request to read the parameter
R/W = 1 request to write the parameter

EXAMPLE 1

You need to write a value next to the **0E Preset** parameter (index 0Eh), so set 0x8E = 1000 1110:

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

where:
bit 7 = R/W = 1, i.e. "writing the parameter"
bit 6 = bit always set to 0
bits 5 ... 0 = parameter index = 001110 binary value = 0Eh = **0E Preset**

The Preset value to be set in **0E Preset** must be entered in the next four bytes 9 ... 12 -**Parameter value (Bytes 9 ... 12)**.

EXAMPLE 2

You need to read the value next to the **10 Current velocity** parameter (index 10h), so set 0x10 = 0001 0000:

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

where:
bit 7 = R/W = 0, i.e. "reading the parameter"
bit 6 = bit always set to 0

bits 5 ... 0 = parameter index = 010000 binary value = 10h = **10 Current velocity**

The next four bytes 9 ... 12 -**Parameter value (Bytes 9 ... 12)**- will be ignored.

EXAMPLE 3
You need to read the value next to the **12 Motor Temperature [°C]** parameter (index 12h), so set 0x12 = 0001 0010:

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

where:
bit 7 = R/W = 0, i.e. "reading the parameter"
bit 6 = bit always set to 0
bits 5 ... 0 = parameter index = 010010 binary value = 12h = **12 Motor Temperature [°C]**

The next four bytes 9 ... 12 -**Parameter value (Bytes 9 ... 12)**- will be ignored.

Parameter value (Bytes 9 ... 12)

These bytes contain the value to be assigned to the parameter set in the previous byte 8. 4 data bytes are available for each parameter. If you send a request to read a parameter, these bytes will be ignored.
For the complete list of the available parameters and their meaning please refer to the "7.9 Profibus® programming parameters" section on page 84.

byte 9	byte 10	byte 11	byte 12
Low	...	...	High

WARNING
When you install **Lika RD6** module, the value of each parameter is uploaded at power-on from the GSD file which has been loaded in the PLC. Thus any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message will be temporary: when you turn off the power supply, the set value is lost (except the preset value which is the only variable not included in the GSD file; or unless you set the preset value previously which causes all parameters values to be saved automatically, see on page 91) and the value saved in the PLC will be uploaded at next power-on (thus all values, even if previously saved because of a preset setting, will be overwritten anyway).

This unit also allows to install the **Lika RD6-no param** module. Using this module, the unit does not read the parameters from the PLC when the power is switched on (they are read from the flash memory) and allows the operator to save any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message on the non-volatile memory (by means of the bit 9 **Save parameters** in the **Control Word (Bytes 0 and 1)**). When **Lika RD6-no param** module is installed, it is NOT possible to read and change the values of the **configuration data** parameters in the **Device-specific parameters** page of the TIA **Properties** window (see the "7.1.4 Device view" section on page 52). Thus you are allowed to enter new parameter values only through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message or by using the Modbus interface.

Using the RS-232 Modbus service serial interface it is possible to alter and then save the parameter values; in this way altered values are available again at next power-on. But this is true only until the serial interface is kept active: as soon as the unit is connected to the Profibus network, data in the PLC will be uploaded automatically and therefore the values set through the serial interface will be overwritten.

On the other hand using Siemens TIA PORTAL it is possible to alter any value and then save it permanently in the PLC (except in the -no param module). To save values permanently the operator has to enter the **Device-specific parameters** page of TIA and then alter the desired item (see the "7.1.4 Device view" section on page 52). Values altered in the Control Table of TIA (see the "7.1.13 Control Table" section on page 61) are temporary only.

Byte 13 Not used.

7.7.2 Slave →Master functions

In this section Data_Exchange messages sent by the Slave to the Master are described.

Status word (Bytes 0 and 1)

In this byte the status of the PI controller when the unit is in Data_Exchange operating state is shown.

Byte 0

Axis in position

bit 0 — The value is "=1" when the device reaches and keeps the commanded position (**Target position (Bytes 4 ... 7)**) for the time set next to the **02 Position window time** item. It is kept active until the position error is lower than **01 Position window**. For further information please refer to the "Positioning: position and speed control" section on page 45.

bit 1 — Not used.

Drive enabled

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

SW limit switch +

bit 3 — The value is "=1" should it happen that the device reaches the maximum positive limit (positive limit switch). For more information see the **08 Positive delta** 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 **09 Negative delta** parameter.

Alarm

bit 5 — The value is "=1" when an alarm occurs, see details in the **Alarms (Bytes 2 and 3)** variable.

Axis running

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

Executing a command

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

Byte 1

Target position reached

bit 8 — The value is "=1" when the device reaches the target position set next to the **Target position (Bytes 4 ... 7)** item (it is within the limits set next to the **01 Position window** parameter). The bit is kept active until a new **Target position (Bytes 4 ... 7)** or the **Alarm reset** commands are sent. For more information refer also to the "Positioning: position and speed control" section on page 45.

bits 9 ... 11 — Not used.

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 hindering the movement.

bits 13 ... 15 — Not used.

Alarms (Bytes 2 and 3)

These bytes are meant to show the alarms that are currently active in the device. The available error codes are listed hereafter:

Byte 2

Machine data not valid

bit 0 0001h: — 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 **19 Wrong parameters list** item.

Flash memory error

bit 1 0002h: Internal error, it cannot be restored.

Counting error

bit 2 0004h: For safety reasons, both the absolute encoder position and the incremental encoder position are read and saved to two separate registers. If any difference between the values in the registers is found the error is signalled.

Following error

bit 3 0008h: The difference between the real position and the theoretical position is greater than the value set in the **03 Max following error** parameter; we suggest reducing the work speed.

Axis not synchronized

bit 4 0010h Internal error, it cannot be restored.

Target not valid

bit 5 0020h: The set target position is over the maximum travel limits.

Emergency

bit 6 0040h The bit 7 **Emergency** in **Control Word (Bytes 0 and 1)** has been forced to low value (0); or alarms are active in the unit.

Overcurrent

bit 7 0080h The power supply current is exceeding the maximum ratings.

Byte 3

Electronics overtemperature

bit 8 0100h: The temperature of the MOSFETs detected by an internal probe is exceeding the maximum ratings (see **11 Electronics Temperature [°C]** on page 92). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the range.

Motor overtemperature

bit 9 0200h: The temperature of the motor detected by an internal probe is exceeding the maximum ratings (see **12 Motor Temperature [°C]** on page 92). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the range.

Undervoltage

bit 10	0400h	The power supply voltage is under the minimum ratings allowed. Please ensure that the power supply voltage is within the range.

Parameter number

bit 11	0800h	An invalid parameter number has been entered. See Parameter number (Byte 8) on page 76.

Read-only

bit 12	1000h	The parameter you tried to write to is a read-only parameter (ro parameter) and cannot be written to.
bit 13		Not used.

Hall sequence

bit 14	4000h	An error has been detected in the Hall sensors commutation sequence.

Overvoltage

bit 15	8000h	The power supply voltage is over the maximum ratings allowed. Please ensure that the power supply voltage is within the range. If the alarm is triggered during the braking operation, please consider the counter-electromotive force (back EMF). To prevent such situation from arising, decrease the deceleration ramp or evaluate attentively the characteristics of the 24V power supply pack (capacitor module).

To reset an alarm condition use the **Alarm reset** command, Control Word (Bytes 0 and 1) 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 19 Wrong parameters list), the normal work status can be restored only after having set proper values. Flash memory error and Axis not synchronized alarms cannot be reset.

Current position (Bytes 4 ... 7)

Current position of the device expressed in pulses. It shows the absolute position of the device in the moment when the message is sent.

Parameter number (Byte 8)

This byte contains the index of the parameter whose value is shown in the four following bytes.
Indexes are listed in the "7.9 Profibus® programming parameters" section on page 84.
For any information on sending a request to read or write a parameter, please refer to page 76 ff.

Parameter value (Bytes 9 ... 12)

These bytes contain the value assigned to the parameter whose number is shown in the previous byte 8. 4 data bytes are available for each parameter.
For the complete list of the available parameters and their meaning please refer to the "7.9 Profibus® programming parameters" section on page 84.
For any information on sending a request to read or write a parameter, please refer to page 76 ff.

byte 9	byte 10	byte 11	byte 12
Low	...	...	High

Byte 13 Not used

7.8 DDLM_Slave_Diag

Master device can send a request for diagnostic information to the Slave device at any time.
RD6 devices provide the 6-byte reduced diagnostic.

6-byte diagnostic:

Byte	Description
0	Status 1
1	Status 2
2	Status 3
3	Master ID
4	Manufacturer ID
5	

7.9 Profibus® programming parameters

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

Index Parameter name
[data types, attribute]

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

Data byte:

Data byte			
byte 9	byte 10	byte 11	byte 12
2^{31} to 2^{24}	2^{23} to 2^{16}	2^{15} to 2^{8}	2^{7} to 2^{0}
MSByte	...	...	LSByte

WARNING
When you install **Lika RD6** module, the value of each parameter is uploaded at power-on from the GSD file which has been loaded in the PLC. Thus any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message will be temporary: when you turn off the power supply, the set value is lost (except the preset value which is the only variable not included in the GSD file; or unless you set the preset value previously which causes all parameters values to be saved automatically, see on page 91) and the value saved in the PLC will be uploaded at next power-on (thus all values, even if previously saved because of a preset setting, will be overwritten anyway).

This unit also allows to install the **Lika RD6-no param** module. Using this module, the unit does not read the parameters from the PLC when the power is switched on (they are read from the flash memory) and allows the operator to save any new setting made locally through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)** items of the Data Exchange message on the non-volatile memory (by means of the bit 9 **Save parameters** in the **Control Word (Bytes 0 and 1)**). When **Lika RD6-no param** module is installed, it is NOT possible to read and change the values of the **configuration data** parameters in the **Device-specific parameters** page of the TIA **Properties** window (see the "7.1.4 Device view" section on page 52). Thus you are allowed to enter new parameter values only through **Parameter number (Byte 8)** and

Parameter value (Bytes 9 ... 12) items of the Data Exchange message or by using the Modbus interface.

Using the RS-232 Modbus service serial interface it is possible to alter and then save the parameter values; in this way altered values are available again at next power-on. But this is true only until the serial interface is kept active: as soon as the unit is connected to the Profibus network, data in the PLC will be uploaded automatically and therefore the values set through the serial interface will be overwritten.

On the other hand using Siemens TIA PORTAL it is possible to alter any value and then save it permanently in the PLC (except in the -no param module). To save values permanently the operator has to enter the **Device-specific parameters** page of TIA and then alter the desired item (see the "7.1.4 Device view" section on page 52). Values altered in the Control Table of TIA (see the "7.1.13 Control Table" section on page 61) are temporary only.

7.9.1 Configuration data parameters

00 Distance per revolution

[Unsigned16, rw]

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

Default = 4096 (min. = 1, max. = 4096).

WARNING

After having changed this parameter you must then set new values also next to the **0E Preset** parameter. For a detailed explanation see on page 46 and relevant parameters.

Please note that the parameters listed hereafter are closely related to the **00 Distance per revolution** parameter as they are all expressed in pulses; hence when you change the value in **00 Distance per revolution** also the value in each of them necessarily changes. They are: **01 Position window**, **03 Max following error**, **08 Positive delta**, **09 Negative delta** and **Target position (Bytes 4 ... 7)**. The **Current position (Bytes 4 ... 7)** value is also affected.

NOTE
If **00 Distance per revolution** is not a power of 2 (2, 4, ..., 2048, 4096), at position control a positioning error could occur having a value equal to one pulse.

01 Position window

[Unsigned16, rw]
This parameter defines the tolerance window limits for the **Target position (Bytes 4 ... 7)** value. As soon as the axis is within the tolerance window limits, the bit 8 **Target position reached** in the **Status word (Bytes 0 and 1)** goes high ("=1"). When the axis is within the tolerance window limits for the time set in the **02 Position window time** parameter, the bit 0 **Axis in position** in the **Status word (Bytes 0 and 1)** goes high ("=1"). This parameter is expressed in pulses. See also the "Positioning: position and speed control" section on page 45.
Default = 1 (min. = 0, max. = 65535).

02 Position window time

[Unsigned16, rw]
It represents the time for which the axis has to be within the tolerance window limits set in the **01 Position window** parameter before the state is signalled through the **Axis in position** status bit of the **Status word (Bytes 0 and 1)**. The parameter is expressed in milliseconds. See also the "Positioning: position and speed control" section on page 45.
Default = 0 (min. = 0, max. = 10000).

03 Max following error

[Unsigned32, rw]
This parameter defines the maximum allowable difference between the real position and the theoretical position of the device. If the device detects a value higher than the one set in this parameter, the **Following error** alarm is triggered and the unit stops. The parameter is expressed in pulses.
Default = 50000 (min. = 0, max. = 1000000).

04 Kp position loop

[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 = 80 (min. = 0, max. = 1000).

05 Ki position loop

[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).

06 Acceleration

[Unsigned16, rw]
This parameter defines the maximum acceleration value that has to be used by the device when reaching both the **0A Jog speed** and the **0B Work speed**. The parameter is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 44.
Default = 10 (min.=1, max.=500)

07 Deceleration

[Unsigned16, rw]
This parameter defines the maximum deceleration value that has to be used by the device when stopping. The parameter is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 44.
Default = 10 (min.=1, max.=500)

08 Positive delta

[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 (Bytes 0 and 1)** (the bit is forced high). The parameter is expressed in pulses.
SW limit switch + = **0E Preset** + **08 Positive delta**.
For further information please refer to the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.
Default = 134 213 631 (min.=0, max.=134 213 631)

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

EXAMPLE

When **00 Distance per revolution** = 1,024 and **0E Preset** = 0, the maximum acceptable value for **08 Positive delta** is:
(1,024 steps per revolution * 32,768 revolutions) – 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 33 553 407
When **00 Distance per revolution** = 256 and **0E Preset** = 0, the maximum acceptable value for **08 Positive delta** is:
(256 steps per revolution * 32,768 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 8 388 351
See further examples in the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.

WARNING

Every time **00 Distance per revolution** and **0E Preset** parameters are changed, **08 Positive delta** and **09 Negative delta** values have to be checked carefully. Each time you change the value in **00 Distance per revolution** you must then update the value in **0E Preset** in order to define the zero of the shaft as the system reference has now changed.
After having changed the parameter in **0E 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 **08 Positive delta** and **09 Negative delta** items. For a detailed explanation see on page 46.

09 Negative delta

[Unsigned32, rw]
This value is used to calculate the maximum backward (negative) limit the device is allowed to reach starting from the preset value. 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 (Bytes 0 and 1)** (the bit is forced high). The parameter is expressed in pulses.
SW limit switch - = **0E Preset** - **09 Negative delta**.
Default = 134 213 631 (min.=0, max.=134 213 631).

WARNING

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

EXAMPLE

When **00 Distance per revolution** = 1,024 and **0E Preset** = 0, the maximum acceptable value for **09 Negative delta** is:
(1,024 steps per revolution * 32,768 revolutions) - 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 33 553 407
When **00 Distance per revolution** = 256 and **0E Preset** = 0, the maximum acceptable value for **09 Negative delta** is:
(256 steps per revolution * 32,768 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 8 388 351
See further examples in the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.

WARNING

Every time **00 Distance per revolution** and **0E Preset** parameters are changed, **08 Positive delta** and **09 Negative delta** values have to be checked carefully. Each time you change the value in **00 Distance per revolution** you must then update the value in **0E Preset** in order to define the zero of the shaft as the system reference has now changed.
After having changed the parameter in **0E 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 **08 Positive delta** and **09 Negative delta** items. For a detailed explanation see on page 46.

0A Jog speed

[Unsigned16, rw]
This parameter contains the maximum speed the device is allowed to reach when using the **Jog +** and **Jog -** functions (see the **Control Word (Bytes 0 and 1)** parameter). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 44.
Default = 2000 (min. = 1, max. = 3000)

0B Work speed

[Unsigned16, rw]
This parameter contains the maximum speed the device is allowed to reach in automatic work mode (movements are controlled using **Start** and **Stop** commands -see the **Control Word (Bytes 0 and 1)** parameter- and are performed in order to reach the position set in **Target position (Bytes 4 ... 7)**). The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 45.
Default = 2000 (min. = 1, max. = 3000)

0C Code sequence

[Unsigned16, rw]

It sets whether the position value output by the device increases (count up information) when the shaft rotates clockwise (0) or counter-clockwise (1). Clockwise and counter-clockwise rotations are viewed from 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 **0E Preset** parameter and then check the values set next to the **08 Positive delta** and **09 Negative delta** items.

0D Jog step length

[Unsigned16, rw]

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

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

7.9.2 Operational data parameters

0E Preset

[Signed32, 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 preset value is activated. As soon as you set a new preset value next to this **0E Preset** parameter, the entered value is automatically saved and activated. If you do not change the **0E Preset** parameter value and need to activate it, then you must just switch the bit 11 **Setting the preset** in the **Control Word (Bytes 0 and 1)** from logic level low ("0") to logic level high ("1"). When the preset setting operation is carried out, the system also triggers an automatic save of all settings on the flash memory, the Offset value is also automatically stored in the memory. The value is expressed in pulses.
Default = 0 (min. = -268 435 456, max. = +268 435 456).

NOTE
We suggest activating the preset when the actuator is in stop. See the **Setting the preset** command on page 74.

WARNING
A new value must be set in **0E Preset** every time the **00 Distance per revolution** value is changed. After having entered a new value in **0E 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 **08 Positive delta** and **09 Negative delta** items. For a detailed explanation see on page 46.

0F Offset

[Signed32, ro]
This variable defines the difference between the position value transmitted by the device and the real position: real position – preset. Following a Preset operation, the Offset value is automatically stored in the memory. The value is expressed in pulses.

10 Current velocity

[Signed32, ro]
This parameter shows the current speed of the device updated at every second. The parameter is expressed in revolutions per minute (rpm).

11 Electronics Temperature [°C]

[Signed8, ro]
This variable shows the temperature of the electronics as detected by an internal probe. The value is expressed in Celsius degrees (°C). The minimum detectable temperature is -20°C.

12 Motor Temperature [°C]

[Signed8, ro]
This variable shows the temperature of the motor as detected by an internal probe. The value is expressed in Celsius degrees (°C). The minimum detectable temperature is -20°C.

13 Current value

[Signed32, ro]
This variable shows the value of the current absorbed by the motor (rated current). The value is expressed in milliamperes (mA).

14 Position following error

[Signed32, 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 **03 Max following error** parameter, then the **Following error** alarm is triggered and the unit stops. The value is expressed in pulses.

15 Software version

[Unsigned16, ro]
It contains the software version of the device.
The meaning of the 16 bits in the index is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB								LSB							
Major number								Minor number							

Value 258 in decimal notation corresponds to the binary representation 00000001 00000010 and has to be interpreted as: 01 02 hex, i.e. version 1.2.

16 Hardware version

[Unsigned16, ro]
It contains the hardware version of the device.
The meaning of the 16 bits in the index is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRIVECOD model								Hardware version							

where:

0 ... 7	= hardware version
8 ... 15	= RD6 model equipped with the following interface (0x10 = MODBUS; 0x11 = Profibus; 0x12 = CANopen; 0x13 = POWERLINK; 0x14 = EtherCAT)

Value 11 01 hex in hexadecimal notation corresponds to the binary representation 0001 0001 0000 0001 and has to be interpreted as follows: hardware version 1 (LSByte = 0x01); RD6 model with Profibus interface (MSByte = 0x11).

17 Positive absolute limit switch

[Signed32, ro]
This is the **SW limit switch +** value (maximum positive limit) calculated according to the values set next to the **0E Preset** and **08 Positive delta** parameters. When the maximum forward limit is reached, the condition is signalled through the **SW limit switch +** status bit 3 of the **Status word (Bytes 0 and 1)**.
SW limit switch + = **0E Preset** + **08 Positive delta**.
The value is expressed in pulses.
Refer also to the EXAMPLE 1 in the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.

18 Negative absolute limit switch

[Signed32, ro]
This is the **SW limit switch -** value (maximum negative limit) calculated according to the values set next to the **0E Preset** and **09 Negative delta** parameters. When the maximum backward limit is reached, the condition is signalled through the **SW limit switch -** status bit 4 of the **Status word (Bytes 0 and 1)**.
SW limit switch - = **0E Preset** - **09 Negative delta**.
The value is expressed in pulses.

Refer also to the EXAMPLE 1 in the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.

19 Wrong parameters list

[Unsigned32, ro]

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

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

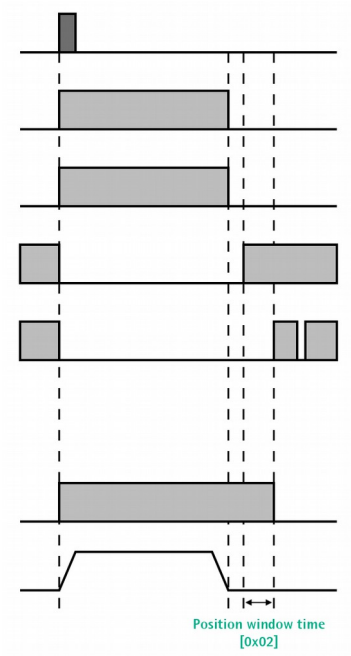
NOTE

The **19 Wrong parameters list** variable can be read by the PLC only if the RD unit goes online properly. On the other hand the DRIVECOD unit can go online only if the PLC sends correct parameters. If, for instance, you alter the value of **00 Distance per revolution** index directly in the GSD file without entering suitable values also next to the related items, the RD unit is not able to go online and thus it is not possible to enter this index and have information. The **19 Wrong parameters list** index can be read by the PLC only if you make changes, for instance, in the variables table. When you alter a parameter through **Parameter number (Byte 8)** and **Parameter value (Bytes 9 ... 12)**, the DRIVECOD unit is necessarily in **Data_Exchange** mode; so data exchange between the Master and the Slave is active and therefore it is possible to read this index. At power-on, the DRIVECOD unit enters the **Data_Exchange** mode only after finalizing the SET_PRM phase: this means that the configuration data has been sent properly to the Slave device.

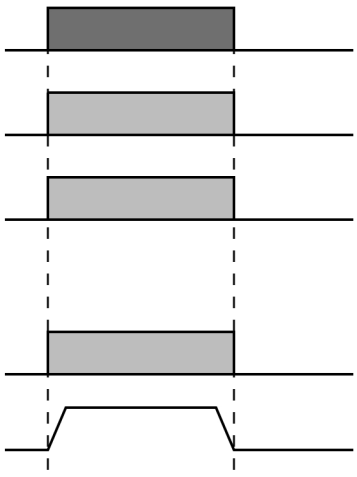
Bit	Parameter
0	Not used
1	00 Distance per revolution
2	06 Acceleration
3	07 Deceleration
4	08 Positive delta
5	09 Negative delta
6	0A Jog speed
7	0B Work speed
8	0C Code sequence
9	0E Preset

10	0D Jog step length
11	04 Kp position loop
12	05 Ki position loop
13	02 Position window time
14	03 Max following error
15	Not used

EXAMPLE 1

Positioning	RD6
Start (bit 6 - Control Word (Bytes 0 and 1)) **Executing a command** (bit 7 - Status word (Bytes 0 and 1)) **Axis running** (bit 6 - Status word (Bytes 0 and 1)) **Target position reached** (bit 8 - Status word (Bytes 0 and 1)) **Axis in position** (bit 0 - Status word (Bytes 0 and 1)) **Drive enabled** (bit 2 - Status word (Bytes 0 and 1): PWM is active) Theoretical velocity	

EXAMPLE 2

Jog +	RD6
Jog + (bit 0 - Control Word (Bytes 0 and 1)) **Executing a command** (bit 7 - Status word (Bytes 0 and 1)) **Axis running** (bit 6 - Status word (Bytes 0 and 1)) **Drive enabled** (bit 2 - Status word (Bytes 0 and 1): PWM is active) Theoretical velocity	

8 Modbus® interface

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

8.1 Configuring the device using Lika's setting up software

RD6 DRIVECOD positioning units can be equipped with several communication interfaces such as EtherCAT, POWERLINK, MODBUS RTU, Profibus-DP, CANopen DS 301 etc. All versions except the MODBUS RTU one are equipped with an RS-232 service serial port in compliance with the MODBUS protocol. It can be used to configure the actuator. For this purpose all versions are supplied with a software expressly developed and released by Lika Electronic in order to allow an easy set up of the device. The program allows the operator to set the working parameters of the device; control manually some movements and functions; and monitor whether the device is running properly. The program is supplied for free and can be installed in any PC fitted with a Windows operating system (Windows XP or later). The executable file to launch the program is **SW_RD6_MODBUS.EXE** and is available in the enclosed documentation or at the address **www.lika.biz > ROTARY ACTUATORS > ROTARY ACTUATORS/POSITIONING UNITS (DRIVECOD)**. The program is designed to be installed simply by copying the executable file to the desired location and there is **no installation** process. To launch it just double-click the file icon. To close the program press the **DISCONNECT** button in the **Serial Configuration** page and then click the **CLOSE** button in the title bar.

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

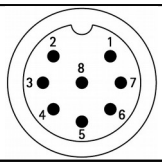
Compatibility	HW-SW	EXE Modbus
	1.0-1.0	from V1.2 to ...

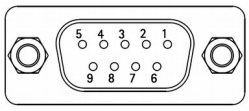
NOTE
Before starting the program, connect the device to the personal computer through serial ports. The serial interface of the DRIVECOD unit is a RS-485 type connector, while the serial standard in the personal computers (when available) is a RS-232 type connector. Therefore you must install a RS-485 / RS-232 converter, easily available in the market. Should the personal computer not be equipped with a serial port (RS-232 or RS-485), you must install a USB / RS-485 converter, easily available in the market too. For any information on the connection scheme and the cable pinout refer to the instruction sheet provided with the converter.

On the DRIVECOD side the cable must be connected to the M12 8-pin male connector service serial port. See the "Electrical connections" section on page 31.

A cable assembly fitted with M12 5-pin / USB connectors and integrated RS-485 converter is available on request; please contact Lika Electronic Technical Assistance & After Sale Service and quote the following code: **EXC-USB4-S54-GN-2-M12MC-S54**.

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





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

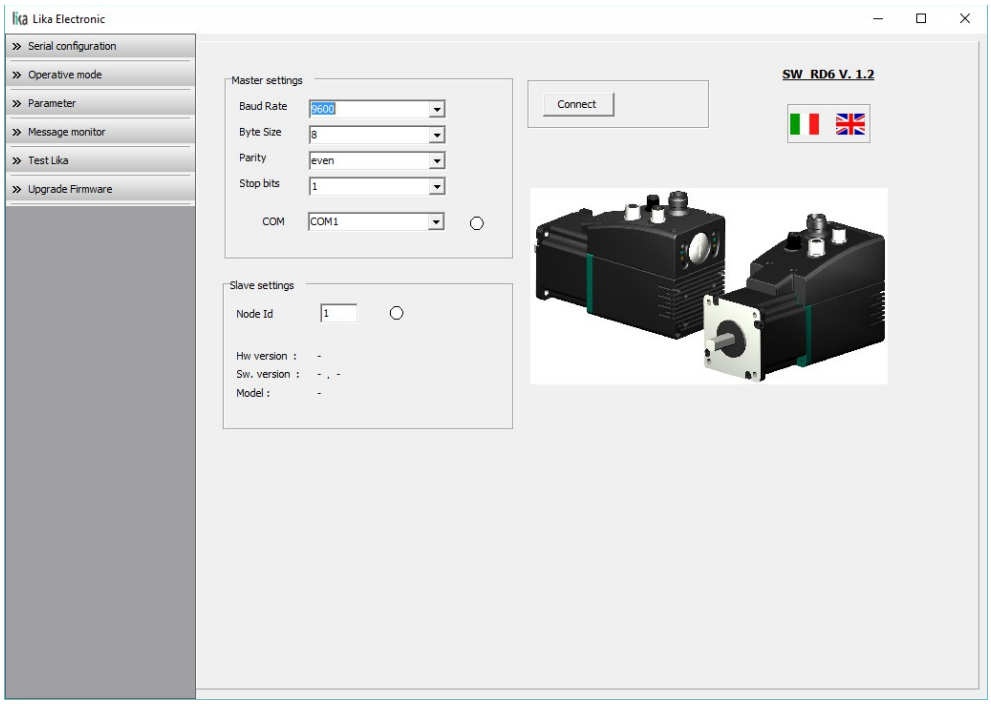
Please note that the configuration parameters of the MODBUS service serial port have fixed values, so the user cannot change them.
They are:

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

The MODBUS address is according to the DIP switch setting. To set the node address of the RD6 device refer to the "4.4.1 Setting the node address (Figure 7)" section on page 38. See also the "8.2 "Serial configuration" page" section right hereafter.

8.2 "Serial configuration" page

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



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

The **Master settings** box in the left side of the page allows you to choose the serial port of the personal computer the RD6 unit is connected to (**COM** drop-down box) and then set the configuration parameters. Serial port settings in the personal computer must compulsorily match those in the connected Lika device.

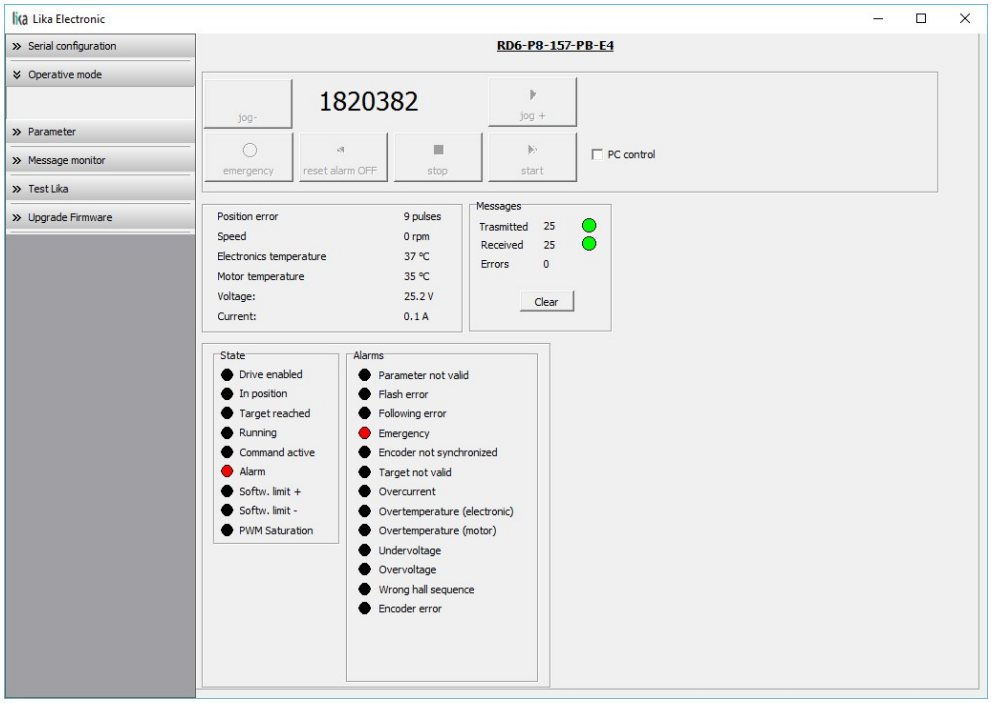
For serial port settings see the previous section.

Then set the node address of the device the personal computer is connected to through the **Slave settings** box (for all RD6-Modbus positioning units default value = **1**).
Now you are ready to establish the connection to the Slave: press the **CONNECT** button on the top of the page.

If the connection is established properly, two green lights placed next to the fields used to choose the **serial port** and set the **node ID** come on, while the **CONNECT** button disappears and is replaced by the **DISCONNECT** button. Furthermore the hardware version and the software version as well as the model of the device are shown in the **Slave settings** box.
The green light next to the **COM** item indicates that the COM port is open successfully.
The green light next to the **Node ID** item indicates that the DRIVECOD unit has been detected and the communication has been established successfully.

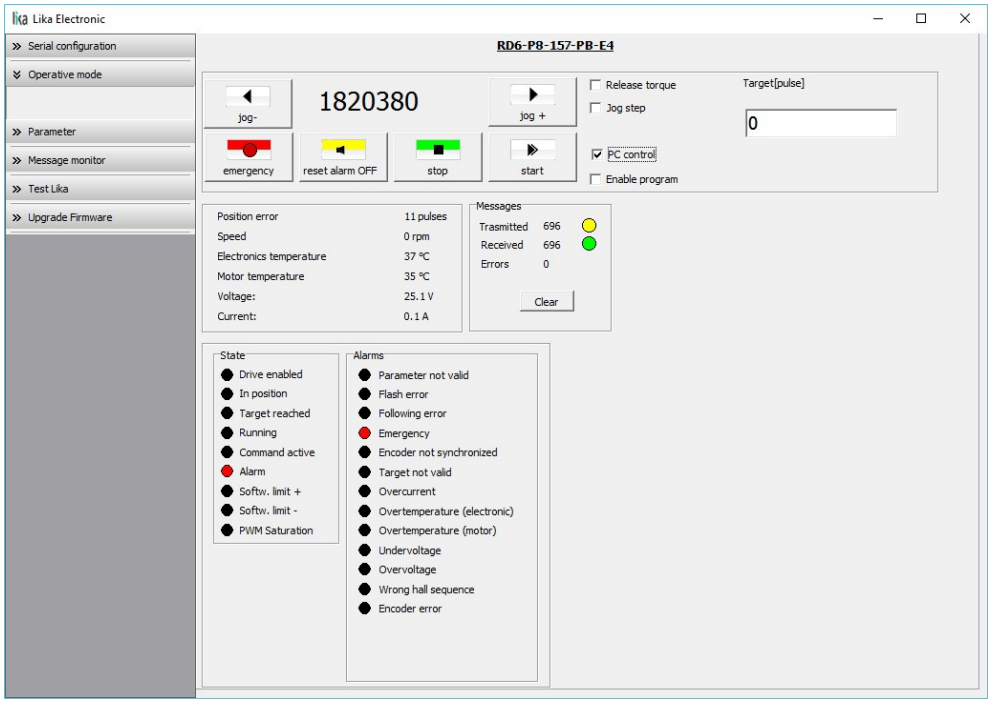
8.3 "Operative mode" page

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



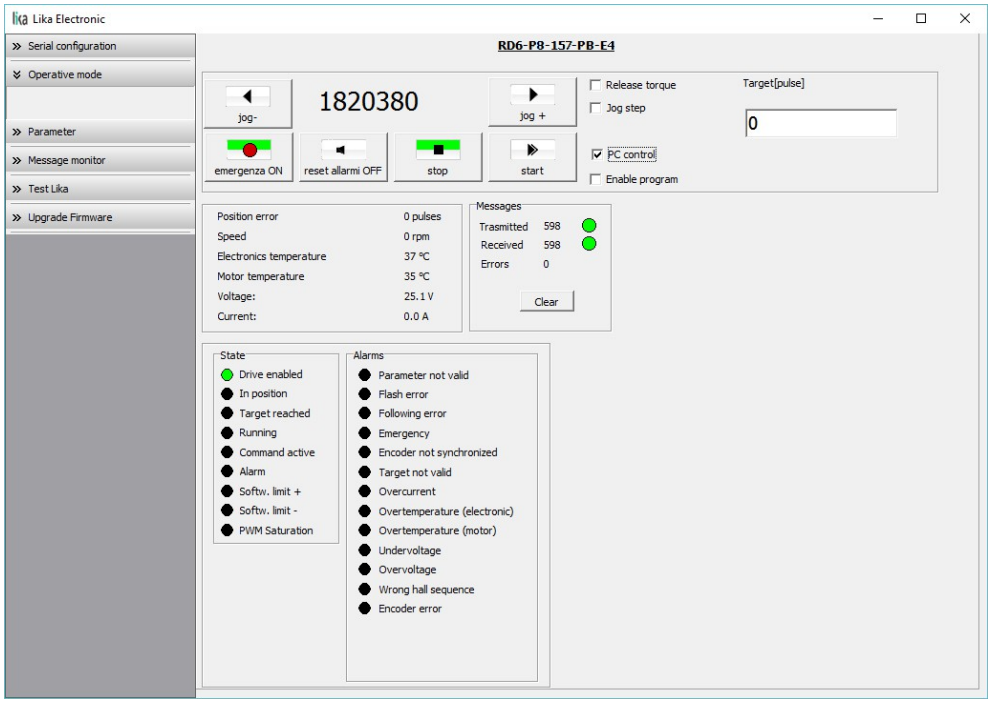
When you first enter the **Operative mode** page, all commands are disabled as the unit is still under Profibus network control. To start programming, controlling manually and monitoring the device through the RS-232 service serial interface in Modbus protocol, it is necessary to enable the available commands by gaining control of the unit in the Modbus network via PC. To do this, select the **PC CONTROL** check box (see **Extra commands register [0x29]** on page 136).

The commands will become available for use as shown in the following page:



When you first enter the **Operative mode** page, for safety reasons the RD6 unit is necessarily in an emergency condition: therefore the **Emergency** button is highlighted in red; furthermore the **Emergency** warning in the bottom left-hand **Alarms** box is lit red while the **Alarm** warning in the bottom left-hand **State** box flashes. To restore the **Idle** state of the device, press the **EMERGENCY** button first and then press the **RESET ALARM** button in this page. Alarm warnings will be reset.

The following page will appear:



In the top left-hand **RD6** box the following functions are available.

Jog -
See the **Jog -** item on page 137.

Pos [pulse]
See the **Current position [0x02-0x03]** item on page 146.

Jog +
See the **Jog +** item on page 137.

Emergency
When an emergency condition occurs, the **Emergency** button is highlighted in red; press the button to restore the normal work condition of the device. When the unit is running, press the button to force an immediate halt in emergency condition. See the **Emergency** item on page 139.

Reset alarm
If an alarm is active, the **RESET ALARM** button is highlighted in yellow; press the button to reset the alarm. See the **Alarm reset** item on page 138.

Stop
Press this button to force a normal halt of the device, respecting the acceleration and deceleration values. See the **Stop** item on page 138.

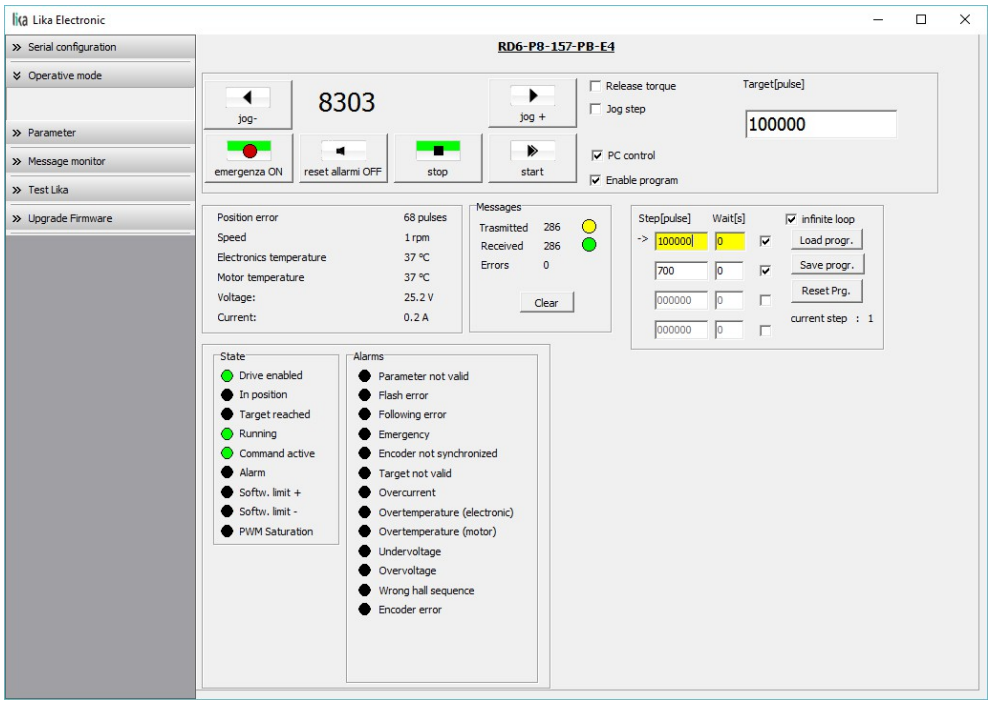
Start
Pressing the button causes the unit to start running in order to reach the position set next to the **Target [pulse]** item. As soon as the commanded position is reached, the device stops and activates the **Axis in position** and **Target position reached** status bits. For a normal halt of the device press the **STOP** button; for an immediate emergency halt press the **EMERGENCY** button. See the **Start** item on page 138.

Release torque
See the **Release axis torque** item on page 140.

Jog step
See the **Incremental jog** item on page 138.

Target [pulse]
See the **Target position [0x2B-0x2C]** item on page 140. Set the position you need the unit to reach and then press the **ENTER** key in the keyboard to confirm it. As soon as you press the **START** button the device starts moving in order to reach the commanded position set next to this **Target [P]** item, then it stops and activates the **Axis in position** and **Target position reached** status bits.

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



The functions available in the **Program** box allow the operator to create and then save work programs for the RD6 unit.
The positions that the device is commanded to reach (target positions) must be set next to the **STEP [pulse]** items; it is possible to enter up to four subsequent positions. Next to the **WAIT [s]** items you must set the interval between one step (commanded movement) and the next. All set values must be confirmed by pressing the **ENTER** key in the keyboard. Before entering a value, each field must be previously enabled by selecting the check box on the right.
The **INFINITE LOOP** check box allows the operator to activate the "infinite loop" function, i.e. the device goes on running and executing the set steps without interruption.

If the **INFINITE LOOP** check box is selected, when you press the **START** button, the device starts moving in order to reach the first commanded position; **STEP [pulse]** and **WAIT [s]** items are highlighted in yellow; as soon as the commanded position set next to the **STEP [pulse]** item is reached, the device stops and the field is highlighted in green, as soon as the set interval has expired (a backward counter is displayed) also the **WAIT [s]** field is highlighted in green and the RD6 unit restarts running in order to reach the second commanded position; and so on, from the first to the fourth commanded position (if enabled) and then again from the first to the fourth commanded position without interruption, until you press the **STOP** button.

If the **INFINITE LOOP** check box is not selected, when you press the **START** button, the device starts running in order to reach the first commanded position; as soon as the commanded position is reached, the device stops and waits for the set interval to expire; you must then press the **START** button again to command the unit to reach the second position; and so on.

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

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

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

RESET PRG. button zero-sets the counter meant to detect the steps in the execution of the running program: when the operator presses the **START** button the device will start running from step 1, i.e. in order to reach the first commanded position, whatever the position reached previously.

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

In the box just below the **RD6** box the following functions are available.

Position error [pulses]
See the **Position following error [0x05-0x06]** item on page 146.

Speed [rpm]
See the **Current velocity [0x04]** item on page 146.

Electronics temperature [°C]
See the **Temperature value [0x07]** item on page 146.

Motor temperature [°C]
See the **Temperature value [0x07]** item on page 146.

Voltage [V]
See the **Motor voltage [0x0A]** item on page 147.

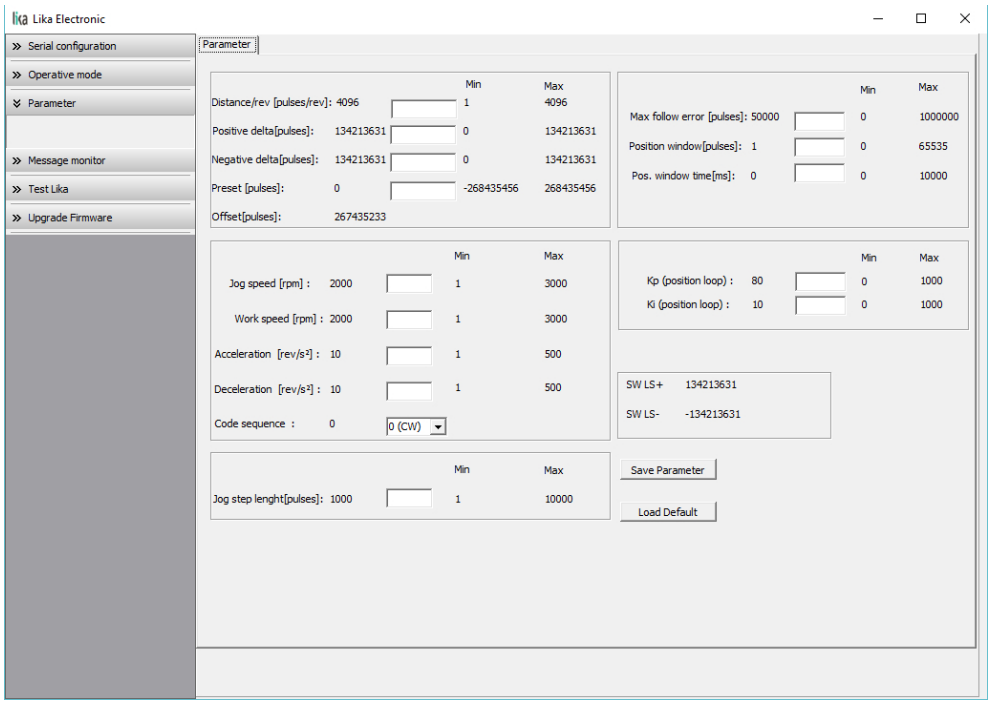
Current [A]
It shows the value of the current absorbed by the motor (rated current). The value is expressed in amperes (A). See the **Current value [0x0B]** item on page 148.

The right-hand **Messages** box allows the operator to have a brief description of the communication between the Master and the Slave, by displaying the Request PDU (Transmitted) and the Response PDU (Received) messages. The fields in the box are meant to show the number of transmitted messages, the number of received messages and the errors: **Transmitted** = Request PDUs; **Received** = Response PDUs; **Errors** = Exception Response PDUs. For complete information on the communication between the Master and the Slave, please refer to the **Message monitor** page (see the "8.5 "Message monitor" page" section on page 109).

In the bottom left-hand **States** and **Alarms** boxes the list of states and alarms available for the RD6 unit is displayed. Active states are highlighted in green; while active alarms are highlighted in red. For a detailed description of the states see the **Status word [0x01]** item on page 144; for a detailed description of the alarms see the **Alarms register [0x00]** item on page 142.

8.4 "Parameter" page

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



In this page the list of the parameters available to set the RD6 positioning units (machine data) is displayed. On the left of each field the values currently loaded in the unit are shown; while the minimum and maximum values allowed are shown on the right. For detailed information on the function and the setting of each parameter refer to the "8.12.1 Holding Register parameters" section on page 129.
To enter a new value type it in the blank field and then press the **ENTER** key in the keyboard. If you set a value that is not allowed (out of range), at confirmation prompt the field is highlighted in red and the RD6 unit is forced in alarm condition (the **Alarm** status bit is activated and the **Machine data not valid** and/or **Emergency** error messages are invoked to appear). Enter a valid value and then press the **RESET ALARM** button in the **Operative mode** page to restore the normal work condition of the device.

To save the entered values on the non-volatile memory of the device press the **SAVE PARAMETER** button. If the power is turned off before saving the values all data not saved will be lost! For any further information on saving the parameters refer to the **Save parameters** variable on page 139.

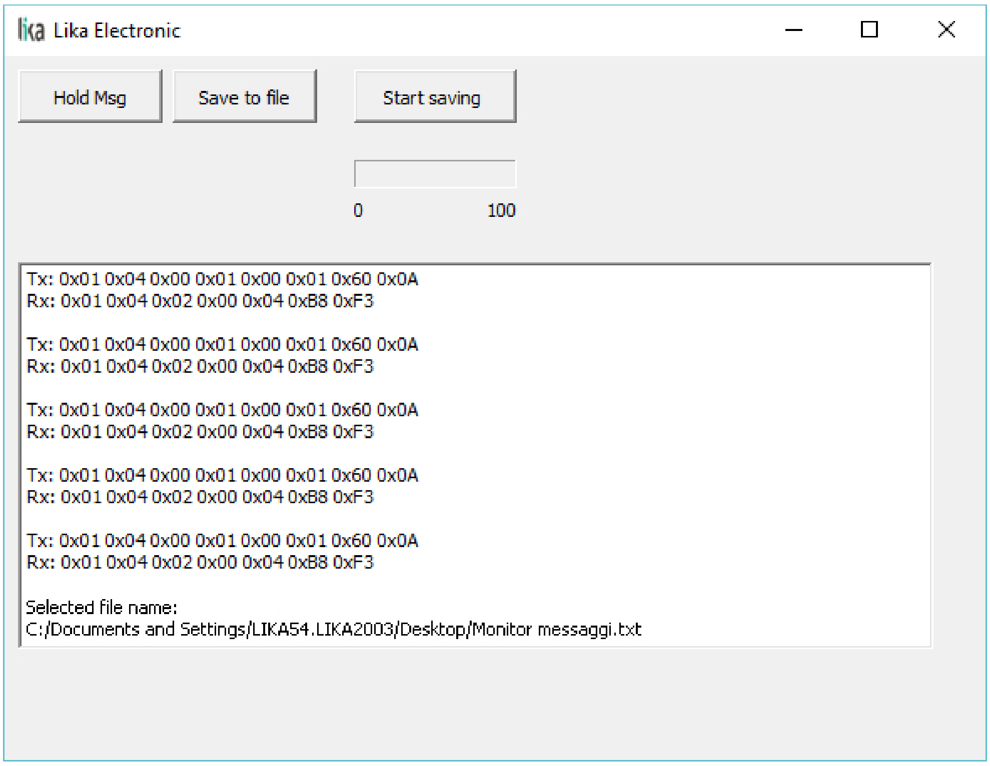
When you need to load the default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) press the **LOAD DEFAULT** button. For any further information on loading the default parameters refer to the **Load default parameters** variable on page 139. The complete list of the machine data parameters and the relevant default values as set by Lika Electronic are available on page 158.

SW LS + / SW LS -

They are available in the box over the **SAVE PARAMETER** and **LOAD DEFAULT** buttons. They show visually the positive and negative limit switch values. See the **Positive delta [0x09-0x0A]** item on page 132 and the **Negative delta [0x0B-0x0C]** item on page 133.

8.5 "Message monitor" page

By pressing the **MESSAGE MONITOR** button in the sidebar menu the operator enters the **Message monitor** page.



This page allows the operator to monitor the communication between the Master and the Slave, by displaying the Request PDU (Tx) and the Response PDU (Rx) messages. When you first enter the page, the field meant to show the messages is blank.
Press the **VIEW MSG** button to display the flow of messages. Once you press the button, data throughput rate between the Master and the Slave starts appearing on the screen. The messages are displayed in hexadecimal notation. After pressing the **VIEW MSG** button, its descriptive label is replaced by the **HOLD MSG** label. Press the **HOLD MSG** button to stop the flow of messages.

You can save the messages to a text file. As soon as you press the **SAVE TO FILE** button, the **Open the log file** dialog box appears on the screen: the operator must type the .txt file name and specify the path where the file has to be located. When you press the **OPEN** button to confirm, the dialog box closes and the full path of the selected file is shown in the display box of the **Message monitor** page. Now press the **START SAVING** button to start saving the messages; the "File opened properly" message appears on the display box. After pressing the **START SAVING** button, its descriptive label is replaced by **STOP SAVING** label. Press the **STOP SAVING** button to stop saving the messages.

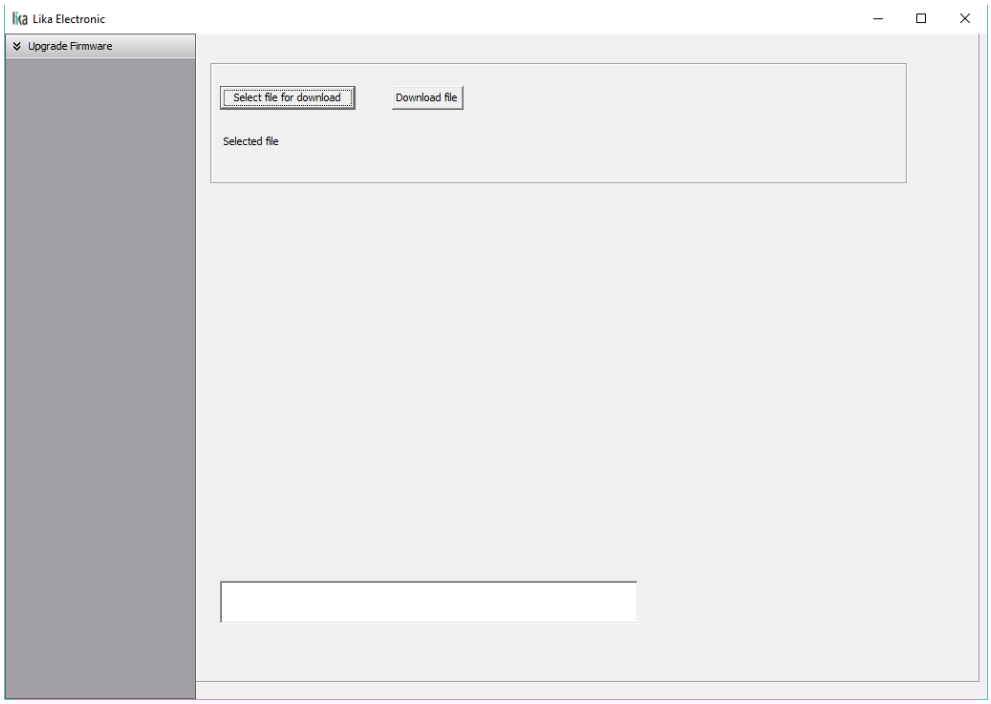
A box in the **Parameter** page offers a brief description of the communication between the Master and the Slave, by displaying the Request PDU (Transmitted) and the Response PDU (Received) messages. The fields in the box are meant to show the number of transmitted messages, the number of received messages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

8.6 "Test Lika" page

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

8.7 "Upgrade Firmware" page

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



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

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

If the latest firmware version is already installed in the DRIVECOD unit, you do not need to proceed with any new firmware installation. Current firmware version can be verified from the **SW VERSION** item in the **Slave settings** box of the **Serial configuration** page after connecting properly to the unit (see on page 99).

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 Modbus RS-232 service port" section on page 34;
2. make sure that the DRIVECOD unit you need to update is the only node connected to the personal computer;
3. connect to the unit, go online and then enter the **Upgrade Firmware** page;
4. when you switch on the power supply, if user program is not present in the flash memory, you are not able to connect to the unit through the **Serial configuration** page; when this happens you need to enter directly the **Upgrade Firmware** page; make sure that the correct serial port of the personal computer connected to the DRIVECOD unit is selected in the **Serial configuration** page; for any further information please refer to the "8.7.1 If an installation issue occurs" section;
5. press the **SELECT FILE FOR DOWNLOAD** button; once you press the button the **Open** dialogue box appears on the screen: the operator must open the folder where the firmware upgrading .BIN file released by Lika Electronic is located;

WARNING

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

For instance: RD6_157_MB_H1S1.0.BIN, where:

- RD6_157 = DRIVECOD unit model;
- MB = bus interface of the DRIVECOD unit (CB = CANopen; EC = EtherCAT; MB = MODBUS; PB = Profibus; PL = POWERLINK);
- H1 = hardware version;
- S1.0 = 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 just confirmed appears next to the **SELECTED FILE** item;
8. now press the **DOWNLOAD FILE** button to start the firmware upgrading process;
9. a download progress bar is displayed in the centre of the page;

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

10. as soon as the operation is carried out successfully, the **UPGRADE INSTALLATION COMPLETED SUCCESSFULLY** message is displayed;
11. the DRIVECOD unit is now in an emergency condition;
12. close and then restart the SW_RD6_MODBUS_X.EXE program; connect to the DRIVECOD unit and restore the normal work condition through the **Operative mode** page.

8.7.1 If an installation issue occurs

While downloading the firmware upgrading program, unexpected conditions may arise which could lead to a failure of the installation process. When such a matter occurs, the download process cannot be carried out successfully and thus the operation is aborted.

CONDITION 1
While downloading data to the flash memory for upgrading the firmware of the unit, if an error occurs which stops the upgrading process (for instance: a voltage drop and/or the switching off of the DRIVECOD unit), the COMMUNICATION ERROR. UPGRADE INSTALLATION ABORTED warning message is invoked to appear in the box in the bottom of the **Upgrade Firmware** page. The upgrading process is necessarily aborted and the unit cannot work as the firmware has been deleted before starting the update. At next power-on the unit is out of order because the user program is not installed in the flash memory. To restore the work condition of the unit, the operator must close and then restart the SW_RD6_MODBUS_X.EXE program. It is not possible to connect to the unit through the **Serial configuration** page; the operator must enter the **Upgrade Firmware** page and restart the firmware installation process from point 6. Always make sure that the correct serial port of the personal computer connected to the DRIVECOD unit is selected in the **Serial configuration** page.

CONDITION 2
While downloading data to the flash memory for upgrading the firmware of the unit, if data transmission is cut off (for instance, due to the disconnection of the serial cable or because the process is terminated), the **COMMUNICATION ERROR. UPGRADE INSTALLATION ABORTED** warning message is invoked to appear in the box in the bottom of the **Upgrade Firmware** page. The upgrading process is necessarily aborted and the unit cannot work as the firmware has been deleted before starting the update. To restore the work condition of the unit, the operator must shut down and then switch on the DRIVECOD unit first, then close and restart the SW_RD6_MODBUS_X.EXE program. At next power-on the unit is out of order because the user program is not installed in the flash memory. It is not possible to connect to the unit through the **Serial configuration** page; the operator must enter the **Upgrade Firmware** page and restart the firmware installation process from point 6. Always make sure that the correct serial port of the personal computer connected to the DRIVECOD unit is selected in the **Serial configuration** page.

8.8 Modbus Master / Slaves protocol principle

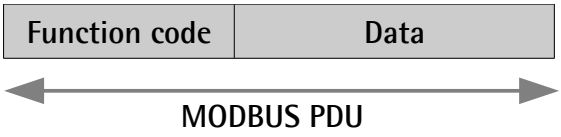
The Modbus Serial Line protocol is a Master – Slaves protocol. One only Master (at the same time) is connected to the bus and one or several (up to 247) Slave nodes are also connected to the same serial bus. A Modbus communication is always initiated by the Master. The Slave nodes will never transmit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

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

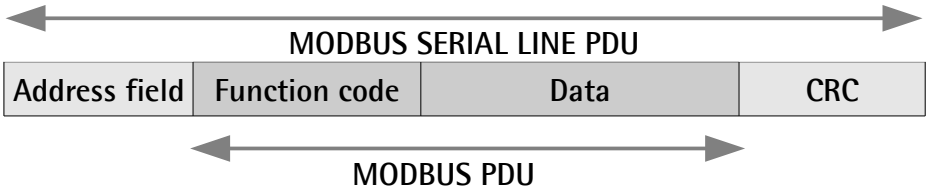
- **UNICAST mode**: in that mode the Master addresses an individual Slave. After receiving and processing the request, the Slave returns a message (a "reply") to the Master. In that mode, a Modbus transaction consists of two messages: a request from the Master and a reply from the Slave. Each Slave must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

8.9 Modbus frame description

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



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



- **ADDRESS FIELD**: on Modbus Serial Line the address field only contains the Slave address. As previously stated (see the "8.8 Modbus Master / Slaves protocol principle" section on page 114), the valid Slave node addresses are in the range of 0 – 247 decimal. The individual Slave devices are assigned addresses in the range of 1 – 247. A Master addresses a Slave by placing the Slave address in the **ADDRESS FIELD** of the message. When the Slave returns its response, it places its own address in the response **ADDRESS FIELD** to let the Master know which Slave is responding.
- **FUNCTION CODE**: the function code indicates to the Server what kind of action to perform. The function code can be followed by a **DATA** field that contains request and response parameters. For any further information on the implemented function codes refer to the "8.11 Function codes" section on page 119.
- **DATA**: the **DATA** field of messages contains the bytes for additional information and transmission specifications that the server uses to take the action defined by the **FUNCTION CODE**. This can include items such as discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the field. The structure of the **DATA** field depends on each **FUNCTION CODE** (refer to the "8.11 Function codes" section on page 119).
- **CRC (Cyclic Redundancy Check)**: error checking field is the result of a "Redundancy Check" calculation that is performed on the message contents. This is intended to check whether transmission has been performed properly. The CRC field is two bytes, containing 16-bit binary

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

The Modbus protocol defines three PDUs. They are:

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

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

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

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

8.10 Transmission modes

Two different serial transmission modes are defined in the Modbus serial protocol: the **RTU** (**Remote Terminal Unit**) **mode** and the **ASCII mode**. The transmission mode defines the bit contents of message fields transmitted serially on the line. It determines how information is packed into the message fields and decoded. The transmission mode and the serial port parameters must be the same for all devices on a Modbus Serial Line. All devices must implement the RTU mode, while the ASCII mode is an option. Lika devices only implement RTU transmission mode, as described in the following section.

8.10.1 RTU transmission mode

When devices communicate on a Modbus serial line using the RTU (Remote Terminal Unit) mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters. Each message must be transmitted in a continuous stream of characters. Synchronization between the messages exchanged by the transmitting device and the receiving device is achieved by placing an interval of at least 3.5 character times between successive messages, this is called "silent interval". In this way a Modbus message is placed by the transmitting device into a frame that has a known beginning and ending point. This allows devices that receive a new frame to begin at the start of the message and to know when the message is completed. So when the receiving device does not receive a message for an interval of 4 character times, it considers the previous message as completed and the next byte will be the first of a new message, i.e. an address.
When baud rate = 9600 bit/s the "silent interval" is 4 ms.
When baud rate = 19200 bit/s the "silent interval" is 2 ms.

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

Coding system: 8-bit binary
Bits per Byte: 1 start bit;
8 data bits, least significant bit (lsb) sent first;
1 bit for parity completion (= Even);
1 stop bit.

Modbus protocol uses a "big-Endian" representation for addresses and data items. This means that when a numerical quantity greater than a single byte is transmitted, the most significant byte (MSB) is sent first.
Each character or byte is sent in this order (left to right):

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

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

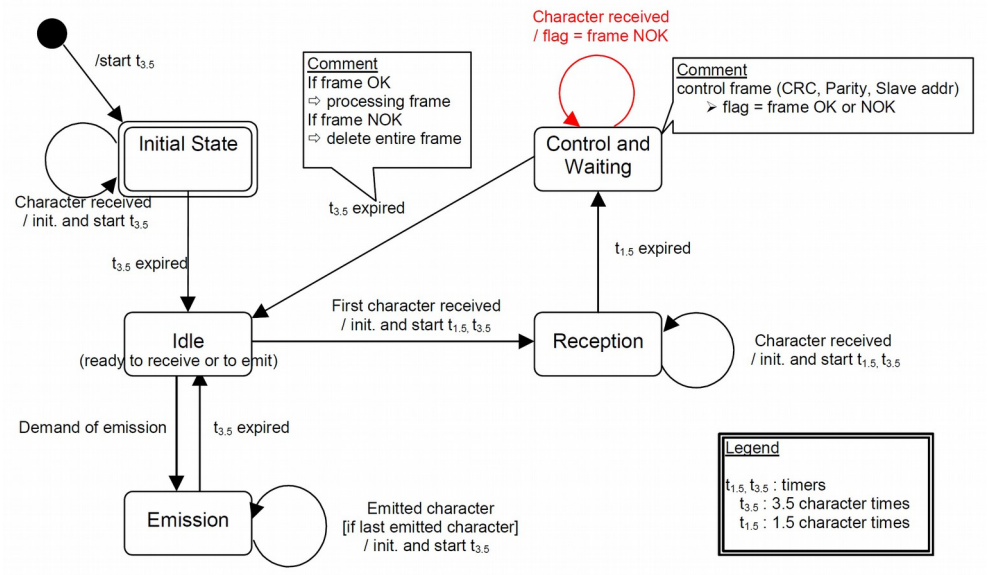
* When "No parity" is activated, the parity bit is replaced by a stop bit.

The default parity mode must be even parity.

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

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

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



- Transition from **Initial State** to **Idle** state needs an interval of at least 3.5 character times (time-out expiration = $t_{3.5}$).
- **Idle** state is the normal state when neither emission nor reception is active. In RTU mode, the communication link is declared in **Idle** state when there is no transmission activity after a time interval equal to at least 3.5 characters ($t_{3.5}$).
- A request can only be sent in **Idle** state. After sending a request, the Master leaves the **Idle** state and cannot send a second request at the same time.
- When the link is in **Idle** state, each transmitted character detected on the link is identified as the start of the frame. The link goes to **Active** state. Then the end of the frame is identified when no more character is transmitted on the link after the time interval of at least $t_{3.5}$.
- After detection of the end of frame, the CRC calculation and checking is completed. Afterwards the address field is analysed to determine if the frame is addressed to the device. If not, the frame is discarded. In order to reduce the reception processing time the address field can be analysed as soon as it is received without waiting the end of frame. In this case the CRC will be calculated and checked only if the frame is actually addressed to the Slave.

8.11 Function codes

As previously stated, the function code indicates to the Server what kind of action to perform. The function code field of a Modbus data unit is coded in one byte. Valid codes are in the range of 1 ... 255 decimal (the range 128 ... 255 is reserved and used for Exception Responses). When a message is sent from a Client to a Server device the function code field tells the Server what kind of action to perform. Function code "0" is not valid.
There are three categories of Modbus function codes, they are: **public function codes**, **user-defined function codes** and **reserved function codes**.
Public function codes are in the range 1 ... 64, 73 ... 99 and 111 ... 127; they are well defined function codes, validated by the MODBUS-IDA.org community and publicly documented; furthermore they are guaranteed to be unique. Ranges of function codes from 65 to 72 and from 100 to 110 are **user-defined function codes**: user can select and implement a function code that is not supported by the specification, it is clear that there is no guarantee that the use of the selected function code will be unique. **Reserved function codes** are not available for public use.

8.11.1 Implemented function codes

Lika RD6 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 "8.12.1 Holding Register parameters" section on page 129.

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	**03**
Register 8 value Lo	**E8**
Register 9 value Hi	**05**
Register 9 value Lo	**DC**

As you can see in the table, **Acceleration [0x07]** parameter (register 8) contains the value 03 E8 hex, i.e. 1000 in decimal notation; **Deceleration [0x08]** parameter (register 9) contains the value 05 DC hex, i.e. 1500 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][03][E8][05][DC][78][8A]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[03][E8] = value of register 8 **Acceleration [0x07]**, 03 E8 hex = 1000 dec
[05][DC] = value of register 9 **Deceleration [0x08]**, 05 DC hex = 1500 dec
[78][8A] = CRC

04 Read Input Register

FC = 04 (Hex = 0x04)

This function code is used to READ from 1 to 125 contiguous input registers in a remote device; in other words, it allows to read some results values and state / alarm messages in a remote device. The Request PDU specifies the starting register address and the number of registers. In the PDU registers are addressed starting at zero. Therefore input registers numbered 1-16 are addressed as 0-15.
The register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits (msb) and the second contains the low order bits (lsb).
For the complete list of input registers accessible using **04 Read Input Register** function code please refer to the "8.12.2 Input Register parameters" section on page 142.

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 "8.12.1 Holding Register parameters" section on page 129.

Request PDU

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

Response PDU

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

Exception Response PDU

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

Here is an example of a request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [0x07]** parameter (register 8).

Request	
Field name	(Hex)
Function	**06**
Register address Hi	**00**
Register address Lo	**07**
Register value Hi	**05**
Register value Lo	**DC**

Response	
Field name	(Hex)
Function	**06**
Register address Hi	**00**
Register address Lo	**07**
Register value Hi	**05**
Register value Lo	**DC**

As you can see in the table, the value 05 DC hex, i.e. 1500 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8).

The full frame needed for the request to write the value 05 DC hex (= 1500 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][05][DC][3A][C2]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[05][DC] = value to be set in the register
[3A][C2] = CRC

The full frame needed to send back a response following the request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [0x07]** parameter (register 8) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][06][00][07][05][DC][3A][C2]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[05][DC] = value set in the register
[3A][C2] = CRC

16 Write Multiple Registers

FC = 16 (Hex = 0x10)

This function code is used to WRITE a block of contiguous registers (1 to 123 registers) in a remote device.
The values to be written are specified in the request data field. Data is packed as two bytes per register.
The normal response returns the function code, starting address and quantity of written registers.
For the complete list of registers accessible using **16 Write Multiple Registers** function code please refer to the "8.12.1 Holding Register parameters" section on page 129.

Request PDU

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

*N = Quantity of registers

Response PDU

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

Exception Response PDU

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

Here is an example of a request to write the values 1500 and 1000 dec in the parameters **Acceleration [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	**05**

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

Register 8 value Lo	**DC**
Register 9 value Hi	**03**
Register 9 value Lo	**E8**

As you can see in the table, the value 05 DC hex, i.e. 1500 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8); the value 03 E8 hex, i.e. 1000 in decimal notation, is set in the **Deceleration [0x08]** parameter (register 9).

The full frame needed for the request to write the values 05 DC hex (= 1500 dec) and 03 E8 hex (= 1000 dec) in the parameters **Acceleration [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][05][DC][03][E8][73][C1]
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)
[05][DC] = value to be set in the register 8 **Acceleration [0x07]**, 05 DC hex = 1000 dec
[03][E8] = value to be set in the register 9 **Deceleration [0x08]**, 03 E8 hex = 1500 dec
[73][C1] = CRC

The full frame needed to send back a response following the request to write the values 05 DC hex (= 1500 dec) and 03 E8 hex (= 1000 dec) in the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) from the Slave having the node address 1 is as follows:

Response PDU (in hexadecimal format)
[01][10][00][07][00][02][F0][09]
where:
[01] = Slave address
[10] = **16 Write Multiple Registers** function code
[00][07] = starting address (**Acceleration [0x07]** parameter, register 8)
[00][02] = number of written registers
[F0][09] = CRC

NOTE
For further examples refer to the "8.14 Programming examples" section on page 152.

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

8.12 Programming parameters

Hereafter the parameters available for RD6 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 encoder parameters can be 1-register long, i.e. 16-bit long, or 2-register long, i.e. 32-bit long.

Unsigned16 parameter structure:

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

Unsigned32 parameter structure:

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

8.12.1 Holding Register parameters

Holding registers (**Machine data** parameters) are accessible for both writing and reading; to read the value set in a parameter use the **03 Read Holding Registers** function code (reading of multiple registers); to write a value in a parameter use the **06 Write Single Register** function code (writing of a single register) or the **16 Write Multiple Registers** (writing of multiple registers); for any further information on the implemented function codes refer to the "8.11.1 Implemented function codes" section on page 119.

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 137.
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 pitch; by setting Distance per revolution [0x00] = 500, at each shaft revolution the system performs a 5 mm pitch with one-hundredth of a millimetre resolution.
Default = 4096 (min. = 1, max. = 4096)

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 46 and relevant parameters.
Please note that the parameters listed hereafter are closely related to the Distance per revolution [0x00] parameter as they are all expressed in pulses; hence when you change the value in Distance per revolution [0x00] also the value in each of them necessarily changes. They are: Position window [0x01], Max following error [0x03-0x04], Positive delta [0x09-0x0A], Negative delta [0x0B-0x0C], Jog step length [0x14] and Target position [0x2B-0x2C]. The Current position [0x02-0x03] value is also affected.

NOTE
If Distance per revolution [0x00] is not a power of 2 (2, 4, ..., 2048, 4096), at position control a positioning error could occur having a value equal to one pulse.

Position window [0x01]

[Register 2, Unsigned16, rw]
This parameter defines the tolerance window limits for the **Target position [0x2B-0x2C]** value. As soon as the axis is within the tolerance window limits, the bit 8 **Target position reached** in the **Status word [0x01]** goes high ("=1"). When the axis is within the tolerance window limits for the time set in the **Position window time [0x02]** parameter, the bit 0 **Axis in position** in the **Status word [0x01]** goes high ("=1"). The parameter is expressed in pulses. See also the "Positioning: position and speed control" section on page 45.
Default = 1 (min. = 0, max. = 65535)

Position window time [0x02]

[Register 3, Unsigned16, rw]
It represents the time for which the axis has to be within the tolerance window limits set in the **Position window [0x01]** parameter before the state is signalled through the **Axis in position** status bit of the **Status word [0x01]**. The parameter is expressed in milliseconds. See also the "Positioning: position and speed control" section on page 45.
Default = 0 (min. = 0, max. = 10000)

Max following error [0x03-0x04]

[Registers 4-5, Unsigned32, rw]
This parameter defines the maximum allowable difference between the real position and the theoretical position of the device. If the device detects a value higher than the one set in this parameter, the **Following error** alarm is triggered and the unit stops. The parameter is expressed in pulses.
Default = 50000 (min. = 0, max. = 1000000)

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 = 80 (min. = 0, max. = 1000)

Ki position loop [0x06]

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

Acceleration [0x07]

[Register 8, Unsigned16, rw]
This parameter defines the maximum acceleration value that has to be used by the device when reaching both the **Jog speed [0x0D]** and the **Work speed [0x0E]**. The parameter is expressed in revolutions per second^2 [rev/s^2]. See also the "6.2 Movements: jog and positioning" section on page 44.
Default = 10 (min. = 1, max. = 500)

Deceleration [0x08]

[Register 9, Unsigned16, rw]
This parameter defines the maximum deceleration value that has to be used by the device 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 44.
Default = 10 (min. = 1, max. = 500)

Positive delta [0x09-0x0A]

[Registers 10-11, Unsigned32, rw]
This value is used to calculate the maximum forward (positive) limit the device is allowed to reach starting from the preset value. Should it happen that the maximum forward limit is reached, a signalling is activated through the **SW limit switch +** status bit of the **Status word [0x01]** (the bit is forced high). The parameter is expressed in pulses.
SW limit switch + = **Preset [0x12-0x13]** + **Positive delta [0x09-0x0A]**.
For further information please refer to the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.
Default = 134 213 631 (min. = 0, max. = 134 213 631)

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 * 32,768 revolutions) - 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 33 553 407
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 * 32,768 revolutions) - 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 8 388 351

See further examples in the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.

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 46.

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 "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.

Default = 134 213 631 (min. = 0, max. = 134 213 631)

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 * 32,768 revolutions) – 1 step – 1,024 steps (i.e. 1 revolution for safety reasons) = 33 553 407

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 * 32,768 revolutions) – 1 step – 256 steps (i.e. 1 revolution for safety reasons) = 8 388 351
See further examples in the "6.3 Distance per revolution, Preset, Positive delta and Negative delta" section on page 46.

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 46.

Jog speed [0x0D]

[Register 14, Unsigned16, rw]
This parameter contains the maximum speed the device is allowed to reach when using the **Jog +** and **Jog -** functions (see the **Control Word [0x2A]** parameter). The parameter is expressed in revolutions per minute (rpm). See also the "Jog: speed control" section on page 44.
Default = 2000 (min. = 1, max. = 3000)

Work speed [0x0E]

[Register 15, Unsigned16, rw]
This parameter contains the maximum speed the device is allowed to reach in automatic work mode (movements are controlled using the **Start** and **Stop** commands -see the **Control Word [0x2A]** parameter- and are performed in order to reach the position set in **Target position [0x2B-0x2C]**). The parameter is expressed in revolutions per minute (rpm). See also the "Positioning: position and speed control" section on page 45.
Default = 2000 (min. = 1, max. = 3000)

Code sequence [0x0F]

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

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

Offset [0x10-0x11]

[Registers 17-18, Integer32, ro]
This variable defines the difference between the position value transmitted by the device and the real position: real position – preset. Following a Preset operation, the Offset value is automatically stored in the memory. The value is expressed in pulses.

Preset [0x12-0x13]

[Registers 19-20, Integer32, rw]
Use this parameter to set the Preset value. The Preset function is meant to assign a desired value to a physical position of the axis. The chosen physical position will get the value set next to this item and all the previous and the following positions will get a value according to it. The preset value will be set for the position of the axis in the moment when the value is entered. The preset value is activated when the bit 11 **Setting the preset** in the **Control Word [0x2A]** register is switched from logic level low ("0") to logic level high ("1"). Following a Preset operation, the Offset value is automatically stored in the memory. The value is expressed in pulses.
Default = 0 (min. = -268 435 456, max. = +268 435 456)

NOTE
We suggest activating the preset when the actuator is in stop. See the **Setting the preset** command on page 140.

WARNING
A new value must be set in the **Preset [0x12-0x13]** registers every time the **Distance per revolution [0x00]** value is changed. After having entered a new value in **Preset [0x12-0x13]** it is not necessary to set new values for travel limits as the Preset function calculates them automatically and initializes again the positive and negative limits according to the values set in the **Positive delta [0x09-0x0A]** and **Negative delta [0x0B-0x0C]** items. For a detailed explanation see on page 46.

Jog step length [0x14]

[Register 21, Unsigned16, rw]
If the incremental jog function is enabled (bit 4 **Incremental jog** in the **Control Word [0x2A]** = 1), the activation of the bits **Jog +** and **Jog -** causes a single step toward the positive or negative direction having the length, expressed in pulses, set next to this item to be executed at rising edge; then the actuator stops and waits for another command.
Default = 1000 (min. = 1, max. = 10000).

Extra commands register [0x29]

[Register 42, Unsigned16, rw]
Byte structure of the **Extra commands register [0x29]**:

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

Byte 0
bit 0: Not used.

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

bits 2 ... 7 Not used.

Byte 1 Not used.

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

Control Word [0x2A]

[Register 43, Unsigned16, rw]

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

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

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

Byte 0

Jog +

bit 0

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

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

Jog -

bit 1

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

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

Stop

bit 2

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

Alarm reset

bit 3

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

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

Incremental jog

bit 4

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

bit 5

Not used.

Start

bit 6

When the bit value switches from "0" to "1", the device moves in order to reach the set target position (see **Target position [0x2B-0x2C]** on page 140). For a complete description of the position control see on page 45. This bit has to be switched back to "0" after the device has started.

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

Emergency

bit 7

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

Byte 1

Watch dog enable

bit 8

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

Save parameters

bit 9

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

Load default parameters

bit 10

The default parameters (they are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode) are restored at each rising edge of the bit; in other words, the default parameters loading operation is performed each time this

bit is switched from logic level low ("0") to logic level high ("1"). The complete list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 158.

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 135.

Release axis torque

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

bits 13 ... 15 — Not used.

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

Target position [0x2B-0x2C]

[Registers 44-45, Integer32, rw]
It sets the position to be reached, otherwise referred to as commanded position. The value is expressed in pulses. When the **Start** command is sent while the **Stop** and **Emergency** bits are "=1" and the alarm condition is off, the device moves in order to reach the target position set next to this item.
As soon as the axis is within the tolerance window limits set next to the **Position window [0x01]** register, the bit 8 **Target position reached** in the **Status word [0x01]** goes high ("=1"). When the position is within the tolerance window limits set next to the **Position window [0x01]** register, after the delay set next to the **Position window time [0x02]** item, the bit 0 **Axis in position** in the **Status word [0x01]** goes high ("=1").
For more information refer also to the "Positioning: position and speed control" section on page 45.
Default = 0 (min. = 0, max. = within maximum positive limit / maximum negative limit)

NOTE

Position override function

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

NOTE

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

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

WARNING

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

NOTE

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

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

8.12.2 Input Register parameters

The **Input Register** parameters are accessible for reading only; to read the value set in an input register parameter use the **04 Read Input Register** function code (reading of multiple input registers); for any further information on the implemented function codes refer to the "8.11.1 Implemented function codes" section on page 119.

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 encoder position and the incremental encoder position are read and saved to two separate registers. If any difference between the values in the registers is found the error is signalled.

Following error

bit 3 — The difference between the real position and the theoretical position is greater than the value set in the **Max following error [0x03-0x04]** parameter; we suggest reducing the 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 146). 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 146). Please wait some minutes for the actuator to cool down. Ensure that the operating temperature is within the range.

Undervoltage

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

Watch dog

bit 11 — When the Watch dog function is enabled (bit 8 **Watch dog enable** in **Control Word [0x2A]** is set to "=1"), if the device does not receive a message from the Server within 1 second, the system forces an alarm condition (the **Watch dog** alarm bit is activated). The alarm is invoked to appear as soon as the Modbus network communication is restored. The Watch dog function is a safety timer that uses a time-out to detect loop or deadlock conditions. For instance, should the serial communication be cut off while a command is still active and running -a jog command for example- the Watch dog safety system immediately takes action and commands a safety stop of the device; furthermore an alarm is triggered.

bits 12 and 13 — Not used.

Hall sequence

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

Overvoltage

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

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

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

Status word [0x01]

[Register 2, Unsigned16, ro]
This register contains information about the current state of the device.
Byte structure of the Status word [0x01] register:

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

Byte 0

Axis in position

bit 0 — The value is "=1" when the device reaches and keeps the set position (Target position [0x2B-0x2C]) for the time set next to the Position window time [0x02] register. It is kept active until the position error is lower than Position window [0x01]. For

further information please refer to the "Positioning: position and speed control" section on page 45.

bit 1 — Not used.

Drive enabled

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

SW limit switch +

bit 3 — The value is "=1" should it happen that the device reaches the maximum positive limit (positive limit switch). For more information see the **Positive delta [0x09-0x0A]** parameter.

SW limit switch -

bit 4 — The value is "=1" should it happen that the device reaches the maximum negative limit (negative limit switch). For more information see the **Negative delta [0x0B-0x0C]** parameter.

Alarm

bit 5 — The value is "=1" when an alarm occurs, see details in the **Alarms register [0x00]** variable.

Axis running

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

Executing a command

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

Byte 1

Target position reached

bit 8 — The value is "=1" when the device reaches the target position set next to the **Target position [0x2B-0x2C]** item (it is within the limits set next to the **Position window [0x01]**). The bit is kept active until a new **Target position [0x2B-0x2C]** value or the **Alarm reset** command are sent. For more information

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

bits 9 ... 11 — Not used.

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.

bits 13 ... 15 — Not used.

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.

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 and 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 RD6 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 RD6 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								Hardware version							

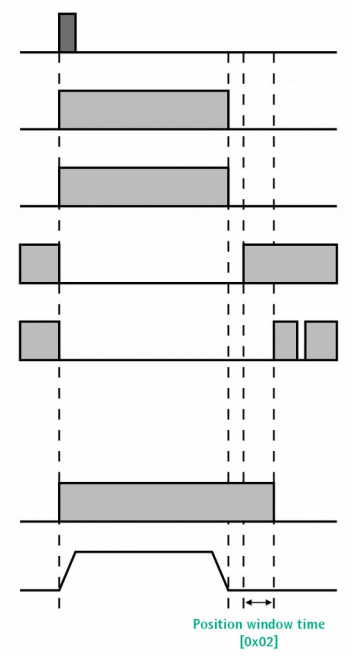
where:

00 ... 07	= hardware version
08 .. 15	= RD6 model equipped with the following interface: 0x10 = Modbus; 0x11 = Profibus; 0x12 = CANopen; 0x13 = POWERLINK; 0x14 = EtherCAT

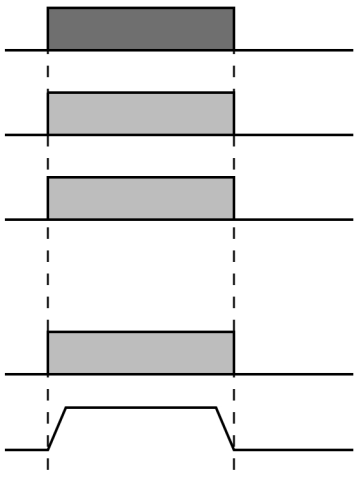
Value 10 01 hex in hexadecimal notation corresponds to the binary representation 0001 0000 0000 0001 and has to be interpreted as follows: hardware version 1 (LSbyte = 0x01); RD6 model with Modbus interface (MSbyte = 0x10).

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	RD6
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 +	RD6
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	

8.13 Exception codes

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

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

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

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

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

8.14 Programming examples

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

8.14.1 Using the 03 Read Holding Registers function code

EXAMPLE 1
Request to read the parameters **Acceleration [0x07]** (register 8) and **Deceleration [0x08]** (register 9) to the Slave having the node address 1.

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

Response PDU
[01][03][04][03][E8][05][DC][78][8A]
where:
[01] = Slave address
[03] = **03 Read Holding Registers** function code
[04] = number of bytes (2 bytes for each register)
[03][E8] = value of register 8 **Acceleration [0x07]**, 03 E8 hex = 1000 dec
[05][DC] = value of register 9 **Deceleration [0x08]**, 05 DC hex = 1500 dec
[78][8A] = CRC

Acceleration [0x07] parameter (register 8) contains the value 03 E8 hex, i.e. 1000 in decimal notation; **Deceleration [0x08]** parameter (register 9) contains the value 05 DC hex, i.e. 1500 in decimal notation.

8.14.2 Using the 04 Read Input Register function code

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

Request PDU
[01][04][00][02][00][02][D0][0B]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[00][02] = starting address (**Current position [0x02-0x03]** parameter, register 3)
[00][02] = number of requested registers
[D0][0B] = CRC

Response PDU
[01][04][04][00][00][2F][F0][E7][F0]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[04] = number of bytes (2 bytes for each register)
[00][00] = value of register 3 **Current position [0x02-0x03]**, 00 00 hex = 0 dec
[2F][F0] = value of register 4 **Current position [0x02-0x03]**, 2F F0 hex = 12272 dec
[E7][F0] = CRC

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

EXAMPLE 2
Request to read the **Alarms register [0x00]** variable (register 1) to the Slave having the node address 1.

Request PDU
[01][04][00][00][00][01][31][CA]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[00][00] = starting address (**Alarms register [0x00]** variable, register 1)
[00][01] = number of requested registers

[31][CA] = CRC

Response PDU
[01][04][02][00][81][79][50]
where:
[01] = Slave address
[04] = **04 Read Input Register** function code
[02] = number of bytes (2 bytes for each register)
[00][81] = value of register 1 **Alarms register [0x00]**, 00 81 hex = 0000 0000 1000 0001 bin
[79][50] = CRC

This means that in the **Alarms register [0x00]** variable (register 1) the bits 0 and 7 are active (logic level high = 1), i.e. (see on page 142): **Machine data not valid** and **Emergency**.

8.14.3 Using the 06 Write Single Register function code

EXAMPLE 1
Request to write the value 05 DC hex (= 1500 dec) in the **Acceleration [0x07]** parameter (register 8) of the Slave having the node address 1.

Request PDU
[01][06][00][07][05][DC][3A][C2]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[05][DC] = value to be set in the register
[3A][C2] = CRC

Response PDU
[01][06][00][07][05][DC][3A][C2]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][07] = address of the register (**Acceleration [0x07]** parameter, register 8)
[05][DC] = value set in the register
[3A][C2] = CRC

The value 05 DC hex, i.e. 1500 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8).

EXAMPLE 2
Request to write the value 00 84 hex in the **Control Word [0x2A]** variable (register 43) of the Slave having the node address 1.

Request PDU
[01][06][00][2A][00][84][A8][61]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[00][84] = value to be set in the register
[A8][61] = CRC

Response PDU
[01][06][00][2A][00][84][A8][61]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[00][84] = value set in the register
[A8][61] = CRC

The value 00 84 hex = 0000 0000 1000 0100 in binary notation is set in the **Control Word [0x2A]** variable (register 43). In other words, the **Stop** and **Emergency** bits are forced to the logical level high (bit 2 = 1; bit 7 = 1): the unit is ready to execute the motion command as requested.

EXAMPLE 3
Request to write the value 0A 80 hex in the **Control Word [0x2A]** variable (register 43) of the Slave having the node address 1.

Request PDU
[01][06][00][2A][0A][80][AF][02]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[0A][80] = value to be set in the register
[AF][02] = CRC

Response PDU
[01][06][00][2A][0A][80][AF][02]
where:
[01] = Slave address
[06] = **06 Write Single Register** function code
[00][2A] = address of the register (**Control Word [0x2A]** variable, register 43)
[0A][80] = value set in the register
[AF][02] = CRC

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

8.14.4 Using the 16 Write Multiple Registers function code

EXAMPLE 1

Request to write the values 1500 and 1000 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][05][DC][03][E8][73][C1]
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)
[05][DC] = value to be set in the register 8 **Acceleration [0x07]**, 05 DC hex = 1000 dec
[03][E8] = value to be set in the register 9 **Deceleration [0x08]**, 03 E8 hex = 1500 dec
[73][C1] = 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 05 DC hex, i.e. 1500 in decimal notation, is set in the **Acceleration [0x07]** parameter (register 8); the value 03 E8 hex, i.e. 1000 in decimal notation, is set in the **Deceleration [0x08]** parameter (register 9).

9 Default parameters list

Parameters list	Default value		
Control Word (Bytes 0 and 1)	0		
Target position (Bytes 4 ... 7)	0		
00 Distance per revolution	4096		
01 Position window	1		
02 Position window time	0		
03 Max following error	50000		
04 Kp position loop	80		
05 Ki position loop	10		
06 Acceleration	10		
07 Deceleration	10		
08 Positive delta	134213631		
09 Negative delta	134213631		
0A Jog speed	2000		
0B Work speed	2000		
0C Code sequence	0		
0D Jog step length	1000		
0E Preset	0		

Parameters list	Default value		
Distance per revolution [0x00] PPR	4096		
Position window [0x01] P	1		
Position window time [0x02] ms	0		
Max following error [0x03-0x04] P	50000		
Kp position loop [0x05]	80		
Ki position loop [0x06]	10		
Acceleration [0x07] rev/s^2	10		
Deceleration [0x08] rev/s^2	10		
Positive delta [0x09-0x0A] P	134213631		
Negative delta [0x0B-0x0C] P	134213631		
Jog speed [0x0D] rpm	2000		
Work speed [0x0E] rpm	2000		
Code sequence [0x0F]	0		
Preset [0x12-0x13] P	0		
Jog step length [0x14] P	1000		
Control Word [0x2A]	0		
Target position [0x2B-0x2C]	0		

This page intentionally left blank

Document release	Release date	Description	HW	SW	Interface
1.0	07.09.2016	First issue	1.0	1.0	1.2

Lika Electronic
Via S. Lorenzo, 25 • 36010 Carrè (VI) • Italy

Tel. +39 0445 806600
Fax +39 0445 806699

info@lika.biz • www.lika.biz