

## EM58 EC HS58 EC HM58 EC



**EtherCAT** 

in compliance with ETG.1000

- EM58 27-bit multiturn encoder for standard purposes
- HS58 18-bit singleturn encoder for high precision demands
- HM58 30-bit multiturn encoder for high end applications
- Complies with ETG.1000 specifications
- Implements CoE & FoE protocols and the EtherCAT State Machine

#### Suitable for the following models:

- EM58, EM58S EC
- EMC58, EMC59, EMC60 EC
- HS58, HS58S EC
- HSC58, HSC59, HSC60 EC
- HM58, HM58S EC
- HMC58, HMC59, HMC60 EC

#### General Contents

1 - Safety summary	16
2 - Identification	18
3 - Mechanical installation	19
4 - Electrical connections	24
5 - Getting started	29
6 - Quick reference with TwinCAT	30
7 - EtherCAT® interface	51
8 - Default parameters list	85

This publication was produced by Lika Electronic s.r.l. 2019. All rights reserved. Tutti i diritti riservati. Alle Rechte vorbehalten. Todos los derechos reservados. Tous droits réservés.

This document and information contained herein are the property of Lika Electronic s.r.l. and shall not be reproduced in whole or in part without prior written approval of Lika Electronic s.r.l. Translation, reproduction and total or partial modification (photostat copies, film and microfilm included and any other means) are forbidden without written authorisation of Lika Electronic s.r.l.

The information herein is subject to change without notice and should not be construed as a commitment by Lika Electronic s.r.l. Lika Electronic s.r.l. reserves the right to make all modifications at any moments and without forewarning.

This manual is periodically reviewed and revised. As required we suggest checking if a new or updated edition of this document is available at Lika Electronic s.r.l.'s website. Lika Electronic s.r.l. assumes no responsibility for any errors or omissions in this document. Critical evaluation of this manual by the user is welcomed. Your comments assist us in preparation of future documentation, in order to make it as clear and complete as possible. Please send an e-mail to the following address [info@lika.it](mailto:info@lika.it) for submitting your comments, suggestions and criticisms.

The logo for Lika Electronic s.r.l. consists of the word "lika" in a bold, lowercase, sans-serif font. The letter "i" has a dot above it. The logo is positioned in the bottom right corner of the page.

# General contents

User's guide.....	1
General contents.....	3
Subject Index.....	6
Typographic and iconographic conventions.....	7
Preliminary information.....	8
Glossary of EtherCAT terms.....	9
<b>1 - Safety summary.....</b>	<b>16</b>
1.1 Safety.....	16
1.2 Electrical safety.....	16
1.3 Mechanical safety.....	16
<b>2 - Identification.....</b>	<b>18</b>
<b>3 - Mechanical installation.....</b>	<b>19</b>
3.1 Solid shaft encoders.....	19
3.1.1. Customary installation.....	19
3.1.2 Installation using fixing clamps (code LKM-386).....	20
3.1.3 Installation using a mounting bell (code PF4256).....	20
3.2 Hollow shaft encoders.....	21
3.2.1 EMC58, HSC58, HMC58.....	21
3.2.2 EMC59, HSC59, HMC59.....	22
3.2.3 EMC60, HSC60, HMC60.....	23
<b>4 - Electrical connections.....</b>	<b>24</b>
4.1 EtherCAT interface and power supply connectors.....	24
4.2 Network configuration: topologies, cables, hubs, switches - Recommendations.....	25
4.3 Addressing.....	26
4.4 Line Termination.....	26
4.5 Ground connection.....	26
4.6 Diagnostic LEDs.....	27
<b>5 - Getting started.....</b>	<b>29</b>
<b>6 - Quick reference with TwinCAT.....</b>	<b>30</b>
6.1 System configuration using TwinCAT software system from Beckhoff.....	30
6.1.1 Setting the Network Card.....	30
6.1.2 Add new I/O modules (Boxes).....	33
6.2 Setting the communication mode.....	37
6.2.1 Synchronous with SM3.....	37
6.2.2 Synchronous with DC (SYNCO).....	38
6.3 Process Data Objects.....	39
6.4 CoE Object Dictionary.....	40
6.5 Online Data.....	41
6.6 EEPROM upgrade.....	42
6.7 Firmware upgrade.....	47
<b>7 - EtherCAT® interface.....</b>	<b>51</b>
7.1 Basic Information on EtherCAT® Protocol.....	51
7.1.1 Data transfer.....	52
7.1.2 ISO/OSI Layer model.....	53
7.1.3 Topology.....	53
7.1.4 Line Termination.....	54

7.1.5 Addressing.....	55
7.1.6 Communication mode.....	56
FreeRun.....	56
Synchronous with SM3.....	57
Synchronous with DC SYNC0.....	57
7.1.7 EtherCAT State Machine (ESM).....	58
7.1.8 Slave configuration.....	59
7.1.9 Timing and Synchronization.....	59
Sync Manager.....	60
Buffered Mode (3-Buffer Mode).....	60
Mailbox Mode (1-Buffer Mode).....	60
7.2 CANopen Over EtherCAT (CoE).....	62
7.2.1 XML file.....	62
7.2.2 Communication messages.....	63
7.2.3 Process Data Objects (PDO).....	64
7.2.4 Service Data Objects (SDO).....	64
7.2.5 Object dictionary.....	65
Communication Profile Area objects (DS301).....	67
<b>1000-00 Device type</b> .....	67
<b>1008-00 Device Name</b> .....	67
<b>1009-00 Hardware version</b> .....	67
<b>100A-00 Software version</b> .....	67
<b>1010-01 Store parameters</b> .....	68
<b>1011-01 Restore default parameters</b> .....	68
<b>1018 Identity</b> .....	69
01 Vendor ID.....	69
02 Product code.....	69
03 Revision.....	69
04 Serial number.....	69
<b>1A00-01 PDO mapping parameter</b> .....	70
01 Mapped Object 001.....	70
<b>1C00 Sync Manager Communication Type</b> .....	70
01 SM MailBox Receive (SM0).....	70
02 SM MailBox Send (SM1).....	70
03 SM PDO output (SM2).....	70
04 SM PDO input (SM3).....	70
<b>1C12-00 Sync Manager RxPDO Assigned</b> .....	71
<b>1C13-01 Sync Manager TxPDO Assigned</b> .....	71
01 Subindex 001.....	71
<b>1C33 Sync Manager input parameter</b> .....	71
01 Sync Type.....	71
02 Cycle time.....	71
03 Shift Time.....	71
04 Sync modes supported.....	72
05 Minimum cycle time.....	72
06 Calc and Copy time.....	72
Standardised Profile Area objects (DS406).....	73
<b>6000-00 Operating parameters</b> .....	73
Code sequence.....	73
Scaling function.....	73
<b>6001-00 Units per revolution</b> .....	74

6002-00 Total Measuring Range.....	75
6003-00 Preset.....	77
6004-00 Position value.....	79
6500-00 Operating status.....	79
Code sequence.....	79
Scaling function.....	79
6501-00 Hardware counts per revolution.....	80
6502-00 Hardware number of turns.....	80
6503-00 Errors.....	80
6504-00 Supported errors.....	80
6505-00 Warnings.....	81
6506-00 Supported warnings.....	81
6509-00 Offset.....	81
7.2.6 SDO Abort codes.....	82
7.2.7 Emergency Error Codes.....	83
7.2.8 AL Status Error Codes.....	83
7.3 File Over EtherCAT (FoE).....	84
<b>8 – Default parameters list.....</b>	<b>85</b>

# Subject Index




<b>1</b>		
1000-00 Device type.....	67	
1008-00 Device Name.....	67	
1009-00 Hardware version.....	67	
100A-00 Software version.....	67	
1010-01 Store parameters.....	68	
1011-01 Restore default parameters.....	68	
1018 Identity.....	69	
1A00-01 PDO mapping parameter.....	70	
1C00 Sync Manager Communication Type.....	70	
1C12-00 Sync Manager RxPDO Assigned.....	71	
1C13-01 Sync Manager TxPDO Assigned.....	71	
1C33 Sync Manager input parameter.....	71	
<b>6</b>		
6000-00 Operating parameters.....	73	
6001-00 Units per revolution.....	74	
6002-00 Total Measuring Range.....	75	
6003-00 Preset.....	77	
6004-00 Position value.....	79	
6500-00 Operating status.....	79	
6501-00 Hardware counts per revolution.....	80	
6502-00 Hardware number of turns.....	80	
6503-00 Errors.....	80	
6504-00 Supported errors.....	80	
6505-00 Warnings.....	81	
6506-00 Supported warnings.....	81	
6509-00 Offset.....	81	
<b>B</b>		
BOOTSTRAP.....	58	
<b>C</b>		
Calc and Copy time.....	72	
Code sequence.....	73, 79	
		Cycle time.....71
<b>D</b>		
		Diagnostic data.....83
<b>H</b>		
		Hardware error.....83
<b>I</b>		
		INIT.....58
<b>M</b>		
		Mapped Object 001.....70
		Minimum cycle time.....72
<b>O</b>		
		OPERATIONAL.....58
<b>P</b>		
		PRE-OPERATIONAL.....58
		Product code.....69
<b>R</b>		
		Revision.....69
<b>S</b>		
		SAFE-OPERATIONAL.....58
		Scaling function.....73, 79
		Serial number.....69
		Shift Time.....71
		SM MailBox Receive (SM0).....70
		SM MailBox Send (SM1).....70
		SM PDO input (SM3).....70
		SM PDO output (SM2).....70
		Subindex 001.....71
		Sync modes supported.....72
		Sync Type.....71
<b>V</b>		
		Vendor ID.....69

# 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 the device and the interface are coloured in **GREEN**;
- alarms are coloured in **RED**;
- states are coloured in **FUCSIA**.

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

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

# Preliminary information

This guide is designed to describe the technical characteristics, installation and use of the **EtherCAT encoders of the EMx58x series and Hx58x series**. For complete technical information please refer to the product datasheet.

**EtherCAT** is the open standard for Industrial Ethernet. Its technology enables improved performances and allows to meet efficiency and productivity requirements in any complex industrial system. Real time communication, deterministic synchronization, high speed up to 100 Mbit/s full duplex over long distances, flexible network topologies, complete diagnostics, IT integration are among the key benefits.

EtherCAT encoders comply with ETG.1000 specifications and implement the CoE (CANopen over EtherCAT) and FoE (for firmware update) protocols as well as the EtherCAT State Machine. Thus they offer full scaling, preset, code sequence, position and velocity readout, diagnostic information etc.

EM58 is the low cost multiturn version offering 27-bit resolution (8192 cpr x 16384 rev.). H- high resolution series comes in both singleturn (18-bit resolution, 262144 cpr) and multiturn (30-bit resolution, 65536 cpr x 16384 rev.) versions. Mechanically they afford standard 58-mm flange diameter housing with both hollow ( $\varnothing$  14, 15 mm) and solid ( $\varnothing$  6, 8, 9.52, 10, 12 mm) shafts. Capable of 6.000 RPM and working temperatures between  $-25^{\circ}\text{C}$  and  $+85^{\circ}\text{C}$  ( $-13^{\circ}\text{F}$   $+185^{\circ}\text{F}$ ) they provide IP65-rated protection.

To make it easier to read the text, this guide is 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 **EtherCAT Interface**, both general and specific information is given on the EtherCAT interface. In this section the interface features and the objects implemented in the unit are fully described.



# Glossary of EtherCAT terms

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

<b>Acknowledge telegram (AT)</b>	Telegram, in which each Slave inserts its data.
<b>Actual value</b>	Value of a variable at a given instant.
<b>Algorithm</b>	Completely determined finite sequence of operations by which the values of the output data can be calculated from the values of the input data.
<b>Application</b>	Function or data structure for which data is consumed or produced. Software functional element specific to the solution of a problem in industrial-process measurement and control.
<b>Application class</b>	Configuration of a Drive Object with a set of functional objects and supported by standard telegrams.
<b>Application mode</b>	Type of application that can be requested from a PDS.
<b>Application objects</b>	Multiple object classes that manage and provide a run time exchange of messages across the network and within the network device.
<b>Application process</b>	Part of a distributed application on a network, which is located on one device and unambiguously addressed.
<b>Application relationship</b>	Cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation. This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities.
<b>Attribute</b>	Description of an externally visible characteristic or feature of an object, property or characteristic of an entity. The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behaviour of an object. Attributes are divided into class attributes and instance attributes.
<b>Axis</b>	Logical element inside an automation system (e.g. a motion control system) that represents some form of movement.
<b>Basic Slave</b>	Slave device that supports only physical addressing of data.
<b>Behaviour</b>	Indication of how an object responds to particular events.
<b>Bit</b>	Unit of information consisting of a 1 or a 0. This is the smallest data unit that can be transmitted.

<b>CANopen</b>	Application layer protocol as defined in EN 50325-4.
<b>Channel</b>	Representation of a single physical or logical management object of a Slave to control conveyance of data.
<b>CIP™</b>	Common Industrial Protocol (see IEC 61158 Type 2, IEC 61784-1 and IEC 61784-2 CPF2).
<b>Class</b>	Description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
<b>Client</b>	Object which uses the services of another (Server) object to perform a task. Initiator of a message to which a Server reacts.
<b>Clock synchronization</b>	Representation of a sequence of interactions to synchronize the clocks of all time receivers by a time Master.
<b>Commands</b>	Set of commands from the application control program to the PDS to control the behaviour of the PDS or functional elements of the PDS.
<b>Communication cycle</b>	Accumulation of all telegrams between two Master synchronization telegrams.
<b>Communication object</b>	Component that manages and provides a run time exchange of messages across the network.
<b>Connection</b>	Logical binding between two application objects within the same or different devices.
<b>Consume</b>	Act of receiving data from a provider.
<b>Consumer</b>	Node or sink receiving data from a provider.
<b>Control</b>	Purposeful action on or in a process to meet specified objectives.
<b>Control device</b>	Physical unit that contains – in a module/subassembly or device – an application program to control the PDS.
<b>Control unit</b>	Control device.
<b>Control word</b>	Two adjacent bytes inside the Master data telegram containing commands for the addressed drive.
<b>Controller</b>	Controlling device which is associated with one or more drives (axes) a host for the overall automation.
<b>Conveyance path</b>	Unidirectional flow of APDUs across an application relationship.
<b>Cycle time</b>	Time span between two consecutive cyclically recurring events.
<b>Cyclic</b>	Events which repeat in a regular and repetitive manner.
<b>Cyclic data</b>	Part of the telegram which does not change its meaning during cyclic operation of the interface. High priority real-time data that is transferred by a CIP Motion connection on a periodic basis.
<b>Data</b>	Generic term used to refer to any information carried over a

	fieldbus.
<b>Data consistency</b>	Means for coherent transmission and access of the input-or-output-data object between and within Client and Server.
<b>Data exchange</b>	Demand dependent; non cyclic transmission (service channel).
<b>Data type</b>	Relation between values and encoding for data of that type according to the definitions of IEC 61131-3. Set of values together with a set of permitted operations.
<b>Data type object</b>	Entry in the object dictionary indicating a data type.
<b>Default gateway</b>	Device with at least two interfaces in two different IP subnets acting as router for a subnet.
<b>Device</b>	Field device. Networked independent physical entity of an industrial automation system capable of performing specified functions in a particular context and delimited by its interfaces. Entity that performs control, actuating and/or sensing functions and interfaces to other such entities within an automation system. Physical entity connected to the fieldbus composed of at least one communication element (the network element) and which may have a control element and/or a final element (transducer, actuator, etc.).
<b>Device profile</b>	Collection of device dependent information and functionality providing consistency between similar devices of the same device type. Representation of a device in terms of its parameters and behaviour according to a device model that describes the device's data and behaviour as viewed through a network, independent from any network technology.
<b>Diagnosis information</b>	All data available at the Server for maintenance purposes.
<b>Distributed clocks</b>	Method to synchronize Slaves and maintain a global time base.
<b>DL</b>	Data-link-layer.
<b>DLPDU</b>	Data-link-protocol-data-unit.
<b>Drive Object</b>	Functional element of a Drive Unit.
<b>Drive Unit</b>	Logical device which comprises all functional elements related to one central processing unit.
<b>Error</b>	Discrepancy between a computed, observed or measured value or condition and the specified or theoretically correct value or condition.
<b>Error class</b>	General grouping for related error definitions and corresponding error codes.
<b>Error code</b>	Identification of a specific type of error within an error class.
<b>EtherCAT State Machine</b>	EtherCAT Slave is a state machine; communication and operating characteristics depend on the current state of the

	device.
<b>Event</b>	Instance of a change of conditions.
<b>Event data</b>	Medium priority real-time data that is transferred by a CIP Motion connection only after a specified event occurs.
<b>Feed forward</b>	Command value used to compensate the lag in the control loop.
<b>Feedback variable</b>	Variable which represents a controlled variable and which is returned to a comparing element.
<b>Fieldbus memory management unit</b>	Function that establishes one or several correspondences between logical addresses and physical memory.
<b>Fieldbus memory management unit entity</b>	Single element of the fieldbus memory management unit: one correspondence between a coherent logical address space and a coherent physical memory location.
<b>Frame</b>	Denigrated synonym for DLPDU.
<b>FreeRun</b>	Asynchronous communication mode.
<b>Full Slave</b>	Slave device that supports both physical and logical addressing of data.
<b>Functional element</b>	Entity of software or software combined with hardware, capable of accomplishing a specified function of a device.
<b>HMI</b>	Human Machine Interface.
<b>Host</b>	Device that covers the automation functionality of an automation device.
<b>I/O data</b>	Input data and output data that would typically need to be updated on a regular basis (e.g. periodic change of state), such as commands, set-points, status and actual values.
<b>Identification number (IDN)</b>	Designation of operating data under which a data block is preserved with its attribute, name, unit, minimum and maximum input values, and the data.
<b>Index</b>	Address of an object within an application process.
<b>Input data</b>	Data transferred from an external source into a device, resource or functional element.
<b>Interface</b>	Shared boundary between two entities defined by functional characteristics, signal characteristics, or other characteristics as appropriate.
<b>Little endian</b>	Data representation of multi-octet fields where the least significant octet is transmitted first.
<b>Logical power drive system</b>	Model which includes PDS and communication network accessible through the generic PDS interface.
<b>Mapping</b>	Correspondence between two objects in that way that one object is part of the other object.
<b>Mapping parameters</b>	Set of values defining the correspondence between application objects and process data objects.

<b>Master</b>	Device that controls the data transfer on the network and initiates the media access of the Slaves by sending messages and that constitutes the interface to the control system. Node, which assigns the other nodes the right to transmit.
<b>Master data telegram (MDT)</b>	Telegram, in which the Master inserts its data.
<b>Medium</b>	Cable, optical fibre or other means by which communication signals are transmitted between two or more points.
<b>Message</b>	Ordered series of octets intended to convey information. Normally used to convey information between peers at the application layer.
<b>Model</b>	Mathematical or physical representation of a system or a process, based with sufficient precision upon known laws, identification or specified suppositions.
<b>Motion</b>	Any aspect of the dynamics of an axis.
<b>Motion Axis Object</b>	Object that defines the attributes, services, and behaviour of a motion device based axis (or PDS) according to the CIP Motion specification, including Communications, Device Control, and Basic Drive FE elements as defined in IEC 61800-7.
<b>Network</b>	Set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways.
<b>Node</b>	Single DL-entity as it appears on one local link. End-point of a link in a network or a point at which two or more links meet [derived from IEC 61158-2].
<b>Object</b>	Abstract representation of a particular component within a device. An object can be: <ol style="list-style-type: none"> <li>1. an abstract representation of the capabilities of a device. Objects can be composed of any or all of the following components: <ul style="list-style-type: none"> <li>○ data (information which changes with time);</li> <li>○ configuration (parameters for behaviour);</li> <li>○ methods (things that can be done using data and configuration);</li> </ul> </li> <li>2. a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behaviour.</li> </ol>
<b>Object dictionary</b>	Data structure addressed by Index and Sub-index that contains description of data type objects, communication objects and application objects. List of objects with unique 16-bit index and 8-bit sub-index as defined in EN 50325-4.
<b>Operating cycle</b>	Period of the control loop within the drive or the control unit.
<b>Operating mode</b>	Characterization of the way and the extent to which the human operator intervenes in the control equipment.

<b>Output data</b>	Data originating in a device, resource or functional element and transferred from them to external systems.
<b>P-Device</b>	Field device and the host for the Drive Objects.
<b>Parameter</b>	Data element that represents device information that can be read from or written to a device, for example through the network or a local HMI.
<b>PDO</b>	Process Data Object.
<b>PDS</b>	Power Drive System.
<b>Process data</b>	Collection of application objects designated to be transferred cyclically or acyclically for the purpose of measurement and control.
<b>Process Data Object (PDO)</b>	Communication object with real-time capability. Structure described by mapping parameters containing one or several process data entities.
<b>Producer</b>	Node or source sending data to one or many consumers.
<b>Profile</b>	Representation of a PDS interface in terms of its parameters, parameter assemblies and behaviour according to a communication profile and a device profile.
<b>Protocol</b>	Convention about the data formats, time sequences, and error correction in the data exchange of communication systems.
<b>Reference variable</b>	Input variable to a comparing element in a controlling system which sets the desired value of the controlled variable and is deducted from the command variable.
<b>Resource</b>	Processing or information capability.
<b>Segment</b>	Collection of one real Master with one or more Slaves.
<b>Server</b>	Object which provides services to another (Client) object.
<b>Service</b>	Operation or function than an object and/or object class performs upon request from another object and/or object class.
<b>Service data</b>	Lower priority real-time data associated with a service message from the controller that is transferred by a CIP Motion connection on a periodic basis.
<b>Set-point</b>	Value or variable used as output data of the application control program to control the PDS.
<b>Slave</b>	DL-entity accessing the medium only after being initiated by the preceding Slave or the Master. Node, which is assigned the right to transmit by the Master.
<b>Standard telegram</b>	Set of input data and output data for an application mode.
<b>Status</b>	Set of information from the PDS to the application control program reflecting the state or mode of the PDS or a functional element of the PDS.
<b>Status word</b>	Two adjacent bytes inside the drive telegram containing status information.

<b>Subindex</b>	Sub-address of an object within the object dictionary.
<b>Supervisor</b>	Engineering device which manages provisions of configuration data (parameter sets) and collections of diagnosis data from P-Devices and/or controllers.
<b>Switch</b>	MAC bridge as defined in IEEE 802.1D.
<b>Sync Manager</b>	Sync Manager has the task of synchronizing data transfer between Master and Slave and prevents the same memory area from being written by different events. Collection of control elements to coordinate access to concurrently used objects.
<b>Sync manager channel</b>	Single control elements to coordinate access to concurrently used objects.
<b>Synchronised</b>	Condition where the local clock value on the drive is locked onto the Master clock of the distributed System Time.
<b>Synchronous with DC SYNC0</b>	In this operating mode data is sampled and then copied into Sync Manager buffer simultaneously at SYNC0 event generated by the ESC capture/compare unit.
<b>Synchronous with SM3</b>	In this mode data is sampled and then copied into Sync Manager buffer as soon as previous data was read from the Master (SM event); in this way new sampled data is synchronous with Master readings.
<b>System Time</b>	Absolute time value as defined in the CIP Sync specification in the context of a distributed time system where all devices have a local clock that is synchronised with a common Master clock.
<b>Telegram</b>	Message.
<b>Time stamp</b>	System Time stamp value associated with the CIP Motion connection data that conveys the absolute time when the associated data was captured, or that can also be used to determine when the associated data shall be applied.
<b>Topology</b>	Physical network architecture with respect to the connection between the stations of the communication system.
<b>Type</b>	Hardware or software element which specifies the common attributes shared by all instances of the type.
<b>Use case</b>	Class specification of a sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system.
<b>Variable</b>	Software entity that may take different values, one at a time.

# 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 "4 - Electrical connections" section on page 24;
- in compliance with the 2014/30/EU norm on electromagnetic compatibility, following precautions must be taken:
  - before handling and installing, 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. Provide the ground connection as close as possible to the unit. We suggest using the ground point provided in the connection cap (use one TCEI M3 x 6 cylindrical head screw with two tooth lock washers).



## 1.3 Mechanical safety

- Install the device following strictly the information in the "3 - Mechanical installation" section on page 19;
- mechanical installation has to be carried out with stationary mechanical parts;
- do not disassemble the encoder;
- do not tool the encoder 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 declared by manufacturer;
- unit with solid shaft: in order to guarantee maximum reliability over time of mechanical parts, we recommend a flexible coupling to be installed to connect the encoder and user's shaft; make sure the misalignment tolerances of the flexible coupling are respected;



- unit with hollow shaft: the encoder can be mounted directly on a shaft whose diameter has to respect the technical characteristics specified in the purchase order and clamped by means of the collar and, when requested, the anti-rotation pin.

## 2 - Identification

The device can be identified through the **order code** and the **serial number** printed on the label applied to its enclosure. 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.



**Warning:** encoders having order code ending with "/Sxxx" may have mechanical and electrical characteristics different from standard and be supplied with additional documentation for special connections (Technical info).

### 3 - Mechanical installation



**WARNING**

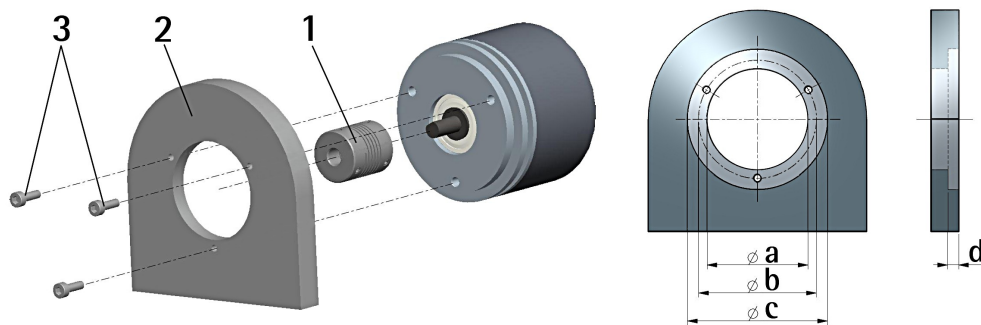
Installation and maintenance operations must be carried out by qualified personnel only, with power supply disconnected and mechanical parts absolutely in stop.

For any information on the mechanical data and the electrical characteristics of the encoder please refer to the [technical catalog](#).

#### 3.1 Solid shaft encoders

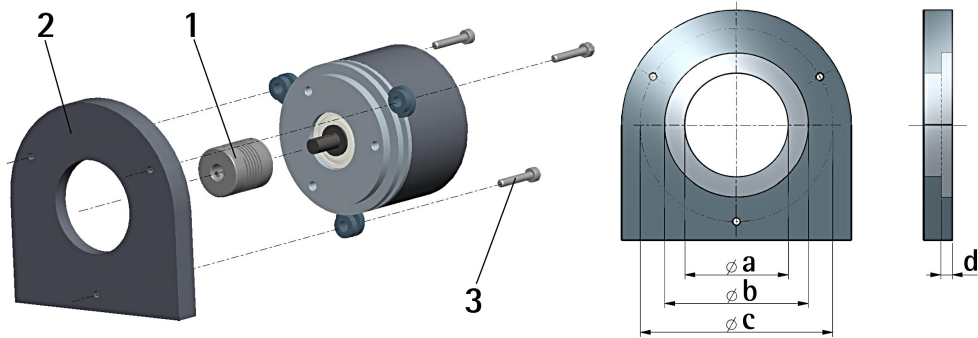
- Mount the flexible coupling **1** on the encoder shaft;
- fix the encoder to the flange **2** (or to the mounting bell) by means of screws **3**;
- secure the flange **2** to the support (or the mounting bell to the motor);
- mount the flexible coupling **1** on the motor shaft;
- make sure the misalignment tolerances of the flexible coupling **1** are respected.

##### 3.1.1. Customary installation



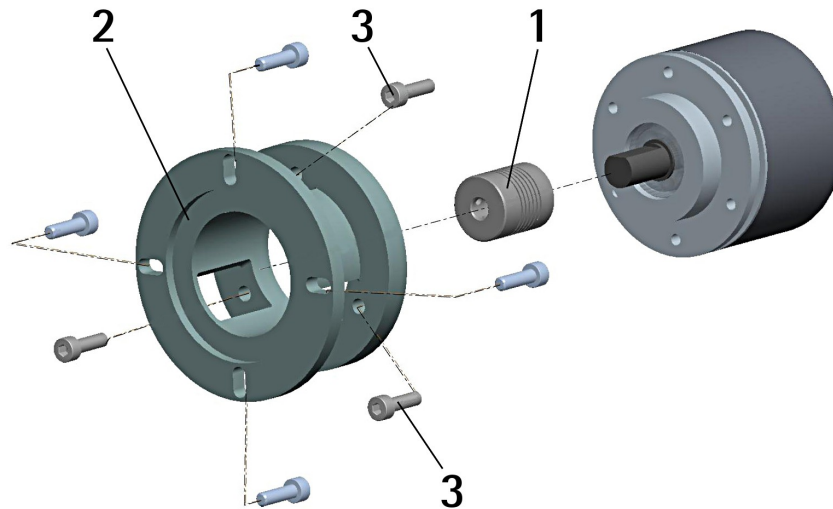
	a [mm]	b [mm]	c [mm]	d [mm]
EM58, HS58, HM58	-	42	50 F7	4
EM58S, HS58S, HM58S	36 H7	48	-	-

3.1.2 Installation using fixing clamps (code LKM-386)



	a [mm]	b [mm]	c [mm]	d [mm]
EM58, HS58, HM58	-	50 F7	67	4
EM58S, HS58S, HM58S	36 H7	-	67	-

3.1.3 Installation using a mounting bell (code PF4256)



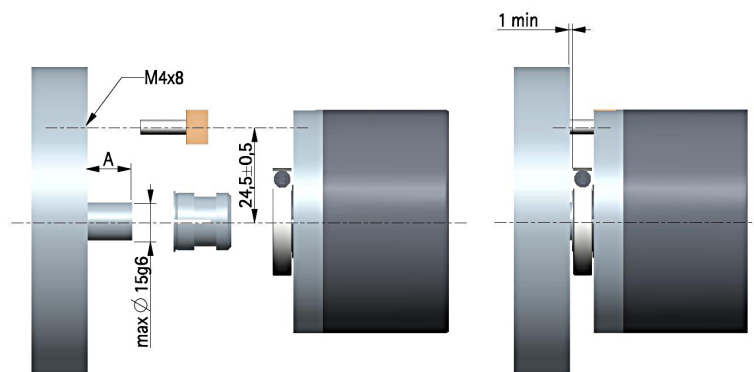
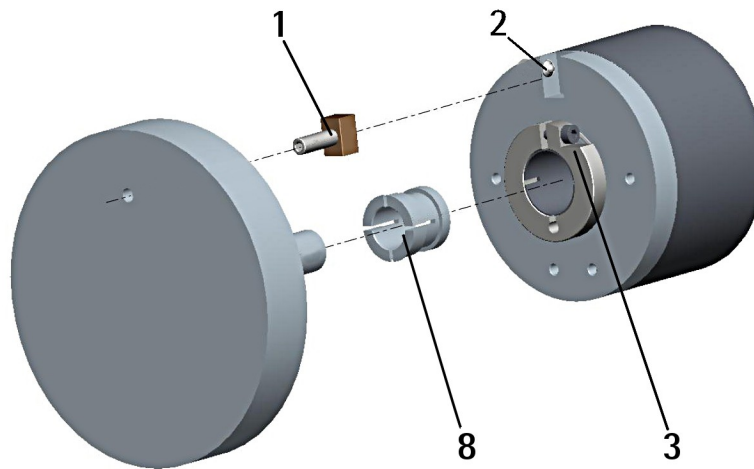
**NOTE**

In order to guarantee reliability over time of the encoder mechanical parts, we recommend a flexible coupling to be installed between the encoder and the motor shaft. Make sure the misalignment tolerances of the flexible coupling are respected.

### 3.2 Hollow shaft encoders

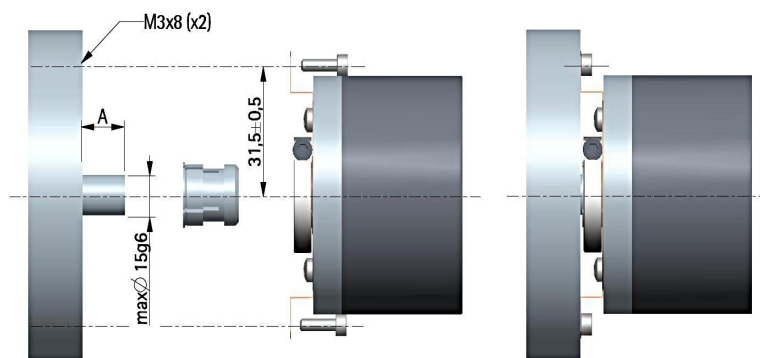
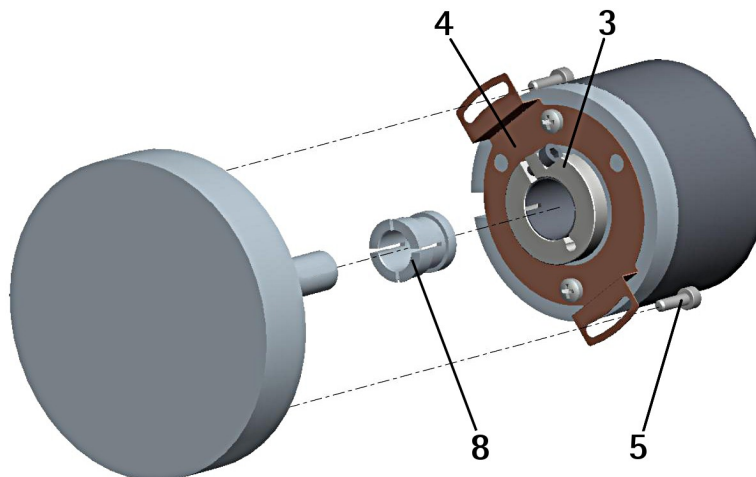
#### 3.2.1 EMC58, HSC58, HMC58

- Fasten the anti-rotation pin **1** to the rear of the motor (secure it using a locknut);
- mount the encoder on the motor shaft using the reducing sleeve **8** (if supplied). Avoid forcing the encoder shaft;
- insert the anti-rotation pin **1** into the slot on the flange of the encoder; this secures it in place by grub screw **2**, preset at Lika;
- fix the collar **3** to the encoder shaft (apply some threadlocker to the screw **3**).



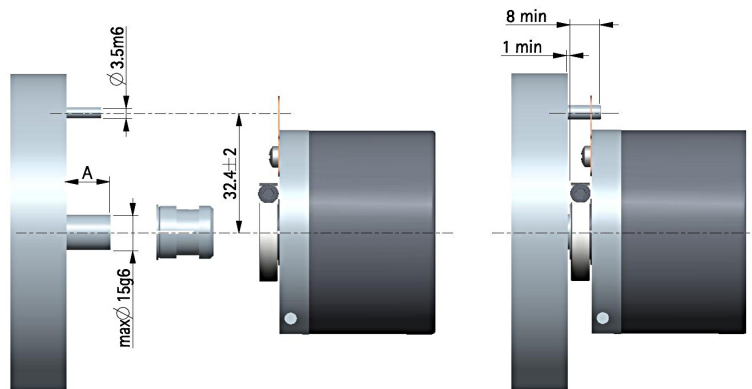
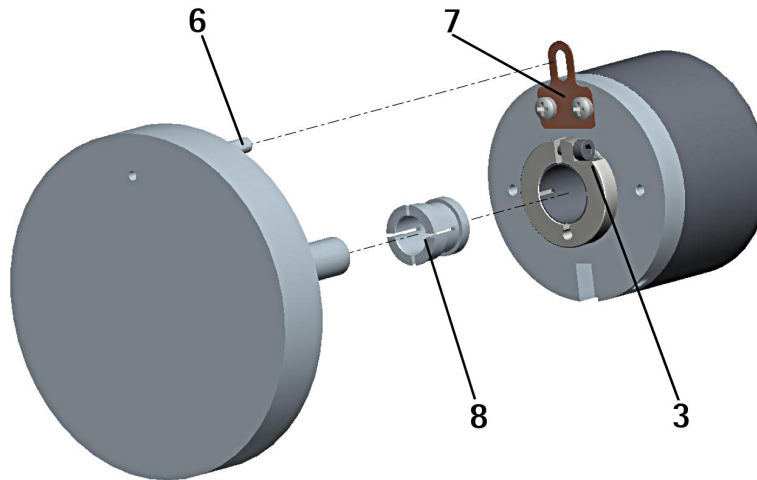
## 3.2.2 EMC59, HSC59, HMC59

- Mount the encoder on the motor shaft using the reducing sleeve **8** (if supplied). Avoid forcing the encoder shaft;
- fasten the fixing plate **4** to the rear of the motor using two M3 x 8 cylindrical head screws **5**;
- fix the collar **3** to the encoder shaft (apply some threadlocker to the screw **3**).



## 3.2.3 EMC60, HSC60, HMC60

- Fix the tempered pin **6** to the rear of the motor;
- mount the encoder on the motor shaft using the reducing sleeve **8** (if supplied). Avoid forcing the encoder shaft;
- make sure the anti-rotation pin **6** is inserted properly into the fixing plate **7**;
- fix the collar **3** to the encoder shaft (apply some threadlocker to the screw **3**).

**NOTE**

You are strongly advised not to carry out any mechanical operations (drilling, milling, etc.) on the encoder shaft. This could cause serious damages to the internal parts and an immediate warranty loss. Please contact our technical personnel for the complete availability of "custom made" shafts.

## 4 - Electrical connections



### WARNING

Power supply must be turned off before performing any electrical connection! Installation, electrical connection and maintenance operations must be carried out by qualified personnel only, with power supply disconnected. Mechanical components must be in stop.

Do not remove the connection cap of the encoder. Damage may be caused to internal components.



No user serviceable parts inside the connection cap!

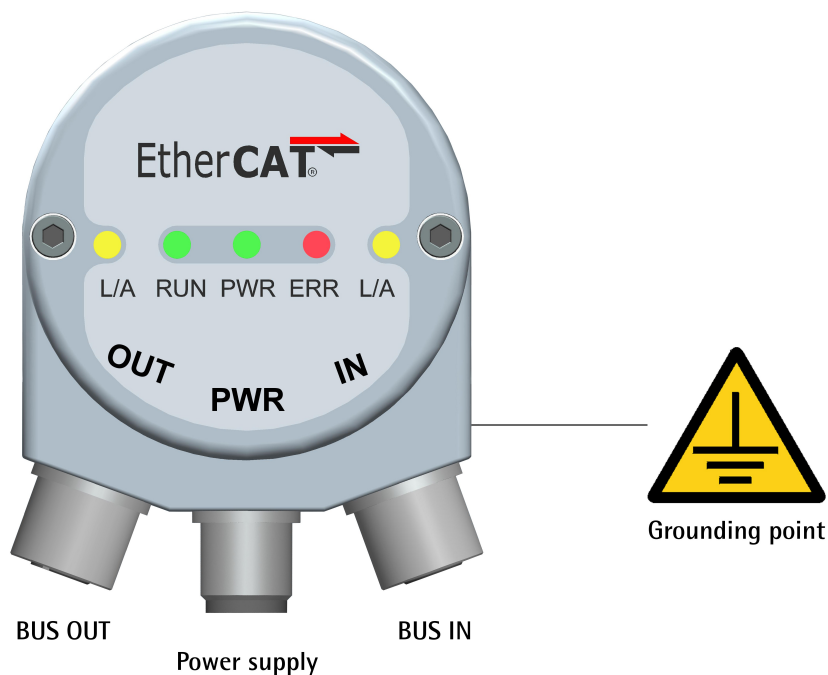


Figure 1

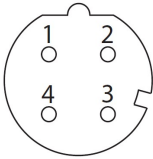
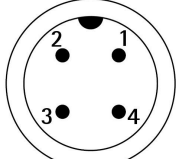
### 4.1 EtherCAT interface and power supply connectors

The connection cap is fitted with three M12 connectors with pin-out in compliance with the EtherCAT® standard. Therefore you can use standard EtherCAT cables commercially available.



Please note that input ECATIN (BUS IN) and output ECATOUT (BUS OUT) connectors are not interchangeable! BUS IN connector must be networked towards the EtherCAT Master.

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

M12 4-pin (frontal side)	BUS IN & BUS OUT	POWER SUPPLY
	 <p>D coding female</p>	 <p>A coding male</p>
Pin	Description	Description
1	Tx Data +	+10Vdc +30Vdc
2	Rx Data +	not connected
3	Tx Data -	0Vdc
4	Rx Data -	not connected

#### 4.2 Network configuration: topologies, cables, hubs, switches - Recommendations

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

Line, tree or star: EtherCAT supports almost any topology. The bus or line structure known from the fieldbuses thus also becomes available for Ethernet, without the quantity limitations implied by cascaded switches or hubs.

The Fast Ethernet physics (100BASE-TX) enables a cable length of 100 m (328 ft) between two devices. Since up to 65,535 devices can be connected, the size of the network is almost unlimited.

The Ethernet protocol according to IEEE 802.3 remains intact right up to the individual device; no sub-bus is required. In order to meet the requirements of a modular device like an electronic terminal block, the physical layer in the coupling device can be converted from twisted pair or optical fiber to LVDS (alternative Ethernet physical layer, standardized in [4.5]). A modular device can thus be extended very cost-efficiently. Subsequent conversion from the backplane physical layer LVDS to the 100BASE-TX physical layer is possible at any time – as usual with Ethernet.

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

### 4.3 Addressing

It is not necessary to assign a physical address to the device because the addressing of the Slave is automatic at power-on during the initial scanning of the hardware configuration.

The field for addressing is 32-bit long, there are three kinds of addressing:

- Auto Increment Addressing = Position Addressing: 16 bits indicate the physical position of the Slave inside the network while 16 bits are scheduled for local memory addressing; when the Slave receives the frame then it increments the position address and the Slave receiving address 0 is the addressed device;
- Fixed Addressing = 16 bits indicate the physical address of the Slave inside the network while 16 bits are scheduled for addressing the local memory;
- Logical Address = the Slave is not provided with its own individual address, but it can read and write data in a section of the total memory space available (4 Gigabytes).

For complete information refer to the "7.1.5 Addressing" section on page 55.

### 4.4 Line Termination

EtherCAT network needs no line termination because the line is terminated automatically; in fact every Slave is able to detect the presence of the downstream Slaves. For complete information refer to the "7.1.4 Line Termination" section on page 54.

### 4.5 Ground connection

To minimize noise connect properly the shield and/or the connector housing and/or the frame to ground. Connect properly the cable shield to ground on user's side. Lika's EC- pre-assembled cables are fitted with shield connection to the connector ring nut in order to allow grounding through the body of the device. Lika's E- connectors have a plastic gland, thus grounding is not possible. If metal connectors are used, connect the cable shield properly as recommended by the manufacturer. 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. We suggest using the ground point provided in the cap (see Figure 1, use 1 TCEI M3 x 6 cylindrical head screw with 2 tooth lock washers).

#### 4.6 Diagnostic LEDs

Five LEDs located in the rear side of the connection cap are designed to show the operating or fault status of the EtherCAT® interface.

The LEDs operation is according to the EtherCAT specifications, see ETG1300\_S\_R\_V1i1i0\_IndicatorLabelingSpecification.pdf.

LED states	Definition
ON	The indicator shall be constantly ON.
OFF	The indicator shall be constantly OFF.
Flickering	The indicator shall turn ON and OFF iso-phase with a frequency of 10 Hz: ON for 50 ms and OFF for 50 ms.
Blinking	The indicator shall turn ON and OFF iso-phase with a frequency of 2.5 Hz: ON for 200 ms followed by OFF for 200 ms.
Single flash	The indicator shall show one short flash (200 ms) followed by a long OFF phase (1000 ms).
Double flash	The indicator shall show a sequence of two short flashes (200 ms), separated by an OFF phase (200 ms), and followed by a long OFF phase (1000 ms).

LED	Meaning
-----	---------

<b>L/A Link/ Activity (yellow)</b>	It shows the current state of the physical links (IN and OUT) and the activity in the links
OFF	condition: port closed, link: YES, activity: N.A.
FLICKERING	condition: port open, link: YES, activity: YES
ON	condition: port open, link: YES, activity: NO

<b>RUN (green)</b>	It shows the current state of the EtherCAT State Machine (ESM)
OFF	The encoder is in <b>INIT</b> state
BLINKING	The encoder is in <b>PRE-OPERATIONAL</b> state
SINGLE FLASH	The encoder is in <b>SAFE-OPERATIONAL</b> state
ON	The encoder is in <b>OPERATIONAL</b> state
FLICKERING	The encoder is in <b>BOOT</b> state

<b>PWR (green)</b>	It shows the current state of power supply
OFF	The encoder power supply is switched OFF
ON	The encoder power supply is switched ON

<b>ERR (red)</b>	It shows the current error state
OFF	No error
FLICKERING	Error while loading parameters from flash memory at start-up; encoder parameters have not been saved correctly on flash memory
BLINKING	Invalid configuration
SINGLE FLASH	Local error (see ETG1000.6, "EtherCAT Specification - Part 6")

---

DOUBLE FLASH	Watchdog timeout
ON	Memory error and ESC controller not active

## 5 - Getting started



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

- Mechanically install the device (see on page 19);
- perform the electrical and network connections (see on page 24);
- you do not need to set the node address and the transmission rate (see on page 55);
- you do not need to set any line termination (see on page 54);
- switch on the +10Vdc +30Vdc power supply;
- if you want to use the physical resolution (see the **6501-00 Hardware counts per revolution** object and the **6502-00 Hardware number of turns** object), please check that the **Scaling function** parameter is disabled (the bit 2 in the **6000-00 Operating parameters** object = 0; see on page 73);
- otherwise, if you need a custom resolution, enable the **Scaling function** parameter (the bit 2 in the **6000-00 Operating parameters** object = 1; see on page 73) and then set the resolution you need for your application next to the **6001-00 Units per revolution** and **6002-00 Total Measuring Range** objects (see on page 74);
- if you need you can enter the Preset value next to the **6003-00 Preset** object and then set it in the desired position; see on page 77;
- save the new setting values (use the **1010-01 Store parameters** object; see on page 68).

## 6 - Quick reference with TwinCAT

Lika encoders are Slave devices and support "CANopen over EtherCAT" (CoE) mode for data transfer. In particular, they support the "CANopen DS 301 Communication profile".

For any omitted specification on CANopen® protocol, please refer to "CiA Draft Standard Proposal 301. Application Layer and Communication Profile" and "CiA Draft Standard 406. Device profile for encoders" documents available at the address [www.can-cia.org](http://www.can-cia.org).

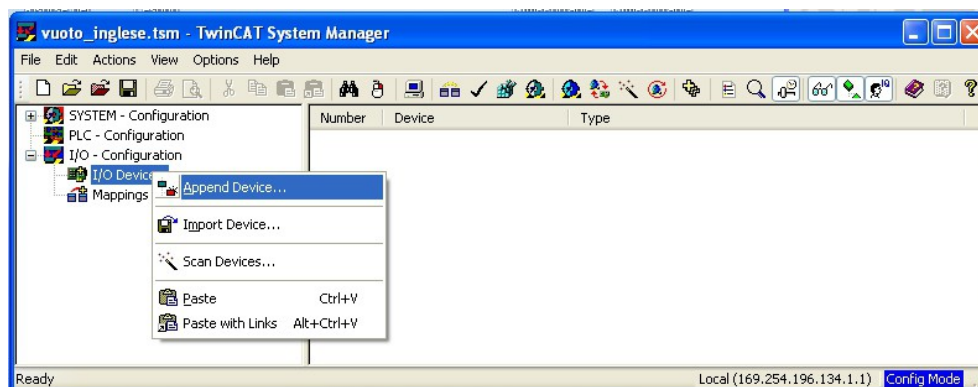
For any omitted specification on EtherCAT® protocol, please refer to "ETG.1000 EtherCAT Specification" documents available at the address [www.ethercat.org](http://www.ethercat.org).

### 6.1 System configuration using TwinCAT software system from Beckhoff

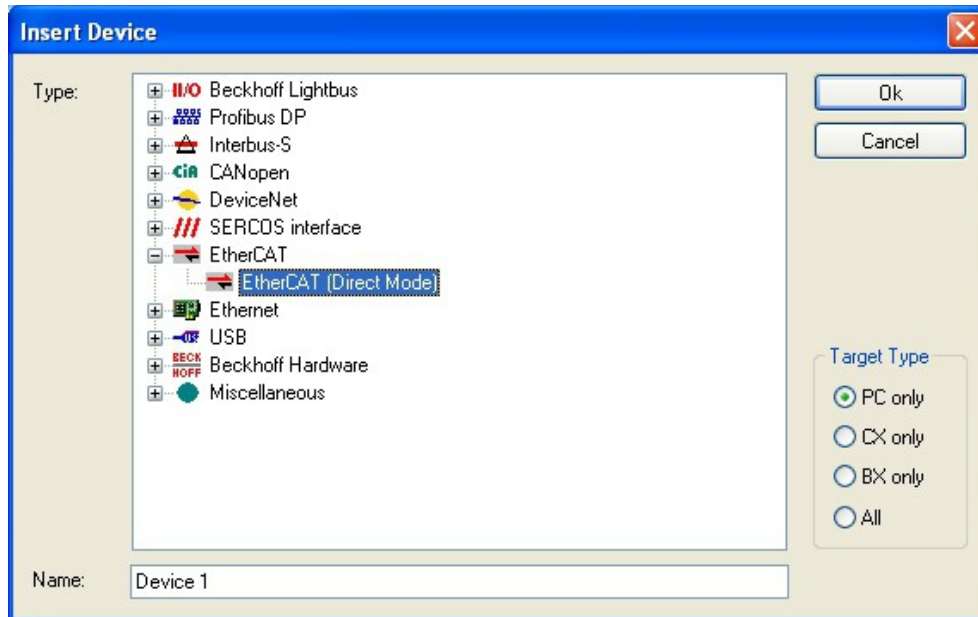
#### 6.1.1 Setting the Network Card

Launch **TwinCAT System Manager**.

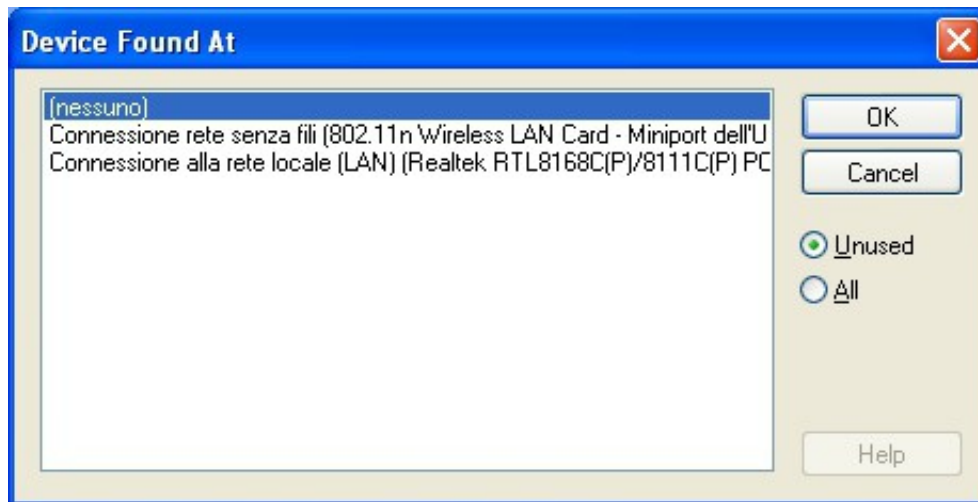
In the left pane of the main window extend the devices tree and select the **I/O Devices** item; right-click the **I/O Devices** item and then press the **Append Device...** command.



In the **Insert Device** window select **EtherCAT** and then **EtherCAT (Direct Mode)** item and confirm pressing the **OK** button.



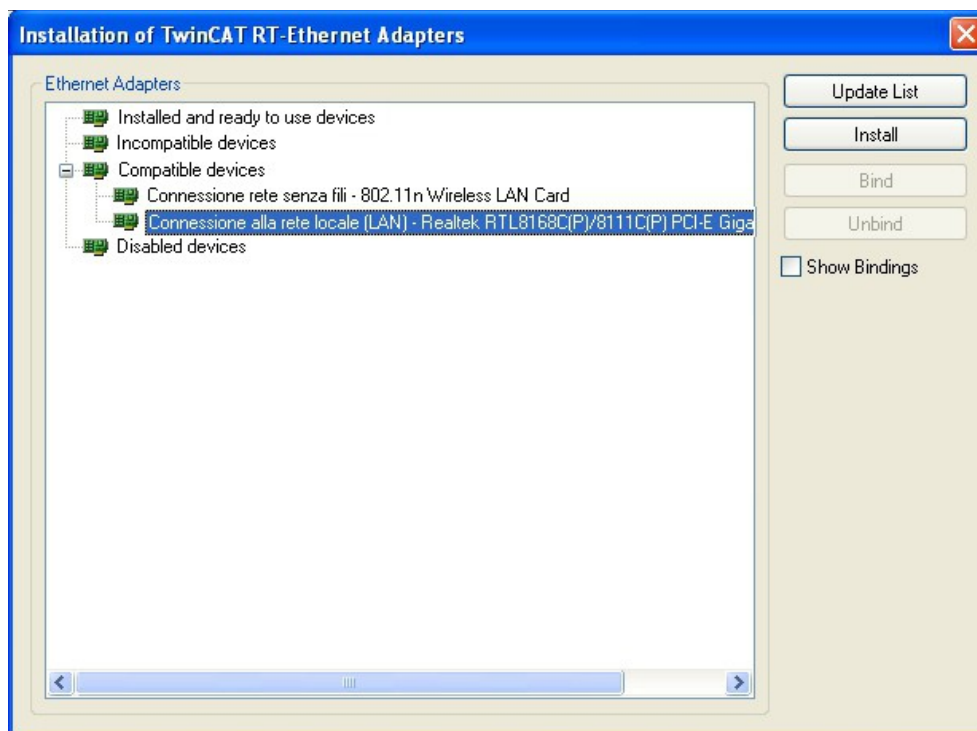
If a network card has been already installed properly, the following window will appear and show the list of the installed devices.



Select the network card you want to use and then confirm the choice by pressing the **OK** button.

If there are no network cards installed, you must install one before proceeding. To do this, on the menu bar of the **TwinCAT System Manager** main window, select the **Options** menu and then press the **Show Real Time Ethernet Compatible Devices...** command.

The **Installation of TwinCAT RT – Ethernet Adapters** window will appear.

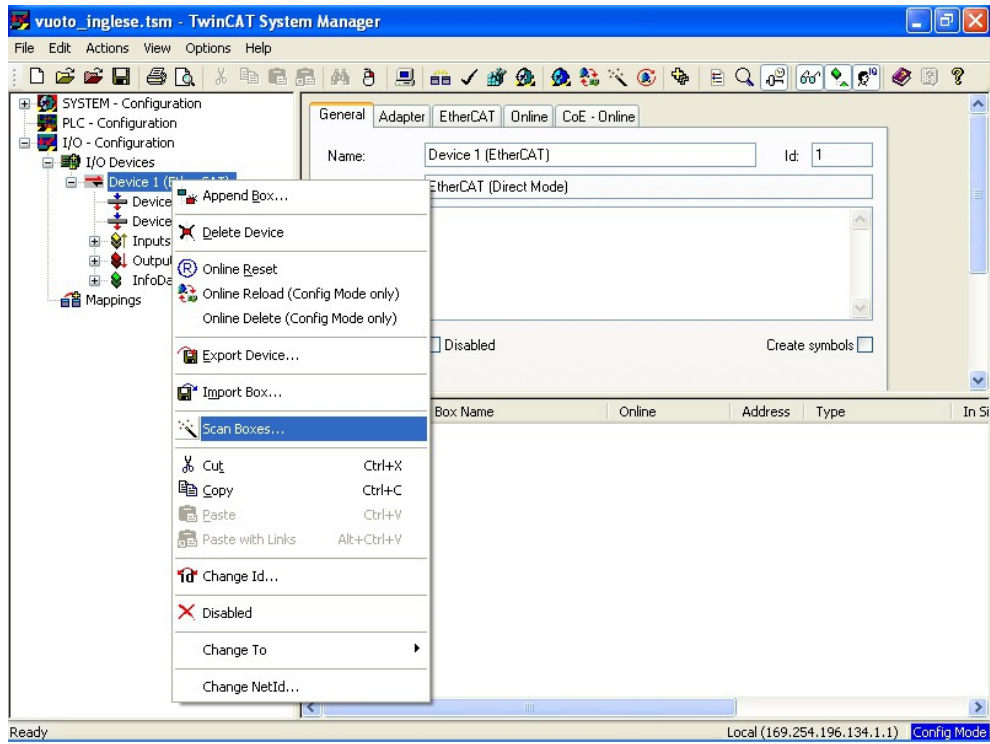


Now select the **Compatible Devices** item and choose the network card you want to install; finally press the **Install** button to confirm your choice.

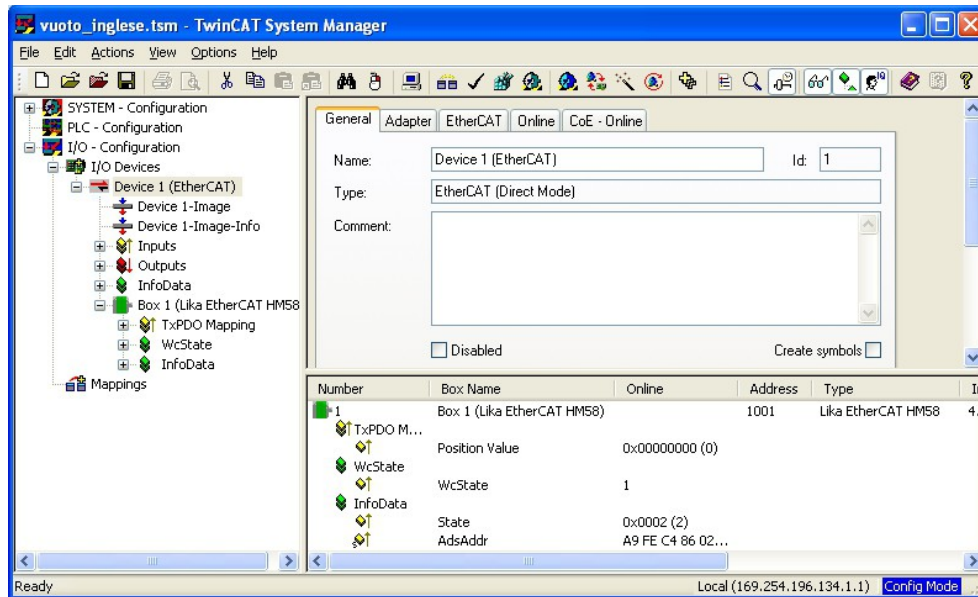


### 6.1.2 Add new I/O modules (Boxes)

If one or more devices are connected to the network and switched ON, right-click the **Device 1 (EtherCAT)** item in the left pane of the **TwinCAT System Manager** main window and press the **Scan Boxes...** command.



At the end of the process some information will be listed in the right page as in Figure here below.



If devices are not already connected to the network it is necessary to use the XML file supplied with the encoder: **Lika\_Ex58\_Hx58\_EC\_Vx.xml** (see at [www.lika.biz](http://www.lika.biz) > **ROTARY ENCODERS** > **ABSOLUTE ENCODERS** > **ETHERCAT**). For older versions of the file please contact Lika Electronic's After Sales Service Dept.



#### WARNING

Before installing the XML file please check that it is compatible with the firmware version and the EEPROM version of the device; the XML file version, the firmware version and the EEPROM version must always comply. For example: if the firmware version is H1\_S4 (Hardware version: 1; Software version: 4), it is mandatory that the EEPROM version is S4, therefore you must then install the XML file version V4. For a description of the EEPROM upgrade procedure please refer to the section "6.6 EEPROM upgrade" on page 42. For a description of the firmware upgrade procedure please refer to the section "6.7 Firmware upgrade" on page 47.



#### WARNING

It is mandatory that in an EtherCAT network all devices are provided with the same version of the firmware, EEPROM and XML file. So when you need to replace an old encoder installed in your network, then you must either upgrade all the encoders in the network to the last version compatible with the new encoder; or you must downgrade the new encoder to the older version compatible with the encoders already installed in the network.



**WARNING**

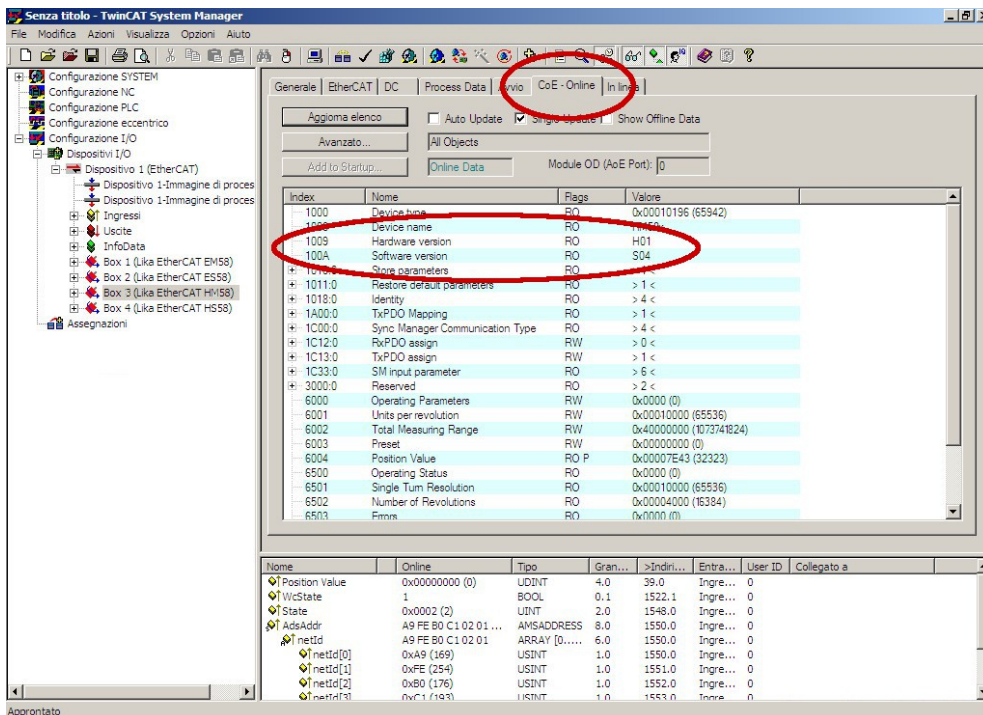
H- series has been implemented starting from version V1.  
E- series has been implemented starting from version V4.



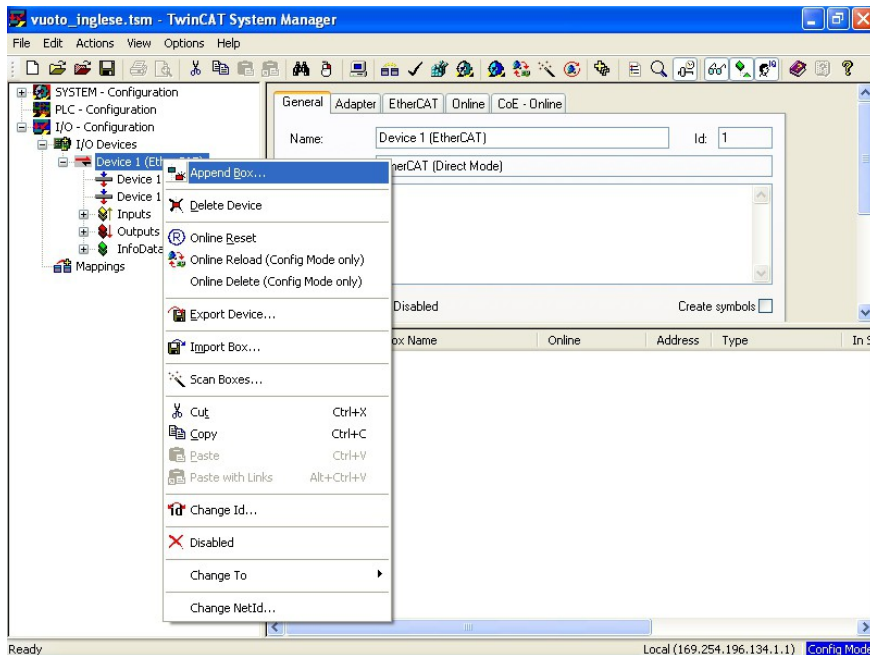
**WARNING**

In the H- series all versions are compatible for upgrade and downgrade except version V1.  
In the E- series all versions are compatible for upgrade and downgrade starting from version V4.

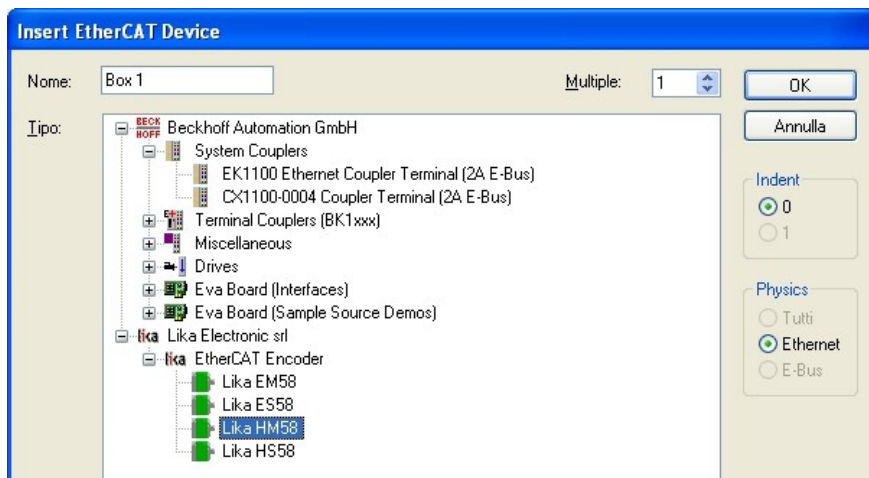
If you want to know the firmware version of a device, press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item in the left pane of the **TwinCAT System Manager** main window: some tabbed pages for configuring and managing the device will appear in the right pane. Enter the **CoE - Online** page and refer to the **1009-00 Hardware version** and **100A-00 Software version** indexes.



Right-click the **Device 1 (EtherCAT)** item in the left pane of the **TwinCAT System Manager** main window and press the **Append Box...** command.



The **Insert EtherCAT Device** window will appear.



In the **Insert EtherCAT Device** window that appears select **Lika Electronic srl** and then **EtherCAT Encoder** items; now choose from the list the encoder you want to install:

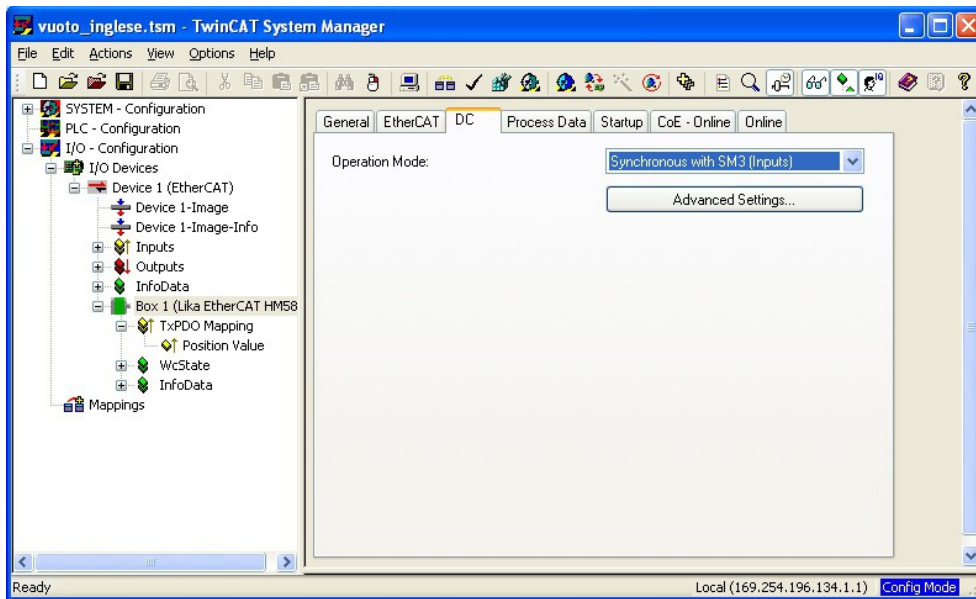
- Lika EM58: multi-turn encoder E- series;
- Lika ES58: single-turn encoder E- series;
- Lika HM58: multi-turn encoder H- series;
- Lika HS58: single-turn encoder H- series.

Press the **OK** button to confirm your choice.

## 6.2 Setting the communication mode

### 6.2.1 Synchronous with SM3

In the left pane of the **TwinCAT System Manager** main window press the **Box** (Lika EtherCAT EM58, ES58, HM58 or HS58) item: some tabbed pages for configuring and managing the device will appear in the right pane. Enter **DC** page. Select the **Synchronous with SM3 (Inputs)** option in the **Operation Mode** box.

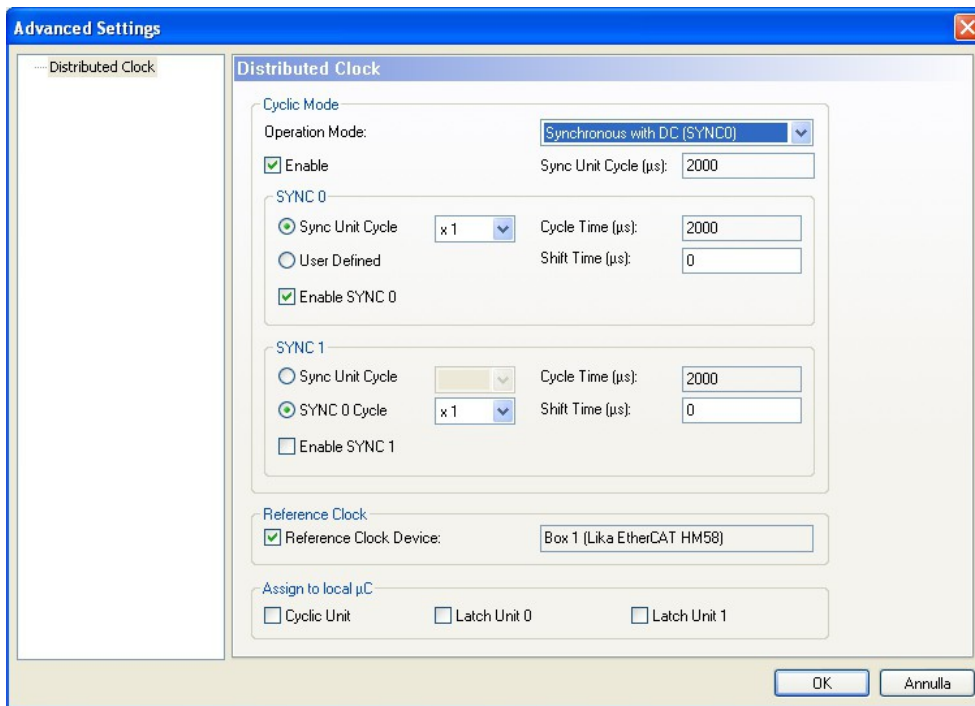


For any further information on the Synchronous with SM3 operation mode please refer to the "Synchronous with SM3" section on page 57 and to the [1C33 Sync Manager input parameter](#) object on page 71.

## 6.2.2 Synchronous with DC (SYNCO)

In the left pane of the **TwinCAT System Manager** main window press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item: some tabbed pages for configuring and managing the device will appear in the right pane. Enter **DC** page.

Select the **Synchronous with DC (SYNCO)** option in the **Operation Mode** box. Then press the **Advanced Settings...** button. The **Advanced Settings** window will appear.

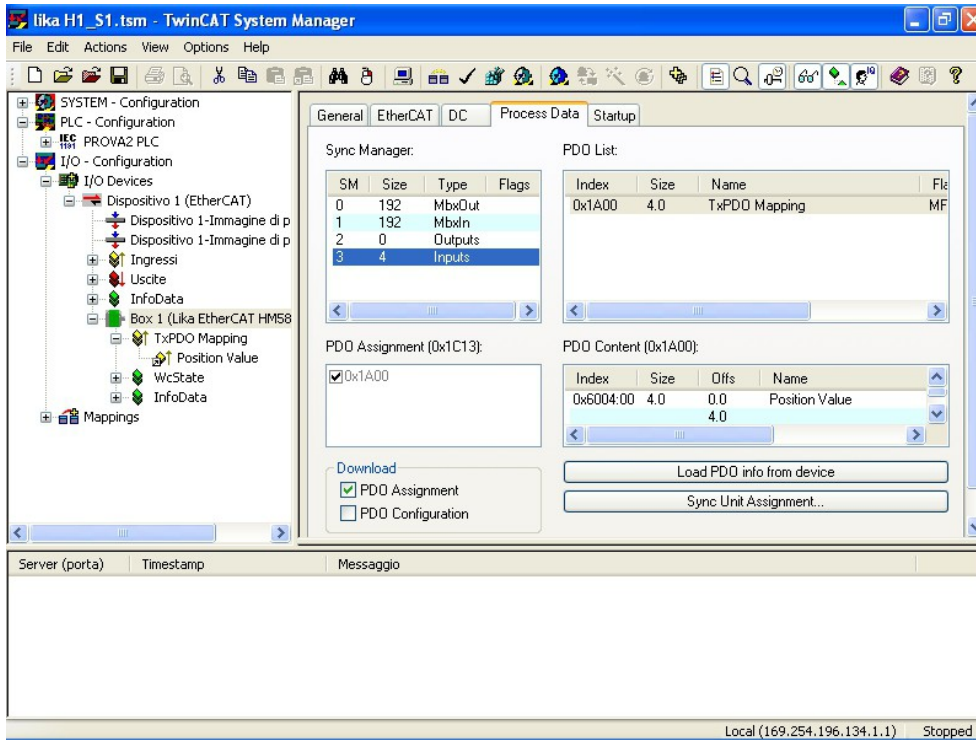


In the section tabbed **SYNC 0** set the cycle time next to the **Sync Unit Cycle** box; sync time is calculated as multiple (or sub-multiple) of the value set in the **Sync Unit Cycle (μs)** item right above.

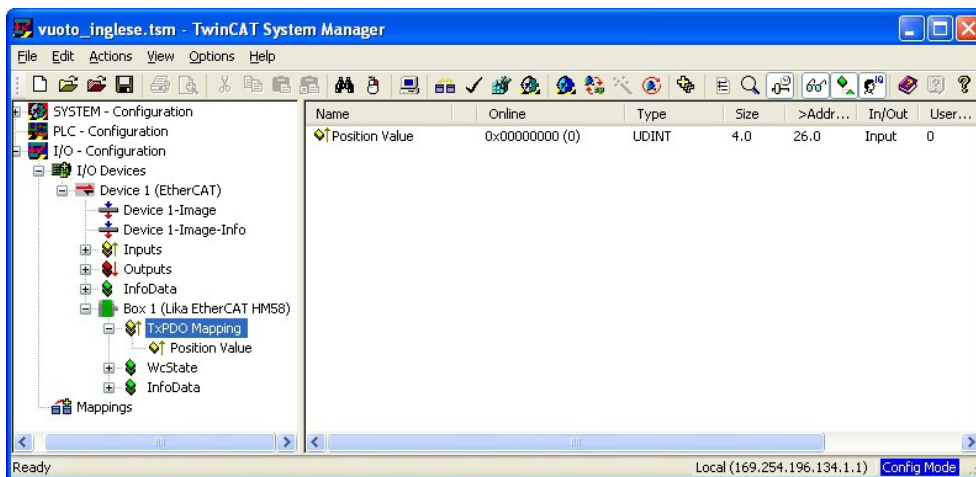
For any further information on the Synchronous with DC operation mode please refer to the "Synchronous with DC SYNCO" section on page 57 and to the **1C33 Sync Manager input parameter** object on page 71.

### 6.3 Process Data Objects

In the left pane of the **TwinCAT System Manager** main window press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item. Expand the box to see Process Data Outputs (PDO). Some tabbed pages for configuring and managing the device will appear in the right pane. Enter the **Process Data** page. In this page process data objects (TxPDO Mapping) are shown.

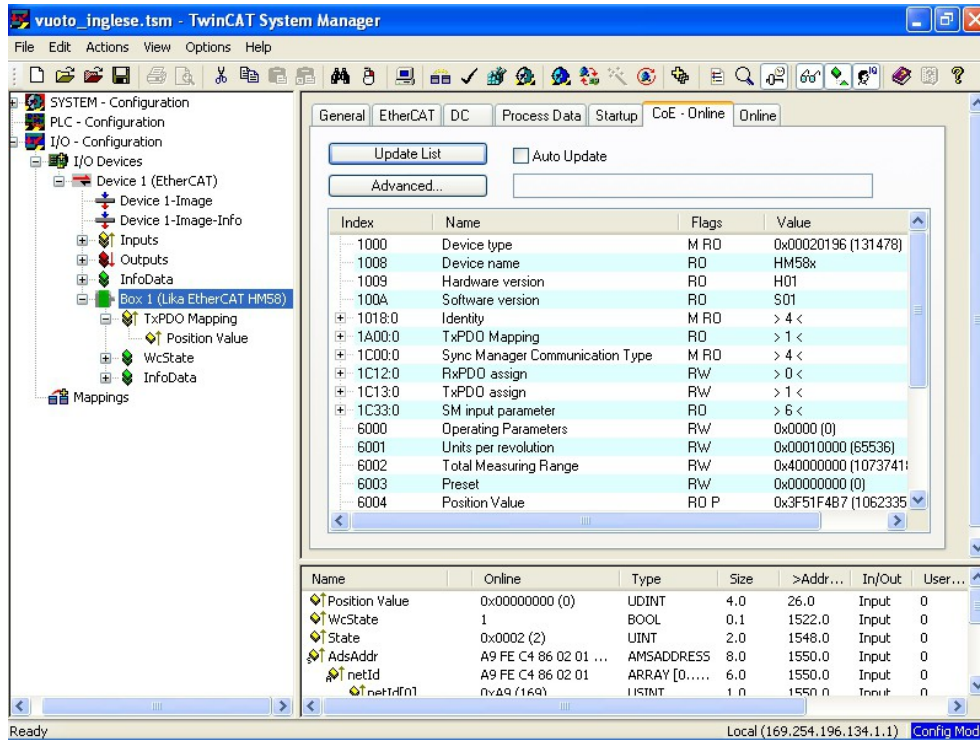


Process data objects can be displayed also by pressing the **TxPDO Mapping** item in the left pane of the **TwinCAT System Manager** main window; data is listed in the right pane.

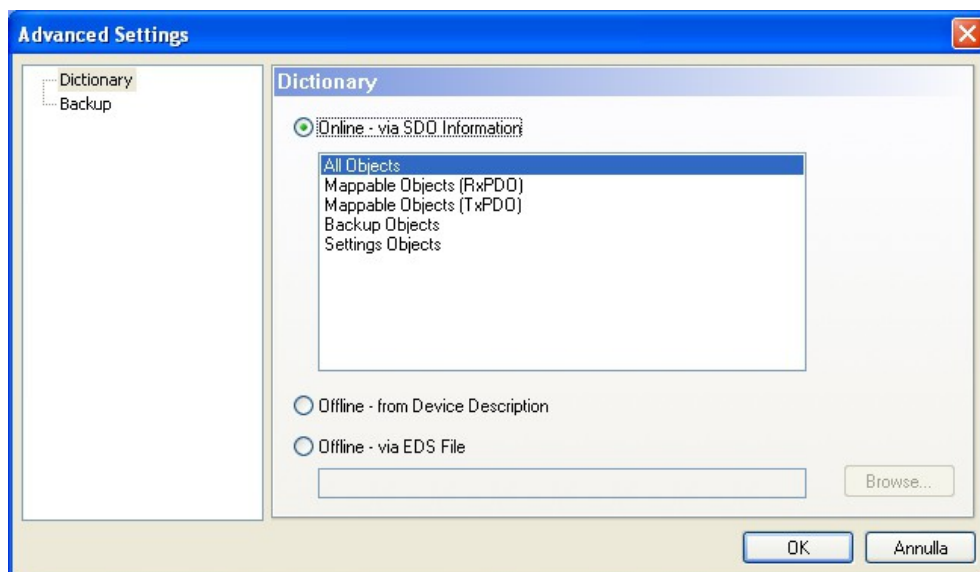


### 6.4 CoE Object Dictionary

In the left pane of the **TwinCAT System Manager** main window press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item: some tabbed pages for configuring and managing the device will appear in the right pane. Enter the **CoE - Online** page. In this page the object dictionary is shown. This is the offline version of the object dictionary as read from the XML file.



Objects can also be read directly from the encoder; to do this click the **Advanced...** button: the **Advanced Settings** window will appear.

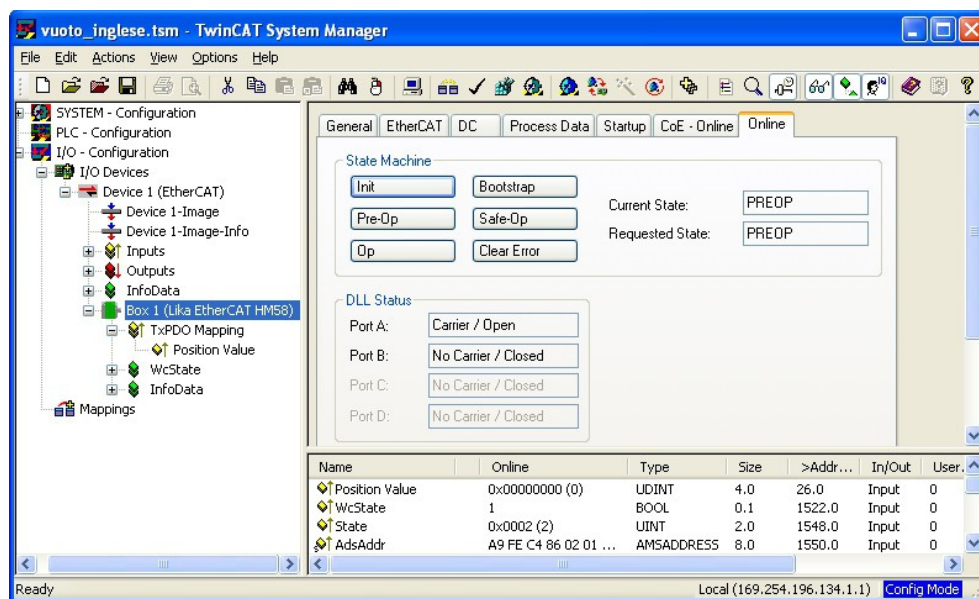




Select the **Dictionary** item in the left pane and then choose the **Online - via SDO Information** option in the **Dictionary** page; press the **OK** button to confirm.

### 6.5 Online Data

In the left pane of the **TwinCAT System Manager** main window press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item: some tabbed pages for configuring and managing the device will appear in the right pane. Enter the **Online** page to check the state of the encoder.



To display the encoder process data in real time, click the **Safe-OP** button if you want to display inputs only; click the **OP** button if you want to display both inputs and outputs.



#### WARNING

The structure of Data Objects (PDOs and SDOs) requires bytes to be sent from the Least Significant Byte (LSB) to the Most Significant Byte (MSB).

On the contrary in TwinCAT write and read data from MSB to LSB.

Furthermore in TwinCAT also strings must be entered in the reverse order:

- read default values: Data byte = 64 61 6F 6Chex = "daol" in ASCII code (means "load" if read in reverse);
- save parameters: Data byte = 65 76 61 73hex = "evas" in ASCII code (means "save" if read in reverse).

## 6.6 EEPROM upgrade



### WARNING

The EEPROM upgrade process 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 EEPROM program is installed, then the unit may not be updated correctly, in some cases preventing the unit from working.



### WARNING

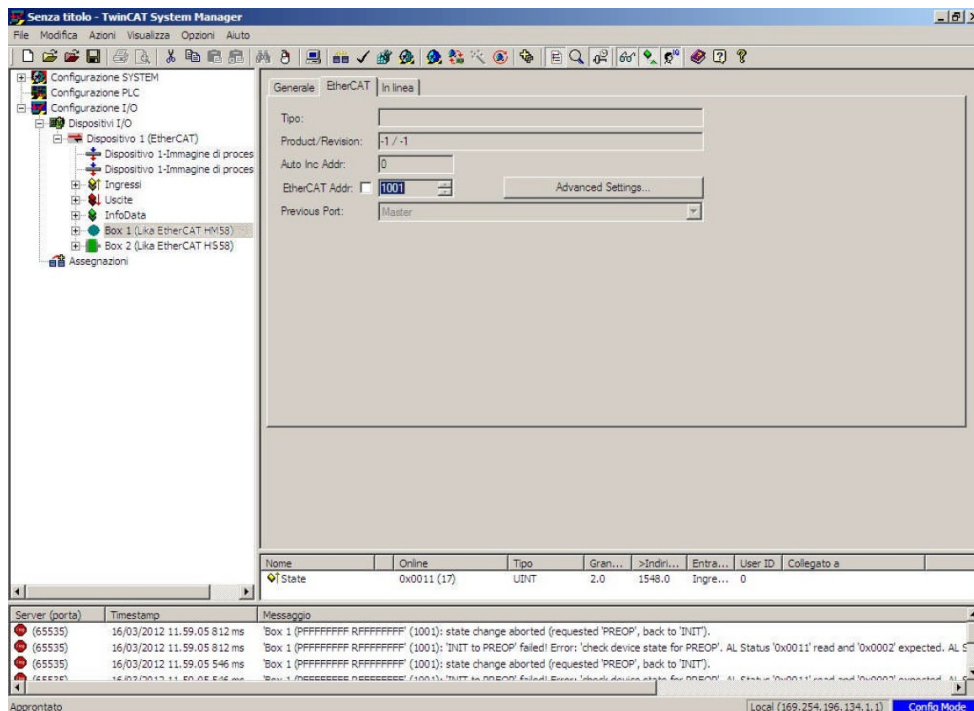
The XML file version, the firmware version and the EEPROM version must always comply. For example: if the firmware version is H1\_S4 (Hardware version: 1; Software version: 4), it is mandatory that the EEPROM version is S4, therefore you must then install the XML file version V4.



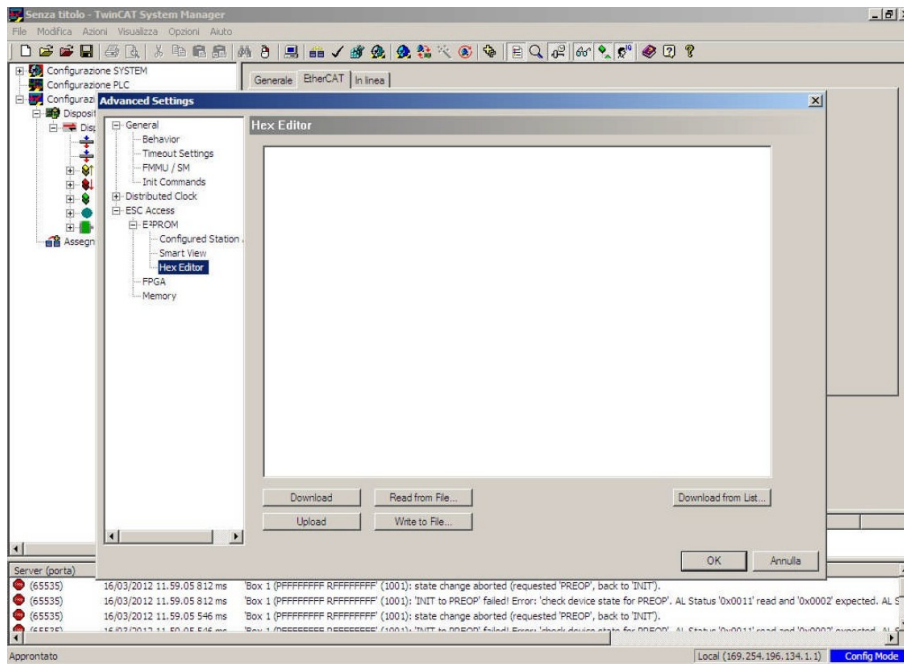
### WARNING

It is mandatory that in an EtherCAT network all devices are provided with the same version of the firmware, EEPROM and XML file. So when you need to replace an old encoder installed in your network, then you must either upgrade all the encoders in the network to the last version compatible with the new encoder; or you must downgrade the new encoder to the older version compatible with the encoders already installed in the network.

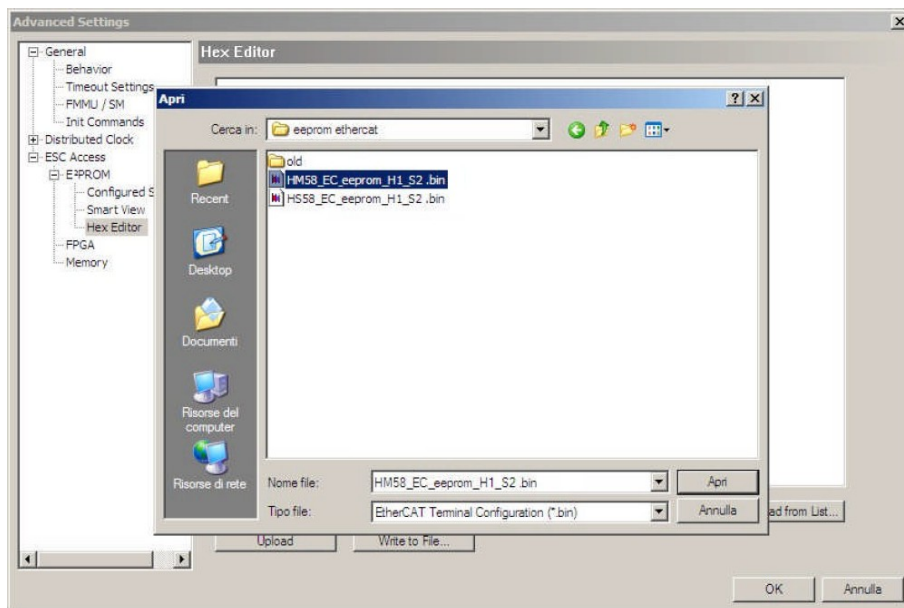
1. In the left pane of the **TwinCAT System Manager** main window press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item of the encoder you need to update: some tabbed pages for configuring and managing the device will appear in the right pane. Enter the **EtherCAT** page.



2. Press the **Advanced Settings...** button; the **Advanced Settings** page will appear; in the directory tree on the left expand the **ESC Access** directory, then expand the **E<sup>2</sup>PROM** directory, finally select the **HEX Editor** item.



3. Press the **Read from File...** button and select the .BIN file provided by Lika Electronic to upgrade the EEPROM; please make sure you select the file suitable for the model you need to upgrade (for example: if you have to upgrade an H- series multiturn encoder then you must select the file **HM58\_EC\_eeprom\_Hx\_Sy.bin**); finally press the **Open** button.

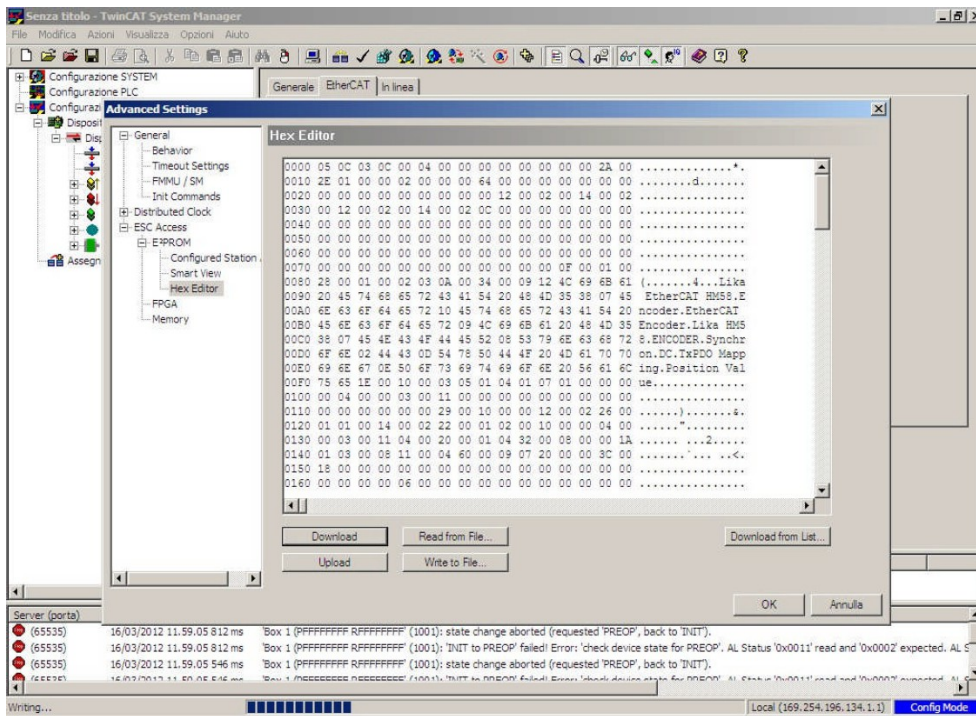




**NOTE**

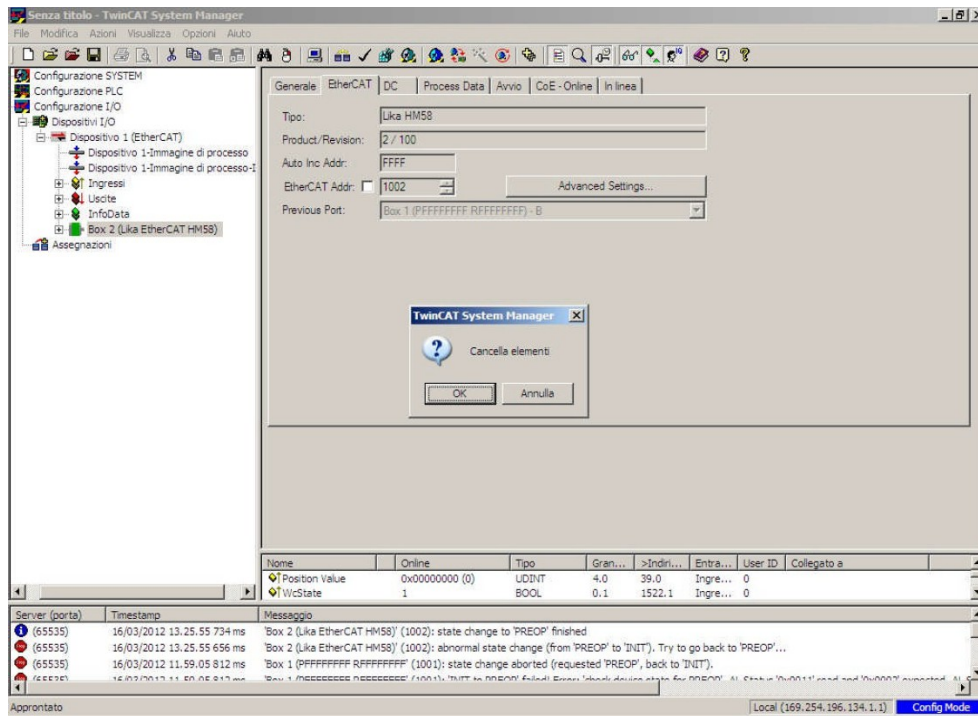
In the .BIN file Hx is the hardware version of the encoder, while Sy is the software version.

4. Move back to the previous **Advanced Settings** page and press the **Download** button. Now wait until the EEPROM writing process is carried out. The progress bar below in the page displays the progress of the operation. As soon as the process is carried out press the **OK** button.

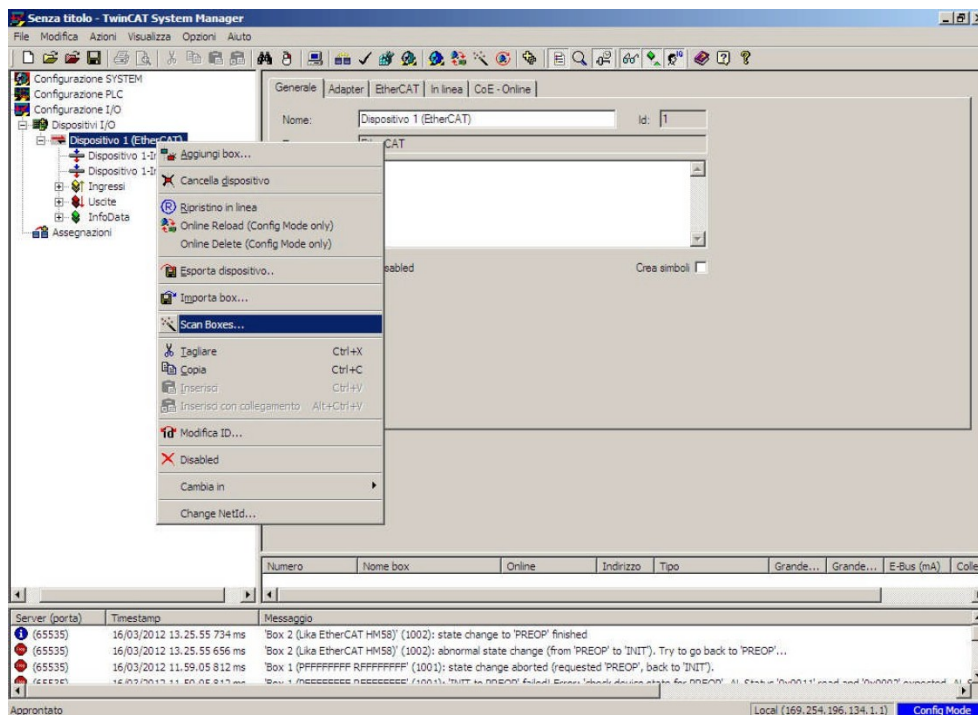


5. Now turn the power supply off, then on again.

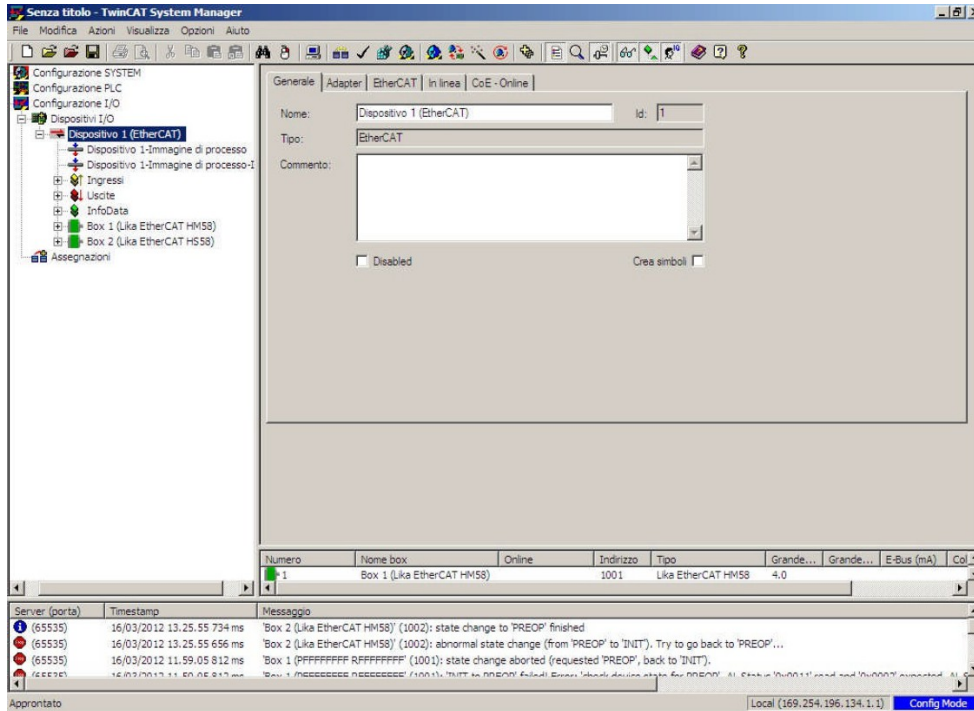
- In the left pane of the **TwinCAT System Manager** main window delete all the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** items in the list. Select each box and then press the **DEL** key in the PC keyboard. Press **OK** to confirm.



- In the left pane of the **TwinCAT System Manager** main window select and right-click the **Device 1 (EtherCAT)** item; press the **Scan Boxes...** command in the menu.



- At the end of the scanning process all the devices available in the network are listed as shown in the Figure here below.



## 6.7 Firmware upgrade



### WARNING

The firmware upgrade process 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 unit from operating.



### WARNING

The XML file version, the firmware version and the EEPROM version must always comply. For example: if the firmware version is H1\_S4 (Hardware version: 1; Software version: 4), it is mandatory that the EEPROM version is S4, therefore you must then install the XML file version V4.



### WARNING

It is mandatory that in an EtherCAT network all devices are provided with the same version of the firmware, EEPROM and XML file. So when you need to replace an old encoder installed in your network, then you must either upgrade all the encoders in the network to the last version compatible with the new encoder; or you must downgrade the new encoder to the older version compatible with the encoders already installed in the network.

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 unit. Lika encoders 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.

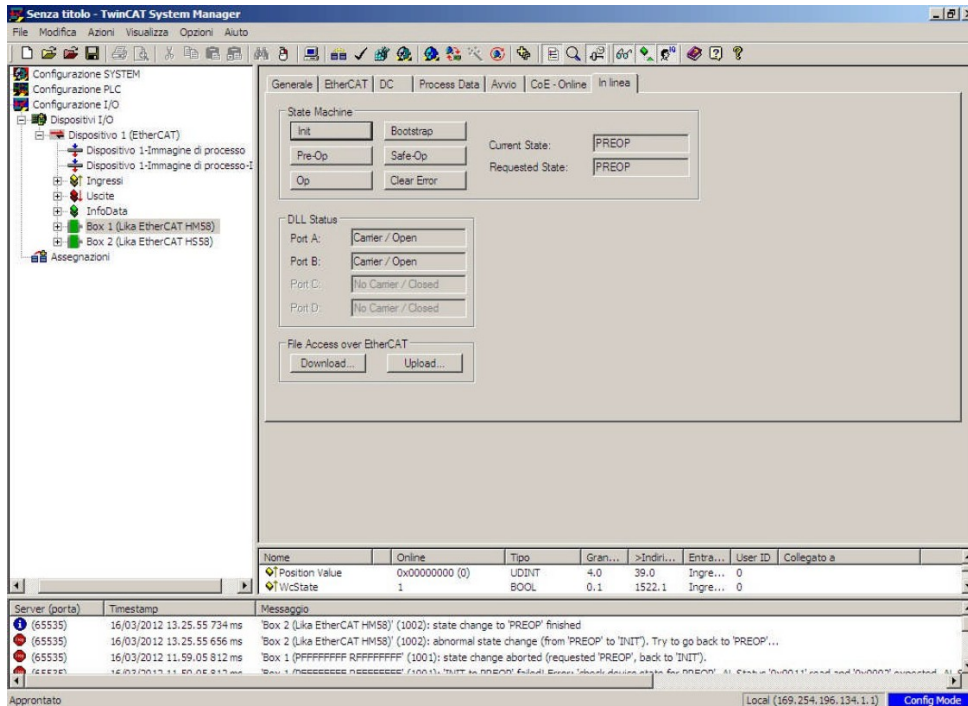
The firmware upgrading program consists of a single file having .EFW extension. It is released by Lika Electronic Technical Assistance & After Sale Service.



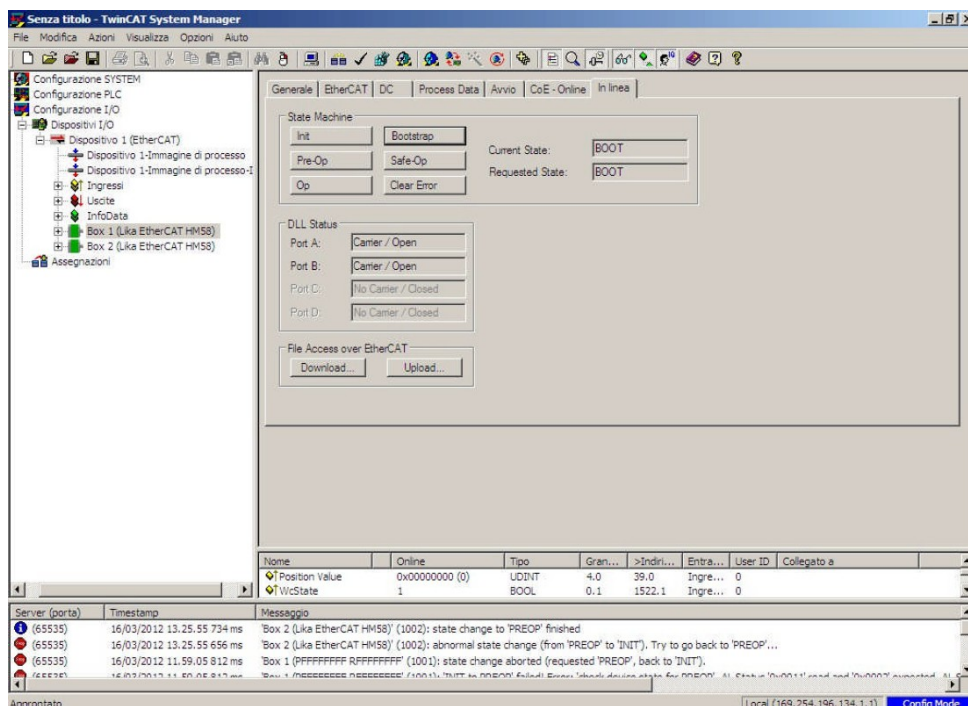
### WARNING

You must upgrade the EEPROM before upgrading the firmware.

1. In the left pane of the **TwinCAT System Manager** main window press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item of the encoder you need to update: some tabbed pages for configuring and managing the device will appear in the right pane. Enter the **Online** page.

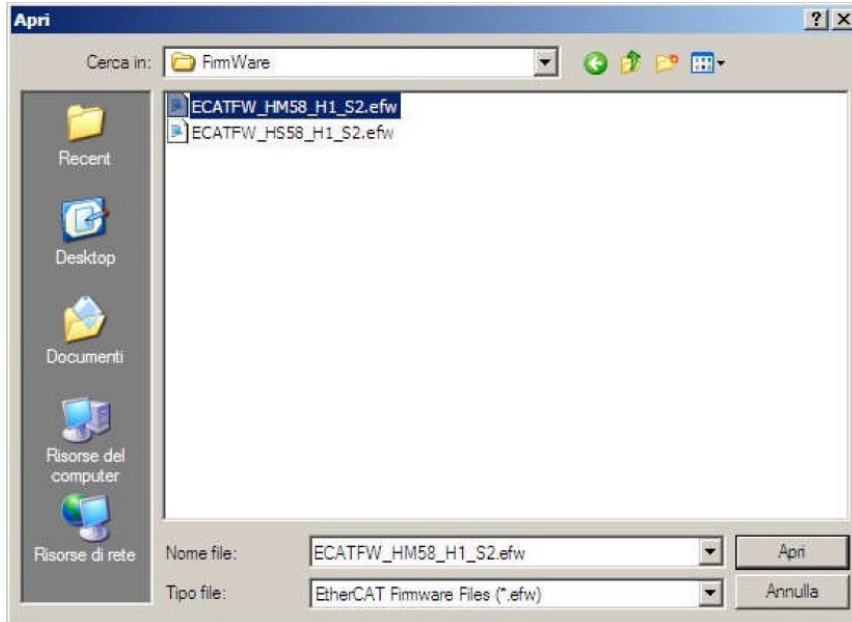


2. Press the **Bootstrap** button in the **State Machine** box; in the **BOOT** state the encoder is ready to accept the firmware upgrade download process (the **BOOT** message appears next to the **Current State** item in the same box).





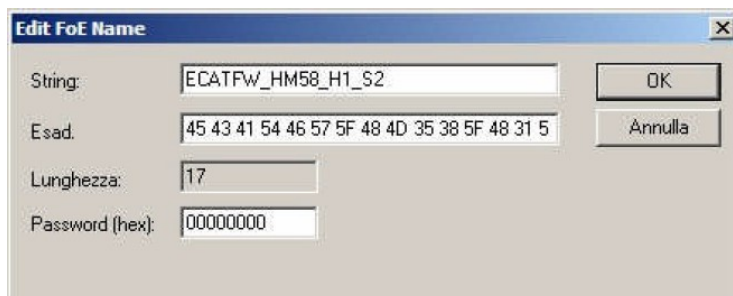
- Now press the **Download...** button in the **File Access Over EtherCAT** box; in the **Open** window that appears select the .EFW file provided to upgrade the firmware; please make sure you select the exact file of model you need to upgrade (for example: if you have to upgrade an H-series multiturn encoder then you must select the file ECATFW\_HM58\_Hx\_Sy.efw); finally press the **Open** button.



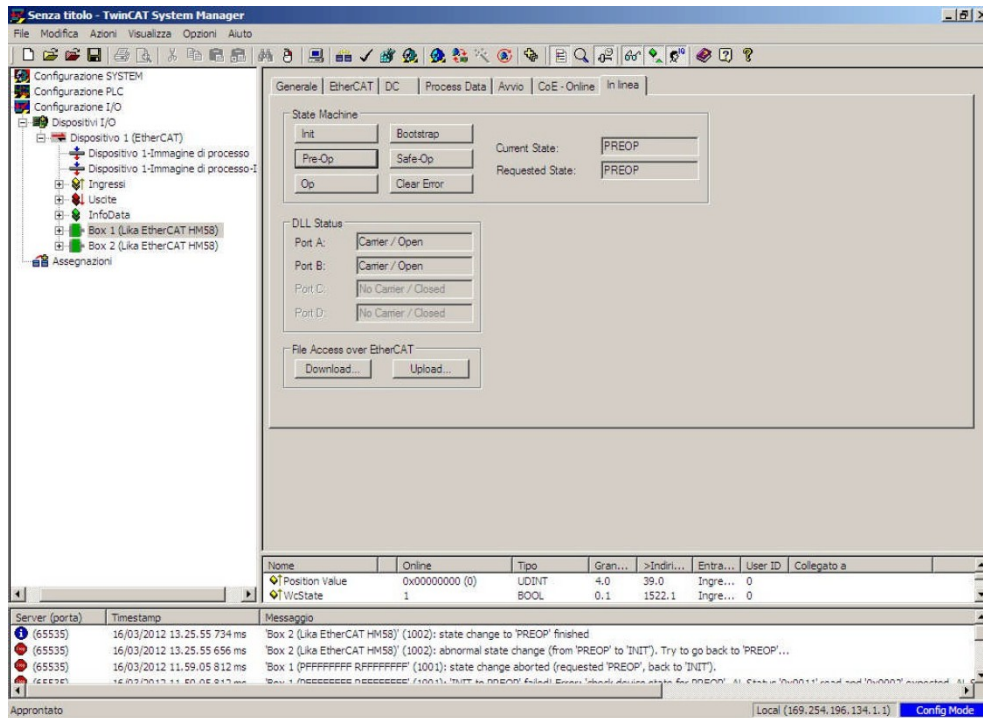
**NOTE**

In the .EFW file Hx is the hardware version of the encoder, while Sy is the software version.

- In the **Edit FoE Name** page that appears on the screen enter the password 00000000hex next to the **Password (hex)** item below in the page and then press the **OK** button to confirm. Now wait until the firmware file saving process is carried out. The progress bar below in the page displays the progress of the operation.



- To check whether the firmware upgrade procedure has been completed successfully enter the **Online** page in the **TwinCAT System Manager** main window and press the **Pre-Op** button in the **State Machine** box; if everything is ok, the encoder enters the **PREOPERATIONAL** state (the **PREOP** message appears next to the **Current State** item in the same box).



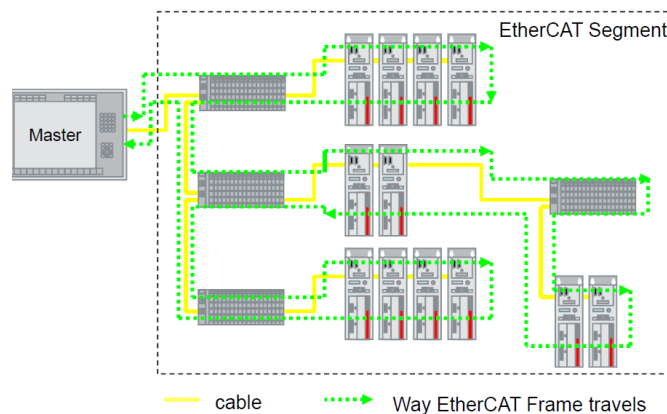


An EtherCAT frame can contain several telegrams and a complete control cycle often requires more than one frame.

### 7.1.1 Data transfer

Usually, in a data bus system, Master controller sends online a data request and then waits for data to be processed and sent back from each Slave node; this does not comply with a real-time system because the Master receives data from the Slaves in different moments and the whole system cannot be synchronized. In EtherCAT the real-time characteristic of the system is quite improved because data are processed "on-the-fly", using one single frame to acquire all data from all Slaves.

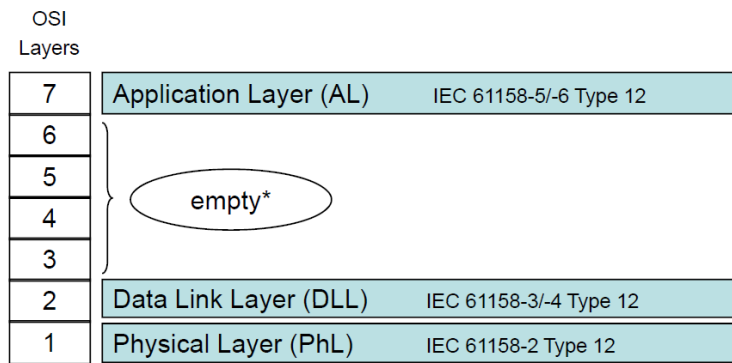
In fact the frame sent by the Master is read by each Slave node the data is addressed to while the telegram passes through the device; similarly, input data is inserted while the telegram passes through. Then the telegram is forwarded to the next device. Telegrams are only delayed by a few nanoseconds. The last Slave issues back the complete frame to the Master with all the requested data (again passing through all the Slaves).



This efficient data flow is guaranteed by the 100BASE-TX full-duplex structure of EtherCAT bus which is fitted with two separated lines for transmitting and receiving data.

Moreover the protocols exchange takes place inside the hardware and it is thus independent from the CPU and the software processing.

### 7.1.2 ISO/OSI Layer model



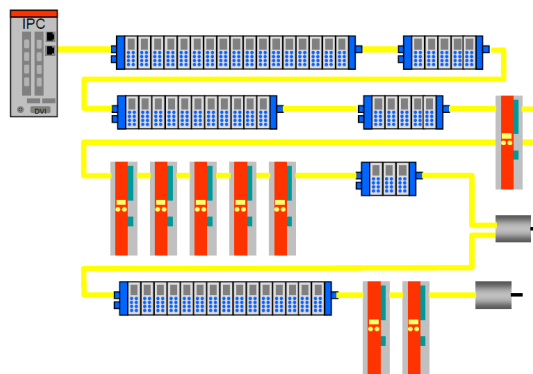
\* "Empty" means that the layer behaviour exists, but is not shown explicitly.

### 7.1.3 Topology

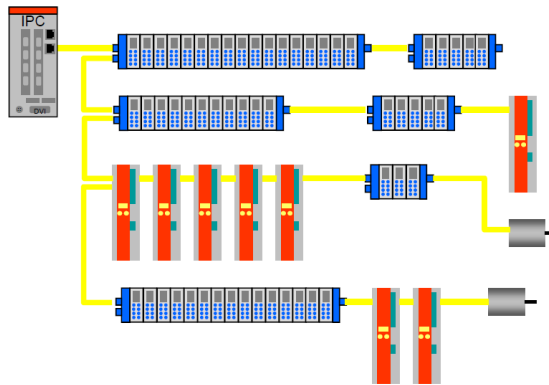
Several topologies of connection are supported by the EtherCAT networks: line, tree, daisy-chain, star, ...). EtherCAT networks can be configured in almost any topology in the same structure. The maximum length of the cable between two Slaves is 100 m / 328 ft; standard EtherCAT cables commercially available can be used.

The choice of the topology depends on the structural characteristics of the plant and it is made in order to reduce the complexity and time for cabling. Inside an EtherCAT network up to 65,535 devices can be connected. Some topology examples are shown in the Figures below:

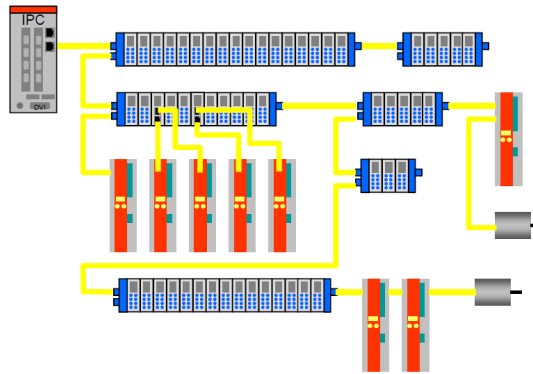
LINE topology:



TREE topology:



DAISY CHAIN with drop lines topology:



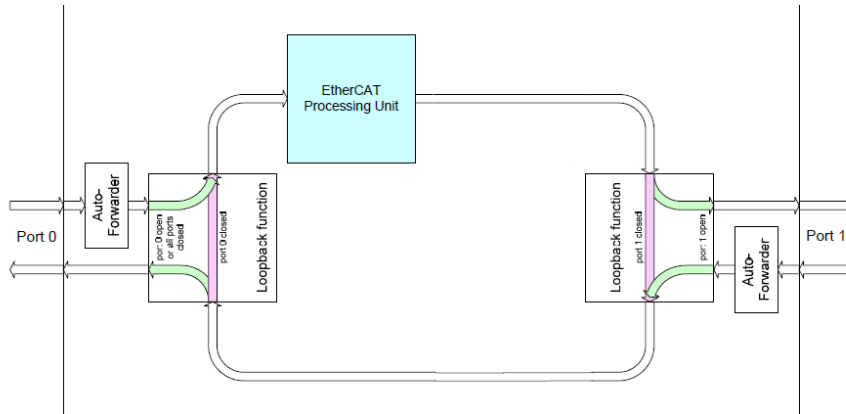
#### 7.1.4 Line Termination

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

An EtherCAT Slave is able to detect the presence of the signal in the outgoing line (Port 0) or in the return line (Port 1).

If the Slave is not able to detect the signal in its return line then it closes the communication ring by short-circuiting the TX signal of its outgoing line with the RX signal of its return line; in this way a telegram received through the outgoing line is processed and sent back through the TX of the return line.

The Slave sends a "carrier signal" or a telegram on TX of the outgoing line continuously and, once the next Slave is connected again, a signal on RX of the return line is detected again; so the short circuit is removed and the telegrams are sent on TX of the outgoing line.



### 7.1.5 Addressing

It is not necessary to assign a physical address to the device (for instance using a dip-switch) because the addressing of the Slave is automatic at power on during the initial scanning of the hardware configuration.

8 Bit	8 Bit	32 Bit		11 Bit	2	1	1	1	16 Bit
Cmd	Idx	Address		Len	R	C	R	M	IRQ
APxx		16 Bit	16 Bit						
		Position	Offset						
FPxx		Address	Offset						
Lxx		Logical Address							

← Auto Increment Addressing (Position addressing)  
 ← Fixed Physical Addressing (Node addressing)  
 ← Logical Addressing

The field for addressing is 32-bit long; there are three kinds of addressing:

- Auto Increment Addressing = Position Addressing: 16 bits indicate the physical position of the Slave inside the network while 16 bits are scheduled for local memory addressing; when the Slave receives the frame then it increments the position address and the Slave receiving the address 0 is the addressed device;
- Fixed Addressing = 16 bits indicate the physical address of the Slave inside the network while 16 bits are scheduled for addressing the local memory;
- Logical Addressing = the Slave is not provided with its own individual address, but it can read and write data in a section of the total memory space available (4 Gigabytes).

### 7.1.6 Communication mode

Lika encoders with EtherCAT interface support the following operating modes:

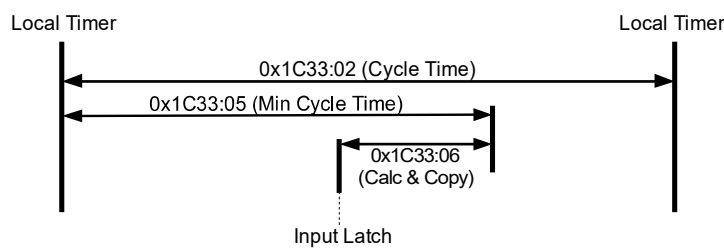
- FreeRun: asynchronous mode;
- SM3 event: synchronous mode;
- DC: distributed clock synchronization mode (synchronous mode).

For a system that requires high performances in real time (closed-loop applications) we suggest using DC mode; if real time requirements are not so mandatory SM3 or Freerun modes can be used instead.

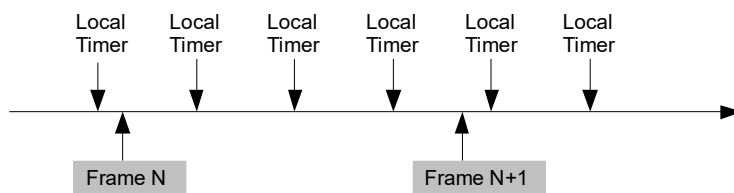
A reference parameter is the "Jitter": it represents the temporal fluctuation of the instant data sampling; in other words data sampled by the micro-controller is available in ECAT DPRAM memory after a certain time and the measure of the variability over time is the "jitter".

#### FreeRun

Asynchronous mode; the encoder position is sampled directly from EtherCAT frame sent by the Master; the position update is performed by an internal timer of the controller every 100 microseconds.



This operating mode has a high sampling jitter (up to 100 microseconds) and can be chosen only when cycle times are quite longer than the jitter if we want to ensure a sufficient real-time system performance.



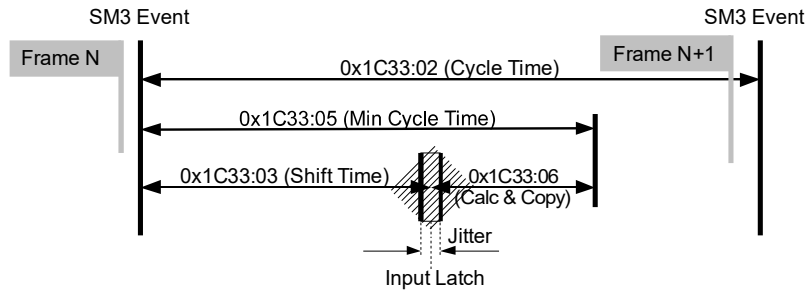
Description	Min	Typ	Max	
Jitter	0		100	µsec
Cycle Time	1000		64000	µsec

See the [1C33 Sync Manager input parameter](#) entry on page 71.



### Synchronous with SM3

In this mode data is sampled and then copied into the Sync Manager buffer as soon as previous data was read from the Master (SM event); in this way new sampled data is synchronous with the Master readings.



New data will be read by the Master at the next cycle (following SM3 event), so if the cycle time is too long, data could be relatively old for a real-time system. The main advantage is that data is updated exactly when Master is reading (synchronous mode).

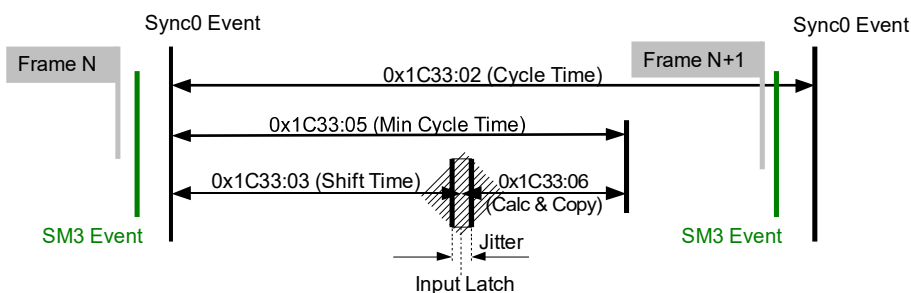
Description	Min	Typ	Max	
Jitter	0		7.2	ns
Cycle Time	62.5		64000	μs

See the [1C33 Sync Manager input parameter](#) entry on page 71.

### Synchronous with DC SYNC0

In this operating mode data is sampled and then copied into the Sync Manager buffer simultaneously at SYNC0 event generated by the ESC capture/compare unit.

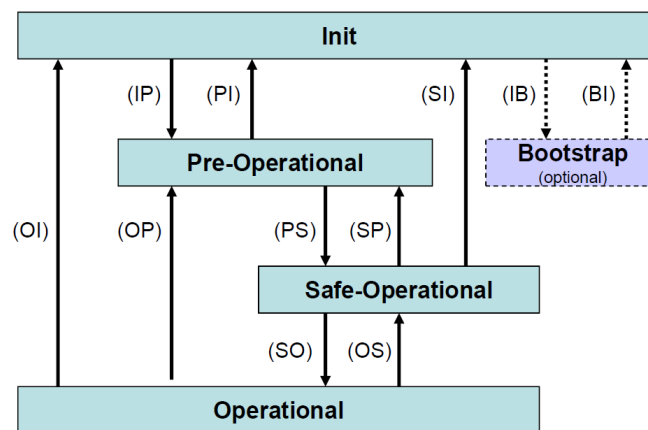
Time required for accomplishing these operations is set in object [1C33 Sync Manager input parameter](#) object; in particular in the **03 Shift Time** entry (1C33hex, sub3) and in the **06 Calc and Copy time** entry (1C33hex, sub6).



In this operating mode "Jitter" is a fundamental parameter in the sampling of two consecutive data. The main advantage of this mode is that there is a direct relation between the sampling instant and the absolute time of the system; in this way, if we know the shift times of the Slaves, we can have an exact image of the system at a given moment (with a tolerance equal to the jitter).

Description	Min	Typ	Max	
Jitter	0	100	200	µsec
Cycle Time	62.5		64000	µsec

### 7.1.7 EtherCAT State Machine (ESM)



EtherCAT Slave is a state machine; the communication and the operating characteristics depend on the current state of the device:

- **INIT**: it is the default state after power-on; in this state there is not direct communication between the Master and the Slave on the Application Layer; some configuration registers are initialized and the Sync Managers are configured.
- **PRE-OPERATIONAL** (PREOP): in this state the mailbox is active; both the Master and the Slave can use the mailbox and its protocols for exchanging specific initialization parameters of the application. Exchange of Process Data (PDO) is forbidden.
- **SAFE-OPERATIONAL** (SAFEOP): in this state the Master and the Slave can issue only input process data, while the output process data is still in the **SAFE-OPERATIONAL** state;
- **OPERATIONAL** (OP): in this state the Master and the Slave are enabled to send both input process data and output process data.
- **BOOTSTRAP** (BOOT): no process data communication. Communication only via mailbox on Application Layer available. Special mailbox configuration is

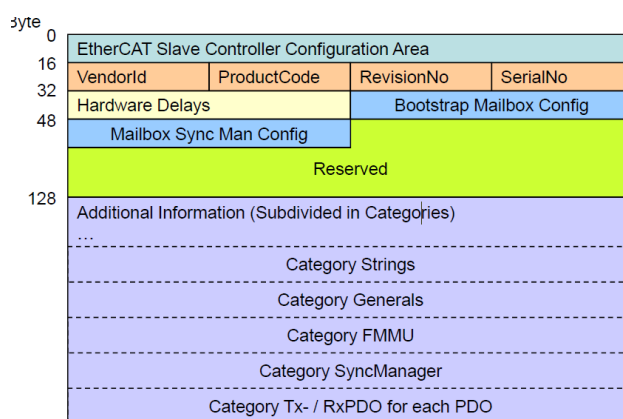
possible, e.g. larger mailbox size. In this state usually the FoE protocol is used for firmware download.

The current state of the Slave is signalled through the green **RUN** LED, see on page 27.

### 7.1.8 Slave configuration

The configuration of the Slave communication characteristics (Sync Manager configuration, addresses, synchronization modes, PDO mapping, ...) can be made both by using the XML file (EtherCat Slave Information - ESI) or by loading data directly from EEPROM (Slave Information Interface SII).

EEPROM content (SII):



### 7.1.9 Timing and Synchronization

The main feature of EtherCAT is its almost ideal representation of a real-time system.

Hence the Master has to synchronize all the Slaves at the same time in order to build a system where all nodes have the same reference time; this goal can be achieved by using "distributed clocks".

The Master downloads its clock into one of the Slaves (customarily the first Slave) which becomes the reference clock for all the Slaves in the network; so it has the task of synchronizing the other Slaves. The Master controller periodically sends a special synchronization-telegram where the reference Slave writes its own "current time". This telegram is then sent to all the other Slaves that, in this way, provide for a new re-synchronization of their own clock in order to avoid possible drifts.

This synchronization of the reference time is very important in order to have an up-to-date "snapshot" of the system and accordingly take simultaneous actions in high sensitive applications such as the coordination in axis control operations.

Besides, the EtherCAT Slave Controller (ESC) is fitted with a capture/compare unit that provides accurate synchronization signals (SYNCO or interrupts): they are sent to the local micro-controller so that it is able to synchronize its own clock to the Slaves clock.

### Sync Manager

Sync Manager has the task of synchronizing data transfer between the Master and the Slave and prevents the same memory area from being written by different events.

There are two synchronization modes:

- 3-Buffer Mode;
- 1-Buffer Mode.

Synchronisation mode is initialized through the XML file or by loading data directly from EEPROM (SII).

### Buffered Mode (3-Buffer Mode)

In this mode new data can be accessed at any time by both the EtherCAT Master and the ESC controllers; no timing restrictions are imposed.

Three buffers are necessary (three consecutive memory areas); one buffer is always available to the ESC controller for writing and one buffer always contains updated data to be read by the Master.

Customarily this mode is used for cyclic data exchange, i.e. process data communication.

### Mailbox Mode (1-Buffer Mode)

In this mode a "handshake" between the Master and the Slave must be used; in fact one only memory buffer is available to both the Master and the Slave for writing and reading; the Master (or the Slave) is enabled to write only when the buffer is empty, that is when the Slave (or the Master) has finished reading the data buffer. And vice versa: the Master (or the Slave) is enabled to read only when the buffer is empty, that is when the Slave (or the Master) has finished writing the data buffer. The mailbox mode is typically used for application layer protocols and exchange of acyclic data (e.g. parameter settings).

The encoder features four Sync Managers, see the [1C00 Sync Manager Communication Type](#) object on page 70:

- **Sync Manager 0 (SM MailBox Receive, SM0)**  
Used for mailbox write transfers (Master to Slave).  
The module has a configurable write mailbox size with default size of 276 bytes, corresponding to 255 bytes plus relevant protocol headers and padding.
- **Sync Manager 1 (SM MailBox Send, SM1)**  
Used for mailbox read transfers (Slave to Master).  
The module has a configurable read mailbox size with default size of 276 bytes, corresponding to 255 bytes plus relevant protocol headers and padding.

- **Sync Manager 2 (SM PDO output, SM2)**  
It contains the RxPDOs (i.e., Sync Manager 2 holds the Read Process Data).
- **Sync Manager 3 (SM PDO input, SM3)**  
It contains the TxPDOs (i.e., Sync Manager 3 holds the Write Process Data).

## 7.2 CANopen Over EtherCAT (CoE)

Lika encoders are Slave devices and support the "CanOpen Over EtherCAT" (COE) mode for data transfer. In particular, they support the "CANopen DS 301 Communication profile", Class 2 and the "CANopen DS 406 Device profile for encoders".

For any omitted specification on CANopen® protocol, please refer to the "CiA Draft Standard Proposal 301. Application Layer and Communication Profile" and to the "CiA Draft Standard 406. Device profile for encoders" documents available at the address [www.can-cia.org](http://www.can-cia.org).

For any omitted specification on EtherCAT® protocol, please refer to the "ETG.1000 EtherCAT Specification" document available at the address [www.ethercat.org](http://www.ethercat.org).

### 7.2.1 XML file

EtherCAT® encoders are supplied with their own XML file **Lika\_Ex58\_Hx58\_EC\_Vx.xml** (see at [www.lika.biz](http://www.lika.biz) > **ROTARY ENCODERS** > **ABSOLUTE ENCODERS** > **ETHERCAT**).

For any information on the firmware upgrade procedure refer to the "6.7 Firmware upgrade" section on page 47. For any information on the EEPROM upgrade procedure refer to the "6.6 EEPROM upgrade" section on page 42.

The XML file has to be installed on EtherCAT® Master device.



#### WARNING

Before installing the XML file please check that it is compatible with the firmware version and the EEPROM version of the device; the XML file version, the firmware version and the EEPROM version must always comply. For example: if the firmware version is H1\_S4 (Hardware version: 1; Software version: 4), it is mandatory that the EEPROM version is S4, therefore you must then install the XML file version V4.



#### WARNING

It is mandatory that in an EtherCAT network all devices are provided with the same version of the firmware, EEPROM and XML file. So when you need to replace an old encoder installed in your network, then you must either upgrade all the encoders in the network to the last version compatible with the new encoder; or you must downgrade the new encoder to the older version compatible with the encoders already installed in the network.



#### WARNING

H- series has been implemented starting from version V1.  
E- series has been implemented starting from version V4.

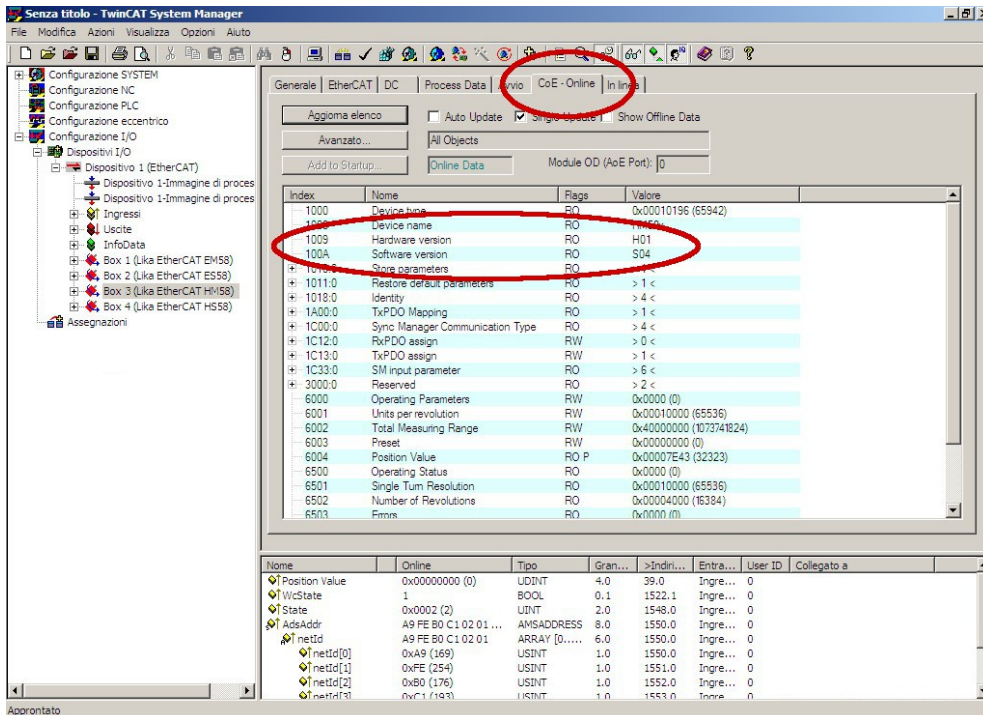


**WARNING**

In the H- series all versions are compatible for upgrade and downgrade except version V1.

In the E- series all versions are compatible for upgrade and downgrade starting from version V4.

If you want to know the firmware version of a device, press the **Box (Lika EtherCAT EM58, ES58, HM58 or HS58)** item in the left pane of the **TwinCAT System Manager** main window: some tabbed pages for configuring and managing the device will appear in the right pane. Enter the **CoE - Online** page and refer to the **1009-00 Hardware version** and **100A-00 Software version** indexes.



**7.2.2 Communication messages**

EtherCAT Datagram of CoE mode has the following structure:

Mbx Header	CoE Cmd			Cmd specific data
type = 3	Number	res	Type	
6 bytes	9 bits	3 bits	3 bits	0 ... 1478 bytes

Mbx Header = 3 CoE mode

Number = 0 in case of SDO messages

≠ 0 in case of PDO messages, it defines the type of service

res reserved bits

- Type = 0 reserved
- = 1 Emergency message
- = 2 SDO request
- = 3 SDO response
- = 4 Transmitted PDO (TxPDO)
- = 5 Received PDO (RxPDO)
- = 6 Remote transmission request of TxPDO
- = 7 Remote transmission request of RxPDO
- = 8 SDO information
- = 9 ... 15 reserved

Cmd specific data      PDO messages: are the process data, e.g. position value  
                                  SDO messages: standard CANopen frame

Transmit (tx) or receive (rx) "Type" is viewed from the Slave side.

### 7.2.3 Process Data Objects (PDO)

PDO messages are used for transmitting or receiving process data in real time; data to be transmitted or received is defined in PDO Mapping and managed by Sync Manager PDO Mapping.

### 7.2.4 Service Data Objects (SDO)

SDO messages are issued via Mailbox (low priority data); Segmented SDO Service and SDO Complete Access are not supported (transfer of low size data and one sub-index at a time).

"CoE Cmd type" = 2 or 3

Structure of "Cmd specific data":

Cmd specific data				
SDO control	Index	Sub index	Data	Data optional
8 bits	16 bits	8 bits	32 bits	1 ... 1470 bytes

SDO control      standard CANopen SDO Service

Index              parameter index

Sub index         parameter sub-index

Data                parameter value

Data optional    optionally, more then 4 bytes of data can be sent in one frame.  
                          Full mailbox size usable.

Index and sub-index values are described in the "Object dictionary".



### 7.2.5 Object dictionary

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

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

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

The **Communication Profile Area** at indexes from 1000h to 1FFFh contains the communication specific parameters for the EtherCAT network. These entries are common to all devices. PDO objects and SDO objects are described in this section. The Communication Profile Area objects comply with the "CiA Draft Standard Proposal 301. Application Layer and Communication Profile". Refer to the "Communication Profile Area objects (DS301)" section on page 67.

The **Manufacturer Specific Profile Area** at indexes from 2000h to 5FFFh is free to add manufacturer-specific functionality. No items are available in this area.

The **Standardised Device Profile Area** at indexes from 6000h to 9FFFh contains all data objects common to a class of devices that can be read or written via the network. The device profiles may use entries from 6000h to 9FFFh to describe the device parameters and the device functionality. The Standardised Device Profile Area objects comply with the "CiA Draft Standard 406 CANopen Device profile for encoders". Refer to the "Standardised Profile Area objects (DS406)" section on page 73.

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

#### Index-subindex Object name

[data types, attribute]

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

Signed8 / Unsigned8 data type:

Process data bytes							
byte 4							
7	6	5	4	3	2	1	0
MSbit		...				LSbit	

Signed16 / Unsigned16 data type:

Process data bytes	
byte 4	byte 5
LSByte	MSByte

Signed32 / Unsigned32 data type:

Process data bytes			
byte 4	byte 5	byte 6	byte 7
LSByte	...	...	MSByte

**NOTE**

Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the **1010-01 Store parameters** object, see on page 68.

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

**Communication Profile Area objects (DS301)****1000-00 Device type**

[Unsigned32, ro]

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

Default = 0001 0196h = singleturn encoder, in compliance with DS 406

0002 0196h = multiturn encoder, in compliance with DS 406

**1008-00 Device Name**

[String, ro]

It shows the manufacturer device name, expressed in hexadecimal ASCII code.

Default = 4553353878h = "ES58x" = singleturn encoder, E- series

454D353878h = "EM58x" = multiturn encoder, E- series

4853353878h = "HS58x" = singleturn encoder, H- series

484D353878h = "HM58x" = multiturn encoder, H- series

**1009-00 Hardware version**

[String, ro]

It shows the hardware version of the device, expressed in hexadecimal ASCII code.

**EXAMPLE**

483031h = H01 = Hardware version 01

Default = device dependent

**100A-00 Software version**

[String, ro]

It shows the software version of the device, expressed in hexadecimal ASCII code.

**EXAMPLE**

533032h = S02 = Software version 02

Default = device dependent

**1010-01 Store parameters**

[Unsigned32, rw]

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

Write "save" in hexadecimal ASCII code in the data bytes:

Master → Encoder

Cmd specific data							
Cmd	Index		Sub	Data			
23	10	10	01	73	61	76	65
				s	a	v	e

Encoder → Master (confirmation)

Cmd specific data							
Cmd	Index		Sub	Data			
60	10	10	01	00	00	00	00

**1011-01 Restore default parameters**

[Unsigned32, rw]

This object allows the operator to restore all parameters to default values. The default parameters are set at the factory by Lika Electronic engineers to allow the operator to run the device for standard operation in a safe mode. A list of machine data and relevant default parameters preset by Lika Electronic engineers is available on page 85.

Write "load" in hexadecimal ASCII code in the data bytes:

Master → Encoder

Cmd specific data							
Cmd	Index		Sub	Data			
23	11	10	01	6C	6F	61	64
				l	o	a	d

Encoder → Master (confirmation)

Cmd specific data							
Cmd	Index		Sub	Data			
60	11	10	01	00	00	00	00



**NOTE**

To save the default values execute the "Store parameters" function (see the **1010-01 Store parameters** object). When the power is turned off, parameters not saved are lost.

**1018 Identity**

[Unsigned8, ro]

This object contains general information about the device. Sub-index 00 contains the number of the entries.

Default = 4

**01 Vendor ID**

[Unsigned32, ro]

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

Default = 0000 012Eh

**02 Product code**

[Unsigned32, ro]

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

Default = 00000001h Lika EtherCAT HS58 and ES58, singleturn encoder

00000002h Lika EtherCAT HM58 and EM58, multiturn encoder

**03 Revision**

[Unsigned32, ro]

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

Default = 00010001h Lika EtherCAT ES58 and EM58 series

00010000h Lika EtherCAT HS58 and HM58 series

7	...	0	15	...	8	23	...	16	31	...	24
Minor revision number						Major revision number					
LSB			...			...			MSB		

**04 Serial number**

[Unsigned32, ro]

It provides the Serial Number of the device. It is 0 if no serial number is provided.

Default = 0000 0000h

**1A00-01 PDO mapping parameter**

[Unsigned8, ro]

This object contains the mapping parameters for the PDOs the EtherCAT device is able to transmit. Sub-index 00 contains the number of entries.

**01 Mapped Object 001**

[Unsigned32, rw]

Sub-index 01 contains the information of the mapped application object 001. The object describes the content of the PDO by its index, sub-index and length. The length contains the length of the application object in bits. This may be used to verify the mapping.

7	0	15	8	23	16	31	24
Length		Sub-Index		Index			
LSB						MSB	

Default = 6004 0020h = **6004-00 Position value** object, length 32 bits

**1C00 Sync Manager Communication Type**

[Unsigned8, ro]

This object contains the number and type of Sync Manager Communication Types supported by the encoder. Sub-index 00 specifies the number of Sync Manager channels. Refer also to the "Sync Manager" section on page 60.

**01 SM MailBox Receive (SM0)**

[Unsigned8, ro]

Used for mailbox write transfers (Master to Slave).

Default = 01h

**02 SM MailBox Send (SM1)**

[Unsigned8, ro]

Used for mailbox read transfers (Slave to Master).

Default = 02h

**03 SM PDO output (SM2)**

[Unsigned8, ro]

It contains the RxPDOs (i.e. Sync Manager 2 holds the Read Process Data).

Default = 03h

**04 SM PDO input (SM3)**

[Unsigned8, ro]

It contains the TxPDOs (i.e. Sync Manager 3 holds the Write Process Data).

Default = 04h

**1C12-00 Sync Manager RxPDO Assigned**

[Unsigned8, ro]

This object specifies whether the device uses Receive PDO messages. This device does not support Receive PDO messages.

Default = 00h

**1C13-01 Sync Manager TxPDO Assigned**

[Unsigned32, ro]

This object specifies whether the device uses Transmit PDO messages. Sub-index 00 specifies the number of entries, i.e. the number of assigned TxPDOs.

**01 Subindex 001**

This device uses TxPDO messages to send the position value.

Default = 0000 1A00h = **1A00-01 PDO mapping parameter** object

**1C33 Sync Manager input parameter**

**1C33 Sync Manager input parameter** object contains the input synchronization parameters. Some of them are calculated dynamically and depend on both the encoder configuration (programmed resolution, counting direction, ...) and the synchronization mode (SM or DC). Sub-index 00 contains the number of entries.

**01 Sync Type**

[Unsigned16, rw]

It allows to select the synchronization mode. For more information refer to page 56.

0: FreeRun: see on page 56;

1: Synchronous with SM3 Event: see on page 57;

2: DC mode synchronous with SYNC0 event: see on page 57.

Default = 1

**02 Cycle time**

[Unsigned32, ro]

This parameter depends on the **01 Sync Type** selected. Application cycle time, i.e. interval between two position samplings (internal timer). The value is expressed in nanoseconds (ns).

If 0 = "FreeRun": interval between two position samplings (internal timer).

If 1 = "Synchronous with SM3": minimum interval between two SM3 events.

If 2 = "DC mode synchronous with SYNC0 event": SYNC0 cycle time.

**03 Shift Time**

[Unsigned32, ro]

Interval between the synchronization event and the moment of inputs latching from hardware. This parameter is calculated dynamically and expressed in nanoseconds (ns).

#### 04 Sync modes supported

[Unsigned16, ro]

It shows the list of the supported synchronization modes.

bit 0: FreeRun (supported)

bit 1: Synchronous with SM3 (supported)

bit 2: Synchronous with DC SYNC0 (supported)

Default = 7

#### 05 Minimum cycle time

[Unsigned32, ro]

Min. duration of the encoder internal cycle time. This parameter is calculated dynamically and depends on the operating parameters and the position value. It is expressed in nanoseconds (ns).

#### 06 Calc and Copy time

[Unsigned32, ro]

Time the internal micro-controller (DSP) needs to make calculations on latched optical reading of position and then copy updated data from local memory to ESC memory (Sync Manager) before they are available to EtherCAT. This parameter is calculated dynamically and depends on the operating parameters and the position value. It is expressed in nanoseconds (ns).



#### NOTE

Always save the new values after setting in order to store them in the non-volatile memory permanently. Use the [1010-01 Store parameters](#) object, see on page 68.

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



Standardised Profile Area objects (DS406)

6000-00 Operating parameters

[Unsigned16, rw]

Bit	Function	bit = 0	bit = 1
0	<b>Code sequence</b>	<b>CW (clockwise)</b>	CCW (counter clockwise)
1	not used		
2	<b>Scaling function</b>	<b>Disabled</b>	Enabled
3 ... 15	not used		

Default values are highlighted in bold

Default = 0000h

**Code sequence**

This is intended to set whether the count is increasing (count up information) when the shaft of the rotary encoder rotates clockwise (CW) or counter-clockwise (CCW). Setting 0 (bit 0 = 0) causes the encoder counting to increase when the encoder shaft rotates clockwise; setting 1 (bit 0 = 1) causes the encoder counting to increase when the encoder shaft rotates counter-clockwise. CW and CCW rotations are viewed from the shaft end.

To know whether the **Code sequence** is currently set to CW or CCW, you can read the bit 0 **Code sequence** in the **6500-00 Operating status** object, see on page 79.



**WARNING**

Every time you change the **Code sequence**, then you are required to set a new preset value (see the **6003-00 Preset** object) and finally save the new parameters (see the **1010-01 Store parameters** object).

**Scaling function**

This is meant to disable (0) / enable (1) the scaled parameters **6001-00 Units per revolution** and **6002-00 Total Measuring Range**.

When the scaling function is disabled (bit 2 = 0), the encoder uses the physical singleturn resolution and the physical multiturn resolution (i.e. the hardware counts per revolution and the number of hardware revolutions, see the encoder identification label and the objects **6501-00 Hardware counts per revolution** and **6502-00 Hardware number of turns**) to arrange the absolute position information; the **6001-00 Units per revolution** and **6002-00 Total Measuring Range** objects are ignored.

On the contrary, when the scaling function is enabled (bit 2 = 1), the user is allowed to enter the custom singleturn resolution in the **6001-00 Units per revolution** object and the custom total resolution in the **6002-00 Total Measuring Range** object and these values are used to calculate the position information.

To know whether the **Scaling function** is currently enabled, you can read the bit 2 **Scaling function** of the **6500-00 Operating status** object, see on page 79.



#### WARNING

Every time you enable the scaling function and/or change the scaling values (see the **6001-00 Units per revolution** and **6002-00 Total Measuring Range** objects), then you are required to set a new preset value (see the **6003-00 Preset** object) and finally save the new parameters (see the **1010-01 Store parameters** object).

#### 6001-00 Units per revolution

[Unsigned32, rw]



#### WARNING

This object is active only if the bit 2 **Scaling function** in the **6000-00 Operating parameters** object is set to "=1"; otherwise it is ignored and the system uses the physical values (**6501-00 Hardware counts per revolution** and **6502-00 Hardware number of turns**) to calculate the position information.

This object sets a custom number of distinguishable steps per revolution (custom singleturn resolution).

To avoid counting errors, check that

$$\frac{\text{6501-00 Hardware counts per revolution}}{\text{6001-00 Units per revolution}} = \text{integer value}$$

You are allowed to set whatever integer value less than or equal to the **maximum number of physical steps per revolution** (see the hardware counts per revolution in the encoder identification label and the **6501-00 Hardware counts per revolution** object).

Default = 0000 2000h (8,192) for EM58 series  
 0040 0000h (262,144) for HS58 series  
 0001 0000h (65,536) for HM58 series



#### WARNING

When you set a new value next to the **6001-00 Units per revolution** object, please always check also the **6002-00 Total Measuring Range** object value and be sure that the resulting number of revolutions complies with the **Hardware number of revolutions** of the device (see the **6502-00 Hardware number of turns** object).

$$\frac{6002-00 \text{ Total Measuring Range}}{6001-00 \text{ Units per revolution}} \leq \text{Number of physical revolutions}$$

Let's suppose that the EM58 encoder is programmed as follows:

**6001-00 Units per revolution:** 8,192 cpr

**6002-00 Total Measuring Range** = 33 554 432<sub>10</sub> = 8,192 (cpr) \* 4,096 (rev.)

Let's set a new singleturn resolution, for instance: **6001-00 Units per revolution** = 360 cpr.

If we do not change the **6002-00 Total Measuring Range** value at the same time, we will get the following result:

$$\text{Number of revolutions} = \frac{33\ 554\ 432 \text{ (6002-00 Total Measuring Range)}}{360 \text{ (6001-00 Units per revolution)}} = 93,206.755\dots$$

As you can see, the encoder is required to carry out more than 93,000 revolutions, this cannot be because the hardware number of revolutions can be max. 16,384. When this happens, the encoder falls into an error signalling the faulty condition through the diagnostic LEDs (see on page 27).



**WARNING**

When you enable the scaling function (bit 2 **Scaling function** = 1), please enter scaled values next to the **6001-00 Units per revolution** and **6002-00 Total Measuring Range** objects that are consistent with the physical values. In the case of inconsistent values, the system will warn about the wrong parametrization and fault condition by means of the dedicated objects and visually by means of the diagnostic LEDs.



**WARNING**

Every time you change the scaled values (see the **6001-00 Units per revolution** and **6002-00 Total Measuring Range** objects), then you are required to set a new preset value (see the **6003-00 Preset** object).

**6002-00 Total Measuring Range**

[Unsigned32, rw]



**WARNING**

This object is active only if the bit 2 **Scaling function** in the **6000-00 Operating parameters** object is set to "=1"; otherwise it is ignored and the system uses the physical values (**6501-00 Hardware counts per revolution**

and **6502-00 Hardware number of turns**) to calculate the position information.

This object sets a custom number of distinguishable steps over the total measuring range. The total resolution of the encoder results from the product of **6001-00 Units per revolution** by the required **Number of revolutions**.

You are allowed to set whatever integer value less than or equal to the **overall hardware resolution** (see the encoder identification label as well as **6501-00 Hardware counts per revolution** and **6502-00 Hardware number of turns** objects). The overall hardware resolution results from:

**6501-00 Hardware counts per revolution** \* **6502-00 Hardware number of turns**.

We recommend the **Number of revolutions** to be set to a power of 2.

The set **Number of revolutions** results from the following calculation:

$$\text{Number of revolutions} = \frac{\text{6002-00 Total Measuring Range}}{\text{6001-00 Units per revolution}}$$

Setting the **Number of revolutions** to a value which is a power of 2 is meant to avoid problems when using the device in endless operations requiring the physical zero to be overstepped. If you set the **Number of revolutions** which is not a power of 2, a counting error is generated before the physical zero.

Default = 0800 0000h (134 217 728) for EM58 series  
 0040 0000h (262 144) for HS58 series  
 4000 0000h (1 073 741 824) for HM58 series



**WARNING**

When you set a new value next to the **6002-00 Total Measuring Range** object, please always check also the **6001-00 Units per revolution** object value and be sure that the resulting number of revolutions complies with the Hardware number of revolutions of the device (max. 16,384 revolutions).

Let's suppose that the encoder is programmed as follows:

**6001-00 Units per revolution:** 8,192 cpr

**6002-00 Total Measuring Range** = 134 217 728<sub>10</sub> = 8,192 (cpr) \* 16,384 (rev.)

Let's set a new total resolution, for instance: **6002-00 Total Measuring Range** = 360.

As the **6002-00 Total Measuring Range** must be greater than or equal to the **6001-00 Units per revolution**, the above setting is not allowed.



**WARNING**

Every time you change the value in this object then you are required to set a new preset value (see the **6003-00 Preset** object) and finally save the new parameters (see the **1010-01 Store parameters** object).

**EXAMPLE**

We install the HM5816/16384EC-6 multiturn rotary encoder.

The physical resolution is as follows:

- **Physical singleturn resolution:** 6501-00 Hardware counts per revolution = 65,536 ( $2^{16}$ )
- **Physical multiturn resolution:** 6502-00 Hardware number of turns = 16,384 turns ( $2^{14}$ )
- **Total hardware resolution:** 6501-00 Hardware counts per revolution \* 6502-00 Hardware number of turns = 1 073 741 824 ( $2^{16} + 2^{14} = 2^{30}$ )

In the specific installation 2,048 counts/rev. \* 1,024 turns are required:

- Enable the scaling function: 6000-00 Operating parameters, bit 2 Scaling function = "1"
- Singleturn resolution: 6001-00 Units per revolution = 2,048 (0000 0800h)
- Multiturn resolution: 6002-00 Total Measuring Range = 2,048 \* 1,024 = 2 097 152 (0020 0000h)

**NOTE**

We suggest setting values which are power of 2 ( $2^n$ : 2, 4, ..., 2048, 4096, 8192,...) to be set in the 6001-00 Units per revolution and 6002-00 Total Measuring Range objects to avoid counting errors.

**WARNING**

If 6001-00 Units per revolution and/or 6002-00 Total Measuring Range values change, the 6003-00 Preset object must be updated according to the new resolution. A new preset operation is required.

**6003-00 Preset**

[Unsigned32, rw]

This object allows to set the encoder position to a Preset value. The Preset function is meant to assign a desired value to a physical position of the encoder. The chosen physical position will get the value set next to this object and all the previous and following positions will get a value according to it. This function can be useful, for instance, when the zero position of the encoder and the zero position of the axis need to match. The preset value will be set for the position of the encoder in the moment when the preset value is sent. We suggest setting the preset value when the encoder is in stop.

Default = 0000 0000h



**EXAMPLE**

Let's take a look at the following example to better understand the preset function and the meaning and use of the related objects and commands: **6003-00 Preset** and **6509-00 Offset**.

The encoder position which is transmitted results from the following calculation:

**Transmitted value = read position** (it does not matter whether the position is physical or scaled) + **6003-00 Preset** - **6509-00 Offset**.

If you never set the **6003-00 Preset** and you never performed the preset setting, then the transmitted value and the read position are necessarily the same as **6003-00 Preset** = 0 and **6509-00 Offset** = 0.

When you set the **6003-00 Preset** and then execute the preset setting, the system saves the current encoder position in the **6509-00 Offset** object. It follows that the transmitted value and the **6003-00 Preset** are the same as read position - **6509-00 Offset** = 0; in other words, the value set next to the **6003-00 Preset** object is paired with the current position of the encoder as you wish.

For example, let's assume that the value "50" is set next to the **6003-00 Preset** object and you execute the preset setting when the encoder position is "1000". In other words, you want to receive the value "50" when the encoder reaches the physical position "1000".

We will obtain the following information sequence:

**Transmitted value = read position** ("1000") + **6003-00 Preset** ("50") - **6509-00 Offset** ("1000") = 50.

The following transmitted value will be:

**Transmitted value = read position** ("1001") + **6003-00 Preset** ("50") - **6509-00 Offset** ("1000") = 51.

And so on.

To set the preset value you must send the following command:

**Set the Preset value 6003-00 Preset** (= 1000 = 3E8h)

Master → Encoder

Cmd specific data							
Cmd	Index		Sub	Data			
23	03	60	00	E8	03	00	00

Encoder → Master (Set confirmation)

Cmd specific data							
Cmd	Index		Sub	Data			
60	03	60	00	00	00	00	00



**NOTE**

- If the scaling function is disabled (see the bit 2 **Scaling function** in the **6000-00 Operating parameters** object = 0), then **6003-00 Preset** must be less than or equal to the **total hardware resolution** (i.e. **6501-00 Hardware counts per revolution** \* **6502-00 Hardware number of turns**) - 1.

- If the scaling function is enabled (see the bit 2 **Scaling function** in the **6000-00 Operating parameters** object = 1), then **6003-00 Preset** must be less than or equal to **6002-00 Total Measuring Range** - 1.



**WARNING**

Check the value in the **6003-00 Preset** object and perform the preset operation every time you change the value next to the **Code sequence** parameter or the **6001-00 Units per revolution** and/or **6002-00 Total Measuring Range** objects.

**6004-00 Position value**

[Unsigned32, ro]

This object contains the information about the current position of the encoder. The output value is scaled according to the scaling parameters, if the scaling function is enabled, see the bit 2 **Scaling function** of the **6000-00 Operating parameters** object. The **6004-00 Position value** object is mapped in the **1A00-01 PDO mapping parameter** object, see on page 70.

**6500-00 Operating status**

[Unsigned16, ro]

Bit	Function	bit = 0	bit = 1
0	<b>Code sequence</b>	CW Clockwise	CCW Counter- clockwise
1	not used		
2	<b>Scaling function</b>	Disabled	Enabled
3 ... 15	not used		

**Code sequence**

It shows the value that is currently set through the bit 0 **Code sequence** in the **6000-00 Operating parameters** object. If the bit is "0" the output encoder position value has been set to increase when the encoder shaft rotates clockwise; if the bit is "1" instead the output encoder position value has been set to increase when the encoder shaft rotates counter-clockwise. To set the code sequence to either CW or CCW you must set the bit 0 **Code sequence** in the **6000-00 Operating parameters** object to 0 / 1. For any further information on setting and using the counting direction function refer to the **6000-00 Operating parameters** object on page 73.

**Scaling function**

It shows the value that is currently set through the bit 2 **Scaling function** in the **6000-00 Operating parameters** object. In other words, it is intended to show whether the scaling function is enabled or disabled. If the bit is "0", the scaling function is disabled; if the value is "1" instead the scaling function is enabled. To disable / enable the scaling function you must set the bit 2 **Scaling**

**function** in the **6000-00 Operating parameters** object to 0 / 1. For any further information on setting and using the scaling function refer to the **6000-00 Operating parameters** object on page 73.

#### 6501-00 Hardware counts per revolution

[Unsigned32, ro]



#### WARNING

This object is active only if the bit 2 **Scaling function** in the **6000-00 Operating parameters** object is set to "=0"; otherwise it is ignored and the system uses the custom values (**6001-00 Units per revolution** and **6002-00 Total Measuring Range**) to calculate the position information.

This object is intended to show the number of physical distinguishable steps provided per each turn by the hardware (physical singleturn resolution, see the hardware counts per revolution in the encoder identification label).

If you want to set a custom singleturn resolution see the **6001-00 Units per revolution** object.

#### 6502-00 Hardware number of turns

[Unsigned32, ro]



#### WARNING

This object is active only if the bit 2 **Scaling function** in the **6000-00 Operating parameters** object is set to "=0"; otherwise it is ignored and the system uses the custom values (**6001-00 Units per revolution** and **6002-00 Total Measuring Range**) to calculate the position information.

This object is intended to show the number of physical turns provided by the hardware of the encoder (physical multiturn resolution, see the hardware revolutions in the encoder identification label).

The **Total hardware resolution** results from **6501-00 Hardware counts per revolution** \* **6502-00 Hardware number of turns**.

If you want to set a custom number of turns see the **6001-00 Units per revolution** and **6002-00 Total Measuring Range** objects.

#### 6503-00 Errors

[Unsigned16, ro]

The corresponding bits of supported errors are set (see the **6504-00 Supported errors** object).

#### 6504-00 Supported errors

[Unsigned16, ro]



This object contains the information on the error alarms supported by the encoder. No error alarms are supported in this encoder.  
Default = 0000h (No errors supported).

#### 6505-00 Warnings

[Unsigned16, ro]

The corresponding bits of supported warnings are set (see the [6506-00 Supported warnings](#) object).

#### 6506-00 Supported warnings

[Unsigned16, ro]

This object contains the information on the warnings supported by the encoder.  
bits 0 ... 11 = not supported  
bit 12 = wrong parameters loaded from flash memory at power-on.  
bits 13 ... 15 = not supported  
Default = 1000h

#### 6509-00 Offset

[Unsigned32, ro]

This object contains the Offset value. As soon as you activate the preset, the current position of the encoder is saved in this object. The offset value is then used in the preset function in order to calculate the encoder position value to be transmitted. To zero set the value in this object you must upload the factory default values (see the [1011-01 Restore default parameters](#) object on page 68).

For any further information on the preset function and the meaning and use of the related objects [6003-00 Preset](#) and [6509-00 Offset](#) refer to page 77.



#### NOTE

To save the new parameters execute the store parameters function (see the [1010-01 Store parameters](#) object on page 68).

When the power is turned off, parameters not saved are lost.

### 7.2.6 SDO Abort codes

SDO transfer could be unsuccessful; causes of error are listed and described in the SDO Abort Codes. Here follows the list of the available SDO Abort Codes. For complete information see ETG1000.6 "EtherCAT Specification – Part 6. Application Layer protocol specification", par. 5.6.2.7.2 table 40.

Abort code	Description
0503 0000h	Toggle bit not changed.
0504 0000h	SDO protocol timeout.
0504 0001h	Client/Server command specifier not valid or unknown.
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write only object.
0601 0002h	Attempt to write a read only object.
0602 0000h	The object does not exist in the object dictionary.
0604 0041h	The object cannot be mapped into the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to a hardware error.
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Subindex does not exist.
0609 0030h	Value range of parameter exceeded (only for write access).
0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.
0609 0036h	Maximum value is less than minimum value.
0800 0000h	General error
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present.

Refer also to the "4.6 Diagnostic LEDs" section on page 27.

### 7.2.7 Emergency Error Codes

Emergency Service is used by the Server for transmitting diagnostic messages to the client using MailBox; Error Codes are listed and described in the ETG1000.6 "EtherCAT Specification – Part 6. Application Layer protocol specification", par. 5.6.4.2 table 50.

Error Code		Error Register	Diagnostic Data				
Byte (0)	Byte (1)	Byte (2)	Byte (3)	Byte (4)	Byte (5)	Byte (6)	Byte (7)

Error Code	<p>State Transition Errors of state machine: (for detailed description see ETG1000.6 par. 5.6.4.3)</p> <p>A000hex: transition error from <b>PRE-OPERATIONAL</b> to <b>SAFE-OPERATIONAL</b></p> <p>A001hex: transition error from <b>SAFE-OPERATIONAL</b> to <b>OPERATIONAL</b></p> <p>Encoder errors:</p> <p>5000hex: <b>Hardware error</b></p> <p>5001hex: <b>Diagnostic data</b> (wrong parameters loaded from flash memory)</p>
Error Register	EtherCAT state machine current status (ESM)
Diagnostic Data	information about possible error causes (see ETG1000.6 par. 5.6.4.3.2–5).

Refer also to the "4.6 Diagnostic LEDs" section on page 27.

### 7.2.8 AL Status Error Codes

If the state transition requested by the Master through the "AL Control Register" is unsuccessful, the Slave sets to 1 the "Error Indicator Bit" in "AL Status Register" and writes the cause of the error in "AL Status Code Register". Values and descriptions of "AL Status Code" are available in ETG1000.6 "EtherCAT Specification – Part 6. Application Layer protocol specification", par.5.3.2 Table 11.

### 7.3 File Over EtherCAT (FoE)

Lika encoders are devices that allow the firmware update using the protocol "File over EtherCAT (FoE)".

For any specification on FoE protocol, please refer to "ETG.1000 EtherCAT Specification" document available at the address [www.ethercat.org](http://www.ethercat.org).

Please refer also to the "6.7 Firmware upgrade" section on page 47.

## 8 – Default parameters list

Default values are expressed in hexadecimal notation.

Parameters list	Default values		
<b>1000-00 Device type</b>	0001 0196 = singleturn encoder 0002 0196 = multiturn encoder		
<b>1008-00 Device Name</b>	4553353878 = "ES58x" = singleturn encoder E- series 454D353878 = "EM58x" = multiturn encoder E- series 4853353878 = "HS58x" = singleturn encoder H- series 484D353878 = "HM58x" = multiturn encoder H- series		
<b>1009-00 Hardware version</b>	device dependent		
<b>100A-00 Software version</b>	device dependent		
<b>1018 Identity</b>			
<b>01 Vendor ID</b>	0000 012E		
<b>02 Product code</b>	0000 0001 = Lika EtherCAT ES58 & HS58, singleturn encoder 0000 0002 = Lika EtherCAT EM58 & HM58, multiturn encoder		
<b>03 Revision</b>	0001 0001 = Lika EtherCAT ES58 & EM58 series 0001 0000 = Lika EtherCAT HS58 & HM58 series		
<b>04 Serial number</b>	0000 0000		
<b>1A00-01 PDO mapping parameter</b>			
<b>01 Mapped Object 001</b>	6004 0020		
<b>6000-00 Operating parameters</b>	0000		
<b>Bit 0 Code sequence</b>	0 = disabled		
<b>Bit 2 Scaling function</b>	0 = CW		
<b>6001-00 Units per revolution</b>	0000 2000 (8,192) for EMxx 0040 0000 (262,144) for HSxx 0001 0000 (65,536) for HMxx		

<p><b>6002-00 Total Measuring Range</b></p>	<p>0800 0000 (134 217 728)  for EMxx  0040 0000 (262,144)  for HSxx  4000 0000 (1 073 741 824)  for HMxx</p>		
<p><b>6003-00 Preset</b></p>	<p>00000 0000</p>		

This page intentionally left blank

Document release	Release date	Description	HW	SW	XML file version
1.0	04.01.2011	First issue	H01	S01	V1
1.1	18.04.2011	General review	H01	S01	V1
1.2	16.09.2011	"7.3 File Over EtherCAT (FoE)" section added	H01 H01 H01 H01 H01	S02 S02 S02 S02 S03	V2 V2.1 V2.2 V2.3 V3
1.3	26.05.2014	"3 - Mechanical installation" section added, EEPROM and firmware upgrade instructions, complete review, Italian / English separate editions	H01 H01 H02	S04 S04 S05	V4 V5 V6
1.4	30.04.2019	General review	H02	S05	V6



This device is to be supplied by a Class 2 Circuit or Low-Voltage Limited Energy or Energy Source not exceeding 30 Vdc. Refer to the order code for supply voltage rate.

Ce dispositif doit être alimenté par un circuit de Classe 2 ou à très basse tension ou bien en appliquant une tension maxi de 30Vcc. Voir le code de commande pour la tension d'alimentation.



Dispose separately

**lika**

**Lika Electronic**

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

Tel. +39 0445 806600

Fax +39 0445 806699



info@lika.biz • www.lika.biz