

HENGSTLER

Technical Manual



ACURO® AC58 EtherCAT® Absolute Encoder

Part No. 2 565 651

Version 3.00

Mod. No. 3 200716TK

HENGSTLER GmbH
Uhlandstr. 49
78554 Aldingen / Germany
Tel. +49 (0) 7424-89 0
Fax +49 (0) 7424-89 500
E-Mail: info@hengstler.com
www.hengstler.com

© by HENGSTLER

Hengstler GmbH has created the text and diagrams contained in this document with care. However, we cannot accept responsibility for any errors or omissions. Notification regarding any errors and suggestions for improvement are welcome. We reserve the right to make technical and/or other changes at any time in the interest of continual product development or for other reasons.

All information contained in this manual is provided without regard to any possible patent protection.

All rights reserved. Reproduction, translation and/or distribution of this document, or extracts thereof, are permitted only by express authorization from Hengstler GmbH. ACURO® is a registered trademark of Hengstler GmbH. The Hengstler name and the Hengstler logo are registered trademarks of Hengstler GmbH. Beckhoff® and TwinCAT® are registered and licensed trademarks of Beckhoff Automation GmbH, Germany. CANopen is a registered trademark of CAN in Automation e.V., Nuremberg. "Windows" and "Microsoft" are registered trademarks of Microsoft Corporation. Other brand and product names are trademarks or registered trademarks of their respective companies.

HENGSTLER GmbH
Uhlandstr. 49
78554 Aldingen / Germany
Tel. +49 (0) 7424-89 0
Fax +49 (0) 7424-89 500
E-Mail: info@hengstler.com
www.hengstler.com

General Manager: Jochen Feiler, Commercial Register Tuttlingen HRB 604 Sp, ID-Nr. DE 811194298

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Document History

Revision	Date	Initials	Status	Description
3.00	2016 May 05	TK	Closed	Series Release, Firmware R3-V1.2.0.1

Table of Contents

List of Tables.....	7
List of Figures.....	7
1 Introduction	8
1.1 About this Document.....	8
1.2 Terms, Abbreviations and Definitions.....	8
1.3 Referenced Documents	9
2 Safety, Installation and Operation Hints	10
2.1 Authorized personnel	10
2.2 Risk of injury due to rotating parts.....	10
2.3 Risk of injury due to safety-critical applications	10
2.4 Risk of damage due to static electricity	10
2.5 Risk of damage due to mechanical overload.....	10
2.6 Risk of damage due to mechanical shock.....	10
2.7 Risk of damage due to overloading	10
2.8 Over-voltage	11
2.9 Dragline mounting prohibited.....	11
2.10 Component	11
2.11 CE Mark	11
3 General Information	12
3.1 Introduction	12
3.2 Fields of Application	12
4 Installation and Operating	13
4.1 Installation Overview.....	13
4.2 Initial Setup.....	13
4.3 LED Interpretation	13
5 EtherCAT®	16
5.1 EtherCAT® Communication Model	16
5.2 Communication Profile (CiA 301)	17
5.3 Encoder Device Profile (CiA 406)	18
5.4 Basic Introduction to EtherCAT®	18
5.4.1 Physical Layer.....	19
5.4.2 Data Link Layer.....	19
5.4.3 Application Layer	20

5.4.4	Synchronization and Distributed Clocks.....	21
5.5	EtherCAT® Details	23
5.5.1	FoE/File Transfer	23
5.5.2	Selection of synchronization mode.....	23
5.5.3	DC Timing with TwinCAT.....	24
5.5.4	Configuration of SyncManager 3 via object dictionary.....	25
5.5.5	Data Transmission	25
6	Object Dictionary	30
6.1	ACURO AC58 EtherCAT® Object Dictionary	32
6.2	Object Dictionary Measured Value Processing	55
6.2.1	Position Scaling.....	55
6.2.2	Preset Function	56
6.2.3	Residual Value Calculation	56
6.2.4	Feature Interactions.....	57
6.2.5	Speed Calculation	58
7	Commissioning with TwinCAT®	59
7.1	Introduction to TwinCAT®	59
7.2	Installation of TwinCAT®	59
7.3	Configuring TwinCAT® to use Sync Manager 3.....	59
7.3.1	Starting TwinCAT®	59
7.3.2	Installation of ACURO EtherCAT® within TwinCAT®	60
7.3.3	Selection of TwinCAT® Ethernet Driver	61
7.3.4	Copying ESI File into TwinCAT® Installation.....	62
7.3.5	Scanning for Devices.....	62
7.3.6	Load Process Data from Device	64
7.3.7	Select Synchronization Mode	65
7.3.8	Adjust Base Time.....	66
7.3.9	Further Steps.....	67
7.4	Configuring TwinCAT® for Distributed Clocks.....	68
7.5	Troubleshooting	70
7.5.1	Checking Data Exchange with TwinCAT®	70
7.5.2	Common Error Messages	70
8	Firmware Update	71
8.1	General Information	71
8.2	Update Procedure using TwinCAT®	71
8.3	Common Error Messages during Update Procedure.....	73

9	Network Performance	74
10	Error Handling and Diagnosis	76
10.1	CoE Emergency Messages	76
10.2	ACURO Specific AL Status Codes.....	78
11	Technical Data	79
11.1	Bus/Power Connections	79
11.2	EtherCAT®	80
12	Appendix	81
12.1	EtherCAT® commands	81
12.2	CoE Object 2002:1	82

List of Tables

Tab. 1: Terms, Abbreviations and Definitions	9
Tab. 2: Communication LEDs	14
Tab. 3: Power LED	15
Tab. 4: Meaning of Elements of EtherCAT Datagram	20
Tab. 5: Adjustments in the object dictionary for synchronization using SyncManager 3	25
Tab. 6: Types of PDO messages	26
Tab. 7: Possible Values of Error Class	27
Tab. 8: List of SDO Abort Codes	29
Tab. 9: Organization of the entire object dictionary	31
Tab. 10: Object Dictionary	55
Tab. 11: Feature Interactions	57
Tab. 12: Overview of the minimum Cycle Times for typical Configurations	75
Tab. 13: CoE emergency messages	77
Tab. 14: ACURO Specific AL Status Codes	78
Tab. 15: Connector Pinouts	79
Tab. 16: EtherCAT Technical Data	80
Tab. 17: EtherCAT Command Codes	81
Tab. 18: Contents of mailbox header	82
Tab. 19: Meaning of mailbox type	82
Tab. 20: Contents of Mailbox Header	83
Tab. 21: Meaning of mailbox type	83
Tab. 22: Contents of CoE frame	84

List of Figures

Fig. 1: ACURO AC58 EtherCAT Connector/LED Locations	13
Fig. 2: EtherCAT in the ISO/OSI Reference Model	16
Fig. 3: CoE Structure	17
Fig. 4: "On the fly" principle of EtherCAT process data extraction and insertion	18
Fig. 5: Structure of EtherCAT datagram	19
Fig. 6: State Diagram of the EtherCAT State Machine	21
Fig. 7: Time Diagram Synchronization with Sync Manager	22
Fig. 8: EtherCAT network with Distributed Clock function	23
Fig. 9: Selection of Synchronization Mode	24
Fig. 10: DC timing with TwinCAT	24
Fig. 11: Errors due to non-integer multiples of Scaled Range to Physical Range	56
Fig. 12: Leap in speed values	58
Fig. 13: TwinCAT menu entry TwinCAT > Show Real Time Ethernet Compatible Devices	60
Fig. 14: Installation of TwinCAT RT-Ethernet Adapters – Compatible Devices	60
Fig. 15: Installation of TwinCAT RT-Ethernet Adapters – Installed and ready-to-use Devices	61
Fig. 16: Selection of Synchronization Mode	68
Fig. 17: Configuration for Activation of SYNC0 Signal within the Advanced Settings	69
Fig. 18: ACURO AC58 EtherCAT LED/Connector Layout	79
Fig. 19: Mailbox header	82
Fig. 20: CoE command header	83
Fig. 21: CoE frame	84

1 Introduction

1.1 About this Document

This manual describes the ACURO AC58 EtherCAT® family of absolute shaft encoders, including specifications, dimensions, software, commissioning and parameterization.

1.2 Terms, Abbreviations and Definitions

Term	Description
CiA	CAN in Automation
CoE	CANopen over EtherCAT
DC	Distributed Clocks
DS	(CiA) Draft Standard
ECS	EtherCAT Slave
EoE	Ethernet over EtherCAT
ENI	EtherCAT Network Information
ESI	EtherCAT Slave Information
ESM	EtherCAT State Machine
ETG	EtherCAT Technology Group
FoE	File Access over EtherCAT
FSoE	Failsafe over EtherCAT
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IRQ	Interrupt Request
ISO	International Organization for Standardization
LSB	Least Significant Byte
LVDS	Low Voltage Differential Signals
MSB	Most Significant Byte
OSI	Open Systems Interconnection
PDO	Process Data Object
SDO	Service Data Object
SM0...3	Sync Manager 0...3
SoE	Servo drive over EtherCAT
TFTP	Trivial File Transfer Protocol
TwinCAT	The Windows Control and Automation Technology
VoE	Vendor specific profile over EtherCAT

Term	Description
XAE	TwinCAT Engineering Component
XAR	TwinCAT Runtime Component

Tab. 1: Terms, Abbreviations and Definitions

All variables, parameters, and data used in this manual employ the LSB/MSB (“Intel”) data format. All IP addresses in this document use host byte order.

1.3 Referenced Documents

This manual refers to the following documents:

- [1] IEC 61158 Part 2-6 Type 12 documents (also available for EtherCAT® Technology Group members as specification documents ETG.1000).
- [2] IEEE 802.3-2012, available from <http://standards.ieee.org/about/get/802/802.3.html>.
- [3] CANopen application layer and communication profile, CiA 301, V.4.2.0, CAN in Automation e.V., <http://www.can-cia.org>, 2011.
- [4] Device profile for encoders, CiA 406, V 3.2.0, CAN in Automation e.V., <http://www.can-cia.org>, 2006.
- [5] EtherCAT® Protocol Enhancements, ETG.1020 S (R) V1.0.0, EtherCAT® Technology Group, <http://www.ethercat.org/>, 2011.
- [6] EtherCAT® Network Information Specification ETG.2100 S (R) V1.0.0, EtherCAT® Technology Group, <http://www.ethercat.org/>, 2009.
- [7] The TFTP Protocol (Revision 2), RFC1350, Internet Engineering Task Force, <http://tools.ietf.org/html/rfc1350>.
- [8] TwinCAT 3 documentation, part of Beckhoff Information System, see <http://infosys.beckhoff.com/content/1033/tcinfosys3/html/startpage.htm?id=12332> (English).
- [9] Hengstler AC58 EtherCAT® Installation Instructions, Art. No. 2 565 653, available from Hengstler GmbH.
- [10] Hengstler AC58 EtherCAT Technical Manual, German language, Art. No. 2 565 652.

2 Safety, Installation and Operation Hints

Care must be taken when installing and using this product. Please refer to Hengstler Installation Instructions Article No. 2 565 653 for detailed information on safety and installation. While much of this information is repeated below, the full, multilingual document should be used to ensure correct and safe installation.

2.1 Authorized personnel

This encoder should only be installed or uninstalled by a qualified technician, as the unit contains sensitive electronic circuitry.

2.2 Risk of injury due to rotating parts

Hair, jewelry or articles of clothing may become caught in rotating shafts or other parts! Prior to commencing any work, disconnect all power supplies and ensure that the working environment is Safe!

2.3 Risk of injury due to safety-critical applications

When the AC58 EtherCAT encoder is used in safety-critical applications which could threat life or physical condition, it is required that position related data supplied by the encoder is checked on plausibility before being used within applications.

2.4 Risk of damage due to static electricity

The CMOS modules contained in this encoder are very sensitive to high voltages, such as those that can arise due to friction in clothing or shoes. *Do not touch connector contacts or electronic components!*

2.5 Risk of damage due to mechanical overload

Rigid mounting will cause constraining forces which will permanently overload and damage the bearings. *Never restrict the freedom of movement of the encoder! Use only the enclosed sheet metal springs or a suitable coupling when mounting the unit!*

2.6 Risk of damage due to mechanical shock

Violent shocks, e.g. hammer blows, can lead to damage of the optical sensing system and the ball bearings. *Never use force! Assembly is simple provided that correct procedures are followed.*

2.7 Risk of damage due to overloading

The unit may only be operated within the electrical, mechanical and other limits specified in the technical data.

2.8 Over-voltage

Over-voltage at the connecting terminals *must be limited to over voltage-class-II values (SELV)*.

2.9 Dragline mounting prohibited

The connecting cable *is not rated for dragline mounting*, only for fixed mounting of the encoder.

2.10 Component

This encoder is a component intended for mounting to other equipment (motor, machine, etc.). It is not intended for direct sale to the end customer.

2.11 CE Mark

Manufacturers integrating this encoder into their products are responsible for compliance with CE guidelines, and for proper use of the CE mark.

3 General Information

3.1 Introduction

We all know that no two industrial applications are alike. That's why the Hengstler AC58 absolute encoder is available with an incredible array of different options and features, including 22-bit single-turn resolution. But what if our standard variants don't meet your needs? Then Hengstler is able to offer custom versions to fulfill your requirements! Extended temperature range, greatly enhanced shock and vibration ratings, and custom shaft sizes and shapes are just a few of the features we've provided our customers recently. This flexibility makes the Hengstler AC58 one of the most versatile encoders on the market, in addition to being one of the most robust.

Now the AC58 product line has been expanded by the addition of the popular EtherCAT® interface. Use of systems employing this open, high performance Ethernet-based system continues to grow rapidly. By offering the AC58 with EtherCAT®, users can now integrate one of the best absolute encoders on the market with virtually any system using EtherCAT®. This simplifies the design process and ensures reliable communications.

3.2 Fields of Application

The field of application for this encoder is industrial processes and controls. Examples of applications include packaging machines, injection molding machines, wood processing machine, assembly and handling technology, conveyor technology, printing and paper machines.

4 Installation and Operating

4.1 Installation Overview

Initial installation of the Hengstler ACURO AC58 EtherCAT® absolute encoder should be accomplished by following Hengstler Installation Instructions, Article No. 2 565 653, available for download from the Hengstler GmbH web site at www.hengstler.com. This document provides valuable information regarding safety, encoder mounting, wiring, etc. Please be sure to keep operation of the encoder within the electrical and mechanical limits specified by Hengstler GmbH, as shown in this and other Hengstler documentation.

Electrical connections and M12 connector pinouts for the encoder can be found in the section “Bus/Power Connections” in section 11.1. CAT 5, 6 or 7 cables can be used. The maximum distance between nodes is limited to 100 meters. Note, however, that IP67 and higher protection ratings are contingent upon the use of an appropriately rated mating connector/cable assembly which has been properly installed by the user.

4.2 Initial Setup

Your Hengstler ACURO AC58 EtherCAT® absolute encoder comes from the factory ready to install with the default EtherCAT parameters and settings. (Please see the section “ACURO AC58 EtherCAT Object Dictionary” for the default values.) If your encoder was ordered with EtherCAT parameters and/or settings other than the defaults, these will have been loaded at the factory, and will be detailed in a document accompanying the encoder itself. Should you wish to change these parameters and/or settings, please refer to the EtherCAT portion (section 5) of this manual. Please note, however, that certain reset commands will reset any special parameters and/or settings that were changed at the factory back to their default values. Care must be taken to avoid accidentally changing desired settings or information while programming.

4.3 LED Interpretation

The tables “Communication LEDs” and “Power LED” explain the states of the Communication LEDs (RUN, ERR and Link/Activity) and Power LED of the ACURO EtherCAT®. The location of the LEDs is as shown in Figure 1.

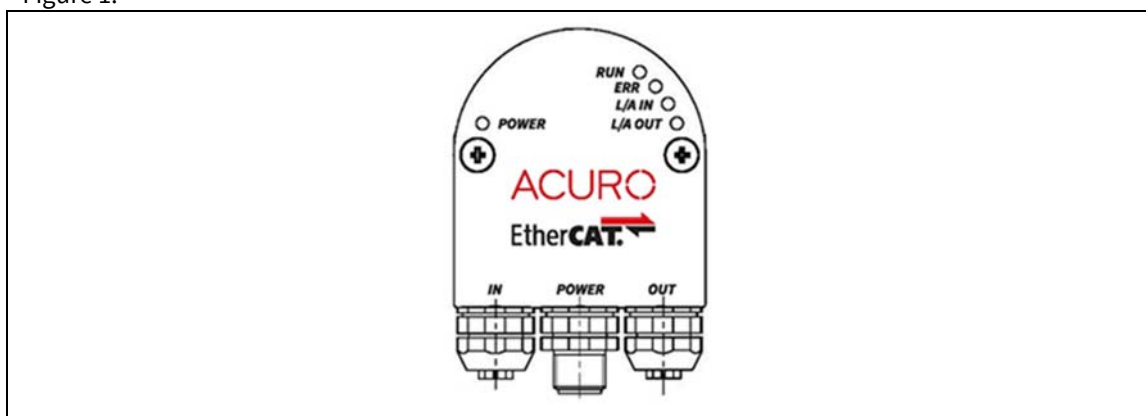
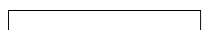
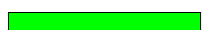
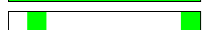
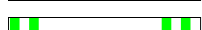
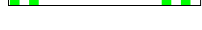



Fig. 1: ACURO AC58 EtherCAT Connector/LED Locations

The LED Indicator States are defined as follows. (Green indicates LED lit, while indicates LED off.)

-  **Off:** LED is not illuminated at all.
-  **On:** LED is illuminated continually.
-  **Single Flash:** LED flashes once, then repeats after a long delay.
-  **Double Flash:** LED flashes twice in quick succession, then repeats after a long delay.
-  **Blinking:** LED flashes on and off continually but slowly with even spacing.
-  **Flickering:** LED flashed on and off continually but rapidly with even spacing.

Name	Indicator State	Color	Meaning
RUN	Off	green	Device is in state INIT
	Blinking		Device is in state PRE-OPERATIONAL
	Single Flash		Device is in state SAFE-OPERATIONAL
	On		Device is in state OPERATIONAL
	Flickering		Device is in state BOOTSTRAP. Firmware download operation in progress
ERR	Double Flash	red	Process Data Watchdog Timeout/ EtherCAT Watchdog Timeout: An application watchdog timeout has occurred Example: Sync Manager Watchdog timeout
	Single Flash		Local Error: Slave device application has changed the EtherCAT state autonomously, due to local error. Example: Device changes its EtherCAT state from Op to SafeOpError due to a synchronization error.
	Blinking		General Configuration Error Example: State change commanded by master is impossible due to register or object settings.
	Off		No Error: The EtherCAT communication of the device is in working condition
LINK/ACTIVITY (L/A IN L/A OUT)	On	green	Port open (link established), no activity
	Flickering		Port open (link established), activity (device sends/receives Ethernet frames)
	Off		Port closed (no link established)

Tab. 2: Communication LEDs

The “Power” LED indicates power supply state and device-specific errors.

Name	Status LED	Meaning
Power	Off	Power supply insufficient
	Green On	Power supply sufficient
	Red Blinking	Sensor Failure. Possible causes are: <ul style="list-style-type: none">▶ Sensor not present▶ Mismatch in sensor type/resolution between sensor and bus hood▶ ERROR bit in BiSS frame is set to 1

Tab. 3: Power LED

5 EtherCAT®

5.1 EtherCAT® Communication Model

The EtherCAT Communication Model is based on the ISO/OSI Communication Model which employs seven (7) Layers.

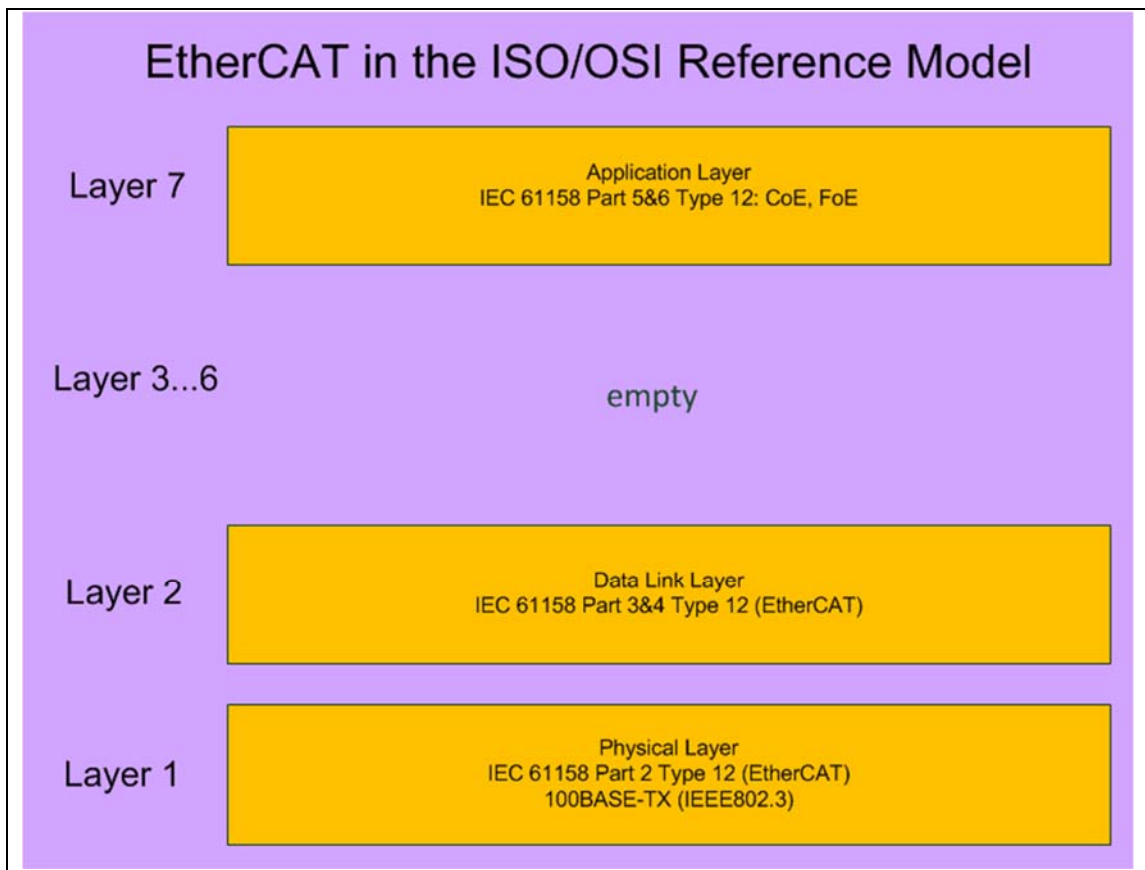


Fig. 2: EtherCAT in the ISO/OSI Reference Model

However, in EtherCAT communications, only three layers are used:

- ▶ Layer 1 (Physical Layer): 100BASE-TX according to IEC 61158 Part 2 Type 12/ETG 1000 Part 2 Type 12 (reference [1]) and IEEE802.3 (reference [2]).
- ▶ Layer 2 (Data Link Layer): according to IEC 61158 Part 3&4 Type 12/ETG 1000 Part 3&4. Type 12 (reference [1]).
- ▶ Layer 7 (Application Layer): according to IEC 61158 Part 5&6 Type 12/ETG 1000 Part 5&6 Type 12 with CoE (CANopen over EtherCAT) communication profile similar to CiA 301 (CANopen Communication Profile) (reference [3]).

5.2 Communication Profile (CiA 301)

The ACURO AC58 EtherCAT® meets the requirements defined in the communication profile (CiA 301, reference [3]) and in the device profile for encoders (CiA 406, reference [4]).

NOTE! The mentioned documents CiA 301 (reference [3]) and CiA 406 (reference [4]) originate from CAN in Automation e.V. These are not official ETG documents.

This is done by the mailbox protocol CoE (CANopen over Ethernet) which is a part of the EtherCAT standard.

EtherCAT allows for

- ▶ auto-configuration of the network,
- ▶ comfortable access to all device parameters.
- ▶ synchronization of the devices,
- ▶ cyclic and event-controlled process data processing,
- ▶ simultaneous data input and output.

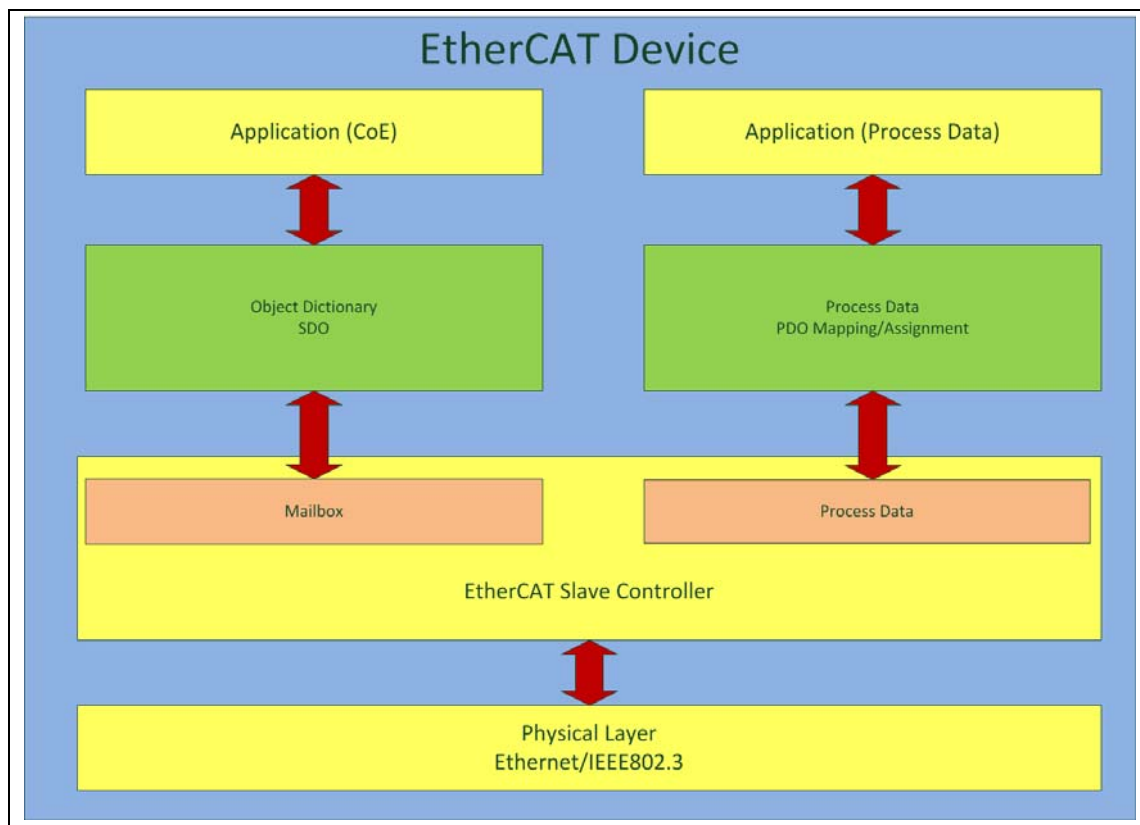


Fig. 3: CoE Structure

EtherCAT uses communication objects with different features:

- ▶ Process Data Objects (PDO) for real-time data
- ▶ Service Data Objects (SDO) for the transfer of parameters and programs

All device parameters are stored in an object dictionary. The object dictionary contains the description, data type and structure of the parameters as well as their addresses (index and sub-index). The directory consists of three parts: communication profile parameters, device profile parameters and manufacturer specific parameters (section 6, "Object Dictionary").

5.3 Encoder Device Profile (CiA 406)

NOTE! The Encoder Device Profile CiA 406 (reference [4]) has not been officially approved by the ETG!

This profile describes a binding, but manufacturer-independent definition of the interface for encoders. The profile not only defines which CoE functions are to be used, but also how they are to be used. This standard permits an open and manufacturer-independent bus system. Furthermore, an addressable area is defined in the profile, to which, depending on the manufacturer, different functions can be assigned.

5.4 Basic Introduction to EtherCAT®

EtherCAT® is based on Ethernet according to the IEEE 802.3 standard.

The ETG has introduced the following analogy that might be helpful as starting point to understand EtherCAT's **"on the fly" principle** of process data extraction and insertion:

An EtherCAT slave handles the Ethernet frames passing through quite similar to a person watching a fast railway train which drives by. The Ethernet frames are like the entire train, they do not stop. Even when watching the Ethernet frames (or the train) through a small window, the entire frame (or train) will be observable. The sub-telegrams are similar to the wagons within the train. They may have variable length. Single bits (or persons) or entire groups may be "extracted" or "inserted".

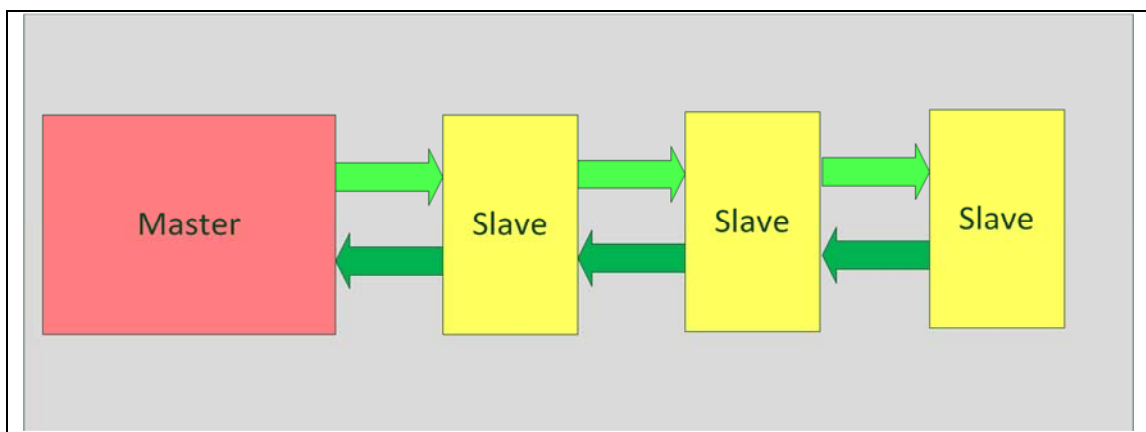


Fig. 4: "On the fly" principle of EtherCAT process data extraction and insertion

This means that each slave extracts the data designated to it "on the fly" while passing through and vice versa it inserts the input data. This causes a delay of only a few nanoseconds. When returning to the Master at the end of its way, the EtherCAT telegram is already fully processed.

EtherCAT uses a special kind of Ethernet frame (with Ethertype 0x88A4) denominated as EtherCAT Datagram, see "Structure of EtherCAT datagram" below.

5.4.1 Physical Layer

EtherCAT® supports three variants of the IEEE 802.3 standard.:

- ▶ 100BASE-TX
- ▶ 100BASE-FX
- ▶ E-Bus/LVDS

100BASE-TX uses shielded twisted-pair copper cables with two pairs of wires. Cables of categories CAT 5, 6 or 7 can be used. Available connectors include RJ45 (mainly in non-industrial applications) and M12 (especially for industrial applications, as the connectors can have protection class IP67). The distance between two nodes is limited to 100 meters for 100BASE-TX.

100BASE-FX uses fiber-optics of various types. The distance between two nodes is limited to 2 km for 100BASE-FX.

E-Bus/LVDS is only used for modular devices.

NOTE! The ACURO AC58 EtherCAT uses only 100BASE-TX with M12 connectors.

5.4.1.1 Topology

EtherCAT® supports various topologies, such as:

- ▶ Line
- ▶ Star
- ▶ Tree structure
- ▶ Daisy chain (with or without branches)
- ▶ Cable redundancy

The number of devices within an EtherCAT network is limited to 65535.

5.4.2 Data Link Layer

5.4.2.1 Frame Structure of EtherCAT Datagram

The following Fig. 5, “Structure of EtherCAT datagram, explains the structure of the EtherCAT datagram:

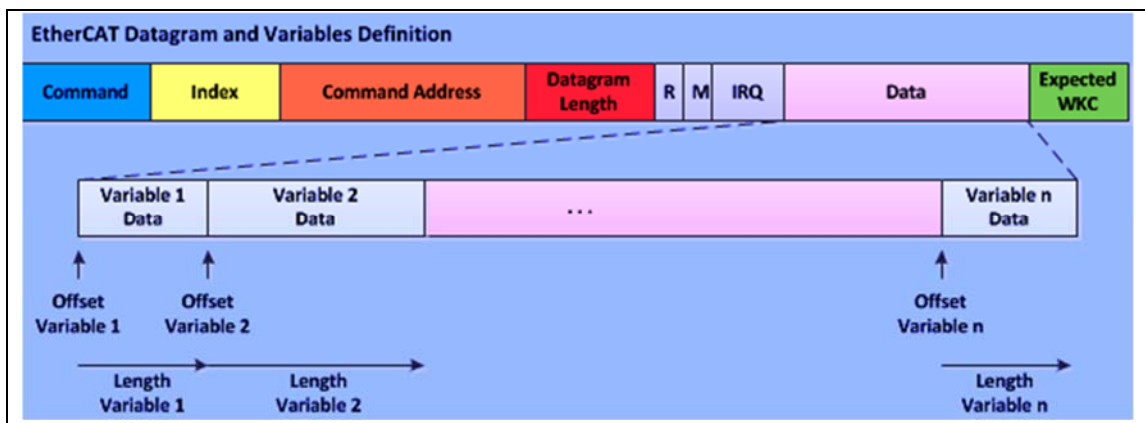


Fig. 5: Structure of EtherCAT datagram

The elements of the EtherCAT datagram have the following meaning:

Datagram-Element	Description
Command	This element of the EtherCAT Datagram contains the information regarding which slave should access the Process Data Unit (PDU).
Index	This is a number which is increased by 1 for each PDU in order to identify the PDU on return.
Command Address	This element of the EtherCAT Datagram contains the address of the data the command will affect. There are different addressing modes, see “EtherCAT® commands” in the appendix.
Datagram Length	This is a 16 bit large element of the EtherCAT Datagram of which 11 bits specify the length of the data field in bytes (limited to a maximum size of 1468 bytes).
Flags	Various flags (for instance R, M)
IRQ	Used internally
Data Field	The Data Field contains the data to be transferred with the EtherCAT Datagram. The length of the Data Field is limited to a maximum size of 1486 bytes.
Expected Working Counter	This element of the EtherCAT Datagram contains the expected value for the working counter on return. The working counter is increased by 1 for every read, by 2 for every write and by 3 for every read/write access. So the EtherCAT Master can compare the actual value of the working counter with the expected one and therefore detect errors easily.

Tab. 4: Meaning of Elements of EtherCAT Datagram

EtherCAT® Addressing

The EtherCAT standard (document ETG 1020, reference [5]) defines the following methods of address assignment:

- ▶ Automatically by EtherCAT master (topology-based). This is always possible.
- ▶ Assignment of “Configured Second Station Alias” by EtherCAT master.

The necessary interfaces and mechanisms for these methods are provided.

5.4.3 Application Layer

5.4.3.1 The EtherCAT® State Machine (ESM)

The states and state changes of the slave application can be described by the EtherCAT State Machine (ESM).

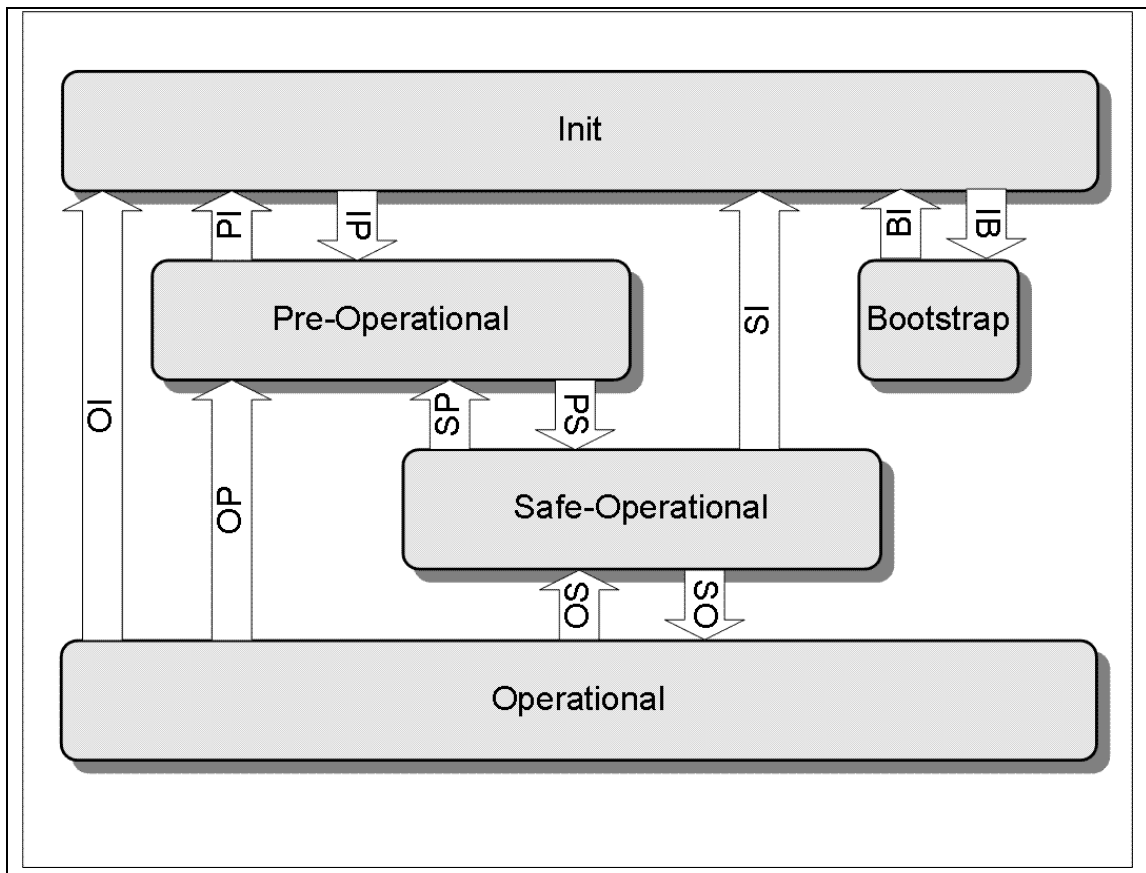


Fig. 6: State Diagram of the EtherCAT State Machine

The ESM implements the following five states:

- ▶ **INIT:** The EtherCAT Slave is initialized in this state. No real process data exchange happens. Previously stored values are loaded.
- ▶ **PRE-OPERATIONAL:** Initialization of the EtherCAT Slave continues. No real process data exchange happens. The master and the slave communicate acyclically via mailbox to set parameters; this means parameterization via SDO is already possible.
- ▶ **SAFE-OPERATIONAL:** In this state, the EtherCAT Slave can process input data. However, the output data are set to a “safe” state. Input data can be written in this state to the default input Sync Manager (usually SM3). However, the outputs are in “safe” state. This means, the EtherCAT master is able to read position values from the ACURO AC58 encoder.
- ▶ **OPERATIONAL:** In this state, the EtherCAT Slave is fully operational. This means, the EtherCAT master is able to read position values from the ACURO encoder in real-time.
- ▶ **BOOTSTRAP:** This is a special state for the boot-up process.

For a more precise description, see the EtherCAT specification (reference [1]).

5.4.4 Synchronization and Distributed Clocks

Synchronization can either be accomplished using a SyncManager or using the most accurate available synchronization mechanism, namely the distributed clocks (DC) technology.

NOTE! ACURO AC58 EtherCAT supplies the synchronization signal SYNC0.

5.4.4.1 Synchronization with Synch Manager

The default synchronization mode of the ACURO EtherCAT® slave allows synchronization with the SyncManager 3 (SM3).

The following figure shows an event sampling diagram for this synchronization mode:

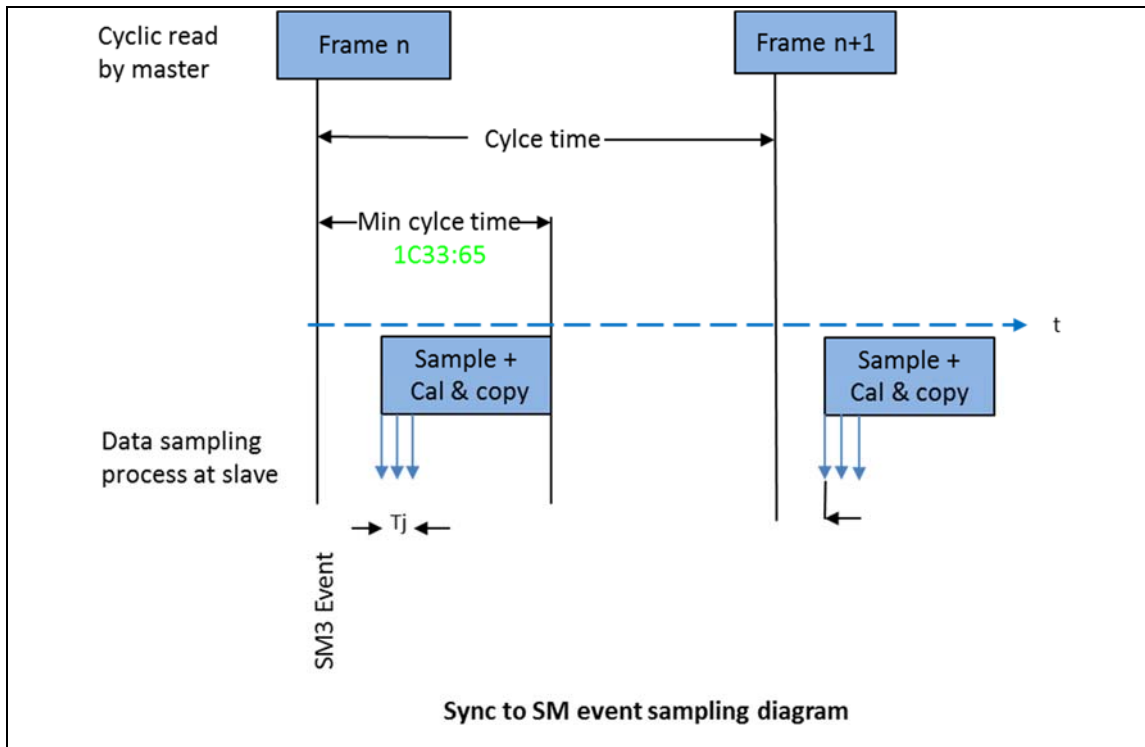


Fig. 7: Time Diagram Synchronization with Sync Manager

5.4.4.2 Distributed Clocks

The devices in an EtherCAT® network can be synchronized by means of the distributed clock (DC) mechanism. Each EtherCAT Slave with DC support has its own local clock. After power on, the local clock of every slave is not synchronized to the global reference clock. In order to synchronize the devices, one of the slaves will be usually selected and used as a reference clock. This reference clock provides the system time for other devices. All other slaves and the master should synchronize themselves to the reference clock.

The bus is controlled by the EtherCAT Master sending EtherCAT frames according to the bus cycle. If the EtherCAT Master supports the distributed clock function, then it can use a special EtherCAT datagram to distribute the system time in the network. The EtherCAT Master sends this datagram cyclically allowing the selected reference clock slave to insert the reference system time and all other slaves to read it.

It is convenient to select the first slave to be a reference clock. Because of the ring topology of the EtherCAT network the datagram will come back to the master after passing each slave. In this way, the latency and jitter of the clock distribution is minimized.

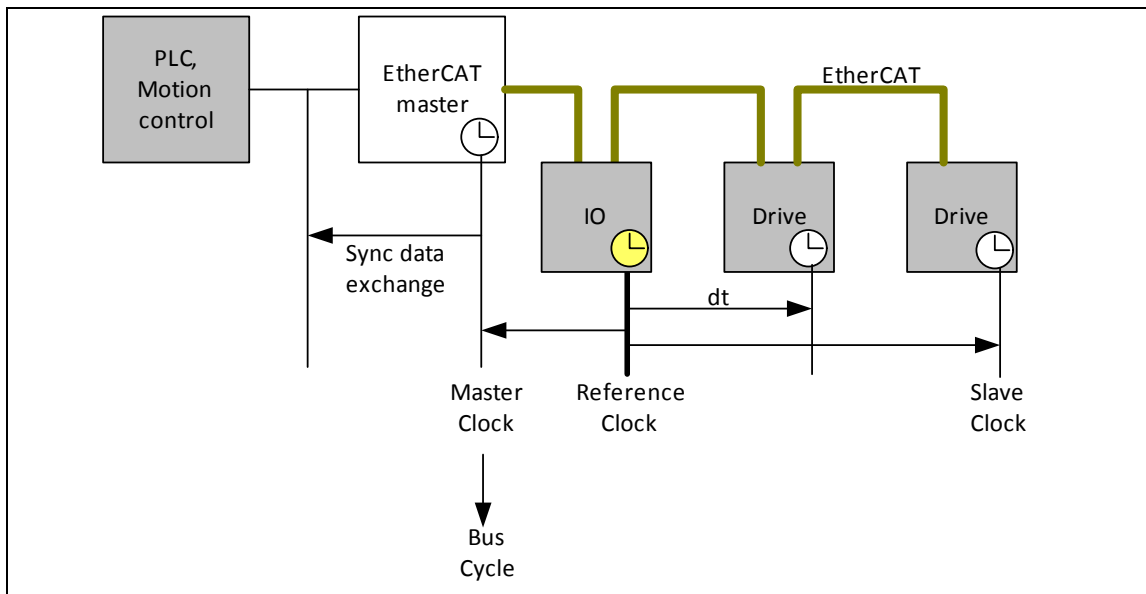


Fig. 8: EtherCAT network with Distributed Clock function.

Fig. 8 (“EtherCAT network with Distributed Clock function.”) shows a typical EtherCAT network with distributed clocks function. The first EtherCAT slave following after the EtherCAT Master is selected to be the reference clock for this EtherCAT network. The frame passing each slave and a transmission line between slaves is delayed for some small time “dt”, which must also be considered for clock synchronization. The EtherCAT Master sends at start-up a Broadcast-Read datagram to a special address of each slave. It commands the slave to latch and save the local timestamp of frame receive time point for each direction. The master can read out the saved timestamps of each slave to calculate the delays in the whole network. This measurement will be repeated many times at start-up for better accuracy.

In order to synchronize itself with the reference clock, each EtherCAT DC Slave compares the own local time with the reference clock time (system time) and adjusts the local clock properly.

5.5 EtherCAT® Details

5.5.1 FoE/File Transfer

Another mailbox protocol defined by the EtherCAT standard is FoE, which stands for File Transfer over EtherCAT. This protocol is intended to easily accomplish file transfers from and to the EtherCAT device. FoE uses a protocol similar to TFTP (Trivial File Transfer Protocol as described in RFC1350, see reference [7]) for file transfer from and to an EtherCAT device.

The EtherCAT protocol stack running within ACURO EtherCAT uses FoE for firmware update. For more information see chapter 8, “Firmware Update”.

5.5.2 Selection of synchronization mode

Selection of SM synchronous mode with SyncManager 3 via TwinCAT

In order to choose this synchronization mode:

- ▶ Select the ACURO in the solution explorer tree.
- ▶ On the DC register card, select the operation mode SM Synchronous:

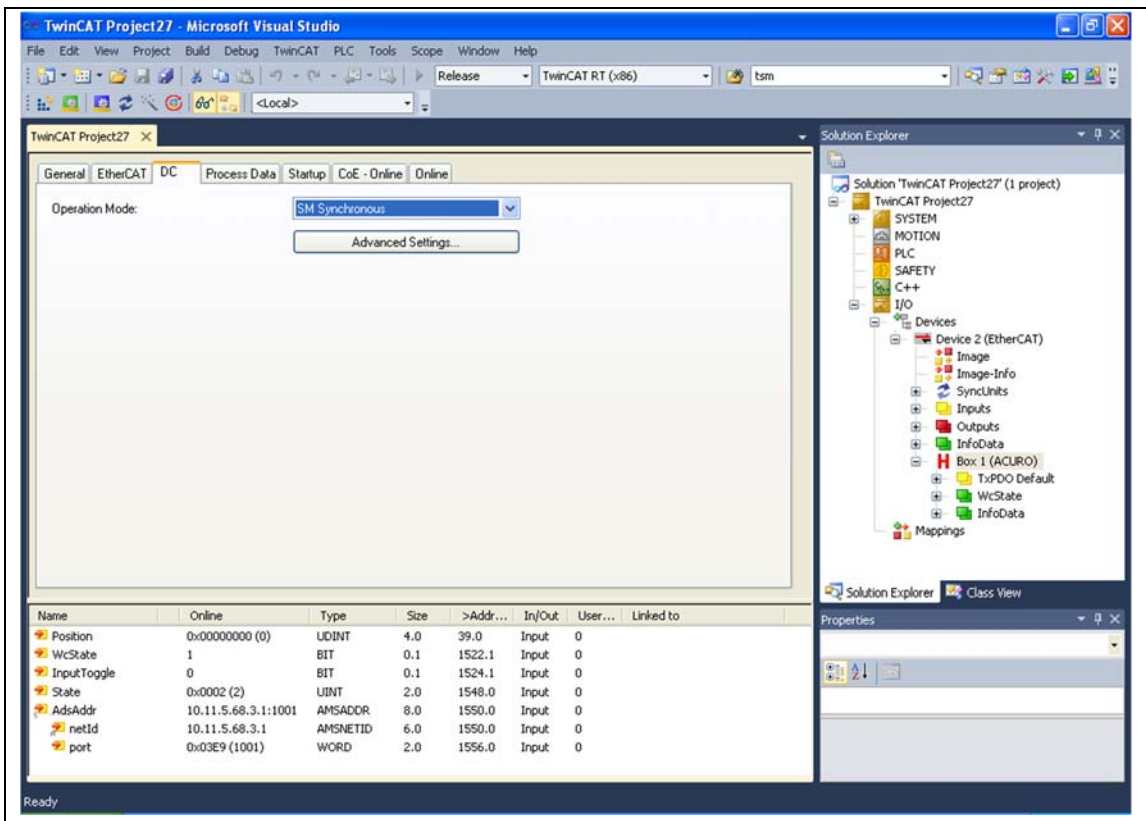


Fig. 9: Selection of Synchronization Mode

5.5.3 DC Timing with TwinCAT

Fig. 10 below shows the timing of the Distributed Clocks with TwinCAT.

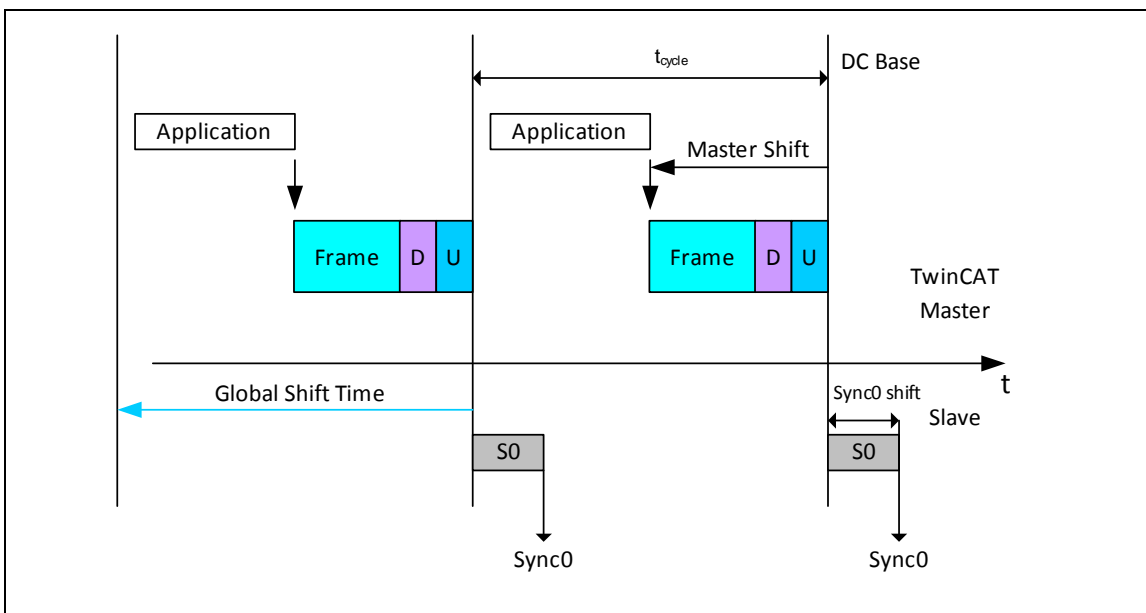


Fig. 10: DC timing with TwinCAT

- ▶ Application: application calculation and IO data exchange
- ▶ Frame: send time of the frame (80 ns / Byte)
- ▶ D: bus delay (runtime via transmission line (~5 ns/m) and slaves ~1 µs/slave)
- ▶ U: user defined offset (safety margin)
- ▶ S0: user defined offset for individual slave

If TwinCAT is used as EtherCAT Master, it calculates the time points to send the frame related to the DC base cycle. The offset is selected to ensure that the data is received by the slave before synchronization event at slave side. Additionally an individual offset can be selected by the user for each slave to define the specific SYNC0 offset (see Fig. 10). The shift value will be written to the slave using a proper InitCmd, which is configured in the EtherCAT Network description file (ENI, see reference [6]). Further, the keyword `<EtherCATConfig><Config><Slave><DC><ShiftTime>` will be defined too (see “Shift time of SYNC0 event in ns” in the ENI specification published by the ETG).

In order to choose the distributed clocks mode, select the option *DC Synchronous* in the selection list *Operation Mode* of the DC register card which is displayed when having the ACURO selected in the solution explorer tree. This is described in section 7.4, “Configuring TwinCAT® for Distributed Clocks”.

5.5.4 Configuration of SyncManager 3 via object dictionary

The following entries of the object dictionary are relevant:

Value	Index	Subindex	Description
Synchronization type	1C33h	01h	Synchronization type 1: Synchronized with AL event of this Sync Manager
Cycle time	1C33h	02h	The cycle time denotes the time between two events in ns.
Minimum cycle time	1C33h	05h	Minimum cycle time (in ns) that can be applied.
Index 1C33h, subindex 01h has to be set to 1 to for synchronization mode.			

Tab. 5: Adjustments in the object dictionary for synchronization using SyncManager 3

For more information concerning the dependence of the achievable cycle times from conditions and configuration state, see section 9, “Network Performance”.

5.5.5 Data Transmission

5.5.5.1 Transfer of Cyclic Process Data

The cyclic process data is organized in PDOs (process data objects). The module provides different PDOs with the position and different additional information. Using the PDO assignment mechanism (see “ACURO AC58 EtherCAT® Object Dictionary”, object 0x1C13), one of these PDOs can be selected and assigned to the input Sync Manager SM3.

NOTE! The contents of the PDOs are fixed, process data can be configured via selecting the PDO to be applied.

NOTE! Only one PDO can be assigned at a time.

The PDO Services TxPDO1 (0x1A00) to TxPDO4 (0x1A03) are available.

The PDO type is adjusted using sub-index 1 of the object 1C13h. Sub-index 0 of this object contains the number of assigned TxPDOs (0 or 1). The following types of PDO messages are defined:

PDO type	Value of index 0x1C13, sub-index 1	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
TxPDO1/ “Default”	0x1A00	Position value (4 bytes) ¹⁾				-	-	-	-
TxPDO2/ “Type 1”	0x1A01	Position value (4 bytes) ¹⁾				Flags ²⁾ (1 byte)	-	-	-
TxPDO3/ “Type 2”	0x1A02	Position value (4 bytes) ¹⁾				Speed ³⁾ (2 bytes)		Reserved for future use ⁴⁾ (2 bytes)	
TxPDO4/ “Type 3”	0x1A03	Position value (4 bytes) ¹⁾				System time at position latch ⁵⁾ (4 bytes)			
¹⁾ See object dictionary index 6004h ²⁾ Warning position flags, see object dictionary index 2004h ³⁾ See object dictionary index 2002h, sub-index 6 ⁴⁾ Ignore contents of these bytes ⁵⁾ See object dictionary index 2009h									

Tab. 6: Types of PDO messages

NOTE! Values are in little endian byte order, i.e. byte 0 = LSB, byte 3 = MSB etc.

5.5.5.2 Transfer of Service Data (Parameter Setting)

All device parameters are filed in the object dictionary (see section 6, “Object Dictionary”) under standardized addresses (index and sub-index) and can be written and read via SDOs (Service Data Objects). SDOs are exchanged between two stations (configuration master and shaft encoder) by means of the handshake procedure (request or confirmation).

There are two SDO services available:

- ▶ one transmit SDO for messages (e.g., confirmation that parameter has been set) from the shaft encoder to the master
- ▶ one receives SDO for messages (e.g., request to set a parameter) from the master to the shaft encoder

For more details, refer to the EtherCAT specification.

5.5.5.3 Error handling

Tab. 8: List of SDO Abort Codes” below lists the possibly occurring SDO abort codes.

These abort codes generally consist of the following elements:

- ▶ Error Class
- ▶ Error Code
- ▶ Additional Code

Error Class

The element Error Class (1 byte) generally classifies the type of error:

Class (hex)	Name	Description
1	vfd-state	Status error in virtual field device
2	application-reference	Error in application program
3	definition	Definition error
4	resource	Resource error
5	service	Error in service execution
6	access	Access error
7	od	Error in object dictionary
8	other	Other error

Tab. 7: Possible Values of Error Class

Error Code

The element Error Code (1 byte) specifies the error cause within an Error Class more precisely. (For Error Class = 8 [“Other error”], only Error Code = 0 [“Other error code”] is defined.) For more detail concerning the error, the Additional Code field is available.

Additional Code

The Additional Code (2 bytes) contains the detailed error description

The following table explains the relationship between SDO Abort Code and Error Class, Error Code and Additional Code, plus gives a short description of the error detail.

SDO Abort Code	Error-Class	Error-Code	Additional Code	Description
0x00000000	0	0	0	No error
0x05030000	5	3	0	Toggle bit not changed – Error in toggle bit at segmented transfer
0x05040000	5	4	0	SDO Protocol Timeout (at service execution)
0x05040001	5	4	1	Client/Server command specifier or ID not valid (i.e. unknown command specifier for SDO Service)
0x05040002	5	4	2	Reserved
0x05040003	5	4	3	Reserved
0x05040004	5	4	4	Reserved
0x05040005	5	4	5	Out of memory - Memory overflow occurred at SDO

SDO Abort Code	Error-Class	Error-Code	Additional Code	Description
				Service execution
0x06010000	6	1	0	Unsupported access to an object
0x06010001	6	1	1	Attempt to read a write –only object (Index may only be written but not read)
0x06010002	6	1	2	Attempt to write a read –only object (Index may only be read but not written- parameter lock active)
0x06020000	6	2	0	Object does not exist in the object dictionary – for instance, wrong index.
0x06040041	6	4	41	Object cannot be mapped to the PDO
0x06040042	6	4	42	The number and length of objects to be mapped would exceed the length of the PDO
0x06040043	6	4	43	General parameter incompatibility reason. (i.e. the data format of the parameter is incompatible for the index)
0x06040044	6	4	44	Reserved
0x06040047	6	4	47	General internal incompatibility in the device. (Device-internal error)
0x06060000	6	6	0	Access failed due to a hardware error. (Device-internal error)
0x06070010	6	7	10	Data type does not match, length of service parameter does not match (i.e. Parameter length error – data format for index has wrong size)
0x06070012	6	7	12	Data type does not match, length of service parameter too high – Data format too large for index
0x06070013	6	7	13	Data type does not match, length of service parameter too low – Data format too small for index
0x06090011	6	9	11	Sub-index does not exist.
0x06090030	6	9	30	Value range of parameter exceeded (only for write access) i.e. value is invalid
0x06090031	6	9	31	Value of parameter written too high.
0x06090032	6	9	32	Value of parameter written too low.
0x06090036	6	9	36	Maximum value is less than minimum value
0x08000000	8	0	0	General error
0x08000020	8	0	20	Data cannot be transferred or stored to the application.
0x08000021	8	0	21	Data cannot be transferred or stored to the application because of local control.
0x08000022	8	0	22	Data cannot be transferred or stored to the

SDO Abort Code	Error-Class	Error-Code	Additional Code	Description
				application because of the present device state.
0x08000023	8	0	23	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error).

Tab. 8: List of SDO Abort Codes

6 Object Dictionary

All features and parameters of an EtherCAT® device are stored in its object dictionary.

The entire data of the object dictionary is partly stored in the non-volatile memory of the shaft encoder to be protected against power failures and will be copied into the main memory (RAM) upon Power-on or Reset.

- ▶ If data in the object dictionary is changed, the change will only be made in the main memory and will be lost upon Power-off or Reset.
- ▶ If the data is to be saved permanently, the data must explicitly be stored in the non-volatile memory via the object 1010h (Store Parameters). *The existing data in the non-volatile memory will be overwritten by this procedure!*

Access to the object dictionary (read or write parameters) is gained via the SDO services which are described in section 5.5.5.2 "Transfer of Service Data (Parameter Setting)".

The object dictionary consists of several different groups:

- ▶ Features that are valid for all CoE devices (similar to CiA 301 plus a few extensions)
- ▶ Features that are valid for shaft encoders (to device profile CiA 406)
- ▶ Features that are manufacturer dependent

The address (index and sub-index), which points to any entry in the object dictionary, is, except for the features depending upon the manufacturer, standardized within the profiles. It is thus ensured that all devices always execute the functions described in the profile (standard and optional functions) within the same index. This is one of the requirements for an open system and the exchangeability of the devices.

The entries in the object dictionary are addressed by means of a 16 bit index. Each index entry can be further subdivided by a sub-index.

Structure of an object description

The description of the object dictionary entries is structured as follows:

Index (hex)	Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
-------------	-----------------	------	--------	-----------	--------	---------	---------------

Index:	16 bit address of the entry
Sub-index:	8 bit pointer on the subentry; is only used with complex data structures (e.g. Record, Array); if there is no sub-entry, then the Sub-index=0
Name:	Short description of the function
Object:	NULL: Entry without data DOMAIN: Larger variable data set, e.g. program code DEFTYPE: Definition of the data types, e.g. Boolean, float, unsigned16 etc. DEFSTRUCT: Definition of a record-entry, e.g. PDO Mapping Structure VAR: Single data value, e.g. Boolean, float, unsigned16, string etc. ARRAY: Field with similar data, e.g. unsigned16 data RECORD: Field with an arbitrary mixture of data
Data Type:	Data type, e.g. Boolean, float, unsigned16, integer etc.
Access:	Granted access rights to the object: rw read and write access ro read-only access const read-only access, value is a constant parameter
Meaning:	Explanation of described object
Default Value:	Default value of described object

Organization of the entire object dictionary:

Index (hex)	Objects
0000	not used
0001 - 001F	static data types
0020 - 003F	complex data types
0040 - 005F	manufacturer-specific data types
0060 - 0FFF	reserved
1000 - 1FFF	communication profile
2000 - 5FFF	manufacturer-specific profile
6000 - 9FFF	standardized device profile
A000 - FFFF	reserved

Tab. 9: Organization of the entire object dictionary

6.1 ACURO AC58 EtherCAT® Object Dictionary

The important elements, objects and values of the CoE object dictionary are defined below.

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1000:0	Device Type	VAR	UNSIGNED32	ro	Device type of the EtherCAT slave. The object dictionary is modeled according to the CANopen encoder profile CiA 406. However, the device profile CiA 406 is not currently approved by ETG. Therefore, object 0x1000 contains the value 0x00000000	0x00000000
1001:0	Error Register	VAR	UNSIGNED8	ro	Error register Bit 0: Generic error (0 = no error, 1 = error) Bit 1..7: reserved (0)	0x00
1003:0	Pre-defined Error Field	ARRAY	UNSIGNED8	rw	Error history: stores the latest 8 values of the emergency error code; (see section 10.1). The Additional information field contains the first two bytes of the manufacturer-specific error code of the corresponding emergency. These two bytes are byte-swapped in order to be easily human-readable in TwinCAT. Upon sending an emergency message, every new error is stored at sub-index 1; older entries are moved to the next higher sub-index. Sub-index 0 contains the number of stored entries. Writing 0x00 to sub-index 0 deletes the error history.	0x00
1003:1	Pre-defined Error Field	ARRAY	UNSIGNED32	ro	Error history: newest stored error	0x00
1003:2 ... 1003:8	Pre-defined Error Field	ARRAY	UNSIGNED32	ro	Error history: older errors	0x00

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1008:0	Manufacturer Device Name	VAR	VISIBLE_STRING	const	Device name of the EtherCAT slave as non-zero-terminated string in ASCII code	“ACURO”
1009:0	Manufacturer Hardware Version	VAR	VISIBLE_STRING	const	Hardware version of the EtherCAT slave as non-zero-terminated string in ASCII code.	0x03
100A:0	Manufacturer Software Version	VAR	VISIBLE_STRING	const	Software version of the EtherCAT slave firmware as non-zero-terminated string in ASCII code	(defined by the firmware)
1010:0	Store Parameters	ARRAY	UNSIGNED8	ro	Store parameters in non-volatile memory; sub-index 0 contains the highest sub-index that is supported.	0x01

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1010:1	Store All Parameters	ARRAY	UNSIGNED32	rw	<p>Store all user-changeable application parameters.</p> <p>This command stores the parameters specified by the following CoE objects: 0x2000:1..4, 0x2001, 0x2002:1..5, 0x2003:2..3, 0x2120:2..3, 4402, 0x6000, 0x6001, 0x6002, 0x6003, 0x6509</p> <p>Store is performed by writing the signature 0x65766173 (ASCII code “save” in little endian format) to sub-index 1. All other values are refused.</p> <p>Read access to sub-index 1 provides information on the storage functionality: Bit 0 = 1: device saves parameters on command Bit 1 = 0: device does not save parameters autonomously Bits 2..31 = 0: reserved</p> <p>NOTE! A write to the non-volatile memory is only performed when at least one parameter differs from the values already stored in the non-volatile memory.</p> <p>NOTE! customer default settings are standard user parameters and are overwritten by “Store all Parameters”.</p> <p>NOTE! Writeable in PRE-OPERATIONAL and OPERATIONAL state.</p>	0x01 (upon read)
1011:0	Restore Default Parameters	ARRAY	UNSIGNED8	ro	<p>Restore default parameters; sub-index 0 contains the highest sub-index that is supported.</p>	0x01

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1011:1	Restore All Parameters	ARRAY	UNSIGNED32	rw	<p>Restore all parameters temporarily to their factory default value. The default values become valid by issuing a reset of the device.</p> <p>Parameter restore is performed by writing the signature 0x64616F6C (ASCII code "load" in little endian format) to sub-index 1. All other values are refused.</p> <p>Read access to sub-index 1 provides information on the restore functionality:</p> <p>Bit 0 = 1: device restores default parameters</p> <p>Bits 1..31 = 0: reserved</p> <p>NOTE! The restore command issues a reboot and causes the user parameter file read to be skipped once. So the restored factory defaults are only valid until the next reset of the device. In order to permanently replace the user parameters with their factory defaults, execute a Store Parameters (object 0x1010).</p> <p>NOTE! customer defaults are written as the last step during production and subsequently act like normal parameters: A restore to firmware defaults overwrites customer's defaults.</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p>	0x01 (upon read)
1018:0	Identity Object	RECORD	Identity	const	Sub-index 0 contains the number of entries	0x04
1018:1	Identity Object: Vendor-ID	RECORD	UNSIGNED32	const	Vendor ID of Hengstler GmbH assigned by the ETG	0x20041961
1018:2	Identity Object: Product Code	RECORD	UNSIGNED32	const	<p>Product code of the EtherCAT slave in BCD coding.</p> <p>For the ACURO family, the product code is 547.</p>	0x00000547

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1018:3	Identity Object: Revision Number	RECORD	UNSIGNED32	const	Revision number of the EtherCAT slave Revision is checked for compatibility between ESI file and device and does not change as long as communication features remain the same. A consecutive number is used, beginning with 1 and incremented by 1 for every change to the device or its firmware that affects the ESI file.	--
1018:4	Identity Object: Serial Number	RECORD	UNSIGNED32	const	Serial Number of the EtherCAT slave (of bus hood, if differing from encoder base)	--
1A00:0	TxPDO1 Mapping Parameter	RECORD	UNSIGNED8	ro	PDO mapping TxPDO "Default" Sub-index 0 contains the number of mapping entries in this PDO.	1
1A00:1	TxPDO1 Mapping: Subindex 01	RECORD	UNSIGNED32	ro	1 st PDO mapping entry of TxPDO "Default": Object 0x6004:00 Position Value	0x60040020
1A01:0	TxPDO2 Mapping Parameter	RECORD	UNSIGNED8	ro	PDO mapping TxPDO "Type 1" Sub-index 0 contains the number of mapping entries in this PDO.	2
1A01:1	TxPDO2 Mapping: Subindex 01	RECORD	UNSIGNED32	ro	1 st PDO mapping entry of TxPDO "Type 1": Object 0x6004:00 Position Value	0x60040020
1A01:2	TxPDO2 Mapping: Subindex 02	RECORD	UNSIGNED32	ro	2 nd PDO mapping entry of TxPDO "Type 1": Object 0x2004:00 Warning Position Indication Flags	0x20040008
1A02:0	TxPDO3 Mapping Parameter	RECORD	UNSIGNED8	ro	PDO mapping TxPDO "Type 2" Sub-index 0 contains the number of mapping entries in this PDO.	3
1A02:1	TxPDO3 Mapping: Subindex 01	RECORD	UNSIGNED32	ro	1 st PDO mapping entry of TxPDO "Type 2": Object 0x6004:00 Position Value	0x60040020

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1A02:2	TxPDO3 Mapping: Subindex 02	RECORD	UNSIGNED32	ro	2 nd PDO mapping entry of TxPDO “Type 2”: Object 0x2002:06 Actual Speed	0x20020610
1A02:3	TxPDO3 Mapping: Subindex 03	RECORD	UNSIGNED32	ro	3 rd PDO mapping entry of TxPDO “Type 2”: Reserved for future use.	0x20030610
1A03:0	TxPDO4 Mapping Parameter	RECORD	UNSIGNED8	ro	PDO mapping TxPDO “Type 3” Sub-index 0 contains the number of mapping entries in this PDO. NOTE! The mapping of TxPDO4 “Type 3” is the same as in PDO type “Type 1” plus an additional entry for the system time at position latch.	2
1A03:1	TxPDO4 Mapping: Subindex 01	RECORD	UNSIGNED32	ro	1 st PDO mapping entry of TxPDO “Type 3”: Object 0x6004:00 Position Value	0x60040020
1A03:2	TxPDO4 Mapping: Subindex 02	RECORD	UNSIGNED32	ro	2 nd PDO mapping entry of TxPDO “Type 3”: Object 0x2009:00 System Time	0x20090020
1C00:0	Sync Manager Communication Type	ARRAY	UNSIGNED8	ro	Number of used Sync Manager channels	4
1C00:1	Sync Manager Communication Type	ARRAY	UNSIGNED8	ro	Communication type Sync Manager 0	1 (mailbox receive)
1C00:2	Sync Manager Communication Type	ARRAY	UNSIGNED8	ro	Communication type Sync Manager 1	2 (mailbox send)
1C00:3	Sync Manager Communication Type	ARRAY	UNSIGNED8	ro	Communication type Sync Manager 2	0 (process data output unused)
1C00:4	Sync Manager Communication Type	ARRAY	UNSIGNED8	ro	Communication type Sync Manager 3	4 (process data input, slave to master)
1C10:0	Sync Manager 0 PDO Assignment	ARRAY	UNSIGNED8	rw	No PDO assignment allowed for Sync Manager 0. Sub-index 0 has a fixed value of 0.	0

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1C11:0	Sync Manager 1 PDO Assignment	ARRAY	UNSIGNED8	rw	No PDO assignment allowed for Sync Manager 1. Sub-index 0 has a fixed value of 0.	0
1C12:0	Sync Manager 2 PDO Assignment	ARRAY	UNSIGNED8	rw	No PDO assignment allowed for Sync Manager 2. Sub-index 0 is implemented rw, but only accepts a value of 0. NOTE! Only writeable in PRE-OPERATIONAL state.	0
1C13:0	Sync Manager 3 PDO Assignment – Number of assigned PDOs	ARRAY	UNSIGNED8	rw	Number of assigned TxPDOs (0 or 1) To change the assignment, sub-index 0 must be written to 0, then the assignment is written and then sub-index 0 to the number of assigned PDOs. NOTE! The device only supports 1 assigned TxPDO at a time. NOTE! Only writeable in PRE-OPERATIONAL state.	1
1C13:1	Sync Manager 3 PDO Assignment	ARRAY	UNSIGNED16	rw	PDO mapping object index of assigned PDO: 0x1A00: TxPDO 1 ... 0x1A03: TxPDO 4 NOTE! Only writeable in PRE-OPERATIONAL state.	0x1A00: TxPDO 1 ("Default")
1C33:0	Sync Manager Synchronization SM3	RECORD	UNSIGNED8	ro	Number of Synchronization Parameters	32
1C33:1	Synchronization Type	RECORD	UNSIGNED16	rw	Synchronization type 0x01: synchronized with SM3 event 0x02: synchronized with DC Sync0 event NOTE! Only writeable in PRE-OPERATIONAL state.	0x01: Synchronized with SM3 event
1C33:4	Synchronization Types supported	RECORD	UNSIGNED16	ro	Bit 1: Synchronous supported Bit 4:2 : DC Type supported: 001 = DC Sync0	0x00000006

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
1C33:5	Minimum Cycle Time	RECORD	UNSIGNED32	ro	Indicates the minimum cycle time for the current configuration in nanoseconds.	62.5 μ s
1C33:6	Calc & Copy Time Inputs	RECORD	UNSIGNED32	ro	Indicates the time needed for position latch, readout and processing for the current configuration NOTE! Calc & Copy time only available in DC synchronized mode. Otherwise, this value is always zero.	-
1C33:12	Cycle Time Too Small Counter	RECORD	UNSIGNED16	ro	This error counter is incremented when the cycle time is too small so that the local cycle cannot be completed and input data cannot be provided before the next SM event.	
1C33:32	Sync Error	RECORD	BOOL	ro	1: synchronization error occurred	
2000:0	Warning Positions	ARRAY	UNSIGNED8	ro	Defines up to 4 warning positions based on the scaled position value. Depending on the setting in sub-index 1..4, the corresponding bit in object 0x2004 Warning Position Indication Flags or in TxPDO 2 is set when the position exceeds or falls below the programmed value. Sub-index 0: Number of warning positions	4

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
2000:1 ... 2000:4	Warning Position 1.4	ARRAY	UNSIGNED32	rw	<p>Warning position 1 .. 4:</p> <p>Bit 0..29: warning position in the range 0 to total measuring range (object 0x6002)</p> <p>Bit 30: 1 activates warning when position is equal to or lower than the warning position</p> <p>Bit 31: 1 activates warning when position is equal to or higher than the warning position</p> <p>NOTE! Warning position value range is max. 30 bits. If the total measuring range in object 0x6002 is larger, warning position cannot cover the whole measuring range.</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p>	<p>Bit 0..29: 0</p> <p>Bit 30: 0</p> <p>Bit 31: 0</p>
2001	Configured Offset Value	VAR	SIGNED32	rw	<p>Configured Offset Value</p> <p>NOTE! The offset value 0x2001 is manufacturer-specific and differs from both 0x650a:1 and 0x6509. It is added after scaling, preset calculation and modulo operation. So it serves to shift the value range of the scaled position from 0 .. "Total Measuring Range" (0x6002) to "Offset value" (0x2001) .. (0x6002 + 0x2001).</p> <p>NOTE! The configured offset value is only applied if scaling function is enabled (object 0x6000 Bit 2 = 1)</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p>	0
2002:0	Speed	RECORD	UNSIGNED8	ro	Highest supported sub-index	6

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
2002:1	Integration Time	RECORD	UNSIGNED16	rw	Integration time dT for speed calculation dPos/dT in milliseconds. Value range: 1...100 ms. Accurate speed calculation requires an integration time of at least 50 ms. The integration time is round up to an integer multiple of the EtherCAT cycle time.	100
2002:2	Multiplier Value	RECORD	UNSIGNED16	rw	Multiplier for speed scaling. Allows configuring the unit in which the speed value in 0x2002:6 shall be indicated. A value of "0" is ignored.	1
2002:3	Divider Value	RECORD	UNSIGNED16	rw	Divider for speed scaling. Allows configuring the unit in which the speed value in 0x2002:6 shall be indicated. A value of "0" is ignored.	1
2002:4	Speed Lower Limit	RECORD	UNSIGNED16	rw	Lower limit for speed (detects whether encoder runs too slow) Value range 0, 1..16383 If the absolute value of scaled speed falls below the limit configured here, the Frequency exceeded warning (object 0x6505, bit 0) is set and the corresponding emergency is thrown. A value of 0 disables lower speed limit check.	0
2002:5	Speed Upper Limit	RECORD	UNSIGNED16	rw	Upper limit for speed (detects whether encoder runs too fast) Value range 0, 1 .. 16383 If the absolute value of scaled speed exceeds the limit configured here, the Frequency exceeded warning (object 0x6505, bit 0) is set and the corresponding emergency is thrown. A value of 0 disables upper speed limit check.	0

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
2002:6	Actual Speed	RECORD	SIGNED16	ro	<p>Scaled speed value</p> <p>NOTE! Speed is calculated based on the difference of raw position values per integration time. Position scaling via CoE object 0x6001 is not taken into account. For multiplier = divider = 1, actual speed indicates raw position increments per second.</p> <p>NOTE! The actual speed is only valid as long as it is the range of Signed16 [-32767, 32765].</p> <p>NOTE! The actual speed can be converted to rotations per minute (RPM) by the following formula:</p> $\text{Actual Speed in RPM} = \frac{\text{Actual Speed in CoE object 2002:6} * \text{Multiplier Value in CoE object 2002:2}}{\text{Divider Value in CoE object 2002:3} * \text{Single Turn Resolution in CoE object 6501:0} * 60}$ <p>NOTE! Value of 0x7FFE indicates that the actual speed has reached the positive maximum value and is therefore invalid.</p> <p>NOTE! Value of 0x7FFF indicates an EtherCAT timing fault, which occurs, when the EtherCAT cycle time is too large for speed calculation. The fault can be resolved by reducing the DC cycle time or by increasing in the integration time in CoE object 2002:1.</p> <p>NOTE! Actual speed is only available when CoE object 1C13:1 “Sync Manager 3 PDO Assignment” equals 0x1A02. Otherwise its contents is undefined.</p>	

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
					<p>NOTE! The accuracy of the current speed data depends on the accuracy of the EtherCAT cycle time (operation mode “DC Sync0 for synchronization” is selected). The current speed can differ by 1 rpm but no more than 1% of the real speed.</p> <p>NOTE! The EtherCAT cycle time should not exceed 50% of the integration time defined in CoE object 2002:1. Otherwise a timing fault will occur (CoE object 2002:6 becomes 0x7FFF).</p> <p>NOTE! For single turn encoders, the EtherCAT cycle time should not exceed 1ms. Otherwise a timing fault will occur (CoE object 2002:6 becomes 0x7FFF).</p>	
2003:0	Acceleration	RECORD	UNSIGNED8	ro	Reserved for future use	
2003:1	Integration Time	RECORD	UNSIGNED16	ro	Reserved for future use	
2003:2	Multiplier Value	RECORD	UNSIGNED16	rw	Reserved for future use	
2003:3	Divider Value	RECORD	UNSIGNED16	rw	Reserved for future use	
2003:4/5	(unused)				Reserved for future use	
2003:6	Actual Acceleration	RECORD	SIGNED16	ro	Reserved for future use.	

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
2004	Warning Position Indication Flags	VAR	UNSIGNED8	ro	<p>Warning Position Indication Flags: the bits are set to 1 when the position equals, or either exceeds or falls below, depending upon the bit number, the programmed warning position.</p> <p>Bit 0: position equal to or above warning position 1 Bit 1: position equal to or below warning position 1 Bit 2: position equal to or above warning position 2 Bit 3: position equal to or below warning position 2 Bit 4: position equal to or above warning position 3 Bit 5: position equal to or below warning position 3 Bit 6: position equal to or above warning position 4 Bit 7: position equal to or below warning position 4</p> <p>NOTE! Warning Position Indication Flags are only available when configured in PDO. Otherwise read access to 0x2004 delivers a value of 0x00.</p>	0x00

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
2006	Encoder Protocol	VAR	UNSIGNED8	ro	<p>The protocol which is internally used by the bus hood for communication with the encoding module.</p> <p>1: BiSS-C 2: BiSS-C unidirectional 3: SSI Standard 4: SSI Extended 67: SSI Standard Gray Coded 68: SSI Extended Gray Coded 129: BiSS-C with inverted data line 130: BiSS-C unidirectional with inverted data line 131: SSI Standard with inverted data line 132: SSI Extended with inverted data line 195: SSI Standard with inverted data line, gray coded 196: SSI Extended with inverted data line, gray coded</p>	--
2009	System Time	VAR	UNSIGNED32	ro	<p>System time at last position latch in ns.</p> <p>NOTE! System time is only available when configured in PDO. Otherwise read access to 0x2009 returns a value of 0.</p>	---

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
2120:0	Temperature	RECORD	UNSIGNED8	ro	<p>Highest supported sub-index</p> <p>The encoder contains temperature limits in its protected area and checks actual temperature against these limits.</p> <p>In case of a BiSS-C encoder, the actual temperature can be read during process data exchange. In this case, the user has the possibility to write temperature limits (differing from the ones stored in the encoder ASIC). The bus hood cyclically checks actual temperature against these limits and generates a warning in object 0x6505 when the local temperature range is exceeded.</p>	4
2120:1	(unused)	RECORD	SIGNED8	ro		
2120:2	Temperature Lower Limit	RECORD	SIGNED8	rw	<p>Temperature lower limit in °C</p> <p>Value range: -128 °C to 127 °C</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p>	-40 °C
2120:3	Temperature Upper Limit	RECORD	SIGNED8	rw	<p>Temperature upper limit in °C</p> <p>Value range: -128 °C to 127 °C</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p>	+85 °C
2120:4	Actual Temperature	RECORD	SIGNED8	ro	<p>Actual temperature in °C</p> <p>The value is read out of the encoder cyclically every 1..2 sec for limit check in the bus hood. The latest buffered value is provided when an SDO upload is initiated.</p> <p>NOTE! If the actual temperature exceeds 127° C, the value in this sub-index remains limited to 127° C.</p> <p>NOTE! If the Encoder Protocol (object 0x2006) is set to “SSI standard” or “BiSS unidirectional”, this value is always zero.</p>	-
2121	Production Date	VAR	UNSIGNED16	ro	<p>Production month and year of bus hood in BCD format: 0xMMYY with MM = 01..12 and YY = 00..99</p>	--

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
2122	Order Number	VAR	VISIBLE_STRING	ro	Order number (of bus hood, if differing from encoder base)	"1547298"
4400:0	Debug Counters	RECORD	UNSIGNED8	ro	Diagnostic object to count encoder communication errors and EtherCAT synchronization misses.	2
4400:1	BiSS Global Error Counter	RECORD	UNSIGNED32	rw	Sum of all errors detected in BiSS/SSI communication Writing 0 resets the global error counter 0x4400:1, as well as BiSS master related counters in 0x4401.	0
4400:2	SM3 Missed Counter	RECORD	UNSIGNED32	rw	Counts "SyncManager 3 missed" events, i.e. input data has not been consumed by the SM3 EtherCAT frame during SYNC0 cycle. Writing 0 resets the counter.	0
4401:0	BissM Error Counters	RECORD	UNSIGNED8	ro	Diagnostic object with detailed counters for BiSS/SSI master errors.	21
4401:1 .. N	Debug Counters	RECORD	UNSIGNED32	ro	Various counters for various BiSS master error conditions and for residual value storage monitoring. Writing 0 to 0x4400:1 resets BiSS master related counters in 0x4401	
4402	Encoder Comm. Error Threshold	VAR	UNSIGNED8	rw	Threshold for encoder communication error emergency. Every erroneous BiSS/SSI frame increments a counter. Every error-free frame decrements the counter. When this counter reaches the threshold, the corresponding emergency is thrown NOTE! Only writeable in PRE-OPERATIONAL state.	1

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
6000:0	Operating Parameters	VAR	UNSIGNED16	rw	<p>Bit 0: code sequence: 0 =clock-wise, 1 = counter-clock-wise</p> <p>Bit 1: unused</p> <p>Bit 2: Scaling function control: 1= enable (see objects 0x6001, 0x6002, 0x2001)</p> <p>Bits 3..11: unused</p> <p>Bit 12: Residual value calculation (1 = enable); change of this bit becomes valid only after a reset of the device</p> <p>Bits 13..15: unused</p> <p>NOTE! setting code sequence to counter-clockwise causes a leap of the position value from (current position) to (position range – current position) and vice versa.</p> <p>NOTE! changing code sequence (bit 0) or enabling/disabling scaling (bit 2) “deletes” the preset: (see object 0x6003).</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p> <p>NOTE! Residual value calculation is only available for multi turn encoders.</p> <p>NOTE! Bits 1, 3..11 and 13..15 must be set to “0”.</p>	<p>Bit 0 = 0 (clock wise)</p> <p>Bit 1 = 0 Bit 2 = 0 (disable)</p> <p>Bits 3..11 = 0 Bit 12 = 0 (disable)</p> <p>Bits 13..15 = 0</p>
6001:0	Measuring Units per Revolution	VAR	UNSIGNED32	rw	<p>Specifies the desired resolution per revolution. The corresponding scaling factor is calculated in the device: $scf = (0x6001/0x6501)$.</p> <p>NOTE! change of the resolution 0x6001 “deletes” the preset (see object 0x6003). The scaled position is recalculated with the new scaling based on an offset of 0 (causing a leap in object 0x6004).</p> <p>Value range: $1 .. 2^{32} - 1$</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p>	Default: physical resolution per revolution (single turn resolution)

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
6002:0	Total Measuring Range in Measuring Units	VAR	UNSIGNED32	rw	<p>Specifies the desired total measuring range. The scaled position runs from 0 .. (0x6002 - 1). After the specified number of measuring steps, it returns to zero (modulo operation).</p> <p>NOTE! The allowed value range for object 0x6002 is 1 .. max. total physical resolution. In order to support the full range of 32 bits, a value of 0 is interpreted as 0x100000000, resulting in a scaled position range of 0 .. (2³² - 1).</p> <p>NOTE! change of the measuring range 0x6002 “deletes” the preset (see object 0x6003). The scaled position is recalculated with the new scaling based on an offset of 0 (causing a leap in object 0x6004).</p> <p>NOTE! Without residual value calculation, only values of (2x * 0x6001) with x = 0..12 are allowed. Otherwise</p> <ul style="list-style-type: none"> ▶ a leap in the scaled position occurs at every zero crossing of the physical position (every turn for a single-turn encoder, every 2x turns for an x-bit multi-turn encoder). ▶ the EtherCAT cycle time increases <p>NOTE! With residual value calculation, any value is allowed. In this mode, the residual error which occurs at every zero crossing of the physical position is compensated.</p> <p>NOTE! Only writeable in PRE-OPERATIONAL state.</p>	<p>Default: total physical resolution.</p> <p>Examples:</p> <p>AC58/1219: total resolution = 2¹² * 2¹⁹ = 2147483648 = x80000000</p> <p>AC58/1220: total resolution = 2¹² * 2²⁰ = 4294967296 = 0x100000000 → 0x00000000</p>

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
6003:0	Preset Value	VAR	UNSIGNED32	rw	<p>The position value of the encoder is set to the value specified in object 0x6003. Internally, the device calculates a corresponding offset value (offset = preset – position) which can be read via object 0x6509.</p> <p>NOTE! The preset should only be set to a value between 0 and [Total Measuring Range – 1]. Setting a higher preset can cause erroneous position data.</p> <p>Manufacturer-specific: Writing 0xFFFFFFFF to the preset value “deletes” the preset: object 0x6509 “Offset” is set to 0.</p> <p>NOTE! Changing code sequence or enabling/disabling scaling (0x6000) or changing scaling (0x6001/0x6002) “deletes” the preset: 0x6509 “Offset” is set to 0. The scaled position is recalculated with the new scaling based on an offset of 0 (causing a leap in object 0x6004).</p> <p>object 0x6003 “Preset value” is set to 0xFFFFFFFF to indicate that no valid preset has been set. Since the new scaling value is not set until the encoder enters the OPERATIONAL mode, after changing the scaling value the encoder must be placed in the OPERATIONAL mode before returning to the PRE-OPERATIONAL mode for the entering of the new preset value.</p> <p>NOTE! subsequent read accesses to this object deliver</p> <ul style="list-style-type: none"> - the last written preset value or - 0xFFFFFFFF if reset, see previous notes <p>regardless whether the physical position has changed in the meantime or not.</p>	0xFFFFFFFF

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
6004:0	Position Value	VAR	UNSIGNED32	ro	<p>Current position with applied scaling, preset and offset.</p> <p>NOTE! with scaling enabled, the range of object 0x6004 is limited to the minimum of the following values:</p> <ul style="list-style-type: none"> ▶ Measuring units per revolution * Multi-turn resolution (CoE 0x6001 * 0x6502) and ▶ Total measuring range configured in CoE 0x6002. 	
6500:0	Operating Status	VAR	UNSIGNED16	ro	This object reflects the actual settings in object 0x6000 "Operating parameters". See there for a description of the bits.	0x00000000
6501:0	Single-Turn Resolution	VAR	UNSIGNED32	ro	Number of measuring steps per revolution that are output for the absolute single-turn position value	<p>Depends on the encoder type. E.g. AC58/1217: single-turn resolution = 2^{17} = 131072 = 0x20000</p>
6502:0	Number of distinguishable Revolutions	VAR	UNSIGNED16	ro	Number of revolutions that the multi-turn encoder can measure (Multi-turn resolution).	<p>Depends on the encoder type. E.g. AC58/1217: multi-turn resolution = 2^{12} = 4096 = 0x1000</p>

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
6503:0	Alarms	VAR	UNSIGNED16	ro	<p>Additionally to the emergency messages, this object provides further alarm indications. An alarm is set if a malfunction in the encoder can lead to an incorrect position value. If an alarm occurs, the corresponding bit is set to 1 as long as the error is present.</p> <p>Bit 0: Position error (0 = no error, 1 = error)</p> <p>This bit is set when the encoder reports an error via the alarm/error bit in the cyclic data via BiSS/SSI or when an error occurs in the communication between bus hood and encoder.</p> <p>Bit 1..15: reserved (0)</p> <p>When an alarm occurs, an emergency message with error code 0x1000 (generic error) is generated as well. In the error register (object 0x1001), bit 0 "Generic error" is set as well. The manufacturer-specific error field of the emergency is specified in 10.1</p>	0x0000
6504:0	Supported Alarms	VAR	UNSIGNED16	ro	<p>Provides information on supported alarms by the encoder.</p> <p>Bit 0 = 1: Position error supported</p> <p>Bit 1..15: reserved (0)</p>	0x0001

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
6505:0	Warnings	VAR	UNSIGNED16	ro	<p>Warnings indicate that tolerance for certain internal parameters of the encoder have been exceeded. In contrast to alarms, warnings do not imply incorrect position values. A warning bit is set to 1 as long as the corresponding tolerance is exceeded.</p> <p>Bit 0: Frequency exceeded (0 = not exceeded, 1 = one of the speed limits specified in object 0x2002 is exceeded)</p> <p>Bit 1: reserved</p> <p>Bit 2: CPU watchdog state (0 = OK, 1 = CPU watchdog status reset generated)</p> <p>Bit 3: Operating time limit warning (0 = none, 1 = 100,000 hours exceeded)</p> <p>Bit 4..13: reserved (0)</p> <p>Bit 14: EtherCAT synchronization Warning (0 = OK, 1 = SM watchdog or cycle missed occurred at least once (The bit is reset when the corresponding CoE counter object is written to 0)</p> <p>Bit 15: Temperature limit exceeded (0=not exceeded, 1=one of the temperature limits specified in object 0x2120 is exceeded)</p> <p>When a frequency exceeded warning occurs (bit 0), an emergency message with error code 0xFF00 (device specific) is generated as well.</p> <p>When a CPU watchdog state or operating time limit warning occurs (bit 2 or 3), an emergency message with error code 0x5000 (device hardware) is generated as well.</p>	0x0000

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
					The error register (object 0x1001) is not influenced by the warnings in object 0x6505; the error register is copied unchanged in the respective emergency message, the manufacturer-specific error field of the emergency is specified in 10.1.	
6506:0	Supported Warnings	VAR	UNSIGNED16	ro	Provides information on supported warnings by the encoder. Bit 0 = 1: Frequency exceeded warning supported Bit 1 = 0 (reserved) Bit 2 = 1: CPU watchdog state warning supported Bit 3 = 1: Operating time limit warning supported Bit 4..13 = 0: (reserved) Bit 14 = 1: Synchronization warning supported Bit 15 = 1 if temperature is available from encoder	0xC00D
6507:0	Profile and Software Version	VAR	UNSIGNED32	ro	Provides the implemented encoder device profile version and the manufacturer-specific software version. Byte 0: Profile version minor Byte 1: Profile version major Byte 2: Software version minor Byte 3: Software version major	Depends on the implemented software version. E.g. 0x01010302: Software version 1.1 profile version 3.2
6508:0	Operating Time	VAR	UNSIGNED32	ro	Provides the operating time since power-on in 0.1 hours resolution. NOTE! if the operating time exceeds 1,000,000 (100,000 h) an operating time limit warning is generated (see object 0x6505).	-

Index Sub-index (hex)	Name	Object	Data Type	Access	Meaning	Default Value
6509:0	Offset Value	VAR	SIGNED32	ro	Provides the offset value calculated by the preset function (see object 0x6003)	0
650A:0	Module Identification	ARRAY	UNSIGNED8	ro	Provides manufacturer-specific offset value Sub-index 0 = highest supported sub-index	0x01
650A:1	Manufacturer Offset value	ARRAY	SIGNED32	ro	According to profile, this value gives information on the shift of the zero-point in the number of positions from the physical zero point of the encoder disk. The value is only changeable by the encoder manufacturer. For the ACURO encoder series, this offset value is always zero.	0x00000000
650B:0	Serial Number	VAR	UNSIGNED32	ro	Provides the encoder serial number and is hard-wired to the identity object 0x1018, sub-index 04.	-

Tab. 10: Object Dictionary

6.2 Object Dictionary Measured Value Processing

6.2.1 Position Scaling

This section contains the processing chain to calculate the position value (CoE 0x6004) from raw sensor data. Raw position in case of multi-turn encoder is defined in Formula 1.

$$\text{RawPosition} = (\text{MultiturnValue} * \text{SingleTurnResolution}) + \text{SingleturnValue}$$

Formula 1: Calculation of raw position

The position value is calculated by Formula 2.

$$\text{POS} = (((\text{RawPosition} * \text{MUL}) / \text{DIV}) + \text{OFS}) \% \text{MOD}$$

Formula 2: Calculation of position value CoE 0x6004

The operands given in Formula 2 are pre-calculated from values given by the encoder device profile CiA406. This is necessary to be independent from fieldbus stacks and profiles and also to get an optimized range of values. The operands MOD and OFS are taken directly from CoE objects 0x6002, "Total measuring range in measuring units" and 0x6509, "Offset value":

$$\text{MOD} = \text{CoE } 0x6002$$

$$\text{OFS} = \text{CoE } 0x6509$$

Formula 3: Calculation of MOD and OFS

The calculation of `MUL` and `DIV` depends on CoE objects 0x6001 “Measuring units per revolution” and 0x6501 “Single-turn resolution”. Currently they are filled directly with these values:

`MUL = CoE 0x6001`

`DIV = CoE 0x6501`

Formula 4: Calculation of MUL and DIV

NOTE! The resulting scaled position must fit into an `UINT32` data type. It is in the responsibility of the user to configure all scaling parameters accordingly.

NOTE! It has to be taken into account that `OFS` is an `INT32` data type and therefore the position can only contain an offset of $+\frac{1}{2} * ((RPS * MUL) / DIV) - 1$ or $-\frac{1}{2} * ((RPS * MUL) / DIV)$.

6.2.2 Preset Function

The position value of an ACURO AC58 EtherCAT® encoder can be changed by the use of a preset value. This allows the encoder to be placed in any position when installed, and a preset value used to match the position output to the desired machine position. For example, the encoder can be made to read “0” when a machine is at its home position.

If the parameters are first saved to memory using the “Store All Parameters” function, the encoder saves the offset between the actual encoder position value and the position value when the preset is applied. This offset value is then used if power is lost and then restored to ensure that the correct position value is reported even if the encoder shaft has been moved while power was off.

6.2.3 Residual Value Calculation

If the encoder is in continuous operation and the factor between `Scaled Range` and `Physical Range` is not a power of two, a residual error would occur if `Physical Range` is reached. This is shown in Fig. 11. To compensate this error - which occurs at every zero crossing of `Physical Range` - an offset has to be added. The calculation of this offset is called the residual value calculation.

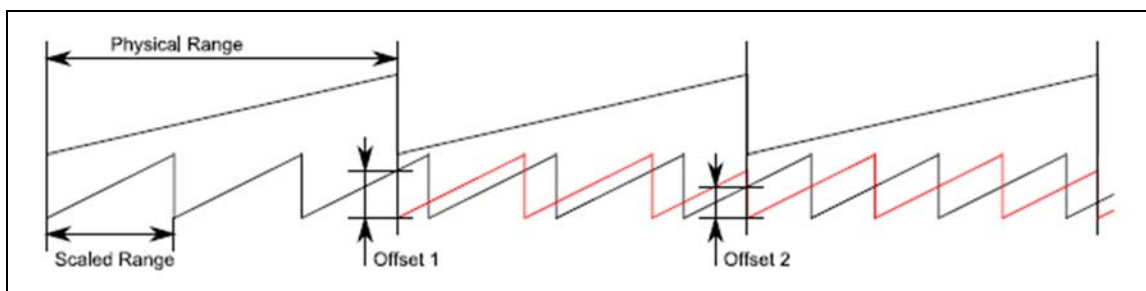


Fig. 11: Errors due to non-integer multiples of Scaled Range to Physical Range

Residual value calculation is activated if bit 12 of CoE object 0x6000 “Operating parameters” is set and the factor between `Scaled Range` (CoE object 0x6002 “Total measuring range in measuring units”) and `Physical Range` is not a power of two.

The encoder saves the parameters which are needed for residual value calculation in the FRAM at every quadrant change of the physical range. This allows detecting a movement of up to 1020 rotations in powered-down state and correcting the residual value from the stored parameters after restarting the device.

NOTE! The Residual Value feature is not available with single-turn versions of the encoder. Therefore, Scaled Range can be used only if the factor between Scaled Range and Physical Range is a power of two. If the factor is other than two, please use the scaling function of the controller to scale the encoder output.

NOTE! The residual value logic is only functional when any unpowered rotation of the multi-turn encoder does not exceed 1020 revolutions and when power is restored for a minimum of 11 continuous seconds while still within these 1020 rotations.

NOTE! If using Residual calculation, the allowed number of quadrant changes is limited to the maximum number of write cycles for this memory (10 trillion).

6.2.4 Feature Interactions

The following chart should help simplify which of the previous and related features may be used together and under which circumstances.

Feature	Scaling	Residual Value	Measuring Units per Revolution	Measuring Range	Preset Value
Object >>	6000:0 bit 2	6000:0 bit 12	6001:0	6002:0	6003:0
	OFF	OFF	NOT POSSIBLE	NOT POSSIBLE	FUNCTIONAL
	ON	OFF	2 ⁿ	2 ⁿ	FUNCTIONAL
	ON	OFF	2 ⁿ	NOT 2 ⁿ	NOT FUNCTIONAL
	ON	OFF	NOT 2 ⁿ	IRRELEVANT	NOT FUNCTIONAL
	ON	ON	IRRELEVANT	IRRELEVANT	FUNCTIONAL

Tab. 11: Feature Interactions

NOTE! Scaling On/Off, Residual Value, Measuring Units Per Revolution and Measuring Range all affect the internally calculated scaling value. Since the new scaling value is not set until the encoder enters the OPERATIONAL mode, after changing any feature that affects the scaling value, the encoder must be placed in the OPERATIONAL mode before returning to the PRE-OPERATIONAL mode for the entering of the new preset value.

6.2.5 Speed Calculation

Raw speed is calculated as difference of the raw position values per integration time, as shown in Formula 5:

$$\text{RawSpeed} = \frac{\text{RawPosition}(t=0) - \text{RawPosition}(t = -\text{IntegrationTime})}{\text{IntegrationTime}}$$

Formula 5: Calculation of raw speed in unscaled measuring units

`IntegrationTime` is given by CoE object 0x2002:1 “Speed integration time” in steps of 1 millisecond. If CoE object 0x2002:1 is set to zero, `IntegrationTime` will be equal to the field bus cycle time.

Due to time quantization, `IntegrationTime` will be rounded up to an integer multiple of the cycle time. Internally, `RawSpeed` is normalized to increments/sec.

NOTE! The resulting scaled speed must fit into an INT16 data type. It is in the responsibility of the user to configure all speed calculation parameters accordingly.

Due to quantization of position values, a leap in speed values is possible at slow speeds as shown in Fig. 12. Therefore `IntegrationTime` needs to be increased according to the application.

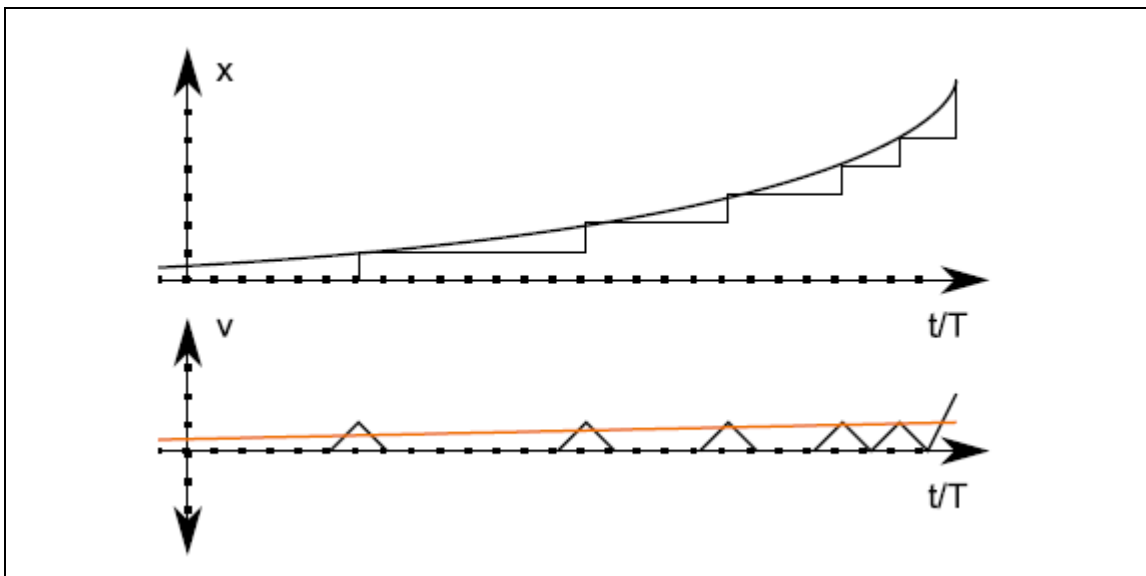


Fig. 12: Leap in speed values

For adjusting the speed unit, `RawSpeed` is scaled by two constants `C_1` and `C_2` as shown in Formula 6. The speed is calculated periodically in intervals of `IntegrationTime`.

$$\text{Speed} = \text{RawSpeed} * C_1 / C_2$$

Formula 6: Getting speed (velocity) from raw velocity

The constant `C_1` is set via CoE object 2002:2 “Multiplier value”, `C_2` via CoE object 2002:3 “Divider value”. The output value `Speed` is stored in CoE object 2002:6.

7 Commissioning with TwinCAT®

7.1 Introduction to TwinCAT®

TwinCAT is a tool set for developing applications for EtherCAT network communication. For commissioning and parameterization of the ACURO AC58 Industry absolute shaft encoders, you will need the TwinCAT I/O Driver component. TwinCAT 3.1 can be downloaded from the download section of the BECKHOFF website at www.beckhoff.com.

TwinCAT provides EtherCAT Master functionality on your industry-standard PC with a standard network adapter.

7.2 Installation of TwinCAT®

First, you have to install the engineering (XAE) and runtime (XAR) components of TwinCAT to your PC. In order to install these components of TwinCAT 3 to your PC, proceed as described in the TwinCAT installation document provided by Beckhoff, see reference [8]. This document also contains a list of the system requirements to be fulfilled.

7.3 Configuring TwinCAT® to use Sync Manager 3

The following description assumes that TwinCAT 3 is installed on your computer in the default folder “C:\TwinCAT”.

As TwinCAT has now been installed successfully, you can continue commissioning and parameterization by starting TwinCAT in the next step and subsequently performing the remaining steps within TwinCAT.

NOTE! The following steps refer both to TwinCAT 3.0 and TwinCAT 3.1 and are not applicable to any previous version of TwinCAT.

7.3.1 Starting TwinCAT®

The EtherCAT configuration software TwinCAT can be started in two ways:

- ▶ Either using the Windows start menu entry *Beckhoff > TwinCAT3 > TwinCAT 3 XAE*
- ▶ Or via the context menu:
 1. Click on the TwinCAT icon using the right mouse button
 2. The TwinCAT context menu appears:



3. Select entry TwinCAT XAE.

7.3.2 Installation of ACURO EtherCAT® within TwinCAT®

Now, TwinCAT needs to install the network adapter to be used. In order to do so:

- Select menu entry *TwinCAT* > *Show Real Time Ethernet Compatible Devices*.

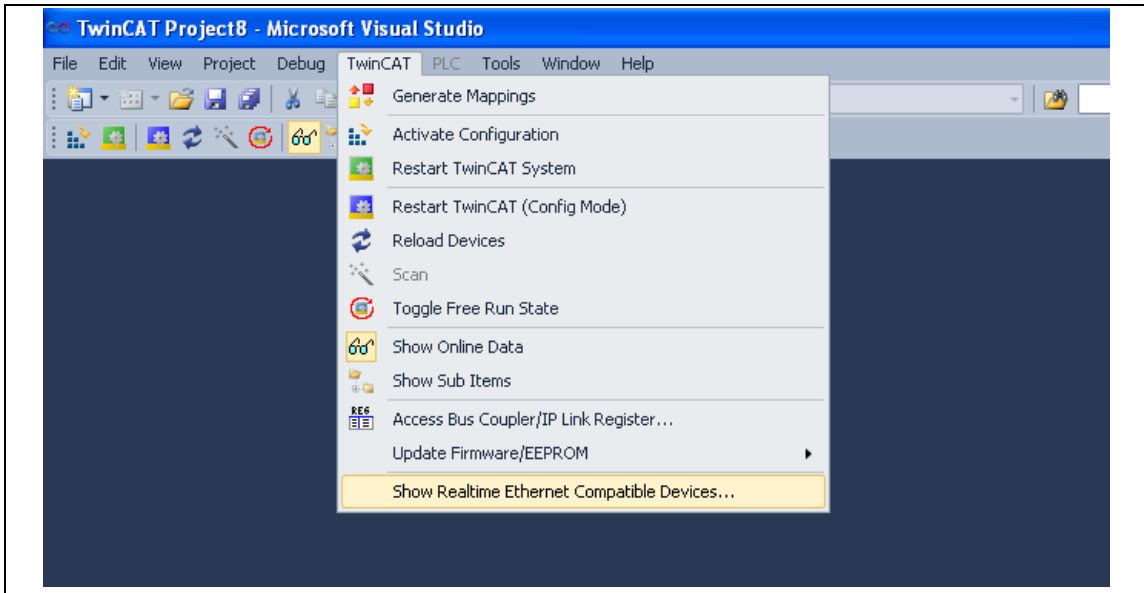


Fig. 13: TwinCAT menu entry *TwinCAT* > *Show Real Time Ethernet Compatible Devices*

The dialog box *Installation of TwinCAT RT-Ethernet Adapters* appears. Both compatible and incompatible network adapters are now listed if present.

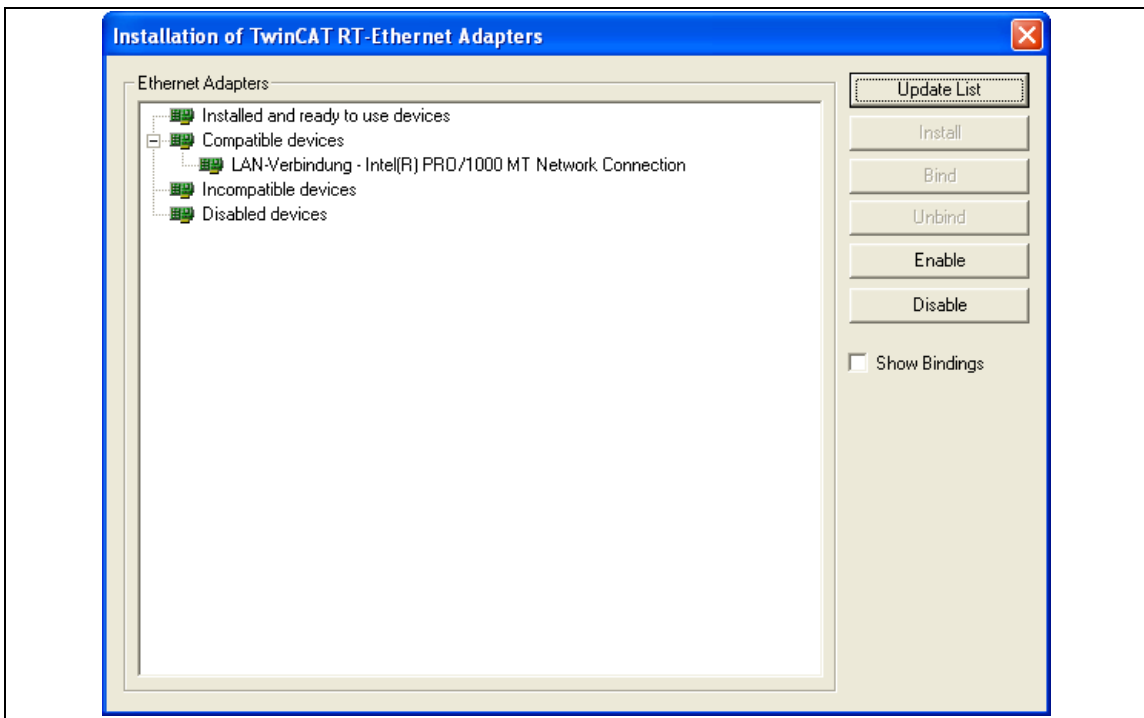


Fig. 14: Installation of TwinCAT RT-Ethernet Adapters – Compatible Devices

7.3.3 Selection of TwinCAT® Ethernet Driver

TwinCAT offers two different drivers for Ethernet-based network adapters:

- ▶ TwinCAT Driver for Real-Time Ethernet (Compatible devices)

The TwinCAT Driver for Real-Time Ethernet only works with network adapters using an Intel chipset. It provides hard real-time support for EtherCAT applications.

- ▶ TwinCAT Real-Time Ethernet Intermediate Driver (Incompatible devices)

The TwinCAT Real-Time Ethernet Intermediate Driver also works with network adapters using another chipset than the Intel chipset. However, this driver has been developed for evaluation purposes in laboratory and testing. It does not provide hard real-time support for EtherCAT applications. Also, it may cause jitter.

In the following example, the compatible driver is installed:

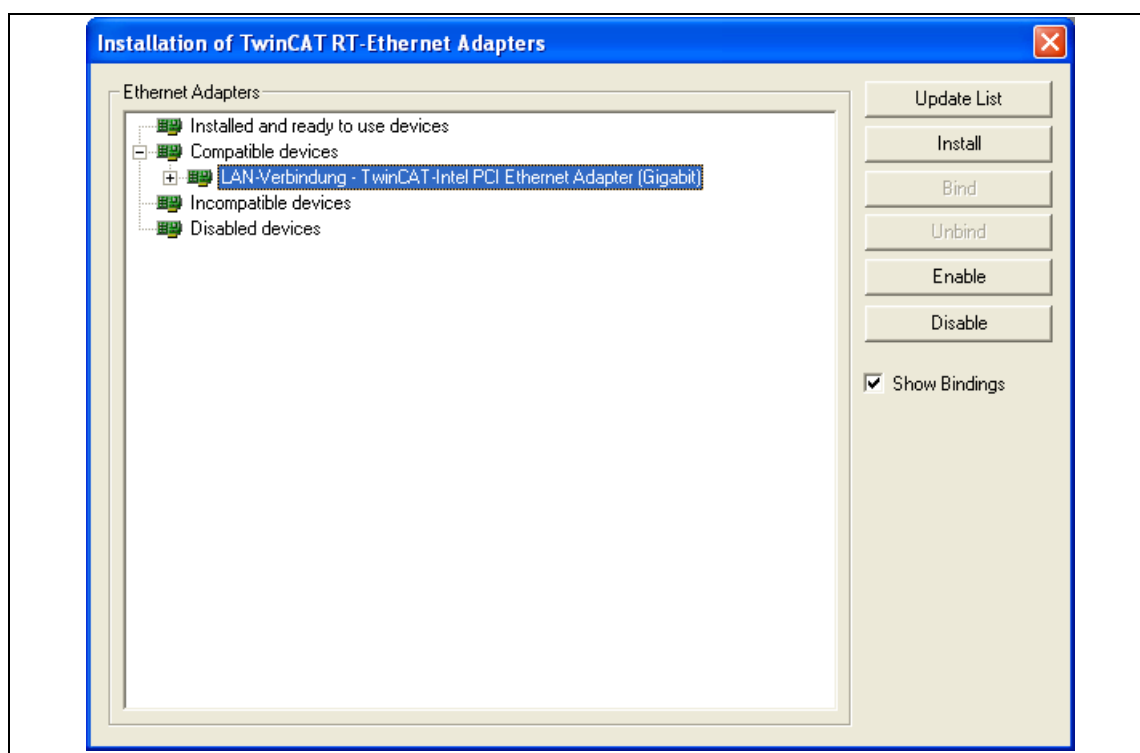


Fig. 15: Installation of TwinCAT RT-Ethernet Adapters – Installed and ready-to-use Devices

- Close TwinCAT now!

7.3.4 Copying ESI File into TwinCAT® Installation

The ESI file (i.e. the XML File containing EtherCAT device descriptions) has to be copied to the TwinCAT installation. If you installed TwinCAT to the directory `C:\TwinCAT`, proceed as follows:
For TwinCAT 3.0:

- Copy the file `ACURO.xml` to directory `C:\TwinCAT\3.0\Config\Io\EtherCAT`.

For TwinCAT 3.1:

- Copy the file `ACURO.xml` to directory `C:\TwinCAT\3.1\Config\Io\EtherCAT`.

Otherwise, you need to adapt the directory path accordingly. These files are available from the Hengstler web site at www.hengstler.com.

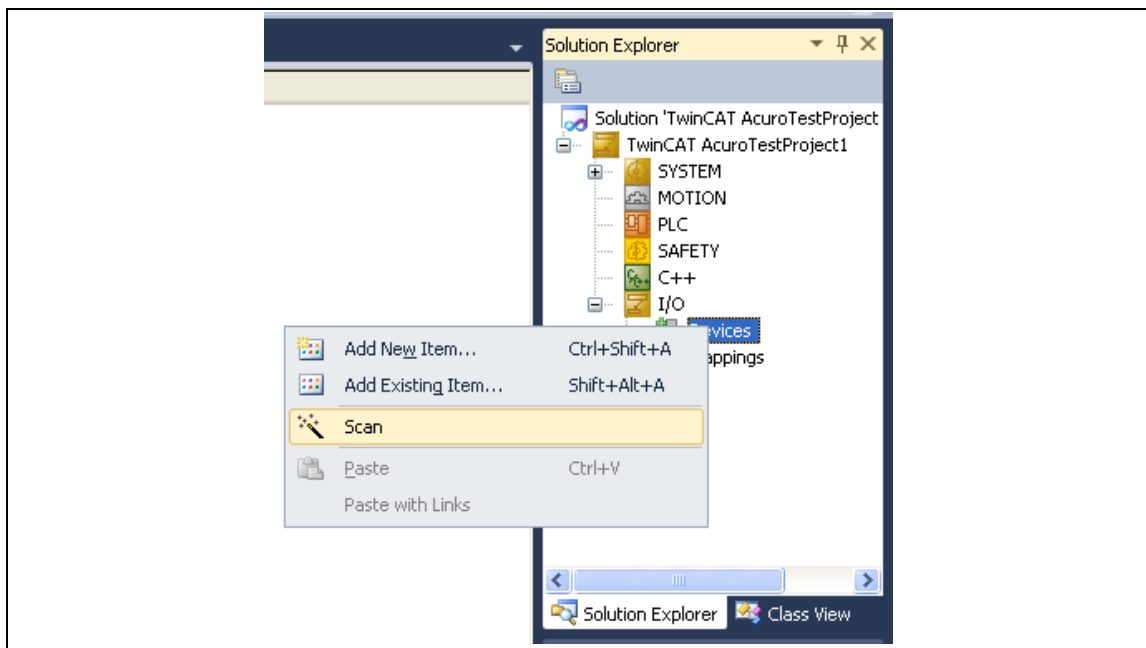
7.3.5 Scanning for Devices

First, TwinCAT has to be restarted in order to scan for devices. This is necessary as during restart, TwinCAT will be forced to read and process the device description file.

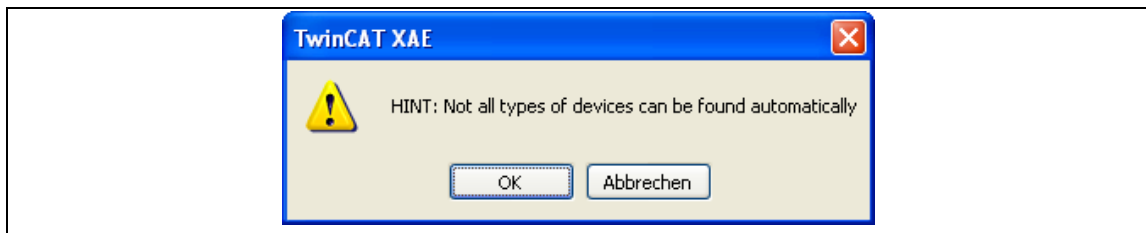
There are two alternative ways to force TwinCAT to scan for devices:

Click the *Scan* icon  within the TwinCAT XAE icon bar.

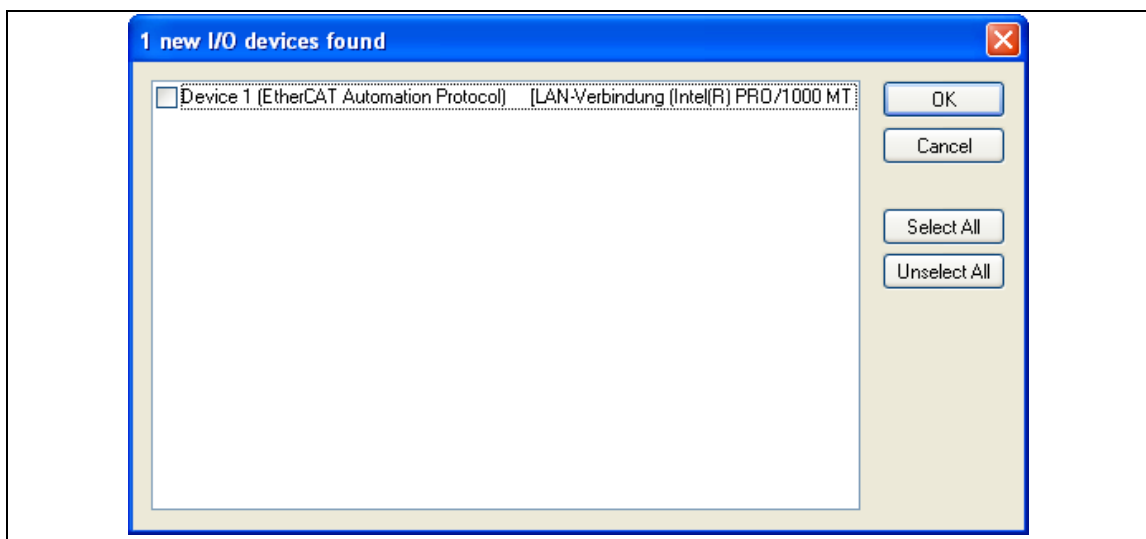
Alternatively, you can select the *Scan* option as shown below from the context menu of entry devices in the tree of the Solution Explorer by right-clicking at that entry:



Next, the following informative dialog box will appear.

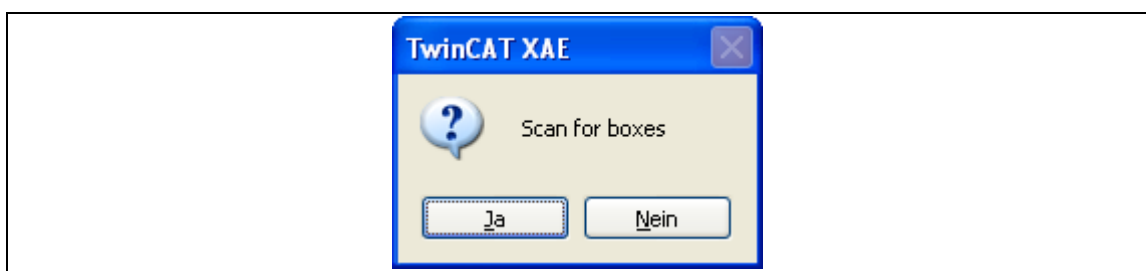


- Click at the **Ok** button to confirm it. TwinCAT will scan for suitable connection and network adapters then.

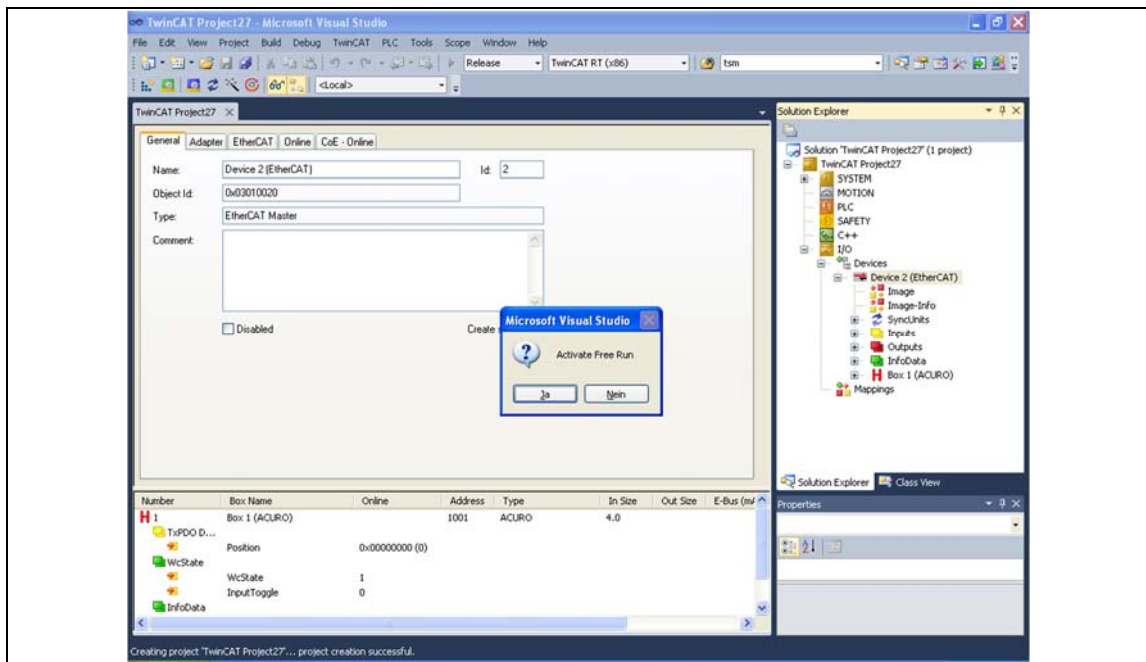


- Check the configured connection/ network adapter.
- Then click at the **Ok** button to select this network adapter as the one being used for scanning.

Again, a dialog box will appear asking you whether to start the main part of the scanning process. Click at the **Ok** button to proceed.



Then, the device scanning process will be initiated.

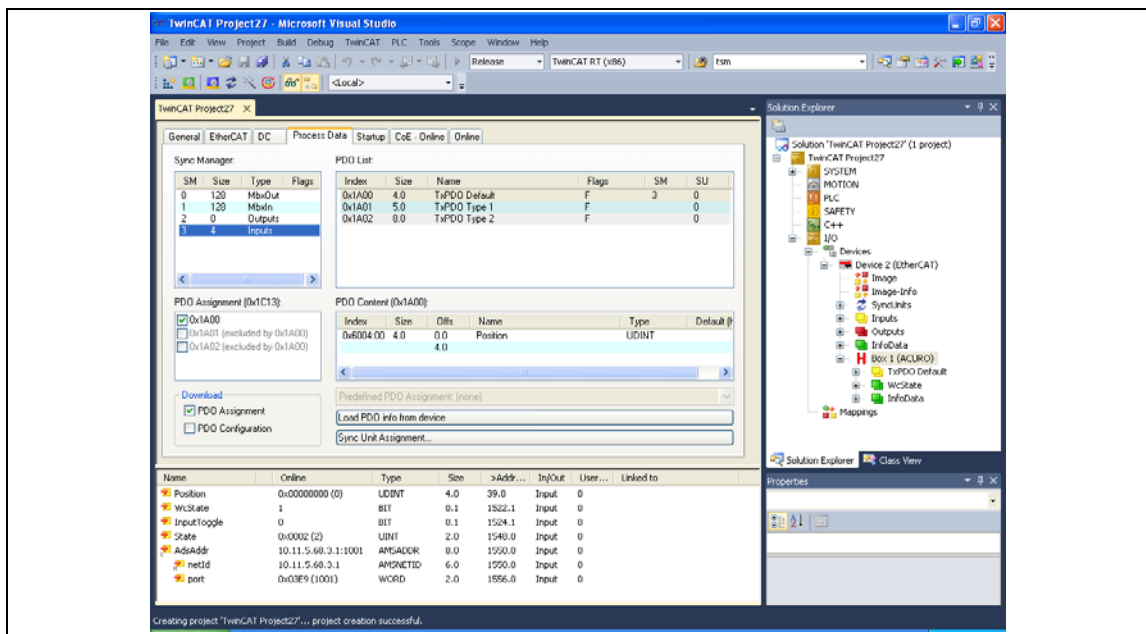


- Click on **Nein** (or **No**) as free run does not need to be activated during the subsequent configurations.

Free run will be activated later. As long as free run has not been activated, the device will not reach the state *Operational*.

7.3.6 Load Process Data from Device

- Select register card **Process Data**:



7.3.7 Select Synchronization Mode

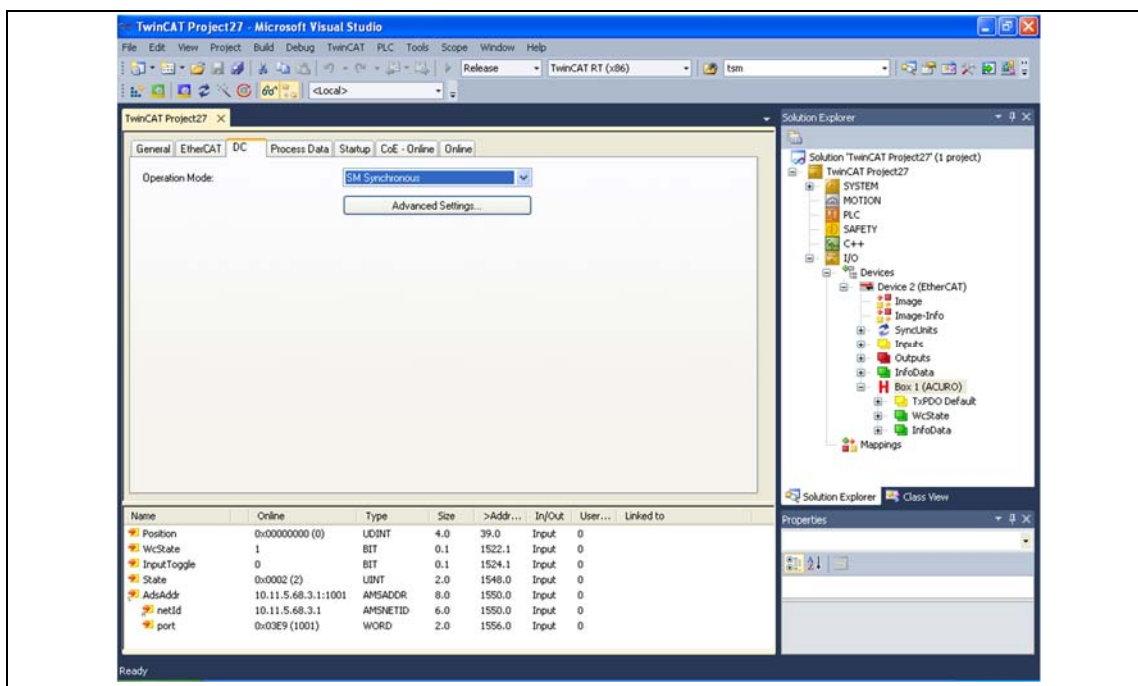
If you want to configure the absolute shaft encoder ACURO EtherCAT® for the first time or you do not need the distributed clocks feature, choosing *SM Synchronous* operation is recommended.

You can adjust this as follows:

- Select register card **DC**.
- Select option **SM Synchronous** in the selection list **Operation Mode** on that register card.

If you want to use the distributed clocks feature for precise time measurement and synchronization, select option **DC SYNC0 for Synchronization** in the selection list **Operation Mode**.

Concerning this topic refer to section 7.4, Configuring TwinCAT® for Distributed Clocks

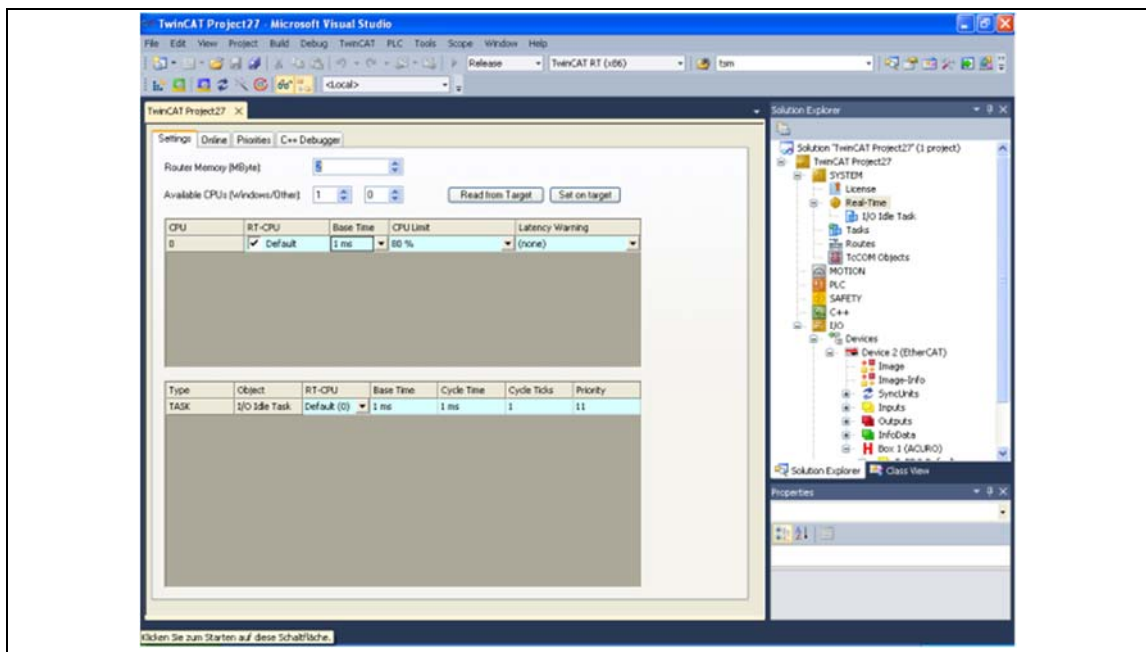


7.3.8 Adjust Base Time

Now adjust the base time and cycle ticks to configure the desired cycle time.

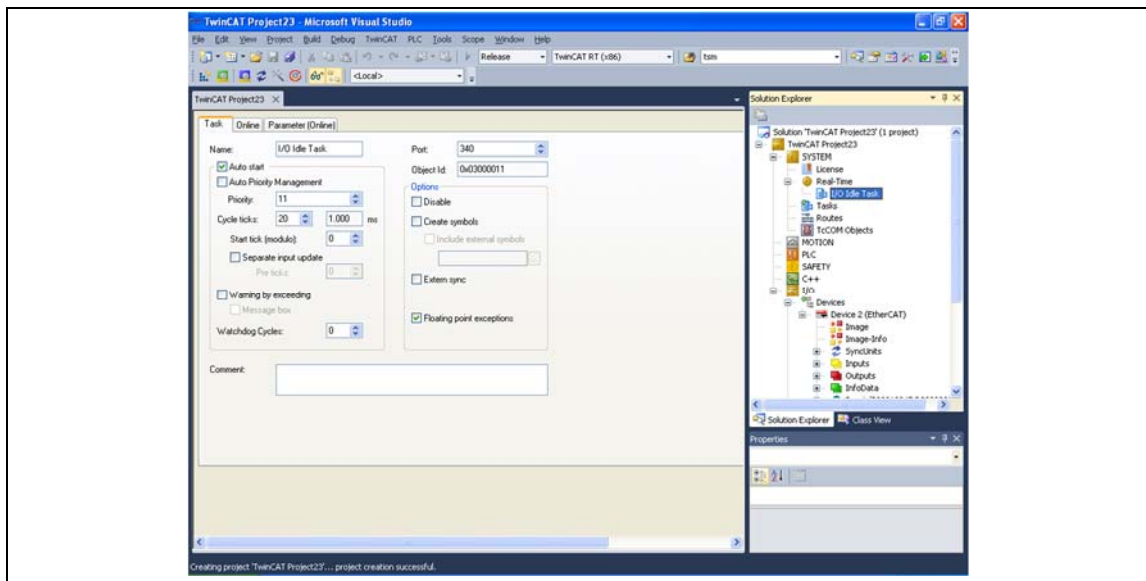
The following example adjusts the base time to 1 millisecond and the cycle ticks to 1 in order to gain a cycle time of 1 millisecond:

- Click on Real-Time within the sub tree *SYSTEM* in the Solution Explorer.
- The register *Real-Time* appears. The register card *Settings* is opened.
- You can now set the Base Time to 1 millisecond using the selection list in the third column named *Base Time* (of the upper table in register card Settings. See below.)



- Click on I/O Idle Task within the sub tree *SYSTEM* in the Solution Explorer.
- The register *I/O Idle Task* appears. The register card *Task* is opened.
- You can now set the Cycle Ticks to 1 (see figure below).


This is the factor to be multiplied with Base Time adjusted in the preceding step. In the current example, the factor 1 multiplied with a Base Time of 1 millisecond will amount to a cycle time of 1 millisecond. When you adjust the Cycle Ticks with the selection list in this dialog, TwinCAT immediately calculates this time value and displays the result right beside the selection list.



Later on, you may adjust the base time to other values, if required.

7.3.9 Further Steps

You can now continue with further steps according to your needs such as creating variables or tasks. If all configuration steps have been finished, then you have to activate the *Free Run*. As the *Free Run* has not yet been activated, you should do this now. Proceed as follows:

- Click at the configuration mode icon .
- The dialog box *Restart TwinCATSystem in Config Mode* appears.
- Click on *Ok*.
- The dialog box *Load I/O Devices* appears.
- Click on *Yes*.
- The dialog box *Activate Free Run* appears.
- Click at *Yes*.

7.4 Configuring TwinCAT® for Distributed Clocks

The concept of *Distributed Clocks (DC)* allows a more precise synchronization of multiple EtherCAT slaves between one another than synchronizing on a passing process data frame *SM Synchronous*. This is mainly caused by master-side frame jitter.

The devices have to be configured accordingly to be able to adjust their local clock to a reference clock. The master decides which device's clock serves as reference clock. In most cases the subsequent device in the topology which supports DC is used as reference clock. In the standard configuration, the master also adjusts its clock to the reference clock.

For more information on DC Timing with TwinCAT, see section 5.5.3, "DC Timing with TwinCAT".

If TwinCAT is used as EtherCAT master, it calculates the time points to send the frame related to the DC base cycle. The offset is selected to ensure that the data is received by the slave before synchronization event at slave side. Additionally an individual offset can be selected by the user for each slave to define the specific SYNC0 offset (see Fig. 10: DC timing with TwinCAT). The shift value will be written to the slave using a proper InitCmd, which is configured in the EtherCAT Network description file (ENI, see reference [6]). Further the keyword `<EtherCATConfig>` `<Config><Slave><DC><ShiftTime>` will be defined too (see "Shift time of SYNC0 event in ns" in the ENI specification published by the ETG).

In order to choose the distributed clocks mode, select the option *DC Synchronous* in the selection list *Operation Mode* of the DC register card which is displayed when having the ACURO selected in the solution explorer tree.

Parameterization for DC in TwinCAT®

For DC Sync choose option *DC SYNC0 for synchronization*, see Fig. 9: Selection of Synchronization Mode" below.

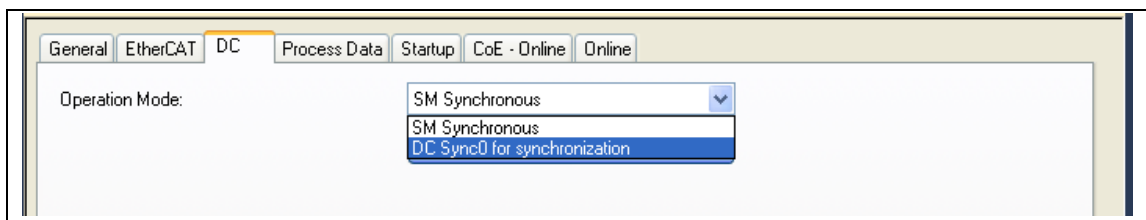



Fig. 16: Selection of Synchronization Mode

After selection of this option, the configuration has to be activated  and the EtherCAT Slave proceeds to state *Operational*.

For a standard installation, see section 7.3.4., "Copying ESI File into TwinCAT® Installation" regarding path information and restart.

If there is no ESI file available, the configuration can alternatively also be performed using the *Advanced Settings*, see Fig. 17, "Configuration for Activation of SYNC0 Signal within the Advanced Settings". Here, the SYNC0 signal is a synchronization signal generated by the local clock of the device.

NOTE! The ACURO AC58 EtherCAT encoder supports the SYNC0 signal.

- On the register card DC, click on the *Advanced Settings* button. The *Advanced Settings* dialog box will appear.

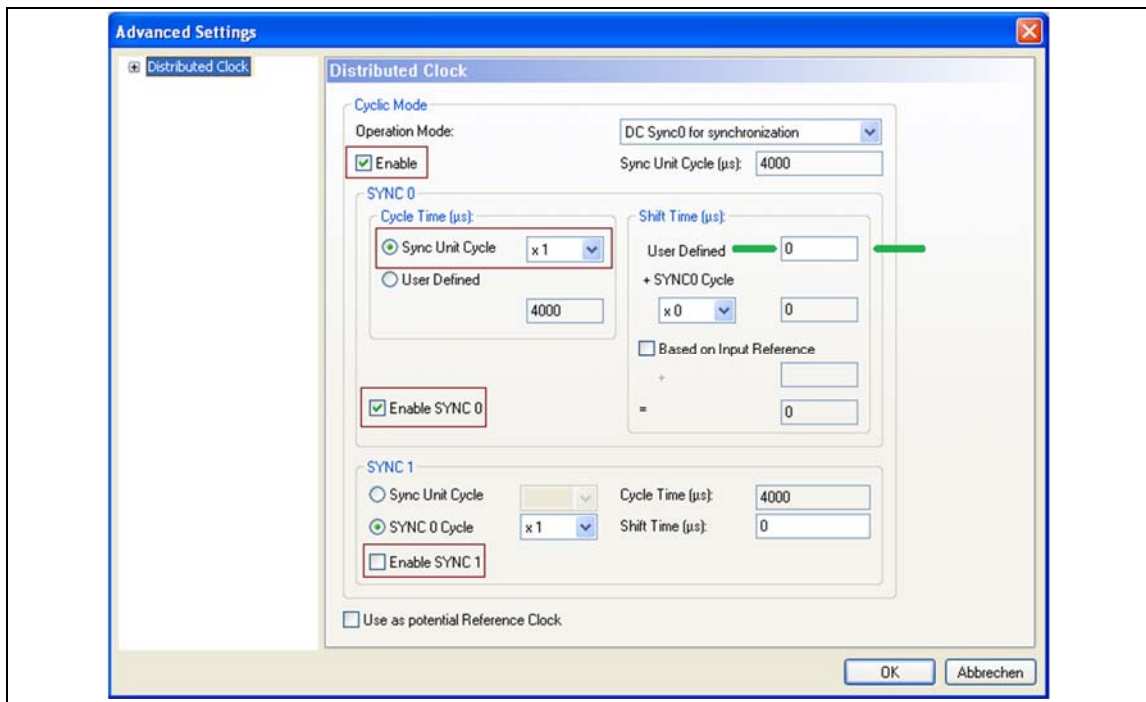


Fig. 17: Configuration for Activation of SYNC0 Signal within the Advanced Settings

The individual slave-specific shift time S_0 mentioned in section 5.5.3, “DC Timing with TwinCAT” and Fig. 10: DC timing with TwinCAT can be adjusted according to your needs in the input field *User Defined* in the *SYNC0>Shift Time (µs)* area of the advanced settings dialog of the DC register card. In the example above, the value 0 has been assigned to this input field (between the green tags).

NOTE! In case of short cycle times, a negative value for the slave SYNC0 Shift Time may be needed in order to trigger earlier reading of input signals than defined by the EtherCAT master as a standard. By default, TwinCAT uses a SYNC shift time of 30% (TwinCAT 2) or 10% (TwinCAT 3) of the cycle time for the inputs. If the sum of Calc & Copy Time (CoE object 0x1C33:6) and frame jitter of the master exceeds the master’s SYNC shift time for inputs, a recommended User Defined Shift Time is:

$$\text{UserDefinedShiftTime} = -(\text{CalcAndCopyTime} + \text{MaximumFrameJitter} - \text{SyncShiftTimeForInputs})$$

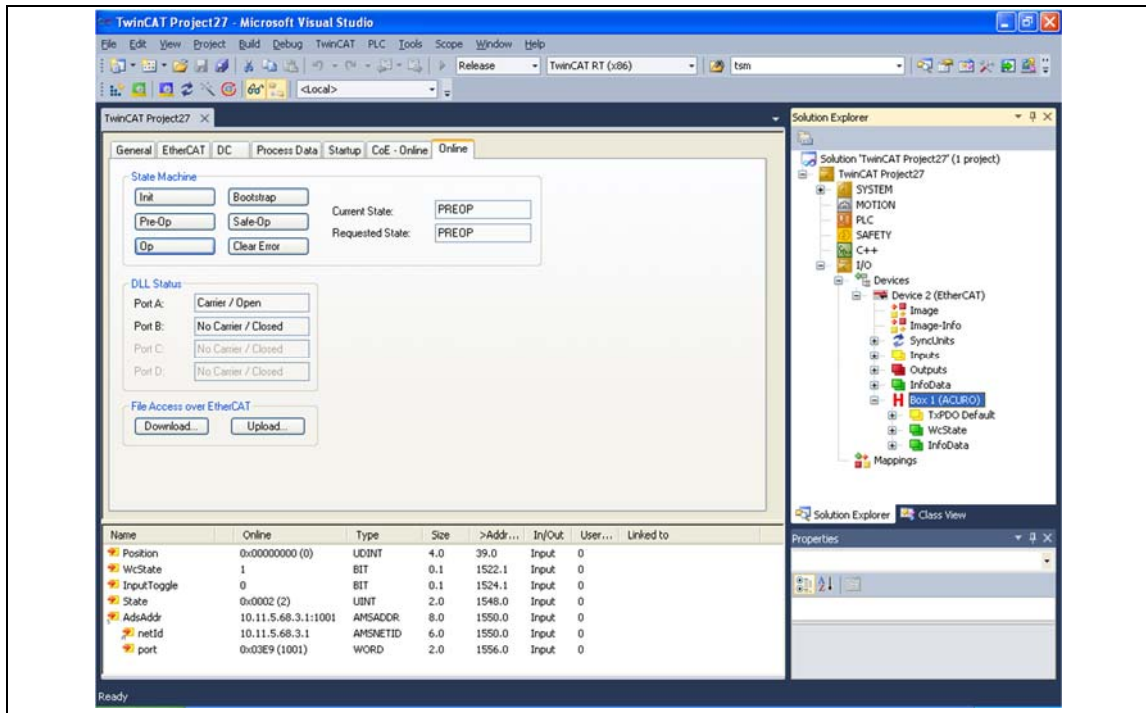
This formula applies when the option “Based on Input reference” is selected in the Advanced Settings. For details on the shift time parameters, see the TwinCAT documentation.

For more information concerning the dependence of the achievable cycle times from conditions and configuration state, see chapter “Network Performance”.

7.5 Troubleshooting

7.5.1 Checking Data Exchange with TwinCAT®

The status of the general data exchange can be checked in the TwinCAT window *Online*.



7.5.2 Common Error Messages

No new I/O devices found



This message is issued if the scan for I/O devices fails. Check connections and retry to scan for I/O devices.

8 Firmware Update

8.1 General Information

The firmware of the ACURO AC58 EtherCAT® shaft encoder device can be updated using the mailbox protocol FoE (File Access over EtherCAT). The transmission of data via FoE is possible in EtherCAT states (ESM):

- ▶ “Bootstrap”,
- ▶ “Pre-Operational”
- ▶ “Safe-Operational”
- ▶ and “Operational”

See section 5.4.3.1, “The EtherCAT® State Machine (ESM)”

NOTE! As the expected bus load is lower in the “Pre-Operational” state, it is recommended to perform firmware updates while in this state.

The following firmware components can be updated:

- ▶ netX EtherCAT Slave Firmware for Encoder

The transferred firmware is checked for consistency and compatibility before being applied.

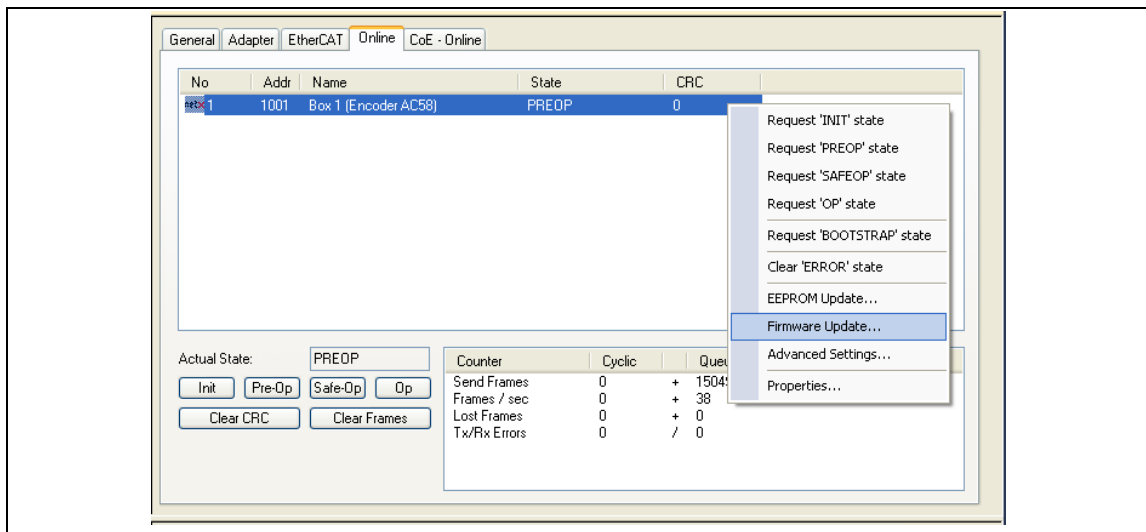
8.2 Update Procedure using TwinCAT®

The firmware update is performed using the FoE capabilities of TwinCAT. Proceed as follows to update the firmware:

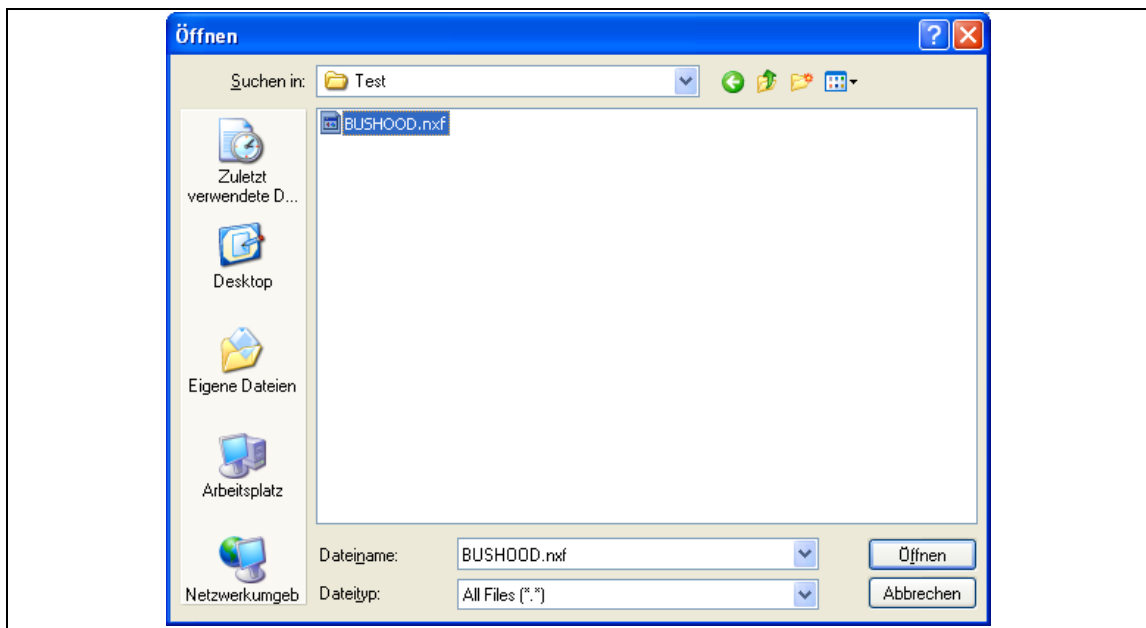
- Scan for devices and boxes in TwinCAT as described in subsection 7.3.5 “Scanning for Devices” and do the following.

NOTE! Do not activate Free Run.

- ↻ Device and box should now be present.
- Select the device within the device tree.
- Open register card *Online*.
- In the displayed list, select the entry for the box (ACURO).
- Select context menu option *Firmware Update* (see below).

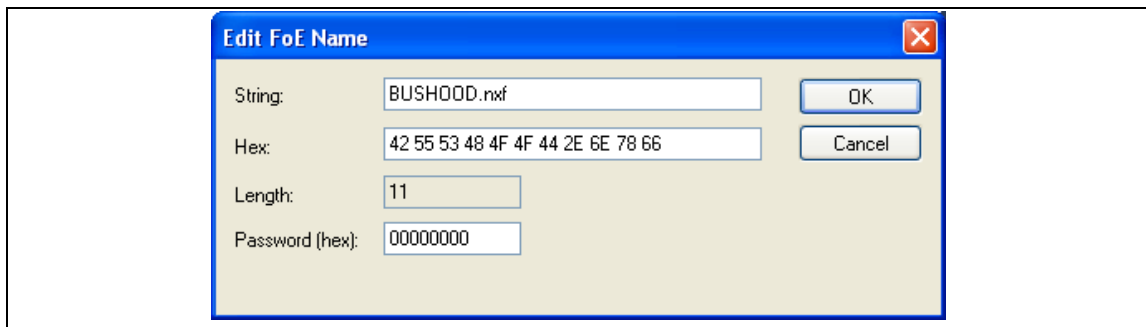


➤ The file selection dialog for the firmware file is displayed. Change file type to "All Files (*.*)"



➤ Select the file containing the firmware (BUSHOOD.nxf).

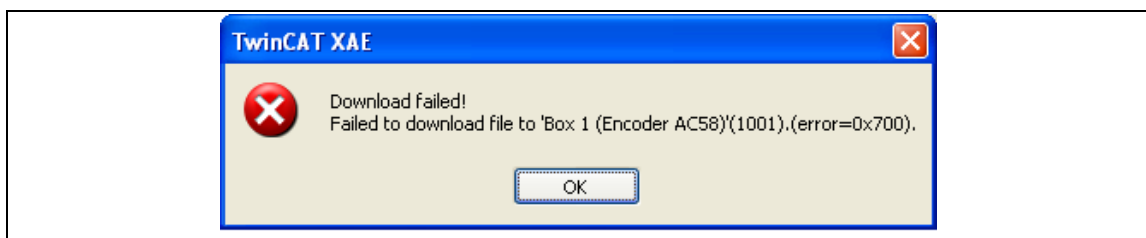
➤ The following dialog is displayed:



- Append the extension `.nxf` to the proposed file name `,BUSHOOD'`.
- Then click on *Ok*.
- ↻ The firmware download is started.
- After the download an error message appears. This message is called by a TwinCAT timeout. The message may be ignored.
- Perform a power-on reset in order to force a restart of the box.
- ↻ The box (shaft encoder) should now be equipped with its new firmware and this firmware should be operational.

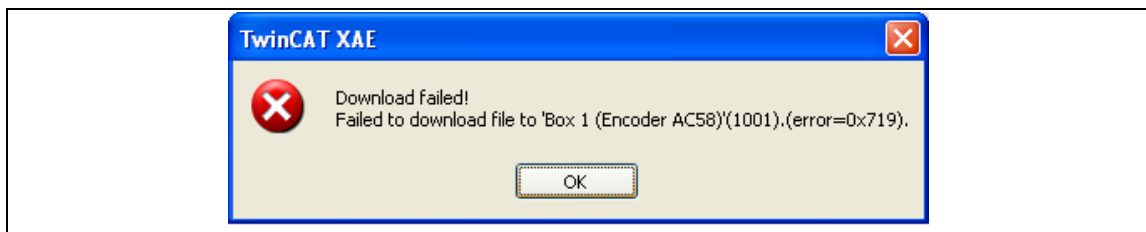
8.3 Common Error Messages during Update Procedure

Error 0x700



If this error occurs when updating the firmware, you forgot to add the extension `*.nxf` to the file name of the firmware file

Error 0x719 Device Timeout



If this error occurs when updating the firmware, you should restart the ACURO EtherCAT by power cycling in order to force it to start up using the new firmware file.

9 Network Performance

The achievable EtherCAT® cycle times are determined by the process data configuration (PDO type) and scaling parameters as well as by the resolution and type of the encoder connected to the EtherCAT bus hood.

The actual Calc & Copy Time and Minimum Cycle Time values are calculated by the bus hood based on the selected PDO and scaling configuration and provided in CoE object 0x1C33 (see section 6.1, “ACURO AC58 EtherCAT® Object Dictionary”).

The following table gives an overview of the minimum cycle times for typical configurations with the following assumptions:

- ▶ An encoder with BiSS-C and 5 MHz clock frequency is used.
- ▶ The total physical resolution of the encoder (ST + MT) does not exceed 32 bits.

PDO Type	Width in Bytes	Configuration	Restriction	Min Cycle Time in μ s
TxPDO1 “Default”	4	Position	Only raw position, no scaling	62.5
			Position scaling with power of 2 for multiplier, divider and modulo (see CoE objects 0x6001, 0x6501 and 0x6002), no residual value calculation	66.6
			Full position scaling with arbitrary values for scaling and modulo and residual value calculation	100
TxPDO2 “Type 1”	5	Position and warning position flags	Raw position, no scaling	62.5
			Position scaling with power of 2 for multiplier, divider and modulo (see CoE objects 0x6001, 0x6501 and 0x6002), no residual value calculation	71.4
			Full position scaling with arbitrary values for scaling and modulo and residual value calculation	100
TxPDO3 “Type 2”	8	Position and speed	Raw position and speed, no scaling	100
			Full position and speed scaling with arbitrary values for scaling and modulo and residual value calculation	125
TxPDO4 “Type 3”	8	Position and system time	Raw position, no scaling	62.5

PDO Type	Width in Bytes	Configuration	Restriction	Min Cycle Time in μs
			Position scaling with power of 2 for multiplier, divider and modulo (see CoE objects 0x6001, 0x6501 and 0x6002), no residual value calculation	71.4
			Full position scaling with arbitrary values for scaling and modulo and residual value calculation	100

Tab. 12: Overview of the minimum Cycle Times for typical Configurations

10 Error Handling and Diagnosis

The ACURO AC58 EtherCAT® supports the following mechanisms for diagnosis and error reporting:

- ▶ EtherCAT system diagnosis via the EtherCAT State Machine.
For more information, see section 5.4.3.1, “The EtherCAT® State Machine (ESM) “
- ▶ Profile-specific objects for alarms and warnings.
For more information, see section 6, “Object Dictionary”.
- ▶ Sending of CoE Emergency Messages (according to the EtherCAT specification, part 6) to signal errors, warnings or diagnoses to the master

For more information, see below.

10.1 CoE Emergency Messages

EMCY Error Code	Error Class	Error Register	Manufacturer specific error code	EMCY source	Meaning
0x1000	Generic Error	Bit 0 “Generic error” set	0xF1, [ERR], 0x00, 0x00, 0x00 [ERR] =contents of the encoder’s error register	Encoder	Error reported by the encoder ASIC (object 0x6503 alarm bit 0 “Position error” set) NOTE! This emergency is thrown once on rising edge of the Error bit in cyclic data from encoder. The error bit in CoE object 0x1001 “Error register” and the emergency are delayed until ERR register has been read from the encoder.
0x1000	Generic Error	(unchanged by the reason of this emergency)	0xF2, [ERR], 0x00, 0x00, 0x00 [ERR] =contents of the encoder’s error register	Encoder	Warning reported by the encoder ASIC (object 0x6503 “Alarms” unchanged) NOTE! This emergency is thrown once on rising edge of the Warning bit in cyclic data from encoder. The error bit in CoE object 0x1001 “Error register” and the emergency are delayed until ERR register has been read from the encoder.
0x1000	Generic Error	Bit 0 “Generic error” set	0x05, 0x00, 0x00, 0x00, 0x00	Bus Hood	Error in communication with encoder This emergency is thrown when in <threshold> BiSS cycles in sequence a communication error occurs. The threshold can be configured via CoE object 0x4402 (see 6)

EMCY Error Code	Error Class	Error Register	Manufacturer specific error code	EMCY source	Meaning
0x5000	Device Hardware	(unchanged by the reason of this emergency)	0x02, 0x00, 0x00, 0x00, 0x00	Bus Hood	Reset performed due to CPU watchdog state (object 0x6505 warning bit 2 “CPU watchdog state reset performed” set)
0x5000	Device Hardware	(unchanged by the reason of this emergency)	0x03, 0x00, 0x00, 0x00, 0x00	Bus Hood	Operating time limit exceeded (object 0x6505 warning bit 3 “operating time” set)
0x8100	Communication	Bit 0 “Generic error” set	0x04, 0x00, 0x00, 0x00, 0x00	Bus Hood	EtherCAT synchronization warning (SM3 has not been accessed by master for 50 times)
0x8100	Communication	Bit 0 “Generic error” set	0x04, 0x00, 0x00, 0x00, 0x00	Bus Hood	EtherCAT synchronization warning (SM3 has not been accessed by master for 50 times)

Tab. 13: CoE emergency messages

- ▶ The above defined CoE emergency messages are sent by the slave when an error/warning occurs.
- ▶ Every time an emergency for a coming error is sent, the information in the pre-defined error field (object 0x1003, see section 6, “Object Dictionary”) is updated. Object 0x1003 offers 2 bytes of “Additional information”. Therefore only the first 2 bytes of the Manufacturer-specific error code are used and also stored in object 0x1003. The first byte indicates the location of error detection:
 - 0xF1 – encoder error
 - 0xF2 – encoder warning
 - 0x01..0x1F – errors detected by bus hood
- ▶ When an error/warning disappears, a CoE emergency message with error code “0x0000” (reset error) is sent by the slave. The value of the error register reflects the status of the device (other errors may still be present). The value of the CoE emergency Manufacturer-specific error code is the same as for the incoming emergency

NOTE! If only warnings are present, this is considered as error-free in the emergency state machine.

10.2 ACURO Specific AL Status Codes

Following ACURO specific AL Status Codes have been defined additionally:

ACURO specific AL Status Codes supported by the EtherCAT Slave Stack		
Value	Meaning	Description
0x8100	WRONG_HW	The current firmware version mismatches the bus hood hardware revision (e.g. firmware version V1.2.0.1 requires hardware revision 3).
0x8101	PARAM_CORRUPT	The FLASH storage area which contains all operating parameters is corrupted.
0x8102	RESID_CORRUPT	The FRAM storage area which contains residual value related data is corrupted.
0x8103	RESID_OVERRUN	The mechanism for storage of residual value related data to FRAM has detected multiple pending residual value storage requests.
0x8104	ENCODER_COMM_ERR	Communication between bus hood and encoder module failed.
0x8105	ENCODER_POS_ERR	The encoder module informed about a possible position fault.

Tab. 14: ACURO Specific AL Status Codes


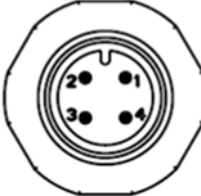

11 Technical Data

11.1 Bus/Power Connections

The Hengstler ACURO AC58 EtherCAT® absolute encoder supports the 100BASE-TX variant of the IEEE 802.3 standard. 100BASE-TX uses shielded twisted-pair copper cables with two pairs of wires. Cables of categories CAT 5, 6 or 7 can be used. M12 connectors are used, in keeping with the industrial applications in which the encoder will be used, and due to the connectors' excellent environmental protection provided by this connector type.

The maximum distance between two nodes is limited to 100 meters.

Connections are made via a bus hood with three (3) M12 connectors. The connectors have the following pinouts.

Pin	Connector		
	Bus Port IN	Power (Supply Voltage)	Bus Port OUT
1	TxD+	UB in	TxD+
2	RxD+	N.C.	RxD+
3	TxD-	0 V in	TxD-
4	RxD-	N.C.	RxD-
Shield	Shield ¹⁾	Shield ¹⁾	Shield ¹⁾
	 <p>M12 connector D-coded</p>	 <p>M12 connector A-coded</p>	 <p>M12 connector D-coded</p>
¹⁾ shield connected to encoder housing			

Tab. 15: Connector Pinouts

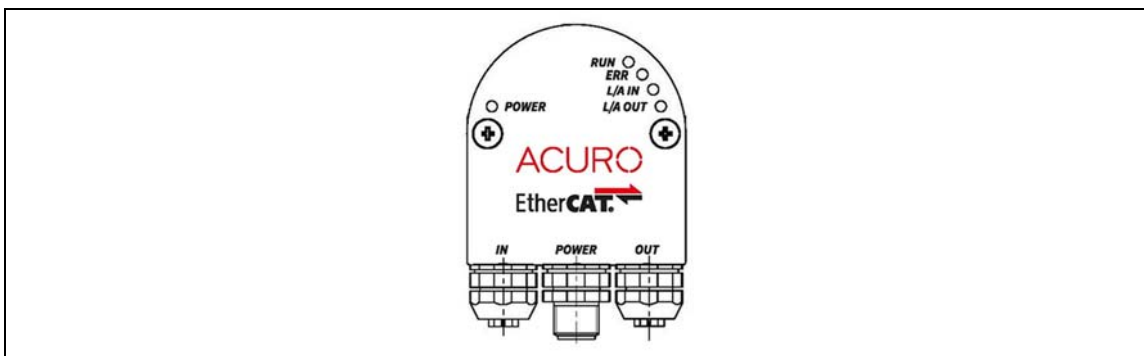


Fig. 18: ACURO AC58 EtherCAT LED/Connector Layout

For information on interpretation of the LEDs on the encoder, please see the section 4.3, “LED Interpretation” in this manual.

11.2 EtherCAT®

ACURO AC58 EtherCAT® has the following supported functions and shown data sizes.

Features	Parameter
Number of total cyclic input data	Depends on selection of PDO type
Default PDO	4 bytes
PDO Type 1	5 bytes
PDO Type 2	8 bytes
PDO Type 3	8 bytes
Number of total cyclic output data	None.
Acyclic Communication	SDO SDO Master-Slave
Slave Type	complex slave
SYNC0 signal	supported
Distributed Clocks (DC)	supported, 32 bit
CoE Emergency	supported
FoE	supported, used for firmware update
Baud rate	100 MBit/s (fixed)
Data transport layer	Ethernet II, IEEE 802.3

Tab. 16: EtherCAT Technical Data

Limitations

- ▶ The standard profile object 0x6002 “*Total measuring range in measuring units*” limits the total resolution (ST * MT) to < 32 bits due to its data type UNSIGNED32.

12 Appendix

12.1 EtherCAT® commands

The following EtherCAT® commands are defined within the EtherCAT specification:

Command code	Command
APRD	Auto increment physical read
APWR	Auto increment physical write
APRW	Auto increment physical read write
FPRD	Configured address physical read
FPWR	Configured address physical write
FPRW	Configured address physical read write
BRD	Broadcast read
BWR	Broadcast write
BRW	Broadcast read write
LRD	Logical read
LWR	Logical write
LRW	Logical read write
ARMW	Auto increment physical read multiple write
FRMW	Configured address physical read multiple write
NOP	No operation

Tab. 17: EtherCAT Command Codes

These commands differ in the addressing method used:

- ▶ Auto increment access (APRD, APWR, APRW and ARMW) uses an automatically incremented position located in the first 16 bits of the command address and a local memory address as offset (last 16 bits of command address).
- ▶ (Fixed) configured address access (FPRD, FPWR, FPRW, FRMW) uses a configured address and a local memory address as offset located in the last 16 bits of command address.
- ▶ Broadcast access (BRD, BWR and BRW) uses a position (first 16 bits of command address) and a local memory address as offset located in the last 16 bits of command address. The position is incremented by each slave at BRD.
- ▶ Logical access (LRD, LWR and LRW) uses 32 bit logical addresses.

12.2 CoE Object 2002:1

EtherCAT® manages cyclic and acyclic communication in a manner very similar to CANopen. This is accomplished via the mailbox protocol CoE (CANopen over EtherCAT). CoE differs only by a few adaptations and extensions from the CANopen standard. In fact, it capsules the CANopen protocol, i.e. the data are transmitted within standard CANopen frames with an additional mailbox header and CoE command header preceding the CANopen frame.

The following figures and tables illustrate the structure of the mailbox header (see Fig. 19), the CoE command header (see Fig. 20) and the CANopen frame itself (see Fig. 21):

Mailbox header (6 bytes)

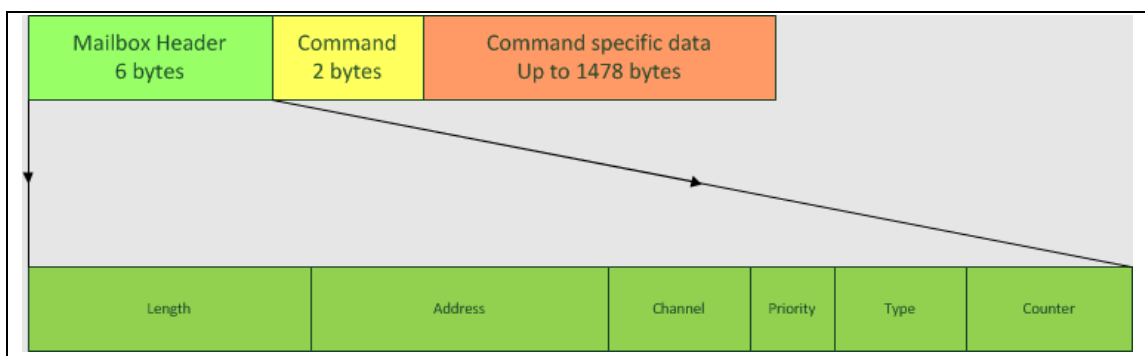


Fig. 19: Mailbox header

Field name	Width	Meaning
Length	16 bits	Length of following data
Address	16 bits	Station address of originator
Channel	6 bits	reserved
Priority	2 bits	reserved
Type	4 bits	Mailbox type, see table below
Count	3 bits	Sequence number (1...7) to be used for duplicate detection

Tab. 18: Contents of mailbox header

The following table explains the meaning of the mailbox type:

Value	Meaning	
0	Mailbox error	
2	EoE	Ethernet over EtherCAT
3	CoE	CANopen application protocol over EtherCAT
4	FoE	File access over EtherCAT
5	SoE	Servo drive over EtherCAT
15	VoE	Vendor specific profile over EtherCAT

Tab. 19: Meaning of mailbox type

CoE command header (2 bytes)

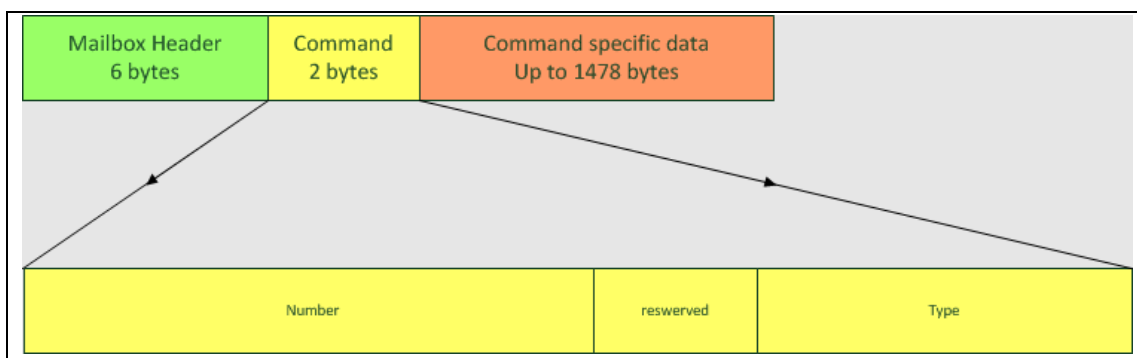


Fig. 20: CoE command header

Field name	Width	Meaning
Number	9 bits	PDO number (in case of PDO transmission)
Reserved	3 bits	Reserved
Type	4 bits	Message type see table below

Tab. 20: Contents of Mailbox Header

The following table explains the meaning of the mailbox type:

Value	Meaning
0	Reserved
1	Emergency message
2	SDO Request
3	SDO Response
4	TxPDO
5	RxPDO
6	Remote transmission request of TxPDO
7	Remote transmission request of RxPDO
8	SDO Information
9-15	Reserved for future use

Tab. 21: Meaning of mailbox type

CoE frame (8 ... 1478 bytes)

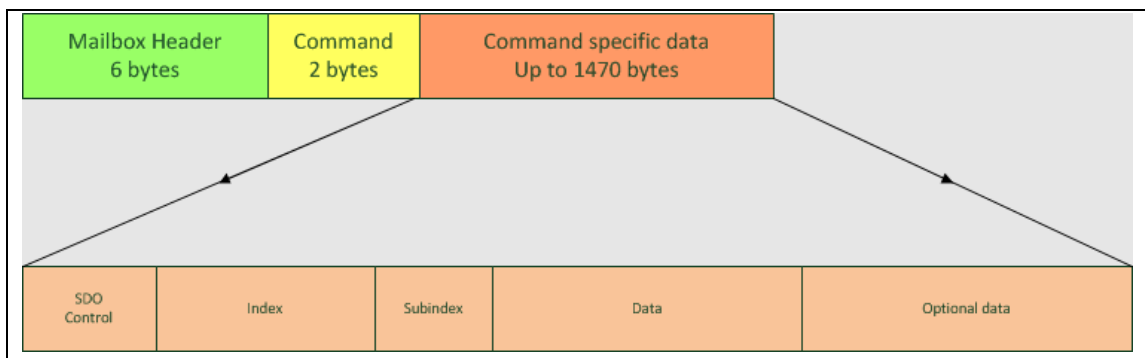


Fig. 21: CoE frame

Field name	Width	Meaning
SDO Control	8 bits	Standard CANopen SDO services
Index	16 bits	Index for object addressing within the object dictionary (not to be confused with the index in section 5.4.2.1)
Subindex	8 bits	Subindex for object addressing within the object dictionary
Data	32 bits	Data for SDO service
Optional data	0 ... 1470 bytes	Full mailbox size usable for further data if required

Tab. 22: Contents of CoE frame