

Documentation | EN

# EtherCAT

System Documentation



Ether**CAT**®



# Table of contents

<b>1</b>	<b>Foreword</b> .....	<b>7</b>
1.1	Notes on the documentation.....	7
1.2	Safety instructions .....	8
1.3	Documentation issue status .....	9
<b>2</b>	<b>EtherCAT basics</b> .....	<b>10</b>
2.1	System overview .....	10
2.1.1	Limits of existing Ethernet communication approaches.....	10
2.1.2	New approach.....	10
2.1.3	System properties.....	12
2.1.4	Application areas .....	16
2.1.5	EtherCAT is open .....	18
2.1.6	Summary and Outlook .....	19
2.2	Basic principles.....	19
2.2.1	EtherCAT State Machine .....	19
2.2.2	CoE interface .....	21
2.3	System features.....	30
2.3.1	EtherCAT cable redundancy.....	30
2.3.2	HotConnect.....	38
2.3.3	EtherCAT data exchange .....	59
2.3.4	Safety over EtherCAT - TwinSAFE.....	61
<b>3</b>	<b>Setup in the TwinCAT System Manager</b> .....	<b>63</b>
3.1	TwinCAT Quick Start .....	63
3.1.1	TwinCAT 2 .....	66
3.1.2	TwinCAT 3 .....	76
3.2	EtherCAT master in TwinCAT .....	89
3.3	General notes on the use of Beckhoff EtherCAT IO components .....	94
3.3.1	Technical classification .....	94
3.3.2	Change management by Beckhoff .....	97
3.3.3	Procurement/update of the elements.....	98
3.3.4	Application project planning in the TwinCAT System Manager 2.x/3.x - creating the configuration .....	99
3.3.5	Configuration comparison in the project .....	100
3.3.6	Determining the installation version in the application.....	101
3.4	Version identification of EtherCAT devices .....	102
3.4.1	Beckhoff Identification Code (BIC).....	106
3.5	Online version identification of EtherCAT devices.....	107
3.6	TwinCAT Development Environment .....	111
3.6.1	Installation of the TwinCAT real-time driver .....	111
3.6.2	Notes regarding ESI device description.....	117
3.6.3	TwinCAT ESI Updater .....	121
3.6.4	Distinction between Online and Offline.....	121
3.6.5	OFFLINE configuration creation .....	122
3.6.6	ONLINE configuration creation.....	127
3.6.7	Standard behavior of the EtherCAT master.....	135

3.7	Notes on distributed clocks.....	147
3.7.1	EtherCAT Distributed Clocks - default settings.....	147
3.7.2	EtherCAT Distributed Clocks - coupling of EtherCAT systems .....	154
<b>4</b>	<b>EtherCAT diagnostics .....</b>	<b>158</b>
4.1	General Notes - EtherCAT Slave Application.....	158
4.2	EtherCAT AL Status Codes.....	166
<b>5</b>	<b>EtherCAT operation - control .....</b>	<b>188</b>
5.1	Operation.....	188
<b>6</b>	<b>EtherCAT operation - timing.....</b>	<b>189</b>
6.1	Concept mapping .....	189
6.2	Mapping Types and Graphical Display.....	190
6.2.1	Mappings .....	190
6.2.2	Context Menu .....	192
6.2.3	Tab "A -> B" or "B -> A" .....	192
6.2.4	Online tab .....	193
<b>7</b>	<b>Distributed Clocks.....</b>	<b>195</b>
7.1	TwinCAT & time.....	195
7.1.1	TwinCAT time sources.....	195
7.1.2	Internal and external EtherCAT synchronization .....	197
7.1.3	Sample programs .....	203
7.2	Basic principles.....	205
7.2.1	EtherCAT Distributed Clocks .....	205
7.2.2	Distributed Clocks settings in the Beckhoff TwinCAT System Manager (2.10) .....	214
7.2.3	Distributed Clocks settings in the Beckhoff TwinCAT System Manager (2.11) .....	220
7.2.4	Distributed Clocks & TwinCAT PLC.....	229
7.2.5	Synchronization modes of an EtherCAT slave .....	233
7.2.6	EKxxxx - Optional Distributed Clocks support .....	248
7.3	External synchronization .....	250
7.3.1	Basic principles.....	250
7.3.2	Sample: EL6692 bridge terminal .....	253
<b>8</b>	<b>Configuration of the terminals .....</b>	<b>261</b>
8.1	General configuration .....	261
8.1.1	EtherCAT subscriber configuration.....	261
8.1.2	Fast mode.....	270
8.2	Advanced configuration .....	272
8.2.1	Behavior.....	272
8.2.2	Timeout Settings.....	277
8.2.3	FMMU / SM.....	278
8.2.4	Mailbox .....	279
8.2.5	Smart View .....	281
8.2.6	Memory.....	283
8.3	Firmware Update EL/ES/EM/ELM/EPxxxx .....	284
8.3.1	Device description ESI file/XML.....	285
8.3.2	Firmware explanation .....	288
8.3.3	Updating controller firmware *.efw.....	289

8.3.4	FPGA firmware *.rbf.....	290
8.3.5	Simultaneous updating of several EtherCAT devices.....	294
<b>9</b>	<b>FAQ .....</b>	<b>295</b>
9.1	Windows Memory Dump.....	295
9.2	MessageBox in the target system .....	299
<b>10</b>	<b>Appendix .....</b>	<b>301</b>
10.1	UL notice .....	301
10.2	Training.....	302
10.3	Support and Service .....	302



# 1 Foreword

## 1.1 Notes on the documentation

### Intended audience

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning these components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

### Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

### Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

### Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents: EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702 with corresponding applications or registrations in various other countries.



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

### Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

## 1.2 Safety instructions

### Safety regulations

Please note the following safety instructions and explanations!  
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

### Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

### Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

### Description of instructions

In this documentation the following instructions are used.  
These instructions must be read carefully and followed without fail!

#### **DANGER**

##### **Serious risk of injury!**

Failure to follow this safety instruction directly endangers the life and health of persons.

#### **WARNING**

##### **Risk of injury!**

Failure to follow this safety instruction endangers the life and health of persons.

#### **CAUTION**

##### **Personal injuries!**

Failure to follow this safety instruction can lead to injuries to persons.

#### **NOTE**

##### **Damage to environment/equipment or data loss**

Failure to follow this instruction can lead to environmental damage, equipment damage or data loss.



##### **Tip or pointer**

This symbol indicates information that contributes to better understanding.



## 1.3 Documentation issue status

Version	Comment
5.5	<ul style="list-style-type: none"> <li>• Addenda chapter "Displaying channels in InfoData"</li> <li>• Structural update</li> </ul>
5.4	<ul style="list-style-type: none"> <li>• Update chapter "Setup in the TwinCAT System Manager"</li> <li>• Structural update</li> </ul>
5.3	<ul style="list-style-type: none"> <li>• Addendum sample program</li> <li>• Structural update</li> </ul>
5.2	<ul style="list-style-type: none"> <li>• Update chapter „Distributed Clocks“, "TwinCAT &amp; time"</li> </ul>
5.1	<ul style="list-style-type: none"> <li>• Addenda chapter „TwinCAT Quick Start“</li> </ul>
5.0	<ul style="list-style-type: none"> <li>• Migration</li> <li>• Structural update</li> </ul>
4.4	<ul style="list-style-type: none"> <li>• Update section "Internal and external EtherCAT synchronization"</li> </ul>
4.3	<ul style="list-style-type: none"> <li>• Adjustment of CurTaskTime</li> </ul>
4.2	<ul style="list-style-type: none"> <li>• Update folder structure</li> </ul>
4.1	<ul style="list-style-type: none"> <li>• Addenda section "Notes on ESI device description"</li> <li>• Section "EtherCAT AL Status Codes" added</li> </ul>
4.0	<ul style="list-style-type: none"> <li>• Addenda section "Setup in TwinCAT System Manager"</li> </ul>
3.3	<ul style="list-style-type: none"> <li>• Addendum section "Online version identification of EtherCAT devices"</li> </ul>
3.2	<ul style="list-style-type: none"> <li>• Addendum section "Hot Connect, Fast Hot Connect"</li> </ul>
3.1	<ul style="list-style-type: none"> <li>• Addendum section "Online configuration creation 'Scanning'"</li> </ul>
3.0	<ul style="list-style-type: none"> <li>• Addendum section "EtherCAT diagnostics"</li> </ul>
2.9	<ul style="list-style-type: none"> <li>• Addendum sample program</li> </ul>
2.8	<ul style="list-style-type: none"> <li>• Addendum section "CoE interface – parameter management in EtherCAT system"</li> </ul>
2.7	<ul style="list-style-type: none"> <li>• Addendum section "Identification of the current connection point"</li> </ul>
2.6	<ul style="list-style-type: none"> <li>• Addendum note on logger window</li> </ul>
2.5	<ul style="list-style-type: none"> <li>• Addendum section "EtherCAT PDO settings"</li> </ul>
2.4	<ul style="list-style-type: none"> <li>• Addendum section "EtherCAT"</li> </ul>
2.3	<ul style="list-style-type: none"> <li>• Addendum section "Cable redundancy"</li> </ul>
2.2	<ul style="list-style-type: none"> <li>• Addendum section "EtherCAT data exchange"</li> </ul>
2.1	<ul style="list-style-type: none"> <li>• Addendum section "Bus coupler DC support"</li> </ul>
2.0	<ul style="list-style-type: none"> <li>• Addendum UL note</li> <li>• Addendum sample</li> </ul>
1.9	<ul style="list-style-type: none"> <li>• "Firmware" section updated</li> <li>• Addendum sample</li> </ul>
1.8	<ul style="list-style-type: none"> <li>• Addendum section "Cable redundancy"</li> </ul>
1.7	<ul style="list-style-type: none"> <li>• Addendum CoE principles</li> </ul>
1.6	<ul style="list-style-type: none"> <li>• Correction TwinCAT times</li> </ul>
1.5	<ul style="list-style-type: none"> <li>• Addendum samples</li> </ul>
1.4	<ul style="list-style-type: none"> <li>• Correction TwinCAT times</li> </ul>
1.3	<ul style="list-style-type: none"> <li>• TwinCAT 2.11 description added</li> </ul>
1.2	<ul style="list-style-type: none"> <li>• HotConnect description added</li> <li>• Addendum FastMode</li> </ul>
1.1	<ul style="list-style-type: none"> <li>• Master cable redundancy description added</li> </ul>
1.0	<ul style="list-style-type: none"> <li>• Corrections, 1st public issue</li> </ul>

## 2 EtherCAT basics

### 2.1 System overview

#### 2.1.1 Limits of existing Ethernet communication approaches

There are many different approaches that try and provide real-time capability for Ethernet: for example, the CSMA/CD access procedure is disabled via higher-level protocol layers and replaced by the time slices procedure or polling; other propositions use special switches that distribute Ethernet packets in a precisely controlled timely manner. Whilst these solutions may be able to transport data packets more or less quickly and accurately to the connected Ethernet nodes, the times required for the redirection to the outputs or drive controllers and for reading the input data strongly depend on the implementation. A sub-bus is usually also required, particularly in modular I/O systems, which, like the Beckhoff K-Bus, can be synchronized and fast, but nevertheless always adds small delays to the communication that cannot be avoided.

If individual Ethernet frames are used for each device, the usable data rate is very low in principle: The shortest Ethernet frame is 84 bytes long (incl. inter-packet gap IPG). If, for example, a drive cyclically sends 4 bytes of actual value and state information and accordingly receives 4 bytes of set value and control word information, at 100% bus load (i.e. with infinitely short response time of the drive) a usable data rate of only  $4/84 = 4.7\%$  is achieved. At an average response time of  $10 \mu\text{s}$ , the rate drops to 1.9%. These limitations apply to all real-time Ethernet approaches that send an Ethernet frame to each device (or expect a frame from each device), irrespective of the protocols used within the Ethernet frame.

#### 2.1.2 New approach

EtherCAT technology overcomes these inherent limitations of other Ethernet solutions: the Ethernet packet is no longer received, then interpreted and copied as process data at every connection. The newly developed FMMU (fieldbus memory management unit) in each I/O terminal reads the data addressed to it, whilst the telegram continues through the device. Similarly, input data are inserted while the telegram passes through. The telegrams are only delayed by a few nanoseconds.

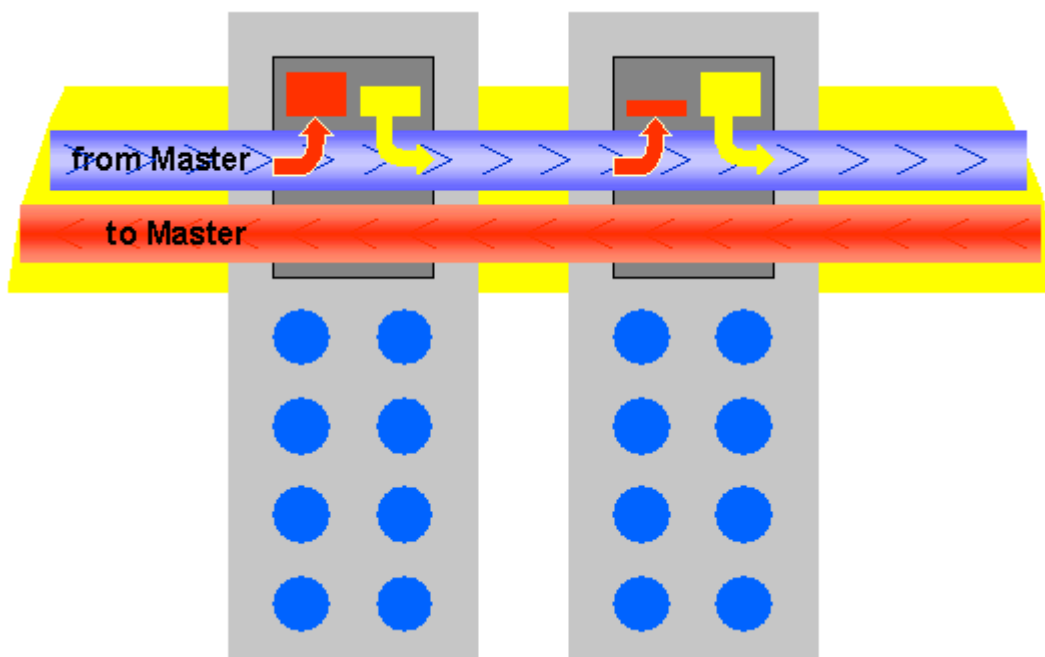


Fig. 1: Dynamic telegram processing

Since an Ethernet frame reaches the data of many devices both in send and receive direction, the usable data rate increases to approx. 80%. The full-duplex features of 100BaseTx are fully utilized, so that effective data rates of  $> 100 \text{ Mbit/s}$  (up to 80% of  $2 \times 100 \text{ Mbit/s}$ ) can be achieved.

## Ethernet up to the terminal

The Ethernet protocol conforms to IEEE 802.3 remains intact right up to the individual terminals; no sub-bus is required. Only the transfer physics is converted in the coupler from twisted pair or optical fiber physics to E-bus, in order to meet the requirements of the electronic terminal block.

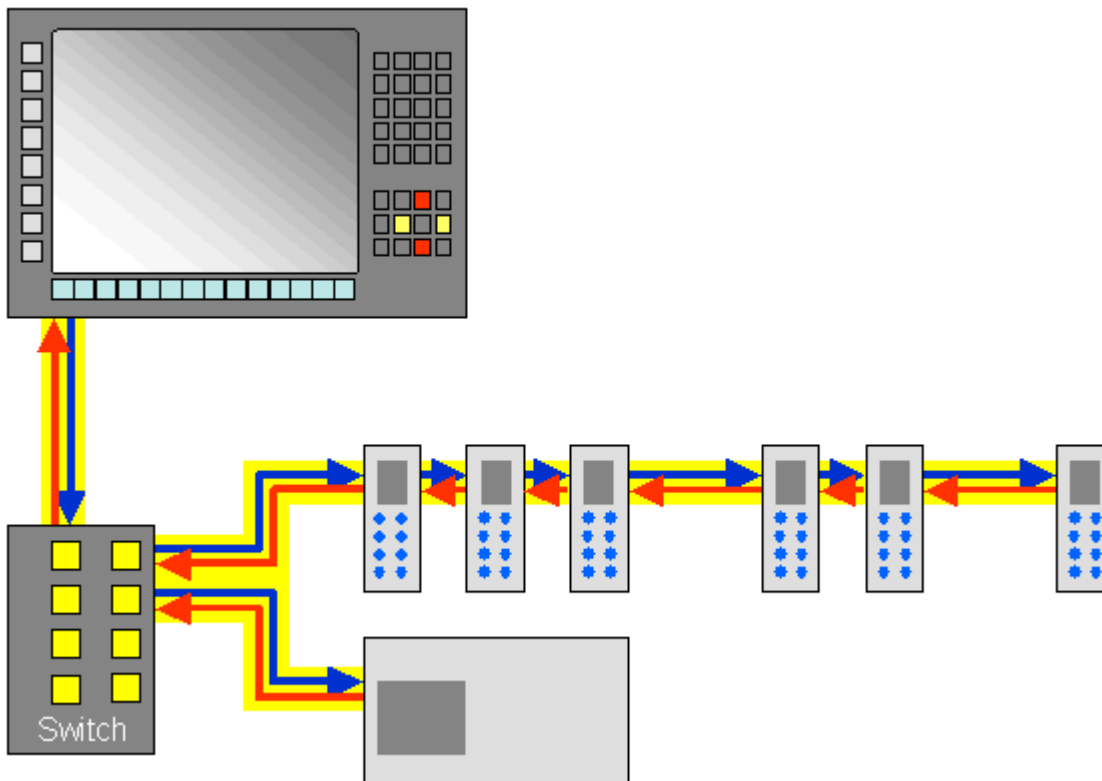


Fig. 2: Full duplex Ethernet in the ring, one frame for many devices. The EtherCAT system architecture increases the communication efficiency.

On the control side, the TwinCAT Y driver for Ethernet complements the FMMU technology. This integrates transparently into the system, so that it appears as an operating system-compatible network driver, and additionally as a TwinCAT fieldbus card. An internal prioritization and buffer is provided at the transmitter end which always finds a free transmission channel for Ethernet frames from the real-time system that may be queuing. The operating system's Ethernet frames are only a later transmitted in the gaps if sufficient time is available.

At the receiving end, all the Ethernet frames received are examined by the TwinCAT I/O system, and those with real-time relevance are filtered out. All other frames are passed on to the operating system after examination, outside the context of the real-time system.

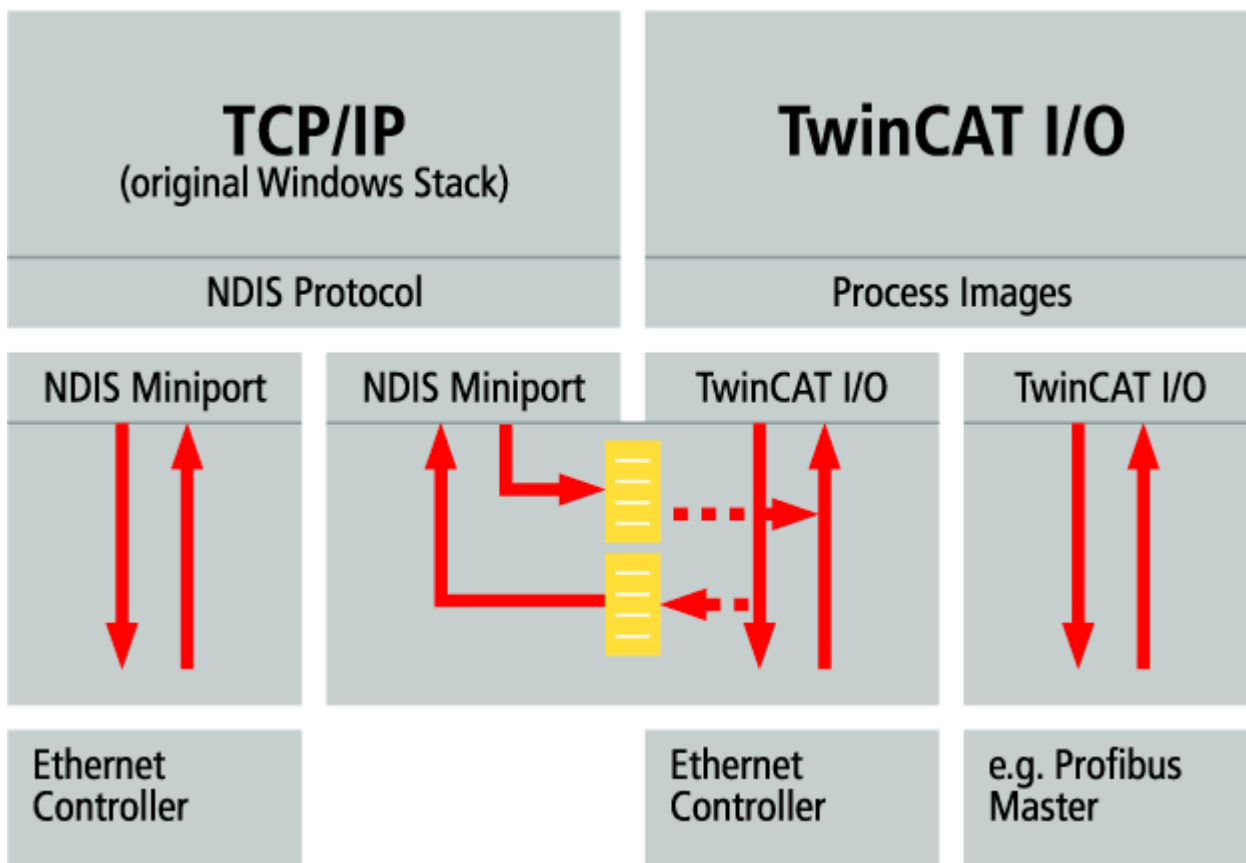


Fig. 3: Operating system-compatible TwinCAT Y driver:

Very inexpensive, commercially available standard network interface cards (NIC) are used as hardware in the controller. The data transfer to the PC takes place via DMA (direct memory access). Therefore, no CPU performance is lost for network access.

Since the Ethernet functionality of the operating system is fully maintained, all operating system-compatible protocols can be operated in parallel on the same physical network. This not only includes standard IT-protocols such as TCP/IP, HTTP, FTP or SOAP, but also practically all Industrial-Ethernet-protocols such as Modbus TCP, ProfiNet or Ethernet/IP.

### 2.1.3 System properties

#### Protocol

The EtherCAT protocol is optimized for process data and is transported directly within the Ethernet frame thanks to a special Ether-type. It may consist of several sub-telegrams, each serving a particular memory area of the logical process images that can be up to 4 gigabytes in size. The data sequence is independent of the physical order of the Ethernet terminals in the network; addressing can be in any order. Broadcast, Multicast and communication between slaves are possible. Transfer directly in the Ethernet frame is used in cases where EtherCAT components are operated in the same subnet as the control computer.

However, EtherCAT applications are not limited to a subnet: EtherCAT UDP packs the EtherCAT protocol into UDP/IP datagrams. This enables any control with Ethernet protocol stack to address EtherCAT systems. Even communication across routers into other subnets is possible. In this variant, system performance obviously depends on the real-time characteristics of the control and its Ethernet protocol implementation. The response times of the EtherCAT network itself are hardly restricted at all: the UDP datagram only has to be unpacked in the first station.

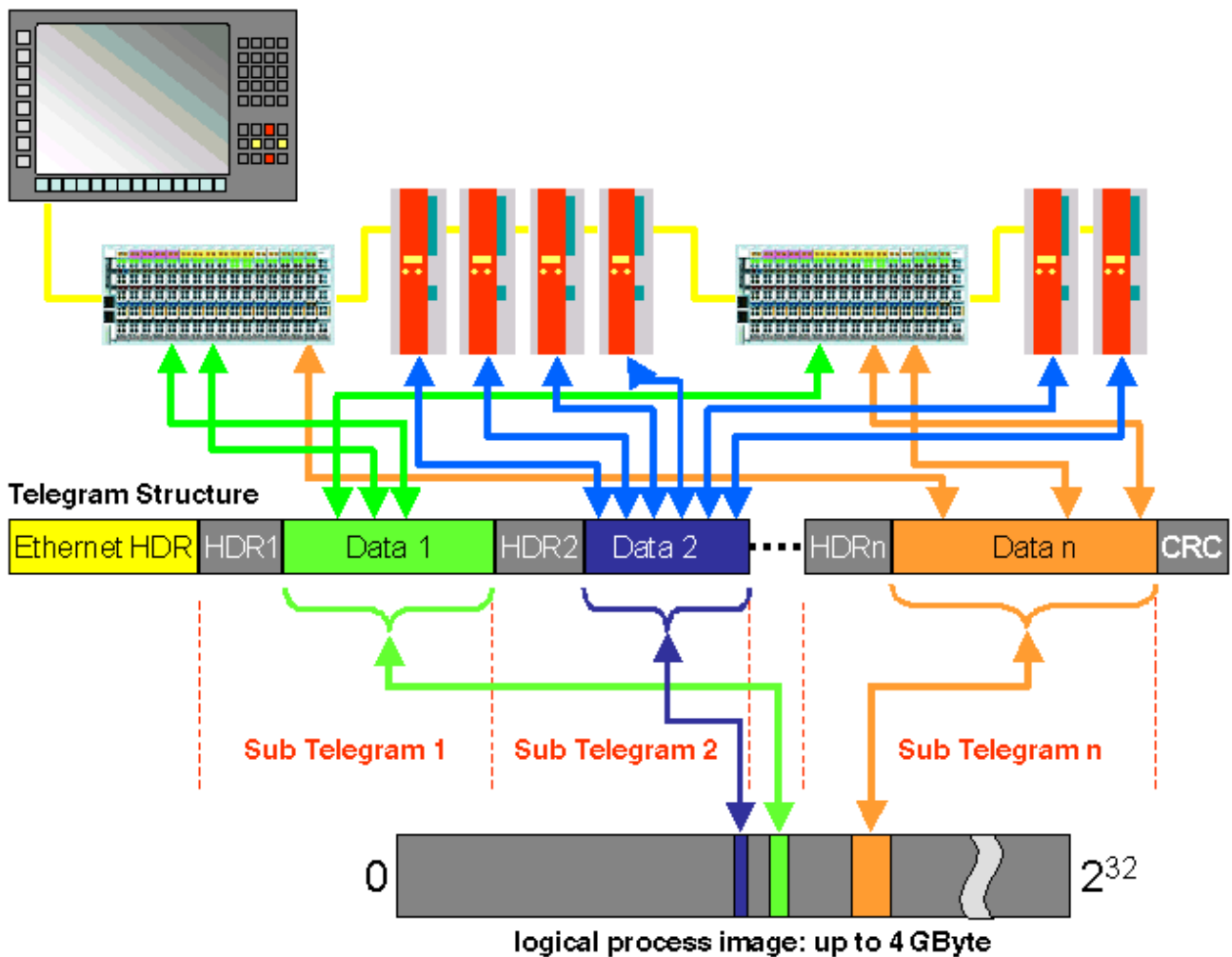


Fig. 4: EtherCAT protocol structure

Protocol structure: The process image allocation is freely configurable. Data are copied directly in the I/O terminal to the desired location within the process image: no additional mapping is required. The available logical address space is with very large (4 GB).

**Topology**

Line, tree or star: EtherCAT supports almost any topology. The bus or line structure known from the fieldbuses thus also becomes available for Ethernet. Particularly useful for system wiring is the combination of line and junctions or stubs. The required interfaces exist on the couplers; no additional switches are required. Naturally, the classic switch-based Ethernet star topology can also be used.

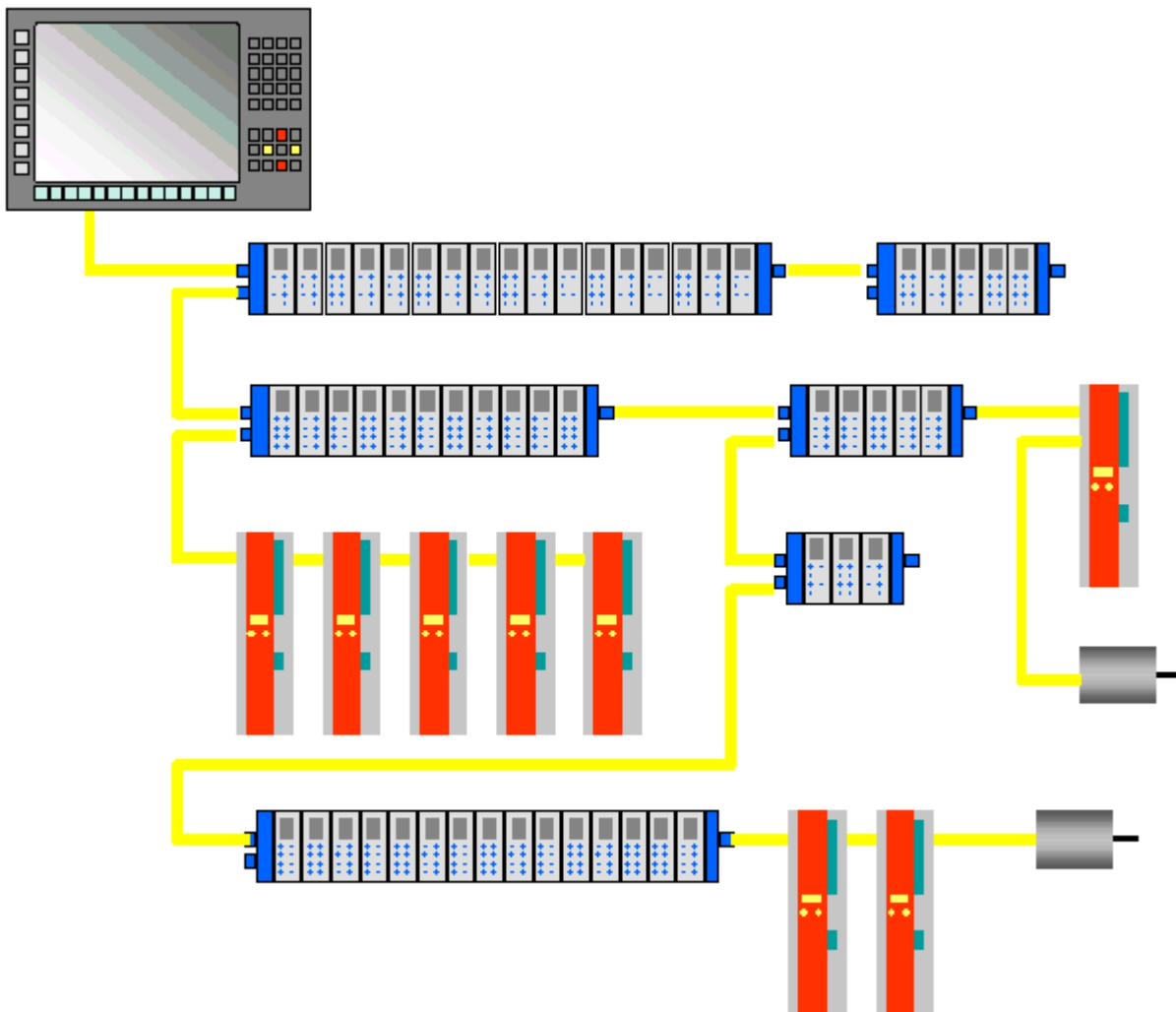


Fig. 5: Sample: Ethernet topology

Maximum wiring flexibility:

with or without switch, line or tree topologies, can be freely selected and combined.

Wiring flexibility is further maximized through the choice of different cables. Flexible and cost-effective standard Ethernet patch cables transfer the signals in Ethernet mode (100Base-TX). The complete bandwidth of the Ethernet network - such as different optical fibers and copper cables - can be used in combination with switches or media converters.

### Distributed Clocks

Accurate synchronization is particularly important in cases where spatially distributed processes require simultaneous actions. This may be the case, for example, in applications where several servo axes carry out coordinated movements simultaneously.

The most powerful approach for synchronization is the accurate alignment of distributed clocks, as described in the new IEEE 1588 standard. In contrast to fully synchronous communication, where synchronization quality suffers immediately in the event of a communication fault, distributed aligned clocks have a high degree of tolerance vis-à-vis possible fault-related delays within the communication system.

With EtherCAT, the data exchange is fully based on a pure hardware machine. Since the communication utilizes a logical (and thanks to full-duplex Fast Ethernet also physical) ring structure, the mother clock can determine the run-time offset to the individual daughter clocks simply and accurately - and vice versa. The distributed clocks are adjusted based on this value, which means that a very precise network-wide time base with a jitter of significantly less than 1 microsecond is available.

However, high-resolution distributed clocks are not only used for synchronization, but can also provide accurate information about the local timing of the data acquisition. For example, controls frequently calculate velocities from sequentially measured positions. Particularly with very short sampling times, even a small temporal jitter in the displacement measurement leads to large step changes in velocity. With EtherCAT new, extended data types are introduced as a logical extension (time stamp and oversampling data type). The local time is linked to the measured value with a resolution of up to 10 ns, which is made possible by the large bandwidth offered by Ethernet. The accuracy of a velocity calculation then no longer depends on the jitter of the communication system. It is orders of magnitude better than that of measuring techniques based on jitter-free communication.

## Performance

EtherCAT reaches new dimensions in network performance. Protocol processing is purely hardware-based through an FMMU chip in the terminal and DMA access to the network card of the master. It is thus independent of protocol stack run-times, CPU performance and software implementation. The update time for 1000 I/Os is only 30  $\mu$ s - including terminal cycle time. Up to 1486 bytes of process data can be exchanged with a single Ethernet frame - this is equivalent to almost 12000 digital inputs and outputs. The transfer of this data quantity only takes 300  $\mu$ s.

The communication with 100 servo axes only takes 100  $\mu$ s. During this time, all axes are provided with set values and control data and report their actual position and status. Distributed clocks enable the axes to be synchronized with a deviation of significantly less than 1 microsecond.

The extremely high performance of the EtherCAT technology enables control concepts that could not be realized with classic fieldbus systems. For example, the Ethernet system can now not only deal with velocity control, but also with the current control of distributed drives. The tremendous bandwidth enables status information to be transferred with each data item. With EtherCAT, a communication technology is available that matches the superior computing power of modern Industrial PCs. The bus system is no longer the bottleneck of the control concept. Distributed I/Os are recorded faster than is possible with most local I/O interfaces. The EtherCAT technology principle is scalable and not bound to the baud rate of 100 Mbaud – extension to Gbit Ethernet is possible.

## Diagnostics

Experience with fieldbus systems shows that availability and commissioning times crucially depend on the diagnostic capability. Only faults that are detected quickly and accurately and which can be precisely located can be corrected quickly. Therefore, special attention was paid to exemplary diagnostic features during the development of EtherCAT.

During commissioning, the actual configuration of the I/O terminals should be checked for consistency with the specified configuration. The topology should also match the saved configuration. Due to the built-in topology recognition down to the individual terminals, this verification can not only take place during system start-up, automatic reading in of the network is also possible (configuration upload).

Bit faults during the transfer are reliably detected through evaluation of the CRC checksum: The 32 bit CRC polynomial has a minimum hamming distance of 4. Apart from breaking point detection and localization, the protocol, physical transfer behavior and topology of the EtherCAT system enable individual quality monitoring of each individual transmission segment. The automatic evaluation of the associated error counters enables precise localization of critical network sections. Gradual or changing sources of error such as EMC influences, defective push-in connectors or cable damage are detected and located, even if they do not yet overstrain the self-healing capacity of the network.

## Integration of standard Bus Terminals from Beckhoff

In addition to the new Bus Terminals with E-Bus connection (ELxxxx), all Bus Terminals from the familiar standard range with K-bus connection (KLxxxx) can be connected via the BK1120 or BK1250 Bus Coupler. This ensures compatibility and continuity with the existing Beckhoff Bus Terminal systems. Existing investments are protected.

## 2.1.4 Application areas

### Control loops via the bus

In conjunction with fast machine control systems the outstanding performance of EtherCAT can be demonstrated particularly well. The bus system offers access speeds similar to local I/Os. EtherCAT enables not only the position control loop, but also the velocity control loop or even the current control loop to be closed via the bus, resulting in cost-effective drive controllers. EtherCAT therefore takes full advantage of the performance of fast PC-based control technology. The IPCs become small and economical, since slots are avoided: Expansion cards are also controlled via EtherCAT.

EtherCAT is not only suitable for very fast applications. Since all protocol processing takes place in hardware, the technology only places low demands on the master connection: a commercially available Ethernet Controller is sufficient. The master only has to send EtherCAT frames cyclically or as required – mapping, addressing, node monitoring etc. takes place in the slave ASICs. The input process image arrives at the master fully sorted and therefore ready prepared for the control application. For simple cyclic applications with a process image size of up to 1486 bytes, only one telegram has to be processed in the master, consisting almost exclusively of the process data. Larger process images are distributed over several frames.

EtherCAT may therefore be the fieldbus technology placing the lowest demands on the master.

### EtherCAT instead of PCI

With increasing miniaturization of the PC-components, the physical size of Industrial PCs is increasingly determined by the number of required slots. The bandwidth of Fast Ethernet, together with the data width of the EtherCAT communication hardware (FMMU chip) enables new directions: Interfaces that are conventionally located in the IPC are transferred to intelligent interface terminals at the EtherCAT. Apart from the decentralized I/Os, axes and operating devices, complex systems such as fieldbus masters, fast serial interfaces, gateways and other communication interfaces can be addressed. Even further Ethernet devices without restriction on protocol variants can be connected via decentralized switch port terminals. The central IPC becomes smaller and therefore cost-effective. An Ethernet interface is sufficient for complete communication with the periphery.

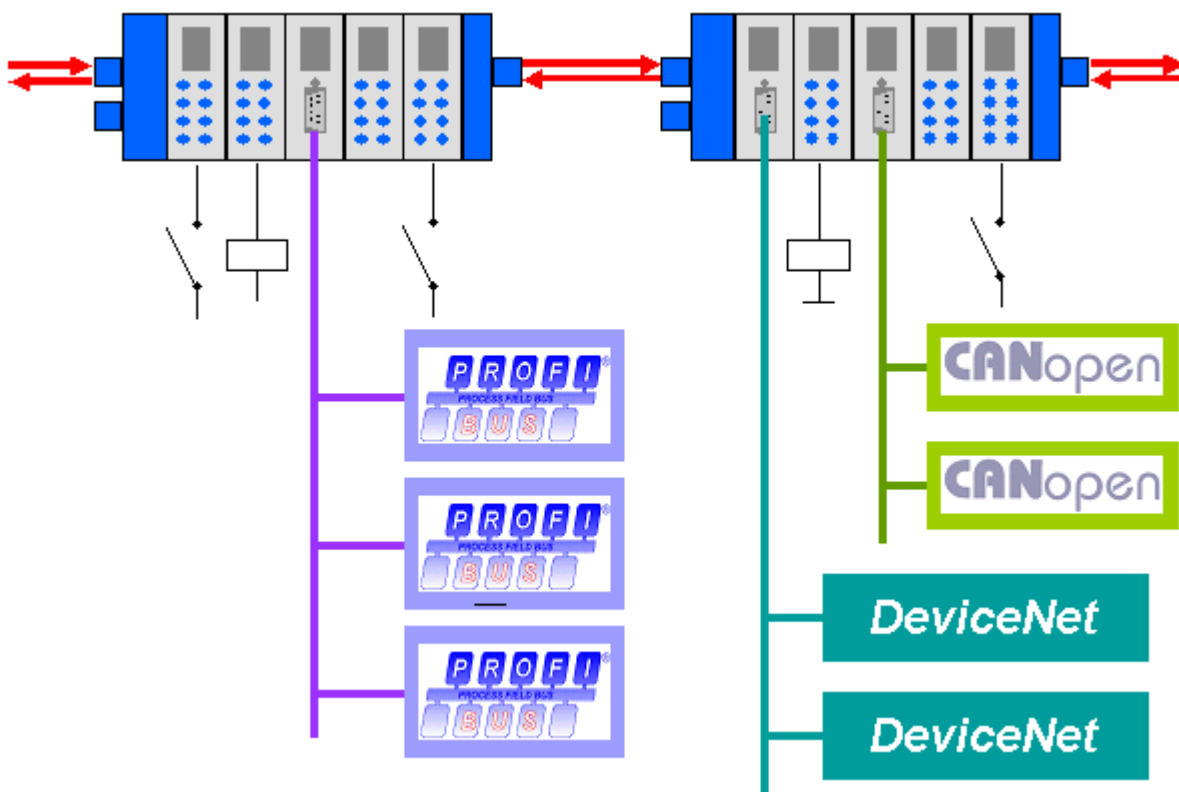


Fig. 6: Decentralization through intelligent interface terminals on the EtherCAT fieldbus



Fieldbus devices (e.g. Profibus, CANopen, DeviceNet, AS interface, etc.) are integrated through decentral fieldbus master terminals.

## 2.1.5 EtherCAT is open

The EtherCAT technology is not only fully Ethernet-compatible, but also characterized by a particular open design: the protocol tolerates other Ethernet-based services and protocols on the same physical network - usually only with minimum loss of performance. There is no restriction on the type of Ethernet device that can be connected within the EtherCAT strand via a switch port. Devices with fieldbus interface are integrated via EtherCAT fieldbus master terminals. The UDP protocol variant can be implemented on each socket interface.

EtherCAT devices can additionally have a TCP/IP stack and therefore externally behave like a normal Ethernet device. The master serves as a switch that transfers the frames to the respective devices according to the address information. Instead of switch ports, only the EtherCAT addresses automatically assigned during startup are used.

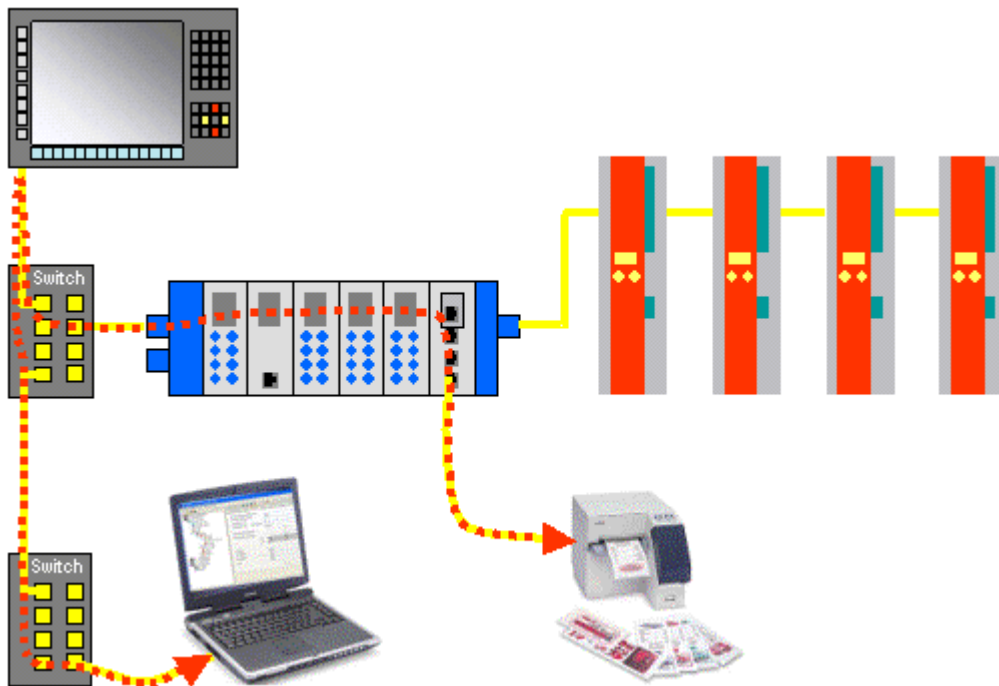


Fig. 7: Switch port terminals tunnel Ethernet frames through the EtherCAT protocol

The switch port terminals work accordingly: any Ethernet devices can be connected. The Ethernet frames are tunneled via the EtherCAT protocol, as usual in the Internet (e.g. VPN, PPPoE (DSL) etc.). The EtherCAT network is fully transparent for the Ethernet device, and the real-time characteristics of the EtherCAT are not impaired.

### EtherCAT Technology Group

The EtherCAT Technology Group (ETG, <http://www.ethercat.org>) is the international manufacturer and user association with a mission to disseminate, further develop and support EtherCAT. All companies are invited to become members.

### Internationally standardized

The EtherCAT Technology Group is an official standardization partner of IEC (International Electrotechnical Commission). The EtherCAT specification was published by the IEC as IEC/PAS 62407 and is available from the IEC (<http://www.iec.ch>). It is currently integrated in the following fieldbus standards: IEC 61158, IEC 61800-7 and ISO 15745.

## 2.1.6 Summary and Outlook

EtherCAT is characterized by outstanding performance, very simple wiring and openness for other protocols. EtherCAT sets new standards where conventional fieldbus systems reach their limits: 1000 I/O in 30 µs, optionally twisted pair cable or optical fiber and, thanks to Ethernet and Internet-technologies, optimum vertical integration. With EtherCAT, the costly Ethernet star topology can be replaced with a simple line structure - no expensive infrastructure components are required. Optionally, EtherCAT may also be wired in the classic way using switches, in order to integrate other Ethernet devices. Where other real-time-Ethernet approaches require special connections in the controller, EtherCAT manages with very cost-effective standard Ethernet cards (NIC).

With EtherCAT, Ethernet down to the terminal block becomes technically feasible and economically sensible. Full Ethernet compatibility, Internet technologies even in very simple devices, maximum utilization of the large bandwidth offered by Ethernet, outstanding real-time characteristics at low costs are outstanding features of this new network. As a high-speed bus and I/O bus for Industrial PCs or in combination with smaller control technology, EtherCAT is expected to be widely used in a wide range of applications.

## 2.2 Basic principles

### 2.2.1 EtherCAT State Machine

The state of the EtherCAT slave is controlled via the EtherCAT State Machine (ESM). Depending upon the state, different functions are accessible or executable in the EtherCAT slave. Specific commands must be sent by the EtherCAT master to the device in each state, particularly during the bootup of the slave.

A distinction is made between the following states:

- Init
- Pre-Operational
- Safe-Operational and
- Operational
- Boot

The regular state of each EtherCAT slave after bootup is the OP state.

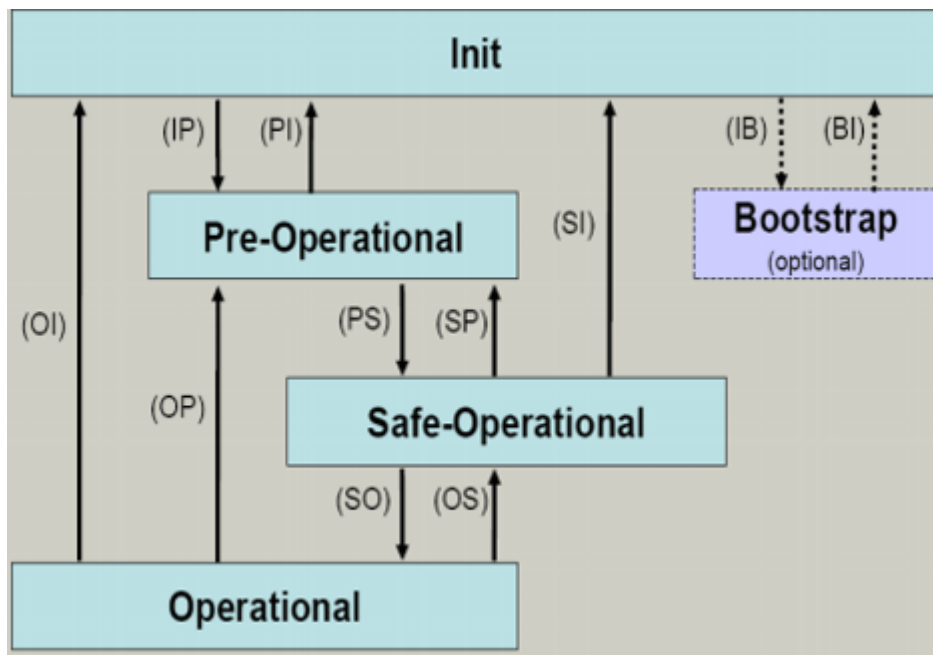


Fig. 8: States of the EtherCAT State Machine

## Init

After switch-on the EtherCAT slave in the *Init* state. No mailbox or process data communication is possible. The EtherCAT master initializes sync manager channels 0 and 1 for mailbox communication.

## Pre-Operational (Pre-Op)

During the transition between *Init* and *Pre-Op* the EtherCAT slave checks whether the mailbox was initialized correctly.

In *Pre-Op* state mailbox communication is possible, but not process data communication. The EtherCAT master initializes the sync manager channels for process data (from sync manager channel 2), the FMMU channels and, if the slave supports configurable mapping, PDO mapping or the sync manager PDO assignment. In this state the settings for the process data transfer and perhaps terminal-specific parameters that may differ from the default settings are also transferred.

## Safe-Operational (Safe-Op)

During transition between *Pre-Op* and *Safe-Op* the EtherCAT slave checks whether the sync manager channels for process data communication and, if required, the distributed clocks settings are correct. Before it acknowledges the change of state, the EtherCAT slave copies current input data into the associated DP-RAM areas of the EtherCAT slave controller (ECSC).

In *Safe-Op* state mailbox and process data communication is possible, although the slave keeps its outputs in a safe state, while the input data are updated cyclically.

---

### ● Outputs in SAFEOP state

**I** The default set watchdog monitoring sets the outputs of the module in a safe state - depending on the settings in SAFEOP and OP - e.g. in OFF state. If this is prevented by deactivation of the watchdog monitoring in the module, the outputs can be switched or set also in the SAFEOP state.

---

## Operational (Op)

Before the EtherCAT master switches the EtherCAT slave from *Safe-Op* to *Op* it must transfer valid output data.

In the *Op* state the slave copies the output data of the masters to its outputs. Process data and mailbox communication is possible.

## Boot

In the *Boot* state the slave firmware can be updated. The *Boot* state can only be reached via the *Init* state.

In the *Boot* state mailbox communication via the *file access over EtherCAT* (FoE) protocol is possible, but no other mailbox communication and no process data communication.

## 2.2.2 CoE interface

### 2.2.2.1 CoE interface – parameter management in the EtherCAT system

#### Background

Many different devices are used in an automation environment. These devices can be used separately or together in a group on a bus system. Such devices can be controllers, shaft encoders, servo drives, motors, I/O modules or sensors among other things.

Depending upon complexity, such a device must be parameterizable/adjustable for the respective requirement. Parameterization is perhaps not necessary in the case of a simple digital 24 V input with a fixed switching threshold and delay; in the case of a rotary encoder, however, it will not be possible to do without it (e.g. number of lines, absolute or relative, data format, etc.)

On top of that it can be of interest to store data in the device during production or operation. The manufacturer could store production data such as device name, serial number, firmware version, calibration data or date of manufacture, if necessary provided with protection against access or amendment. The user could store user calibration data and the application-specific settings in the device.

In order to create a user-friendly interface for device operation, different organizations have created various standards in which the following are defined:

- The device classes that exist (e.g.: class 'rotary encoder', 'analogue input module')
- The parameters that each representative of such a class has (obligatory and optional elements)
- The place where these parameters are to be found and the mechanism with which they are to be changed.

EtherCAT follows the so called CoE standard here: Can-Application-protocol-over-EtherCAT.

#### Can-over-EtherCAT

The CiA organization (CAN in Automation) pursues among other things the goal of creating order and exchangeability between devices of the same type by the standardization of device descriptions. For this purpose so-called profiles are defined, which conclusively describe the changeable and unchangeable parameters of a device. Such a parameter encompasses at least the following characteristics:

- Index number – for the unambiguous identification of all parameters. The index number is divided into a main index and a subindex in order to mark and arrange associated parameters.
  - Main index
  - Subindex, offset by a colon ':'
- Official name – in the form of an understandable, self-descriptive text
- Specification of changeability, e.g. whether it can only be read or can also be written
- A value – depending upon the parameter the value can be a text, a number or another parameter index.

Sample: the parameter 'Vendor ID' might have the index number 0x4120:01 and the numerical value '2' as the ID of a Beckhoff device.

Since hexadecimal values are favored in the machine environment, the parameter is thus represented from the user's point of view as

```
· 1018:01 Vendor ID RO 0x00000002 (2)
```

with the property RO (read only), because the vendor ID should not be changed by the user.

Such a list of parameters, the whole of the device-specific CoE directory, can become very extensive. The first entries of a Beckhoff EL3152 analog input terminal appear as follows in the TwinCAT System Manager:

Index:Subindex	Name	read/write possible	value
1000	Device type	RO	0x00001389 (5001)
1008	Device name	RO	EL3152
1009	Hardware version	RO	08
100A	Software version	RO	08
[-] 1011:0	Restore default parameters	RO	> 1 <
1011:01	SubIndex 001	RW	0x00000000 (0)
[-] 1018:0	Identity	RO	> 4 <
1018:01	Vendor ID	RO	0x00000002 (2)
1018:02	Product code	RO	0x0C503052 (206581842)
1018:03	Revision	RO	0x00110000 (1114112)
1018:04	Serial number	RO	0x00000000 (0)

Fig. 9: EL3152 CoE directory

The index numbers are specified in the profile; they begin in EtherCAT with 0x1000, because the underlying entries do not have to be displayed.

**CoE directory – availability**

An EtherCAT device can have a CoE directory, but does not need to have one. Simple slaves need no parameter directory or do not have the controller required for administration. On the other hand, the EtherCAT master (like TwinCAT) as a software EtherCAT device can also manage a CoE directory.

If present, the CoE directory is in operation from the PREOP state.

The object directory can be read via the SDO information service (Service Data Objects).

**CoE directory – localization in the EtherCAT Slave**

The CoE directory as a parameter system must be administrated in the device in the firmware (FW) in the local controller. This is the so-called *online directory*, because it is only available to the user if the EtherCAT slave is in operation with operating voltage supplied and, if applicable, can be manipulated via EtherCAT communication.

So that the parameters can be viewed and changed in advance without the presence of a slave, a default copy of the entire directory is usually stored in the device description file ESI (XML). This is called the *offline directory*. Changes in this directory do not affect the later operation of the slave with TwinCAT. The xml files can be obtained from the Beckhoff website in the [Download area](#).

The TwinCAT system manager 2.11 can display both lists and marks this:

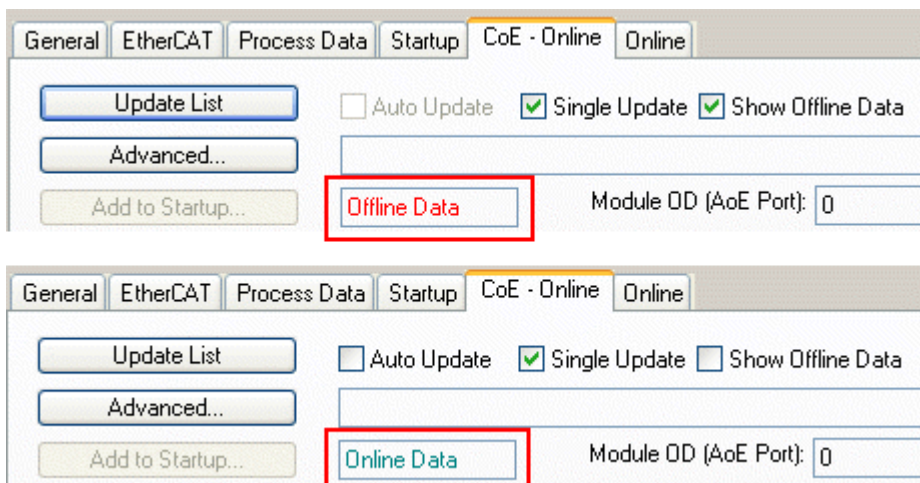


Fig. 10: Online/Offline display

In the online directory	In the offline directory
the actual current slave list is read. This may take several seconds, depending on the size and cycle time	the offline list from the ESI file is displayed. In this case modifications are not meaningful or possible.
the actual identity is displayed	the configured status is shown under Identity
the firmware and hardware version of the equipment according to the electronic information is displayed	no firmware or hardware version is displayed, since these are features of the physical device
a green <b>online</b> is visible in the TwinCAT System Manager, <i>CoE Online</i> tab	a red <b>offline</b> is visible in the TwinCAT System Manager, <i>CoE Online</i> tab

**Classification**

Different CoE parameter types are possible, including string (text), integer numbers, Boolean values or larger byte fields. They can be used to describe a wide range of features. Examples of such parameters include manufacturer ID, serial number, process data settings, device name, calibration values for analog measurement or passwords.

The ranges in the Slave CoE that are important for the application-oriented EtherCAT fieldbus user are

- 0x1000: This is where fixed identity information for the device is stored, including name, manufacturer, serial number etc., plus information about the current and available process data configurations.
- 0x8000: This is where the operational and functional parameters for all channels are stored, such as filter settings or output frequency.

The following ranges are also of interest

- 0x4000: In some EtherCAT devices the channel parameters are stored here (as an alternative to the 0x8000 range).
- 0x6000: Input PDOs ("input" from the perspective of the EtherCAT master)
- 0x7000: Output PDOs ("output" from the perspective of the EtherCAT master)

**Channel-based order**

The CoE directory is located in EtherCAT devices that usually encompass several functionally equivalent channels. e.g. a 4-channel 0 – 10 V analog input terminal also has 4 logical channels and thus 4 identical sets of parameter data for the channels. In order to avoid having to list each channel in the documentation, the placeholder "n" tends to be used for the individual channel numbers.

In the CoE system 16 indices, each with 255 subindices, are generally sufficient for representing all channel parameters. The channel-based order is therefore arranged in  $16_{dec}/10_{hex}$  steps. The parameter range 0x8000 exemplifies this:

- Channel 0: Parameter range 0x8000:00 ... 0x800F:255
- Channel 1: Parameter range 0x8010:00 ... 0x801F:255
- Channel 2: Parameter range 0x8020:00 ... 0x802F:255
- tbc...

This is generally written as 0x80n0.

**CoE directory – changes of value**

Several parameters, in particular the setting parameters of the slave, are variable and can be written by the user from the fieldbus side. This can be done in write or read mode

- via the system manager (Fig. *Manual insertion of a start-up entry*) by the operator clicking on it. The values are then changed directly in the online-connected slave. This is useful for commissioning of the system/slaves. Click on the row of the index to be parameterized and enter a value in the "SetValue" dialog.
- from the control system/PLC via ADS, e.g. through blocks from the TcEtherCAT.lib library. This is recommended for modifications while the system is running or if no System Manager or operating staff are available.

- during the EtherCAT startup through predefined commands, the so-called startup list. The TwinCAT configuration is usually created in advance without EtherCAT slaves actually being present. Then it should be possible before commissioning to adjust known properties such as filter settings offline in order to accelerate commissioning.

## CoE directory – startup list

### Start-up list

**i** Changes made to the local CoE directory of the EtherCAT slaves are lost with the old device in case of exchange. If a device is replaced with a new Beckhoff device, it will have the default settings. It is therefore advisable to link all changes in the CoE list of an EtherCAT slave with the Startup list of the slave, which is processed whenever the EtherCAT fieldbus is started. In this way a replacement EtherCAT slave can automatically be parameterized with the specifications of the user.

If EtherCAT slaves are used which are unable to store local CoE values permanently, the Startup list must be used.

The startup list is used for these cases: the values here, which are entered by the user, are transmitted to the respective slave upon each EtherCAT state transition/start. A startup entry consists of

- Time: the state transition in which the command is sent  
PS (PREOP-->SAFEOP) is usually the correct choice, since an EtherCAT slave then switches to operative input mode.
- Index: Subindex
- Data

The order of the entries is not taken into account: all entries to which a state transition applies are handed over simultaneously as asynchronous commands to the EtherCAT system and executed there, as soon as the bus load allows.

No check is made of whether an identical entry is already present in the slave.

### Sample

In the following, the user scaling is activated in the startup list of an EL3152. Entries required for operation that already exist in the list are marked with a grey background.

Upon right-clicking on the surface, the following dialog appears:

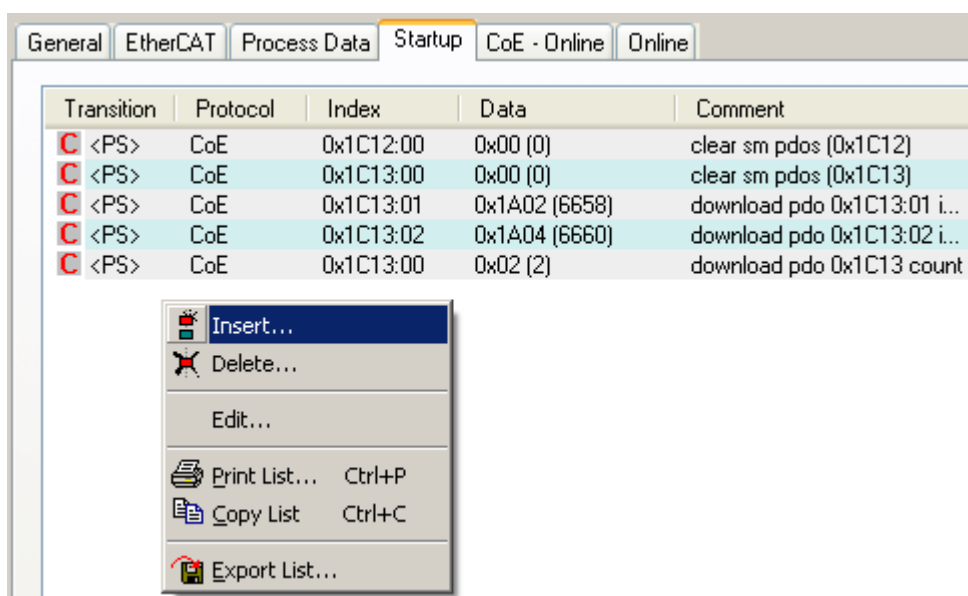


Fig. 11: Manual insertion of a startup entry



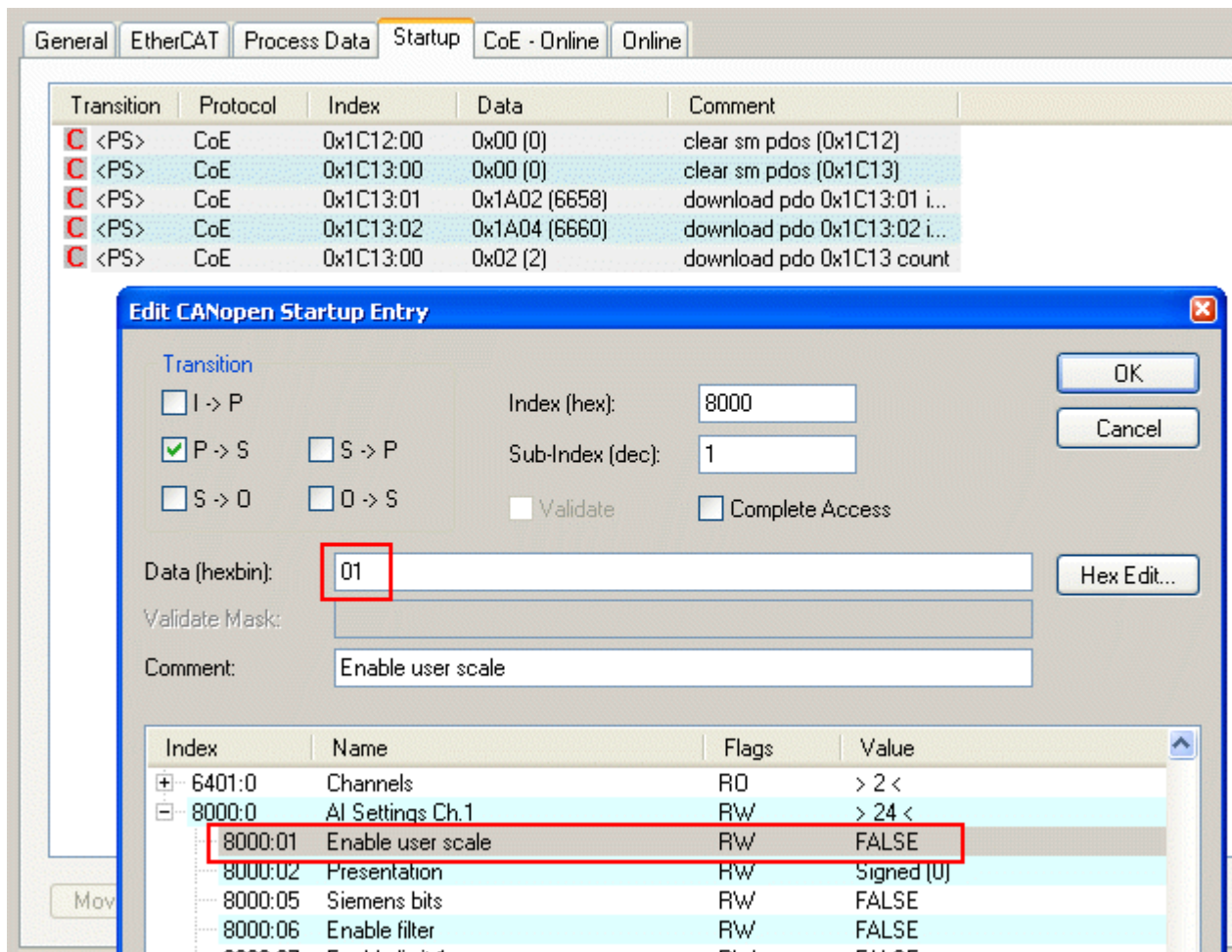


Fig. 12: Edit

The corresponding values are adopted upon clicking the entry 0x8000:01; 01 is entered in *Data* as the desired value. The entry 'P->S' marks the time of the execution.

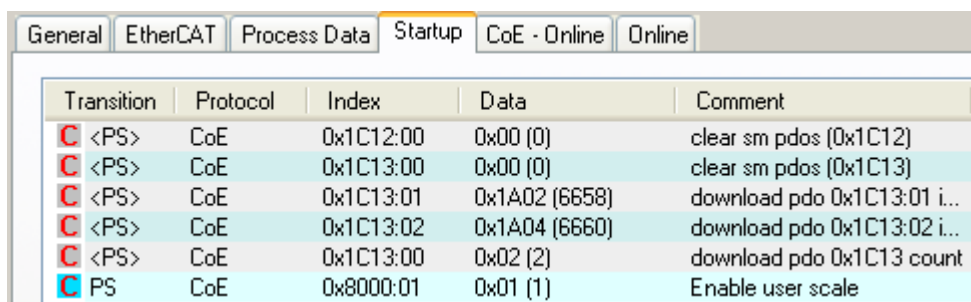


Fig. 13: Changed startup list

Entries created by the user are marked with a light blue background.

**CoE directory - data management**

**● Data management**



If slave CoE parameters are modified online, Beckhoff devices store any changes in a fail-safe manner in the EEPROM, i.e. the modified CoE parameters are still available after a restart. The situation may be different with other manufacturers.

If EtherCAT slaves are used which are unable to store local CoE values permanently, the Startup list must be used.

### Summary of the characteristics

- Not every EtherCAT device must have a CoE directory
- If a CoE directory is present, it is administrated, prepared for querying and writing and stored in the device by the controller.
- The EtherCAT master can be used for viewing/querying/changing, or a local user interface on the device (keypad, screen) allows access.
- Changed settings are stored fail-safe in Beckhoff devices.  
If the device is later exchanged, however, the settings that have been changed from the series standard are lost. The EtherCAT master can then load the changed CoE parameters into the new device at startup, if it is set up appropriately.
- So that a CoE directory is available offline during the preparation for configuration, it can be contained in the device description as a copy.
- The extent to which the CoE directory is supported depends on the capabilities of the EtherCAT master.

### 2.2.2.2 Example program R/W CoE

#### Program description/ function:

After starting this example program, by setting TRUE of the variable *startRead* or *startWrite*, read or write access from/to the CoE directory of a particular EtherCAT participant can be made.

#### Notes:

- An EtherCAT slave with CoE directory on the EtherCAT Master is present; this example is initial configured for usage of a EL3751 terminal; configuration e.g.: IPC/CX + (EK1100) + EL3751 + EL9011
- AmsNetId of the EtherCAT Master is known (it has to be inserted in the code before starting the program and can usually be found in the EtherCAT tab of the EtherCAT master)
- The address of the EtherCAT slave have to be adapted by variable declaration of *userSlaveAddr* as the case may be (usually the terminal, which CoE directory have to be accessed, e.g. 1002, 1007, etc.)
- Depending on which EtherCAT slave is used, the corresponding *nIndex* for the CoE object ID as well as the *nSubIndex* for the sub index of the CoE object must be entered at the places of the value transfer for reading and writing. The correct data length and data type may also need to be adjusted.
- This example program only makes accesses to a specific value identified by the (sub) index of a CoE object. If accesses are to be made to a complete object, the corresponding *FB\_EcCoeSdoReadEx* and *FB\_EcCoeSdoWriteEx* function blocks must be used. See additional documentation of library: Tc2\_EtherCAT:  
<https://infosys.beckhoff.com/>  
TwinCAT 3 → TE1000 XAE → PLC → Libraries → TwinCAT 3 PLC Lib: Tc2\_EtherCAT  
→ CoE interface

This example writes into the CoE object 0x8000 with sub index 0x16 the value 22 and activates thereby filter 1 to „IIR Butterw. LP 5th Ord. 1000 Hz“, or reads out the internal temperature value of the EL3751 terminal from CoE object 0x9000, Sub-Index 0x01. The program variable for writing has already been assigned the value in the variable declaration. For test of writing the CoE directory can be reviewed and the variable (*int16Buffer*) can be watched to check reading.

Download: <https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/5299954699.zip>

#### Preparations for starting the sample programs (tzip file / TwinCAT 3)

- Click on the download button to save the Zip archive locally on your hard disk, then unzip the \*.tzip archive file in a temporary folder.

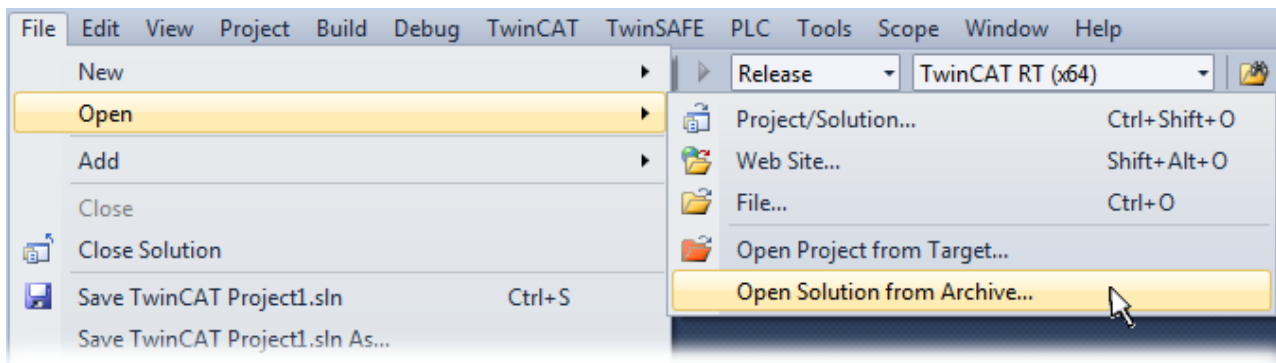


Fig. 14: Opening the \*.tnzip archive

- Select the .tnzip file (sample program).
- A further selection window opens. Select the destination directory for storing the project.
- For a description of the general PLC commissioning procedure and starting the program please refer to the terminal documentation or the EtherCAT system documentation.

### Declaration (ST)

```

PROGRAM MAIN
VAR
  fb_coe_write      : FB_EcCoESdoWrite; // Function Block for writing in CoE
  fb_coe_read       : FB_EcCoESdoRead;  // Function Block for reading from CoE
  userNetId         : T_AmsNetId := 'a.b.c.d.4.1'; // Have to be entered
  userSlaveAddr     : UINT := 1002;      // Have to be entered
  startWrite        : BOOL := FALSE;     // Sign for start writing
  startRead         : BOOL := FALSE;     // Sign for start reading
  nState            : BYTE := 0;         // RW-status
  // Example: Read EL3751 PAI Internal Data: Temperature Value
  int16Buffer       : INT;               // Buffer for reading
  // Example: Select EL3751 Filter1: 22 = IIR Butterw. LP 5th Ord. 1000 Hz:
  uint16Buffer      : UINT:=22;         // Buffer for writing
  bTxPDOState AT%I* : BOOL;             // (PDO for synchronization)
END_VAR

```

### Implementation (ST):

```

CASE nState OF
0:
  IF startWrite THEN
    // Prepare CoE-Access: Write value of CoE object/ sub index:
    fb_coe_write(bExecute := FALSE);
    nState := 1; // Next state for writing
    startWrite := FALSE;
  END_IF
  IF startRead THEN
    // Prepare CoE-Access: Read value of CoE object/ sub index:
    fb_coe_read(bExecute := FALSE);
    nState := 11; // Next state for reading
    startRead := FALSE;
  END_IF
// ===== COE WRITE =====
1:
  // Write entry
  fb_coe_write(
    sNetId:= userNetId,

```

```

nSlaveAddr:= userSlaveAddr,
nSubIndex:= 16#16,
nIndex:= 16#8000,
pSrcBuf:= ADR(uint16Buffer),
cbBufLen:= 2,
bExecute:= TRUE,
tTimeout:= T#1S
);
nState := nState + 1; // Next state
2:
fb_coe_write(); // Execute CoE write until done
IF fb_coe_write.bError THEN
nState := 100; // Error case
ELSE
IF NOT fb_coe_write.bBusy THEN
nState := 0; // Done
END_IF
END_IF
// ===== COE READ =====
11:
// Read entry
fb_coe_read(
sNetId:= userNetId,
nSlaveAddr:= userSlaveAddr,
nSubIndex:= 1,
nIndex:= 16#9000,
pDstBuf:= ADR(int16Buffer),
cbBufLen:= 2,
bExecute:= TRUE,
tTimeout:= T#1S
);
nState := nState + 1; // Next state
12:
fb_coe_read(); // Execute CoE read until done
IF fb_coe_read.bError THEN
nState := 100; // Error case
ELSE
IF NOT fb_coe_read.bBusy THEN
nState := 0; // Done
END_IF
END_IF
100:
; // Error handling..
END_CASE

```

### 2.2.2.3 CoE reset, restoring the default values

The CoE object "Restore default parameters", subindex 001 (if present) can be used to restore the delivery state of the changeable CoE objects in the ELxxxx terminals (see Fig. *Selecting the "Restore default parameters" PDO*).

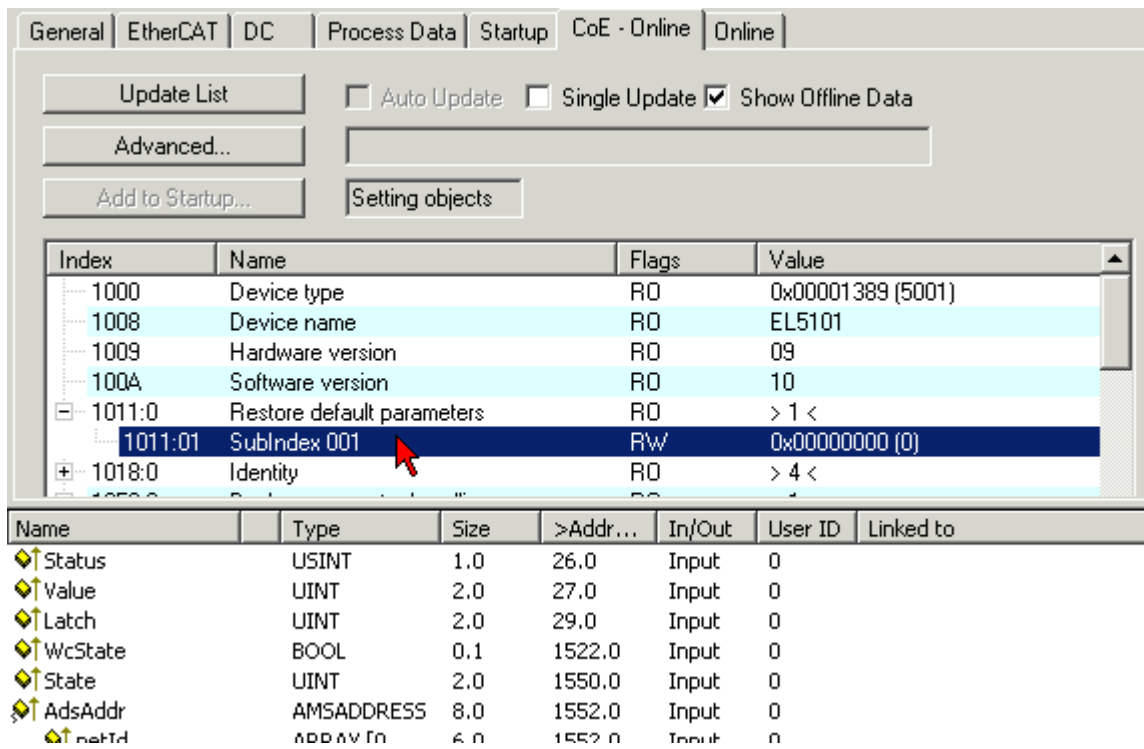


Fig. 15: Selecting the "Restore default parameters" PDO

All changeable entries in the slave are reset to the default values by writing the reset word to these indices.

- The values can only be successfully restored if the reset is directly applied to the online CoE, i.e. to the slave. No values can be changed in the offline CoE.
- TwinCAT must be in the RUN or CONFIG/Freerun state for this; error-free EtherCAT traffic must take place.
- A changeable object can be manipulated beforehand for the purposes of checking; no separate confirmation takes place.
- This reset procedure can also be adopted as the first entry in the startup list of the slave, e.g. in the state transition PREOP->SAFEOP or, as in Fig. *CoE reset as a startup entry*, in SAFEOP->OP. Hence any settings in the slave are reset.

C	<PS>	CoE	0x1C13:00	0x04 (4)	download pdo 0x1C13 count
C	S0	CoE	0x1011:01	0x6C6F6164 (1819238756)	SubIndex 001: Reset all CoE Values
C	S0	CoE	0x8000:01	0x01 (1)	Enable user scale

Fig. 16: CoE reset as a startup entry

- The EtherCAT slave must be at least in the PREOP state for this
- Depending upon firmware and slave, the reset word reads:
  - **64616F6C "load"**<sub>hex</sub> (as a rule in the case of FW since approx. 2008) or
  - **6C6F6164**<sub>hex</sub> "dao" (in the case of earlier FW versions)
- An incorrect entry for the restore value has no effect.

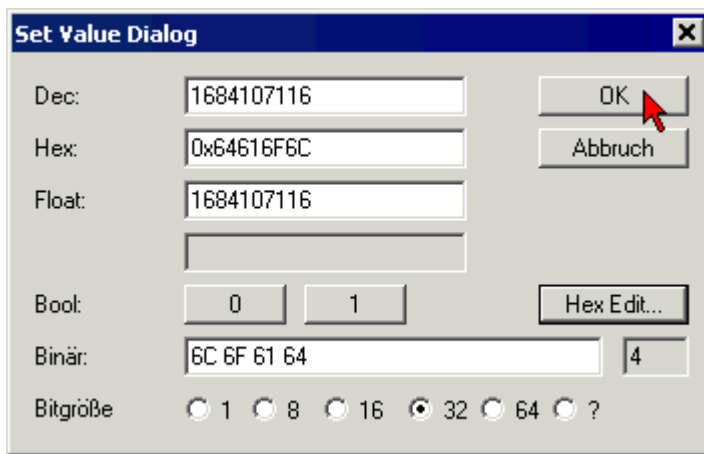


Fig. 17: Entering a restore value in the Set Value dialog

## 2.3 System features

### 2.3.1 EtherCAT cable redundancy

#### 2.3.1.1 The principle

Beckhoff TwinCAT cable redundancy is designed to compensate for the failure of a communication cable section in the EtherCAT system. A ring topology, which normally is operated in both directions, is therefore used. Both branches can nevertheless still be reached if the ring is interrupted at some point.

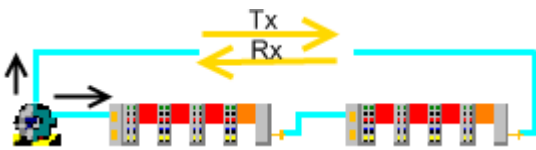


Fig. 18: The cable redundancy principle

A second network port is used for ring closure at the EtherCAT Master IPC. Both cyclic and acyclic frames are sent simultaneously through both ports, and are transported through the system.

- In the absence of any fault, all the EtherCAT slaves are reached in the forward direction from the primary port. This means that they are processed, since the EtherCAT Slave Controller (ESC) is only passed through in the forward sense.
- When there is no fault, all the EtherCAT slaves are reached from the secondary port in the reverse direction - the data in the "redundancy" frame is therefore not changed.

In each case, the EtherCAT frames arrive, possibly modified, at the other port, and are combined again by the EtherCAT Master. If, as a result of a broken cable, the redundancy comes into play, it is then unimportant whether an EtherCAT slave is reached from the primary or redundancy port.

Assuming that, when the redundancy comes into play (due to a damaged cable, damaged plug or electromagnetic interference) the two Ethernet frames are not, by coincidence, both directly affected, then redundant operation continues without interruption or loss of data.

The supplement is single-fault tolerant, i.e. communication with the slaves can continue if the cable is interrupted in *one* place. When the communication is restored the original communication direction is restored. If the communication is interrupted in more than one places, all connections have to be restored before another fault may occur.

It is also possible to start up the system under redundancy conditions.

Due to the nature of this principle, a closed ring topology is most suited to redundant cable operation. Depending on the operating conditions, it is also possible to use a connection point other than the last one for the redundant connection to the controller, see Fig. *Special solution for cable redundancy*.

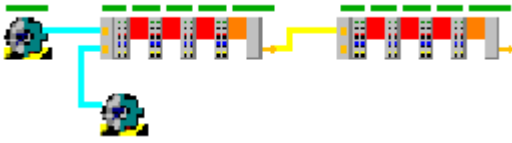


Fig. 19: Special solution for cable redundancy

Due to the synchronization mechanism for Distributed Clocks, the [CU2508](#) port multiplier must be used for a combination of cable redundancy with Distributed Clocks slaves.

### ● Failure of an EtherCAT slave

**i** Note the following if the "Cable Redundancy" supplement is to be used to enable communication with all remaining devices in the event of an EtherCAT device failure (e.g. EK1100 coupler or EPxxxx box).

- Create SyncUnits for the I/O group that could be affected by failure
- Failure of several EtherCAT slaves is not covered. Depending on the location, number and order of the failures the complete EtherCAT array may have to be restarted.  
It is advisable to operate the master and slave states from the application and to disable the corresponding automatic mechanism under System Manager | EtherCAT Device | Advanced Settings.

## 2.3.1.2 Conditions

The use of cable redundancy is subject to the following conditions:

- A supplementary license with the appropriate scope is installed on the IPC (XP/CE). At present, licenses are available for a maximum of 250, 1000 or for an unlimited number of slaves. The TwinCAT EtherCAT redundancy supplement for the Windows NT and CE family is offered for [download](#) on the Beckhoff website
- Platform-dependent/operating system on the target device
  - Windows NT family (XP/XPe/7): the supplement is installed on the target device. Starting in the TwinCAT RUN mode with activated cable redundancy is prevented if no local license is present.
  - Windows CE family (CE, WES): the supplement must be installed on at least one NT platform. After that the installation file ("TwinCAT\_EtherCAT\_Redundancy\_CE.I586\_250.cab") is present under TwinCAT -> CE -> TwinCAT EtherCAT Redundancy. This must be copied to the embedded device and executed there.
- Please refer to the appropriate documentation for the combination of cable redundancy with the HotConnect principle.
- TwinCAT 2.10 from build 1313 or TwinCAT from version 2.11 is used.
- The user creates SyncUnits in the System Manager as required according to the configuration, see section SyncUnits.
- The control PC has two full-value, real-time capable, non-switched Ethernet ports. As a rule embedded devices (BXxxxx / CXxxxx) are not suitable.
- Parallel wiring from an EK1122 for the purposes of cable redundancy is not permitted.
- The supplementary EtherCAT cable redundancy secures against
  - failure of Ethernet cable sections (e.g. between EK1100 couplers), provided the power supply to the stations is not interrupted.
  - failure of the communications segment of a single terminal, provided the power supply to the subsequent terminals is not interrupted.

- The supplementary *cable redundancy* is only single-fault tolerant. If more than one malfunction occurs in the topology, full I/O communication will only be restored when all the faults have been eliminated. Manual or automation restart of the affected slaves may be required. This is partly also done by the System Manager.
- A combination of cable redundancy with distributed clocks is possible only with the use of the additional [CU2508](#) device.

### 2.3.1.3 Setting up - Network settings

#### **i** Creation of the configuration

It is not possible to set up a configuration through scanning in TwinCAT with redundancy path. You should therefore disconnect the redundancy port in order to scan for or create the slaves.

Once the redundancy supplement has been installed, set up continues under TwinCAT 2.11 as follows:

- Create the primary port manually, or allow it to be found through a scan.

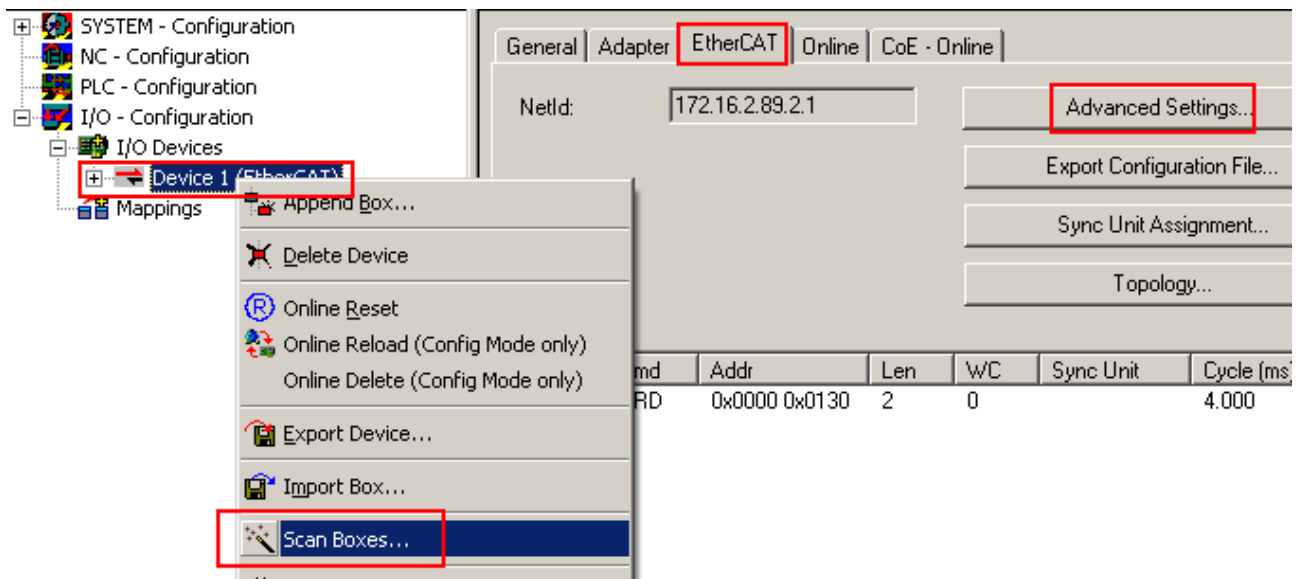


Fig. 20: Setting up the device

- Configure the redundancy properties of the EtherCAT system in the Advanced Settings. Specify the location from which the connection to the redundancy port is to be created - entry points that are not located at the end of a line topology can also be selected here.



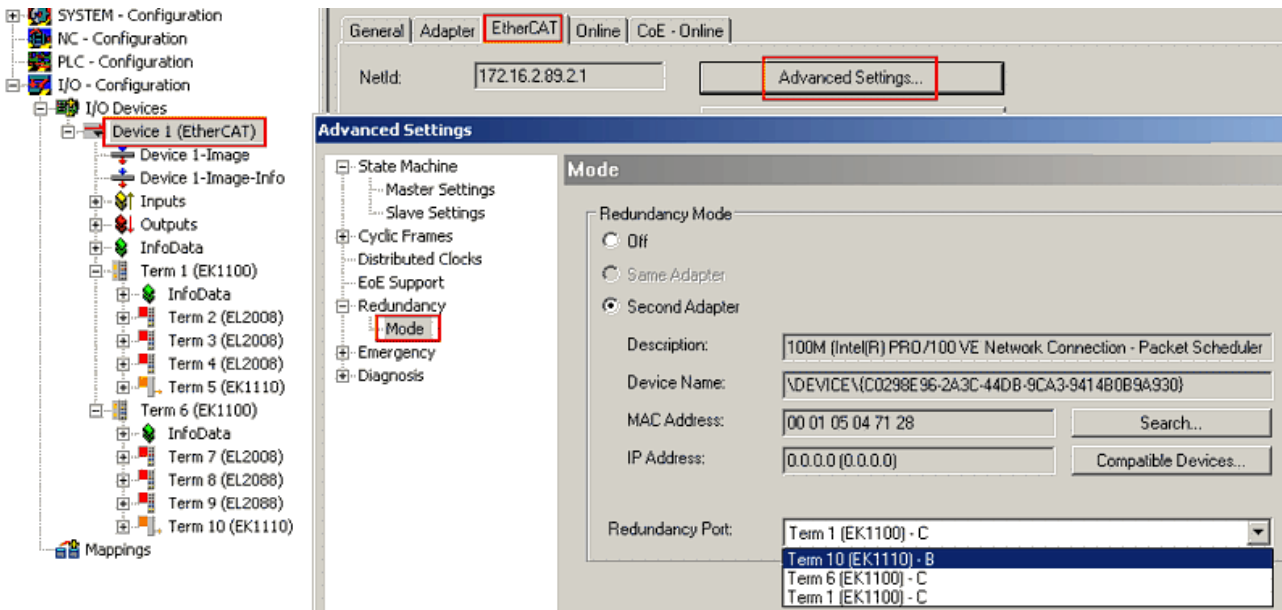


Fig. 21: Parameterizing the redundancy

The cable redundancy is active following activation of the configuration.

### 2.3.1.4 Setting up - SyncUnits setup

SyncUnits only have to be created if failure of an EtherCAT slave is to be expected.

#### **i** Cyclic data

SyncUnits are only relevant for communication via cyclic data. Acyclic communication, e.g. via mailbox, is always controlled by the System Manager.

By default the System Manager tries to integrate all I/O data in a few Ethernet/EtherCAT frames as possible, i.e. the System Manager tries to utilize the maximum frame size of 1518 byte per Ethernet frame or 15 datagrams. The aim is to achieve efficient and optimized communication with minimum load on the fieldbus. A datagram is a telegram with exactly one read/write instruction. A datagram may be a read command over 2 bytes from an analog input terminal or a write command for 400 bytes of data to 10 servo drives.

Ethernet header		Ethernet Data					
14 byte		2 Byte		44 - 1498 Byte		4 Byte	
Ethernet header		Length	Reserv.	Type	1..15 Datagrams		CRC
		Data	WC	Data	WC		

Fig. 22: Structure of an Ethernet frame with EtherCAT protocol data

This is illustrated in Fig. *Structure of an Ethernet frame with EtherCAT protocol data*: the Ethernet frame (row 1) consists of a header (blue), data (yellow) and the checksum CRC (blue). If the Ethernet frame carries EtherCAT protocol data, these data also consist of an EtherCAT header (red) and 1 to 15 datagrams (green). The latter consist of a header, data and a WorkingCounter (WC).

The working counter is the key parameter for the cable redundancy principle. The working counter is a 16-bit number sent by the EtherCAT master with the value "0". Each EtherCAT slave that is accessed by this datagram in write or read mode increases this number by 1, 2 or 3, depending on the type of access. Once the datagram has run through the complete configuration, the working counter therefore returns to the master with a value greater than 0. The master expects a certain value, because it knows how many slaves

have to process this datagram. If the WC returned by the datagram does not match the expected value, one of the slaves did not fulfil its task. The master then has to employ special measures in order to establish which slave is affected and whether the datagram has to be repeated.

The frame structure of the cyclic data based on the current configuration can be viewed in the TwinCAT System Manager (EtherCAT tab of the respective EtherCAT device).

Example for a small configuration:

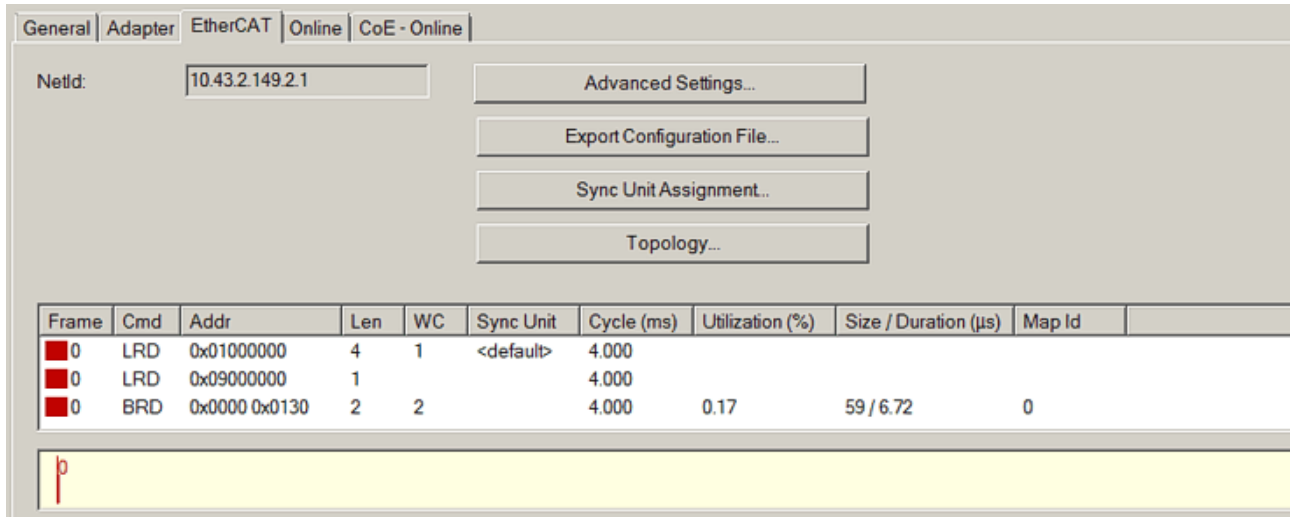


Fig. 23: TwinCAT view, frame structure - small topology

Only one Ethernet frame is used for the cyclic I/O data (red), carrying 3 EtherCAT datagrams, i.e. 2x LRD (LogicalRead) and 1x BRD (BroadcastRead).

With 59 bytes of data the frame is 6.72 µs long and uses 0.17% of the cycle time of 4 ms.

The first datagram has an expected working counter of 1 and a BRD command of 2, see column WC.

In a larger configuration the frame structure becomes more complex:

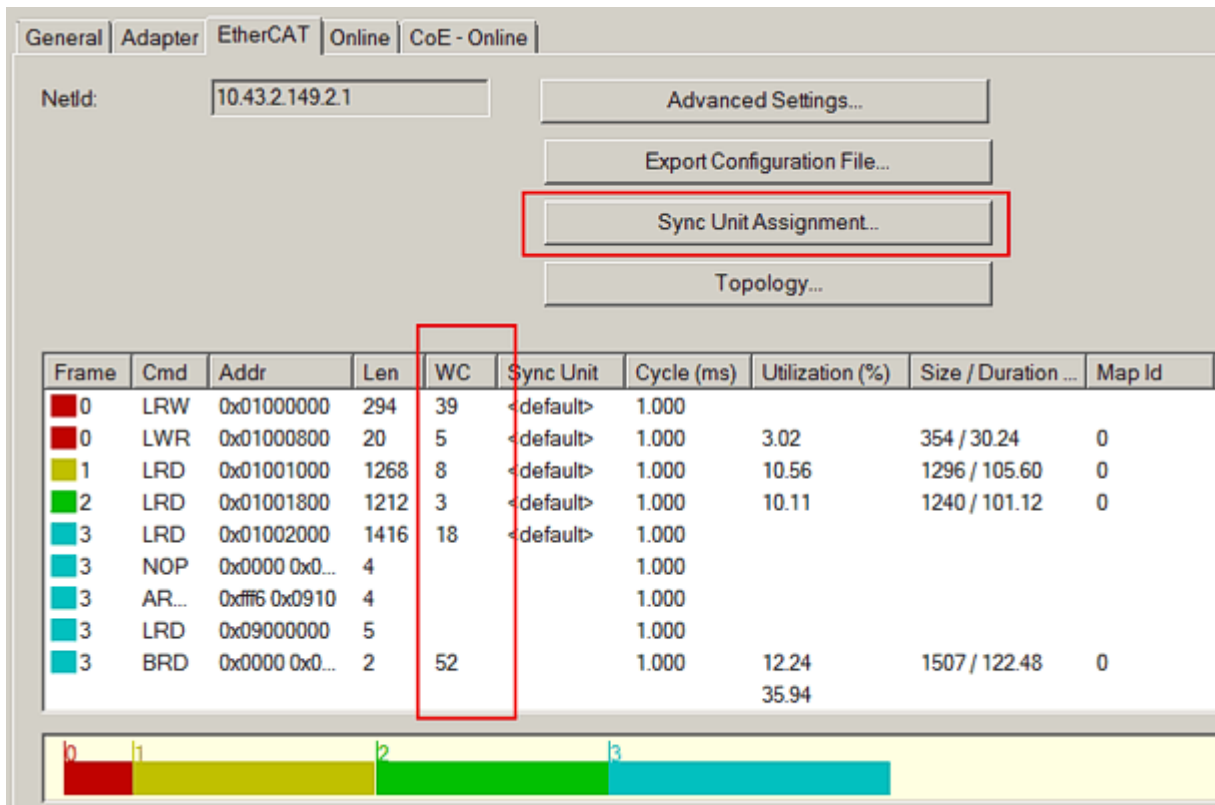


Fig. 24: TwinCAT view, frame structure - larger topology

With a cycle time of 1 ms 3 Ethernet frames are in transit carrying a total of 9 datagrams with expected working counters between 3 and 52. This configuration uses 35.94% of the available bandwidth of 100 Mbit FastEthernet at a cycle time of 1 ms.

**Application to the cable redundancy in the event of slave failure**

If a datagram returns to the master with an incorrect working counter, the master is initially unable to determine which slave failed to deliver data. The master therefore has to declare all input data in this datagram invalid, i.e. in all affected slaves the WC indicator switches to 1=*invalidData*.

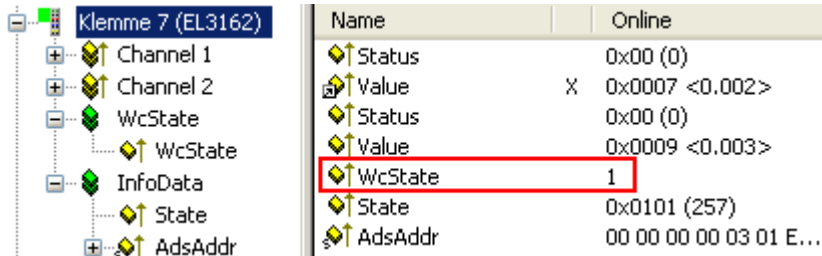


Fig. 25: WC = 1 indicates invalid data

The System Manager sets this WC indicator immediately on receipt of the datagrams for the respective slave. This is illustrated for an EL3162 in Fig. *WC = 1 indicates invalid data*, the input data are frozen, *WcState = 1* and the *State* in accordance with the bit meanings of 0x0101 "Slave in INIT" and "Slave not present".

If an EtherCAT slave fails, e.g. an analog input box EP3174, this would mean other input data contained in the same datagram to make the frame structure efficient would also be discarded. As a remedy the user can manually assign each slave to a so-called SyncUnit. In extreme case each slave could have its own SyncUnit, with associated negative consequences for the fieldbus efficiency and excessive bus load.

The corresponding dialog is accessible via the EtherCAT tab, see Fig. *TwinCAT view, frame structure - larger topology*.

A slave is assigned to a SyncUnit, see Fig. *Entering the SyncUnits in a default frame structure*

- by clicking in the column
- Enter a name (any name) under SyncUnitNames

If a name is allocated more than once, the slaves are operated accordingly in a datagram (logical SyncUnit).

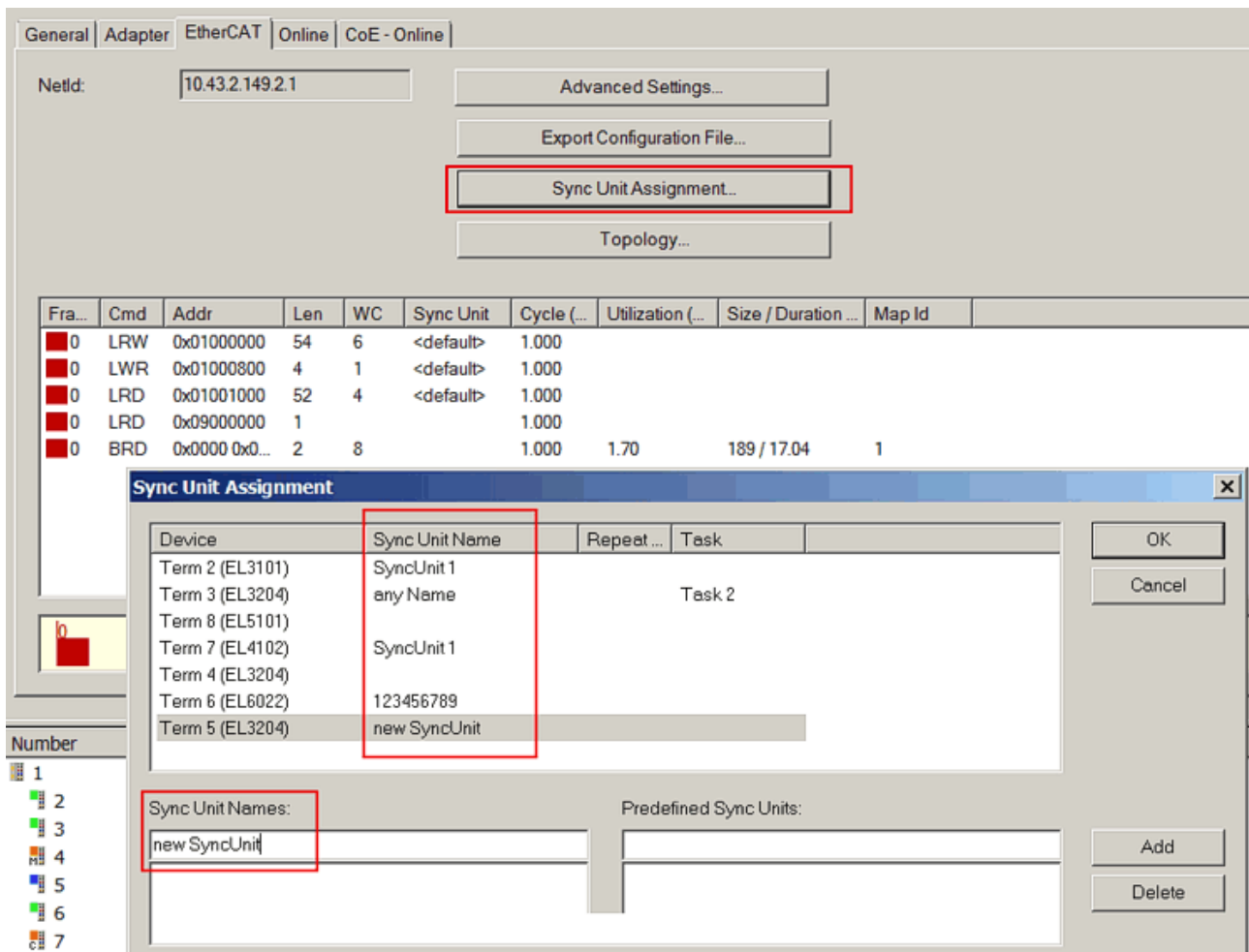


Fig. 26: Entering the SyncUnits in a default frame structure

This results in a new structure of the cyclic Ethernet frame in accordance with Fig. *New frame structure with SyncUnits*:

Frame	Cmd	Addr	Len	WC	Sync Unit	Cycle (ms)	Utilization (%)	Size / Duration ...	Map Id
0	LRW	0x01000000	6	3	<default>	1.000			
0	LRD	0x01000800	16	1	<default>	1.000			
0	LWR	0x01001000	4	1	SyncUnit 1	1.000			
0	LRD	0x01001800	4	1	SyncUnit 1	1.000			
0	LRD	0x01002000	16	1	any Name	1.000			
0	LRW	0x01002800	48	3	123456789	1.000			
0	LRD	0x01003000	16	1	new SyncUnit	1.000			
0	LRD	0x09000000	1			1.000			
0	BRD	0x0000 0x0...	2	8		1.000	2.08	237 / 20.88	1
							2.09		

Fig. 27: New frame structure with SyncUnits

The System Manager will continue to automatically consolidate devices that are not set manually in datagrams and may identify them as <default>.

Typically modules are defined as a separate SyncUnit and communicate via an Ethernet cable connection:

- EK11xx coupler including append EL/ES/EMxxxx terminals
- EP boxes
- AX drives

- ...

If the communication with such a station fails, only the data for this SyncUnit become invalid. The data exchange with all other stations remains unaffected.

### 2.3.1.5 System behavior

TwinCAT handles the redundancy case in real-time, and the user has no intervention option. However, information on the current status is provided by variables in the System Manager.

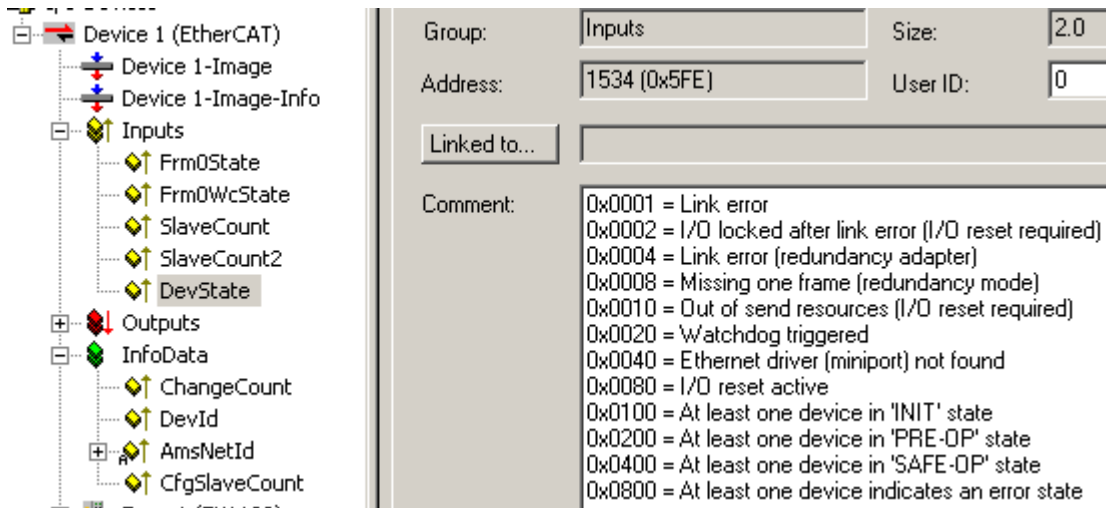


Fig. 28: Available information

- SlaveCount/SlaveCount2  
This indicates how many EtherCAT slaves are at present being reached through the primary or redundancy port. This information refers to the most recent cycle.
- FrmXWcState  
This information refers to the most recent cycle. It is recommended that this accumulated information for all the working counters of an EtherCAT frame is monitored on the application level, even in redundancy mode.
- DevState  
The meanings of the bits
  - 0x0001: Link error at primary port
  - 0x0004: Link error at redundancy port
  - 0x0008: Missing one frame
- refer to the cable redundancy. If all three bits are zero, then the status of the cable redundancy is normal, and the next connection fault can occur.  
Under some circumstances the link status messages can be delayed by a few milliseconds. They can therefore not be used in a real-time context.

The redundancy case is illustrated accordingly in the Topology display.

- Fig. *Typical error patterns in the redundancy case*, top: Interruption at redundancy port
- Fig. *Typical error patterns in the redundancy case*, center: 2 interruptions, communication failure to one terminal block.
- Fig. *Typical error patterns in the redundancy case*, bottom: Terminal in the second terminal block removed while running, causing interruption in power supply to subsequent terminals and failure of communication with them.

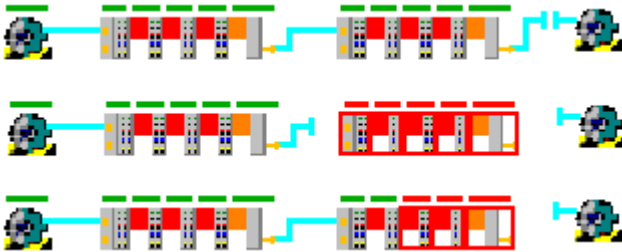


Fig. 29: Typical error patterns in the redundancy case

### ● The redundancy case

**i** Redundancy is an irregular operating condition. It is recommended that occurrence of the redundancy case is detected at an early stage through evaluation of the diagnostic variables shown above, and that the cause is rectified promptly.

## 2.3.2 HotConnect

### 2.3.2.1 The principle

The Beckhoff TwinCAT EtherCAT Hot Connect function allows preconfigured sections to be removed from or added to the data traffic before the start or during operation of the system. This can take place by disconnecting/connecting the communication line, switching the device on or off or by some other measure. This is called "flexible topology" or Hot Connect.

Sample:

In a modular production system, modules with integrated EtherCAT I/O are connected or disconnected with respect to communication during operation.

Following the establishment of a connection to a Hot Connect group, some commissioning actions are carried out that result in a boot up time of several seconds:

- establishment of an Ethernet link
- device parameterization
- EtherCAT boot up INIT -> OP
- if necessary Distributed Clocks synchronization

An accelerated boot up is advantageous in particular in applications in which topology changes are frequently made, such as tool changers. In addition, special Fast Hot Connect components are available that ensure a boot up time of less than 1 second. See [Notes on Fast Hot Connect \[▶ 42\]](#) regarding this.

The Hot Connect/Fast Hot Connect principle is thus an extension to the otherwise generally applicable rule that the sequence/arrangement of the EtherCAT devices in the field must correspond precisely to the configuration created.

No separate license is required for setup, just the EtherCAT devices (couplers and branches) conceived for the purpose.

## ● TwinCAT

The use of the Hot Connect function in the form described here is possible only with TwinCAT 2.11 from build 1539 upwards. In particular, TwinCAT 2.10 does not support any combination with Distributed Clocks.

### Properties and system behavior

- The information in this document applies to both XP and CE-based TwinCAT 2.11 systems.
- Hot Connect groups can be groups of slaves (couplers & terminals) or also individual slaves (drives, terminals, sensors, position encoders).
- Although a Hot Connect group can be physically connected to the EtherCAT network and can receive and send the EtherCAT frame, it only participates in the process data traffic from the point of view of the controller if its address (see below) has been recognized by the master.
- The identification and start-up of a Hot Connect group by the master can take up to several seconds, depending on the position.  
Accelerated boot up can be realized with Fast Hot Connect components.
- The monitoring of the WcState, status and link information from the slaves is strongly recommended.
- Start-up of a Hot Connect group by the master
  - if the EtherCAT master does not detect a valid address in the station, the station will not be accepted into the process data traffic.
  - after a valid address is detected (“switching in” with a DIP switch, changing the SSA), the station is accepted.
  - the above points remain unchanged until the station is switched off or disconnected from the network.
- SyncUnit: The System Manager automatically creates its own SyncUnits for Hot Connect groups which are separate from each other; hence, they are given their own WorkingCounter and their being taken out of operation does not affect the process data of other devices.
- The first device/coupler after the master should not be a configured Hot Connect device, as this slows down link detection in the master.
- The combination with the “cable redundancy” property is partly possible.
- Distributed Clocks-capable slaves can be used.  
When a Distributed Clocks slave (DC slave) is put into operation, its local clock is initialized and then continuously synchronized with the existing network.  
The Hot Connect group is resynchronized following the establishment of a connection; this procedure can take several seconds. The EtherCAT Slaves concerned are held in the SAFEOP state during this time. After successful resynchronization they are switched to the OP state.

## ● EL terminals as a Hot Connect group

The connection/disconnection of KL/KS/EL/ES Terminals when live is not allowed. Therefore, a configuration of individual terminals or terminal blocks below a coupler as a variable Hot Connect group does not make sense. The smallest unit at terminal level is a coupler (EK/BK) or an EP Box.

### Topologies

The following topologies were checked:

#### Type 1: Star

Star topologies are particularly useful for the Hot Connect concept - the branching groups are defined as a Hot Connect group and can be connected/disconnected during operation.

In general, any topology is possible in which an EtherCAT network is connected to an EtherCAT port.

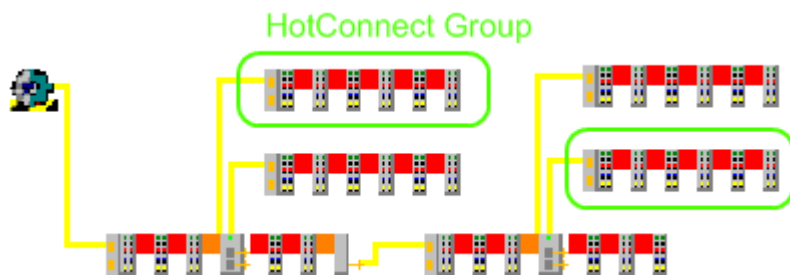


Fig. 30: Star topology

The following are suitable for use as branching points: EK110x, EK150x, EK1122, EP1122 and all slaves that have more than one IN and OUT port.

A Hot Connect group can be connected to any regular free port in the topology.

The combination of cable redundancy (chargeable supplement) in the main circuit and branching Hot Connect groups at radial connections is not permitted for the time being. The use of DC slaves is not yet possible when using cable redundancy.

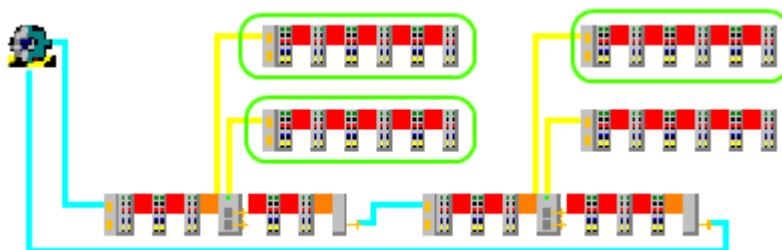


Fig. 31: Star topology with cable redundancy – presently not permitted

**Type 2: Line**

When using the line topology, all devices will be re-initialized when the connection is restored after a separation point.

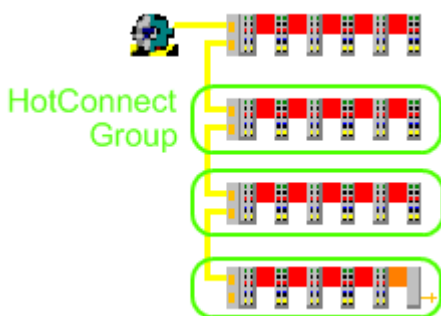


Fig. 32: Line topology

If HC groups and non-HC groups are mixed in the line topology, all non-HC groups must be placed in front of the HC groups and (as is usual for EtherCAT) in the correct order.

In the picture here, a non-HC group is arranged in front of 3 HC groups. In an extreme case, all stations can be HC groups.

The use of cable redundancy (chargeable supplement) with HC groups in the redundancy path is not possible.



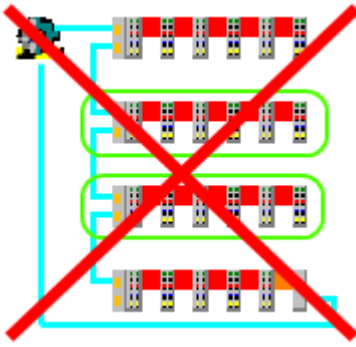


Fig. 33: Line topology with cable redundancy is not possible

**General notes**

- **Stacked Groups**

If Hot Connect groups are operated in physical succession (“stacked”), the higher level group must participate in traffic first before the lower level groups can be put into operation.

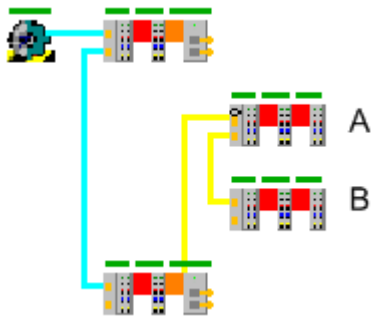


Fig. 34: Stacked groups

Sample: The groups A and B, each with their own address, are connected/switched on together. Group B (in spite of a possibly valid address) is only put into operation if it was possible for Group A to be put into operation regularly by the master.

- **Regular free ports**

The Hot Connect groups can only be connected to *free* Ethernet ports in the configuration - if a group is connected to an irregularly free port, the group is not put into operation, despite having a possibly valid address.

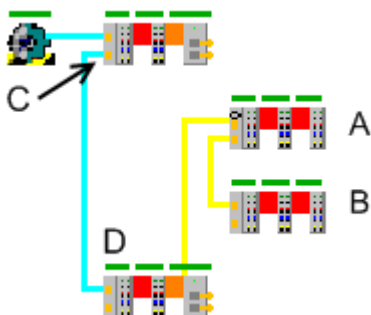


Fig. 35: Free ports

Sample: The coupler D (not a Hot Connect group!) is disconnected from Port C. Hence, port C is irregularly free. A Hot Connect group A or B connected to this port will not be put into operation by the master.

### 2.3.2.2 Fast-Hot-Connect

EtherCAT components that support Fast Hot Connect enable a faster fieldbus boot up following the establishment of a connection. The boot up depends in detail on the number of devices, the topology and activated Distributed Clocks. Whereas the normal establishment of a connection and communication takes several seconds, less than 1 second is possible with FHC components.

#### Properties and system behavior

- Fast Hot Connect is supported from TwinCAT 2.11R3 Build 2221.
- Fast Hot Connect ports are specially marked.



Fig. 36: Identification of FHC port at EK1122-0080 and EK1101-0080

- Standard EtherCAT devices may not be connected to Fast Hot Connect ports. This is to be ensured by measures on the application side, which is easy to implement by means of the topology change that is usually carried out mechanically in such applications.

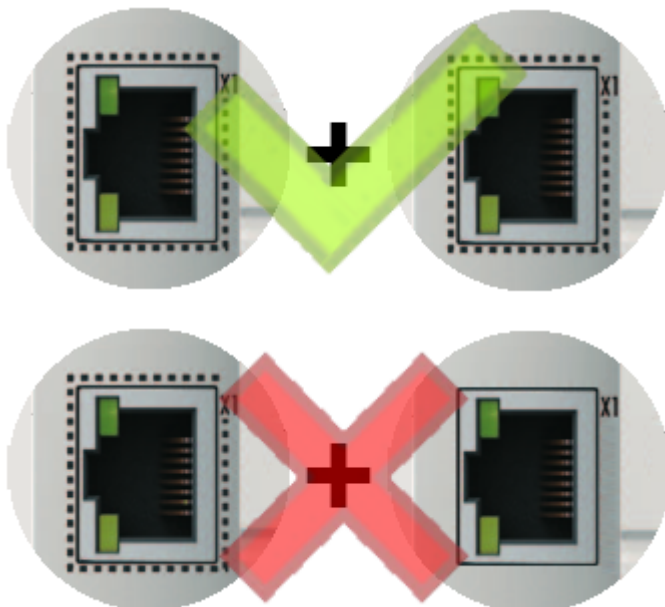


Fig. 37: Recommended combination of Ethernet ports

If corresponding ports are nevertheless connected, a power reset of the devices involved (branch terminal and coupler/box) is required.

- With Fast Hot Connect devices the establishment of an Ethernet connection is accelerated compared to the normal Fast Ethernet connection.  
If in addition the use of Distributed Clocks functions is omitted in the entire topology, then the resynchronization time of the components is also dispensed with. Group boot up of < 1 second is then possible, from plugging in the Ethernet connection to the OP state.
- An incorrect port allocation is detected in the TwinCAT ADS Logger

```

Message
'Term 17 (EK1122-0080)' detected invalid hot connect group at port 3
'Term 21 (EK1122)' detected invalid hot connect group at port 3. Only Fast-Hotconnect slaves are allowed at this port.
    
```

Fig. 38: Detection of incorrect port allocation in the TwinCAT logger.

**Configuration**

The configuration of Fast Hot Connect groups in the TwinCAT System Manager takes place in exactly the same way as Hot Connect groups, specifying the associated group ID.

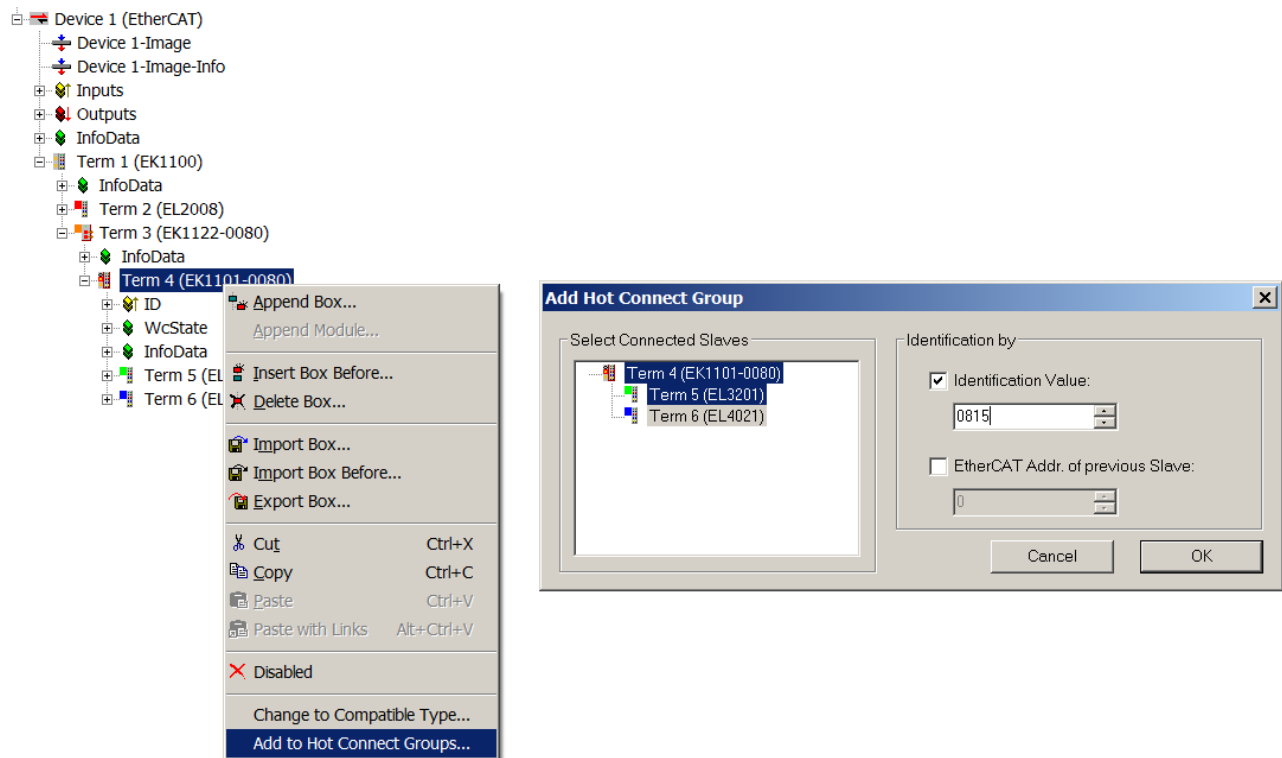


Fig. 39: Configuration of a Fast Hot Connect group

Corresponding Fast Hot Connect ports are marked red in the TwinCAT System Manager.

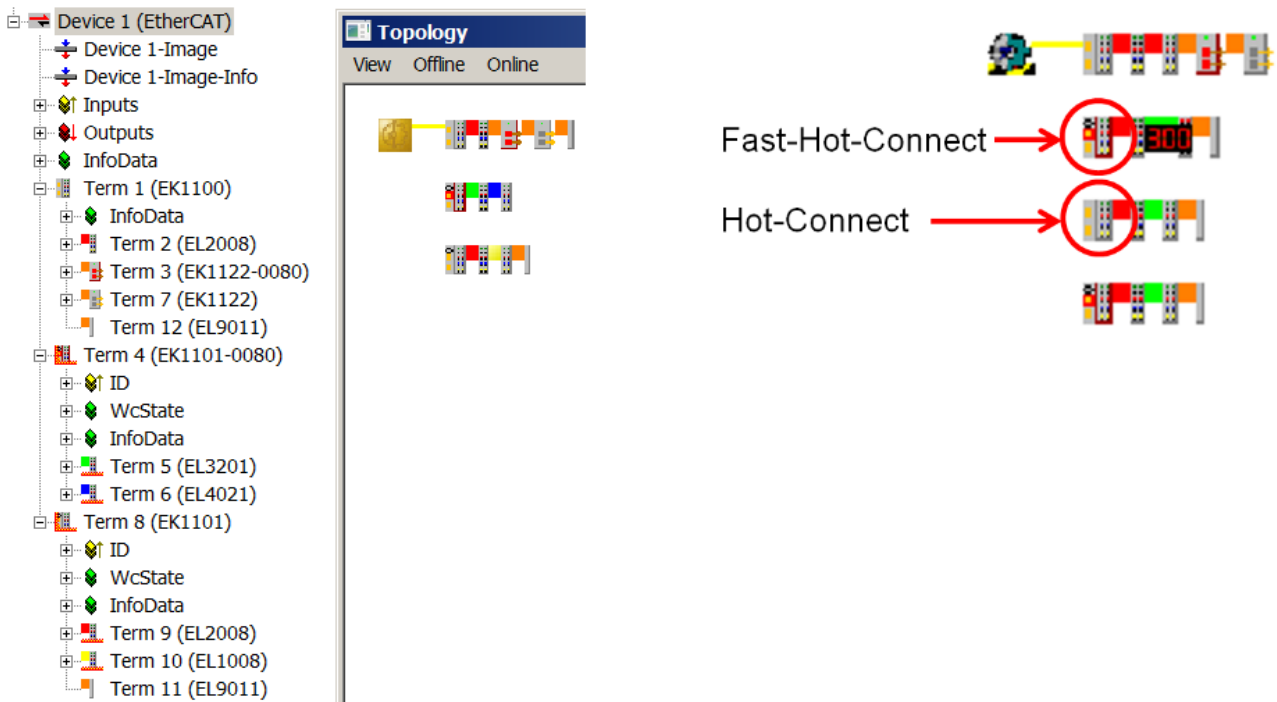


Fig. 40: Marking in the TwinCAT System Manager

A configuration of FHC groups is possible only if at least 1 corresponding junction is present e.g. EK1122-0080.

**Distributed Clocks**

If no Distributed Clocks functions are used, this is visible in the master settings by the absence of "DC in use":

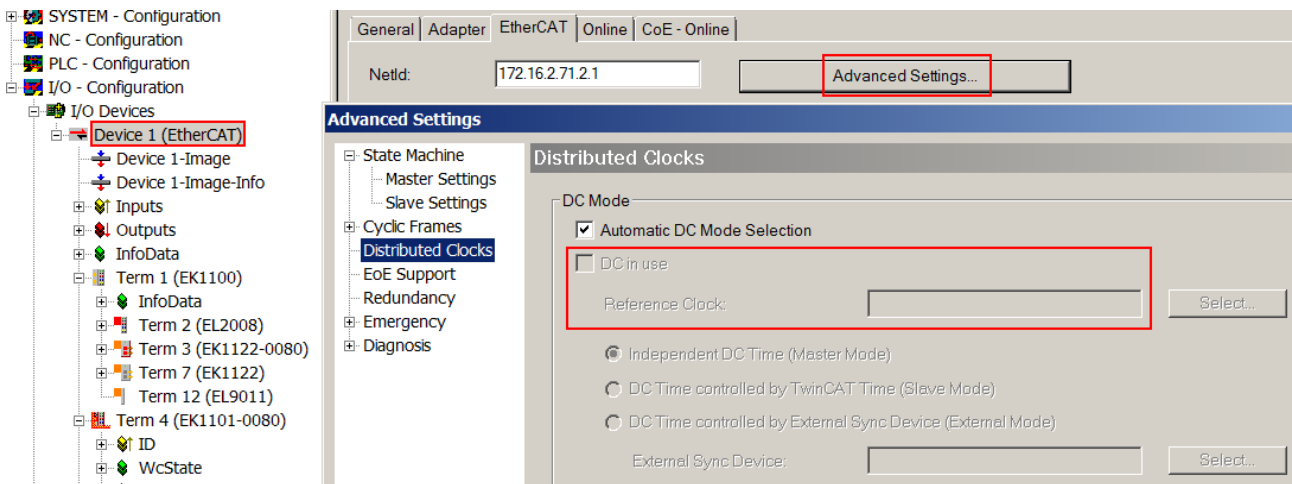


Fig. 41: DC master setting

This setting is automatically selected by the System Manager if there are no EtherCAT slaves in the configuration in which Distributed Clocks is activated. "DC in use" should not be randomly deactivated by the user, because otherwise these devices will no longer function.

### 2.3.2.3 Addressing

#### Addressing and identification

At present three different methods are defined for the explicit identification of Hot Connect stations:

- **SecondSlaveAddress (SSA)**  
 Here, the station address [0..65536] is permanently stored in the E<sup>2</sup>PROM of the EtherCAT slave. At the start the EtherCAT Slave Controller (ESC) loads this address into its register 0x0012, from where it can be read by the EtherCAT master.  
 This address is written
  - via the EtherCAT master/TwinCAT System Manager by the user during the system setup
  - via a user interface on the slave (keyboard, display, selector switch, etc.) at device start-up
- In almost all cases they are used for unambiguous identification of a device, rather than as a flexible station.  
 In case of exchange the current address must be stored again in the replacement device.  
 A changed SSA will usually only be accepted by the ESC after a power restart of the device.  
 The SSA can be checked by the user by reading the register 0x0012 from the PLC (see TcEtherCAT-lib -> FB\_EcPhysicalReadCmd)
- **InputWord/IdentificationValue/Data Word**  
 Here, a bit sequence corresponding to the station address [0..65535] is expected in the process data area of an EtherCAT slave and is interpreted by the master as the station address.  
 TwinCAT 2.10 expects the 16 bits data starting from register 0x1000. From TwinCAT 2.11R2 the information can be stored in any desired place in the slave; the EtherCAT master is informed of the storage location via the ESI description, called ADO (Address Offset) in the dialog (see Fig. *Default setting of an EtherCAT slave that supports the DataWord identification mode*).  
 The Input Word is usually displayed in the configuration as process data in the supporting slave (see EK1501, EK1101).
- **Explicit Device Identification**  
 On request by the master the slave informs the master of its ID [0 to 65535] via the AL status register 0x0134 during the start-up phase.  
 This method is supported from TwinCAT 2.11R3.

#### ● Addressing method

**I**

In general the ESI file belonging to the EtherCAT device contains the information regarding which addressing methods the slave supports. This is displayed in TwinCAT System Manager -> EtherCAT Slave -> Advanced Settings and in general should not be changed by the user.  
 See Fig. (*Default setting of an EtherCAT slave that supports the DataWord identification mode*) Exception: The SSA identification mode is possible in every slave that has an ESC EEPROM (refer to the manufacturer's notes). Beckhoff EtherCAT terminals/boxes usually have an EEPROM.

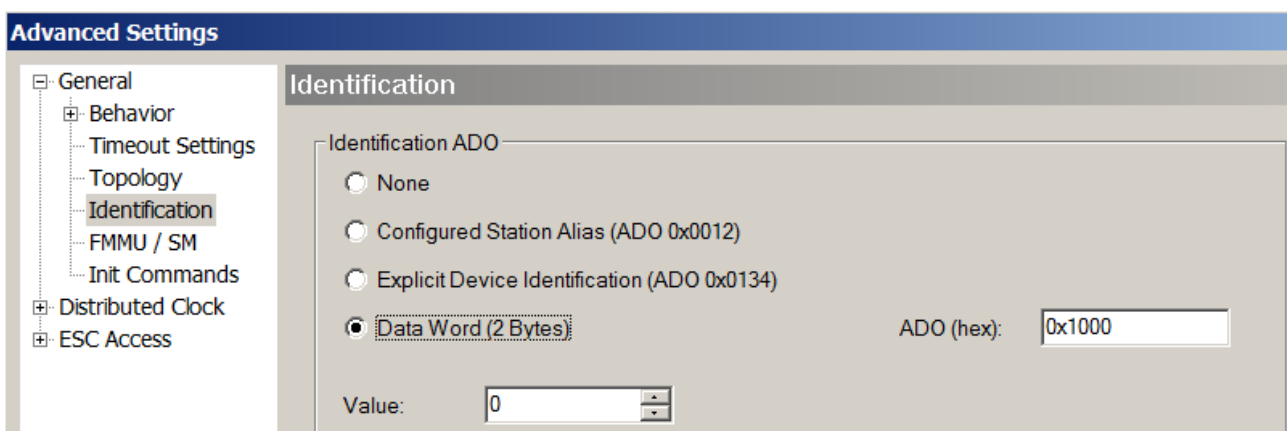


Fig. 42: Default setting of an EtherCAT slave that supports the DataWord identification mode

All EtherCAT slaves are always initially addressed via the auto-increment procedure, after which the EtherCAT master attempts to find the configured station addresses in the field and assign them to the stations/couplers. To this end the outgoing port is opened and closed specifically for the purpose of determining the IDs.

### Address allocation in the slave

#### Explicit Device Identification

The ID setting is to be carried out according to the device manual.

#### InputWord/IdentificationValue

As a rule, a selector switch accessible from the outside on such slaves allows the position to be set; the selection switch can be sealable or protected via a software mechanism. Other devices offer a user interface which allows ID setting.

Sample: Beckhoff EK1101 - free accessibility of the ID switch



Fig. 43: EK1101 with ID switch

#### SecondSlaveAddress

The SSA can be set by the master or via a user interface on the slave. The procedure for setting via the Beckhoff TwinCAT master is described below:

- Put the EtherCAT slave into operation in a simple configuration without addressing. The slave should be in OP, WorkingCounter = 0, no LostFrames
- Proceed via Slave -> EtherCAT -> Advanced Settings -> E<sup>2</sup>PROM to the *SecondAddress* dialog.
- The current address is displayed there; write the *New Second Address* in the slave E<sup>2</sup>PROM.
- The address is accepted after a power restart.
- In this example, the new address  $10_{\text{dec}}/0A_{\text{hex}}$  is set.

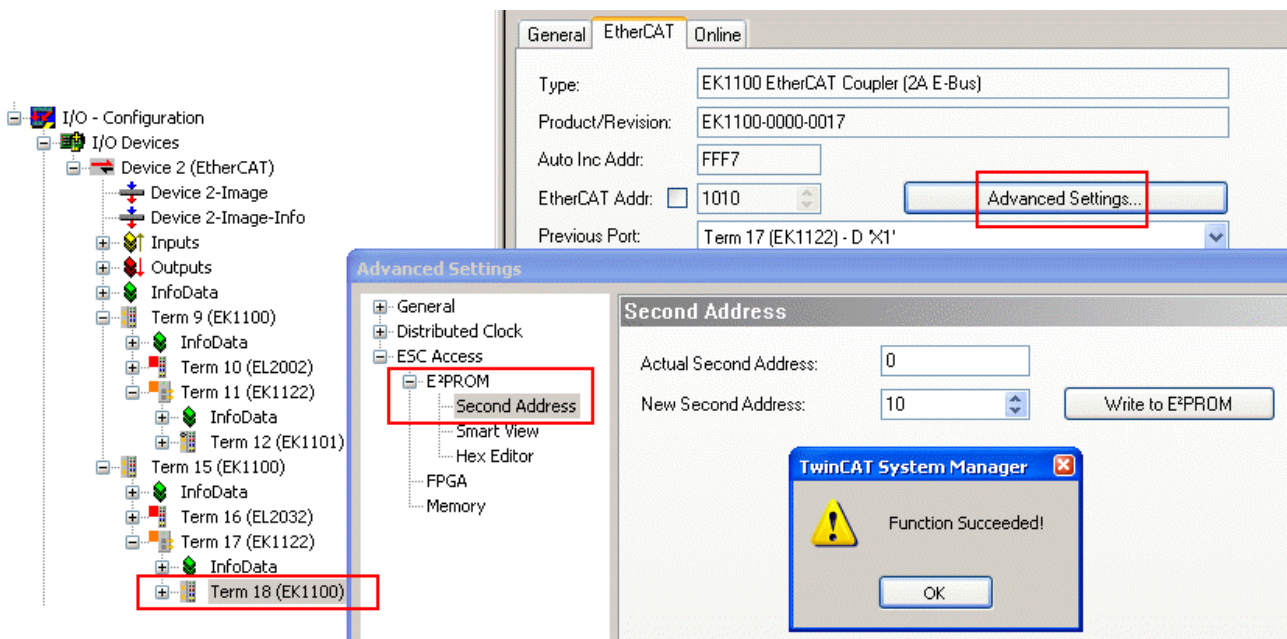


Fig. 44: setting the SSA

You can check whether the slave has accepted the address as follows:

Proceed in the same dialog to *Memory* and check the contents of register 0x0012.

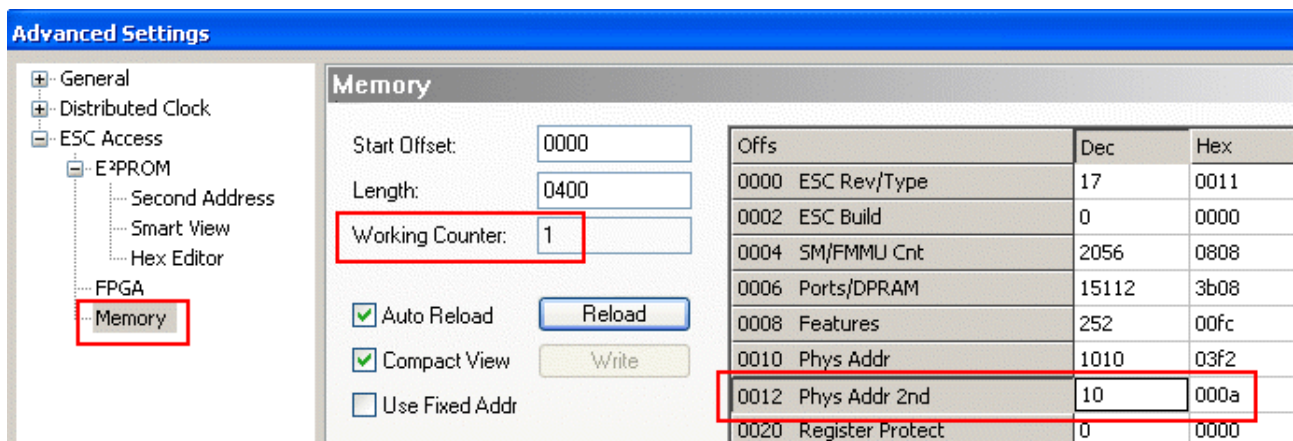


Fig. 45: Checking the SSA

If nothing is displayed:

- Normally Wc = 0 is the “error-free” state; in this case WorkingCounter must be 1, otherwise there will be no communication with the slave for other reasons.
- Check whether you are on the right slave.
- As can be seen by *Length* = 400, the dialog reads 400<sub>hex</sub> bytes from the ESC in one operation. Most third party ESCs do not support this; in this case, no data will be displayed in the columns. Try to find out from *Length* = 2, how many bytes your ESC supports.

**Note about TwinCAT 2.11**

The setting dialog for the SSA was slightly changed in TwinCAT 2.11, but retains the same properties.

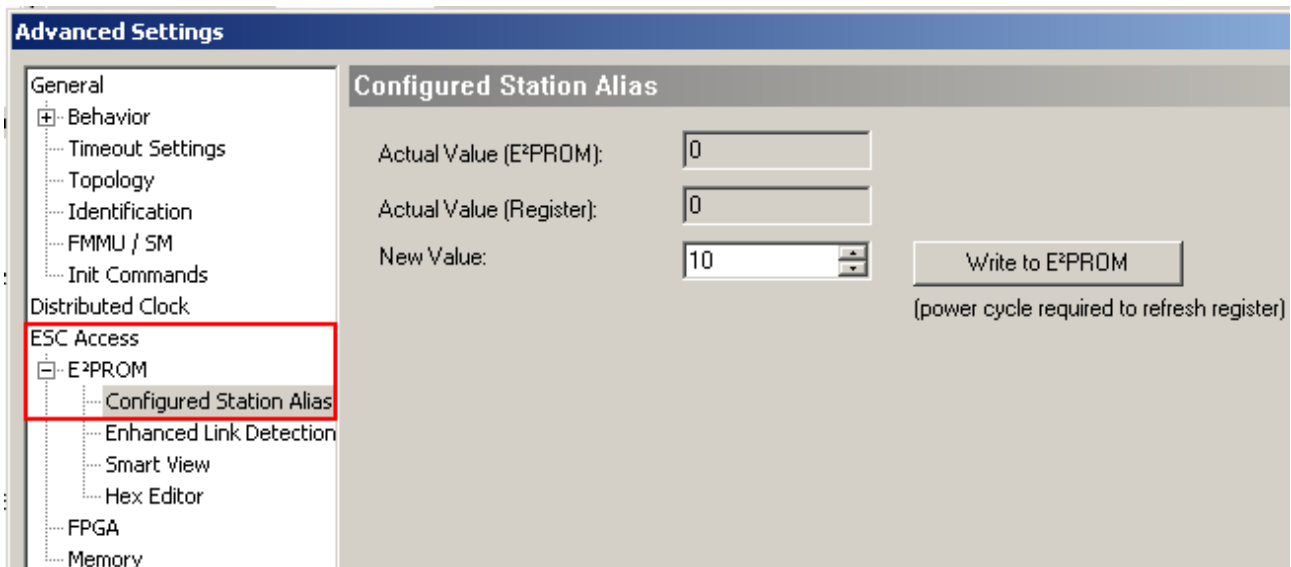


Fig. 46: Setting SSA under TwinCAT 2.11

**Compatible Beckhoff devices**

In general, every EtherCAT slave can be assigned a Hot Connect address, regardless of whether it is an IO terminal, a drive or a module of couplers and terminals.

**i Hot Connect capability**

Check whether the EtherCAT slave you are using is Hot Connect-capable. Beckhoff slaves currently support the function as follows (as of May 2012): (see following table)

Version	SecondSlaveAddress	InputWord	Explicit Device Identification
<b>EK1100</b>	from HW18	-	-
<b>EK1101, EK1501, EK1101-0080</b>	all	all	-
<b>BK1120</b>	from HW09	-	-
<b>EL terminals</b>	In general yes for most series from an XML version xxxx-xxxx-0016	-	-
<b>EP boxes</b>	In general yes for most series from an XML version xxxx-xxxx-0016	-	-
<b>AX2xxx</b>	-	-	-
<b>AX5xxx</b>	all	see documentation	see documentation

**2.3.2.4 Setup TwinCAT 2.10**

The configuration procedure effort under TwinCAT 2.10 is described below by means of an example.

**Step 1: creating the initial configuration**

First of all, all components must be present in the configuration; the components are then connected in the (offline) topology view in the order prescribed by the configuration or the “PreviousPort” specification. Shown here is the offline view of 4 stations with EK1100, output terminals (red) and an EK1122.



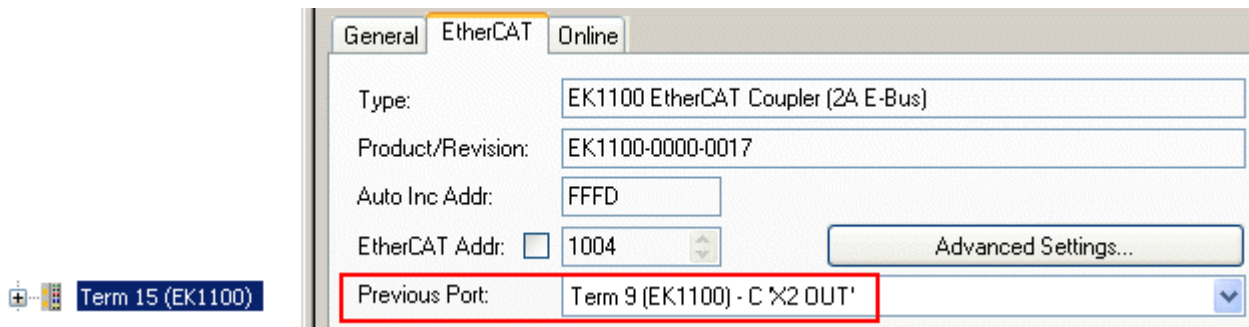


Fig. 47: Creating the configuration

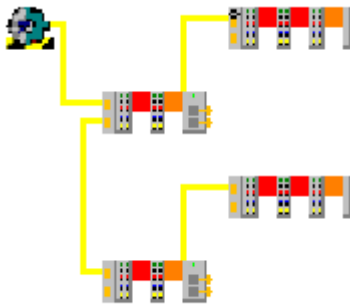


Fig. 48: Creating the configuration

**Step 2: Define as a HotConnect group**

Right clicking on the EtherCAT slave opens the HotConnect group dialogue.

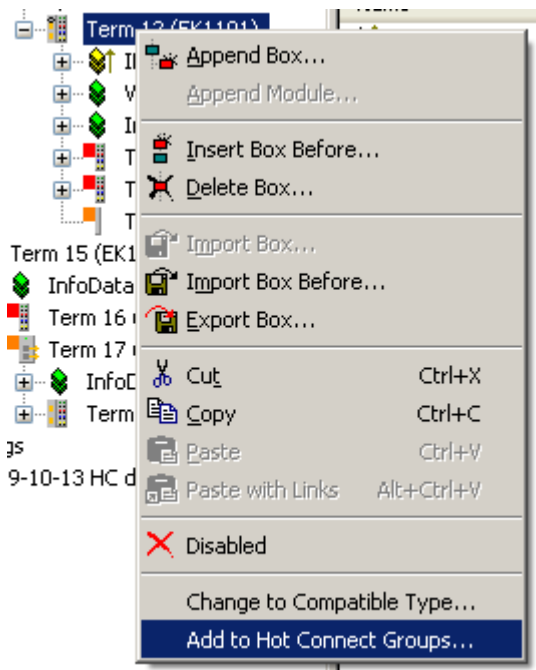


Fig. 49: Assign HotConnect group

Now enter the desired address, paying attention to the method used.

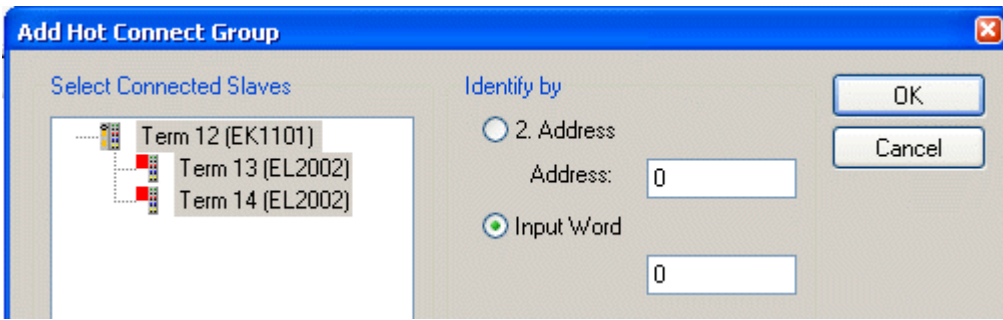


Fig. 50: Assign address

The success of the operation can only be seen in the configuration: the station is removed from the order and is appended below, marked in red. The property *“PreviousPort”* can no longer be selected, as TwinCAT is now expecting the station everywhere.

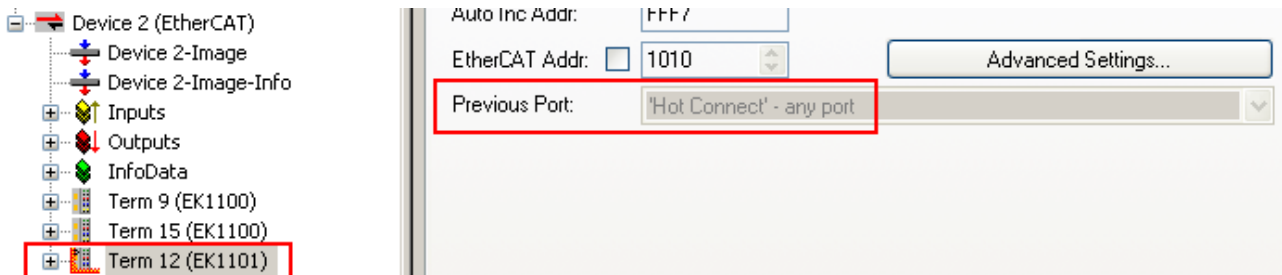


Fig. 51: Hot Connect group

The station similarly releases itself in the (offline) topology view and hangs without a connection in the air.

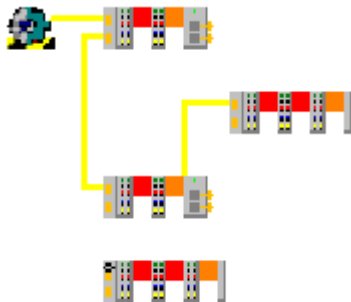


Fig. 52: Topology view

This also applies to all other HotConnect stations.

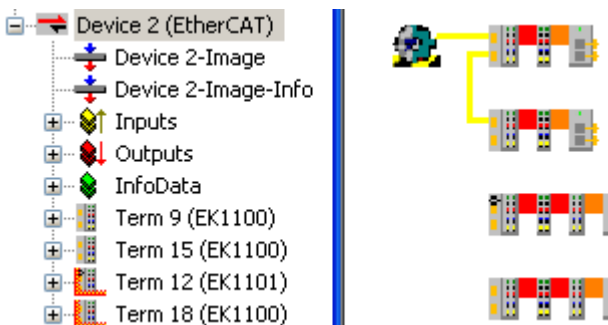


Fig. 53: Topology view

**Step 3: Online operation**

The actual arrangement can only be viewed in the **online** topology view if this configuration is in operation after activation.

Green bars over EtherCAT slaves display the OP state, red bars the INIT state.

In figure *Online topology view* it can be seen that HotConnect Station A was docked with the EK1122, whereas station B does not participate in the data traffic due to absence.

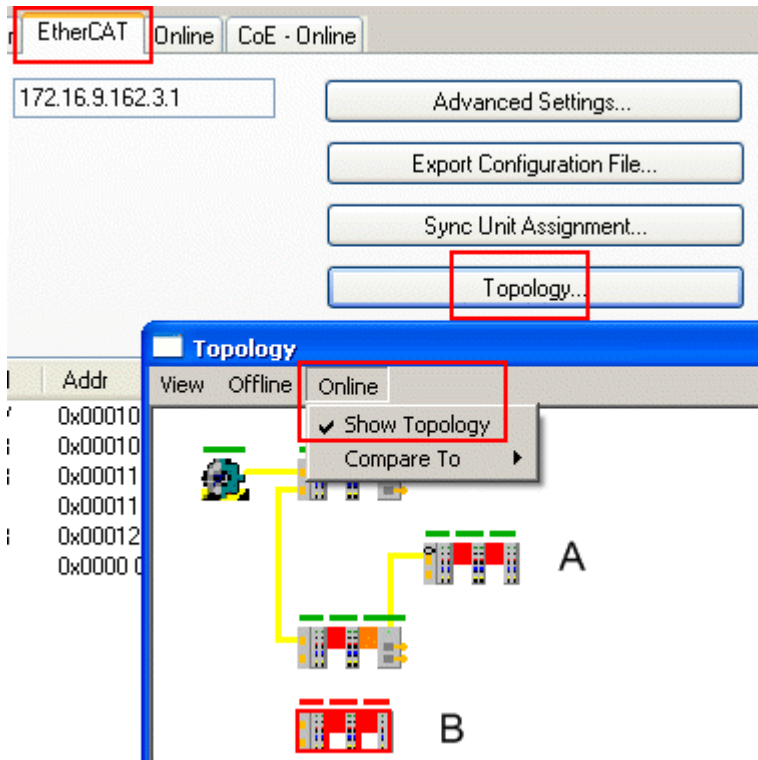


Fig. 54: Online topology view

These slaves are also correctly displayed by station B in the online display as not present.

No	Addr	Name	State	CRC
1	1001	Term 9 (EK1100)	OP	0, 0, 0
2	1002	Term 10 (EL2002)	OP	0, 0
3	1003	Term 11 (EK1122)	OP	0
4	1004	Term 15 (EK1100)	OP	0, 0
5	1005	Term 16 (EL2032)	OP	0, 0
6	1006	Term 17 (EK1122)	OP	0, 0
7	1007	Term 12 (EK1101)	OP	0, 0
8	1008	Term 13 (EL2002)	OP	0, 0
9	1009	Term 14 (EL2002)	OP	0
10	1010	Term 18 (EK1100)	INIT NO_COMM	
11	1011	Term 19 (EL2002)	INIT NO_COMM	
12	1012	Term 20 (EL2002)	INIT NO_COMM	

Fig. 55: Online display

The assigned address can also be subsequently changed via the *HotConnect* tab.

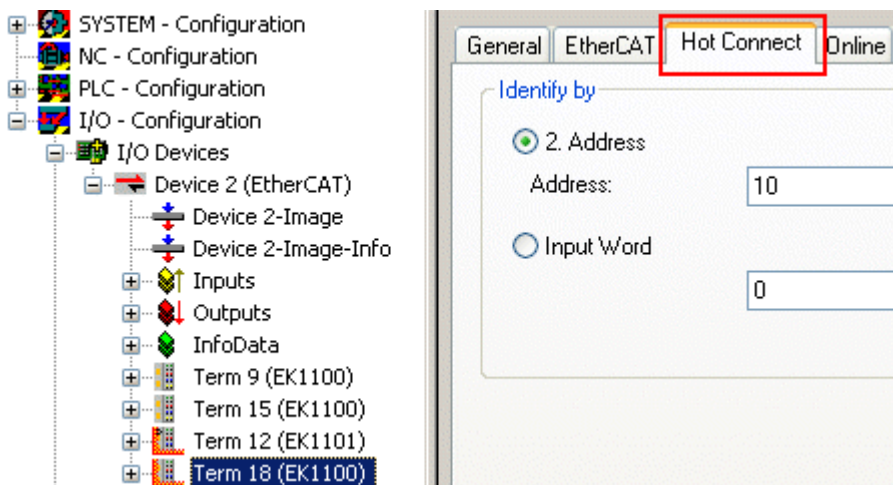


Fig. 56: HotConnect tab

### 2.3.2.5 Setup TwinCAT 2.11R2

The configuration dialogs under TwinCAT 2.11R2 are different, although the principle is the same as for TwinCAT 2.10.

#### Configuration parameterization

Alternatively, the EtherCAT configuration can be set up via an online scan or manually, as described below.

1. First set up a maximum configuration, i.e. the configuration should cover all bus coupler stations that could be used in the system, even if they are not all operated *at the same time*. The order and the associated "Previous Port" parameter are irrelevant.

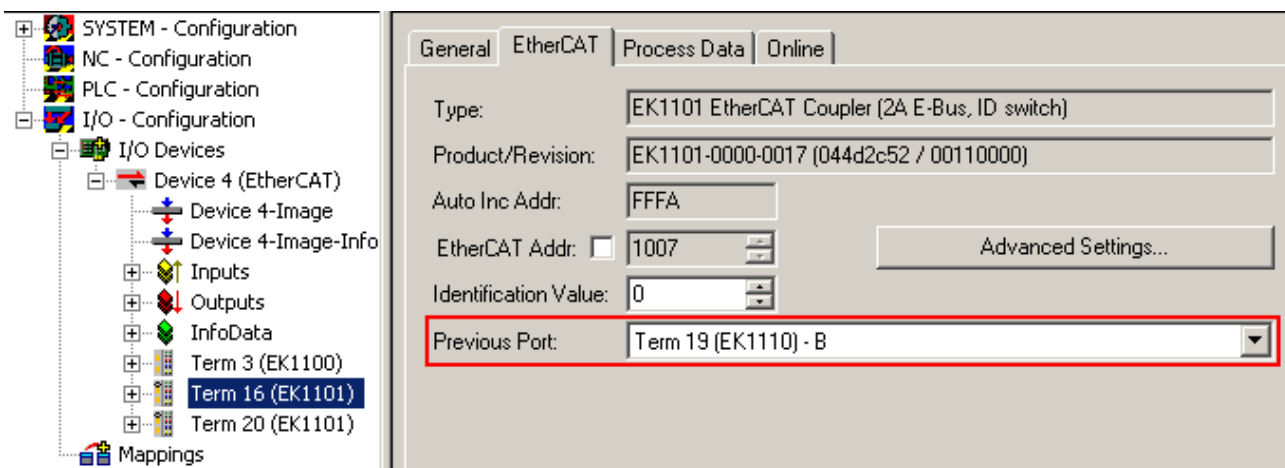


Fig. 57: PreviousPort at bus station

#### HotConnect units

Only units that communicate via Ethernet may be assigned an EtherCAT ID, because only these units can be connected or disconnected during operation. These are devices with RJ45/optical fiber/M8/M12 Ethernet port (couplers, EP boxes, drives, ...). Terminals may not be connected or disconnected while live. Specification as flexible HotConnect group therefore makes no sense. Terminals may be used for unambiguous identification via SSA.

2. The designated flexible bus stations are assigned the HotConnect characteristic in the System Manager. The dialog is called up by right-clicking on a coupler station/EP box.

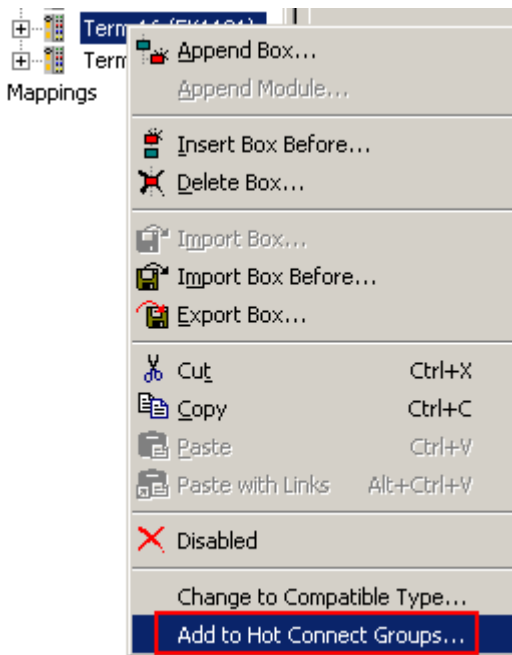


Fig. 58: Calling up the HotConnect dialog

3. In TwinCAT 2.11R2 a coupled bus station can be identified as the “right” one via two properties:
  - the ID “identification value”
  - the EtherCAT address of the previous slave

Activate the associated characteristic and specify the value [0; 65535]. The ID refers to identification through **InputWord** and **SecondSlaveAddress**. The default setting is InputWord. To use SSA proceed as follows.

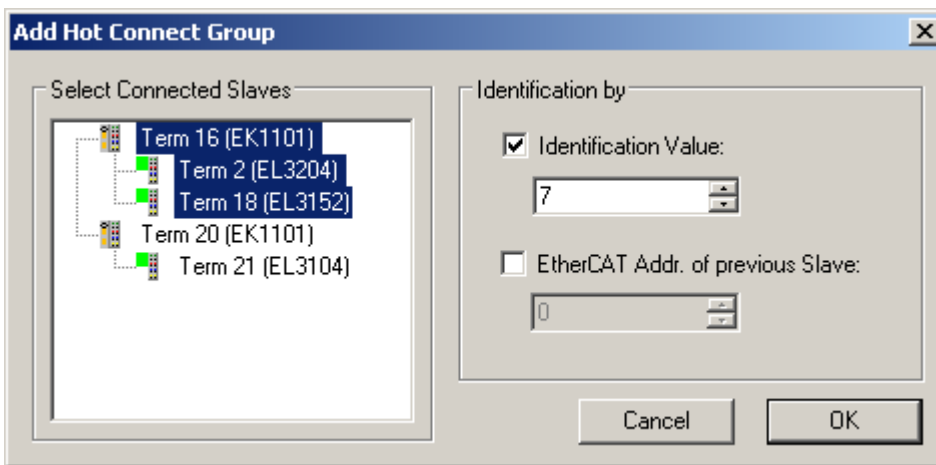


Fig. 59: HotConnect features

**● Identification via “previous slave”**

**i** If “previous slave” is specified as the address, a bus station parameterized in this way can no longer be inserted and operated at any port, but only at the specific EtherCAT port that follows directly after the specified slave. This method therefore significantly restricts the location of such a station. “Previous” refers to the preceding slave in terms of the EtherCAT communication, not necessarily the actual order in the topology. This is particularly relevant for the EK1122/EK1521 multi-port slaves.

4. The station then has an additional “HotConnect” tab, in which the attributes can be modified.

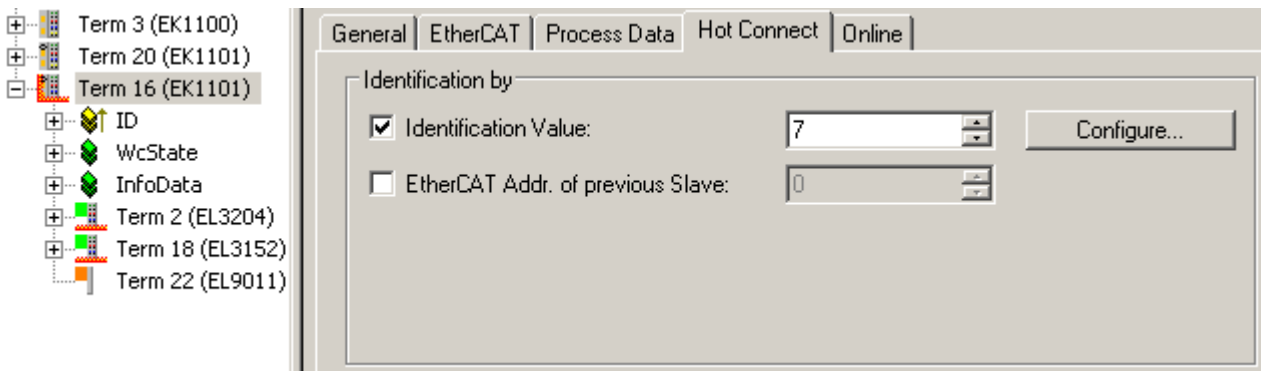


Fig. 60: HotConnect features

Under Properties, the InputWord default identification (from the “DataWord” dialog under process data) can be changed to SecondSlaveAddress via *Configure* (in dialog “Configured Station Alias” from slave register 12<sub>hex</sub>).

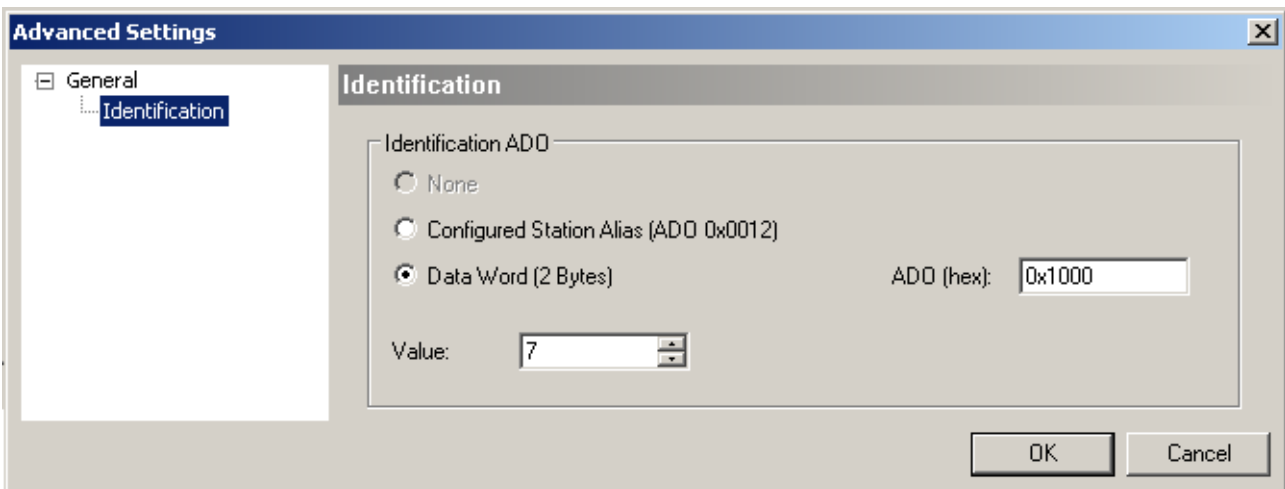


Fig. 61: Setting detail

In the System Manager the device is identified with a red HotConnect icon.

**i Identification via “Data Word”**

The 16 bit identification data are read from the slave at a specified location. TwinCAT 2.10 only supports register 1000<sub>hex</sub>. TwinCAT 2.11 can read from any location, as shown in the dialog in Fig. *Setting detail* “ADO (hex)”. The default setting is register 1000<sub>hex</sub> or the specification from the ESI file.

**HotConnect operation**

For HotConnect operation TwinCAT must be in Config/FreeRun or RUN mode. After changes in the configuration (other ID, ID location etc.) the configuration must be reloaded or TwinCAT restarted.

**Sample 1: Groups not identified but linked**

If stations that have an ID are connected via Ethernet but are not yet identified, the respective slaves come up with status messages:

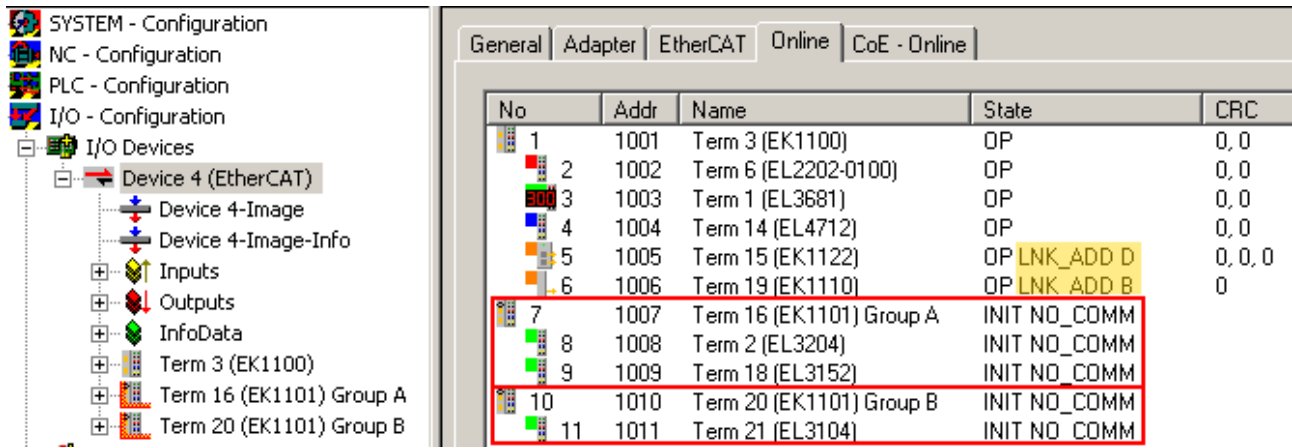


Fig. 62: Status messages

- “LNK\_ADD” together with the port information is shown for the slaves at which the stations were connected: this indicates an unsuspected LINK to unidentified devices.
- The stations themselves (in this case Group A, Group B) are still in INIT state. No communication is possible.

TwinCAT will keep trying to identify the coupled devices.

**Sample 2: Group identified**

The EK1101 “Group A” is now set to the right ID via the ID switches. After a few seconds TwinCAT recognizes this, includes the group in the process data traffic and changes the devices to OP state.

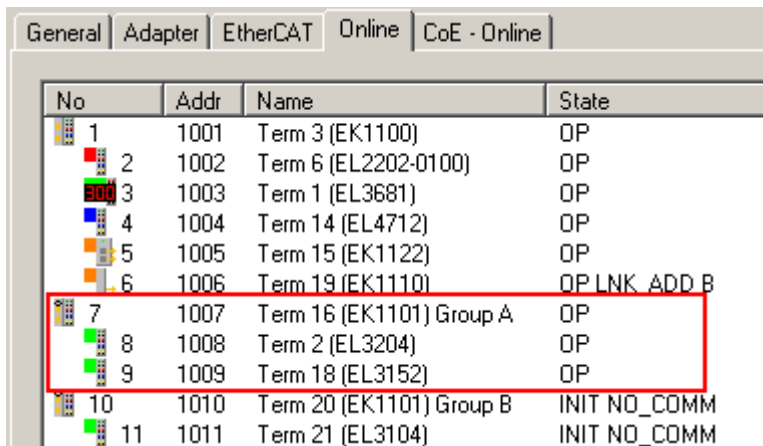


Fig. 63: Group A in operation

The control system indicates this via the process data (WC state, state) of the coupler:

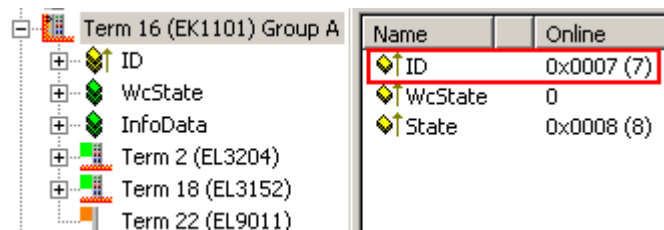


Fig. 64: Process data of the ID module

**Sample 3: Utilization of the topology view**

The online topology view reflects the current actual link status and coupling location.

The offline view shows the ID stations without communication connection, since the connection locations are unknown.

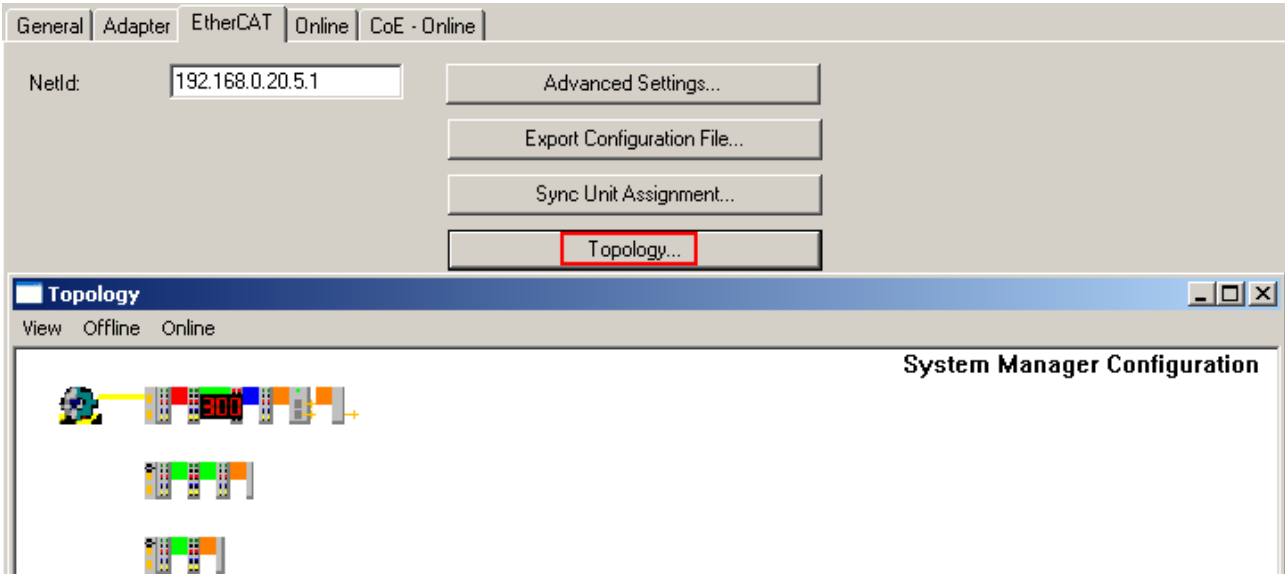


Fig. 65: Offline view

Stations that are not identified or present are shown in red in the online view.

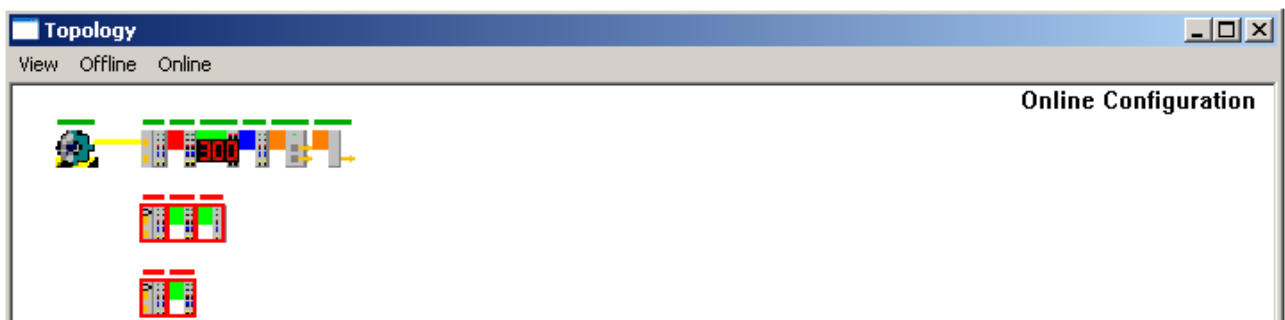


Fig. 66: Online view

If stations are activated by changing the ID (here: switch setting at EK1101) the green OP bar appears.



Fig. 67: Online view

Switching to offline view and back updates the current connection status.





Fig. 68: Current connection status

### 2.3.2.6 Identification

If HotConnect is used, the current topology status can be viewed in the online topology view in the system manager, i.e. which HotConnect group is connected and where. Several ADS accesses are necessary in order to access this information from the controller/PLC.

<https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/2469147275.zip> (PLC and System Manager) takes the following approach:

- a function block contains the necessary code
- the FB can be instanced several times and must be linked with its I/O variables to a HotConnect head (EK coupler, EP-box). As a result of this it receives information about the EtherCAT master and its own ADS port.
- As soon as the Wc (Working Counter) reports operation, the device's own data are queried from the EtherCAT master via ADS, as are the data of the devices that directly follow.
- In addition the identities and names of the neighboring devices are determined. The identifier is the EtherCAT address, which remains constant.

This <https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/2469149451.zip> also contains a visualization template that uses placeholder variables.

#### ● EtherCAT topology in the PLC

**i** In the introductory part of the documentation for the TcEtherCAT.lib there is an extensive sample program for determining various items of information about an EtherCAT network. This documentation is to be found in the information system (online or on DVD): TwinCAT --> TwinCAT PLC --> PC Libraries --> EtherCAT --> Overview.

This sample program uses function blocks from there.

The following structure is assumed:

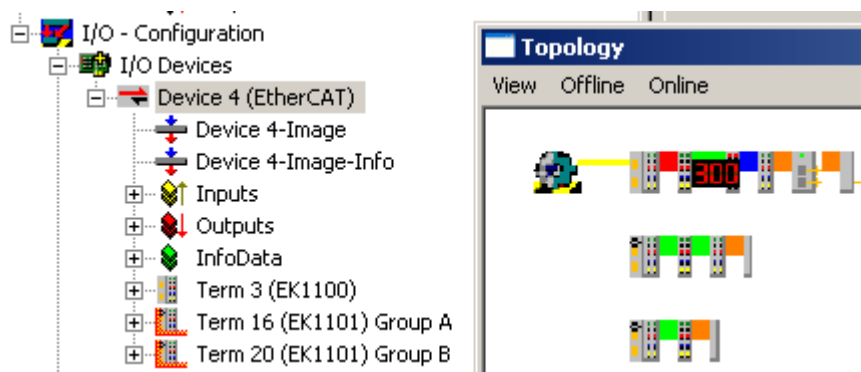


Fig. 69: Demo structure

The first EK1100 coupler is equipped with EK1110 and EK1122 as branch stations for the EK1101s *GroupA* and *GroupB*.

Regarding the procedure: the linking of the address information into the function block is crucial, see Fig. *AdsAddr*.

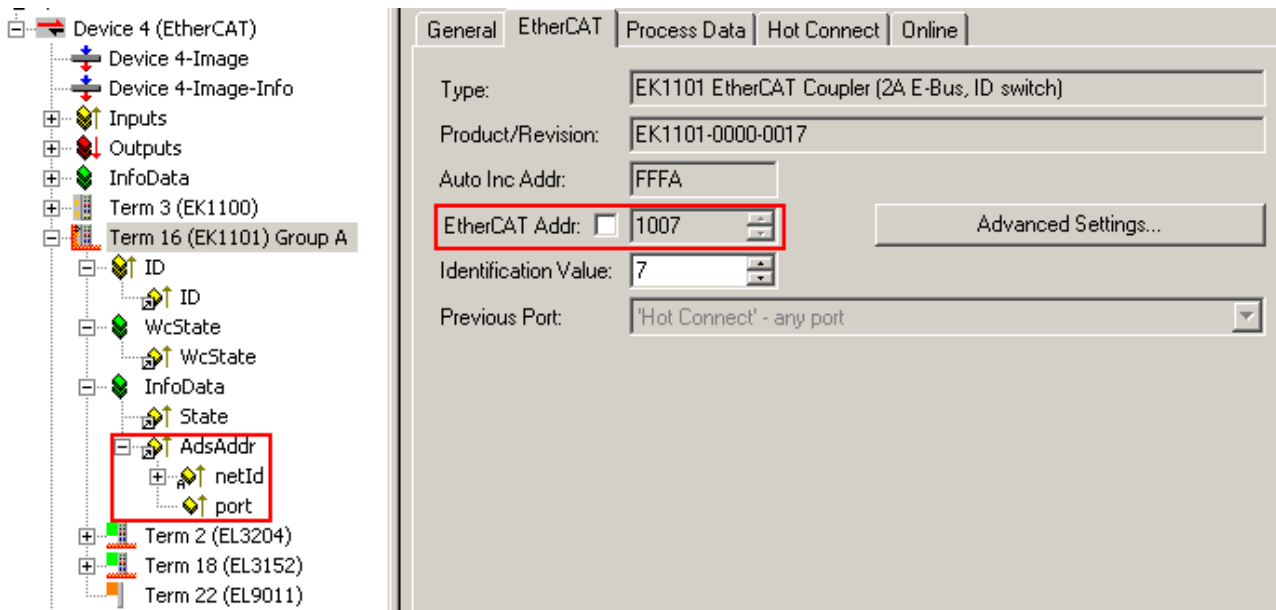


Fig. 70: AdsAddr

In the example the EK1101 HotConnect coupler has the EtherCAT address 1007. The FB can also access the EtherCAT Master NetId and the ADS Port = EtherCAT address via the link.

**ADS address**

**i** If the ADS address is not displayed in the green InfoData, the display can be switched on in the Advanced Settings for the slave. (See Fig. "Switching on the "ADS Address" in the Advanced Settings")

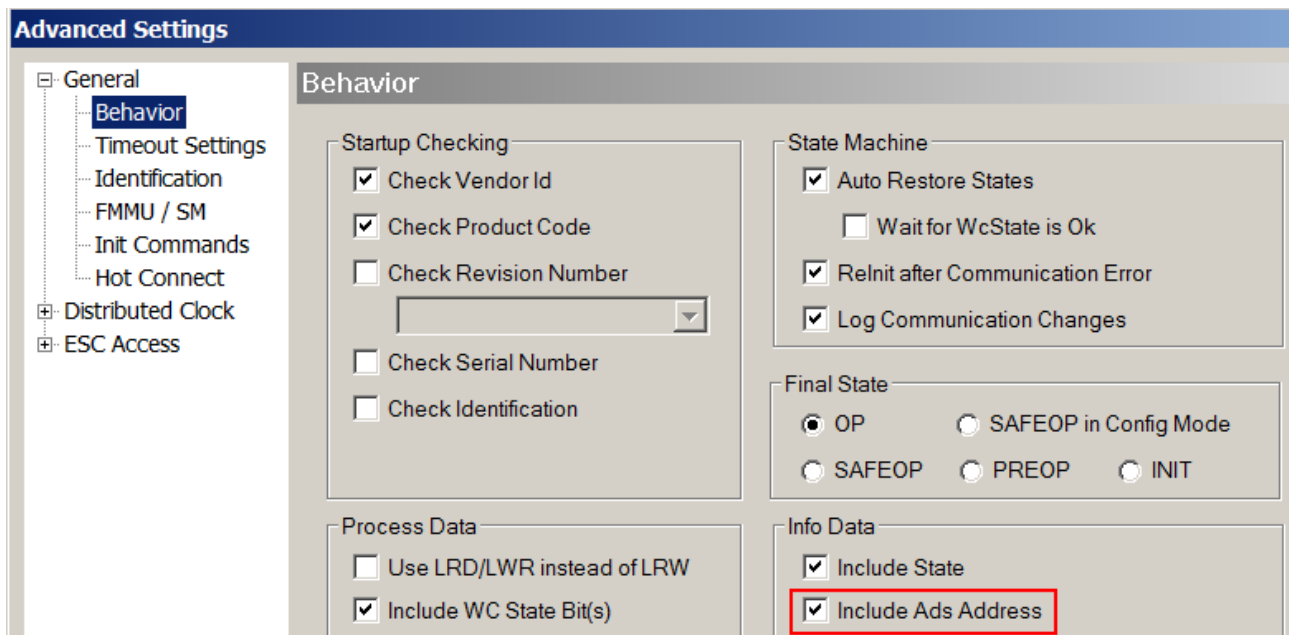


Fig. 71: Switching on the "ADS Address" in the Advanced Settings

The function block outputs a status structure, which supplies information about the own identities of the linked coupler and the connected devices; see Fig. *Information structure*. Since an EtherCAT Slave has a maximum of 4 ports, a maximum of 4 neighboring devices can be determined. The device names are taken from the configuration.

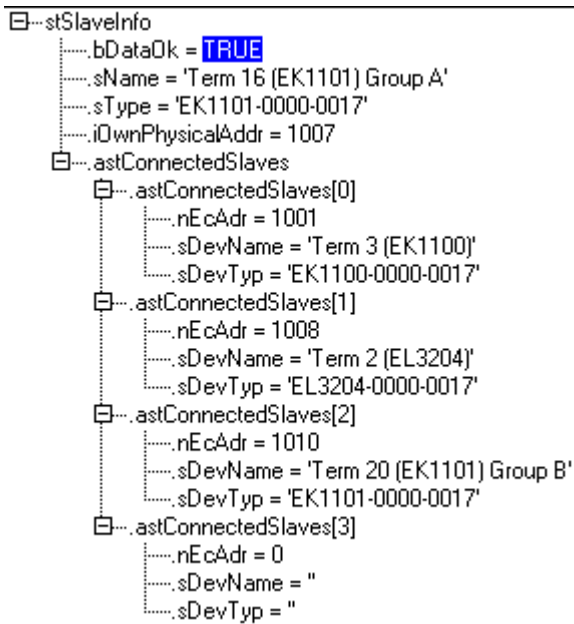


Fig. 72: Information structure

Most important for self-orientation is the predecessor slave on port A/0.

Via this mechanism an entire topology can also be determined from the PLC.

### 2.3.3 EtherCAT data exchange

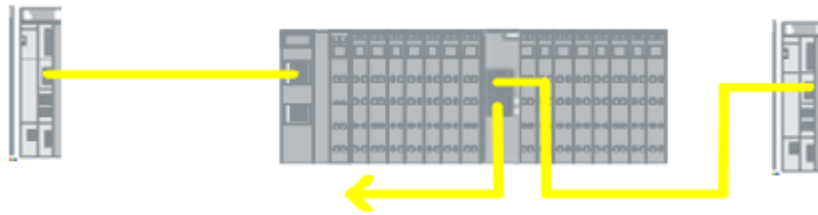
A range of components are available for implementing direct synchronous data exchange between two EtherCAT systems. Depending on the application requirements, the appropriate method can be selected based on the following criteria. The characteristic features are:

- Synchronous data exchange with predefined process data that are specified in the configuration
- Asynchronous data exchange
- Support for ADS over EtherCAT (AoE)
- Support for synchronization of the distributed clocks (DC) in the two systems

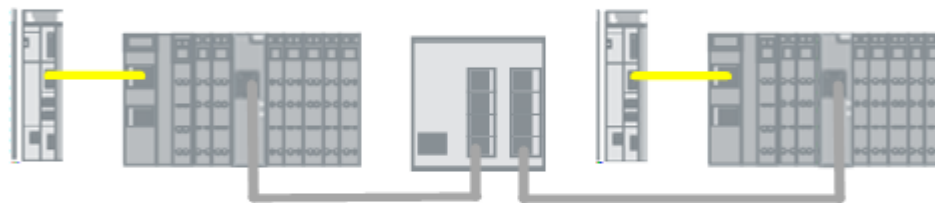
Some characteristics are listed in the following table. This information is only intended as a guide. The actual specification can be found in the respective [online](#) component documentation.

	EL6692	Publisher/Sub- subscriber	EL6601	FC1100	CX50x0-B110
Maximum synchronous data quantity	480 bytes, bidirectional	variable	1024 bytes, bidirectional (publisher/subscriber method)	1024 bytes, bidirectional	
Maximum asynchronous data quantity	-	-	variable	-	
AoE support	yes	yes	-	yes	yes
DC support	yes	-	-	-	-
Note	- recommended for synchronization of EtherCAT systems - TwinCAT 2.11 required	- use of an Ethernet port in both systems as real-time device - recommended for synchronous data exchange	- transfer of the RT devices into an EtherCAT terminal - recommended for synchronous data exchange	- requires free PCI slot in the IPC - TwinCAT 2.11 R2 required	- CX5000 as subordinate autonomous controller with its own IOs is integrated in the higher-level system as an EtherCAT slave - option B110 required - TwinCAT 2.11 R2 required

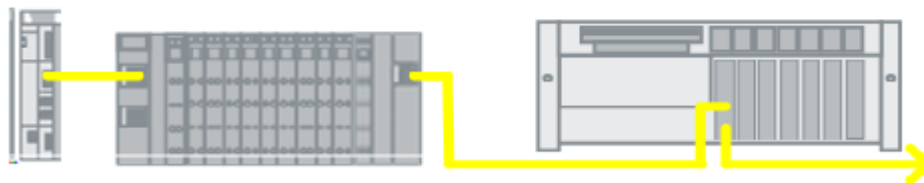
*EL6692 Bridge Terminal*  
 DC synchronization  
 480 bytes cyclic data transport  
 AoE



*EL6601 Switchport Terminal*  
 Publisher/Subscriber cyclic data transport  
 AoE



*FC1100 PCI card*  
 1024 bytes cyclic data transport  
 AoE



*Cx50x2-B110*



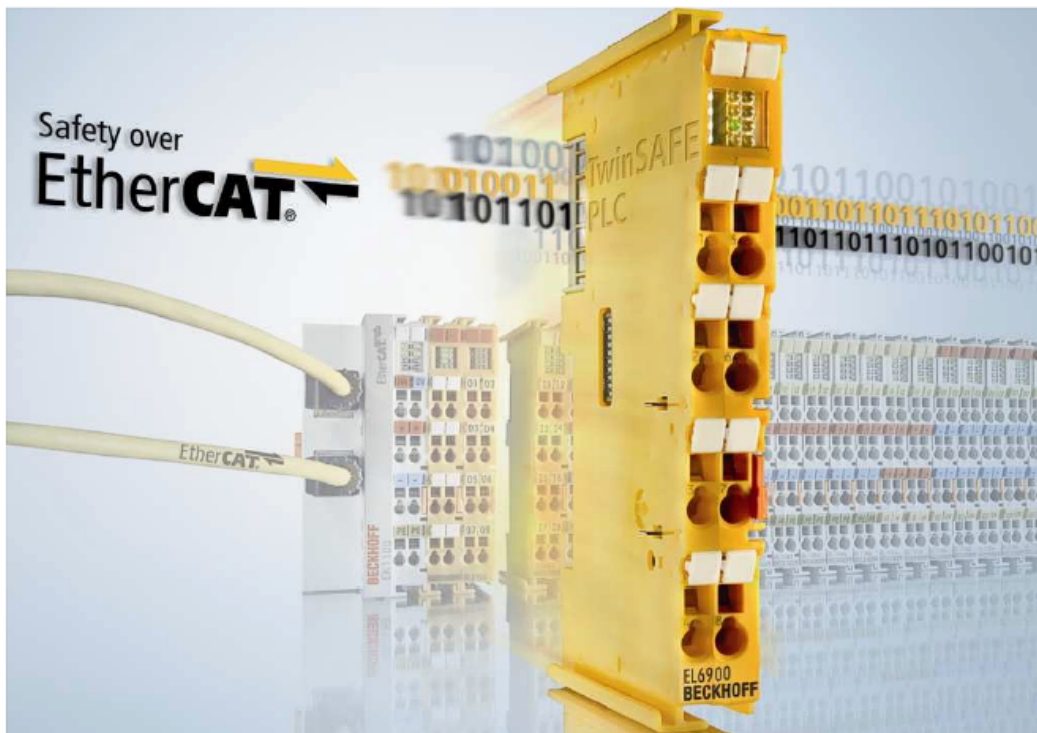
Fig. 73: Topologies for different data exchange methods

### 2.3.4 Safety over EtherCAT - TwinSAFE

The Beckhoff EtherCAT safety products use the TwinSAFE protocol, which is transported as part of the data within the EtherCAT datagrams. The TwinSAFE protocol is basically independent of the communication channel and has its own error detection mechanisms.

In the EtherCAT system an EL6900 logic terminal deals with the safety functions and transfers data to the input and output terminals (EL1904, EL2904) via the TwinSAFE protocol.

For the application and commissioning, the corresponding device documentation and a TwinSAFE application manual are accessible in the [Download section](#) on the Beckhoff website.



Applikationshandbuch

# TwinSAFE

Version: 1.1.0  
Datum: 22.03.2011



**BECKHOFF**

Fig. 74: TwinSAFE application manual

## 3 Setup in the TwinCAT System Manager

### 3.1 TwinCAT Quick Start

TwinCAT is a development environment for real-time control including multi-PLC system, NC axis control, programming and operation. The whole system is mapped through this environment and enables access to a programming environment (including compilation) for the controller. Individual digital or analog inputs or outputs can also be read or written directly, in order to verify their functionality, for example.

For further information please refer to <http://infosys.beckhoff.com>:

- **EtherCAT Systemmanual:**  
Fieldbus Components → EtherCAT Terminals → EtherCAT System Documentation → Setup in the TwinCAT System Manager
- **TwinCAT 2** → TwinCAT System Manager → I/O - Configuration
- In particular, TwinCAT driver installation:  
**Fieldbus components** → Fieldbus Cards and Switches → FC900x – PCI Cards for Ethernet → Installation

Devices contain the terminals for the actual configuration. All configuration data can be entered directly via editor functions (offline) or via the “Scan” function (online):

- **“offline”**: The configuration can be customized by adding and positioning individual components. These can be selected from a directory and configured.
  - The procedure for offline mode can be found under <http://infosys.beckhoff.com>:  
**TwinCAT 2** → TwinCAT System Manager → IO - Configuration → Adding an I/O Device
- **“online”**: The existing hardware configuration is read
  - See also <http://infosys.beckhoff.com>:  
**Fieldbus components** → Fieldbus cards and switches → FC900x – PCI Cards for Ethernet → Installation → Searching for devices

The following relationship is envisaged from user PC to the individual control elements:

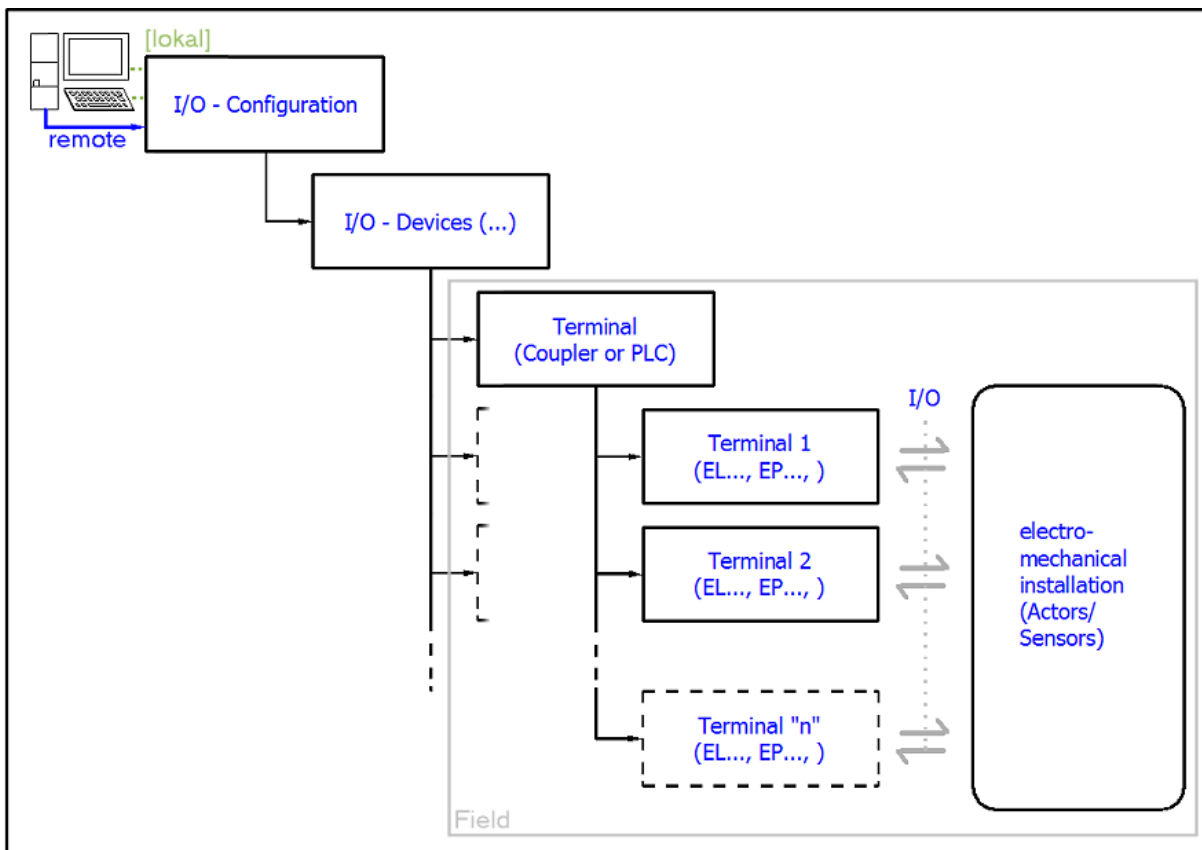


Fig. 75: Relationship between user side (commissioning) and installation

The user inserting of certain components (I/O device, terminal, box...) is the same in TwinCAT 2 and TwinCAT 3. The descriptions below relate to the online procedure.

### Sample configuration (actual configuration)

Based on the following sample configuration, the subsequent subsections describe the procedure for TwinCAT 2 and TwinCAT 3:

- Control system (PLC) **CX2040** including **CX2100-0004** power supply unit
- Connected to the CX2040 on the right (E-bus):  
**EL1004** (4-channel digital input terminal 24 V<sub>DC</sub>)
- Linked via the X001 port (RJ-45): **EK1100** EtherCAT Coupler
- Connected to the EK1100 EtherCAT coupler on the right (E-bus):  
**EL2008** (8-channel digital output terminal 24 V<sub>DC</sub>; 0.5 A)
- (Optional via X000: a link to an external PC for the user interface)



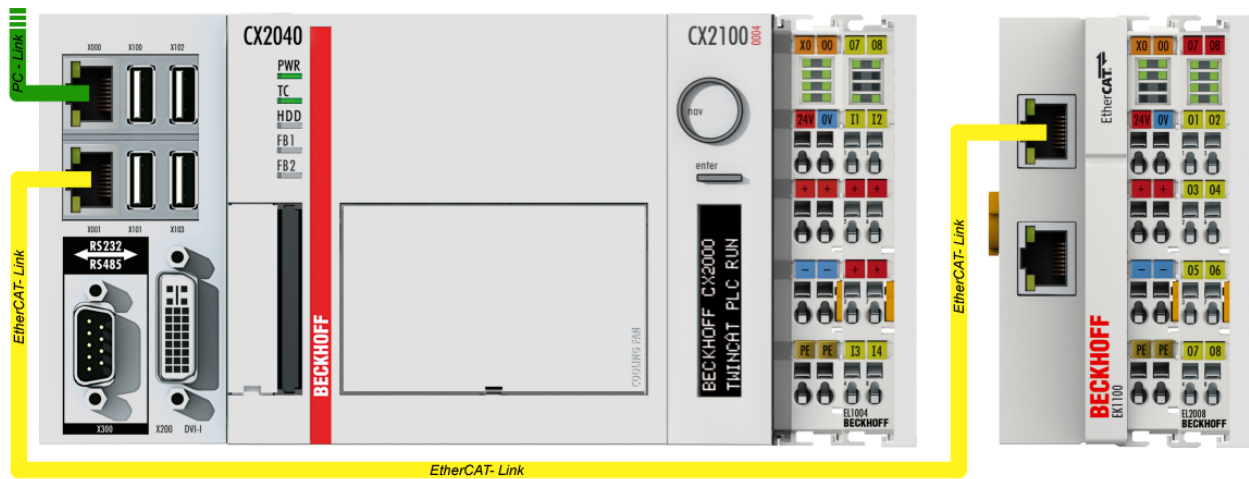


Fig. 76: Control configuration with Embedded PC, input (EL1004) and output (EL2008)

Note that all combinations of a configuration are possible; for example, the EL1004 terminal could also be connected after the coupler, or the EL2008 terminal could additionally be connected to the CX2040 on the right, in which case the EK1100 coupler wouldn't be necessary.

### 3.1.1 TwinCAT 2

#### Startup

TwinCAT basically uses two user interfaces: the TwinCAT System Manager for communication with the electromechanical components and TwinCAT PLC Control for the development and compilation of a controller. The starting point is the TwinCAT System Manager.

After successful installation of the TwinCAT system on the PC to be used for development, the TwinCAT 2 System Manager displays the following user interface after startup:

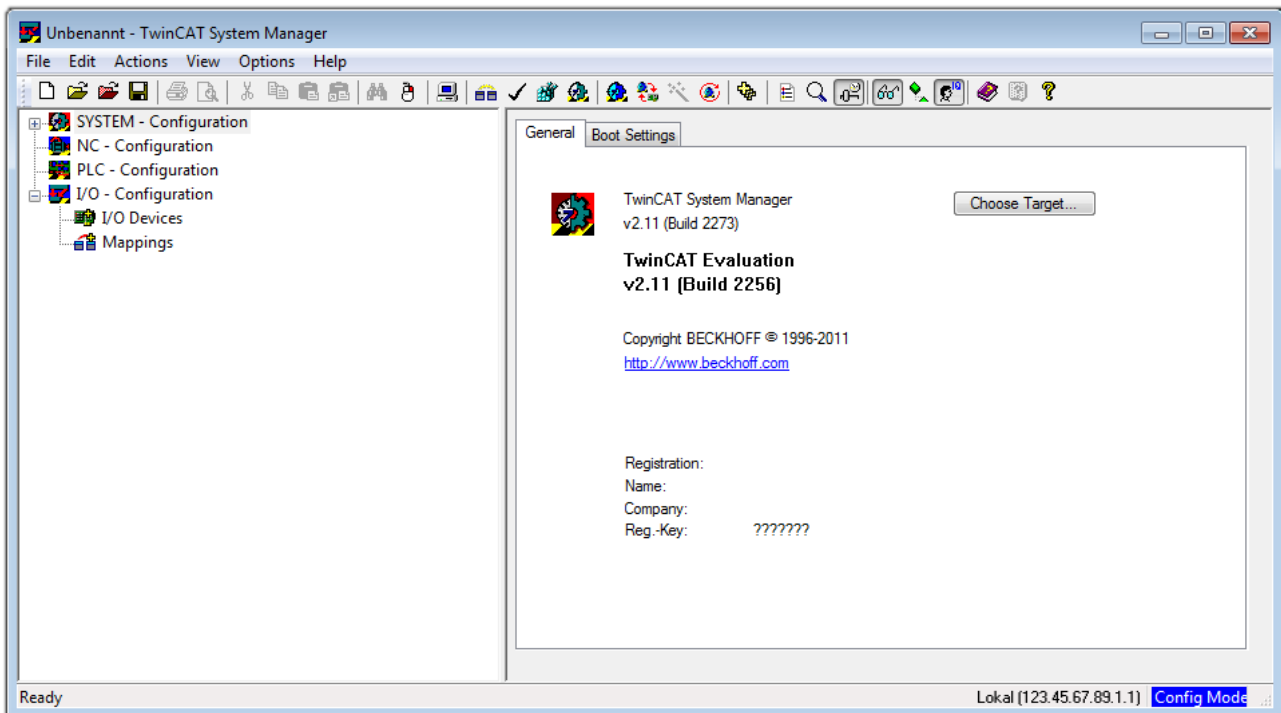



Fig. 77: Initial TwinCAT 2 user interface

Generally, TwinCAT can be used in local or remote mode. Once the TwinCAT system including the user interface (standard) is installed on the respective PLC, TwinCAT can be used in local mode and thereby the next step is “[Insert Device](#) [▶ 68]”.

If the intention is to address the TwinCAT runtime environment installed on a PLC as development environment remotely from another system, the target system must be made known first. In the menu under

“Actions” → “Choose Target System...”, via the symbol “” or the “F8” key, open the following window:

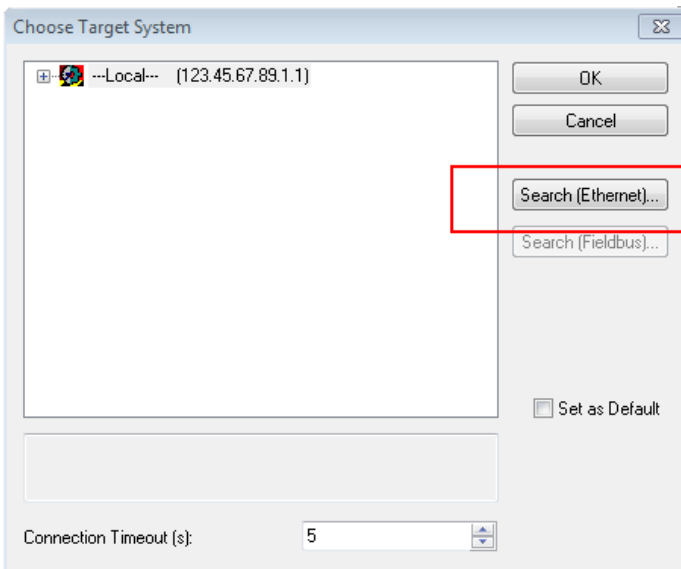


Fig. 78: Selection of the target system

Use “Search (Ethernet)...” to enter the target system. Thus a next dialog opens to either:

- enter the known computer name after “Enter Host Name / IP:” (as shown in red)
- perform a “Broadcast Search” (if the exact computer name is not known)
- enter the known computer IP or AmsNetID.

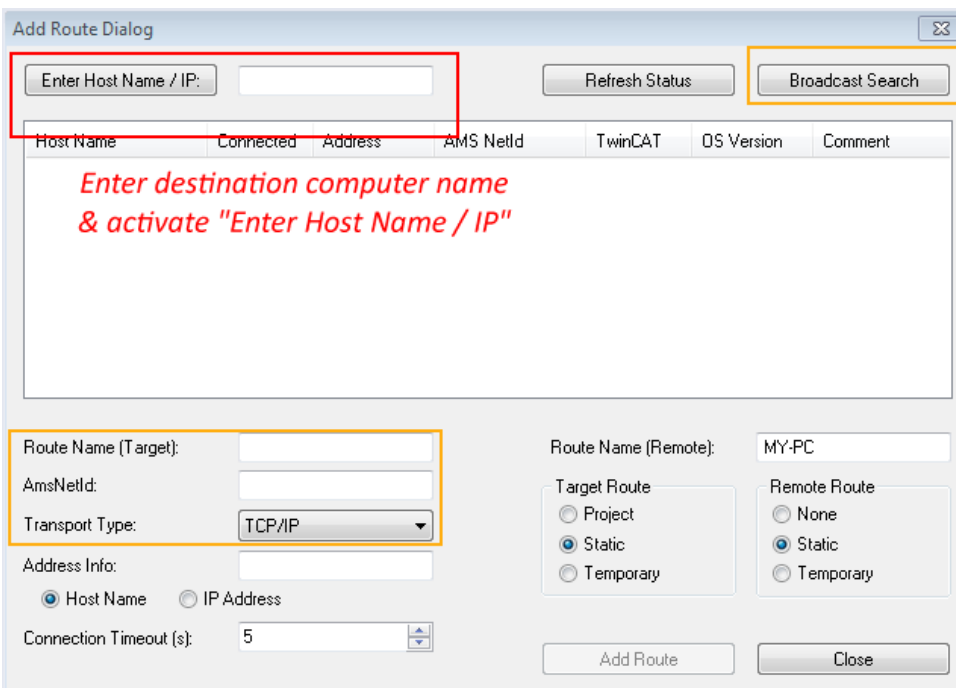
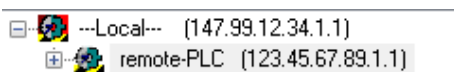


Fig. 79: Specify the PLC for access by the TwinCAT System Manager: selection of the target system



Once the target system has been entered, it is available for selection as follows (a password may have to be entered):



After confirmation with “OK” the target system can be accessed via the System Manager.

## Adding devices

In the configuration tree of the TwinCAT 2 System Manager user interface on the left, select “I/O Devices” and then right-click to open a context menu and select “Scan Devices...”, or start the action in the menu bar

via . The TwinCAT System Manager may first have to be set to “Config mode” via  or via menu “Actions” → “Set/Reset TwinCAT to Config Mode...” (Shift + F4).

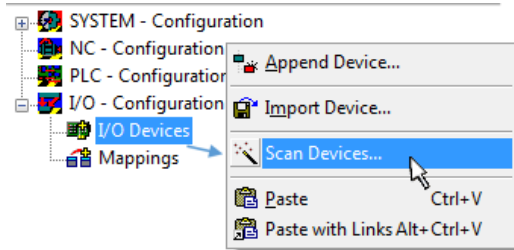


Fig. 80: Select “Scan Devices...”

Confirm the warning message, which follows, and select “EtherCAT” in the dialog:

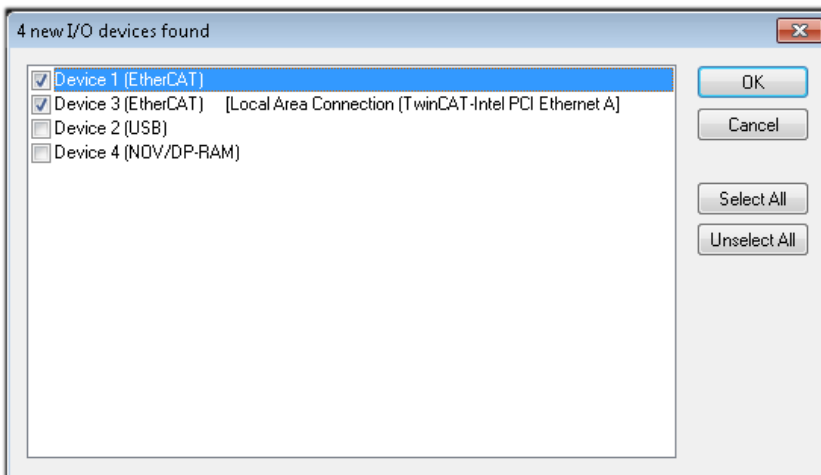


Fig. 81: Automatic detection of I/O devices: selection the devices to be integrated

Confirm the message “Find new boxes”, in order to determine the terminals connected to the devices. “Free Run” enables manipulation of input and output values in “Config mode” and should also be acknowledged.

Based on the [sample configuration](#) [▶ 64] described at the beginning of this section, the result is as follows:

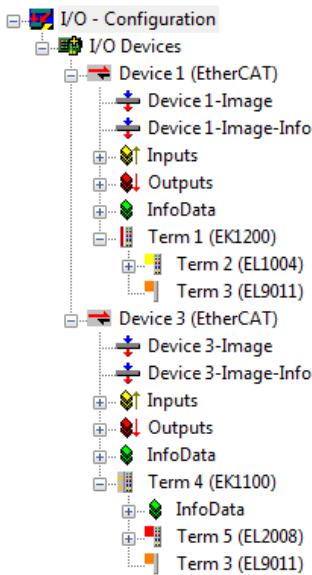


Fig. 82: Mapping of the configuration in the TwinCAT 2 System Manager

The whole process consists of two stages, which may be performed separately (first determine the devices, then determine the connected elements such as boxes, terminals, etc.). A scan can also be initiated by selecting “Device ...” from the context menu, which then reads the elements present in the configuration below:

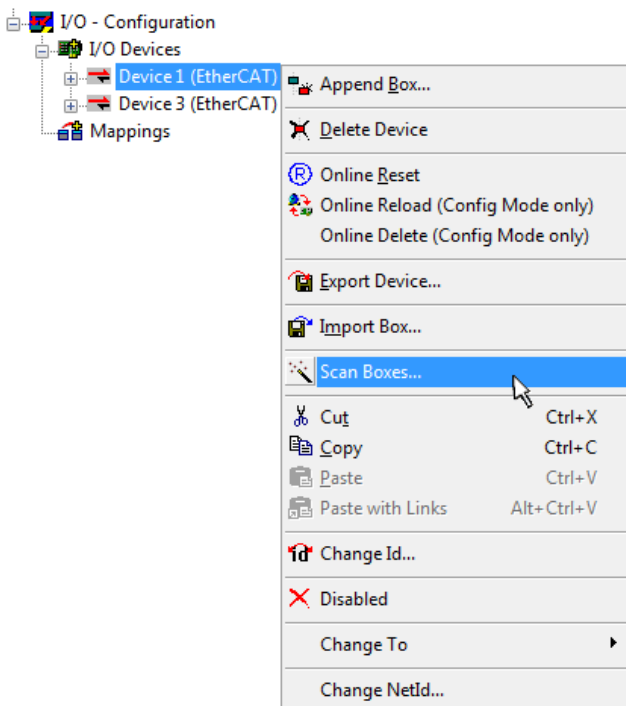


Fig. 83: Reading of individual terminals connected to a device

This functionality is useful if the actual configuration is modified at short notice.

**Programming and integrating the PLC**

TwinCAT PLC Control is the development environment for the creation of the controller in different program environments: TwinCAT PLC Control supports all languages described in IEC 61131-3. There are two text-based languages and three graphical languages.

- **Text-based languages**
  - Instruction List (IL)

- Structured Text (ST)
- **Graphical languages**
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - The Continuous Function Chart Editor (CFC)
  - Sequential Function Chart (SFC)

The following section refers to Structured Text (ST).

After starting TwinCAT PLC Control, the following user interface is shown for an initial project:

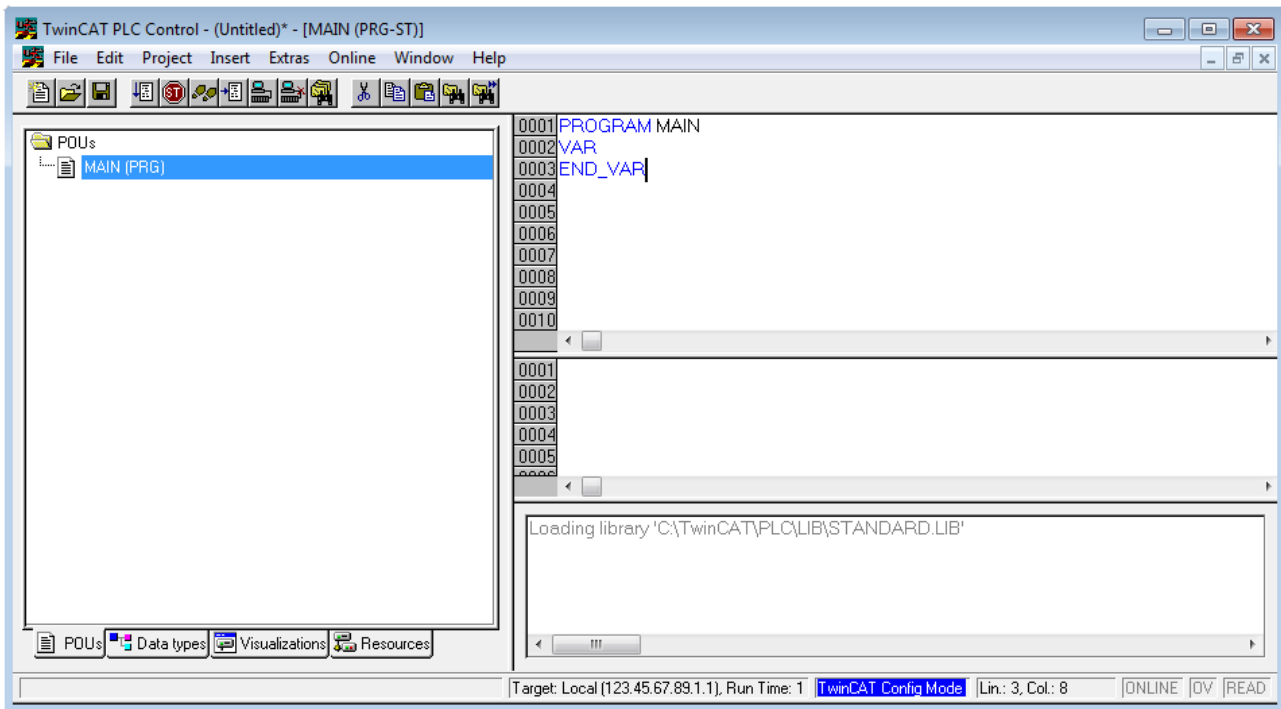


Fig. 84: TwinCAT PLC Control after startup

Sample variables and a sample program have been created and stored under the name "PLC\_example.pro":

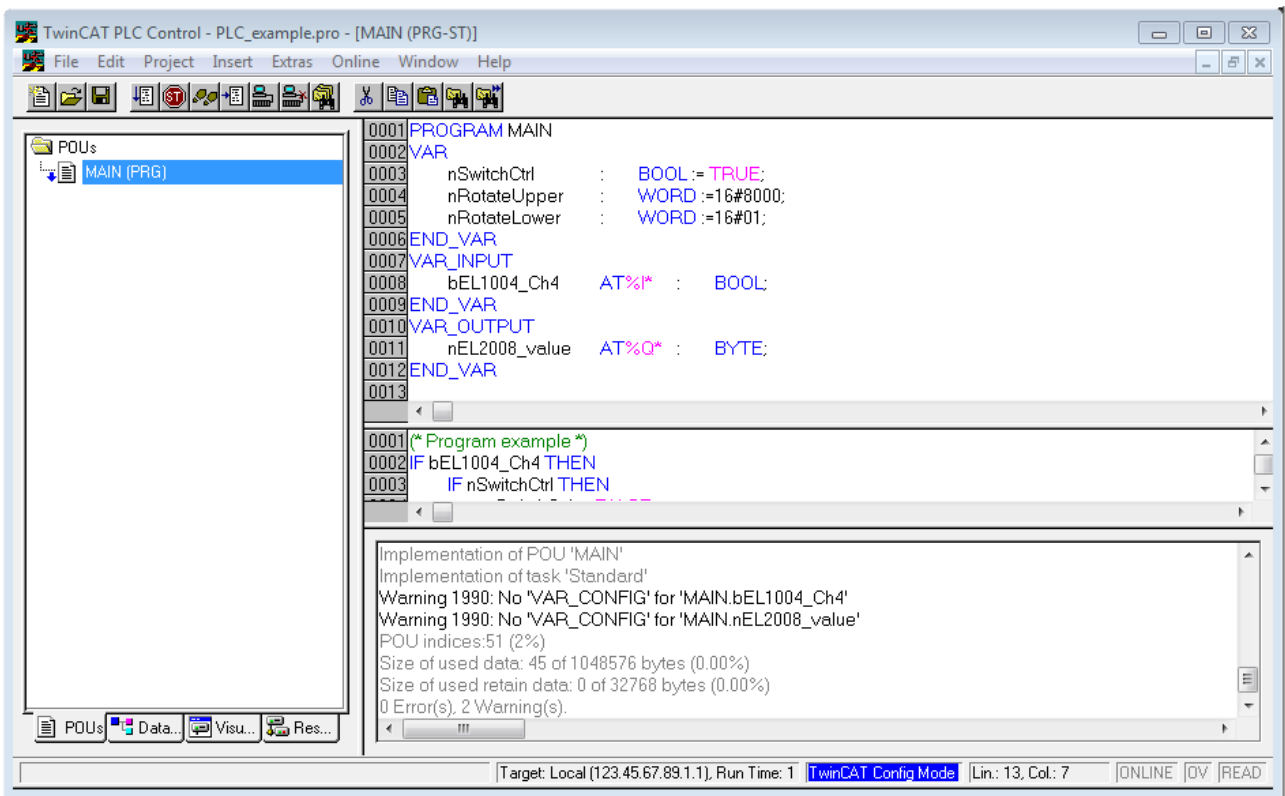


Fig. 85: Sample program with variables after a compile process (without variable integration)

Warning 1990 (missing “VAR\_CONFIG”) after a compile process indicates that the variables defined as external (with the ID “AT%I\*” or “AT%Q\*”) have not been assigned. After successful compilation, TwinCAT PLC Control creates a “\*.tpy” file in the directory in which the project was stored. This file (“\*.tpy”) contains variable assignments and is not known to the System Manager, hence the warning. Once the System Manager has been notified, the warning no longer appears.

First, integrate the TwinCAT PLC Control project in the **System Manager** via the context menu of the PLC configuration; right-click and select “Append PLC Project...”:

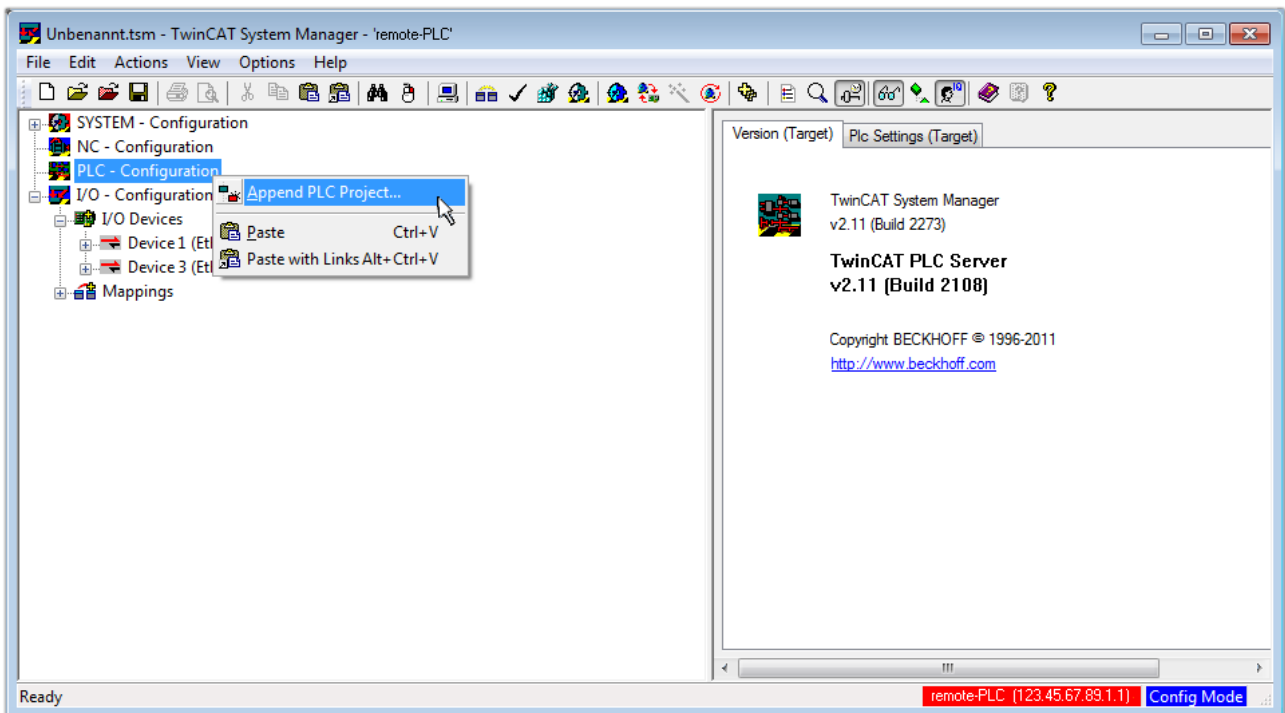


Fig. 86: Appending the TwinCAT PLC Control project

Select the PLC configuration “PLC\_example.tpy” in the browser window that opens. The project including the two variables identified with “AT” are then integrated in the configuration tree of the System Manager:

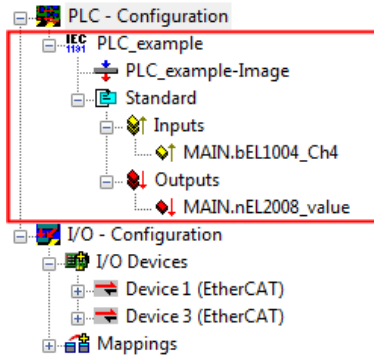


Fig. 87: PLC project integrated in the PLC configuration of the System Manager

The two variables “bEL1004\_Ch4” and “nEL2008\_value” can now be assigned to certain process objects of the I/O configuration.

### Assigning variables

Open a window for selecting a suitable process object (PDO) via the context menu of a variable of the integrated project “PLC\_example” and via “Modify Link...” “Standard”:

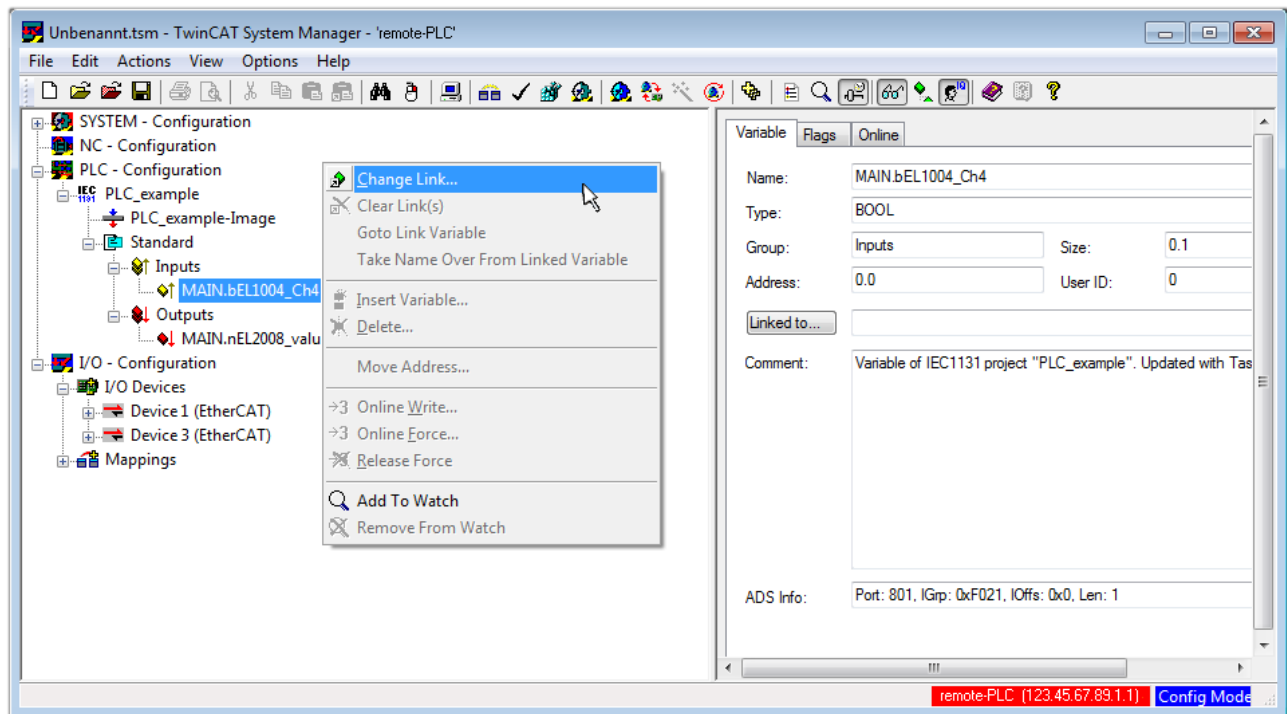


Fig. 88: Creating the links between PLC variables and process objects

In the window that opens, the process object for the variable “bEL1004\_Ch4” of type BOOL can be selected from the PLC configuration tree:



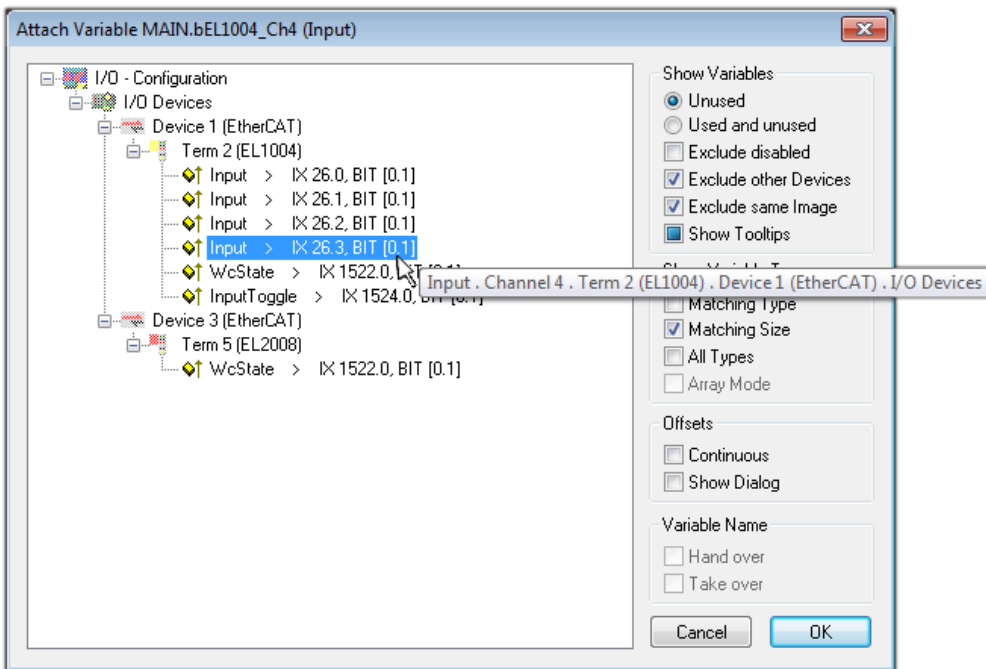


Fig. 89: Selecting PDO of type BOOL

According to the default setting, certain PDO objects are now available for selection. In this sample the input of channel 4 of the EL1004 terminal is selected for linking. In contrast, the checkbox “All types” must be ticked for creating the link for the output variables, in order to allocate a set of eight separate output bits to a byte variable. The following diagram shows the whole process:

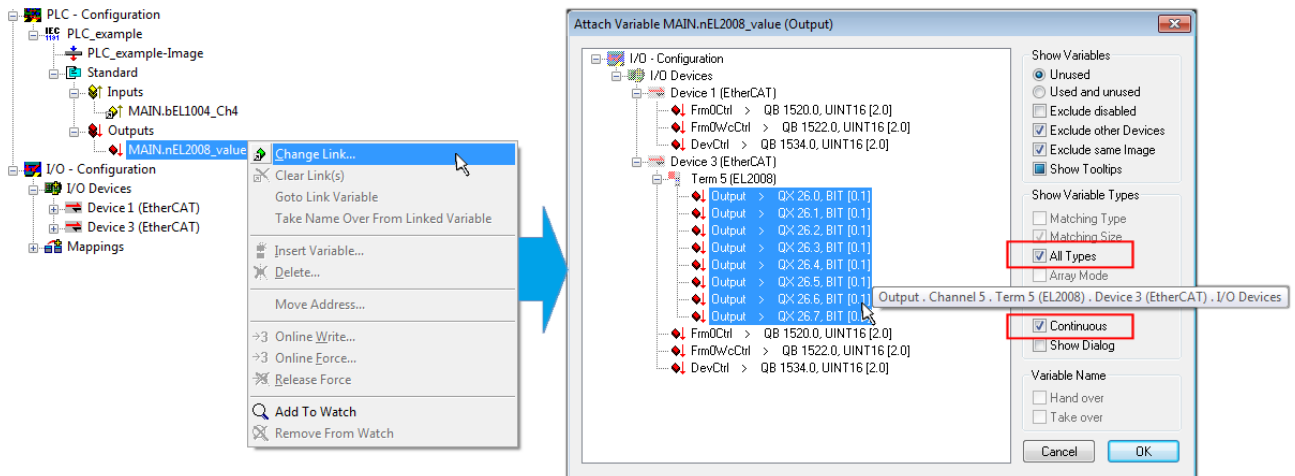



Fig. 90: Selecting several PDOs simultaneously: activate “Continuous” and “All types”

Note that the “Continuous” checkbox was also activated. This is designed to allocate the bits contained in the byte of the variable “nEL2008\_value” sequentially to all eight selected output bits of the EL2008 terminal. In this way it is possible to subsequently address all eight outputs of the terminal in the program with a byte corresponding to bit 0 for channel 1 to bit 7 for channel 8 of the PLC. A special symbol (  ) at the yellow or red object of the variable indicates that a link exists. The links can also be checked by selecting a “Goto Link Variable” from the context menu of a variable. The object opposite, in this case the PDO, is automatically selected:

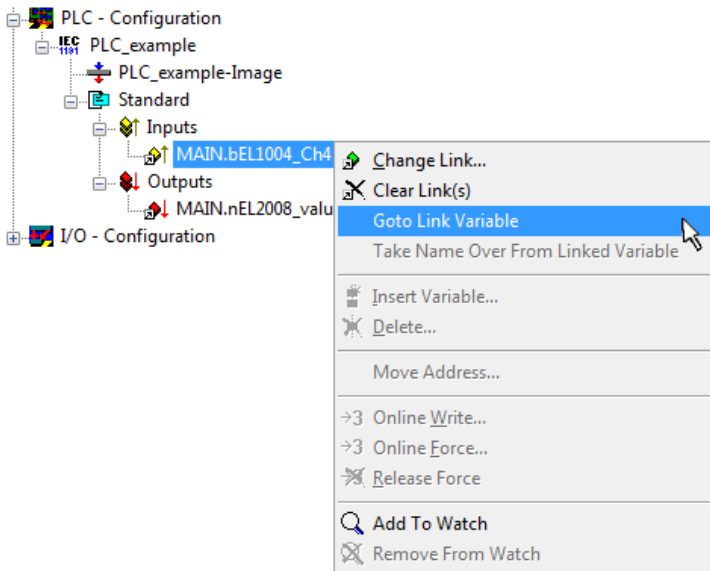

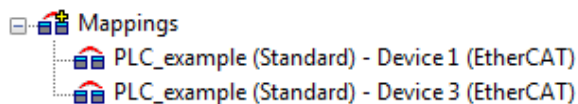


Fig. 91: Application of a “Goto Link” variable, using “MAIN.bEL1004\_Ch4” as a sample

The process of assigning variables to the PDO is completed via the menu selection “Actions” → “Generate

Mappings”, key Ctrl+M or by clicking on the symbol  in the menu.


This can be visualized in the configuration:




The process of creating links can also take place in the opposite direction, i.e. starting with individual PDOs to variable. However, in this example it would then not be possible to select all output bits for the EL2008, since the terminal only makes individual digital outputs available. If a terminal has a byte, word, integer or similar PDO, it is possible to allocate this a set of bit-standardized variables (type “BOOL”). Here, too, a “Goto Link Variable” from the context menu of a PDO can be executed in the other direction, so that the respective PLC instance can then be selected.

### Activation of the configuration

The allocation of PDO to PLC variables has now established the connection from the controller to the inputs and outputs of the terminals. The configuration can now be activated. First, the configuration can be verified

via  (or via “Actions” → “Check Configuration”). If no error is present, the configuration can be

activated via  (or via “Actions” → “Activate Configuration...”) to transfer the System Manager settings to the runtime system. Confirm the messages “Old configurations are overwritten!” and “Restart TwinCAT system in Run mode” with “OK”.

A few seconds later the real-time status **RTime 0%** is displayed at the bottom right in the System Manager. The PLC system can then be started as described below.

### Starting the controller

Starting from a remote system, the PLC control has to be linked with the Embedded PC over Ethernet via “Online” → “Choose Run-Time System...”:

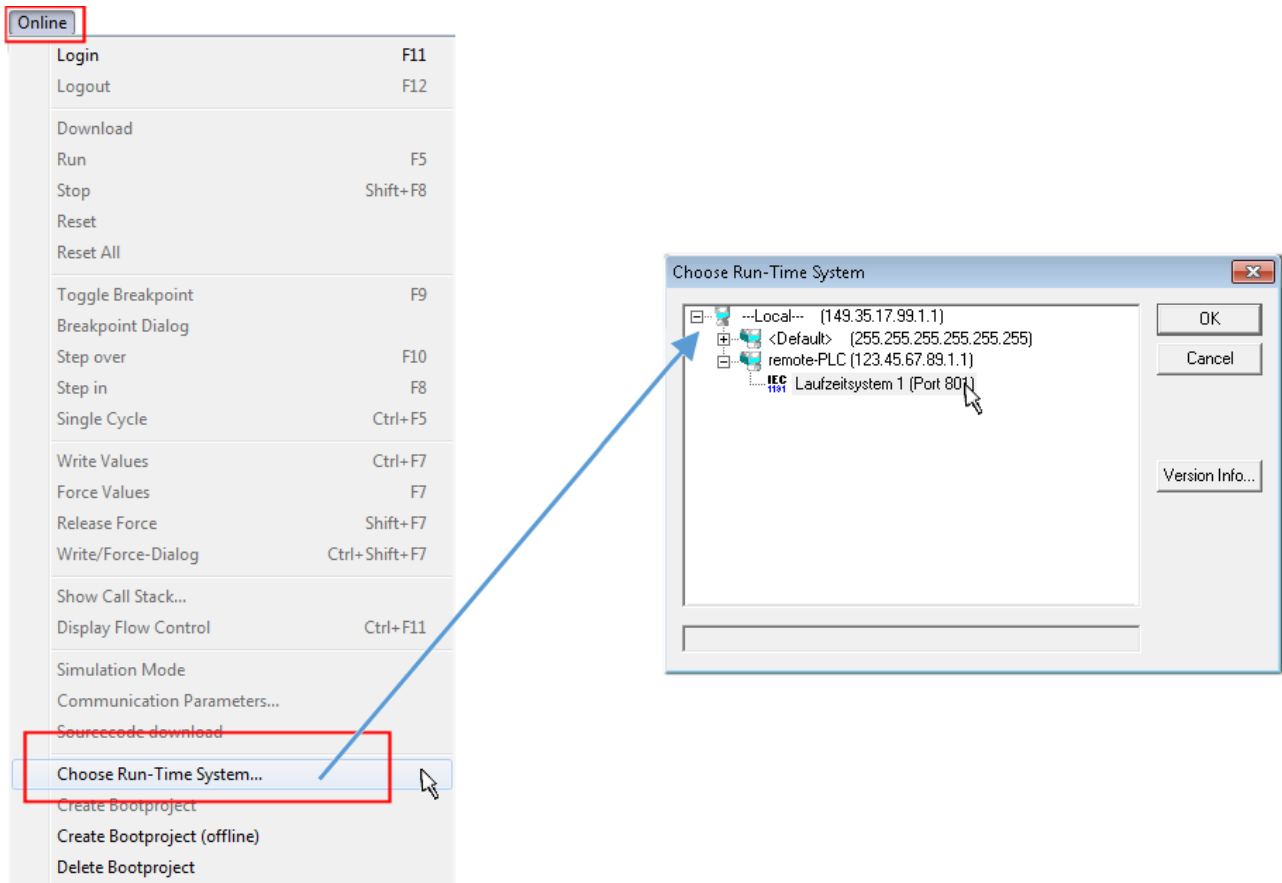



Fig. 92: Choose target system (remote)

In this sample “Runtime system 1 (port 801)” is selected and confirmed. Link the PLC with the real-time

system via menu option “Online” → “Login”, the F11 key or by clicking on the symbol . The control program can then be loaded for execution. This results in the message “No program on the controller! Should the new program be loaded?”, which should be acknowledged with “Yes”. The runtime environment is ready for the program start:

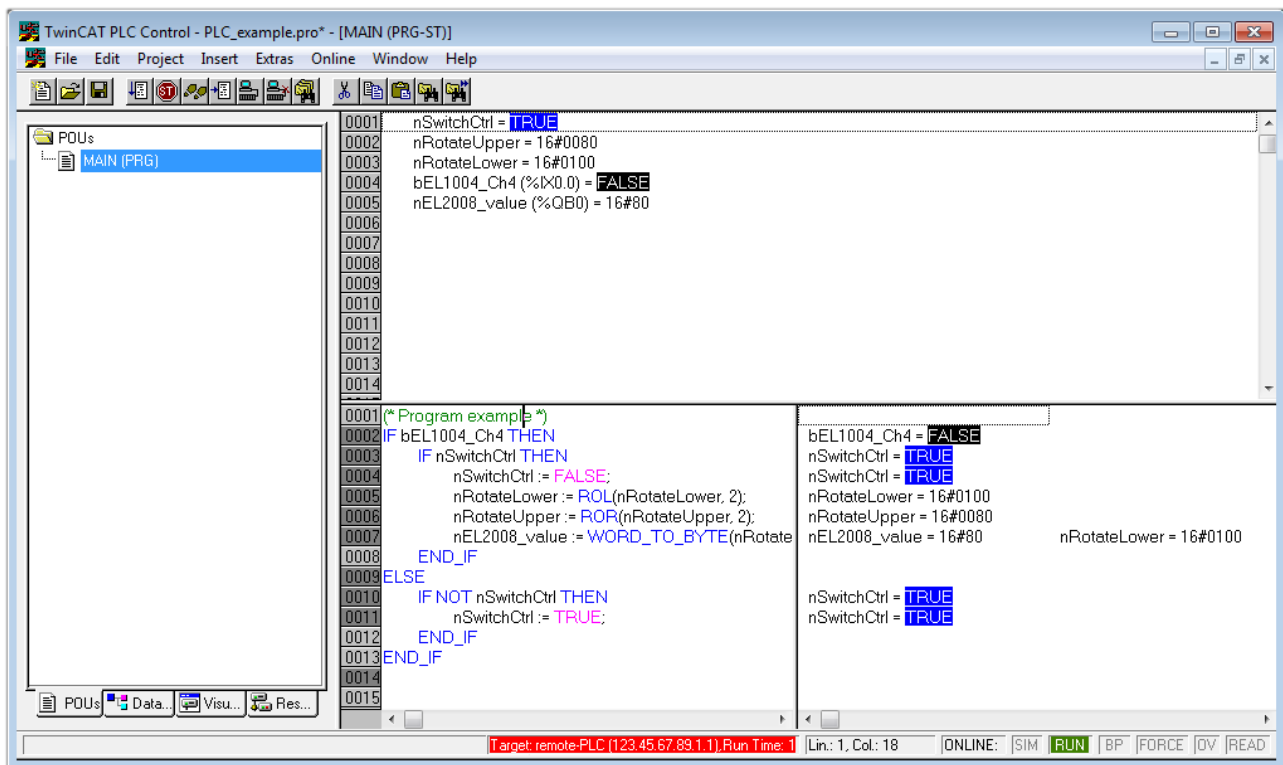


Fig. 93: PLC Control logged in, ready for program startup

The PLC can now be started via “Online” → “Run”, F5 key or .

### 3.1.2 TwinCAT 3


#### Startup

TwinCAT makes the development environment areas available together with Microsoft Visual Studio: after startup, the project folder explorer appears on the left in the general window area (cf. “TwinCAT System Manager” of TwinCAT 2) for communication with the electromechanical components.

After successful installation of the TwinCAT system on the PC to be used for development, TwinCAT 3 (shell) displays the following user interface after startup:



Fig. 94: Initial TwinCAT 3 user interface

First create a new project via  **New TwinCAT Project...** (or under “File”→“New”→“Project...”). In the following dialog make the corresponding entries as required (as shown in the diagram):

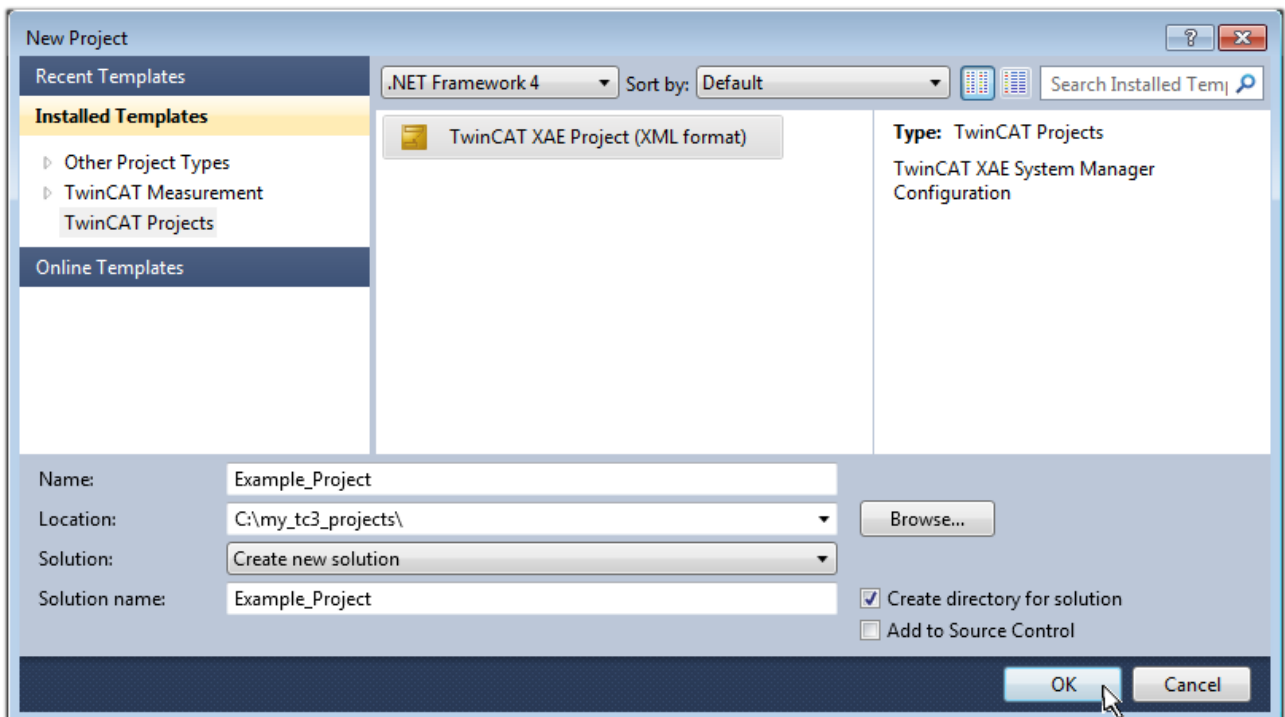


Fig. 95: Create new TwinCAT project

The new project is then available in the project folder explorer:

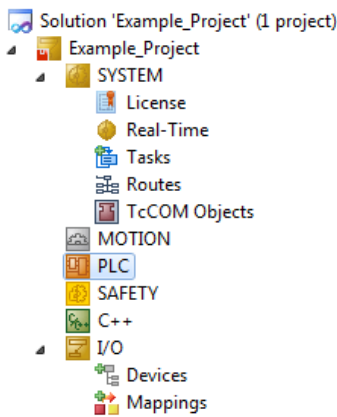
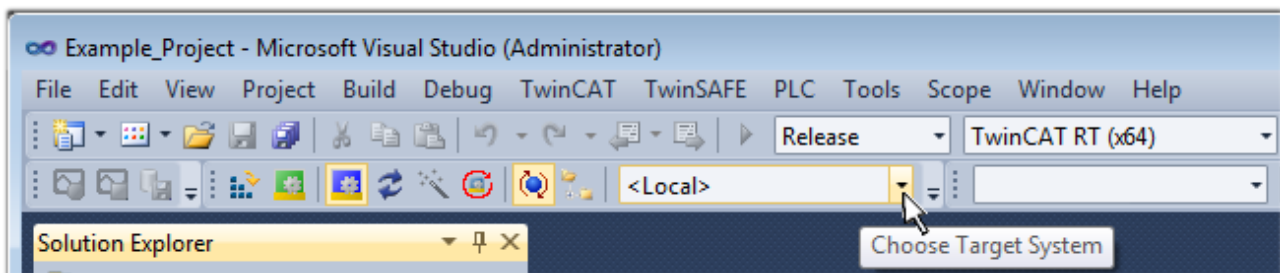


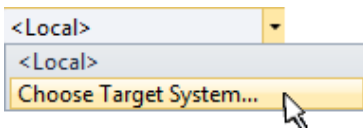
Fig. 96: New TwinCAT3 project in the project folder explorer

Generally, TwinCAT can be used in local or remote mode. Once the TwinCAT system including the user interface (standard) is installed on the respective PLC, TwinCAT can be used in local mode and thereby the next step is “Insert Device [▶ 79]”.

If the intention is to address the TwinCAT runtime environment installed on a PLC as development environment remotely from another system, the target system must be made known first. Via the symbol in the menu bar:



expand the pull-down menu:



and open the following window:

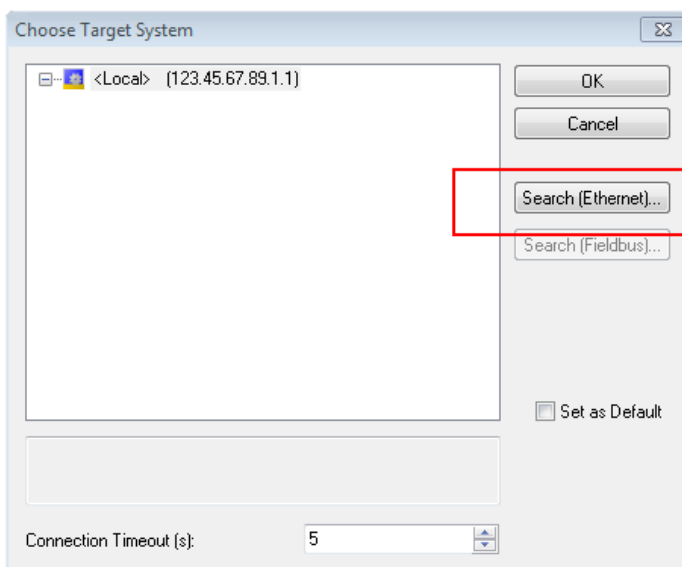


Fig. 97: Selection dialog: Choose the target system

Use “Search (Ethernet)...” to enter the target system. Thus a next dialog opens to either:

- enter the known computer name after “Enter Host Name / IP:” (as shown in red)
- perform a “Broadcast Search” (if the exact computer name is not known)
- enter the known computer IP or AmsNetID.

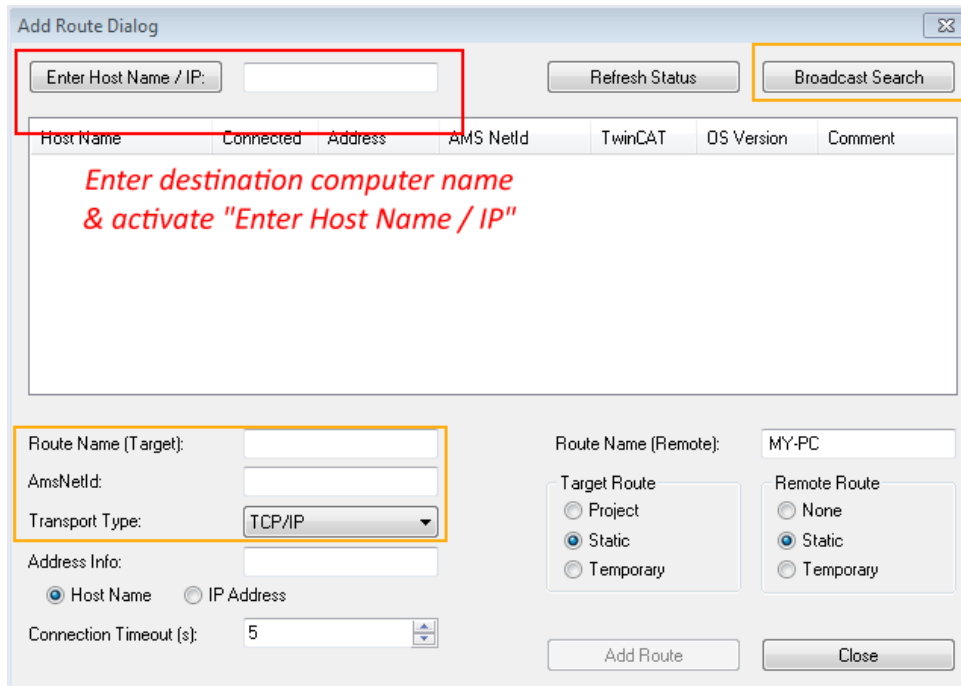
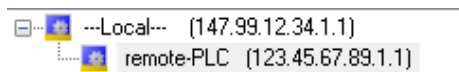


Fig. 98: Specify the PLC for access by the TwinCAT System Manager: selection of the target system


Once the target system has been entered, it is available for selection as follows (a password may have to be entered):




After confirmation with “OK” the target system can be accessed via the Visual Studio shell.

**Adding devices**

In the project folder explorer of the Visual Studio shell user interface on the left, select “Devices” within

element “I/O”, then right-click to open a context menu and select “Scan” or start the action via  in the

menu bar. The TwinCAT System Manager may first have to be set to “Config mode” via  or via the menu “TwinCAT” → “Restart TwinCAT (Config mode)”.

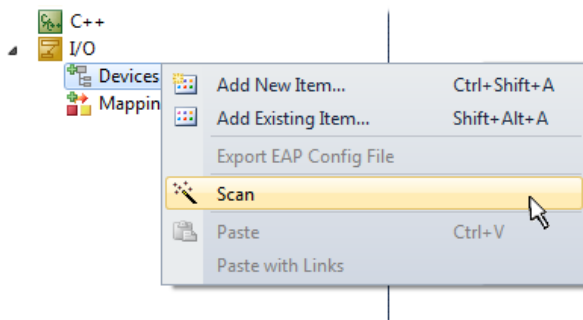


Fig. 99: Select “Scan”

Confirm the warning message, which follows, and select “EtherCAT” in the dialog:

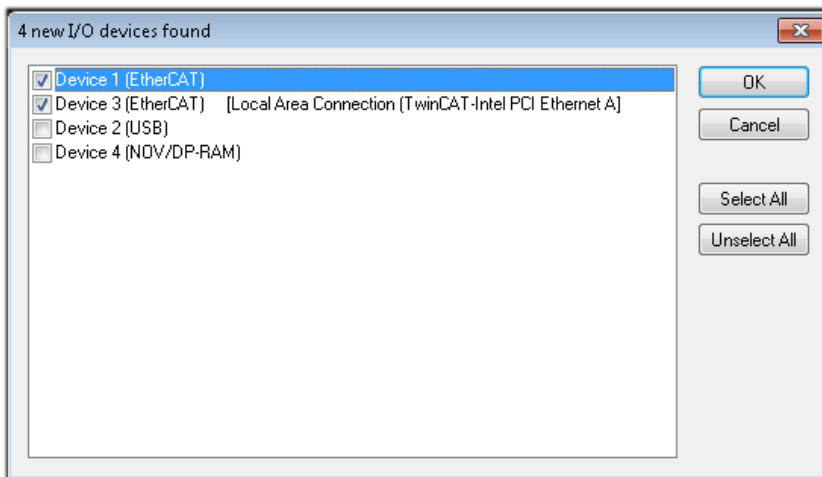


Fig. 100: Automatic detection of I/O devices: selection the devices to be integrated

Confirm the message “Find new boxes”, in order to determine the terminals connected to the devices. “Free Run” enables manipulation of input and output values in “Config mode” and should also be acknowledged.

Based on the [sample configuration \[▶ 64\]](#) described at the beginning of this section, the result is as follows:

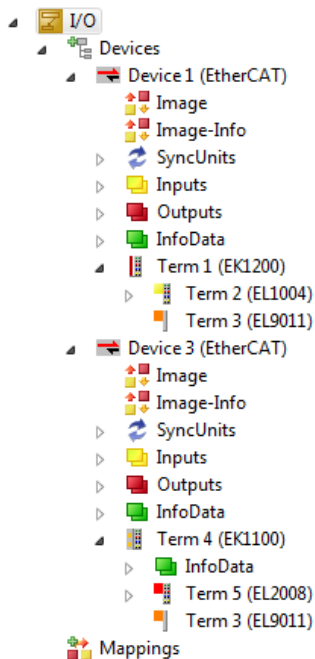


Fig. 101: Mapping of the configuration in VS shell of the TwinCAT3 environment

The whole process consists of two stages, which may be performed separately (first determine the devices, then determine the connected elements such as boxes, terminals, etc.). A scan can also be initiated by selecting “Device ...” from the context menu, which then reads the elements present in the configuration below:



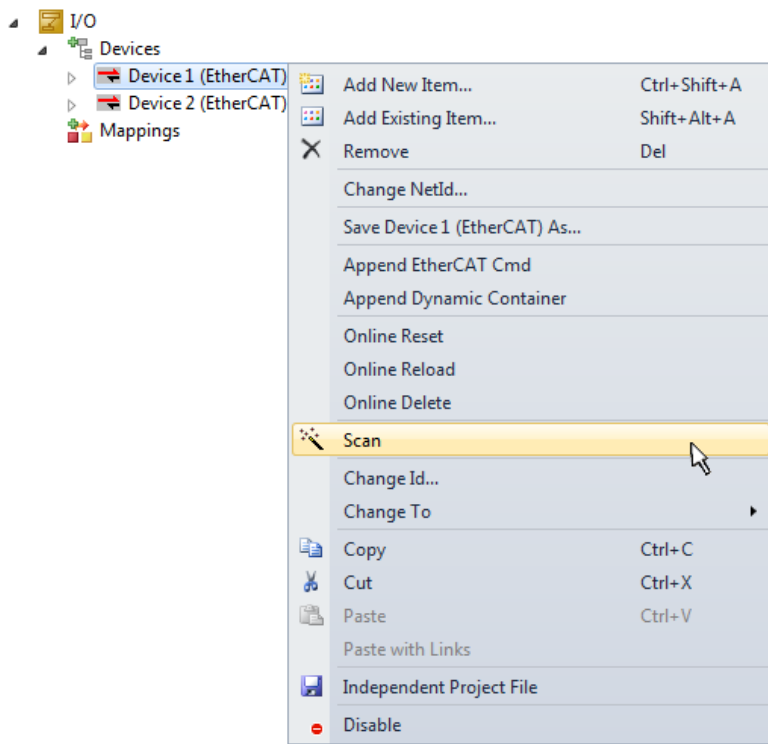


Fig. 102: Reading of individual terminals connected to a device

This functionality is useful if the actual configuration is modified at short notice.

**Programming the PLC**

TwinCAT PLC Control is the development environment for the creation of the controller in different program environments: TwinCAT PLC Control supports all languages described in IEC 61131-3. There are two text-based languages and three graphical languages.

- **Text-based languages**
  - Instruction List (IL)
  - Structured Text (ST)
- **Graphical languages**
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - The Continuous Function Chart Editor (CFC)
  - Sequential Function Chart (SFC)

The following section refers to Structured Text (ST).

In order to create a programming environment, a PLC subproject is added to the project sample via the context menu of "PLC" in the project folder explorer by selecting "Add New Item....":

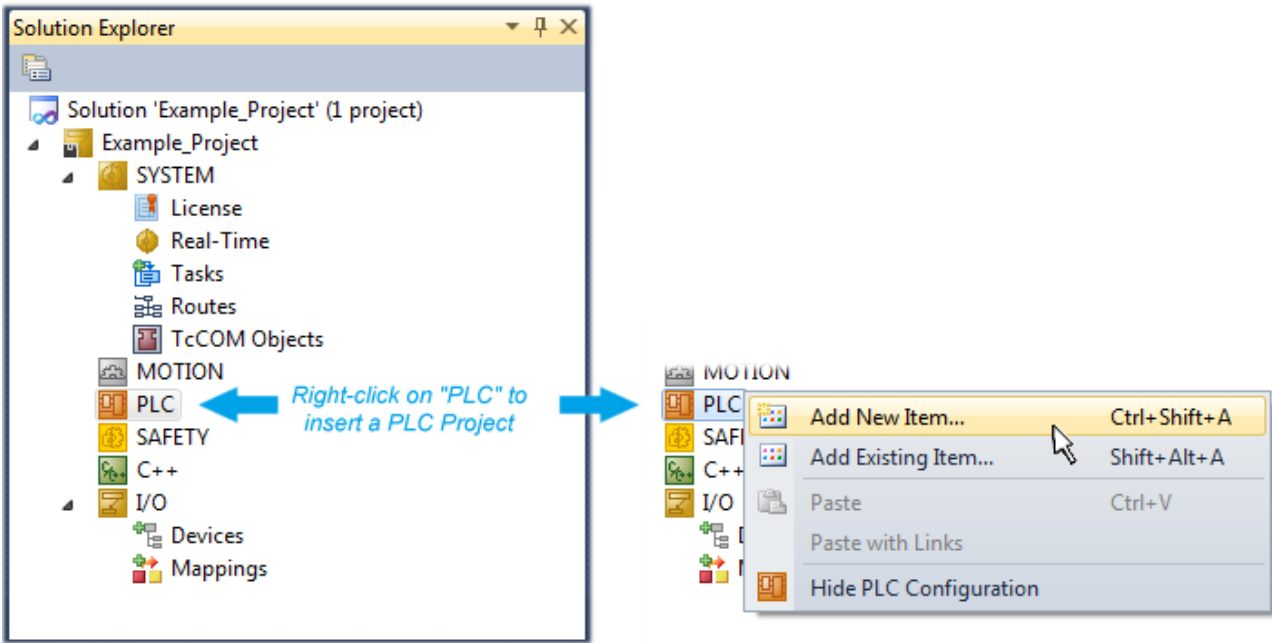


Fig. 103: Adding the programming environment in “PLC”

In the dialog that opens select “Standard PLC project” and enter “PLC\_example” as project name, for example, and select a corresponding directory:

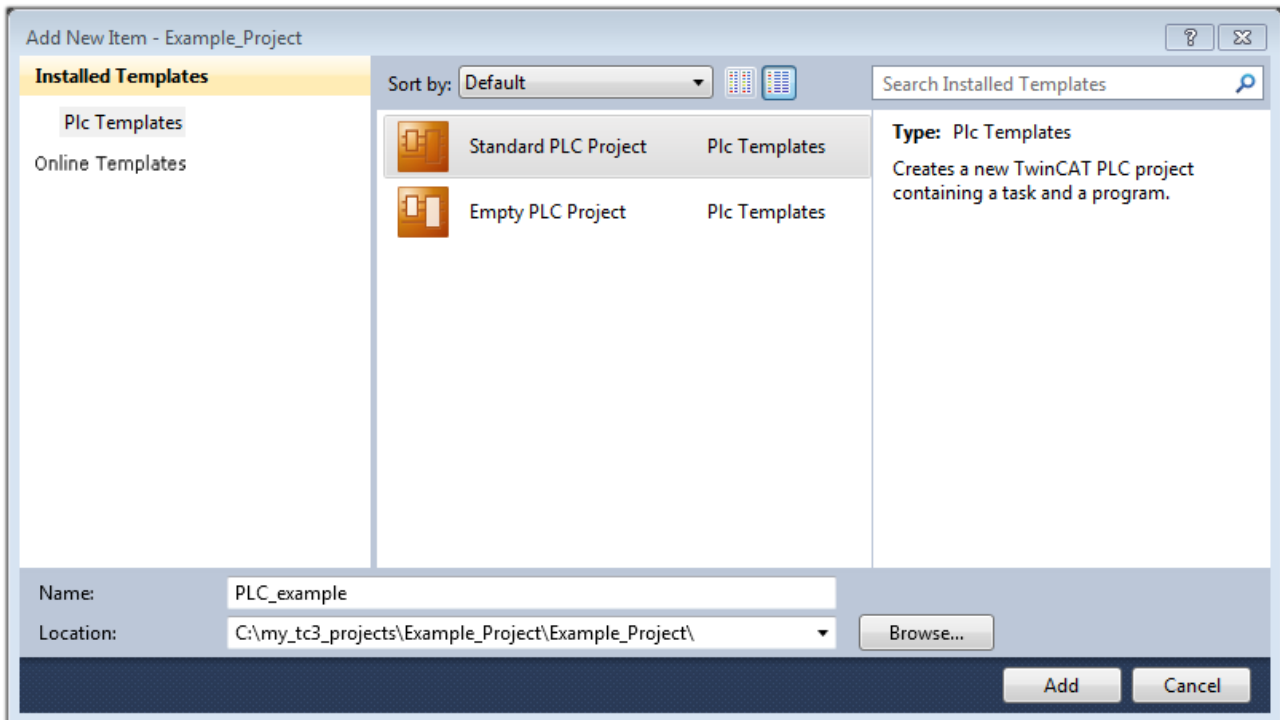


Fig. 104: Specifying the name and directory for the PLC programming environment

The “Main” program, which already exists by selecting “Standard PLC project”, can be opened by double-clicking on “PLC\_example\_project” in “POUs”. The following user interface is shown for an initial project:

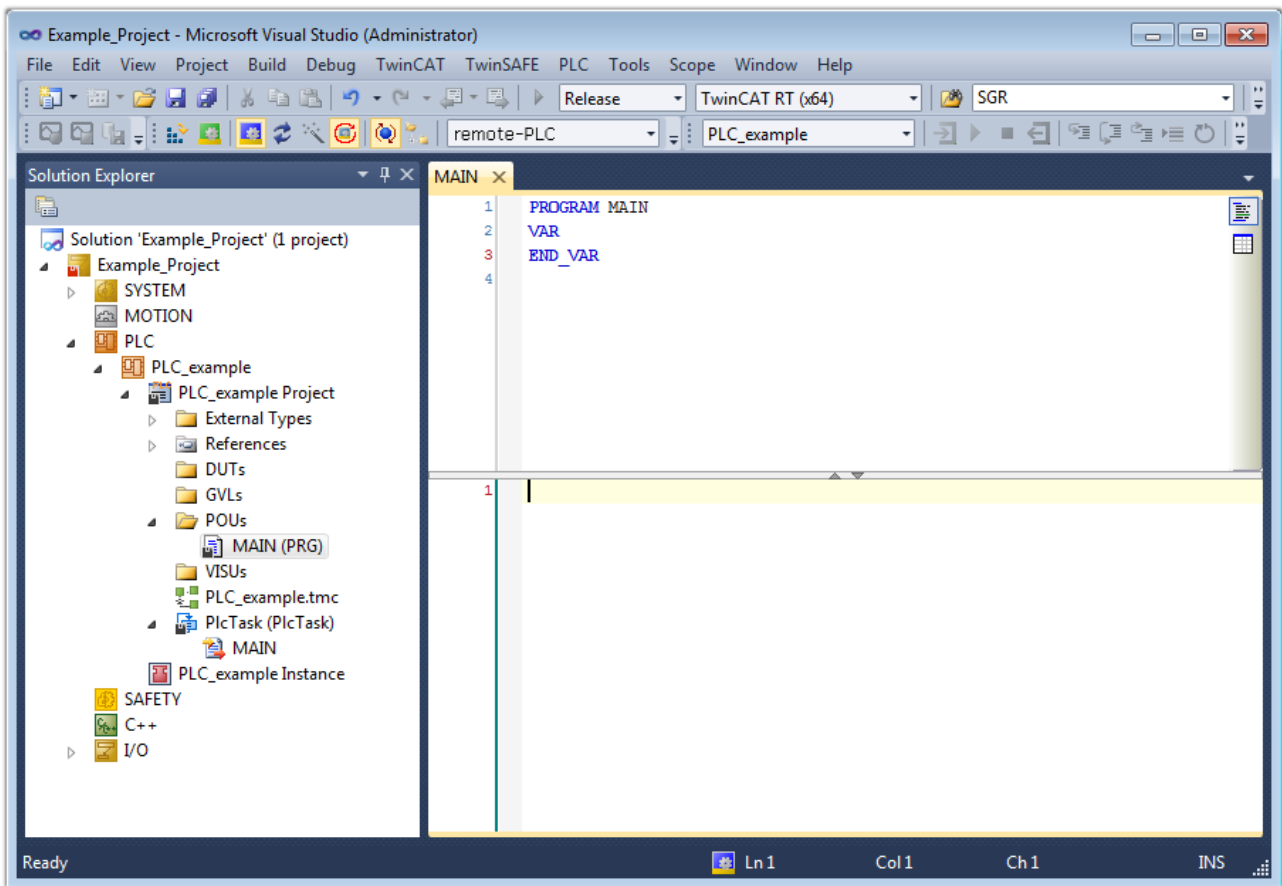


Fig. 105: Initial “Main” program of the standard PLC project

To continue, sample variables and a sample program have now been created:

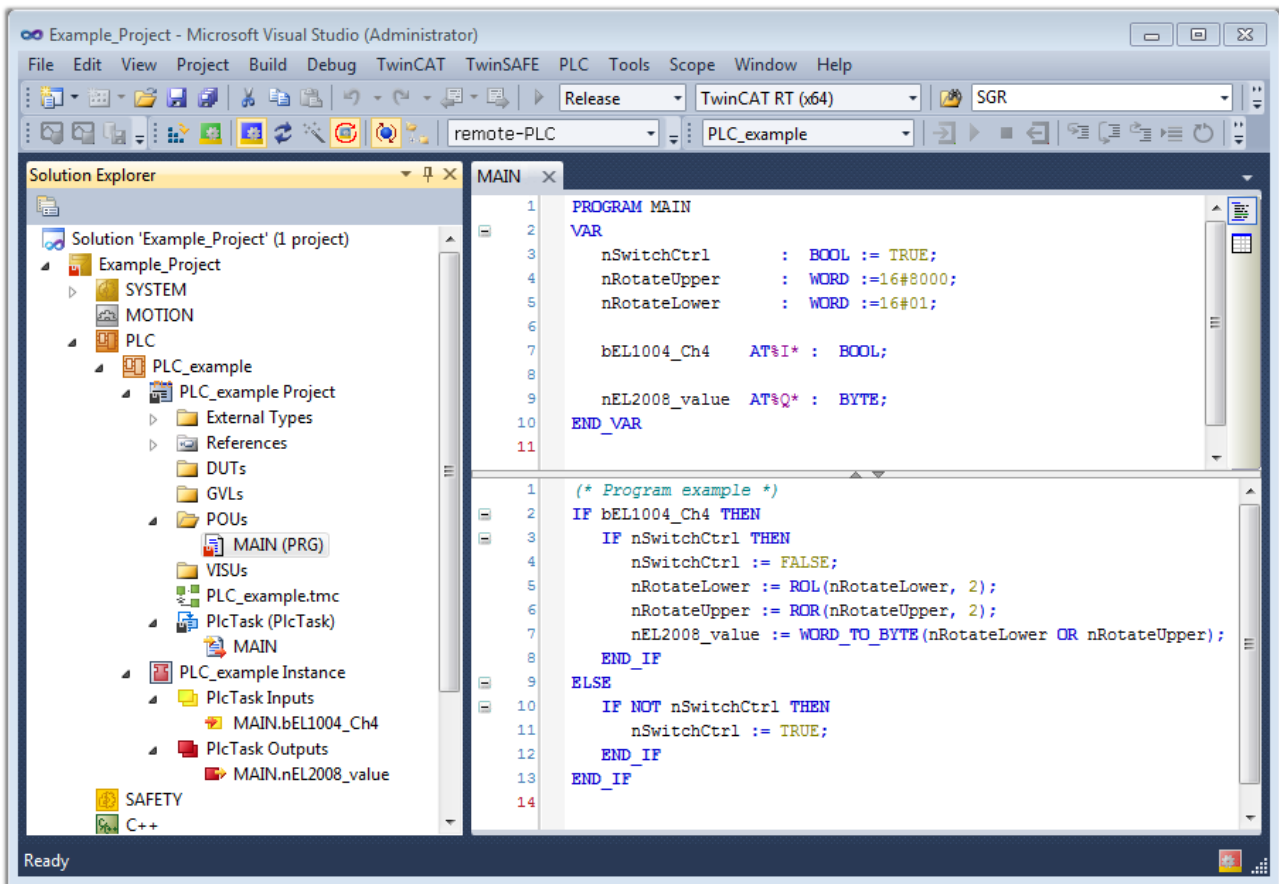


Fig. 106: Sample program with variables after a compile process (without variable integration)

The control program is now created as a project folder, followed by the compile process:

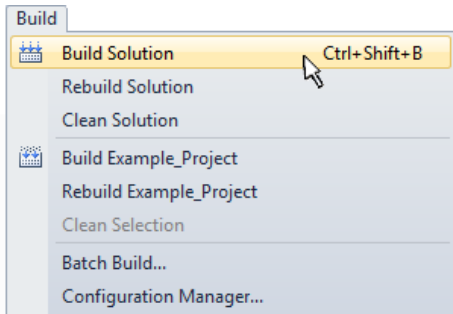
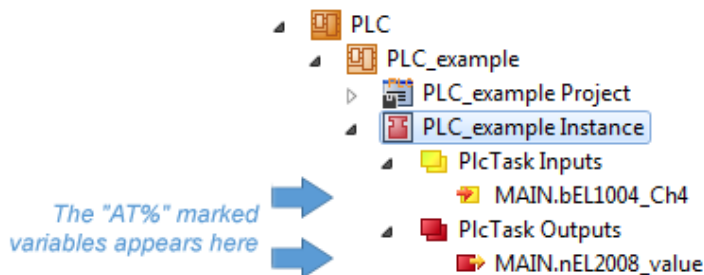


Fig. 107: Start program compilation

The following variables, identified in the ST/ PLC program with “AT%”, are then available in under “Assignments” in the project folder explorer:



**Assigning variables**

Via the menu of an instance - variables in the “PLC” context, use the “Modify Link...” option to open a window for selecting a suitable process object (PDO) for linking:

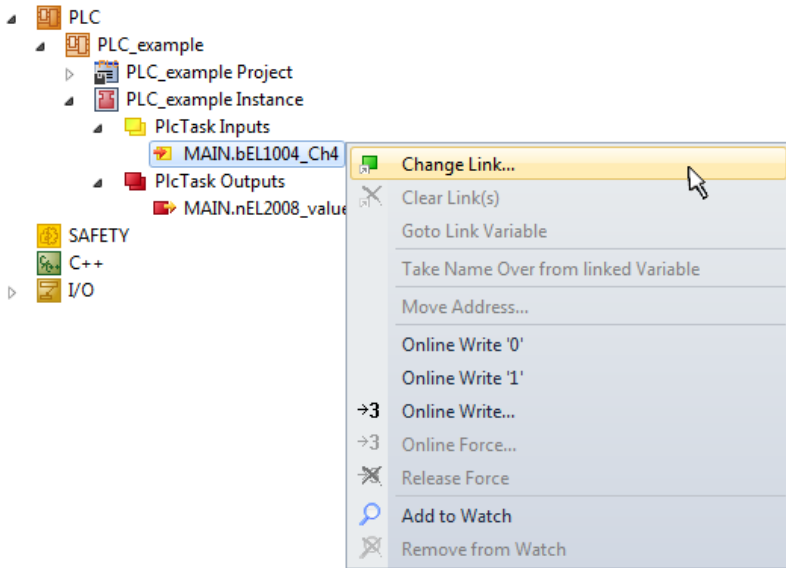


Fig. 108: Creating the links between PLC variables and process objects

In the window that opens, the process object for the variable “bEL1004\_Ch4” of type BOOL can be selected from the PLC configuration tree:

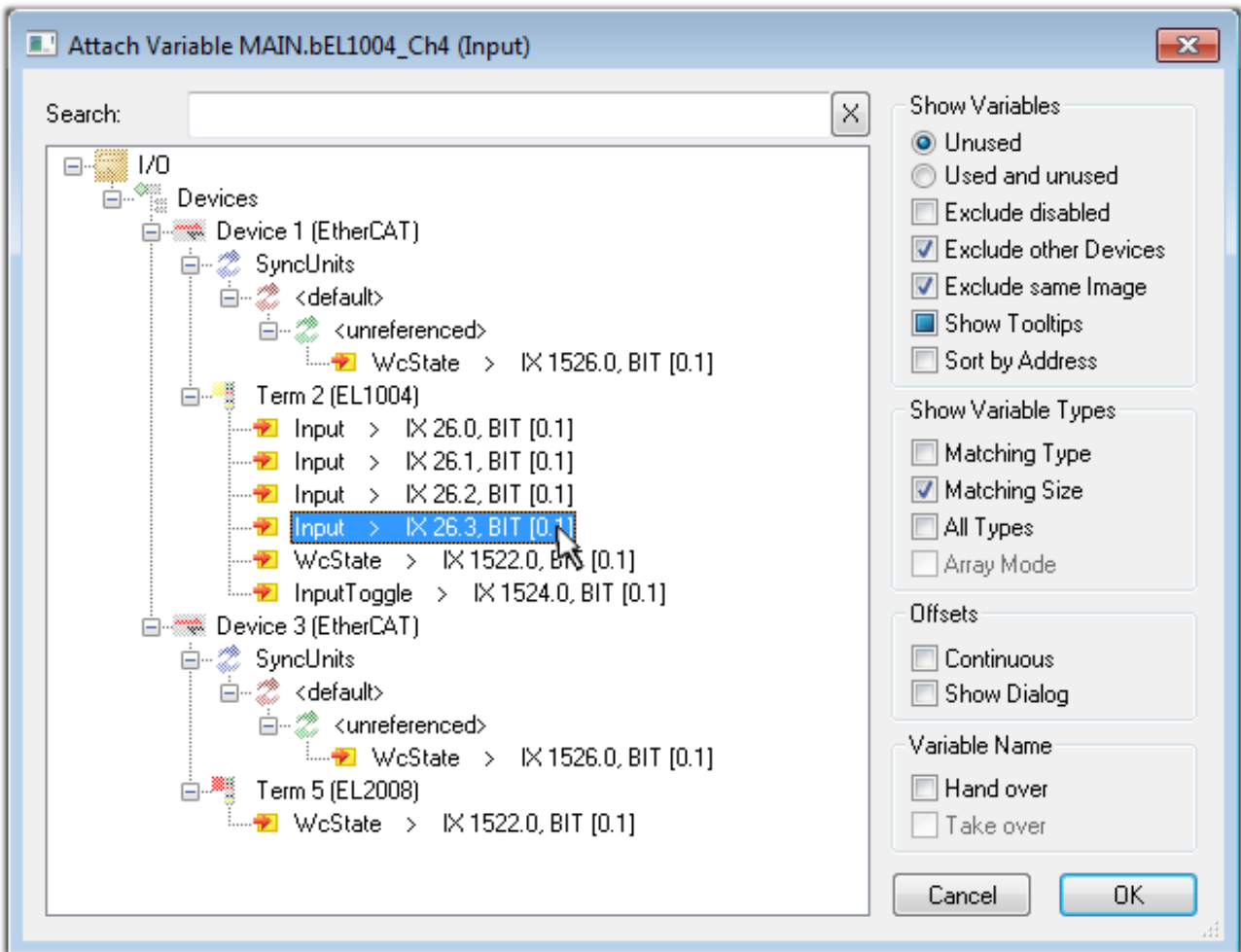


Fig. 109: Selecting PDO of type BOOL

According to the default setting, certain PDO objects are now available for selection. In this sample the input of channel 4 of the EL1004 terminal is selected for linking. In contrast, the checkbox “All types” must be ticked for creating the link for the output variables, in order to allocate a set of eight separate output bits to a byte variable. The following diagram shows the whole process:

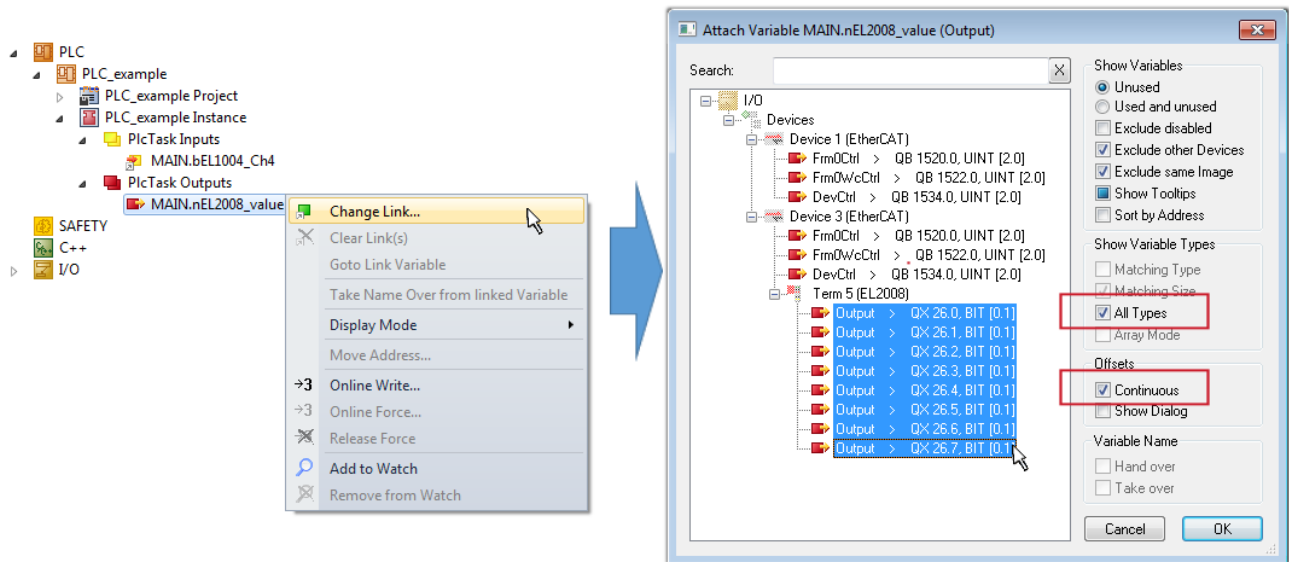



Fig. 110: Selecting several PDOs simultaneously: activate “Continuous” and “All types”

Note that the “Continuous” checkbox was also activated. This is designed to allocate the bits contained in the byte of the variable “nEL2008\_value” sequentially to all eight selected output bits of the EL2008 terminal. In this way it is possible to subsequently address all eight outputs of the terminal in the program with a byte corresponding to bit 0 for channel 1 to bit 7 for channel 8 of the PLC. A special symbol (  ) at the yellow or red object of the variable indicates that a link exists. The links can also be checked by selecting a “Goto Link Variable” from the context menu of a variable. The object opposite, in this case the PDO, is automatically selected:

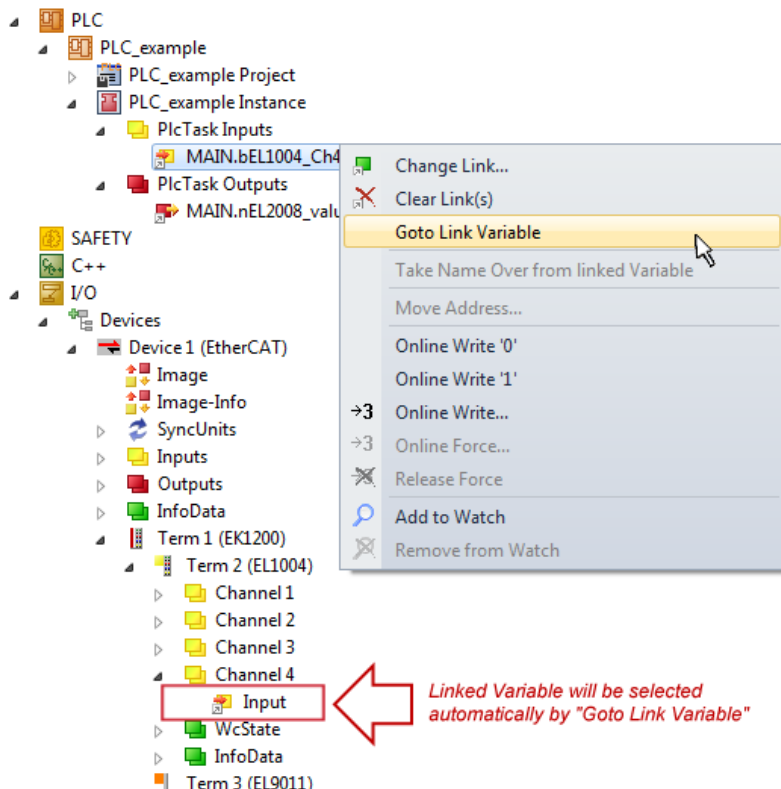


Fig. 111: Application of a “Goto Link” variable, using “MAIN.bEL1004\_Ch4” as a sample

The process of creating links can also take place in the opposite direction, i.e. starting with individual PDOs to variable. However, in this example it would then not be possible to select all output bits for the EL2008, since the terminal only makes individual digital outputs available. If a terminal has a byte, word, integer or

similar PDO, it is possible to allocate this a set of bit-standardized variables (type "BOOL"). Here, too, a "Goto Link Variable" from the context menu of a PDO can be executed in the other direction, so that the respective PLC instance can then be selected.

**Note on the type of variable assignment**

**i** The following type of variable assignment can only be used from TwinCAT version V3.1.4024.4 onwards and is only available for terminals with a microcontroller.

In TwinCAT it is possible to create a structure from the mapped process data of a terminal. An instance of this structure can then be created in the PLC, so it is possible to access the process data directly from the PLC without having to declare own variables.

The procedure for the EL3001 1-channel analog input terminal -10...+10 V is shown as an example.

1. First the required process data must be selected in the "Process data" tab in TwinCAT.
2. After that, the PLC data type must be generated in the tab "PLC" via the check box.
3. The data type in the "Data Type" field can then be copied using the "Copy" button.

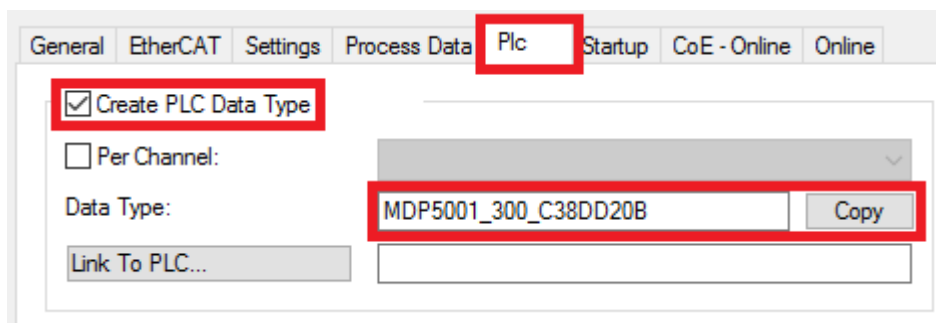


Fig. 112: Creating a PLC data type

4. An instance of the data structure of the copied data type must then be created in the PLC.

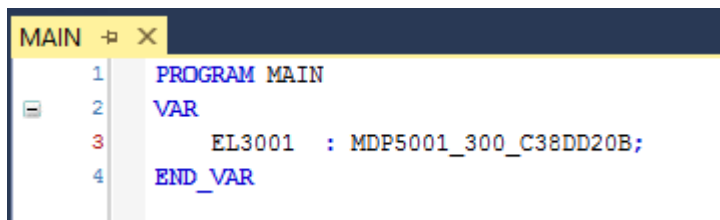


Fig. 113: Instance\_of\_struct

5. Then the project folder must be created. This can be done either via the key combination "CTRL + Shift + B" or via the "Build" tab in TwinCAT.
6. The structure in the "PLC" tab of the terminal must then be linked to the created instance.

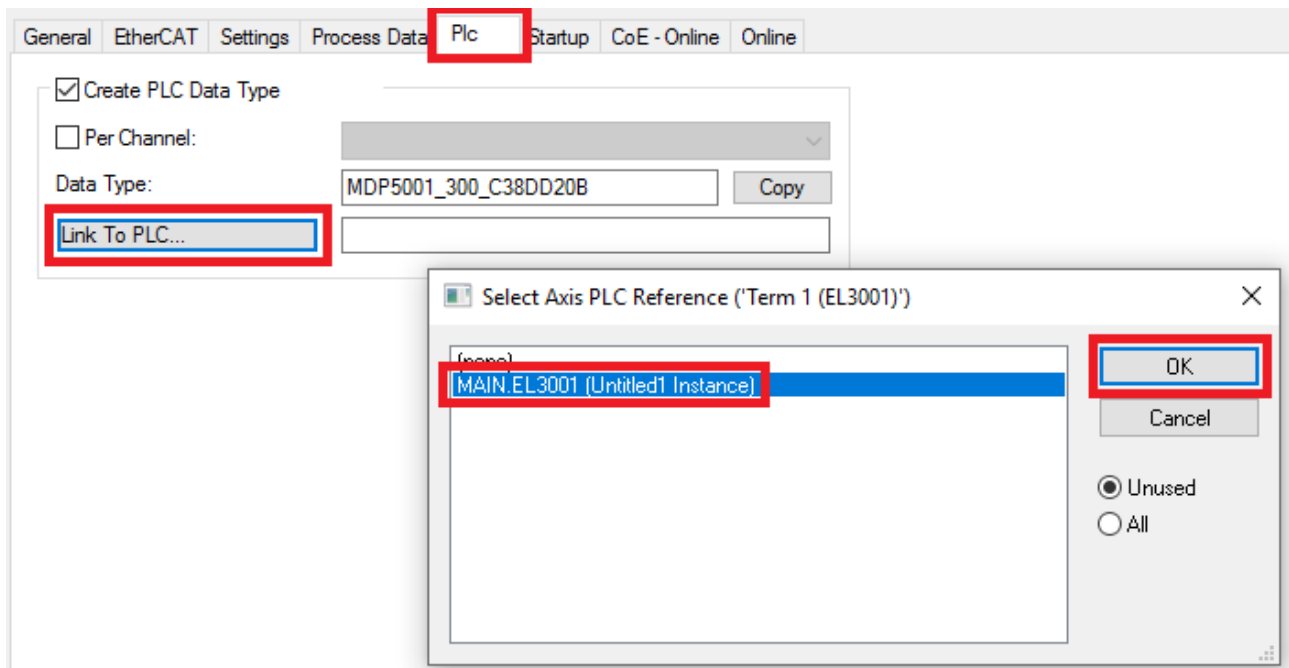


Fig. 114: Linking the structure

7. In the PLC the process data can then be read or written via the structure in the program code.

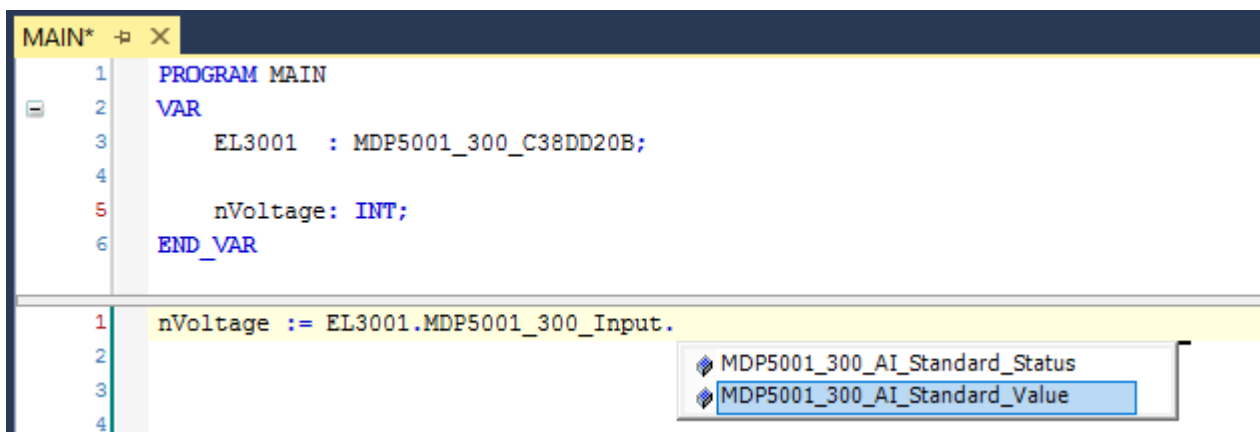
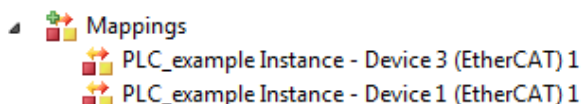


Fig. 115: Reading a variable from the structure of the process data


**Activation of the configuration**

The allocation of PDO to PLC variables has now established the connection from the controller to the inputs

and outputs of the terminals. The configuration can now be activated with  or via the menu under “TwinCAT” in order to transfer settings of the development environment to the runtime system. Confirm the messages “Old configurations are overwritten!” and “Restart TwinCAT system in Run mode” with “OK”. The corresponding assignments can be seen in the project folder explorer:





A few seconds later the corresponding status of the Run mode is displayed in the form of a rotating symbol

 at the bottom right of the VS shell development environment. The PLC system can then be started as described below.



### Starting the controller

Select the menu option “PLC” → “Login” or click on  to link the PLC with the real-time system and load the control program for execution. This results in the message *No program on the controller! Should the new program be loaded?*, which should be acknowledged with “Yes”. The runtime environment is ready for

program start by click on symbol , the “F5” key or via “PLC” in the menu selecting “Start”. The started programming environment shows the runtime values of individual variables:

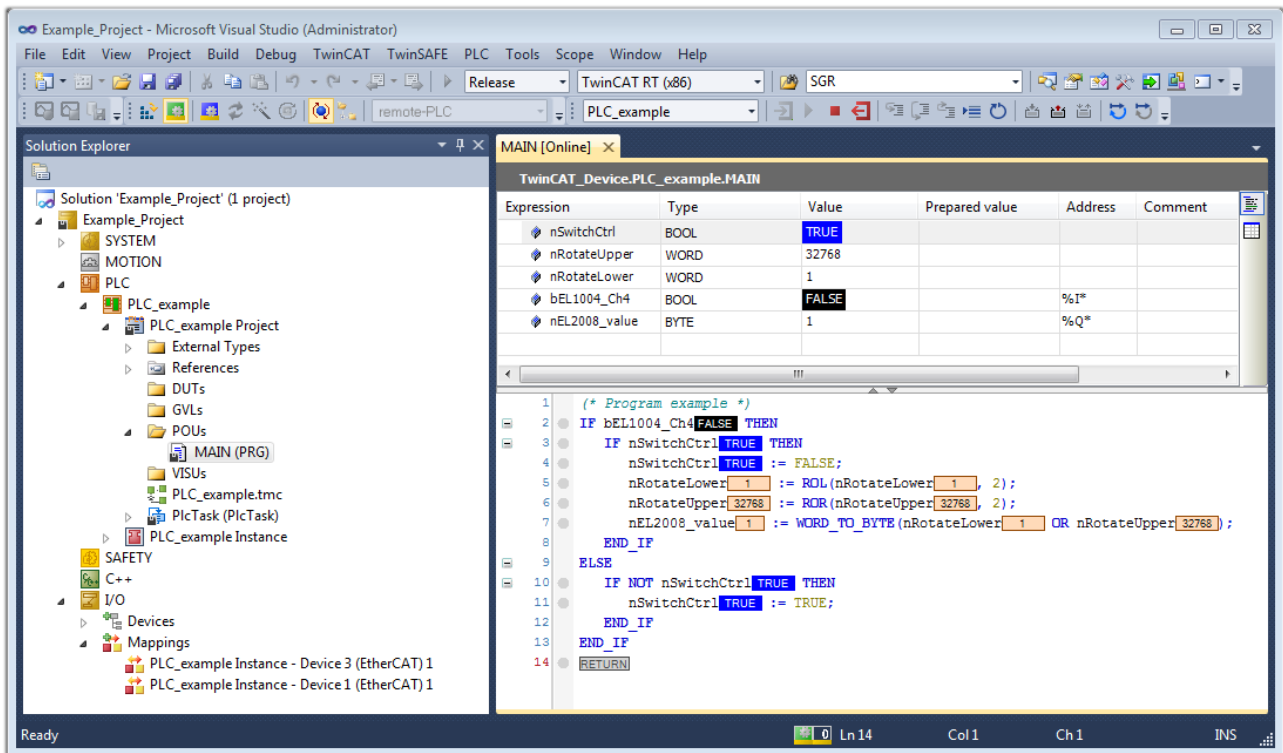


Fig. 116: TwinCAT development environment (VS shell): logged-in, after program startup

The two operator control elements for stopping  and logout  result in the required action (accordingly also for stop “Shift + F5”, or both actions can be selected via the PLC menu).

## 3.2 EtherCAT master in TwinCAT

The Beckhoff TwinCAT System Manager as configuration interface for the I/O environment in the application supports configuration and commissioning of the EtherCAT fieldbus through various automatic actions. The virtual EtherCAT "device" is set up as an independent element in the configuration tree of the System Manager. Its properties can be accessed via associated Properties windows. EtherCAT is defined via the following elements:

- [EtherCAT master in TwinCAT \[► 90\]](#)
- [TwinCAT real-time environment \[► 92\]](#)
- [Ethernet interface \[► 93\]](#)
- [connected EtherCAT slaves \[► 94\]](#)

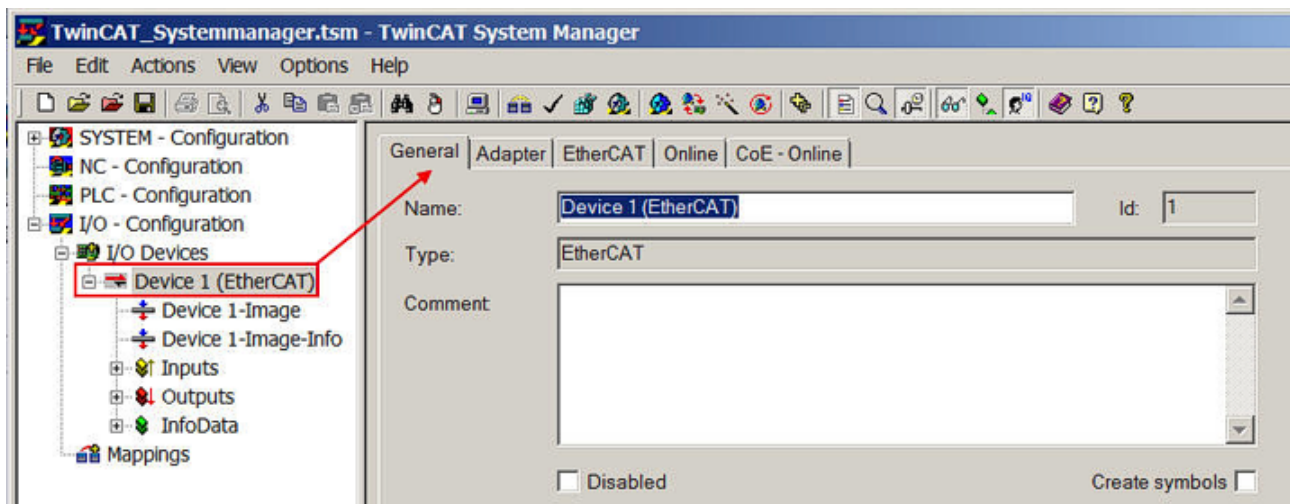


Fig. 117: EtherCAT device in the configuration tree

### The EtherCAT master as a virtual software device

EtherCAT as an Ethernet-based real-time fieldbus requires a physical Ethernet interface at the controller and a real-time trigger as required for connection with the I/O environment. It enables the EtherCAT master to communicate with the connected EtherCAT environment.

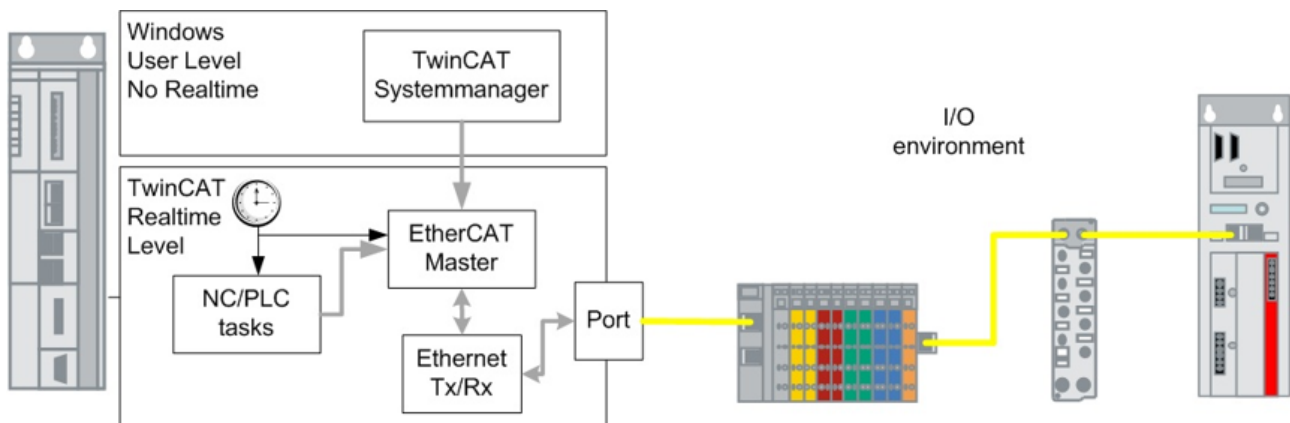


Fig. 118: EtherCAT master in the IPC environment

The EtherCAT master as a software device in TwinCAT:

- can assemble Ethernet telegrams with EtherCAT datagrams from the process data supplied by the PLC/NC/task and send them via the assigned Ethernet port.
- can unpack process data received from the I/O environment and return them to the tasks.
- deals with the construction of cyclic and acyclic telegrams.
- controls the distributed clock synchronization.
- analyses the diagnostic information of the EtherCAT slaves.
- carries out event-oriented diagnostics or responds to modified topologies.
- can only manage *one* EtherCAT system with up to 65,535 slaves and is linked with *one* Ethernet interface ("RJ45 port") for this purpose. A second port can be allocated to the EtherCAT master for the purpose of cable redundancy.
- manages the communication with other EtherCAT masters in the same TwinCAT system, if several masters are present in the configuration.
- is the sole generator of EtherCAT telegrams in an EtherCAT system
- ....

EtherCAT communication consists of **cyclic** and **acyclic** Ethernet telegrams.

The **cyclic** telegrams form the normal process data and can usually not be modified during system runtime. A known configuration, i.e. the number of connected EtherCAT slaves, always results in the same minimum quantity of process data, which have to be sent with the master for cyclic communication between the devices. The cyclic (i.e. at constant intervals triggering task, e.g. a PLC task with a 10 ms runtime) triggers the communication with the EtherCAT field. Once the communication is complete the EtherCAT master returns the process data. On IPC/embedded systems TwinCAT generally operates with constant cycle times between 50  $\mu$ s and > 100 ms. The arithmetic operations of the task must be completed within this cycle time. The high communication performance of EtherCAT (high 100 Mbit/s data throughput) enables the sent EtherCAT telegrams to return to the controller before the start of the next cycle, even in large configurations containing more than 1000 devices. Task execution and bus communication is therefore configured as a synchronous task in TwinCAT. A key factor for high-quality execution is low jitter, i.e. the tasks should be executed and restarted with high time precision.

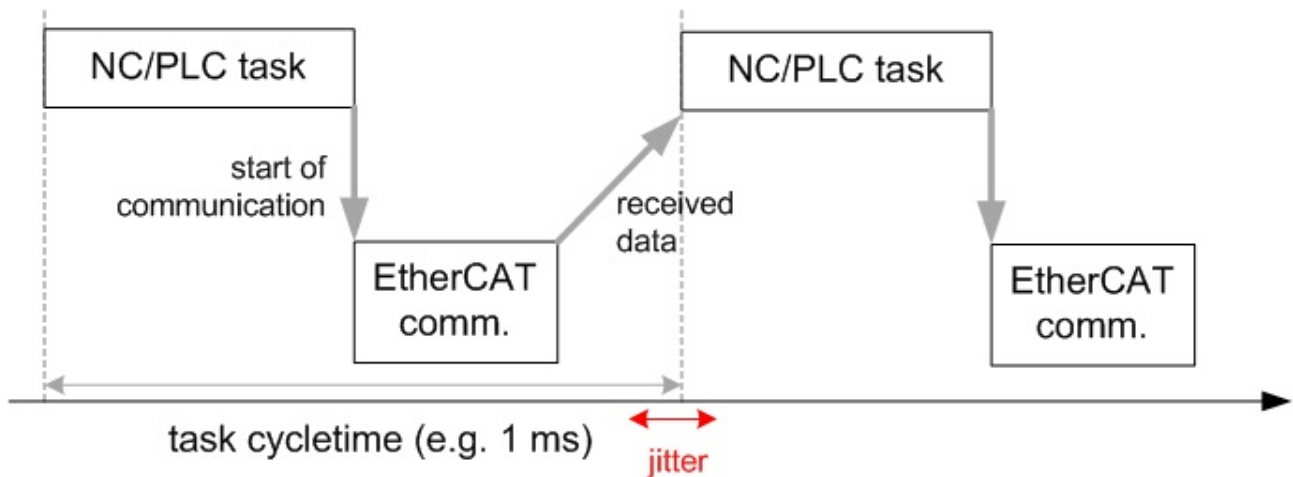


Fig. 119: synchronous execution of task and EtherCAT communication

In Fig. *Synchronous execution of task and EtherCAT communication*, the communication starts after the task. It is also possible to send frames at the same time as the task start time (“I/O at task begin”).

The **acyclic** telegrams are created by the EtherCAT master as required and sent or received in the intervals between the cyclic data. These telegrams are used for gathering diagnostic information and mailbox communication with individual devices (e.g. for firmware updates) or subordinate fieldbuses. Acyclic telegrams are sent from a pipeline without real-time claim in the order they are requested, depending on space, and are therefore also referred to as “queues frames”.

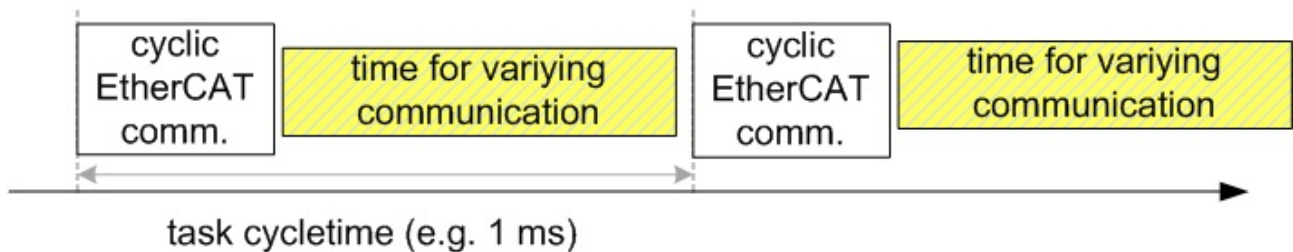


Fig. 120: Distinction between fixed (cyclic) and variable EtherCAT telegrams (acyclic)

The TwinCAT System Manager also maps this time relationship in every configuration created. In Fig. *Configuration-dependent time division in the System Manager*, a 100  $\mu$ s task can be seen that communicates with about 20 slaves and needs an Ethernet frame (red) with several datagrams for the cyclic communication.

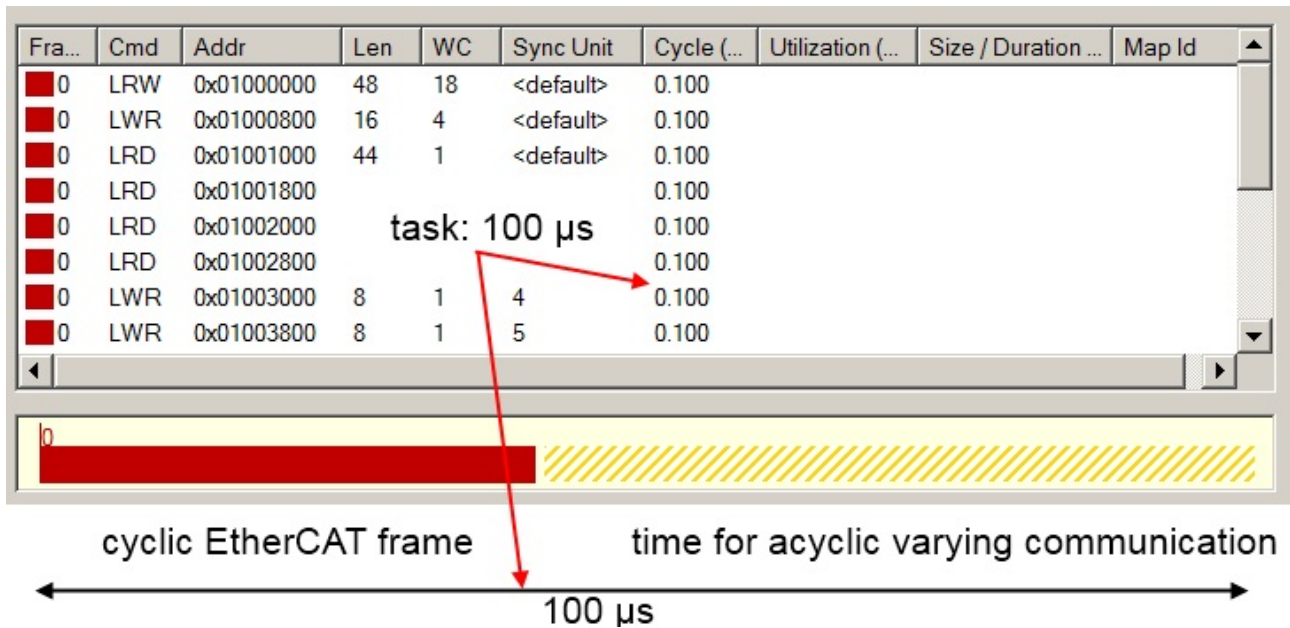


Fig. 121: Configuration-dependent time division in the System Manager

### The real-time environment

The EtherCAT master itself “only” controls the construction and interpretation of the EtherCAT telegrams. A TwinCAT-based controller is based on cyclic execution of a task with constant repeat rate (cycle time). Usual cycle times in a TwinCAT environment range from 50 µs over 1 ms to several 100 ms. The cycle time is selected during configuration setup depending on the processing power of the controller, the bus devices, the executed programs, the application requirements and other factors.

Cyclically executed task may include NC calculations, PLC code, visualizations or R3 applications created and triggered by the customer. The tasks may have different cycle times, but have to be weighted in terms of priority. Higher priority tasks may interrupt/pause lower priority tasks. The lower priority task continues as soon as the priority list permits this. If more than one CPU core is used (only possible with TwinCAT 3), several tasks can be executed in parallel.

In a correctly dimensioned and parameterized TwinCAT system all configured tasks can be executed within the specified cycle time once during the whole cycle, even if they have different priorities. The settings automatically applied by the System Manager during configuration setup usually ensure stable operation of the configuration. The whole cycle is defined by the slowest task. For example, with a 1 ms and a 100 ms task on the same system, the fast task is executed 100 times before the slow task is executed.

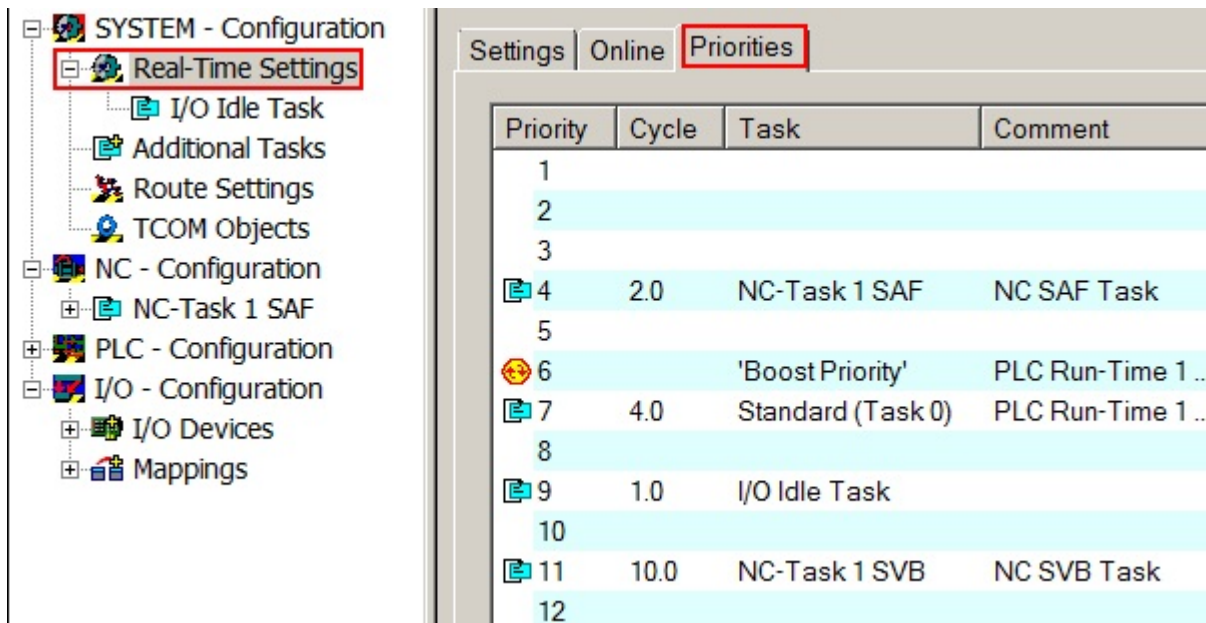


Fig. 122: Priority list in the TwinCAT System Manager

The real-time characteristics of TwinCAT in Windows-based systems (2000, NT, XP, 7, CE; WES, WEC) ensure low jitter and therefore high constancy and exact timing, even with very short cycle times of 50 μs and without dedicated timing hardware.

In principle each task can have its own I/O image and therefore control its own fieldbus cycle and devices. In Fig. 2 *Tasks with own Ethernet frames* (Task 1 ms: "red" frame, task 10 ms: "yellow" frame), two tasks (1 + 10 ms) can be seen that each trigger their own I/O cycles and thus cause their own Ethernet frames.

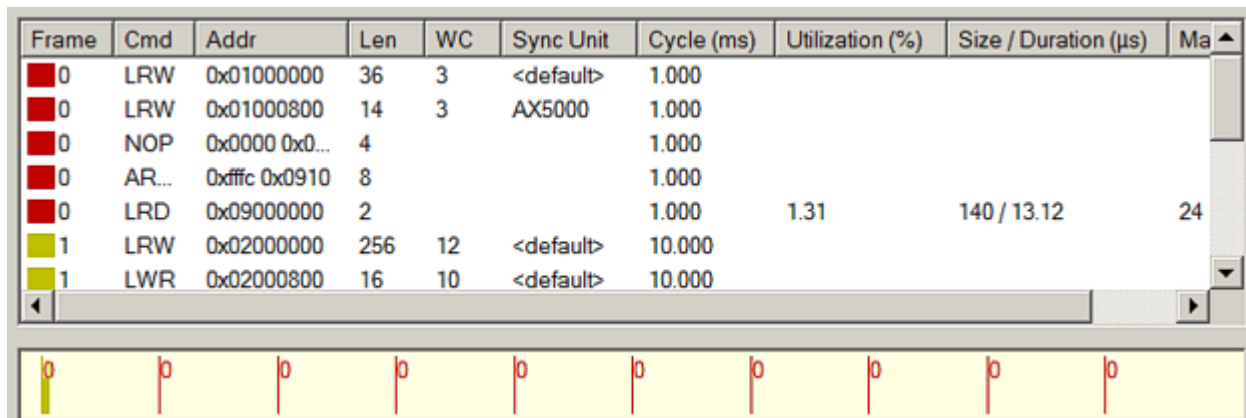


Fig. 123: Tasks with own Ethernet frames (Task 1 ms: "red" frame, task 10 ms: "yellow" frame)

**The Ethernet interface**

EtherCAT is currently (2011) standardized based on ISO/OSI layer 2 with Fast Ethernet = 100 Mbit/s down to the slave layer. The interface therefore has to support Fast Ethernet as a minimum. See also notes regarding EtherCAT infrastructure.

The Ethernet interface allocated to the EtherCAT master must meet the requirements of the application:

- external requirements: temperature, vibration, pull-out protection, buckling protection, contamination. An RJ45, M12 or M8 link is usually used for the connection to the controller/IPC.
- real-time capability  
The time-critical properties of the Ethernet interface must meet the requirements of the application and should not delay telegrams sent by the EtherCAT master at the correct time. This applies to the software implementation (driver, stack management) and the electronic configuration (PCI or USB connection, DMA, NDIS management, upstream switch).

## EtherCAT Slaves

The following factors should be considered for the configuration:

- the properties of an EtherCAT slave are defined for the EtherCAT master in the device description file ESI (EtherCAT Slave Information). This XML file is provided by the respective device manufacturer. By default the TwinCAT System Manager looks for these ESI files under C:\TwinCAT\IO. An ESI file may contain several device descriptions and revisions.
- For EL/ES terminals connected in the terminal strand (LVDS configuration) the E-bus current consumption must be considered. The System Manager keeps track of the power demand specified in the ESI description and issues a warning if there is a risk of overload of the previous coupler. The usual capacity is 2 A. Negative values in column "E-bus" are not allowed and may lead to errors that are difficult to reproduce.

Number	Box Name	Address	Type	In Size	Out Size	E-Bus (mA)
18	Klemme 18 (EL2004)	1018	EL2004		0.4	440
19	Klemme 19 (EL2004)	1019	EL2004		0.4	340
20	Klemme 20 (EL2004)	1020	EL2004		0.4	240
21	Klemme 21 (EL2004)	1021	EL2004		0.4	140
22	Klemme 22 (EL2004)	1022	EL2004		0.4	40
23	Klemme 23 (EL2004)	1023	EL2004		0.4	-60 !
24	Klemme 24 (EL6692)	1024	EL6692	4.0	2.0	-180 !
25	Klemme 25 (EL6731)	1025	EL6731	4.0		-530 !
26	Klemme 26 (EL9010)		EL9010			-530 !

Fig. 124: System Manager view – current calculation

- the maximum cable lengths and attenuation values between devices in the FX/TX layer (Ethernet cable, glass fiber, POF) specified in the physical layer may not be exceeded.
- the number and configuration of the EtherCAT slaves (max. 65,535) has influence on the cycle time of the EtherCAT telegrams.
- the functional reliability of the screen connection of the couplers/terminals and the earthing of the overall application must be ensured.
- supply lines/sensor cables may have to be screened separately.

## 3.3 General notes on the use of Beckhoff EtherCAT IO components

Note: this document provides general assistance. For detailed questions (e.g. "Where can I find the revision?") please refer to the corresponding pages in this [EtherCAT system documentation](#).

Overview:

### 3.3.1 Technical classification

The application and communication properties of a Beckhoff IO device (EtherCAT terminal EL/ES/EM/EK, EtherCAT Box EP/EQ, special designs CU) are determined by the following three elements: the **device**, the **firmware** (optional) and the **EtherCAT slave device description** (ESI).

An electronic automation device with fieldbus connection consists of many further (internal) components and software such as the ESC (EtherCAT Slave Controller, the real-time communication IC); however, the three mentioned above are those that outwardly define the device's properties for the user.

Details of the three components:

The material device itself, the 'hardware' (HW)

- is marked by Beckhoff with the hardware version, e.g. HW01.
- in EtherCAT parlance the device is also often referred to as a **slave** in the sense of the fieldbus topology, because it is addressed by the EtherCAT **master**.

- the hardware version.
  - is printed on the outside of the device, see [serial number/batch number](#) [► 102]
  - can be read from the CoE directory in devices that have one
  - can be read by the EtherCAT master from the EtherCAT/ESI-EEPROM since 2012/01
  - is coded in decimal or hexadecimal depending on the device series

The firmware that runs on it, if applicable

- **Note:** firmware (FW) is the name given to the program code made by the manufacturer (Beckhoff) and executed on a  $\mu$ C (microcontroller, FPGA or processor). Not all devices have to have a  $\mu$ C and thus firmware!
- this can consist of several files if necessary and must be loaded to the IO device. Usually these are \*.efw or \*.rbf files
- is marked by Beckhoff with the firmware version, e.g. FW02
- the firmware version
  - is printed on the outside of the device, see [serial number/batch number](#) [► 102].
  - can be read from the CoE directory in devices that have one.
  - can be read by the EtherCAT master from the ESI-EEPROM since 2012/01.
  - if a firmware update of the device is carried out, the imprint on the housing is to be adapted accordingly by the person performing the update or by the user.

The EtherCAT ESI communication description as the device description file for the EtherCAT master (in case of Beckhoff: integrated in the TwinCAT software)

- describes the EtherCAT communication interface between device and master in all aspects that are relevant to data communication and synchronization
- is on the one hand programmed by Beckhoff on the IO device itself (into the so-called EtherCAT/ESI-EEPROM)
  - so that the device can announce basic information of its own accord on being scanned by the EtherCAT master: size of the process data (PDO), setting options (CoE) and others.
  - in addition this ESI contains the basic settings for the IO device itself that are relevant to its function and are read on the device by the  $\mu$ C or other control components at power-on.
- should on the other hand be available to the EtherCAT master as a file
  - since the EtherCAT master software now 'knows' the slave even without electrical access, the user can also create his bus configuration 'offline', i.e. without live contact to the IO device, as is absolutely necessary when scanning.
  - and in addition the EtherCAT master knows how it must address the slave over EtherCAT and the functions that the latter offers. As a result, cyclic process data and CoE directory of the slave, for example, are determined for the master. Without an ESI file the master does not know the device at all.
  - if an EtherCat slave is to be connected to the NC in TwinCAT, the availability of the ESI device description is absolutely necessary
  - **Note:** there are EtherCAT masters that obtain the device information from the slave only by scanning and do not require any ESI file from the device.  
If necessary TwinCAT can also operate 'on-line' without the availability of an ESI file; however, it is recommended that the correct ESI file be available for TwinCAT for a simple reason: many EtherCAT devices now have a wide range of functions and extensive setting options. This then results in a large ESI file that can perhaps no longer be saved in the local EtherCAT/ESI-EEPROM – although the reduced information which is then available there is sufficient for basic operation of the device, the full function and diagnostic scope of the device is not available.  
In addition, offline configuration is possible only if the ESi file is available.
- is marked by Beckhoff with the so-called revision e.g. -0018  
if there is a change in the contents/function of the ESI, the revision number is usually increased by +1
- the revision
  - can be read by the EtherCAT master from the EtherCAT/ESI-EEPROM.
  - can also be read from the CoE directory in devices that have one.

- has been printed on the outside of the device since 2014/01 → "Rev. xxxx [▶\_102]"
- if a revision update of the device is carried out, the imprint on the housing (if there is one) is to be adapted accordingly by the person performing the update or by the user.

Due to their properties, all three elements can have an effect on

- functional properties, e.g. filter, sample rate, output rate, input sensitivity and others
- temporal behavior e.g. during bootup,
- behavior and diagnosis in case of error
- communication features, e.g. process data, parameter directory,
- Distributed Clocks properties, e.g. types of trigger, synchronicity, latency and others
- exterior appearance

These three elements are to be found as follows in the application world:

### EtherCAT: device components and place of action

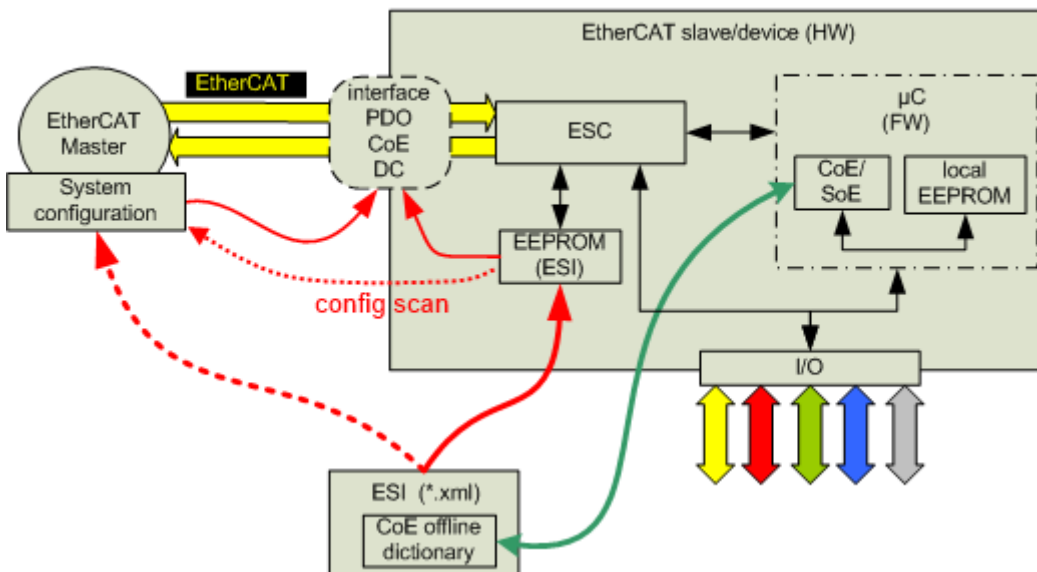


Fig. 125: Components of the EtherCAT device

Explanations on the basis of the principles mentioned above:

- the ESI is programmed in the device (EtherCAT/ESI-EEPROM) and determines the EtherCAT interface from the point of view of the  $\mu\text{C}$ ; the EtherCAT real-time communication is performed by the ESC (EtherCAT Slave Controller).
- the EtherCAT master requires the ESI in order, for example, to incorporate it into its system configuration. To do this it can read the EtherCAT/ESI-EEPROM online. If the ESI is directly available to it, however, then offline configuration is also possible. As a result, master and slave/device now have the same definition of the EtherCAT interface and can communicate with one another.
- the  $\mu\text{C}$  (if there is one) maps the function of the device and communicates with the ESC for this. It also controls the IO side of the device, if this not already done by the ESC.
- the parameter directory 'CoE' (CANopen-over-EtherCAT) of the device is managed by the  $\mu\text{C}$ , i.e. 'online'. The ESI file also contains a copy of this, the so-called 'CoE offline dictionary'. This enables the user to view the possible functions offline in advance and if necessary to preconfigure the start-up list. Changes in the online CoE are saved fail-safe in Beckhoff IO devices and in the AX2000 in general in a local EEPROM (unless parameterized differently). The AX5000, conversely, has an SoE directory (Sercos-over-EtherCAT), which always starts in the default state on power on; changes required by the user must be loaded to the slave, for example by startup list, on each bootup of the EtherCAT master.

Illustrated in a simplified manner, these three elements map the entire device, wherein the ESI is used both in the device and in the master:



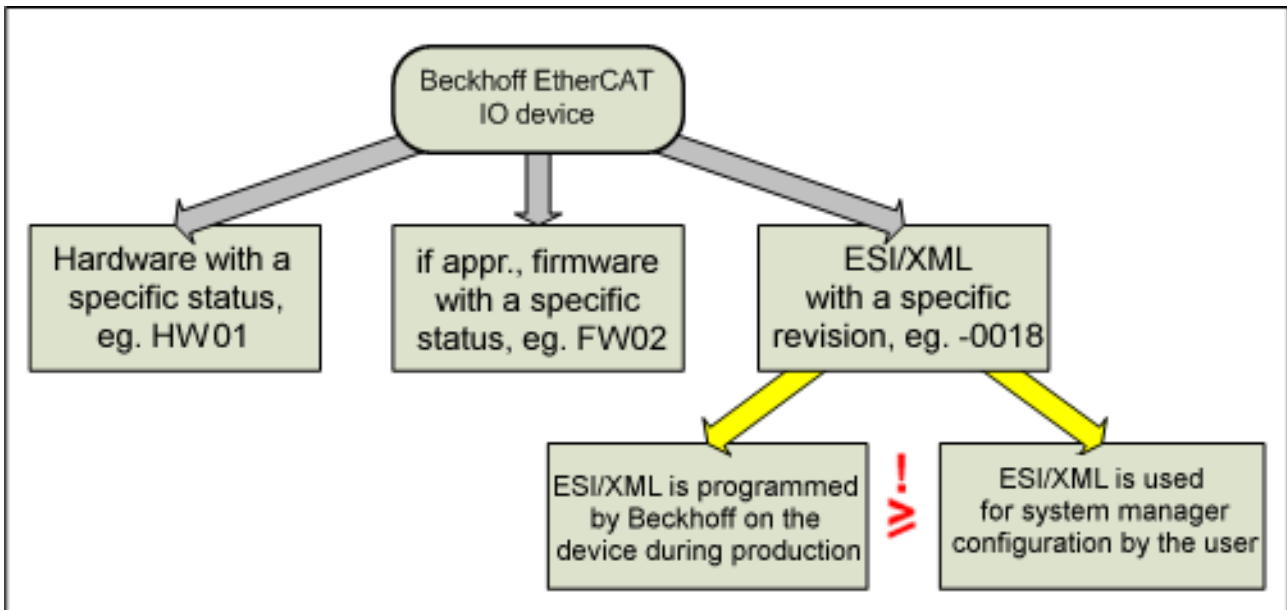


Fig. 126: Simplified illustration of the entire EtherCAT device

Depending on the type, the entire device can also be marked by a sequential, unique device number. See also the [Identification notes \[► 102\]](#).

### 3.3.2 Change management by Beckhoff

Beckhoff reserves the right to change all three elements for itself and unannounced during the product lifetime in order

- to introduce functional improvements and
- to implement new functions

Such a change by Beckhoff

- is then usually marked by a changed HW/FW/revision version.
- takes place in such a way that the Beckhoff EtherCAT IO compatibility rule can be adhered to in existing applications:

**device revision in the system >= device revision in the configuration**

Background: The ESI description also defines the process image, the communication type between master and slave/device and the device functions, if applicable. The physical device (including firmware, if available) has to support the communication queries/settings of the master. This is the case downwardly compatible, i.e. newer devices (higher revision, higher firmware version, higher hardware version) should also support it if the EtherCAT master addresses them as an older revision due to an existing configuration. This permits the later swapping/replacement of devices if necessary without changing the configuration, i.e. without access to the application files.

**Sample:** If the configuration provides for an EL2521-0025-1018 EtherCAT terminal (i.e. revision -1018), then a terminal that is programmed as an EL2521-0025-1018 or higher (- 1019, -1020) can be used as a real IO device. Higher revision also means in many cases a newer firmware version, but at least the same firmware version on the part of Beckhoff production. For Beckhoff and the user this means that the further developed or changed edition of an EtherCAT IO product with incremented hardware, firmware or revision version should support all of the properties and features of the predecessor products with a lower FW or revision version.

### ● ESI device descriptions with revision history



The ESI-XML files (zipped) offered for downloading on the [Beckhoff website](#) contain almost all EtherCAT devices ever published together with their complete revision history. The user thus has the possibility to deliberately integrate predecessor revisions in his configuration.

### ● Beckhoff ESI device descriptions, compatibility



From the IO compatibility rule it follows that, even if the default process data have been changed with regard to the predecessor model in a newer firmware/revision, this device can nevertheless still be converted via the PDO setting (e.g. PredefinedPDOsettings) to the "old" process data or should at least "understand" them, because the new firmware/revision ought to support them without fail. The respective device documentation provides assistance with this on the process data page.

## 3.3.3 Procurement/update of the elements

The three elements can be procured:

### HW:

- **Procurement:** only by purchase from/exchange by Beckhoff
- **Update:** An update of the hardware is usually possible only by exchanging the device.

### FW:

- **Procurement:** by enquiry to Beckhoff/Support
- **Update:** An update of the firmware is usually described in the device documentation and must be carried out using the Beckhoff System Manager. You must check with Beckhoff, or in the documentation specifications, which possible firmware version a certain available device supports in accordance with its hardware version.

**Note/hint:** the readable FW version on the exterior of the device must be manually adapted by the person carrying out the update so that the new FW version is also visibly recognizable.

### ESI/XML:

- **Procurement:** the current version is available for download on the [Beckhoff website](#) and, when a new TwinCAT build is written, the ESI current at the time of writing is integrated into the TwinCAT installation. Depending on which TwinCAT version is installed, the device descriptions located in C:\TwinCAT\IO\ after the TwinCAT installation may not correspond to the current Beckhoff version.
- **Update:**
  - in the System Manager: the ESI files (\*.xml, \*.xsd and if necessary further files in the same folder) in the TwinCAT master directory (e.g. TwinCAT 2: C:\TwinCAT\IO\) must be exchanged. Afterwards is it necessary to restart the System Manager.
  - in the device: An update of the revision is usually described in the device documentation and must be carried out using the Beckhoff System Manager. You must check with Beckhoff, or in the documentation specifications, which possible revision version a certain available device supports in accordance with its hardware version.

**Note/hint:** the readable revision version on the exterior of the device must be manually adapted by the person carrying out the update so that the new revision version is also visibly recognizable.

### "Never touch a running system!"

An update of firmware or revision should be carried out only if there is a justified cause to do so. If both are changed, then as a rule the firmware is to be changed first, then the XML/revision in the EEPROM. Notes in the documentation are to be observed!

There is no right to demand that Beckhoff hands out information on HW/FW/Revision changes.

**i** **Downloading ESI descriptions**

Beckhoff makes the current versions of the ESI device descriptions available for download on its website. These ESIs are thus available to every EtherCAT user, every configuration creator and every customer. However, it is recommended that configuration creators/programmers internally clarify within their company/firm/group which revision versions are to be released for internal company use, so that a uniform configuration version can be maintained over the years in delivered machines. This simplifies the stocking of spare parts.

**3.3.4 Application project planning in the TwinCAT System Manager 2.x/3.x - creating the configuration**

An EtherCAT configuration can be created in the TwinCAT System Manager in three ways:

- by manual configuration creation without an existing device, so-called offline creation → see chapter "[Configuration creation – manual](#) [▶ 122]"
- by online scanning of the device connected to the master --> see chapter "[Configuration creation – online scan](#) [▶ 127]"
- by automated configuration creation without operator intervention, e.g. via automation interface

The TwinCAT System Manager 2.x and TwinCAT 3.0/3.1 behave as follows:

- **Offline creation:** Chapter '[Configuration creation – manual](#)' [▶ 122]

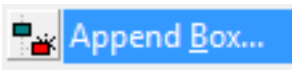


Fig. 127: Context menu "Append Box"

Primarily, only **the highest/last revision** that exists in the ESI directory on the local programming device is presently offered for insertion by the System Manager in the selection dialogue for the IO device to be inserted.

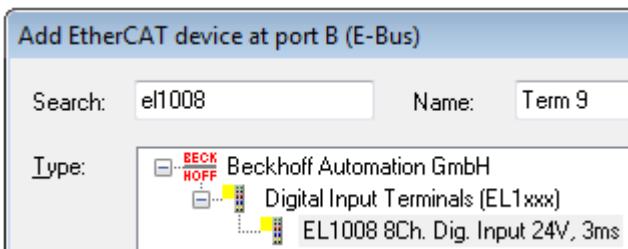


Fig. 128: Selecting and appending an EtherCAT device

Here, for example, the EL1008.

The corresponding checkbox must be activated in order to see the revision.

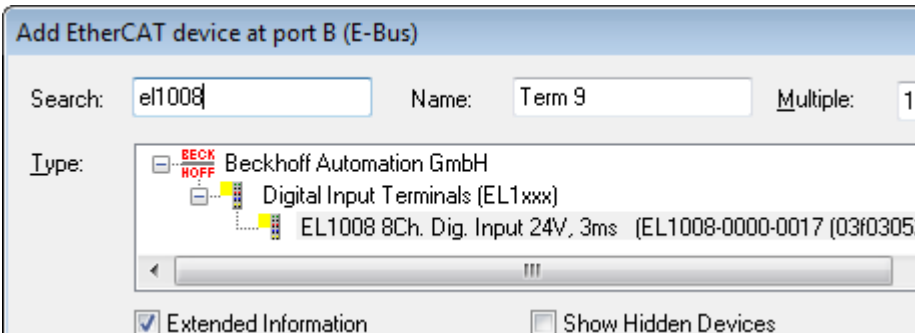


Fig. 129: Activate checkbox "Extended Information" to display revision

If a different/lower revision is to be inserted into the configuration, this must be displayed via Show Hidden Devices and then selected.

Please observe the Beckhoff compatibility rule.

- **Online scan:** Chapter 'Configuration creation – online scan' [▶ 127]

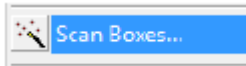


Fig. 130: Context menu "Scan Boxes"

TwinCAT reads the manufacturer, name and revision from the ESI-EEPROM on the IO device found, e.g. "Beckhoff EL2502-0000-0018". The System Manager then looks in its ESI directory for the associated ESI file. If this is missing, the System Manager can read the online description from the device and use it (this is not recommended; observe the ESI description in the technical classification and [scan page](#) [▶ 127]!). Precisely this revision with its properties (process data, CoE directory, etc.) is then integrated into the configuration (the Beckhoff compatibility rule must be observed).

In order to support all functions and diagnostic options of the IO device, users are urgently recommended to interrupt the process if the corresponding ESI file is missing and to procure the corresponding ESI file from the device manufacturer and place it in the ESI directory.

- **Automated creation:**  
the revision stored in the control program is built into the configuration. Otherwise the previous sentences apply.

#### NOTE

##### Revision and functions of the EtherCAT device

Care must be taken when creating a configuration that only those device functions and thus ultimately only those revisions that are supported by the customer's existing terminals are configured (e.g. in the spare part stock).

### 3.3.5 Configuration comparison in the project

The TwinCAT System Manager allows a comparison between the configuration according to the \*.tsm file and the IO combination that is actually connected to the EtherCAT master. The comparison takes place automatically if devices are found during the scanning process.

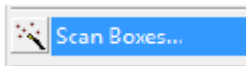


Fig. 131: Context menu "Scan Boxes"

An overview appears showing the differences; for further information see chapter 'Configuration creation – online scan [▶ 127]'

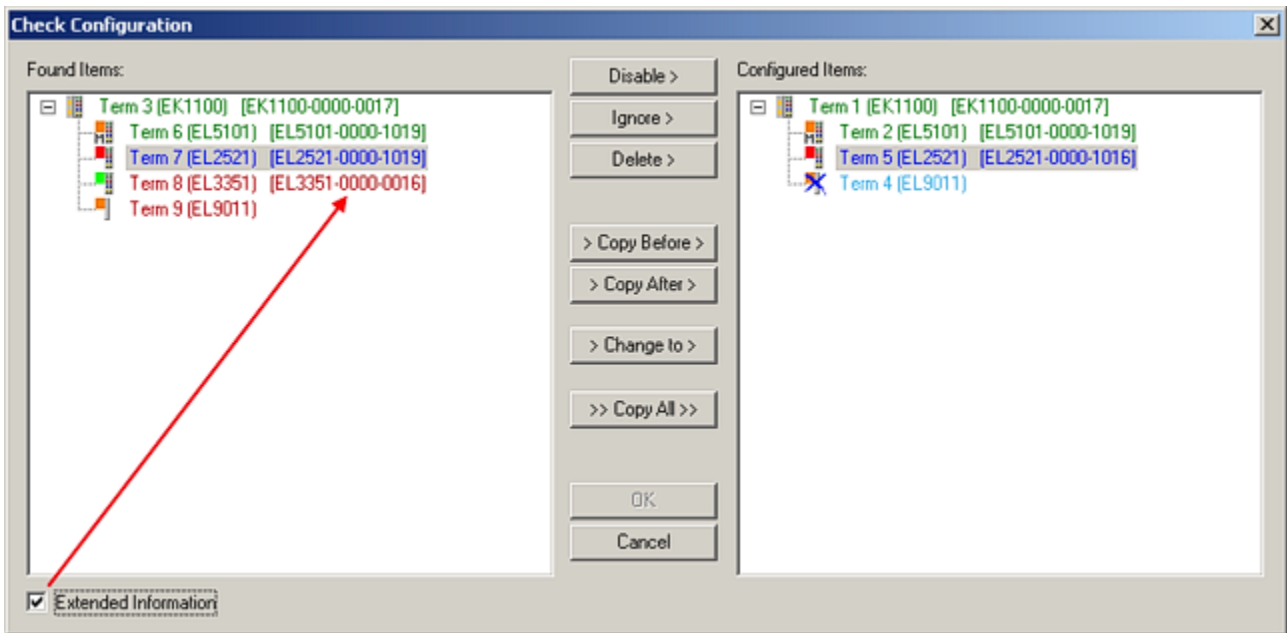


Fig. 132: Configuration comparison

### 3.3.6 Determining the installation version in the application

If an overview of the real EtherCAT IO components installed in the application is desired

- the imprinted designations and serial numbers of the terminals/boxes can be read.
- the HW/FW versions and production date can be read online from the ESI-EEPROM by the System Manager from TwinCAT 2.11R3 b2235

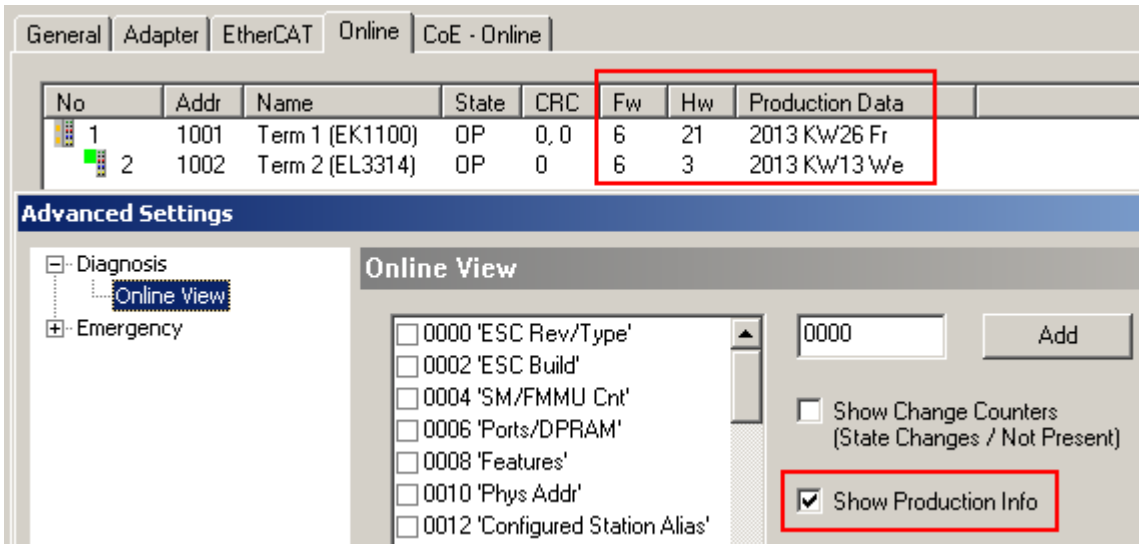


Fig. 133: Reading product information from EEPROM

Blocks for reading this information from the PLC are being prepared. It is possible to export this information to Excel:

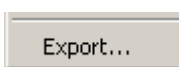


Fig. 134: Exporting the information

See also the [chapter "Online version identification of EtherCAT devices"](#) [▶ 107]

- the information of the identity object can be read into the PLC from devices with a CoE directory by online ADS access

1018:0	Identity	RO	> 4 <
1018:01	Vendor ID	RO	0x00000002 (2)
1018:02	Product code	RO	0x0CF23052 (217198674)
1018:03	Revision	RO	0x03F90000 (66650112)
1018:04	Serial number	RO	0x00000000 (0)

Fig. 135: Reading the information from identity object 1018

See also the sample program in the [chapter "Online version identification of EtherCAT devices" \[► 107\]](#).

## 3.4 Version identification of EtherCAT devices

### Designation

A Beckhoff EtherCAT device has a 14-digit designation, made up of

- family key
- type
- version
- revision

Example	Family	Type	Version	Revision
EL3314-0000-0016	EL terminal (12 mm, non-pluggable connection level)	3314 (4-channel thermocouple terminal)	0000 (basic type)	0016
ES3602-0010-0017	ES terminal (12 mm, pluggable connection level)	3602 (2-channel voltage measurement)	0010 (high-precision version)	0017
CU2008-0000-0000	CU device	2008 (8-port fast ethernet switch)	0000 (basic type)	0000

### Notes

- The elements mentioned above result in the **technical designation**. EL3314-0000-0016 is used in the example below.
- EL3314-0000 is the order identifier, in the case of "-0000" usually abbreviated to EL3314. "-0016" is the EtherCAT revision.
- The **order identifier** is made up of
  - family key (EL, EP, CU, ES, KL, CX, etc.)
  - type (3314)
  - version (-0000)
- The **revision** -0016 shows the technical progress, such as the extension of features with regard to the EtherCAT communication, and is managed by Beckhoff. In principle, a device with a higher revision can replace a device with a lower revision, unless specified otherwise, e.g. in the documentation. Associated and synonymous with each revision there is usually a description (ESI, EtherCAT Slave Information) in the form of an XML file, which is available for download from the Beckhoff web site. From 2014/01 the revision is shown on the outside of the IP20 terminals, see Fig. "EL5021 EL terminal, standard IP20 IO device with batch number and revision ID (since 2014/01)".
- The type, version and revision are read as decimal numbers, even if they are technically saved in hexadecimal.

### Identification number

Beckhoff EtherCAT devices from the different lines have different kinds of identification numbers:

**Production lot/batch number/serial number/date code/D number**

The serial number for Beckhoff IO devices is usually the 8-digit number printed on the device or on a sticker. The serial number indicates the configuration in delivery state and therefore refers to a whole production batch, without distinguishing the individual modules of a batch.

Structure of the serial number: **KK YY FF HH**

- KK - week of production (CW, calendar week)
- YY - year of production
- FF - firmware version
- HH - hardware version

Example with

Ser. no.: 12063A02: 12 - production week 12 06 - production year 2006 3A - firmware version 3A 02 - hardware version 02

Exceptions can occur in the **IP67 area**, where the following syntax can be used (see respective device documentation):

Syntax: D ww yy x y z u

- D - prefix designation
- ww - calendar week
- yy - year
- x - firmware version of the bus PCB
- y - hardware version of the bus PCB
- z - firmware version of the I/O PCB
- u - hardware version of the I/O PCB

Example: D.22081501 calendar week 22 of the year 2008 firmware version of bus PCB: 1 hardware version of bus PCB: 5 firmware version of I/O PCB: 0 (no firmware necessary for this PCB) hardware version of I/O PCB: 1

**Unique serial number/ID, ID number**

In addition, in some series each individual module has its own unique serial number.

See also the further documentation in the area

- IP67: [EtherCAT Box](#)
- Safety: [TwinSafe](#)
- Terminals with factory calibration certificate and other measuring terminals

**Examples of markings**



Fig. 136: EL5021 EL terminal, standard IP20 IO device with serial/ batch number and revision ID (since 2014/01)



Fig. 137: EK1100 EtherCAT coupler, standard IP20 IO device with serial/ batch number

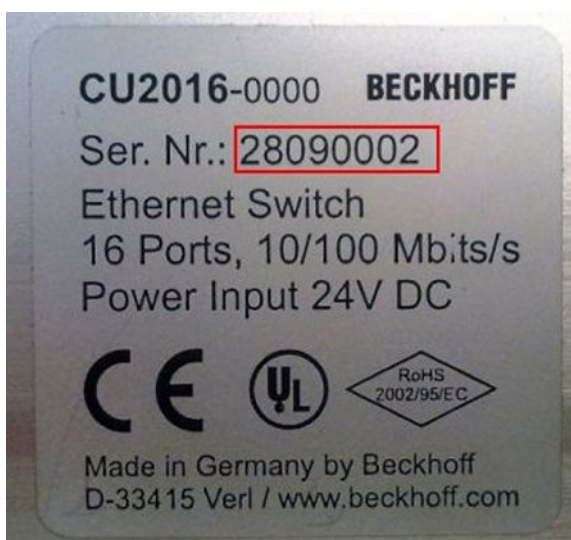


Fig. 138: CU2016 switch with serial/ batch number

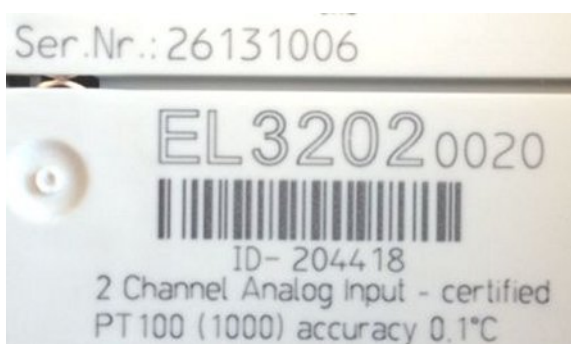


Fig. 139: EL3202-0020 with serial/ batch number 26131006 and unique ID-number 204418



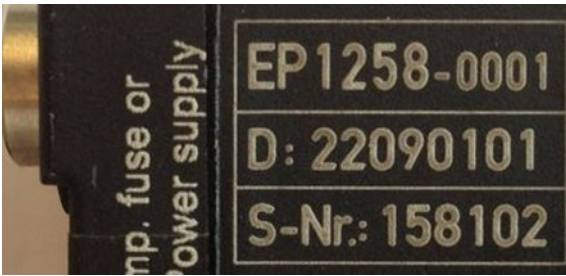


Fig. 140: EP1258-00001 IP67 EtherCAT Box with batch number/ date code 22090101 and unique serial number 158102

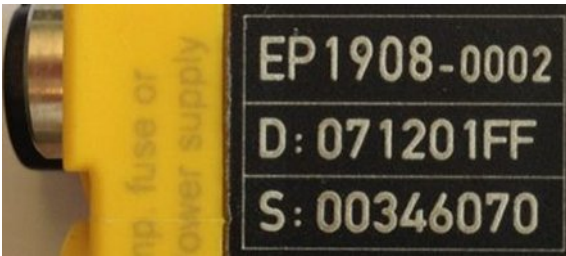


Fig. 141: EP1908-0002 IP67 EtherCAT Safety Box with batch number/ date code 071201FF and unique serial number 00346070



Fig. 142: EL2904 IP20 safety terminal with batch number/ date code 50110302 and unique serial number 00331701



Fig. 143: ELM3604-0002 terminal with unique ID number (QR code) 100001051 and serial/ batch number 44160201

### 3.4.1 Beckhoff Identification Code (BIC)

The Beckhoff Identification Code (BIC) is increasingly being applied to Beckhoff products to uniquely identify the product. The BIC is represented as a Data Matrix Code (DMC, code scheme ECC200), the content is based on the ANSI standard MH10.8.2-2016.

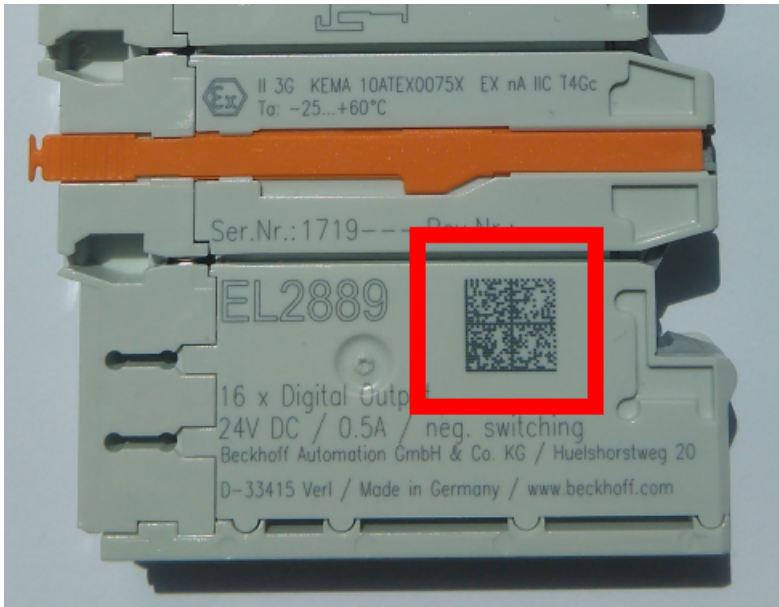


Fig. 144: BIC as data matrix code (DMC, code scheme ECC200)

The BIC will be introduced step by step across all product groups.

Depending on the product, it can be found in the following places:

- on the packaging unit
- directly on the product (if space suffices)
- on the packaging unit and the product

The BIC is machine-readable and contains information that can also be used by the customer for handling and product management.

Each piece of information can be uniquely identified using the so-called data identifier (ANSI MH10.8.2-2016). The data identifier is followed by a character string. Both together have a maximum length according to the table below. If the information is shorter, spaces are added to it. The data under positions 1 to 4 are always available.

The following information is contained:

Item no.	Type of information	Explanation	Data identifier	Number of digits incl. data identifier	Example
1	Beckhoff order number	<b>Beckhoff order number</b>	1P	8	<b>1P</b> 072222
2	Beckhoff Traceability Number (BTN)	<b>Unique serial number, see note below</b>	S	12	<b>S</b> BTNk4p562d7
3	Article description	<b>Beckhoff article description, e.g. EL1008</b>	1K	32	<b>1K</b> EL1809
4	Quantity	<b>Quantity in packaging unit, e.g. 1, 10, etc.</b>	Q	6	<b>Q</b> 1
5	Batch number	Optional: Year and week of production	2P	14	<b>2P</b> 401503180016
6	ID/serial number	Optional: Present-day serial number system, e.g. with safety products	51S	12	<b>51S</b> 678294104
7	Variant number	Optional: Product variant number on the basis of standard products	30P	32	<b>30P</b> F971, 2*K183
...					

Further types of information and data identifiers are used by Beckhoff and serve internal processes.

**Structure of the BIC**

Example of composite information from item 1 to 4 and 6. The data identifiers are marked in red for better display:

**BTN**

An important component of the BIC is the Beckhoff Traceability Number (BTN, item no. 2). The BTN is a unique serial number consisting of eight characters that will replace all other serial number systems at Beckhoff in the long term (e.g. batch designations on IO components, previous serial number range for safety products, etc.). The BTN will also be introduced step by step, so it may happen that the BTN is not yet coded in the BIC.

<b>NOTE</b>
This information has been carefully prepared. However, the procedure described is constantly being further developed. We reserve the right to revise and change procedures and documentation at any time and without prior notice. No claims for changes can be made from the information, illustrations and descriptions in this information.

### 3.5 Online version identification of EtherCAT devices

The production information for an EtherCAT slave device

- Firmware version
- Hardware version
- Date of manufacture

is readable as described above by the serial number marked by laser on the side. The firmware /hardware (FW/HW) version is also electronically readable from there in the case of "complex" devices with CoE capability, but not in the case of simple devices without CoE.

Therefore, the production information listed above has been programmed into EtherCAT Terminals by Beckhoff since the production date July 2012. It is located in the EtherCAT ESI EEPROM, which is a component of every Beckhoff EtherCAT slave.

Information	ESI/XML (configuration)	CoE or, if applicable, default data in the offline dictionary	ESI EEPROM	imprint/label on the side
<b>Availability</b>	<b>all EtherCAT slaves</b>	<b>only in case of complex devices</b>	<b>all EtherCAT slaves</b>	<b>all EtherCAT slaves</b>
<b>Device name</b>	x	0x1008	x	x
<b>Hardware version</b>		0x1009	x ("production info")	x
<b>Firmware version</b>		0x100A	x ("production info")	x
<b>Vendor ID (e.g. Beckhoff: 0x2)</b>		0x1018:01	x	
<b>Product code (32 bit)</b>	x	0x1018:02	x	
<b>Revision (32 bit)</b>	x	0x1018:03	x	x
<b>Serial number</b>		0x1018:04	x	(x)
<b>ID number</b>		(in some series, see respective documentation)		(x)
<b>Date of manufacture</b>			x ("production info")	x

#### Notices on electronic readout

- ESI/XML
  - these data can be read by the user program (PLC, non-RT task) from the EtherCAT master being used
  - for TwinCAT the functions from the TcEtherCAT.lib are to be used here (other EtherCAT masters depending on capability)
  - the data are available online (with live connection to the device) and offline
- CoE (online)
  - these data can be read by the user program (PLC, non-RT task) from the CoE directory of the device over EtherCAT
  - for TwinCAT the functions from the TcEtherCAT.lib are to be used here (other EtherCAT masters depending on CoE capability)
  - the data are available only online  
(In the case of offline access, an EtherCAT master such as TwinCAT can present the data case by case from the ESI Dictionary of the XML file - however, these are then only the data of a foreseen slave).
- ESI EEPROM
  - these data can
    - be displayed online by a TwinCAT 2.11R3 from build 2035
    - be read by the user program (PLC, non-RT task) from the ESI EEPROM of the device over EtherCAT (see sample program)
  - for TwinCAT the functions from the TcEtherCAT.lib are to be used here (other EtherCAT masters depending on CoE capability)
  - the data are available only online
- imprint on the side: not possible electronically.

**Online Production Info (ESI EEPROM)**

**NOTE**

**Data storage in the ESI EEPROM**

In case of an EEPROM update of the EtherCAT slave by the EtherCAT master, e.g. for the purpose of a revision update, the production data located there are overwritten if the EtherCAT master does not exclude these data from being overwritten. From TwinCAT 2.11 the System Manager monitors these data during the update.

From TwinCAT 2.11R3 build 2035 the production information for HW, FW and date can be displayed in the system manager and exported.

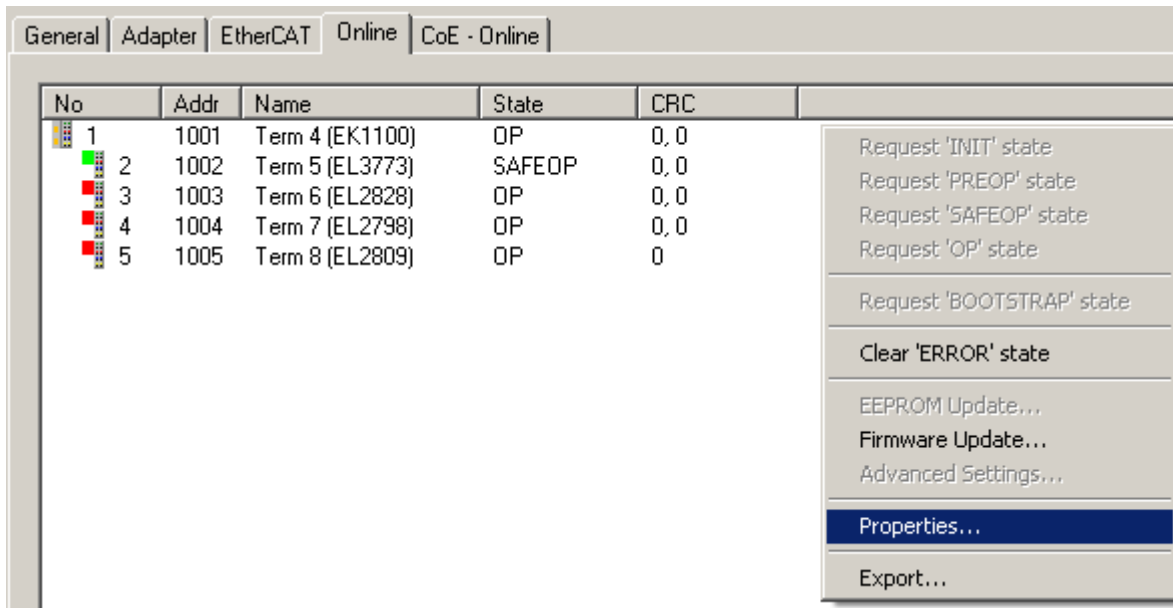


Fig. 145: System manager Online View -> Properties

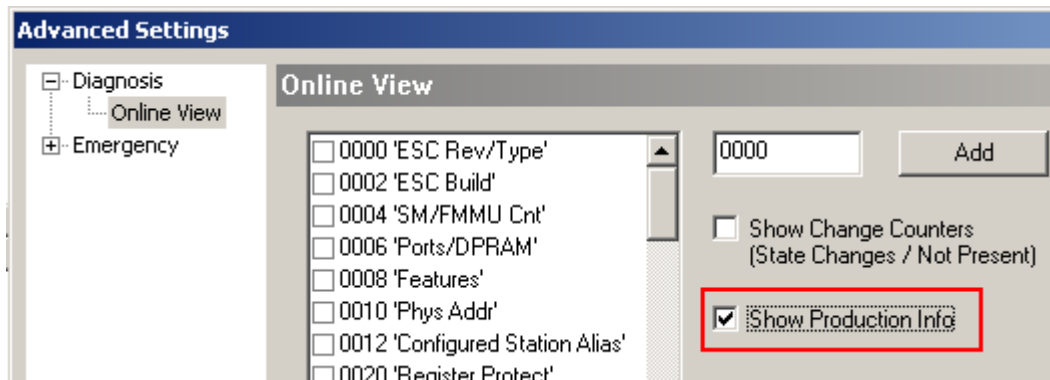


Fig. 146: Activate "Show Production Info"

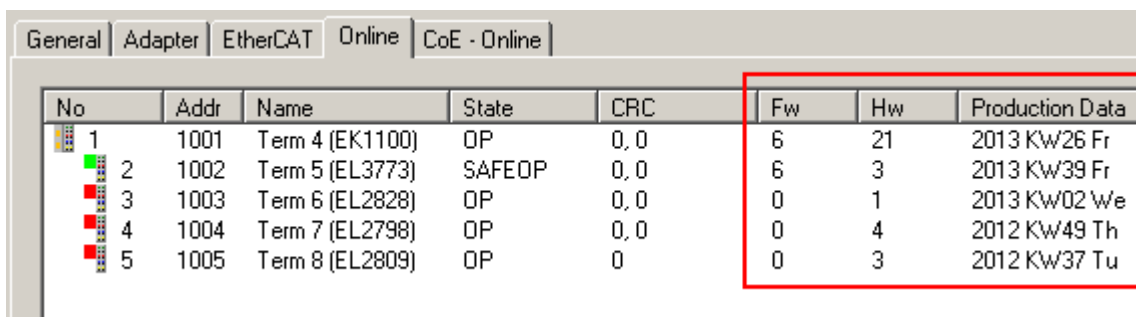


Fig. 147: Online display

Devices without this information display 0 in the overview.

From TwinCAT 3.1, the display is activated via the Advanced Settings of the EtherCAT master:

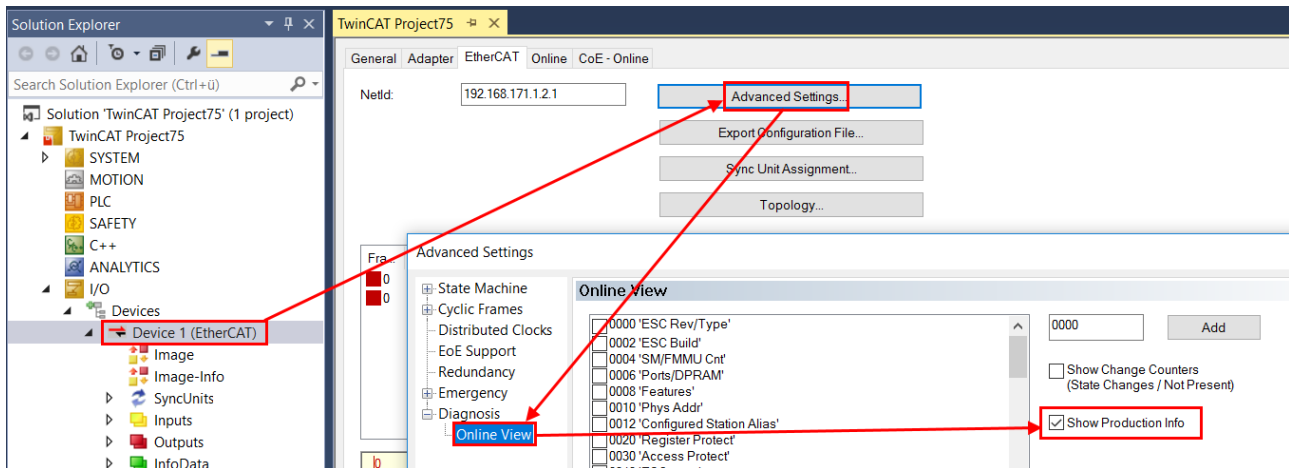


Fig. 148: TwinCAT 3.1: Activate "Show Production Info"

The data can be exported directly to a CSV file.

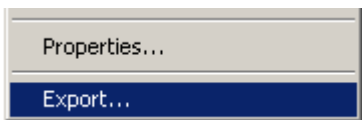


Fig. 149: Export

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Name	PhysAddr	AutoIncAddr	VendorId	ProductionNo	RevisionNo	SerialNo	Hardware Version	Firmware Version	Production Year	Production Week	Production DayOfWeek	State
2	Term 4 (EK1100)	Dx3e9	Dx0	Dx2	Dx44c2c52	Dx120000	Dx0	Dx15	Dx6	2013	26	Fr	Dx8
3	Term 5 (EL3773)	Dx3ea	Dxfff	Dx2	Dxebd3052	Dx130000	Dx0	Dx3	Dx6	2013	39	Fr	Dx4
4	Term 6 (EL2828)	Dx3eb	Dxfffe	Dx2	Dxb0c3052	Dx100000	Dx0	Dx1	Dx0	2013	2	We	Dx8
5	Term 7 (EL2798)	Dx3ec	Dxfffd	Dx2	Dxae3052	Dx110000	Dx0	Dx4	Dx0	2012	49	Th	Dx8
6	Term 8 (EL2809)	Dx3ed	Dxfffc	Dx2	Dxaf93052	Dx110000	Dx0	Dx3	Dx0	2012	37	Tu	Dx8
7													

Fig. 150: Sample of the contents of the csv file

### Sample program for reading and exporting the "production info" from the PLC

#### ● Using the sample programs

**i**

This document contains sample applications of our products for certain areas of application. The application notices provided here are based on typical features of our products and only serve as samples. The notices contained in this document explicitly do not refer to specific applications. The customer is therefore responsible for assessing and deciding whether the product is suitable for a particular application. We accept no responsibility for the completeness and correctness of the source code contained in this document. We reserve the right to modify the content of this document at any time and accept no responsibility for errors and missing information. The program presented below serves as an initial introduction to the options for analyzing the slave data and production information. The user is free to change the program to suit his ideas or to use only part of the code.

 Sample program (<https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/2469151627.zip>)

In this sample program some identification data of the connected EtherCAT slave are read out and displayed in a visualization (Fig. *System Manager Online View -> Properties*). The configured data are read out using the function blocks from TcEtherCAT.lib. Online data such as manufacturer (vendor) and production date are determined by access to the ESI EEPROM. Communication with the EEPROM can be established simply and quickly by the use of the function block 'FB\_ReadEEPROM' included in this project. In addition there is an option to export the data in a file in the text table formats '\*.txt' and '\*.csv'. In the '\*.csv' option the German standard with a semicolon as separator is selected.

### Beckhoff Automation

**PLC-Information**

Projectname:	ReadProductInfo		
Runtime:	1		
Slavecount:	35		
Config. Cyclotime:	1 ms		
Cycle (Min,Act,Max):	3100 ns	9500 ns	131100 ns

**Configured Data:**

Address:	1002
Name:	Klemme 2 (EL2032)
Typ:	EL2032
ProductCode:	133181522
Revision:	1048576

**Online Data are up to date**

Vendor:	Beckhoff Automation GmbH
Productiondate:	FW: 09, HW: 15, TUE, 23/2012
Devicestate:	DP
Linkstate:	DP
State: Finish	

**Saving**

Adresse	Name	Typ	ProductCode
RevisionNo	Vendor	Productiondate	
Path: E:\ExportCSV .csv			
SAVE			

Address	Name
1001	Klemme 1 (EK1100)
1002	Klemme 2 (EL2032)
1003	Klemme 3 (EL1024)
1004	Klemme 4 (EL1252-0050)
1005	Klemme 5 (EL1252)
1006	Klemme 6 (EL1252)
1007	Klemme 7 (EL1819)
1008	Klemme 8 (EL2798)
1009	Klemme 9 (EL2828)
1010	Klemme 10 (EL2809)
1011	Klemme 11 (EL2809)
1012	Klemme 12 (EK1100)
1013	Klemme 13 (EL3318)
1014	Klemme 14 (EL3351)
1015	Klemme 15 (EL3255)
1016	Klemme 16 (EL3351)
1017	Klemme 17 (EL3351)
1018	Klemme 18 (EL2535)
1019	Klemme 19 (EL2602)
1020	Klemme 20 (EL1252)
1021	Klemme 21 (EL2612)
1022	Klemme 22 (EK1100)
1023	Klemme 23 (EL3001)
1024	Klemme 24 (EL3002)
1025	Klemme 25 (EL4102)
1026	Klemme 26 (EL6001)
1027	Klemme 27 (EL2521)
1028	Klemme 28 (EL2521)
1029	Klemme 29 (EL2521)
1030	Klemme 30 (EK1100)
1031	Klemme 31 (EL2612)
1032	Klemme 32 (EL2612)
1033	Klemme 33 (EL2622)
1034	Klemme 34 (EL2622)
1035	Klemme 35 (EL2624)

Fig. 151: Visualization in the sample program

In common spreadsheet programs various sorting and filtering options can be applied to the exported data, e.g. "Table with headers".

	A	B	C	D	E	F	G	H
1	Adress	Name	ConfigType	ProductionCode	RevisionNr	Vendor	ProductionData	2013-10-01-12:00:52.679
2	1001	Terminal 1 (EK1100)	EK1100	72100946	1114112	Beckhoff Automation GmbH	FW: 06, HW: 22, THU, 32/2012	
3	1002	Terminal 2 (EL2032)	EL2032	133181522	1048576	Beckhoff Automation GmbH	FW: 09, HW: 15, TUE, 23/2012	
4	1003	Terminal 3 (EL1024)	EL1024	67121234	1048576	Beckhoff Automation GmbH	FW: 00, HW: 04, MON, 19/2012	
5	1004	Terminal 4 (EL1252-0050)	EL1252-0050	82063442	1048626	Beckhoff Automation GmbH	FW: 00, HW: 00, TUE, 25/2012	
6	1005	Terminal 5 (EL1252)	EL1252	82063442	1376256	Beckhoff Automation GmbH	FW: 01, HW: 06, WED, 09/2013	
7	1006	Terminal 6 (EL1252)	EL1252	82063442	1114112	Beckhoff Automation GmbH	No Productiondata available	
8	1007	Terminal 7 (EL1819)	EL1819	119222354	1114112	Beckhoff Automation GmbH	FW: 00, HW: 03, THU, 50/2012	

Fig. 152: Filtering in the spreadsheet program

## 3.6 TwinCAT Development Environment

### 3.6.1 Installation of the TwinCAT real-time driver

In order to assign real-time capability to a standard Ethernet port of an IPC controller, the Beckhoff real-time driver has to be installed on this port under Windows.

This can be done in several ways. One option is described here.

In the System Manager call up the TwinCAT overview of the local network interfaces via Options → Show Real Time Ethernet Compatible Devices.

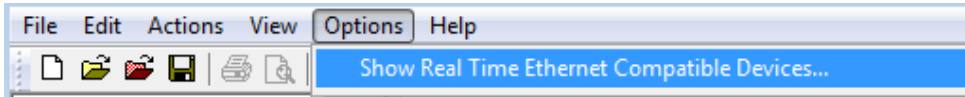


Fig. 153: System Manager “Options” (TwinCAT 2)

This have to be called up by the Menü “TwinCAT” within the TwinCAT 3 environment:

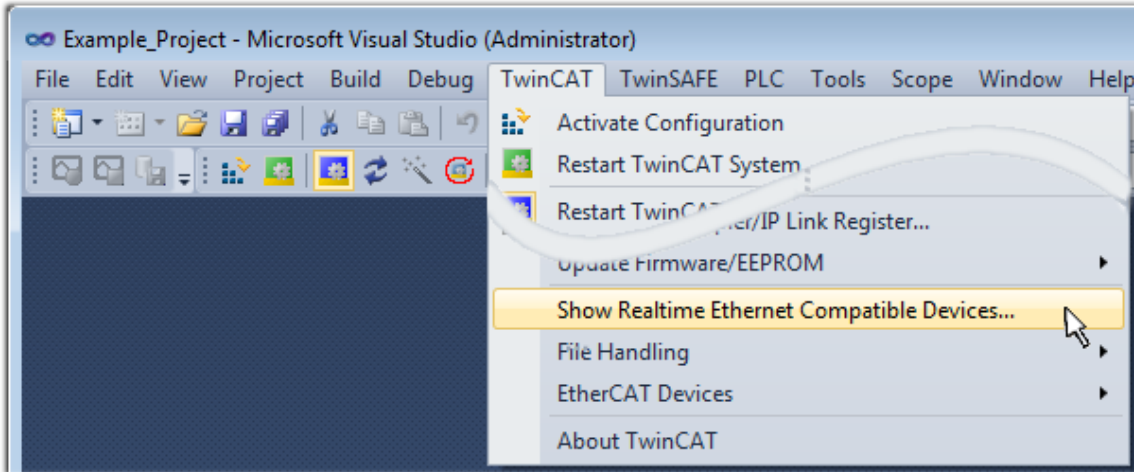


Fig. 154: Call up under VS Shell (TwinCAT 3)

The following dialog appears:

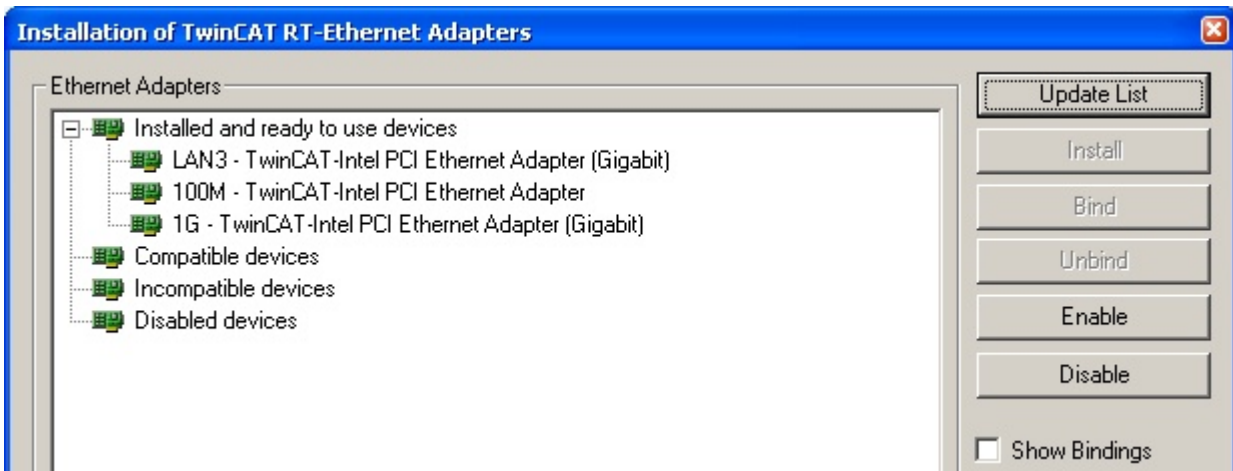


Fig. 155: Overview of network interfaces

Interfaces listed under “Compatible devices” can be assigned a driver via the “Install” button. A driver should only be installed on compatible devices.

A Windows warning regarding the unsigned driver can be ignored.

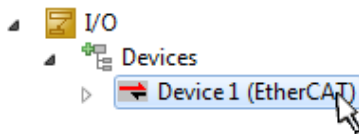
**Alternatively** an EtherCAT-device can be inserted first of all as described in chapter [Offline configuration creation, section “Creating the EtherCAT device” \[► 122\]](#) in order to view the compatible ethernet ports via its EtherCAT properties (tab “Adapter”, button “Compatible Devices...”):





Fig. 156: EtherCAT device properties(TwinCAT 2): click on “Compatible Devices...” of tab “Adapte””

TwinCAT 3: the properties of the EtherCAT device can be opened by double click on “Device .. (EtherCAT)” within the Solution Explorer under “I/O”:



After the installation the driver appears activated in the Windows overview for the network interface (Windows Start → System Properties → Network)

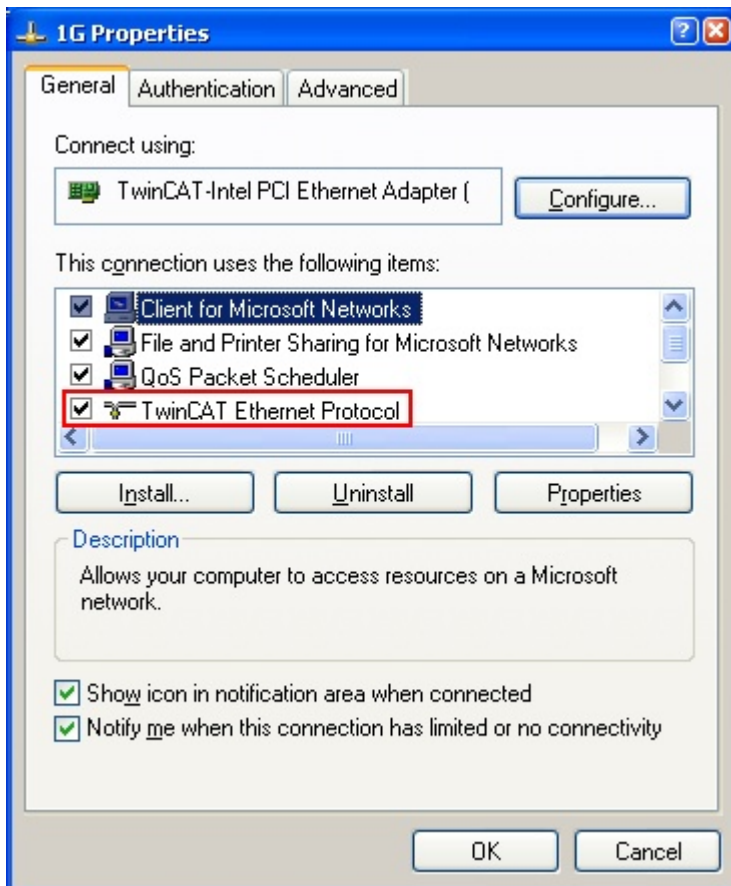


Fig. 157: Windows properties of the network interface

A correct setting of the driver could be:

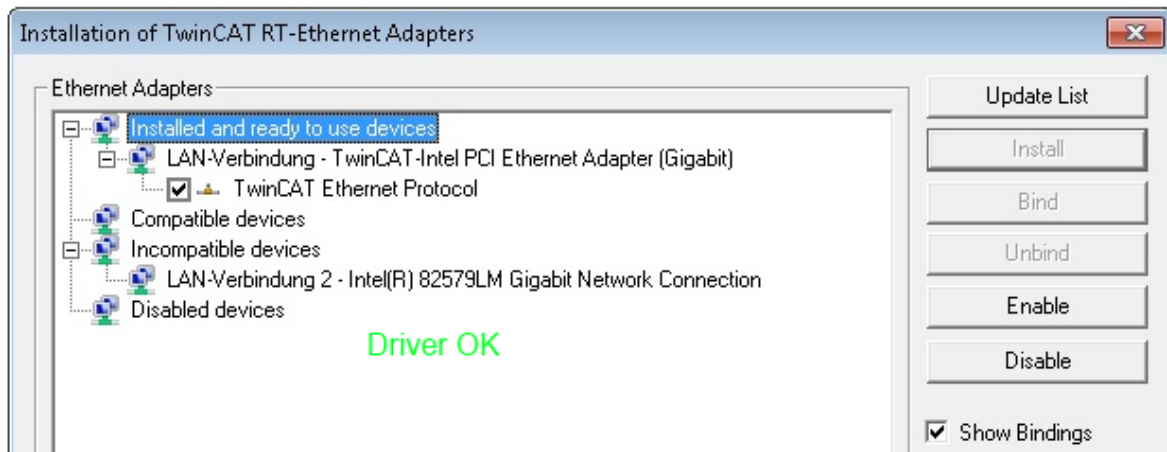


Fig. 158: Exemplary correct driver setting for the Ethernet port

Other possible settings have to be avoided:

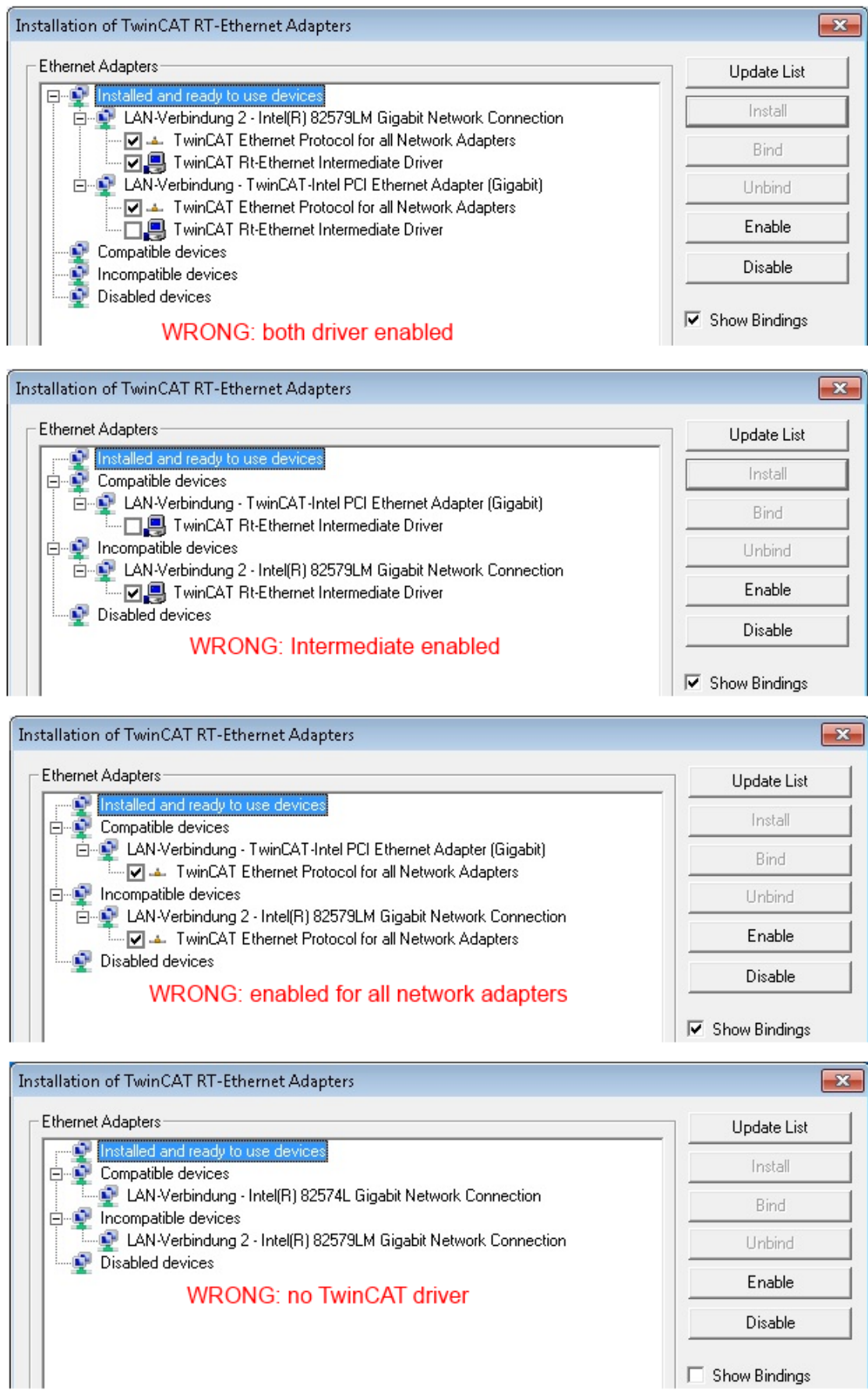


Fig. 159: Incorrect driver settings for the Ethernet port

## IP address of the port used

### **i** IP address/DHCP

In most cases an Ethernet port that is configured as an EtherCAT device will not transport general IP packets. For this reason and in cases where an EL6601 or similar devices are used it is useful to specify a fixed IP address for this port via the “Internet Protocol TCP/IP” driver setting and to disable DHCP. In this way the delay associated with the DHCP client for the Ethernet port assigning itself a default IP address in the absence of a DHCP server is avoided. A suitable address space is 192.168.x.x, for example.

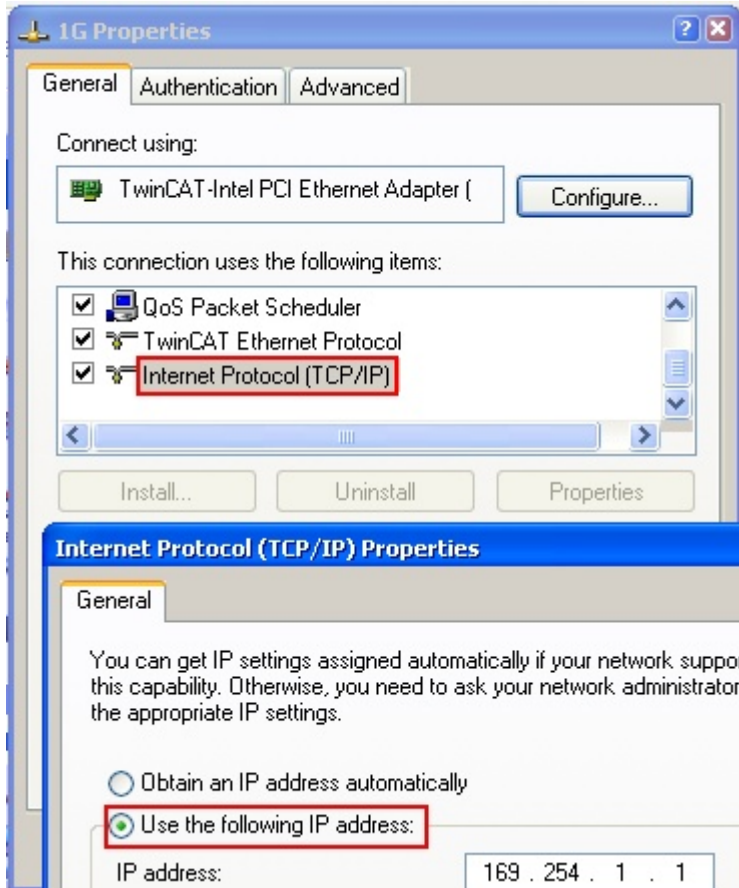


Fig. 160: TCP/IP setting for the Ethernet port

## 3.6.2 Notes regarding ESI device description

### Installation of the latest ESI device description

The TwinCAT EtherCAT master/System Manager needs the device description files for the devices to be used in order to generate the configuration in online or offline mode. The device descriptions are contained in the so-called ESI files (EtherCAT Slave Information) in XML format. These files can be requested from the respective manufacturer and are made available for download. An \*.xml file may contain several device descriptions.

The ESI files for Beckhoff EtherCAT devices are available on the [Beckhoff website](#).

The ESI files should be stored in the TwinCAT installation directory.

Default settings:

- **TwinCAT 2:** C:\TwinCAT\IO\EtherCAT
- **TwinCAT 3:** C:\TwinCAT\3.1\Config\Io\EtherCAT

The files are read (once) when a new System Manager window is opened, if they have changed since the last time the System Manager window was opened.

A TwinCAT installation includes the set of Beckhoff ESI files that was current at the time when the TwinCAT build was created.

For TwinCAT 2.11/TwinCAT 3 and higher, the ESI directory can be updated from the System Manager, if the programming PC is connected to the Internet; by

- **TwinCAT 2:** Option → “Update EtherCAT Device Descriptions”
- **TwinCAT 3:** TwinCAT → EtherCAT Devices → “Update Device Descriptions (via ETG Website)...”

The [TwinCAT ESI Updater \[► 121\]](#) is available for this purpose.



### ESI

The \*.xml files are associated with \*.xsd files, which describe the structure of the ESI XML files. To update the ESI device descriptions, both file types should therefore be updated.

### Device differentiation

EtherCAT devices/slaves are distinguished by four properties, which determine the full device identifier. For example, the device identifier EL2521-0025-1018 consists of:

- family key “EL”
- name “2521”
- type “0025”
- and revision “1018”

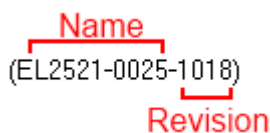


Fig. 161: Identifier structure

The order identifier consisting of name + type (here: EL2521-0010) describes the device function. The revision indicates the technical progress and is managed by Beckhoff. In principle, a device with a higher revision can replace a device with a lower revision, unless specified otherwise, e.g. in the documentation. Each revision has its own ESI description. See [further notes \[► 102\]](#).

## Online description

If the EtherCAT configuration is created online through scanning of real devices (see section Online setup) and no ESI descriptions are available for a slave (specified by name and revision) that was found, the System Manager asks whether the description stored in the device should be used. In any case, the System Manager needs this information for setting up the cyclic and acyclic communication with the slave correctly.

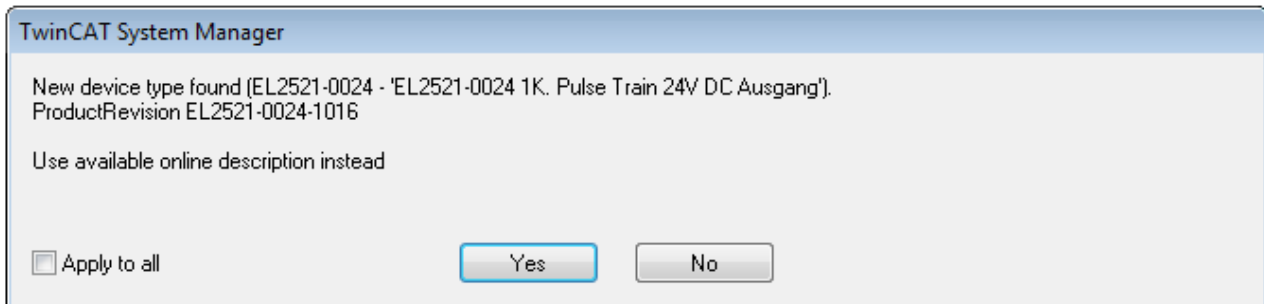


Fig. 162: OnlineDescription information window (TwinCAT 2)

In TwinCAT 3 a similar window appears, which also offers the Web update:

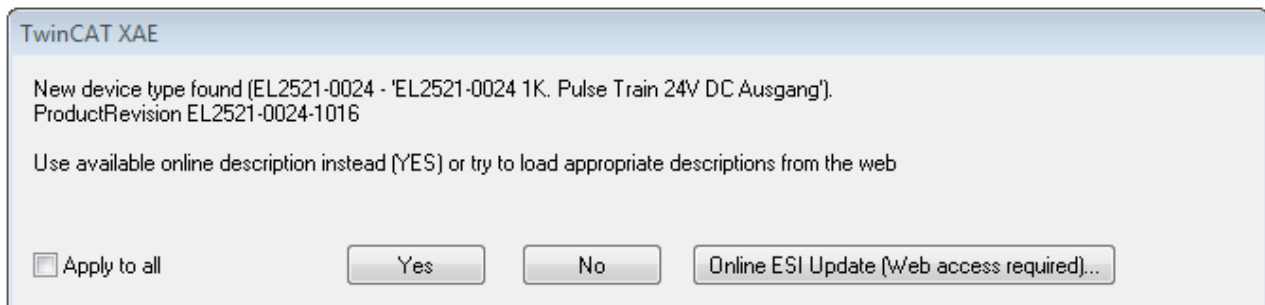


Fig. 163: Information window OnlineDescription (TwinCAT 3)

If possible, the Yes is to be rejected and the required ESI is to be requested from the device manufacturer. After installation of the XML/XSD file the configuration process should be repeated.

### NOTE

#### Changing the “usual” configuration through a scan

- ✓ If a scan discovers a device that is not yet known to TwinCAT, distinction has to be made between two cases. Taking the example here of the EL2521-0000 in the revision 1019
  - a) no ESI is present for the EL2521-0000 device at all, either for the revision 1019 or for an older revision. The ESI must then be requested from the manufacturer (in this case Beckhoff).
  - b) an ESI is present for the EL2521-0000 device, but only in an older revision, e.g. 1018 or 1017. In this case an in-house check should first be performed to determine whether the spare parts stock allows the integration of the increased revision into the configuration at all. A new/higher revision usually also brings along new features. If these are not to be used, work can continue without reservations with the previous revision 1018 in the configuration. This is also stated by the Beckhoff compatibility rule.

Refer in particular to the chapter “[General notes on the use of Beckhoff EtherCAT IO components](#)” and for manual configuration to the chapter “[Offline configuration creation \[► 122\]](#)”.

If the OnlineDescription is used regardless, the System Manager reads a copy of the device description from the EEPROM in the EtherCAT slave. In complex slaves the size of the EEPROM may not be sufficient for the complete ESI, in which case the ESI would be *incomplete* in the configurator. Therefore it's recommended using an offline ESI file with priority in such a case.

The System Manager creates for online recorded device descriptions a new file “OnlineDescription0000...xml” in its ESI directory, which contains all ESI descriptions that were read online.

OnlineDescriptionCache00000002.xml

Fig. 164: File OnlineDescription.xml created by the System Manager

If a slave desired to be added manually to the configuration at a later stage, online created slaves are indicated by a prepended symbol ">" in the selection list (see Figure *Indication of an online recorded ESI of EL2521 as an example*).

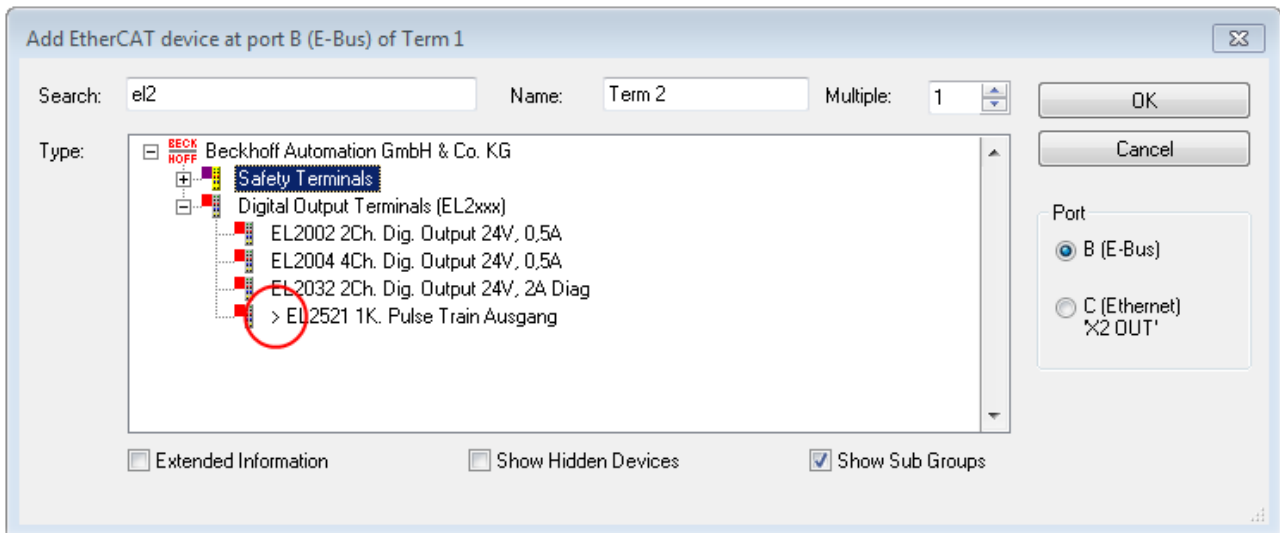


Fig. 165: Indication of an online recorded ESI of EL2521 as an example

If such ESI files are used and the manufacturer's files become available later, the file OnlineDescription.xml should be deleted as follows:

- close all System Manager windows
- restart TwinCAT in Config mode
- delete "OnlineDescription0000...xml"
- restart TwinCAT System Manager

This file should not be visible after this procedure, if necessary press <F5> to update

**i OnlineDescription for TwinCAT 3.x**

In addition to the file described above "OnlineDescription0000...xml", a so called EtherCAT cache with new discovered devices is created by TwinCAT 3.x, e.g. under Windows 7:

```
C:\User\[USERNAME]\AppData\Roaming\Beckhoff\TwinCAT3\Components\Base\EtherCATCache.xml
```

(Please note the language settings of the OS!)

You have to delete this file, too.

**Faulty ESI file**

If an ESI file is faulty and the System Manager is unable to read it, the System Manager brings up an information window.

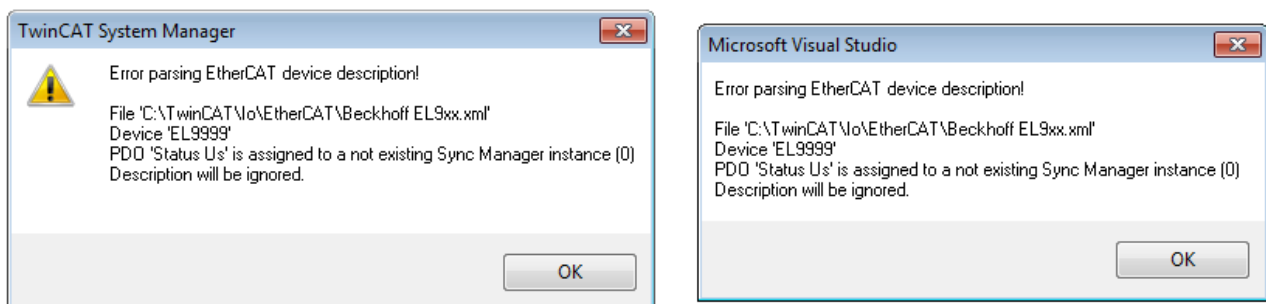


Fig. 166: Information window for faulty ESI file (left: TwinCAT 2; right: TwinCAT 3)

Reasons may include:

- Structure of the \*.xml does not correspond to the associated \*.xsd file → check your schematics
- Contents cannot be translated into a device description → contact the file manufacturer



### 3.6.3 TwinCAT ESI Updater

For TwinCAT 2.11 and higher, the System Manager can search for current Beckhoff ESI files automatically, if an online connection is available:

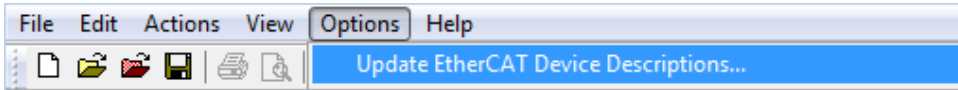


Fig. 167: Using the ESI Updater (>= TwinCAT 2.11)

The call up takes place under:  
 “Options” → “Update EtherCAT Device Descriptions”

Selection under TwinCAT 3:

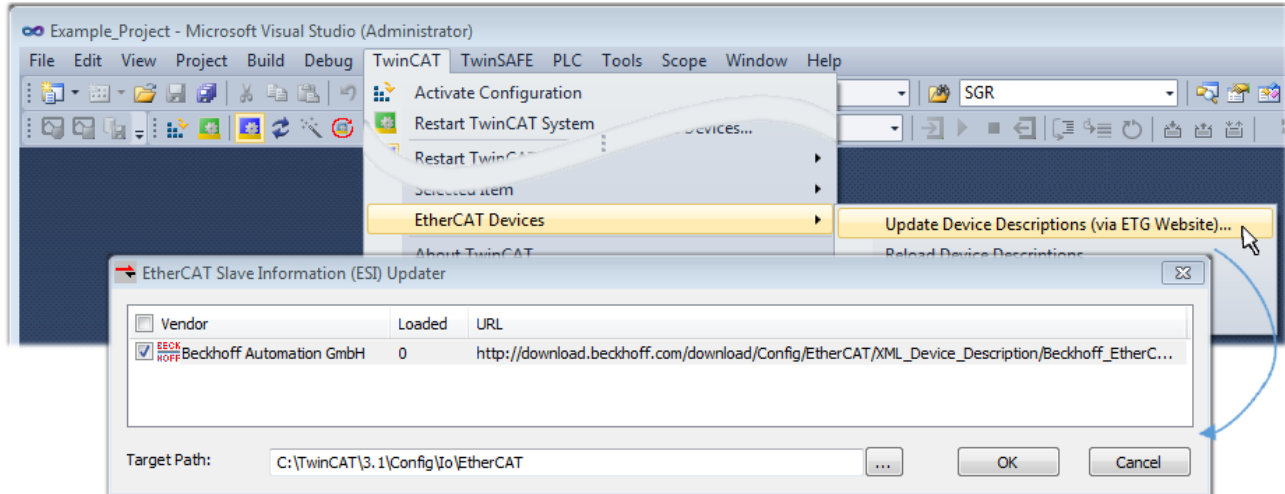


Fig. 168: Using the ESI Updater (TwinCAT 3)

The ESI Updater (TwinCAT 3) is a convenient option for automatic downloading of ESI data provided by EtherCAT manufacturers via the Internet into the TwinCAT directory (ESI = EtherCAT slave information). TwinCAT accesses the central ESI ULR directory list stored at ETG; the entries can then be viewed in the Updater dialog, although they cannot be changed there.

The call up takes place under:  
 “TwinCAT” → “EtherCAT Devices” → “Update Device Description (via ETG Website)...”.

### 3.6.4 Distinction between Online and Offline

The distinction between online and offline refers to the presence of the actual I/O environment (drives, terminals, EJ-modules). If the configuration is to be prepared in advance of the system configuration as a programming system, e.g. on a laptop, this is only possible in “Offline configuration” mode. In this case all components have to be entered manually in the configuration, e.g. based on the electrical design.

If the designed control system is already connected to the EtherCAT system and all components are energised and the infrastructure is ready for operation, the TwinCAT configuration can simply be generated through “scanning” from the runtime system. This is referred to as online configuration.

In any case, during each startup the EtherCAT master checks whether the slaves it finds match the configuration. This test can be parameterised in the extended slave settings. Refer to note “Installation of the latest ESI-XML device description” [▶ 117].

**For preparation of a configuration:**

- the real EtherCAT hardware (devices, couplers, drives) must be present and installed
- the devices/modules must be connected via EtherCAT cables or in the terminal/ module strand in the same way as they are intended to be used later

- the devices/modules be connected to the power supply and ready for communication
- TwinCAT must be in CONFIG mode on the target system.

#### The online scan process consists of:

- [detecting the EtherCAT device \[▶ 127\]](#) (Ethernet port at the IPC)
- [detecting the connected EtherCAT devices \[▶ 128\]](#). This step can be carried out independent of the preceding step
- [troubleshooting \[▶ 131\]](#)

The [scan with existing configuration \[▶ 132\]](#) can also be carried out for comparison.

## 3.6.5 OFFLINE configuration creation

### Creating the EtherCAT device

Create an EtherCAT device in an empty System Manager window.

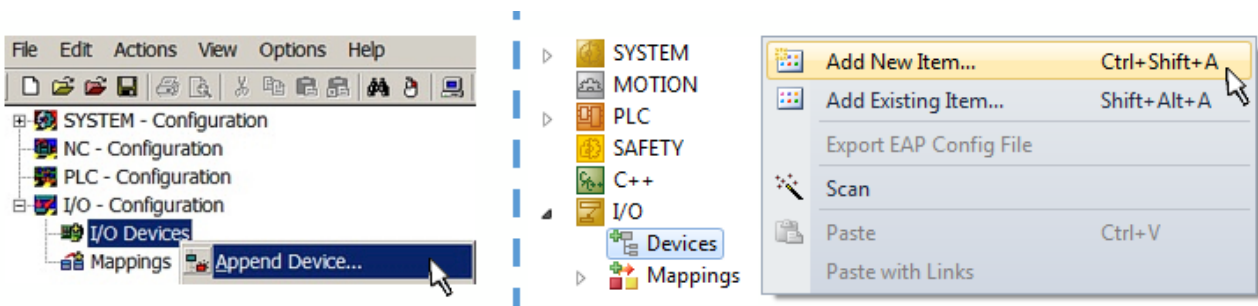


Fig. 169: Append EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

Select type “EtherCAT” for an EtherCAT I/O application with EtherCAT slaves. For the present publisher/ subscriber service in combination with an EL6601/EL6614 terminal select “EtherCAT Automation Protocol via EL6601”.

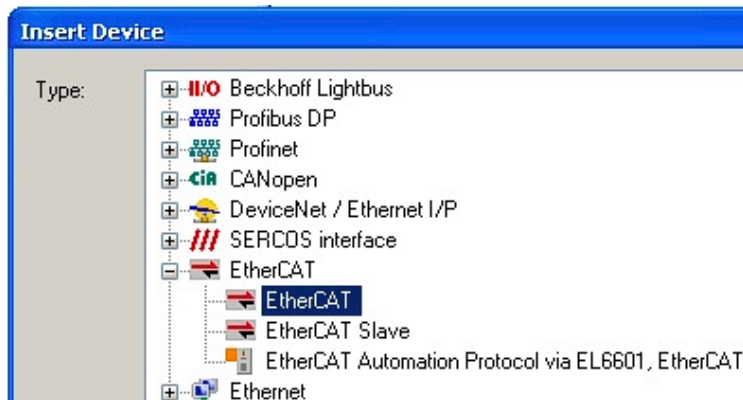


Fig. 170: Selecting the EtherCAT connection (TwinCAT 2.11, TwinCAT 3)

Then assign a real Ethernet port to this virtual device in the runtime system.

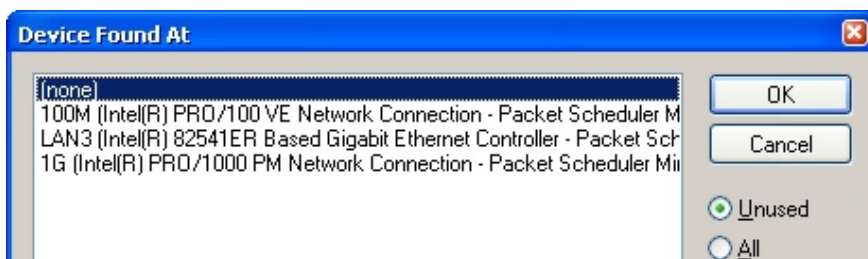


Fig. 171: Selecting the Ethernet port

This query may appear automatically when the EtherCAT device is created, or the assignment can be set/modified later in the properties dialog; see Fig. “EtherCAT device properties (TwinCAT 2)”.

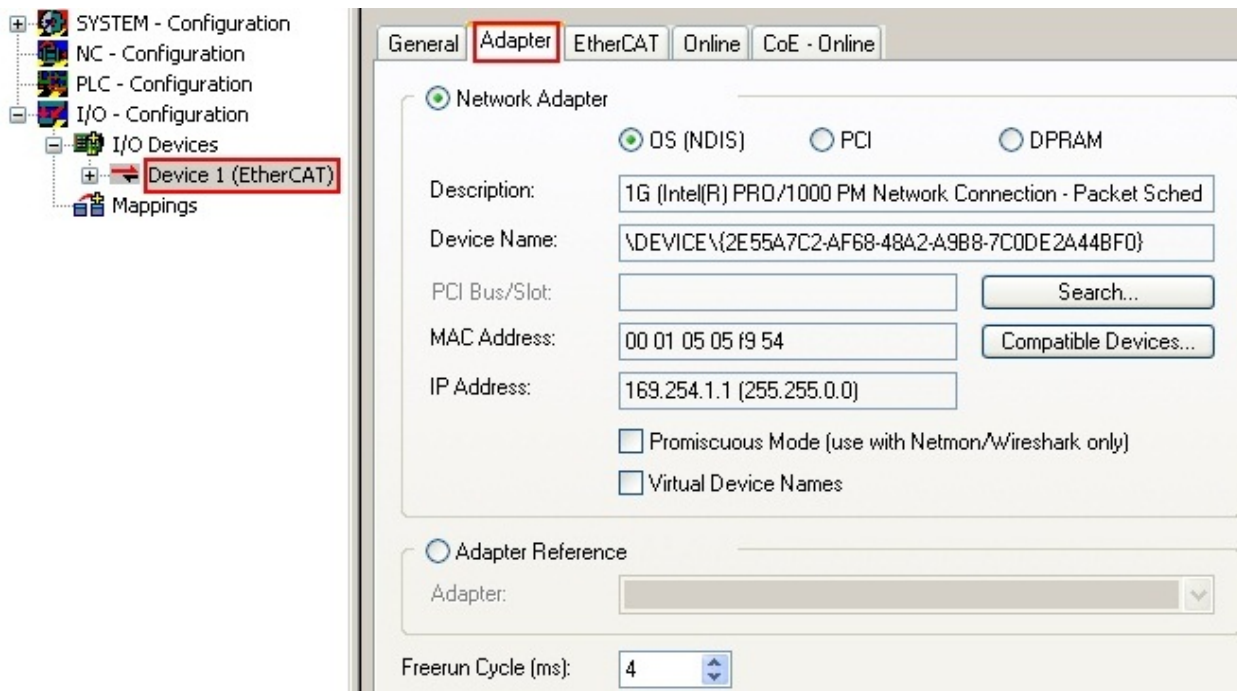
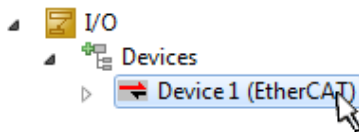


Fig. 172: EtherCAT device properties (TwinCAT 2)

TwinCAT 3: the properties of the EtherCAT device can be opened by double click on “Device .. (EtherCAT)” within the Solution Explorer under “I/O”:



**i** **Selecting the Ethernet port**

Ethernet ports can only be selected for EtherCAT devices for which the TwinCAT real-time driver is installed. This has to be done separately for each port. Please refer to the respective [installation page \[p. 111\]](#).

**Defining EtherCAT slaves**

Further devices can be appended by right-clicking on a device in the configuration tree.

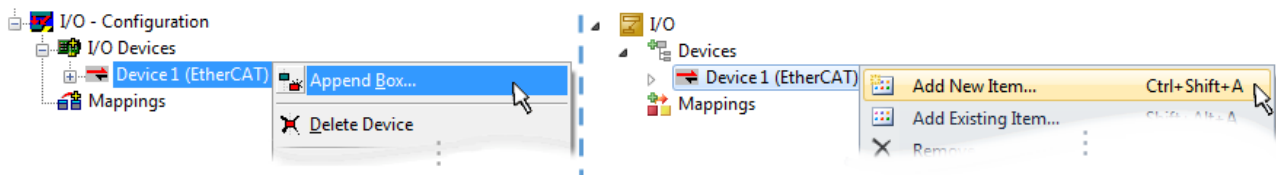


Fig. 173: Appending EtherCAT devices (left: TwinCAT 2; right: TwinCAT 3)

The dialog for selecting a new device opens. Only devices for which ESI files are available are displayed.

Only devices are offered for selection that can be appended to the previously selected device. Therefore the physical layer available for this port is also displayed (Fig. “Selection dialog for new EtherCAT device”, A). In the case of cable-based Fast-Ethernet physical layer with PHY transfer, then also only cable-based devices are available, as shown in Fig. “Selection dialog for new EtherCAT device”. If the preceding device has several free ports (e.g. EK1122 or EK1100), the required port can be selected on the right-hand side (A).

Overview of physical layer

- “Ethernet”: cable-based 100BASE-TX: EK couplers, EP boxes, devices with RJ45/M8/M12 connector

- “E-Bus”: LVDS “terminal bus”, “EJ-module”: EL/ES terminals, various modular modules

The search field facilitates finding specific devices (since TwinCAT 2.11 or TwinCAT 3).

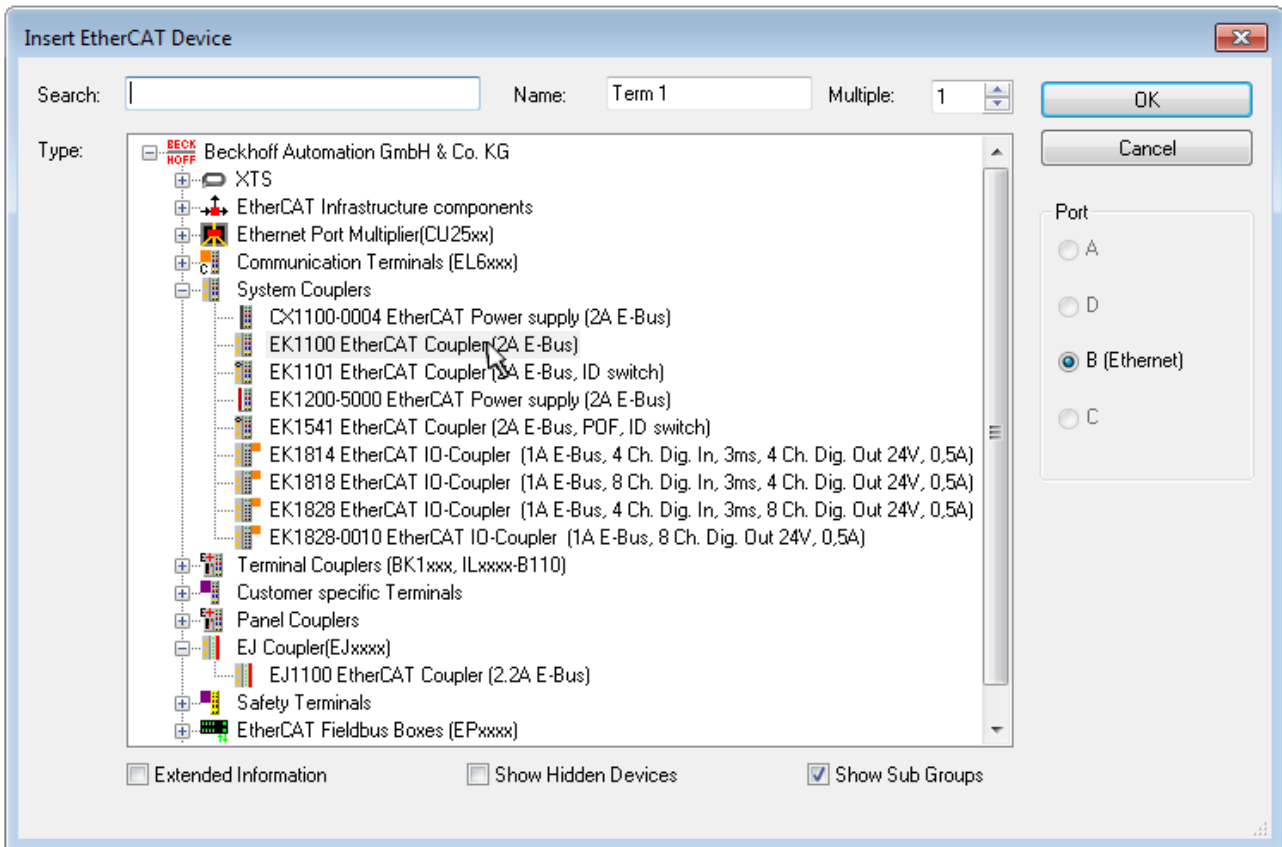


Fig. 174: Selection dialog for new EtherCAT device

By default only the name/device type is used as selection criterion. For selecting a specific revision of the device the revision can be displayed as “Extended Information”.

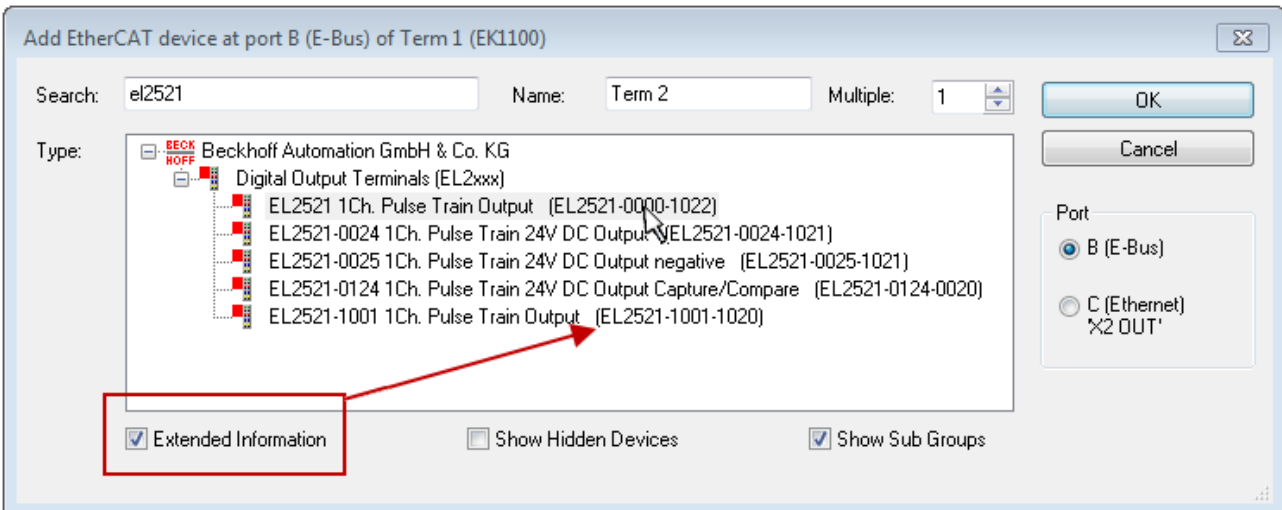


Fig. 175: Display of device revision

In many cases several device revisions were created for historic or functional reasons, e.g. through technological advancement. For simplification purposes (see Fig. “Selection dialog for new EtherCAT device”) only the last (i.e. highest) revision and therefore the latest state of production is displayed in the selection dialog for Beckhoff devices. To show all device revisions available in the system as ESI descriptions tick the “Show Hidden Devices” check box, see Fig. “Display of previous revisions”.

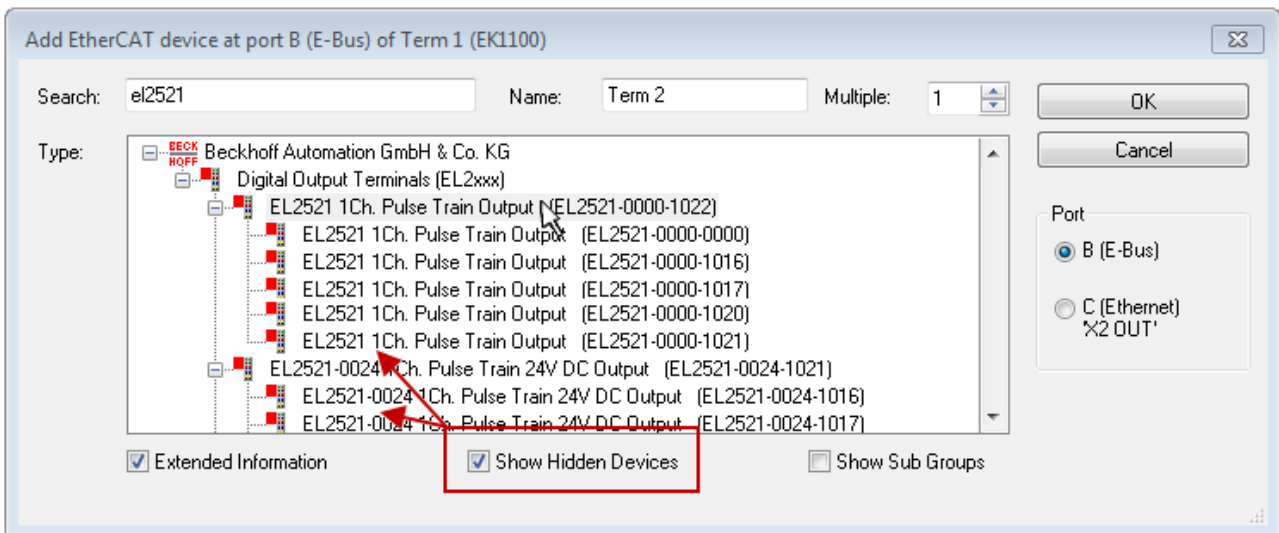


Fig. 176: Display of previous revisions

**i Device selection based on revision, compatibility**

The ESI description also defines the process image, the communication type between master and slave/device and the device functions, if applicable. The physical device (firmware, if available) has to support the communication queries/settings of the master. This is backward compatible, i.e. newer devices (higher revision) should be supported if the EtherCAT master addresses them as an older revision. The following compatibility rule of thumb is to be assumed for Beckhoff EtherCAT Terminals/ Boxes/ EJ-modules:

**device revision in the system >= device revision in the configuration**

This also enables subsequent replacement of devices without changing the configuration (different specifications are possible for drives).

**Example**

If an EL2521-0025-1018 is specified in the configuration, an EL2521-0025-1018 or higher (-1019, -1020) can be used in practice.

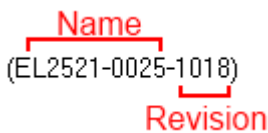


Fig. 177: Name/revision of the terminal

If current ESI descriptions are available in the TwinCAT system, the last revision offered in the selection dialog matches the Beckhoff state of production. It is recommended to use the last device revision when creating a new configuration, if current Beckhoff devices are used in the real application. Older revisions should only be used if older devices from stock are to be used in the application.

In this case the process image of the device is shown in the configuration tree and can be parameterized as follows: linking with the task, CoE/DC settings, plug-in definition, startup settings, ...

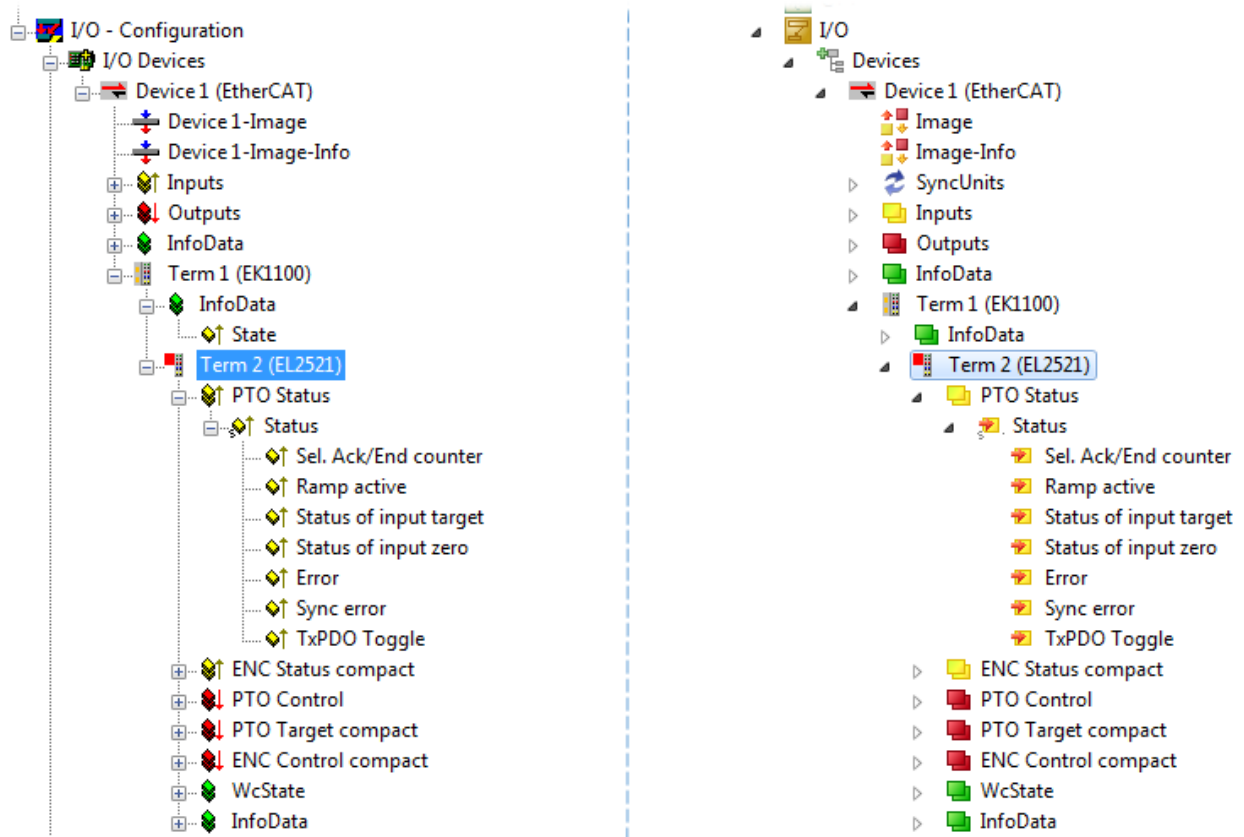




Fig. 178: EtherCAT terminal in the TwinCAT tree (left: TwinCAT 2; right: TwinCAT 3)



### 3.6.6 ONLINE configuration creation

#### Detecting/scanning of the EtherCAT device

The online device search can be used if the TwinCAT system is in CONFIG mode. This can be indicated by a symbol right below in the information bar:



- on TwinCAT 2 by a blue display “Config Mode” within the System Manager window:  .
- on TwinCAT 3 within the user interface of the development environment by a symbol  .

TwinCAT can be set into this mode:

- TwinCAT 2: by selection of  in the Menubar or by “Actions” → “Set/Reset TwinCAT to Config Mode...”
- TwinCAT 3: by selection of  in the Menubar or by “TwinCAT” → “Restart TwinCAT (Config Mode)”

#### ● Online scanning in Config mode

**i** The online search is not available in RUN mode (production operation). Note the differentiation between TwinCAT programming system and TwinCAT target system.

The TwinCAT 2 icon () or TwinCAT 3 icon () within the Windows-Taskbar always shows the TwinCAT mode of the local IPC. Compared to that, the System Manager window of TwinCAT 2 or the user interface of TwinCAT 3 indicates the state of the target system.

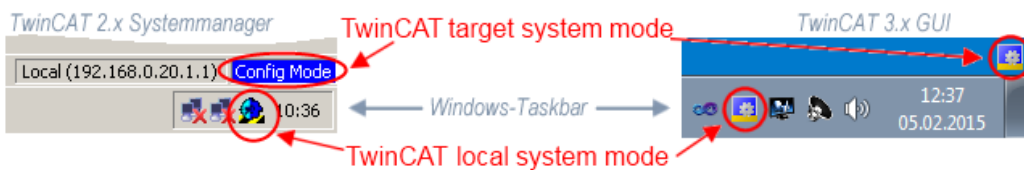


Fig. 179: Differentiation local/target system (left: TwinCAT 2; right: TwinCAT 3)

Right-clicking on “I/O Devices” in the configuration tree opens the search dialog.

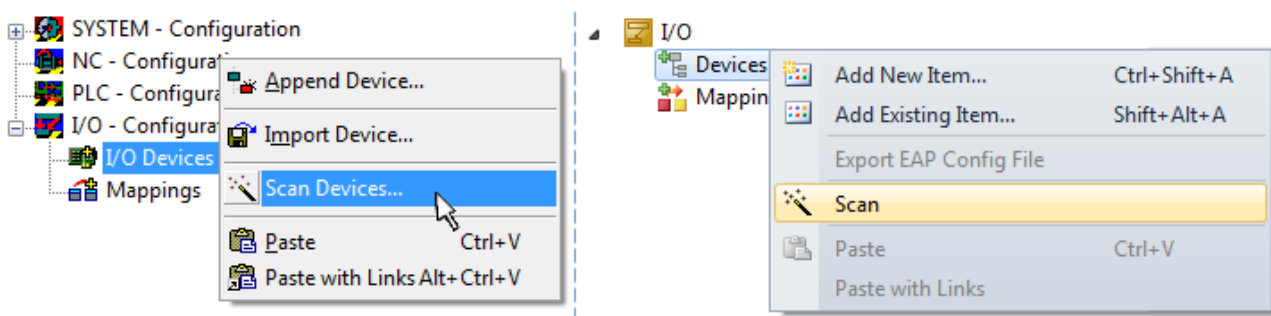


Fig. 180: Scan Devices (left: TwinCAT 2; right: TwinCAT 3)

This scan mode attempts to find not only EtherCAT devices (or Ethernet ports that are usable as such), but also NOVRAM, fieldbus cards, SMB etc. However, not all devices can be found automatically.

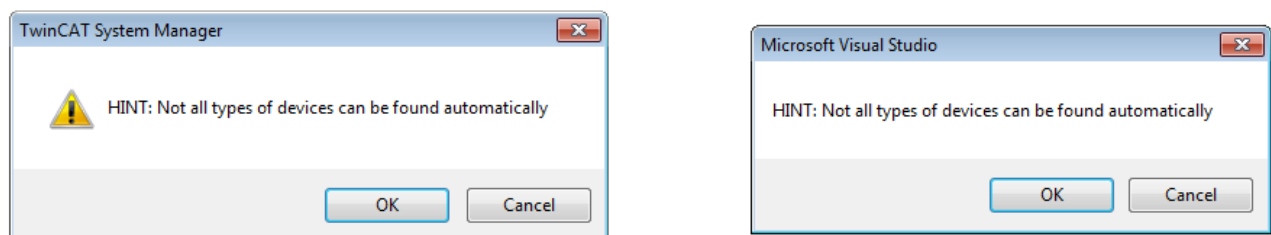


Fig. 181: Note for automatic device scan (left: TwinCAT 2; right: TwinCAT 3)

Ethernet ports with installed TwinCAT real-time driver are shown as “RT Ethernet” devices. An EtherCAT frame is sent to these ports for testing purposes. If the scan agent detects from the response that an EtherCAT slave is connected, the port is immediately shown as an “EtherCAT Device” .

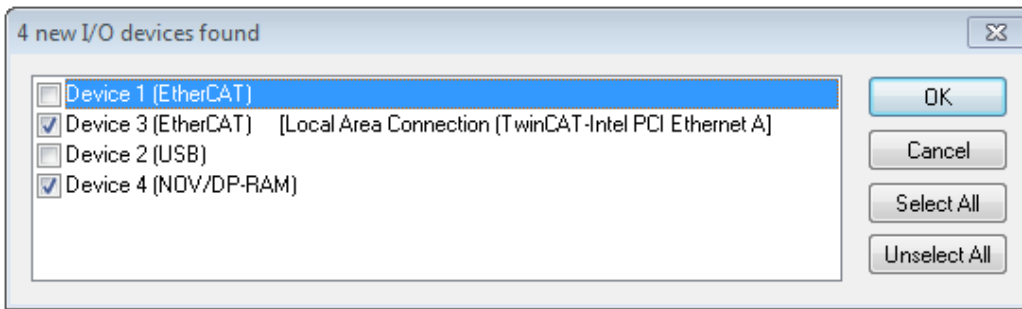


Fig. 182: Detected Ethernet devices

Via respective checkboxes devices can be selected (as illustrated in Fig. “Detected Ethernet devices” e.g. Device 3 and Device 4 were chosen). After confirmation with “OK” a device scan is suggested for all selected devices, see Fig.: “Scan query after automatic creation of an EtherCAT device”.

### ● Selecting the Ethernet port



Ethernet ports can only be selected for EtherCAT devices for which the TwinCAT real-time driver is installed. This has to be done separately for each port. Please refer to the respective [installation page](#) [▶ 111].

## Detecting/Scanning the EtherCAT devices

### ● Online scan functionality



During a scan the master queries the identity information of the EtherCAT slaves from the slave EEPROM. The name and revision are used for determining the type. The respective devices are located in the stored ESI data and integrated in the configuration tree in the default state defined there.

**Name**  
(EL2521-0025-1018)  
**Revision**

Fig. 183: Example default state

## NOTE

### Slave scanning in practice in series machine production

The scanning function should be used with care. It is a practical and fast tool for creating an initial configuration as a basis for commissioning. In series machine production or reproduction of the plant, however, the function should no longer be used for the creation of the configuration, but if necessary for [comparison](#) [▶ 132] with the defined initial configuration. Background: since Beckhoff occasionally increases the revision version of the delivered products for product maintenance reasons, a configuration can be created by such a scan which (with an identical machine construction) is identical according to the device list; however, the respective device revision may differ from the initial configuration.

### Example:

Company A builds the prototype of a machine B, which is to be produced in series later on. To do this the prototype is built, a scan of the IO devices is performed in TwinCAT and the initial configuration “B.tsm” is created. The EL2521-0025 EtherCAT terminal with the revision 1018 is located somewhere. It is thus built into the TwinCAT configuration in this way:



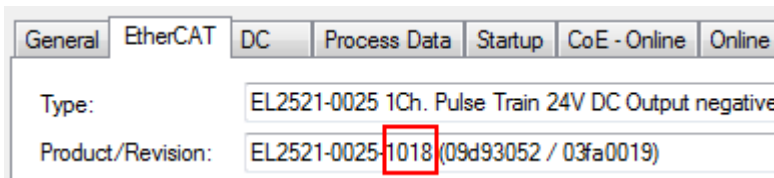


Fig. 184: Installing EthetCAT terminal with revision -1018

Likewise, during the prototype test phase, the functions and properties of this terminal are tested by the programmers/commissioning engineers and used if necessary, i.e. addressed from the PLC “B.pro” or the NC. (the same applies correspondingly to the TwinCAT 3 solution files).

The prototype development is now completed and series production of machine B starts, for which Beckhoff continues to supply the EL2521-0025-0018. If the commissioning engineers of the series machine production department always carry out a scan, a B configuration with the identical contents results again for each machine. Likewise, A might create spare parts stores worldwide for the coming series-produced machines with EL2521-0025-1018 terminals.

After some time Beckhoff extends the EL2521-0025 by a new feature C. Therefore the FW is changed, outwardly recognizable by a higher FW version and a **new revision -1019**. Nevertheless the new device naturally supports functions and interfaces of the predecessor version(s); an adaptation of “B.tsm” or even “B.pro” is therefore unnecessary. The series-produced machines can continue to be built with “B.tsm” and “B.pro”; it makes sense to perform a comparative scan [► 132] against the initial configuration “B.tsm” in order to check the built machine.

However, if the series machine production department now doesn't use “B.tsm”, but instead carries out a scan to create the productive configuration, the revision **-1019** is automatically detected and built into the configuration:

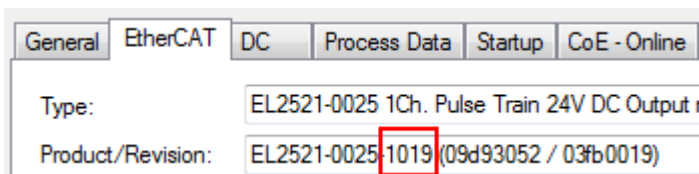


Fig. 185: Detection of EtherCAT terminal with revision -1019

This is usually not noticed by the commissioning engineers. TwinCAT cannot signal anything either, since virtually a new configuration is created. According to the compatibility rule, however, this means that no EL2521-0025-**1018** should be built into this machine as a spare part (even if this nevertheless works in the vast majority of cases).

In addition, it could be the case that, due to the development accompanying production in company A, the new feature C of the EL2521-0025-1019 (for example, an improved analog filter or an additional process data for the diagnosis) is discovered and used without in-house consultation. The previous stock of spare part devices are then no longer to be used for the new configuration “B2.tsm” created in this way. If series machine production is established, the scan should only be performed for informative purposes for comparison with a defined initial configuration. Changes are to be made with care!

If an EtherCAT device was created in the configuration (manually or through a scan), the I/O field can be scanned for devices/slaves.



Fig. 186: Scan query after automatic creation of an EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

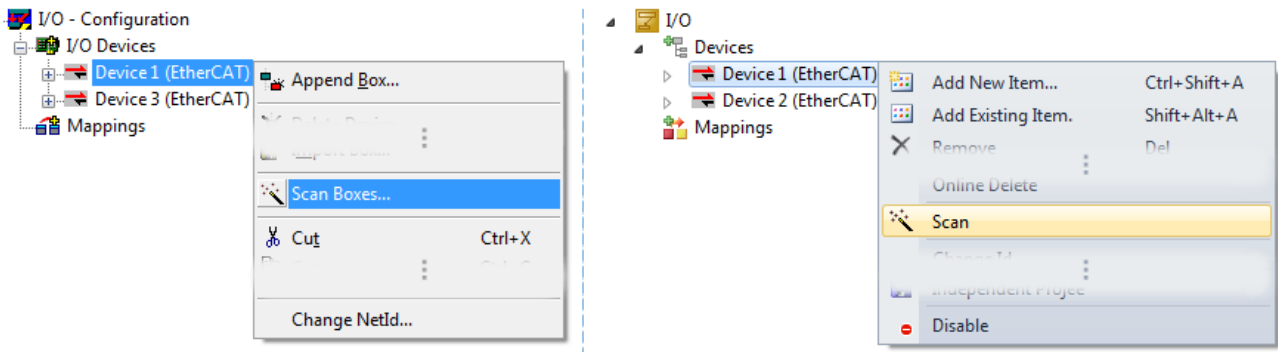


Fig. 187: Manual triggering of a device scan on a specified EtherCAT device (left: TwinCAT 2; right: TwinCAT 3)

In the System Manager (TwinCAT 2) or the User Interface (TwinCAT 3) the scan process can be monitored via the progress bar at the bottom in the status bar.



Fig. 188: Scan progress example by TwinCAT 2

The configuration is established and can then be switched to online state (OPERATIONAL).



Fig. 189: Config/FreeRun query (left: TwinCAT 2; right: TwinCAT 3)

In Config/FreeRun mode the System Manager display alternates between blue and red, and the EtherCAT device continues to operate with the idling cycle time of 4 ms (default setting), even without active task (NC, PLC).



Fig. 190: Displaying of “Free Run” and “Config Mode” toggling right below in the status bar



Fig. 191: TwinCAT can also be switched to this state by using a button (left: TwinCAT 2; right: TwinCAT 3)

The EtherCAT system should then be in a functional cyclic state, as shown in Fig. *Online display example*.

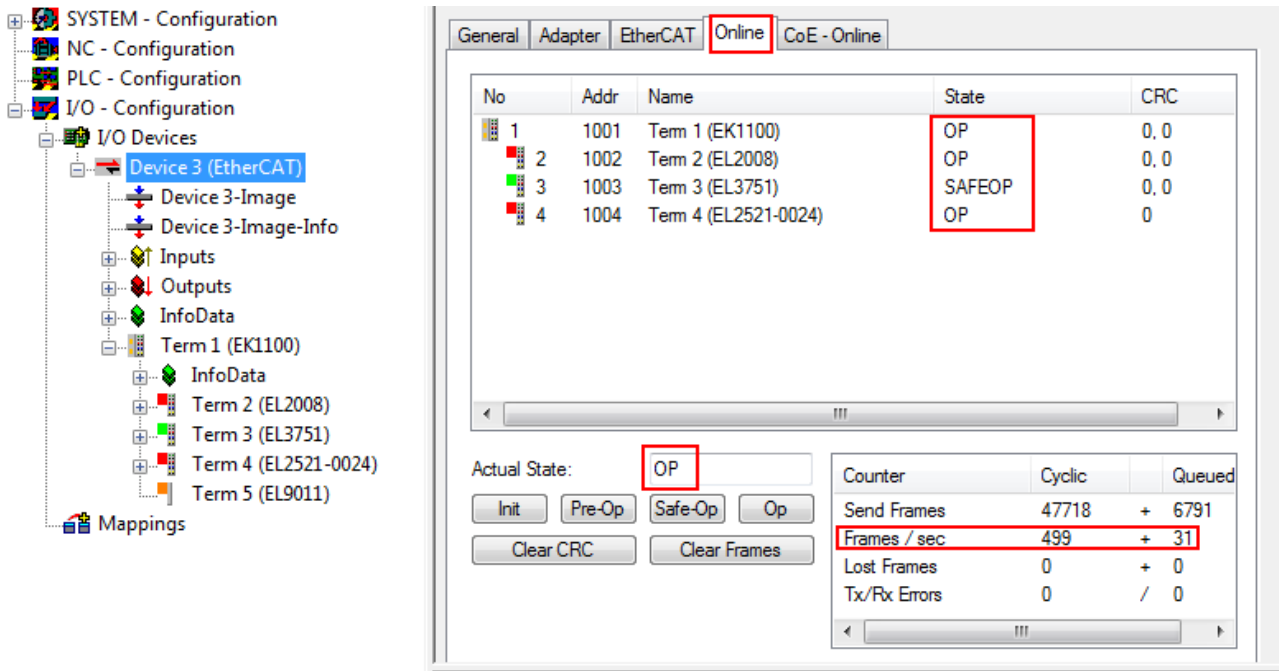


Fig. 192: Online display example

Please note:

- all slaves should be in OP state
- the EtherCAT master should be in “Actual State” OP
- “frames/sec” should match the cycle time taking into account the sent number of frames
- no excessive “LostFrames” or CRC errors should occur

The configuration is now complete. It can be modified as described under [manual procedure \[► 122\]](#).

### Troubleshooting

Various effects may occur during scanning.

- An **unknown device** is detected, i.e. an EtherCAT slave for which no ESI XML description is available. In this case the System Manager offers to read any ESI that may be stored in the device. This case is described in the chapter “Notes regarding ESI device description”.

- **Device are not detected properly**

Possible reasons include:

- faulty data links, resulting in data loss during the scan
- slave has invalid device description

The connections and devices should be checked in a targeted manner, e.g. via the emergency scan.

Then re-run the scan.

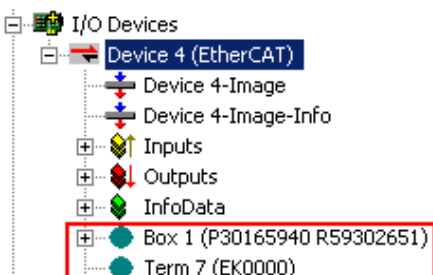


Fig. 193: Faulty identification

In the System Manager such devices may be set up as EK0000 or unknown devices. Operation is not possible or meaningful.

## Scan over existing Configuration

**NOTE****Change of the configuration after comparison**

With this scan (TwinCAT 2.11 or 3.1) only the device properties vendor (manufacturer), device name and revision are compared at present! A “ChangeTo” or “Copy” should only be carried out with care, taking into consideration the Beckhoff IO compatibility rule (see above). The device configuration is then replaced by the revision found; this can affect the supported process data and functions.

If a scan is initiated for an existing configuration, the actual I/O environment may match the configuration exactly or it may differ. This enables the configuration to be compared.



Fig. 194: Identical configuration (left: TwinCAT 2; right: TwinCAT 3)

If differences are detected, they are shown in the correction dialog, so that the user can modify the configuration as required.

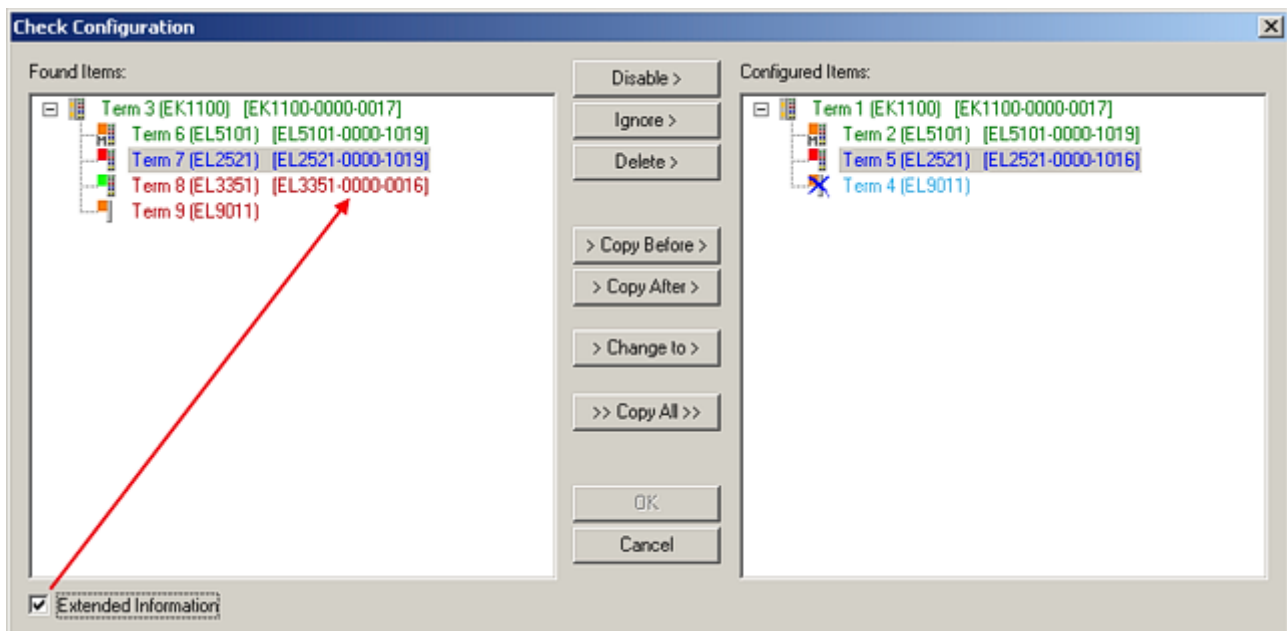


Fig. 195: Correction dialog

It is advisable to tick the “Extended Information” check box to reveal differences in the revision.

Color	Explanation
green	This EtherCAT slave matches the entry on the other side. Both type and revision match.
blue	This EtherCAT slave is present on the other side, but in a different revision. This other revision can have other default values for the process data as well as other/additional functions. If the found revision is higher than the configured revision, the slave may be used provided compatibility issues are taken into account.  If the found revision is lower than the configured revision, it is likely that the slave cannot be used. The found device may not support all functions that the master expects based on the higher revision number.
light blue	This EtherCAT slave is ignored ("Ignore" button)
red	<ul style="list-style-type: none"> <li>This EtherCAT slave is not present on the other side.</li> <li>It is present, but in a different revision, which also differs in its properties from the one specified. The compatibility principle then also applies here: if the found revision is higher than the configured revision, use is possible provided compatibility issues are taken into account, since the successor devices should support the functions of the predecessor devices. If the found revision is lower than the configured revision, it is likely that the slave cannot be used. The found device may not support all functions that the master expects based on the higher revision number.</li> </ul>

**i Device selection based on revision, compatibility**

The ESI description also defines the process image, the communication type between master and slave/device and the device functions, if applicable. The physical device (firmware, if available) has to support the communication queries/settings of the master. This is backward compatible, i.e. newer devices (higher revision) should be supported if the EtherCAT master addresses them as an older revision. The following compatibility rule of thumb is to be assumed for Beckhoff EtherCAT Terminals/ Boxes/ EJ-modules:

**device revision in the system >= device revision in the configuration**

This also enables subsequent replacement of devices without changing the configuration (different specifications are possible for drives).

**Example**

If an EL2521-0025-1018 is specified in the configuration, an EL2521-0025-1018 or higher (-1019, -1020) can be used in practice.

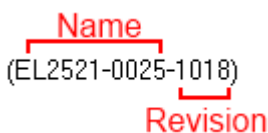


Fig. 196: Name/revision of the terminal

If current ESI descriptions are available in the TwinCAT system, the last revision offered in the selection dialog matches the Beckhoff state of production. It is recommended to use the last device revision when creating a new configuration, if current Beckhoff devices are used in the real application. Older revisions should only be used if older devices from stock are to be used in the application.

In this case the process image of the device is shown in the configuration tree and can be parameterized as follows: linking with the task, CoE/DC settings, plug-in definition, startup settings, ...

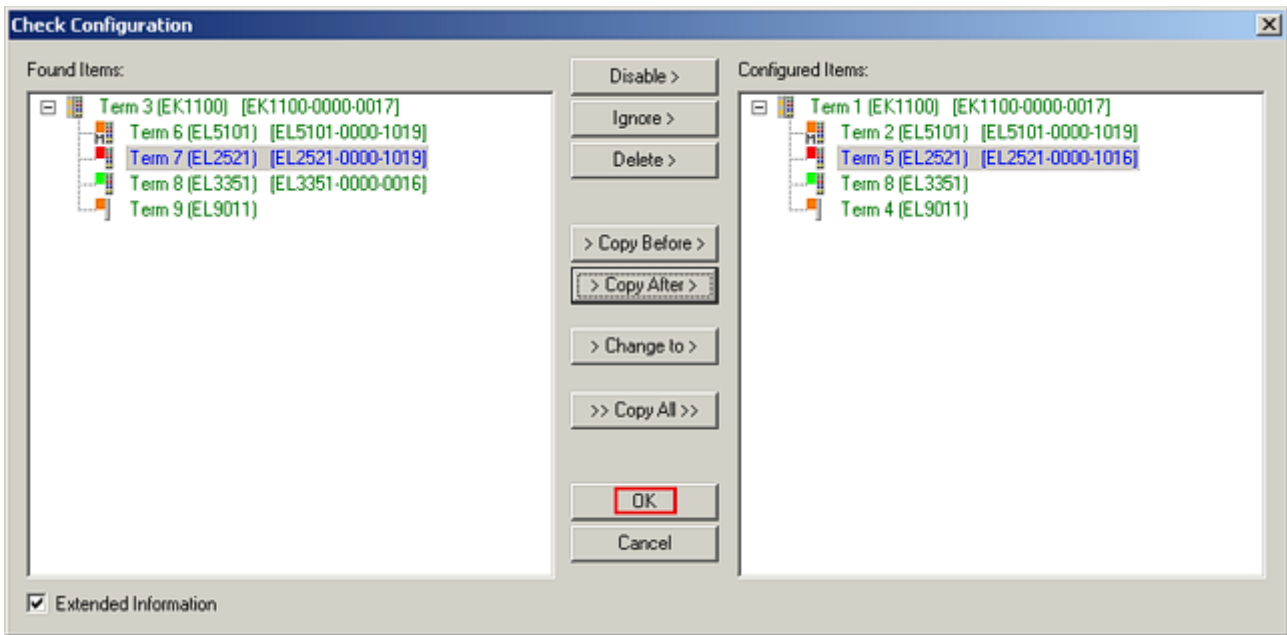


Fig. 197: Correction dialog with modifications

Once all modifications have been saved or accepted, click “OK” to transfer them to the real \*.tsm configuration.

### Change to Compatible Type

TwinCAT offers a function *Change to Compatible Type...* for the exchange of a device whilst retaining the links in the task.

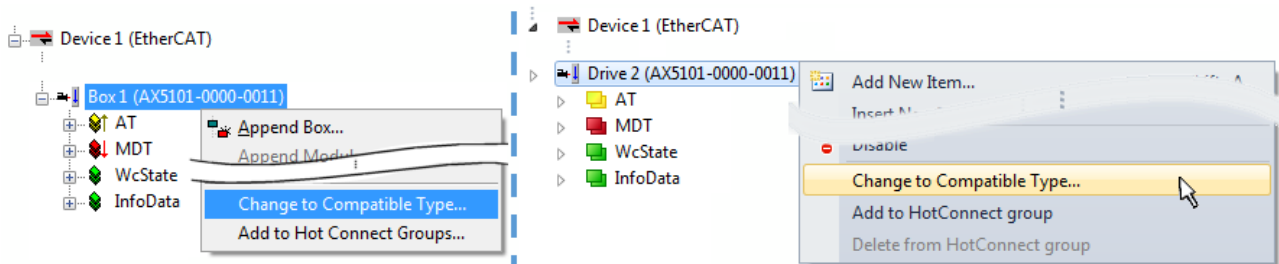


Fig. 198: Dialog “Change to Compatible Type...” (left: TwinCAT 2; right: TwinCAT 3)

This function is preferably to be used on AX5000 devices.

### Change to Alternative Type

The TwinCAT System Manager offers a function for the exchange of a device: Change to Alternative Type

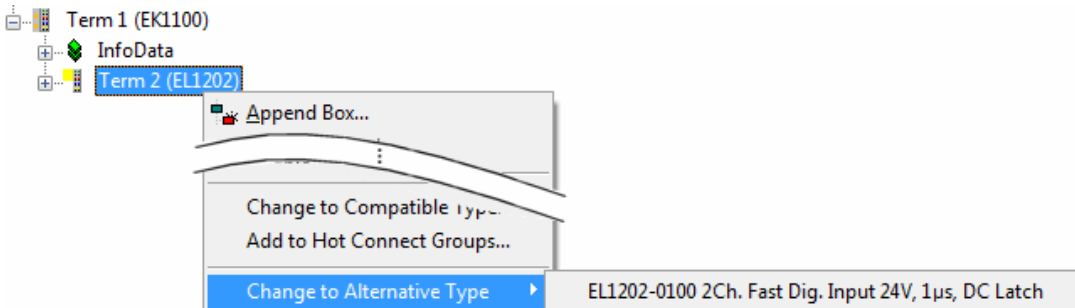


Fig. 199: TwinCAT 2 Dialog Change to Alternative Type

If called, the System Manager searches in the procured device ESI (in this example: EL1202-0000) for details of compatible devices contained there. The configuration is changed and the ESI-EEPROM is overwritten at the same time – therefore this process is possible only in the online state (ConfigMode).


## **3.6.7 Standard behavior of the EtherCAT master**

### **3.6.7.1 General**

For simple and fast commissioning TwinCAT supports the user with suitable default settings and automatisms. In most cases these settings are adequate for stable system operation. The settings, their effects and setting options are described below to facilitate customer specified handling or special behavior. The following topics will be discussed:

- process data information
- EtherCAT master settings
- slave settings
- Sync Task
- distributed clock settings
- EoE

Check the following elements for regular and proper operation of the EtherCAT system:

Element	Control options - online/commissioning engineer	Control options – application
TwinCAT on target system in RUN state (or CONFIG/FREERUN)	TwinCAT icon on target system (not programming system!) is green or blue System Manager info is green or blue/ flashes red (FreeRun) 	Application checks TwinCAT state via ADS
EtherCAT master in OP state	see Fig. <i>Online diagnostics for EtherCAT device, A</i>	EcMasterState query via ADS (PLC: block from TcEtherCAT.lib) ADS NetId of the EcMaster known from the device info data (see Fig. <i>Online diagnostics for EtherCAT device</i> )
All EtherCAT slaves in OP state	see Fig. <i>Online diagnostics for EtherCAT device, B</i>	EcSlavesState query via ADS (PLC: block from TcEtherCAT.lib)
Cyclic telegrams are sent based on cycle time	see Fig. <i>Online diagnostics for EtherCAT device, C</i>	Query via ADS
Acyclic telegrams are sent occasionally	see Fig. <i>Online diagnostics for EtherCAT device, D</i>	Query via ADS
none or few LostFrames/CRC in the slaves	see Fig. <i>Online diagnostics for EtherCAT device, D</i>	Query via ADS
EtherCAT DevState = 0	see Fig. <i>Online diagnostics for EtherCAT device, E</i>	Link to monitoring task
All slaves WorkingCounter = 0 throughout	see Fig. <i>Online diagnostics for EtherCAT device, F</i>	Link to monitoring task or Frm0WcState collect information from EcMaster inputs
No conspicuous outputs in the logger window		- (if diagnostics are OK, causes for logger outputs are determined by other means)
No cycle time exceedings		PLC: integration of TcUtilities.lib, thus access to SystemInfo and SystemTaskInfoArr[] if logged in
No E-bus current violation	see Fig. <i>Online diagnostics for EtherCAT device, G</i>	-
Various watchdogs adhered to (terminal standard 100 ms, FSoE with confirmation 100 ms)		determined through monitoring of states



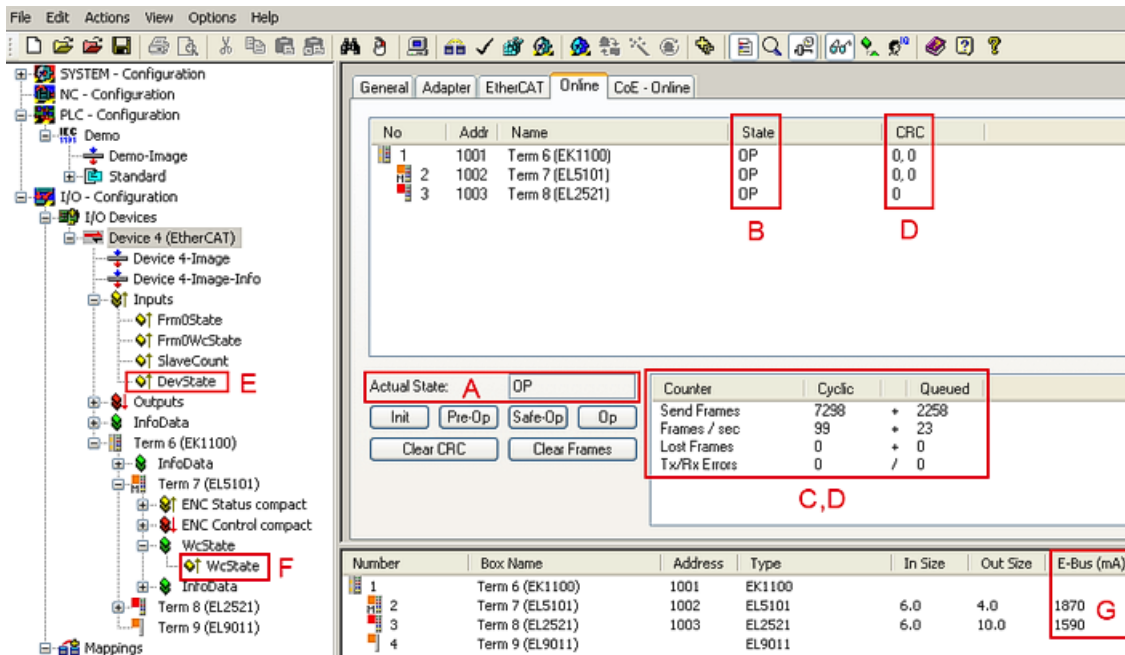


Fig. 200: online diagnostics for EtherCAT device

The standard automatism in the System Manager establish this state as soon as the configuration is activated and TwinCAT is switched to RUN/CONFIG mode.

### 3.6.7.2 Default settings and information

#### Process data information

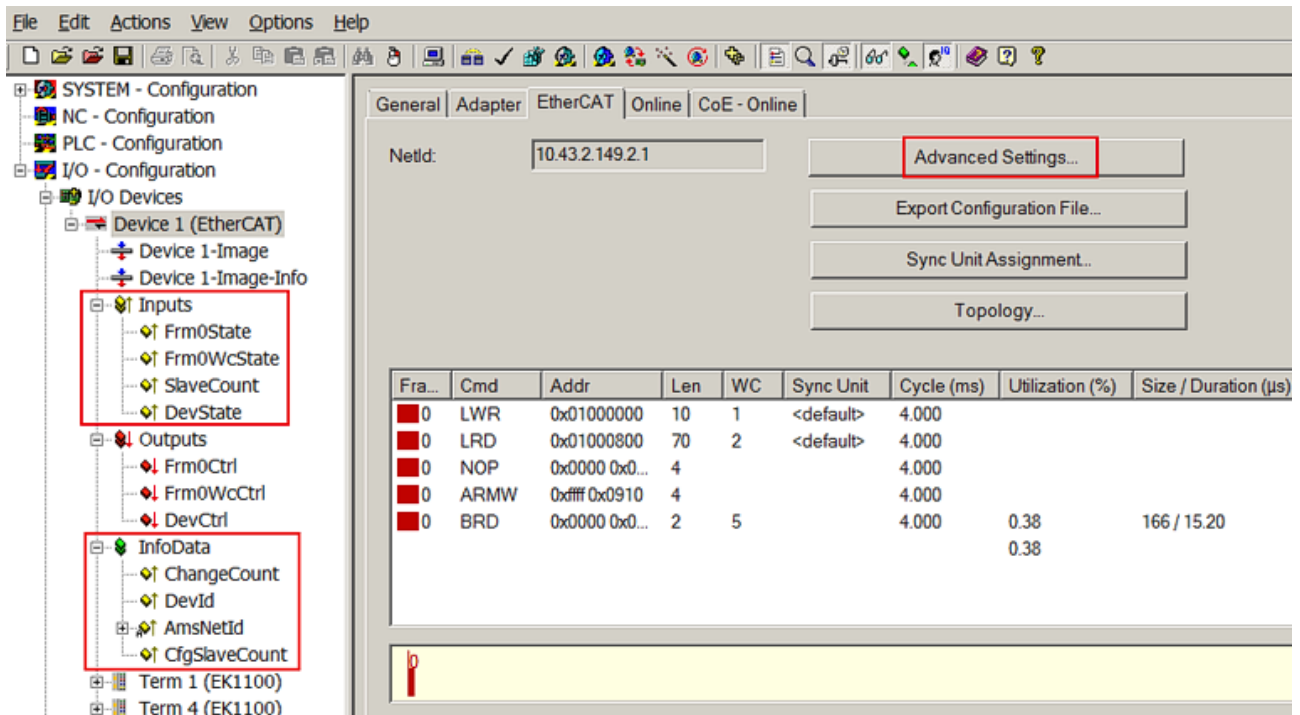


Fig. 201: Process data

The EtherCAT master features info data which provide up-to-date cycle diagnostics (yellow variables) and general information outside the real-time context (green variables). The main parameters are described below:

- **DevState:** target = 0; all slaves are then in OP state, with no link errors etc.

- **Frm0WcState**: should also be 0. Such a variable (Frm0WcState, Frm1WcState, ...) is created for each cyclic Ethernet frame  
As a minimum, an application should check and monitor these two master inputs during each cycle.
- **AmsNetId**: the application (PLC, external task) requires this AMS address for addressing the EtherCAT master or the subordinate slaves via ADS.

Further options are available via Advanced Settings:

### Master settings

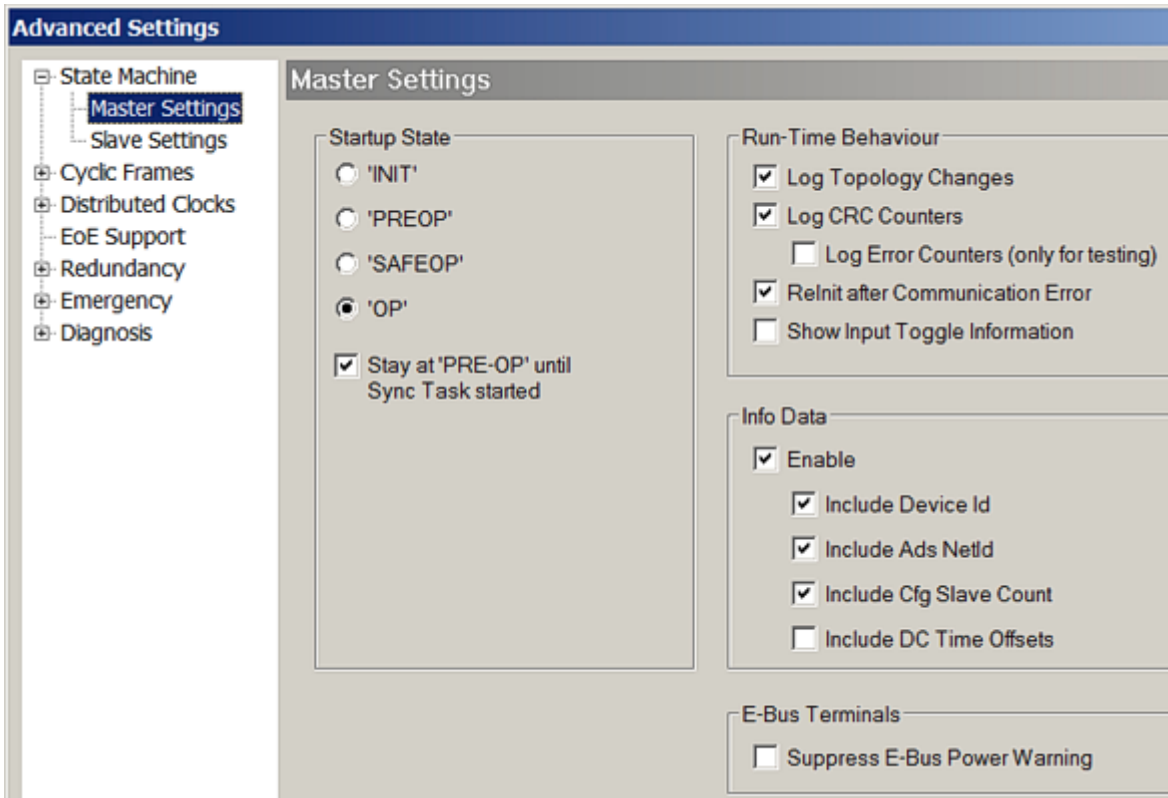
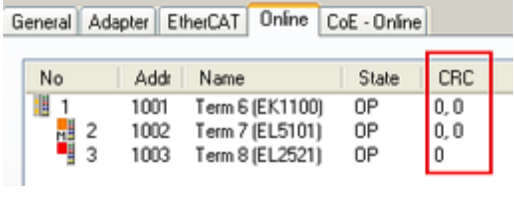
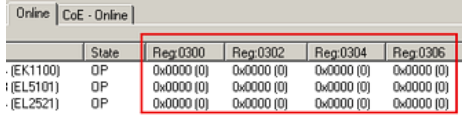
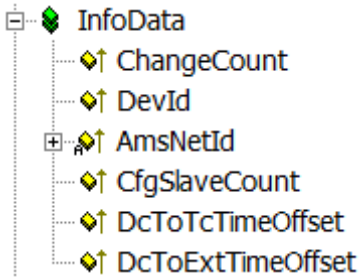


Fig. 202: Master settings

Element	Detail	Explanation	Effects
StartUp State		<p>As soon as TwinCAT is “started” (RUN or Config/FreeRun), the master assumes the state selected here. The transition to OP only takes place once the Sync Task has started.</p> <p>If several tasks run on a system, the Sync Task which includes the distributed control has the highest priority. DC-capable slaves cannot be switched to OP if their local clock is not adjusted properly, or they may come out of OP state again if the control does not succeed (“Sync lost”).</p>	<p>It is recommended to set and monitor the EcMaster state from the application (PLC, as available), (FB_EcGet/SetMasterState from TcEtherCAT.lib).</p> <p>This enables the master to assume OP even after serious communication errors.</p>
Run-Time behavior	Log Topology Changes	<p>Activated by default</p> <p>Online outputs in the logger window are activated</p>	Deactivation not sensible
	Log CRC counters	<p>Activated by default. In OnlineView the CRC errors of the slaves are read from the field and accumulated.</p> 	<p>If the CRC register counters in the slaves are to be displayed in the online view, this option is to be deactivated – i.e. it clears the local x0300ff registers once they have been read in order to prevent overflow at xFF.</p> 
	Log error counters	no function	
	Relnit after communication error	<p>After a communication error during which the master has left the OP state (connection disconnected and lost frames for more than 10 cycles, stations switched off) TwinCAT tries to switch the master back to OP state.</p>	<p>If the EcMaster state is controlled from the application it is essential to disable this option, since otherwise both mechanisms may hinder each other.</p> <p>Both access the master via ADS.</p>
	Show input toggle information	If activated, an additional toggle variable is displayed for input terminals, which can be linked. It changes its state 0/1 whenever a new datagram is received.	
Info data		The display of these (green) non-real-time information data in the System Manager tree can be deactivated here.	<p>Deviceld: useful for access from the application</p> <p>AdsNetId: required for access from the application</p> <p>CfgSlaveCount: number of changes in the configuration so far</p> <p>Dc time offsets: the offsets between external, internal and TwinCAT clock (which are constant at runtime) are displayed. Required for external EtherCAT synchronization.</p> 
E-bus power warning		By default the System Manager issues a warning of the maximum load of an EtherCAT coupler (e.g. EK1100) is about to be exceeded.	

## Slave settings

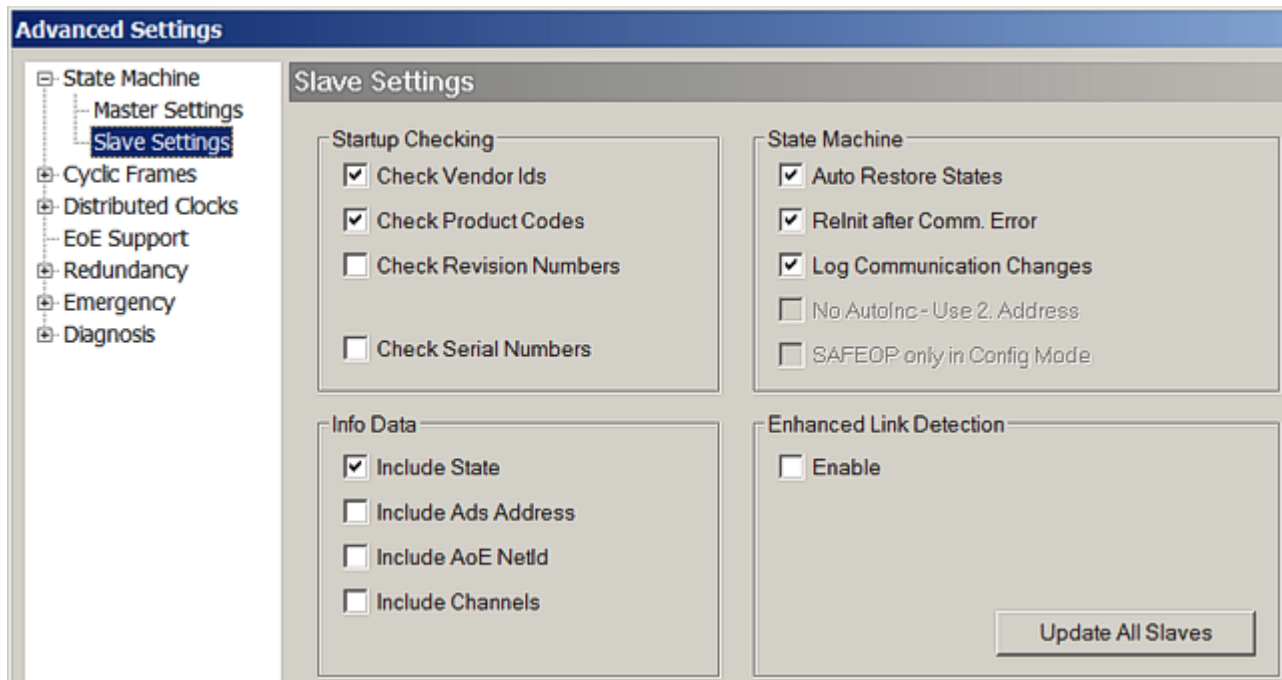
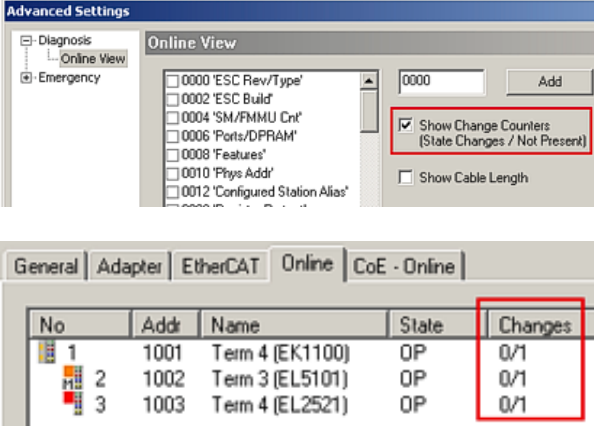


Fig. 203: Slave settings

Element	Detail	Explanation	Effects																				
StartUp checking		During EtherCAT startup the properties of all slaves activated here are checked. However corresponding slave settings have priority!	By default the VendorID and ProductCode (e.g. EI2521-0010) are checked. This is recommended, because it enables devices of the same type but with higher revision number to be used as replacement devices.																				
State machine	Auto restore states	If the slave has left the OP for inherent reasons (loss of power, synchronization error), the EtherCAT master tries to put the slave back to OP state or to the last regularly reached state, if the checkbox is activated.	<p>It is recommended to set and monitor the slave state from the application (PLC, if available, FB_EcGet/SetMasterState from TcEtherCAT.lib).</p> <p>This enables the application to control the slave in accordance with the application-specific requirements.</p> <p>Example (servo axis): the EtherCAT master would switch the axis back to OP state as soon as possible, without deeper knowledge of the safety or functional context. On the contrary, the application can decide whether and when the axis should be switched back to OP state after the serious "Failure State" error.</p> <p>Besides, a DeadLock can occur: If the master resets the slave to OP but only reaches SAFEOP before another fault occurs, the master will then only try to put the slave to SAFEOP. OP state can no longer be reached.</p> <p>It is therefore advisable to control the master and slave state through the application.</p>																				
	Relnit after Comm.Error	If the communication to a slave was interrupted, the master will restart the slave via the INIT state once the connection has been restored, even if the slave had only returned to the SAFEOP state. Thus ensures proper startup and an explicit state of the slave.	If a slave is restarted from INIT --> OP, the outputs are usually disconnected.																				
	Log communication changes	<p>Activated by default. State changes are displayed if the option "Show Change Counter" is activated in online view.</p>  <table border="1" data-bbox="422 1444 1018 1624"> <thead> <tr> <th>No</th> <th>Addr</th> <th>Name</th> <th>State</th> <th>Changes</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1001</td> <td>Term 4 (EK1100)</td> <td>OP</td> <td>0/1</td> </tr> <tr> <td>2</td> <td>1002</td> <td>Term 3 (EL5101)</td> <td>OP</td> <td>0/1</td> </tr> <tr> <td>3</td> <td>1003</td> <td>Term 4 (EL2521)</td> <td>OP</td> <td>0/1</td> </tr> </tbody> </table>	No	Addr	Name	State	Changes	1	1001	Term 4 (EK1100)	OP	0/1	2	1002	Term 3 (EL5101)	OP	0/1	3	1003	Term 4 (EL2521)	OP	0/1	Deactivation not sensible
No	Addr	Name	State	Changes																			
1	1001	Term 4 (EK1100)	OP	0/1																			
2	1002	Term 3 (EL5101)	OP	0/1																			
3	1003	Term 4 (EL2521)	OP	0/1																			
Info data		The display of these (green) non-real-time information data in the System Manager tree can be (de)activated here.	Showing the ADS address in each slave is useful for linking with a slave-specific FUNCTIONBLOCK that will monitor a slave, for example.																				
Enhanced link detection		This function is not intended for the general use.	<p>Using this function on EtherCAT devices that do not support it can lead to substantial and irreversible faults in the EtherCAT communication.</p> <p>In devices that support this function it is already activated through the ESI installed during production.</p>																				

## Sync Task

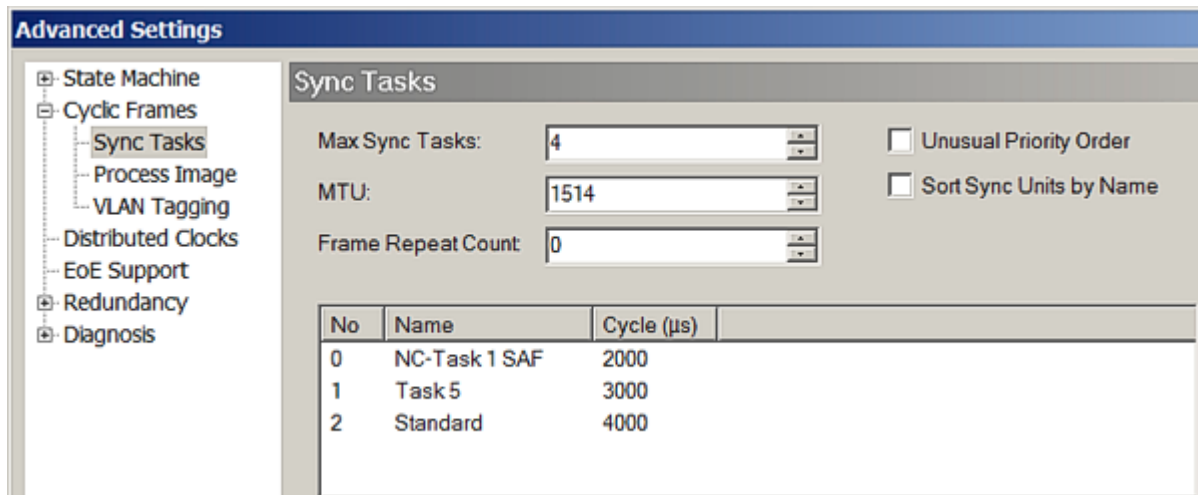


Fig. 204: Sync Task

Element	Explanation
Max Sync Task	<p>TwinCAT 2.10/2.11 supports up to 4 SyncTasks. A SyncTask is a task (PLC, NC) that triggers an I/O update, i.e. with its own EtherCAT frames it communicates with the I/O field using cyclic communication and a fixed cycle time. Fig. <i>Sync Task</i> shows the configuration for 3 active tasks in with cycle times of 2, 3 and 4 ms. Make sure the task prioritization order is correct.</p> <p>If the configuration comprises more than 4 tasks, the tasks with the larger cycle times are set to the slowest task.</p> <p>Hint: This can be useful in cases where a very slow PLC task (&gt; 100 ms) is to be used but the I/O communication has to be faster due to the slave watchdog. In this case the number of "Max Sync Task" should be reduced until only tasks &lt;100 ms remain.</p>
MTU	The "Max Transfer Unit" (MTU) is the maximum byte length of an Ethernet frame with EtherCAT datagrams.
Frame repeat count	<p>The TwinCAT EtherCAT master supports multiple sending of EtherCAT frames for the purpose of enhanced interference immunity.</p> <p>ATTENTION: the used and affected EtherCAT slaves must support this functionality. The slave manufacturer specifies this in the ESI description.</p>

### Distributed Clocks

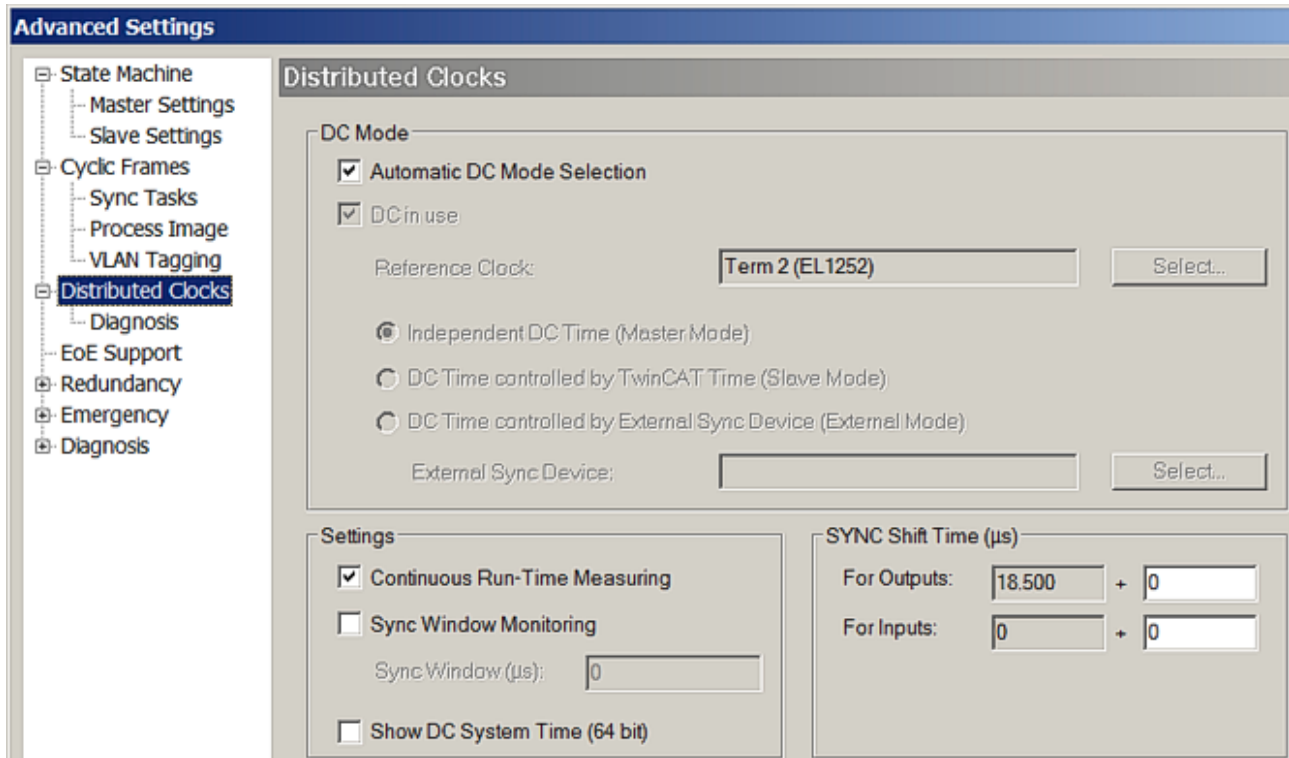


Fig. 205: Distributed Clocks

These settings are discussed in a [separate section](#) [▶ 147].

### EoE support (Ethernet over EtherCAT)

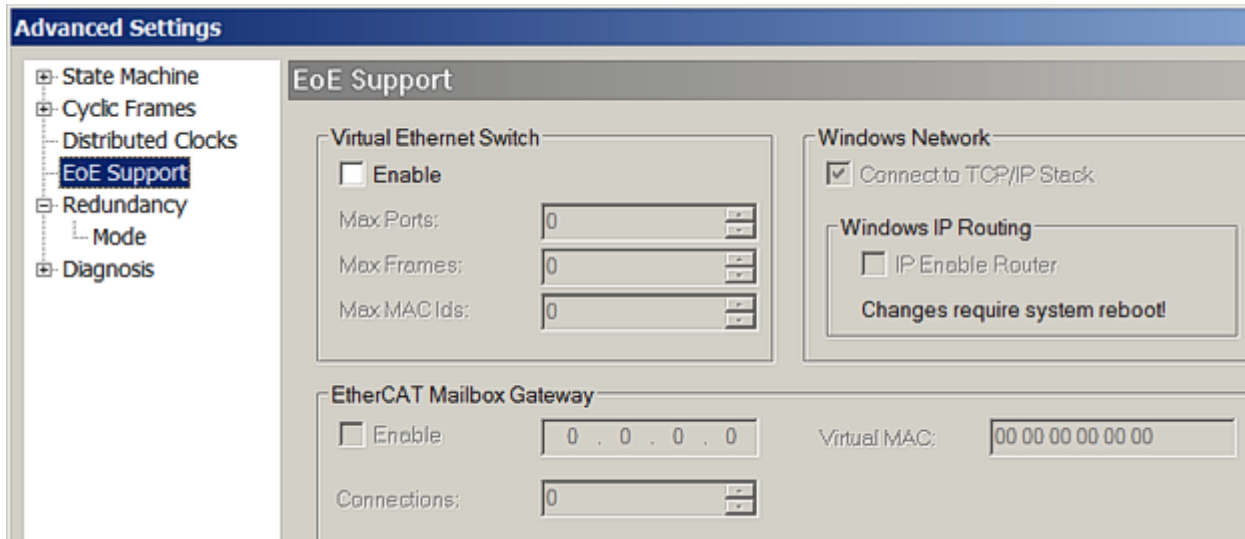


Fig. 206: Ethernet over EtherCAT support

Element	Detail	Explanation	Effects
Virtual Ethernet switch		<p>The forwarding of standard TCP/IP traffic via the virtual switch within the TwinCAT EtherCAT system, is automatically set depending on the slaves used. EL6601 devices (SwitchPort terminals) e.g. result in activation of the VirtualEthernetSwitch and adding of ports.</p> <p>(see Fig. "Activating the VirtualEthernetSwitch")</p> <p>For further information please refer to the respective terminal documentations (EL6601, EL6614).</p>	<p>Explicit activation and specification of ports is required for communication with an intelligent drive via EoE for the purpose of parameterization or firmware updates. In this case a port must be created for each connected device.</p> <p>The number of "Max.Frames" represents the internal queue and can be increased in the event of throughput problems. However, in this case the EtherCAT cycle time and the mailbox-sizes should be checked first.</p> <p>See also EL6601 documentation.</p>

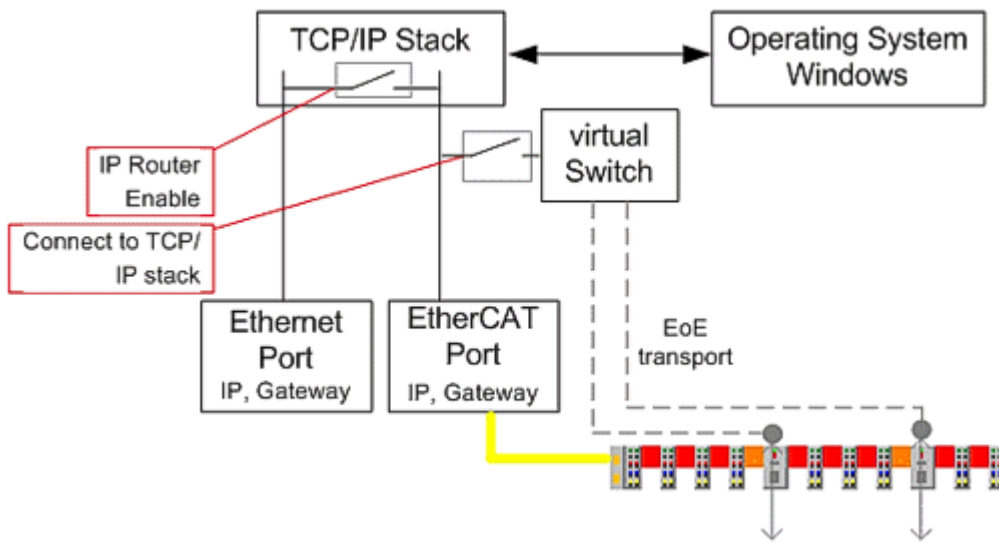


Fig. 207: Activating the VirtualEthernetSwitch

Element	Detail	Explanation	Effects
Windows Network	IP Routing	See diagram above.	
EtherCAT mailbox gateway		This setting is required for special slaves.	



Cable redundancy

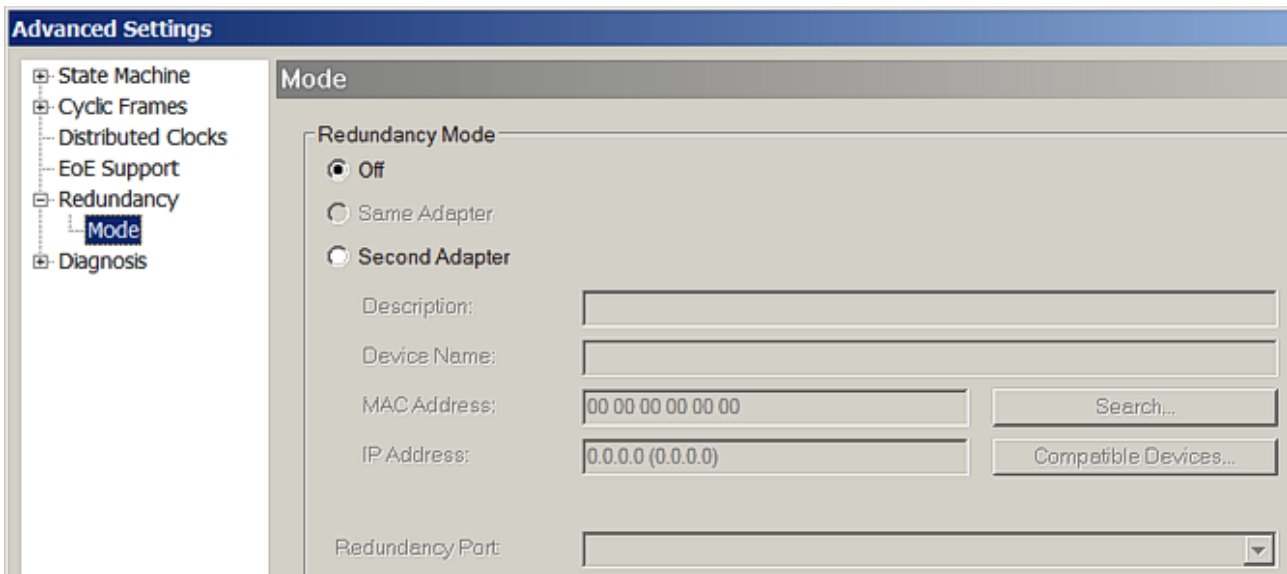


Fig. 208: EtherCAT cable redundancy

The options for media redundancy are described in a separate [Cable Redundancy section](#) [► 30]. A TwinCAT Supplement license is required.

If an Ethernet port is entered here (installation real-time-driver see here), the configuration cannot assume RUN state without a license.

## Sync Unit assignment

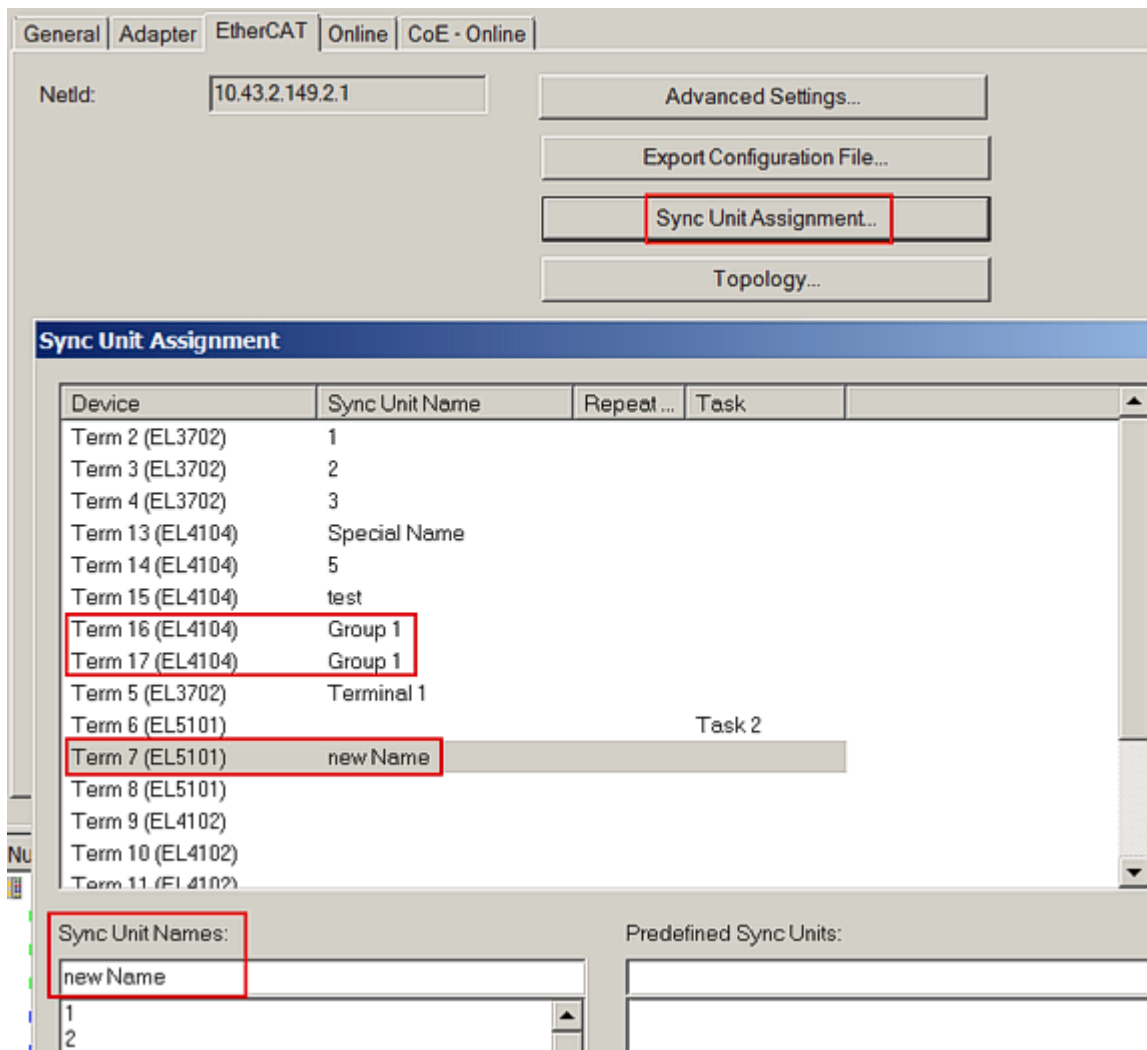


Fig. 209: Sync Unit assignment

The Sync Unit assignment only applies to the cyclic data of the EtherCAT system.

By default the System Manager implements a very efficient assignment of cyclic I/O data and sent datagrams. This means as many as possible/all cyclic data are packed in a single/a few datagrams, resulting in low bus load thanks to low telegram overhead. With only a few, in the best case a *single* datagram as many as possible, in the best case *all* slaves can be addressed.

The working counter is a diagnostic tool in the EtherCAT system. Each slave that is tasked to process a datagram (writing or reading of data) increases the WorkingCounter (WC). The master sends the datagrams with WC=0 and expects them back with WC>0. By checking the WC the master can immediately detect whether all controlled slaves have processed the datagram correctly. If this is not the case, the master cannot trust the returned data and discards *all* input data from this datagram. In addition, it commences acyclic diagnostic measures to determine the fault location.

If working counter errors are expected in a system, e.g. because the flexible "HotConnect" topology concept is used, this dialog can be used to allocate coupler modules or individual slaves/terminals to special datagrams, the so-called. SyncUnits. In extreme cases each slaves is allocated a separate datagram, resulting in very inefficient bus utilization since many datagrams and Ethernet frames have to be sent with associated overhead. Note: an Ethernet frame may contain up to 16 EtherCAT datagrams.

## 3.7 Notes on distributed clocks

### 3.7.1 EtherCAT Distributed Clocks - default settings

#### General

The distributed clock technology in the EtherCAT system enables synchronized operation of local clocks in all EtherCAT devices (master and slaves). If an EtherCAT slave supports distributed clocks (DC), its ESC (EtherCAT slave controller) contains a hardware-based clock (usually 64 bit, in rarer cases 32 bit) with a resolution of 1 bit = 1 ns. These local clocks can be used for synchronous outputs or data acquisition (e.g. of analog value inputs). An EtherCAT slave may support DC, but does not have to. Mixed operation in the EtherCAT system is possible, so long as the EtherCAT master supports DC.

**One** device is the reference clock, all other DC-capable devices are continuously synchronized with an accuracy of generally less than 100 ns. The synchronization process and the method of communication of EtherCAT dictate that the **first** DC-capable slave in the system represents the reference clock ("M" in Fig. *Topology of the EtherCAT system with DC-capable devices*). Via a special telegram the subsequent slaves cyclically receive information about the state of the reference clock and readjust themselves based on this time.

Data from the readjustment process provide important diagnostic information about the state of the distributed clock system.

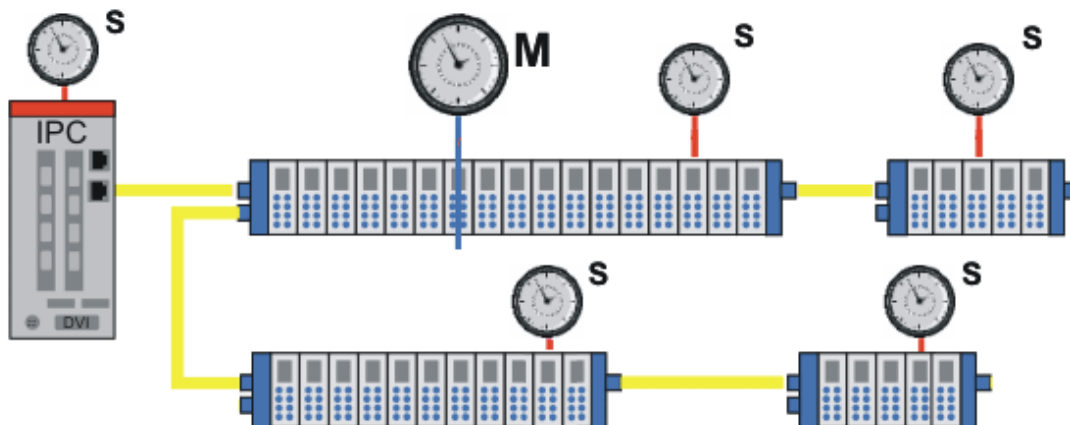


Fig. 210: topology of the EtherCAT system with DC-capable devices

The TwinCAT EtherCAT master deals with the basic topology-dependent calculations, the adjustment during EtherCAT startup and the continuous synchronization. The corresponding settings are specified in the System Manager configurator dialogs.

Additional notes and more detailed information about the distributed clocks system can be found in the respective sections [\[► 205\]](#).

#### **i** Distributed clocks in operation

The distributed clock system is synchronized when EtherCAT starts up during the transition from PREOP to OP. The slaves are set to OP state. In DC-capable slaves proper synchronization in OP state is usually important. Otherwise the slaves automatically return to PREOP state. TwinCAT 2.11 can synchronize such devices and switch them to OP.

#### Default settings for the EtherCAT master

#### **i** Effectiveness of modifications

The distributed clock system is analyzed and calculated when EtherCAT starts up. Changes in the settings of this system therefore always require activation of the modified configuration and an EtherCAT restart.

By default the distributed clocks system (DC) is calculated such that the usual I/O configurations can run in a stable manner. Nevertheless, it may be advisable to adjust the settings during machine commissioning using the diagnostic tools provided.

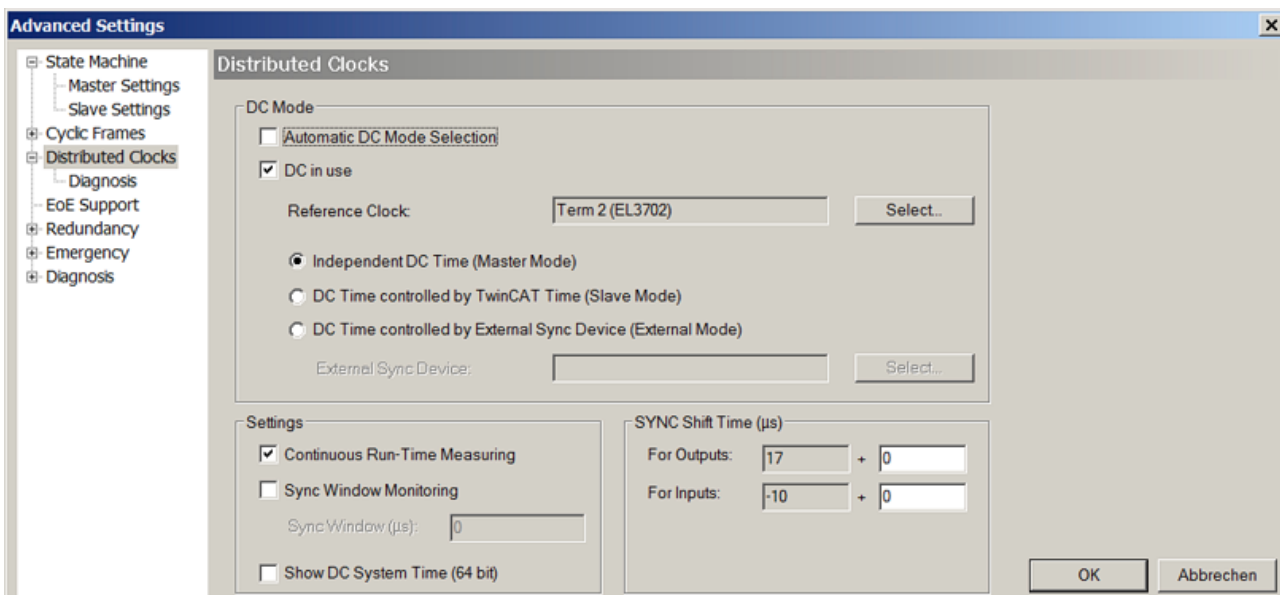


Fig. 211: distributed clock master settings

By default is “Automatic DC mode selection” is active. “DC in use” should only be selected for explicit modifications of the automatic configuration.

Element	Detail	Explanation	Effects
DC mode	Automatic DC mode selection	Default setting, automatic selection of the ReferenceClock	
	DC in use	The ReferenceClock (see next paragraph) and the synchronization direction (see section "Coupling of EtherCAT [▶ 154]") can be selected manually.  If only one EtherCAT device is available in the configuration and DC slaves are used, "Independent DC Time" should be selected (exception: <a href="#">external synchronization [▶ 250]</a> ).	Take care when changing these settings! The stability of the whole system may be impaired.
Settings	Continuous Runtime Measuring	Cyclic measurement of the intervals between the devices during runtime.  This process also takes places for EtherCAT.	For new applications under TwinCAT 2.11 it is advisable to deactivate this function
	Sync Window monitoring	If activated, EtherCAT <i>DevState</i> shows in bit 12 whether all DC devices are maintaining their local clocks within the specified window (see Fig. <i>DevState with SyncWindow monitoring display</i> ). A cyclic BRD command on x092C (system time difference) is used for this.  The information is only usable if the first EtherCAT device also contains the ReferenceClock clock.	
	Show DC system time (64-bit)	If activated, the current DC time is displayed in the inputs of the EtherCAT master as a copy from the master clock. Since the read out process is subject to the fieldbus transport, preference should be given to the PLC blocks in order to obtain the current DC system time.  (see Fig. "DcSysTime" display in the <i>TwinCAT tree</i> )	

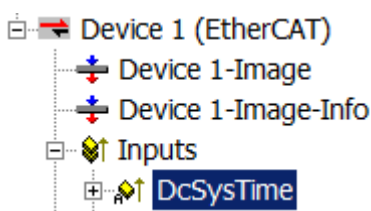


Fig. 212: "DcSysTime" display in the TwinCAT tree

Element	Detail	Explanation	Effects
SYNC shift time		The automatically calculated shift times for inputs and outputs are shown (grey fields). In addition the local slave shift events slave can be offset through manual entries.  Further information can be found in the <a href="#">general distributed clock section [▶ 205]</a> .	The results from the DC diagnostics should be reported here, if applicable.  In the event of synchronization problems, for the outputs +10–20 % of the cycle time, for inputs -10–20 % of the cycle time can be entered here as reference value (unit: $\mu$ s).  Values >100 % of the cycle time are not meaningful.

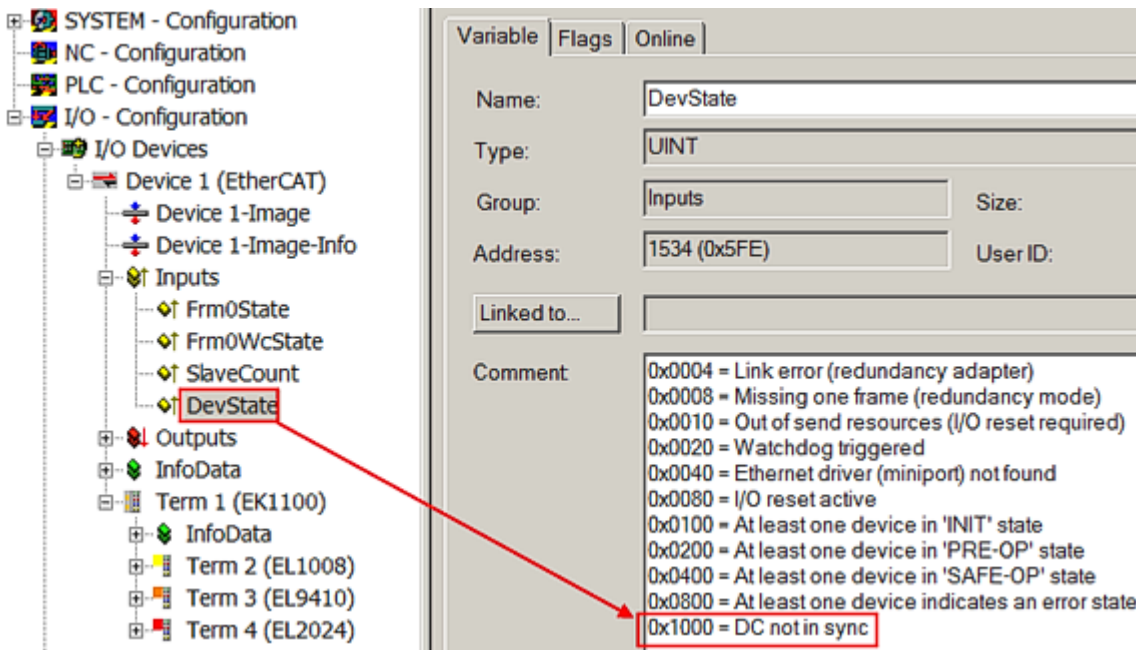


Fig. 213: DevState with SyncWindow display monitoring

**Reference Clock selection**

Manual selection of the reference clock for this EtherCAT system is only required in exceptional circumstances and should be done with caution. TwinCAT generally selects the correct DC-supporting slave, i.e. the first one.

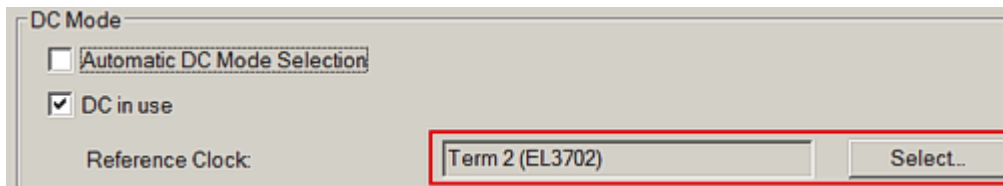


Fig. 214: Manual selection of the reference clock

If a different slave is selected manually

- this one must either support DC explicitly (DC tab is shown in the slave):

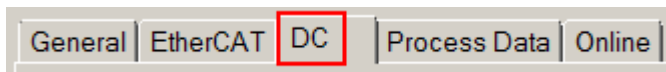


Fig. 215: Distributed Clocks tab (DC)

- or must be manually marked with

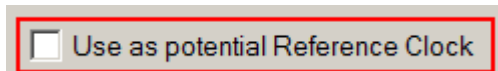


Fig. 216: checkbox for manual selection

in the advanced slave settings.

**i Potential reference clock**

If a slave without the required hardware support is marked as “potential Reference Clock”, the EtherCAT system has no reference clock after startup. The slaves will not switch to OP state and report PREOP\_ERR. From version 2.11R2 build 2032, TwinCAT checks during each startup whether the selected reference clock also supports this function. The following logger message appears: "slave xx is reference clock device, but does not support dcl" (see Fig. "Event logger message: no support by reference clock")

In logger window TwinCAT 2.11R2 (from build 2028) will issue a message “DC not synchronized”, if a slave cannot be switched from PREOP to OP state because it is not synchronized. (see Fig. "Event logger message: no synchronization")


Server (Port)	Timestamp	Message
 (65535)	4/18/2011 1:33:39 PM 718 ms	'Term 2 (EL2032)'(1002): is reference clock devive, but does not support dcl

Fig. 217: Event logger message: no support by reference clock


Server (Port)	Timestamp	Message
 (65535)	25.03.2011 15:22:36 171 ms	'Term 4 (EL4712)': DC not synchronized!

Fig. 218: Event logger message: no synchronization

**Default settings for EtherCAT slaves**

The distributed clocks settings can be found in the advanced settings of the EtherCAT slave.

If the slave is intended by the manufacturer for DC operation, it includes such an operation mode; see Fig. *Default slave setting – slave with and without DC capability* below. Selection of this operation mode activates the integration of the slave into the synchronization mechanism on the master side.

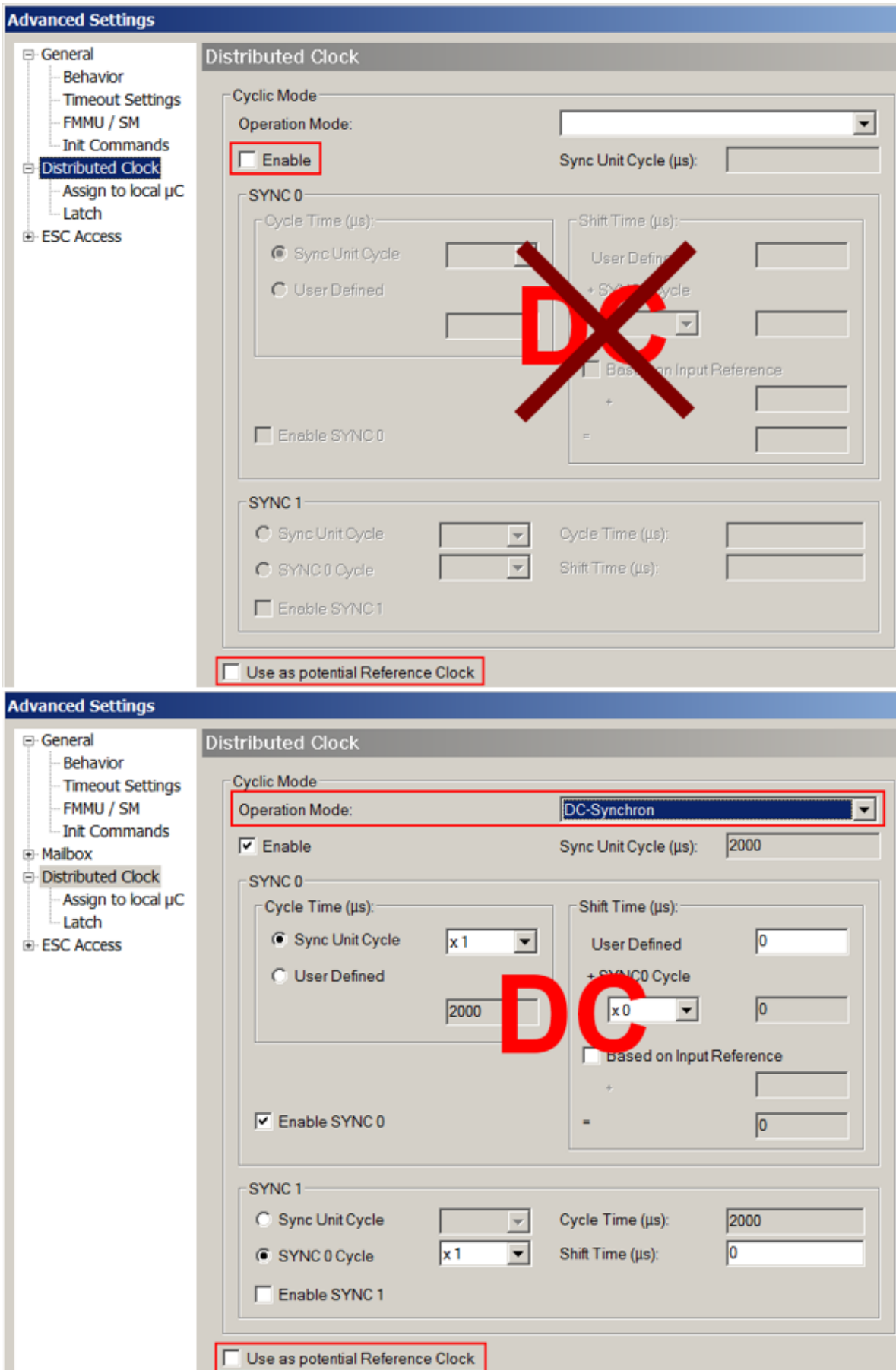


Fig. 219: Default slave setting – slave with and without DC capability



**● Distributed clock activation**

**i** If the settings “Enable” and “Use as potential Reference Clock” are activated for devices that do not support this functionality, the system may malfunction. The function must be enabled by the device manufacturer.

**● Slave shift time settings**

**i** As in the master settings for all slaves, the SYNC times can be offset separately in the individual DC slaves. Not the general [instructions](#) [▶ 220].

**Checking for DC support**

EtherCAT slaves with DC function are

- loaded with the DC configuration during the transition from SAFEOP to PREOP.
- synchronized during the transition from SAFEOP to OP.

If problems occur in the individual phases as a result of incorrect configuration, the target states are not reached and the logger window of the System Manager shows associated information (e.g. “DC invalid sync cfg”)

'PREOP to SAFEOP' failed! Error: 'check device state for SAFEOP'. AL Status '0x0012' read and '0x0004' expected. AL Status Code '0x0030 - DC invalid sync cfg'

Fig. 220: logger window showing information relating to invalid DC configuration of the slave

To check whether and to what extent a device supports distributed clocks, the local clock registers can be displayed online. In the online display call up the Properties dialog by right-clicking in the free space.

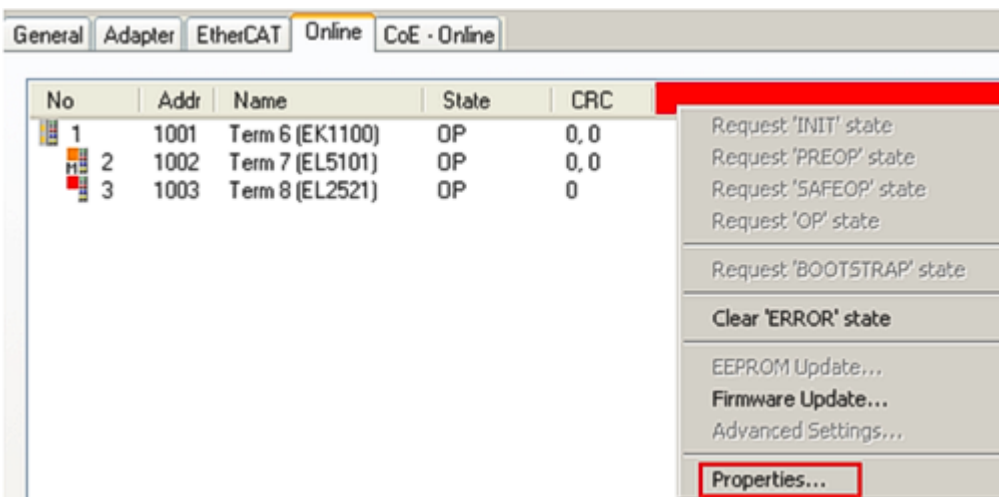


Fig. 221: display of additional register values from the slaves

In this case select the 4 registers 0x0910 to 0x0916. The local DC clock runs in these 2-byte registers. Like certain other slave registers, the System Manager can read these cyclically online.

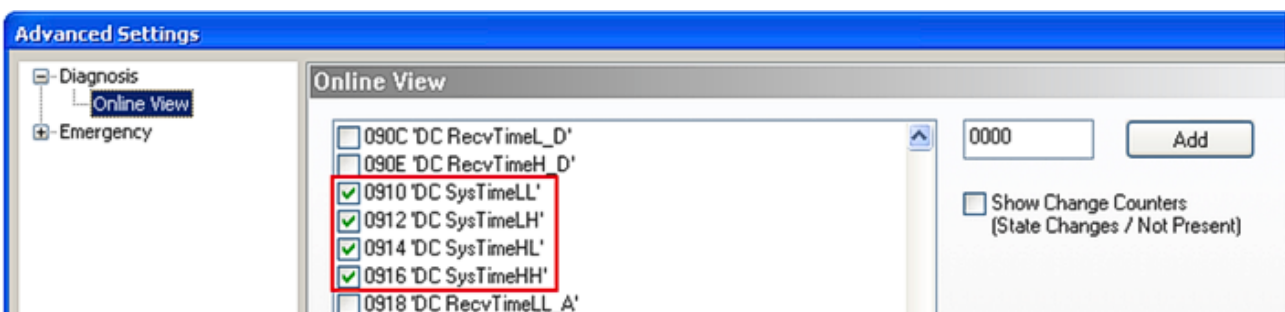


Fig. 222: DC register selection

The register values are displayed and automatically updated by the System Manager.

No	Addr	Name	State	CRC	Reg:0910	Reg:0912	Reg:0914	Reg:0916
1	1001	Term 6 (EK1100)	OP	0, 0	0xBEB9 (48825)	0x7D89 (32137)	0xD516 (54550)	0x04E7 (1255)
2	1002	Term 7 (EL5101)	OP	0, 0	0x904E (36942)	0x7D88 (32136)	0xD516 (54550)	0x04E7 (1255)
3	1003	Term 8 (EL2521)	OP	0, 0	0xC141 (49473)	0x7D89 (32137)	0xD516 (54550)	0x04E7 (1255)
4	1004	Term 10 (EL1002)	OP	0, 0	...	...	...	...
5	1005	Term 5 (EL4712)	OP	0	0x0DBF (3519)	0x7D89 (32137)	...	...

full 64 bit DC clock  
 no DC support in HW  
 32 bit DC support (~ 4.2 sec)

Fig. 223: display of DC register values

The following can be seen in Fig. *Display of DC register values*

- slaves with 64-bit DC support (highlighted in green here)
- slaves without DC support (highlighted in red here)
- slaves with 32-bit DC support (highlighted in yellow here)  
32 bits cover approx. 4.2 seconds. This is adequate for synchronization of the DC system. Information for working with 32/64 bit times in the PLC can be found in the corresponding [section \[► 229\]](#).

### ● Up-to-dateness of the online display

**I** The System Manager reads and displays the values without claim of real-time and consistency. They have informative character and can provide an initial overview. The reading process takes up capacity in the acyclic EtherCAT communication, potentially at the expense of other applications. Not all slave ESC register values can be read by the System Manager/EtherCAT master.

## 3.7.2 EtherCAT Distributed Clocks - coupling of EtherCAT systems

An EtherCAT system with distributed clock functionality has the following properties:

- the controller/IPC uses *one* Ethernet port  
Note: for cable redundancy *two* Ethernet ports are used. However, the combination of distributed clocks (DC) and cable redundancy is only possible with device CU2508, see associated device documentation.
- up to 65,535 slaves are cyclically/acyclically supplied with process data at this port by a real-time EtherCAT master.
- the first DC-capable EtherCAT slave in the system represents the reference clock (Fig. *Distributed clocks system topology: "M"*), to which all subsequent slaves are synchronized (Fig. *Distributed clocks system topology: "S"*)
- in order to keep the real-time synchronous with the DC environment, the internal TwinCAT real-time clock is also synchronized with this field ReferenceClock.

A TwinCAT controller can therefore contain only *one* reference clock, which synchronizes the TwinCAT real-time. All other EtherCAT systems must adjust themselves and their local ReferenceClocks accordingly. All EtherCAT systems nevertheless retain their local ReferenceClocks in first DC EtherCAT slave. This should be taken into account in the following settings.

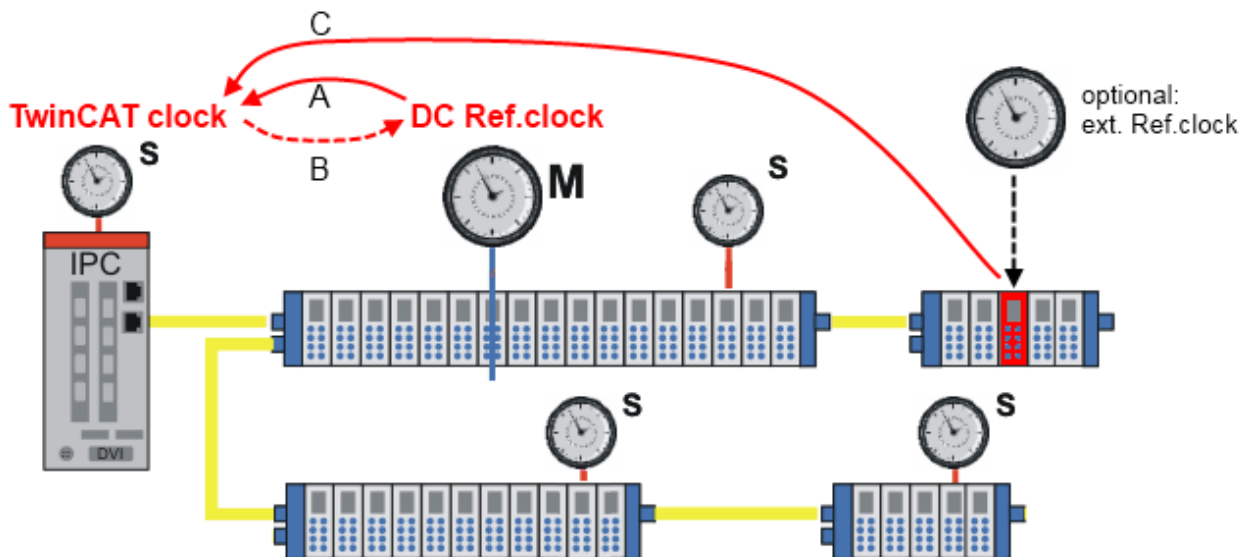


Fig. 224: distributed clocks system topology

The synchronization direction can be set in the advanced settings of the EtherCAT master.

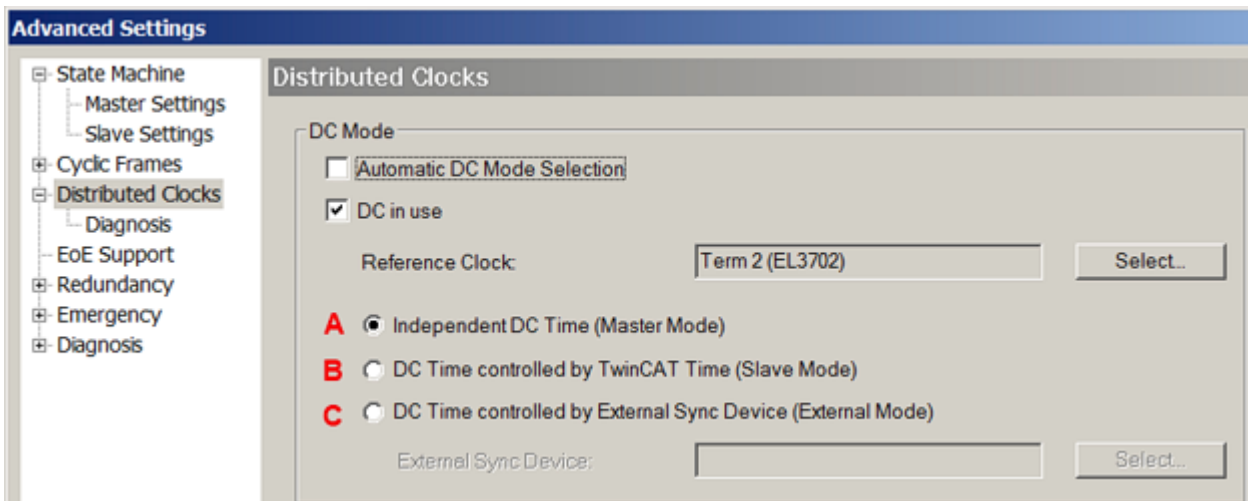


Fig. 225: synchronization direction

If more than one EtherCAT system is used in a TwinCAT control system, i.e. if the I/O configuration contains more than one "EtherCAT device" and some or all of these use distributed clocks functions, proceed as follows:

- in the DC settings set one system to "Independent mode".  
In this system a ReferenceClock sits in a slave and synchronizes all other slaves in this system. The TwinCAT real-time of this clock is also synchronized based on frequency.
- all other DC systems should be set to "Slave Mode".  
These systems also contain the local ReferenceClocks for synchronizing the subsequent devices. However, during the EtherCAT startup and subsequently this ReferenceClock is itself synchronized based on the TwinCAT time and is referred to as "tracking reference clock".

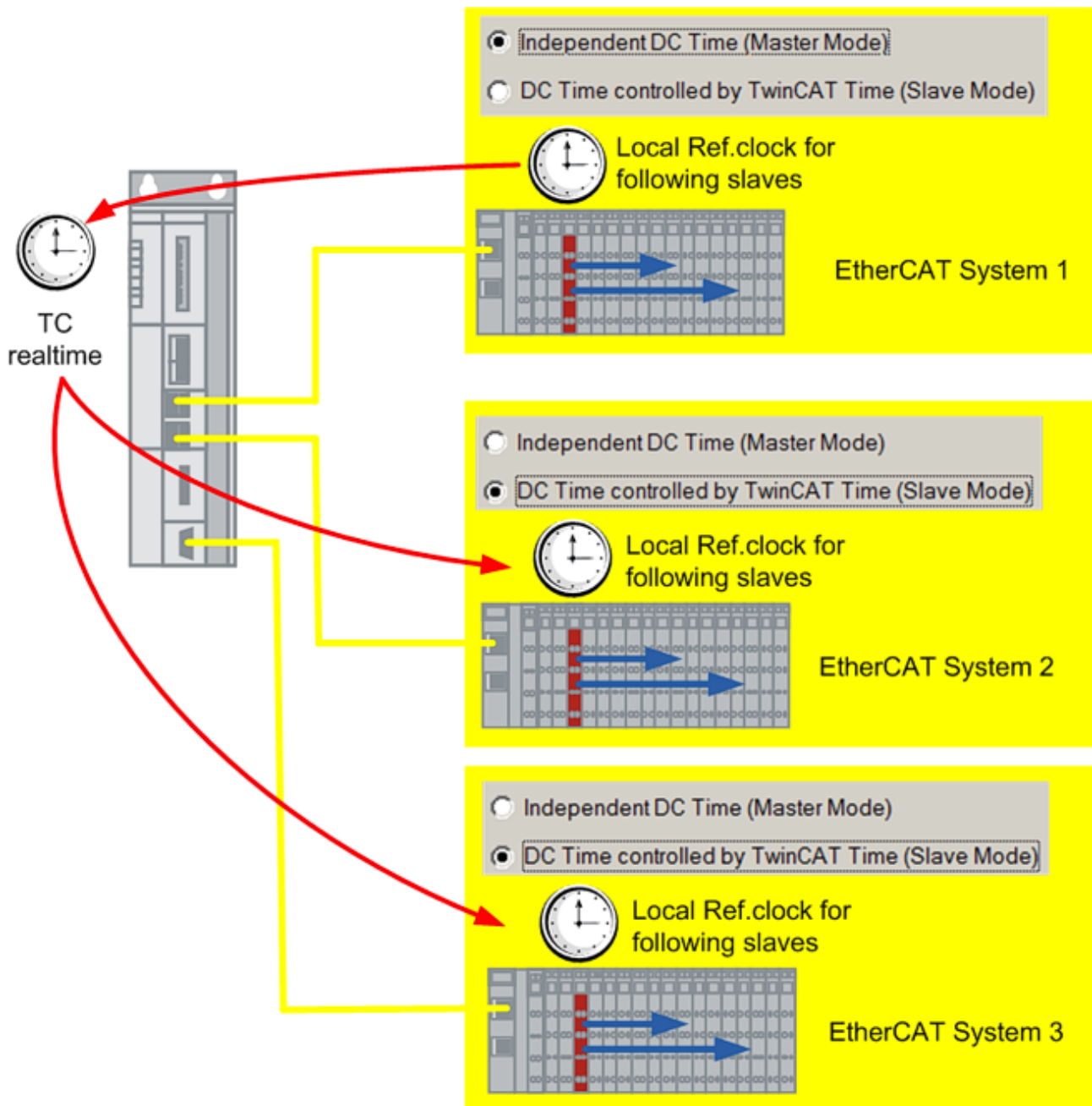


Fig. 226: DC-coupling of EtherCAT systems

**i** **Notes**

- This kind of DC-coupling is only possible from TwinCAT 2.11
- Only EtherCAT slaves without inherent intelligence/firmware should be used as "tracking reference clock". Such slaves are characterized by the lack of CoE/SoE or process data. Simple digital input/output terminals (EL1202-0100, EL2202-0100) or couplers (EK1100) are suitable.
- WARNING! Only devices that support this functionality should be selected as ReferenceClock. See notes in section "[DC standard settings \[P 147\]](#)".

Failure to follow these instructions may have the following effects: DC devices in subordinate systems do not assume OP state and remain in PREOP state. Devices come out of synchronization and OP state due to "SyncLost".

**Tracking accuracy**

As can be seen in Fig. *DC-coupling of EtherCAT systems*, a chain of synchronizations must be executed via the TwinCAT controller through to the "tracking reference clock". The control accuracy of the EtherCAT systems among each other that can be achieved in this way is reduced. The user needs to check whether this approach meets the long-term requirements of the application in terms of control accuracy and stability.

The CU2508 port multiplier is designed for maximum synchronization precision. It supports up to 8 connected EtherCAT systems (or 4 with simultaneous cable redundancy). It also enables the combination of distributed clock function and cable redundancy. The configuration of the CU2508 is transparent, i.e. the configuration may still contain independent EtherCAT devices.

For further information please refer to the [CU2508 documentation](#).

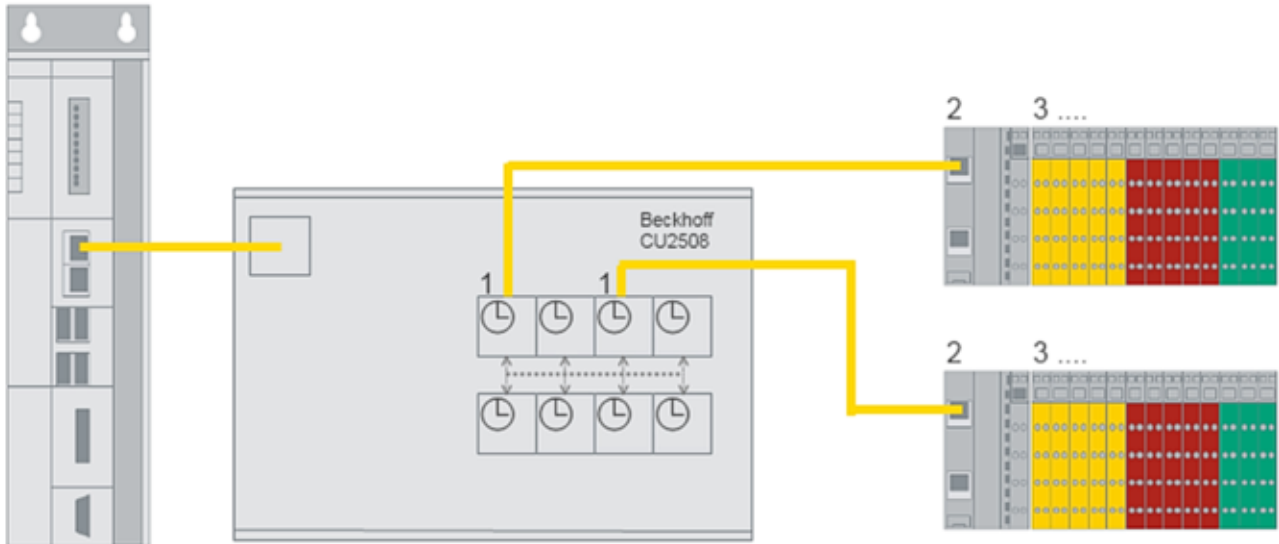


Fig. 227: EtherCAT topology with CU2508 and 2 EtherCAT systems

## 4 EtherCAT diagnostics

### 4.1 General Notes - EtherCAT Slave Application

This summary briefly deals with a number of aspects of EtherCAT Slave operation under TwinCAT. More detailed information on this may be found in the corresponding sections of, for instance, the [EtherCAT System Documentation](#).

#### Diagnosis in real time: WorkingCounter, EtherCAT State and Status

Generally speaking an EtherCAT Slave provides a variety of diagnostic information that can be used by the controlling task.

This diagnostic information relates to differing levels of communication. It therefore has a variety of sources, and is also updated at various times.

Any application that relies on I/O data from a fieldbus being correct and up to date must make diagnostic access to the corresponding underlying layers. EtherCAT and the TwinCAT System Manager offer comprehensive diagnostic elements of this kind. Those diagnostic elements that are helpful to the controlling task for diagnosis that is accurate for the current cycle when in operation (not during commissioning) are discussed below.

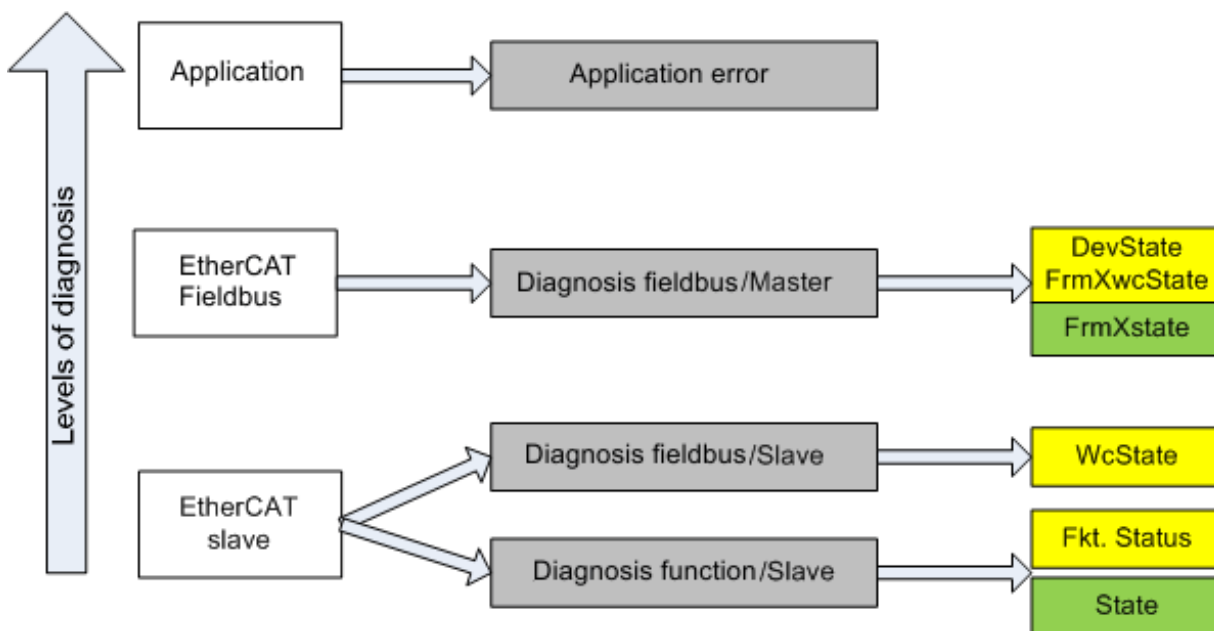


Fig. 228: Selection of the diagnostic information of an EtherCAT Slave

In general, an EtherCAT Slave offers

- communication diagnosis typical for a slave (diagnosis of successful participation in the exchange of process data, and correct operating mode)  
This diagnosis is the same for all slaves.

as well as

- function diagnosis typical for a channel (device-dependent)  
See the corresponding device documentation

The colors in Fig. *Selection of the diagnostic information of an EtherCAT Slave* also correspond to the variable colors in the System Manager, see Fig. *Basic EtherCAT Slave Diagnosis in the PLC*.

Colour	Meaning
yellow	Input variables from the Slave to the EtherCAT Master, updated in every cycle
red	Output variables from the Slave to the EtherCAT Master, updated in every cycle
green	Information variables for the EtherCAT Master that are updated acyclically. This means that it is possible that in any particular cycle they do not represent the latest possible status. It is therefore useful to read such variables through ADS.

Fig. Basic EtherCAT Slave Diagnosis in the PLC shows an example of an implementation of basic EtherCAT Slave Diagnosis. A Beckhoff EL3102 (2-channel analogue input terminal) is used here, as it offers both the communication diagnosis typical of a slave and the functional diagnosis that is specific to a channel. Structures are created as input variables in the PLC, each corresponding to the process image.

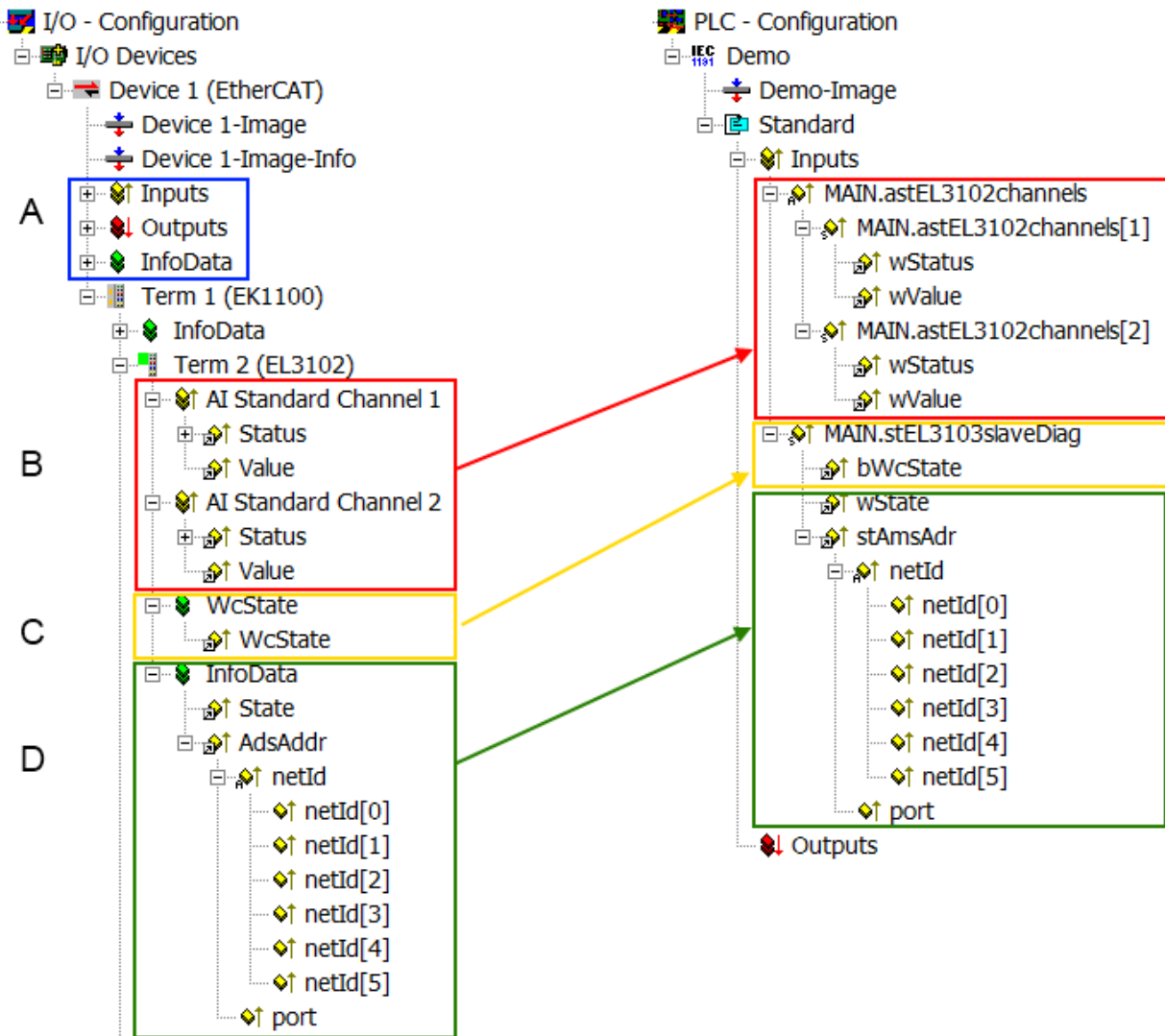


Fig. 229: Basic EtherCAT Slave Diagnosis in the PLC

The following aspects are covered here:

Code	Function	Implementation	Application/evaluation
A	The EtherCAT Master's diagnostic information updated acyclically (yellow) or provided acyclically (green).		At least the DevState is to be evaluated for the most recent cycle in the PLC. The EtherCAT Master's diagnostic information offers many more possibilities than are treated in the EtherCAT System Documentation. A few keywords: <ul style="list-style-type: none"> <li>• CoE in the Master for communication with/through the Slaves</li> <li>• Functions from <i>TcEtherCAT.lib</i></li> <li>• Perform an OnlineScan</li> </ul>
B	In the example chosen (EL3102) the EL3102 comprises two analogue input channels that transmit a single function status for the most recent cycle.	Status <ul style="list-style-type: none"> <li>• the bit significations may be found in the device documentation</li> <li>• other devices may supply more information, or none that is typical of a slave</li> </ul>	In order for the higher-level PLC task (or corresponding control applications) to be able to rely on correct data, the function status must be evaluated there. Such information is therefore provided with the process data for the most recent cycle.
C	For every EtherCAT Slave that has cyclic process data, the Master displays, using what is known as a WorkingCounter, whether the slave is participating successfully and without error in the cyclic exchange of process data. This important, elementary information is therefore provided for the most recent cycle in the System Manager <ol style="list-style-type: none"> <li>1. at the EtherCAT Slave, and, with identical contents</li> <li>2. as a collective variable at the EtherCAT Master (see Point A)</li> </ol> for linking.	WcState (Working Counter) 0: valid real-time communication in the last cycle 1: invalid real-time communication This may possibly have effects on the process data of other Slaves that are located in the same SyncUnit	In order for the higher-level PLC task (or corresponding control applications) to be able to rely on correct data, the communication status of the EtherCAT Slave must be evaluated there. Such information is therefore provided with the process data for the most recent cycle.
D	Diagnostic information of the EtherCAT Master which, while it is represented at the slave for linking, is actually determined by the Master for the Slave concerned and represented there. This information cannot be characterized as real-time, because it <ul style="list-style-type: none"> <li>• is only rarely/never changed, except when the system starts up</li> <li>• is itself determined acyclically (e.g. EtherCAT Status)</li> </ul>	State current Status (INIT..OP) of the Slave. The Slave must be in OP (=8) when operating normally. <i>AdsAddr</i> The ADS address is useful for communicating from the PLC/task via ADS with the EtherCAT Slave, e.g. for reading/writing to the CoE. The AMS-NetID of a slave corresponds to the AMS-NetID of the EtherCAT Master; communication with the individual Slave is possible via the <i>port</i> (= EtherCAT address).	Information variables for the EtherCAT Master that are updated acyclically. This means that it is possible that in any particular cycle they do not represent the latest possible status. It is therefore possible to read such variables through ADS.

### NOTE

#### Diagnostic information

It is strongly recommended that the diagnostic information made available is evaluated so that the application can react accordingly.

#### CoE Parameter Directory

The CoE parameter directory (CanOpen-over-EtherCAT) is used to manage the set values for the slave concerned. Changes may, in some circumstances, have to be made here when commissioning a relatively complex EtherCAT Slave. It can be accessed through the TwinCAT System Manager, see Fig. *EL3102, CoE directory*.



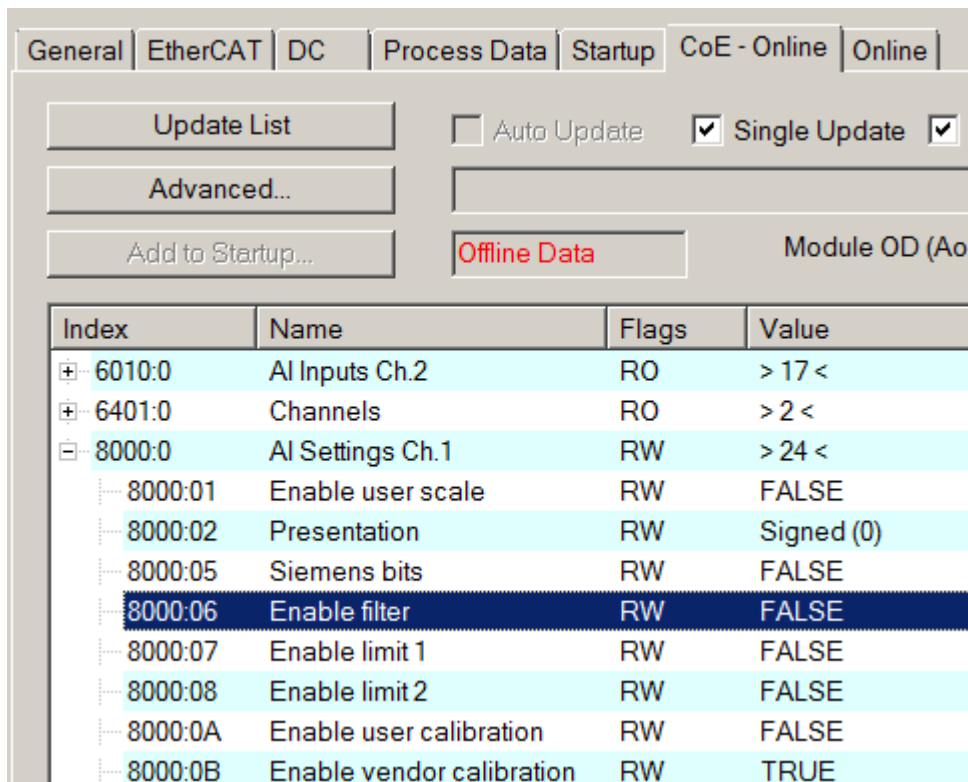


Fig. 230: EL3102, CoE directory

**● EtherCAT System Documentation**

**i** The comprehensive description in the [EtherCAT System Documentation](#) (EtherCAT Basics --> CoE Interface) must be observed!

A few brief extracts:

- Whether changes in the online directory are saved locally in the slave depends on the device. EL terminals (except the EL66xx) are able to save in this way.
- The user must manage the changes to the StartUp list.

**Commissioning aid in the TwinCAT System Manager**

Commissioning interfaces are being introduced as part of an ongoing process for EL/EP EtherCAT devices. These are available in TwinCAT System Managers from TwinCAT 2.11R2 and above. They are integrated into the System Manager through appropriately extended ESI configuration files.

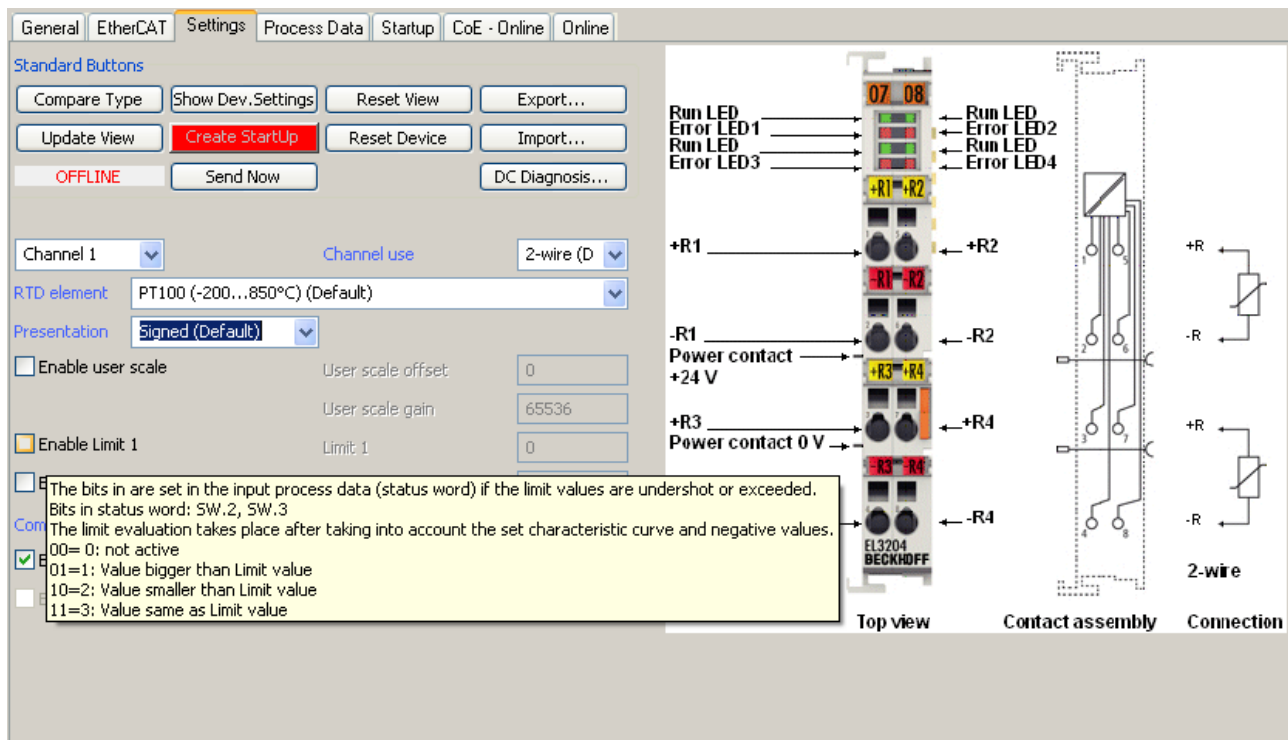


Fig. 231: Example of commissioning aid for a EL3204

This commissioning process simultaneously manages

- CoE Parameter Directory
- DC/FreeRun mode
- the available process data records (PDO)

Although the “Process Data”, “DC”, “Startup” and “CoE-Online” that used to be necessary for this are still displayed, it is recommended that, if the commissioning aid is used, the automatically generated settings are not changed by it.

The commissioning tool does not cover every possible application of an EL/EP device. If the available setting options are not adequate, the user can make the DC, PDO and CoE settings manually, as in the past.

### EtherCAT State: automatic default behaviour of the TwinCAT System Manager and manual operation

After the operating power is switched on, an EtherCAT Slave must go through the following statuses

- INIT
- PREOP
- SAFEOP
- OP

to ensure sound operation. The EtherCAT Master directs these statuses in accordance with the initialization routines that are defined for commissioning the device by the ES/XML and user settings (Distributed Clocks (DC), PDO, CoE). See also the section on "Principles of [Communication, EtherCAT State Machine \[► 19\]](#)" in this connection. Depending how much configuration has to be done, and on the overall communication, booting can take up to a few seconds.

The EtherCAT Master itself must go through these routines when starting, until it has reached at least the OP target state.

The target state wanted by the user, and which is brought about automatically at start-up by TwinCAT, can be set in the System Manager. As soon as TwinCAT reaches the status RUN, the TwinCAT EtherCAT Master will approach the target states.

**Standard setting**

The advanced settings of the EtherCAT Master are set as standard:

- EtherCAT Master: OP
- Slaves: OP  
This setting applies equally to all Slaves.

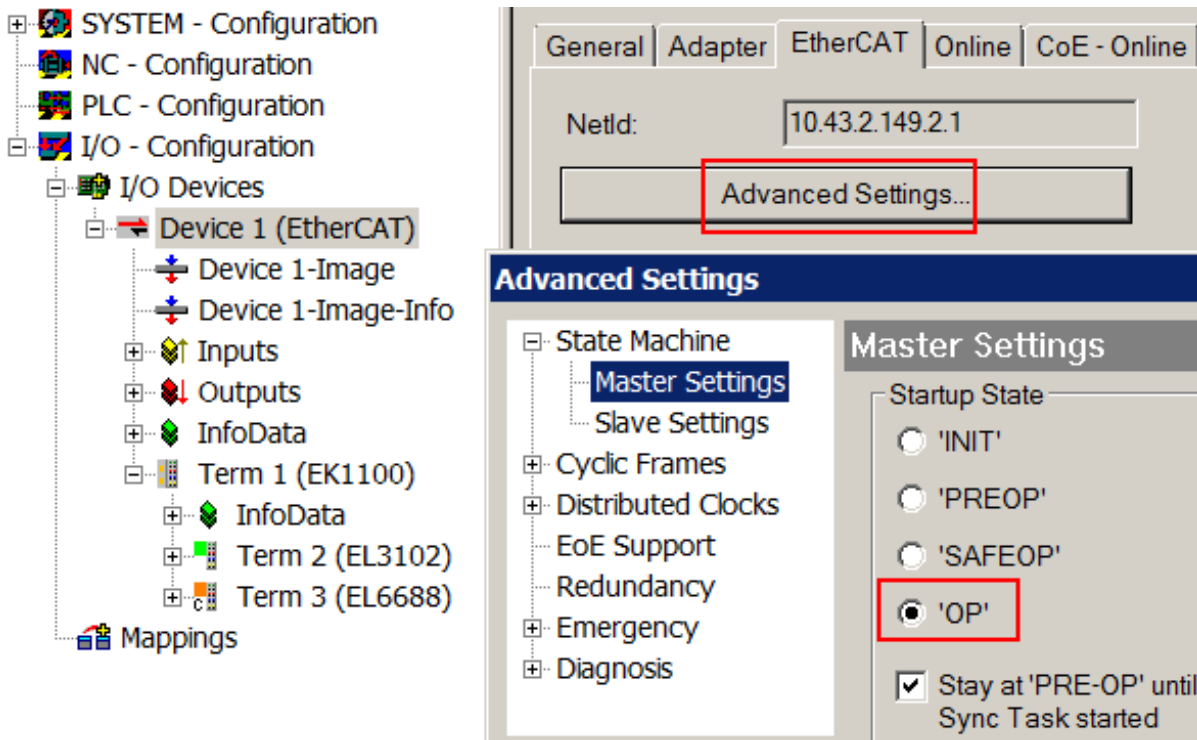


Fig. 232: Default behaviour of the System Manager

In addition, the target state of any particular Slave can be set in the “Advanced Settings” dialogue; the standard setting is again OP.

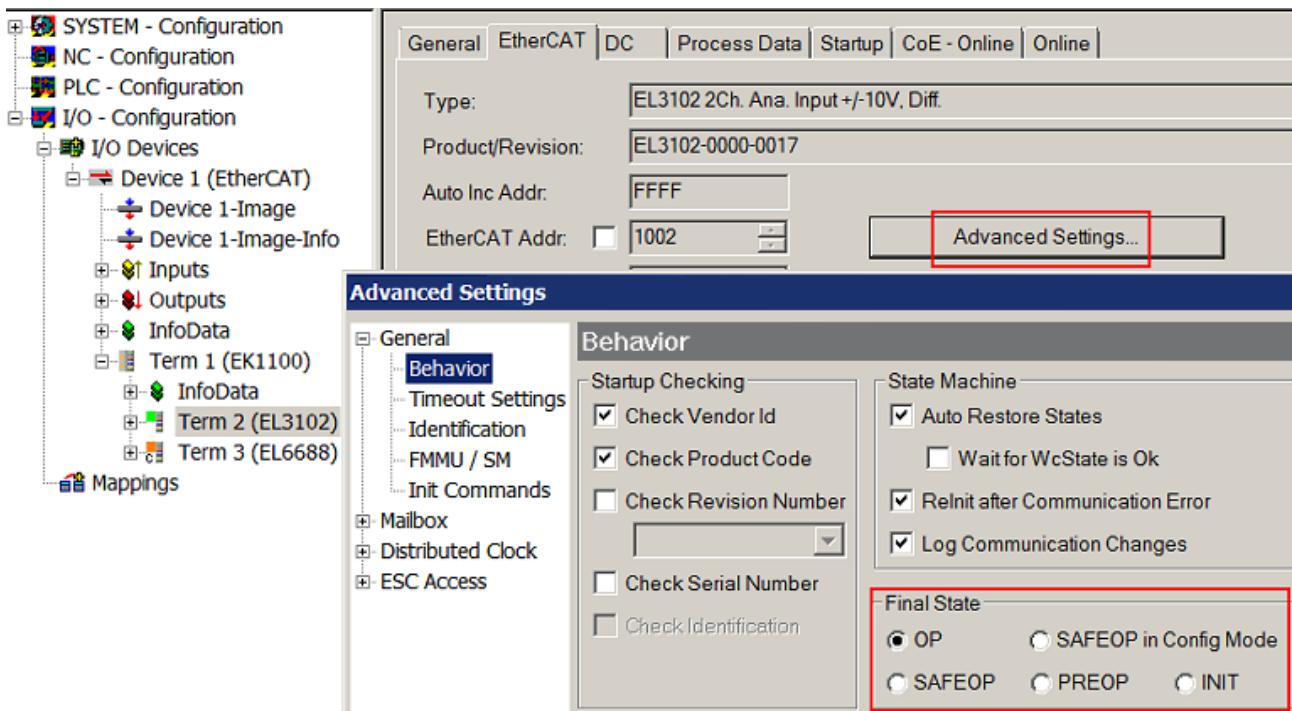


Fig. 233: Default target state in the Slave

## Manual Control

There are particular reasons why it may be appropriate to control the states from the application/task/PLC. For instance:

- for diagnostic reasons
- to induce a controlled restart of axes
- because a change in the times involved in starting is desirable

In that case it is appropriate in the PLC application to use the PLC function blocks from the *TcEtherCAT.lib*, which is available as standard, and to work through the states in a controlled manner using, for instance, *FB\_EcSetMasterState*.

It is then useful to put the settings in the EtherCAT Master to INIT for master and slave.

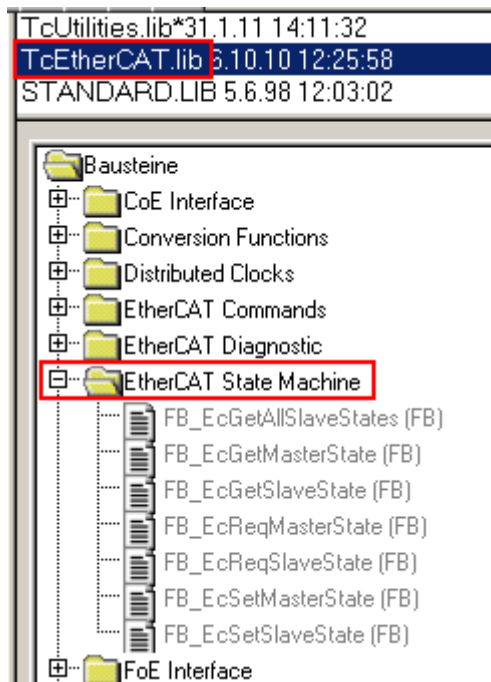


Fig. 234: PLC function blocks

### Note regarding E-Bus current

EL/ES terminals are placed on the DIN rail at a coupler on the terminal strand. A Bus Coupler can supply the EL terminals added to it with the E-bus system voltage of 5 V; a coupler is thereby loadable up to 2 A as a rule. Information on how much current each EL terminal requires from the E-bus supply is available online and in the catalogue. If the added terminals require more current than the coupler can supply, then power feed terminals (e.g. EL9410) must be inserted at appropriate places in the terminal strand.

The pre-calculated theoretical maximum E-Bus current is displayed in the TwinCAT System Manager as a column value. A shortfall is marked by a negative total amount and an exclamation mark; a power feed terminal is to be placed before such a position.

General   Adapter   EtherCAT   Online   CoE - Online						
NetId:		10.43.2.149.2.1		Advanced Settings...		
Number	Box Name	Address	Type	In Size	Out S...	E-Bus (..
1	Term 1 (EK1100)	1001	EK1100			
2	Term 2 (EL3102)	1002	EL3102	8.0		1830
3	Term 4 (EL2004)	1003	EL2004		0.4	1730
4	Term 5 (EL2004)	1004	EL2004		0.4	1630
5	Term 6 (EL7031)	1005	EL7031	8.0	8.0	1510
6	Term 7 (EL2808)	1006	EL2808		1.0	1400
7	Term 8 (EL3602)	1007	EL3602	12.0		1210
8	Term 9 (EL3602)	1008	EL3602	12.0		1020
9	Term 10 (EL3602)	1009	EL3602	12.0		830
10	Term 11 (EL3602)	1010	EL3602	12.0		640
11	Term 12 (EL3602)	1011	EL3602	12.0		450
12	Term 13 (EL3602)	1012	EL3602	12.0		260
13	Term 14 (EL3602)	1013	EL3602	12.0		70
14	Term 3 (EL6688)	1014	EL6688	22.0		-240 !

Fig. 235: Illegally exceeding the E-Bus current

From TwinCAT 2.11 and above, a warning message “E-Bus Power of Terminal...” is output in the logger window when such a configuration is activated:

Message
E-Bus Power of Terminal 'Term 3 (EL6688)' may to low (-240 mA) - please check!

Fig. 236: Warning message for exceeding E-Bus current

**NOTE**

**Caution! Malfunction possible!**

The same ground potential must be used for the E-Bus supply of all EtherCAT terminals in a terminal block!

## 4.2 EtherCAT AL Status Codes

### 4.2.1 Error Code 0x0000

**Meaning**

No error

**Description**

No error

**Current State (or state change)**

Any

**Resulting state**

Current state

**Solution**

n/a

### 4.2.2 Error Code 0x0001

**Meaning**

Unspecified error

**Description**

No error code is defined for occurred error

**Current State (or state change)**

Any

**Resulting state**

Any + E

**Solution**

Read user manual or contact device manufacturer

### 4.2.3 Error Code 0x0002

**Meaning**

No Memory

**Description**

Less hardware memory, slave needs more memory.

Example: For slave configuration, application configuration files are downloaded (possibly via FoE or large CoE objects). The size of those files exceeds the local memory

**Current State (or state change)**

Any

**Resulting state**

Any + E

**Solution**

Download smaller files or objects.

Check user manual.

## 4.2.4 Error Code 0x0004

**Meaning**

Invalid Revision

**Description**

Output/Input mapping is not valid for this hardware or software revision (0x1018:03)

**Current State (or state change)**

P→S

**Resulting state**

P+E

**Solution**

Change mapping or use different hardware

## 4.2.5 Error Code 0x0011

**Meaning**

Invalid requested state change

**Description**

The EtherCAT State Machine (ESM) defines which state changes are allowed. All other state changes are not allowed

Example: If the master requests the slave to go from OP (AL Control = 0x08) directly to BOOT (AL Control = 0x03).

**Current State (or state change)**

P→S, I→O, P→O, O→B, S→B, P→B

**Resulting state**

Current State + E

**Solution**

Go step-by-step from the original state to the desired state.

## 4.2.6 Error Code 0x0012

**Meaning**

Unknown requested state change

**Description**

The ESM defines the following states. They are coded with fixed values (only lower (=right) nibble):

BOOT: AL Control = 0x03

INIT: AL Control = 0x01

PREOP: AL Control = 0x02

SAFEOP: AL Control = 0x04

OP: AL Control = 0x08

The fifth bit of the AL Control (left nibble is 1) is the "Error Acknowledge Bit". If the slave is in AL STATUS = 0x14, i.e. ERROR SAFEOP the master acknowledges this by setting the Acknowledge bit.

Example: If any other value for AL Control than those specified are sent.

**Current State (or state change)**

Any

**Resulting state**

Current State + E

**Solution**

Do only request the defined states

## 4.2.7 Error Code 0x0013

**Meaning**

Boot state not supported

**Description**

Device does not support BOOT state, but the master requests the slave to go to BOOT (AL Control = 0x03)

**Current State (or state change)**

I→B



**Resulting state**

I + E

**Solution**

n/a

## 4.2.8 Error Code 0x0014

**Meaning**

No valid firmware

**Description**

This error code may be returned after a firmware download, if the downloaded file cannot be used by the application controller

**Current State (or state change)**

I→P

**Resulting state**

I + E

**Solution**

Download a firmware that can be supported by the hardware and bootloader. Check Product Code and Revision Number (CoE object 0x1018). If this cannot be read from the firmware any more you may see this in the network configuration (CoE object dictionary) or probably in the ESI file (element Profile: ObjectDictionary:Objects:Object).

## 4.2.9 Error Code 0x0015

**Meaning**

Invalid mailbox configuration

**Description**

Mailbox communication (= acyclic parameter exchange) is done via two memory areas on the EtherCAT Slave Controller (ESC) – the “Output Mailbox” (master -> slave) and the “Input Mailbox” (slave-> master). Those memory areas are protected by SyncManagers to prevent from simultaneous access from master and slave controller at the same time. SyncManagers are hardware entities on the ESC. They are configured via certain registers in the ESC register area (starting at 0x0800). The configuration includes start address, length, and direction (output or input). If those settings differ from those expected by the host controller of the slave this error is returned

**Current State (or state change)**

I→B

**Resulting state**

n/a

**Solution**

Replace previous network description of old slave with the one of the new slave

## 4.2.10 Error Code 0x0016

**Meaning**

Invalid mailbox configuration

**Description**

Example: The slave hardware was replaced while the network configuration remained unchanged. The new hardware expects different mailbox SyncManager settings

**Current State (or state change)**

I→S

**Resulting state**

I + E

**Solution**

Replace previous network description of old slave with the one of the new slave

## 4.2.11 Error Code 0x0017

**Meaning**

Invalid Sync Manager configuration

**Description**

Process data communication (cyclic communication) is done via extra memory areas on the ESC, separated for outputs and inputs. The process data length and the process data SyncManager length have to be the same. If this is not the case or the start address or direction does not match this error is returned.

Example: The process data configuration was changed of the slaves which also changed the length of the data. The change was not activated in the configuration so that the configuration tool would have recalculated the SyncManager settings.

**Current State (or state change)**

P→S, S→O

**Resulting state**

Current State + E

**Solution**

Issue a re-calculation of the EtherCAT configuration

## 4.2.12 Error Code 0x0018

### Meaning

No valid inputs available

### Description

The slave application cannot provide valid input values

Example: A certain hardware which needs to be connected to the slave was disconnected

### Current State (or state change)

O, S→O

### Resulting state

S + E

### Solution

n/a

## 4.2.13 Error Code 0x0019

### Meaning

No valid outputs available

### Description

The slave application cannot receive valid output values.

Example: The slave has a RxPdoToggle output or an "Output Valid" information in its process data. The RxPdoToggle does not toggle or the OutputValid is not true. Therefore the slave has no process data which the application can use. If supported, check the RxPDO Toggle Failed Counter in object 0x1C3x.0E). Also, the Synchronization may have problems (see object 0x10F1:SI2 Sync Error Counter Limit) so that process data are received too late by the slave so that the local slave cycle misses the toggle event. Another reason can be that the PLC stopped working

### Current State (or state change)

O, S→O

### Resulting state

S + E

### Solution

The RxPdoToggle may need to be handled by the PLC program

The outputs valid may have to be set by the PLC program

PLC may have stopped, restart PLC

## 4.2.14 Error Code 0x001A

### Meaning

Synchronization error

### Description

If too many RxPDO Toggle error occur, i.e. the RxPDO Toggle Failed Counter increases the internal limit the slave returns to SAFEPEROR with 0x001A. Multiple synchronization errors. Device is not synchronized any more (used if the causes mirrored by the AL Status Codes 0x2C, 0x2D, 0x32, 0x33, 0x34 cannot be distinguished).

### Current State (or state change)

O, S→O

### Resulting state

S + E

### Solution

n/a

## 4.2.15 Error Code 0x001B

### Meaning

Sync manager watchdog

### Description

The slave did not receive process data within the specified watchdog time. Usually, the WD time is 100ms. The WD is re-started every time it receives new process data, usually when the Output SyncManager (SyncManager2) is written. For devices which have only inputs usually no WD is used. Increasing the WD is not a solution.

Reason: PLC stopped

### Current State (or state change)

O, S

### Resulting state

S + E

### Solution

n/a

## 4.2.16 Error Code 0x001C

### Meaning

Invalid Sync Manager Types

**Description**

n/a

**Current State (or state change)**

O, S, O, P→S

**Resulting state**

S + E

**Solution**

n/a

## 4.2.17 Error Code 0x001D

**Meaning**

Invalid Output Configuration

**Description**

SM configuration for output process data is invalid

**Current State (or state change)**

O, S, O, P→S

**Resulting state**

S + E

**Solution**

n/a

## 4.2.18 Error Code 0x001E

**Meaning**

Invalid Input Configuration

**Description**

SM configuration for input process data is invalid

**Current State (or state change)**

O, S, O, P→S

**Resulting state**

S + E

**Solution**

n/a

**4.2.19 Error Code 0x001F****Meaning**

Invalid Watchdog Configuration

**Description**

The Watchdog is configured in the ESC register 0x0400 and 0x0420. EtherCAT defines default watchdog settings (100ms) or they are defined in the ESI file. If the slave does not accept a change of the expected settings it returns this AL Status Code Example: A slave may not accept that the WD is deactivated.

**Current State (or state change)**

O, S, O, P→S

**Resulting state**

P + E

**Solution**

Use default WD settings

**4.2.20 Error Code 0x0020****Meaning**

Slave needs cold start

**Description**

Slave device require a power off - power on reset

**Current State (or state change)**

Any

**Resulting state**

Current State + E

**Solution**

n/a

**4.2.21 Error Code 0x0021****Meaning**

Slave needs INIT

**Description**

Slave application requests INIT state

**Current State (or state change)**

B, P, S, O

**Resulting state**

Current State + E

**Solution**

n/a

## 4.2.22 Error Code 0x0022

**Meaning**

Slave needs PREOP

**Description**

Slave application requests PREOP state

**Current State (or state change)**

S, O

**Resulting state**

S + E, O + E

**Solution**

n/a

## 4.2.23 Error Code 0x0023

**Meaning**

Slave needs SAFEOP

**Description**

Slave application requests SAFEOP state

**Current State (or state change)**

O

**Resulting state**

O + E

**Solution**

n/a

**4.2.24 Error Code 0x0024****Meaning**

Invalid Input Mapping

**Description**

The process data are described by the configuration (PdoConfig) and PDO assignment (PdoAssign).

PdoConfig: list of actual variables (usually indexes 0x6nnn for inputs and 0x7nnn for outputs). Variables are also called PDO entries. There can be one or several variables with in one list (i.e. within one PDO). The Input PDOs have the index 0x1Amm. The Output PDOs have the index 0x16mm.

PdoAssign: The list of PDOs (object index 0x16nn, 0x1Amm) which are actually part of the process data and hence, are transferred cyclically, are listed in the PDO Assign Objects 0x1C12 (output PDOs) and 0x1C13 (input PDOs). All this can be seen in the SystemManager on the TAB "Process Data". If the mapping which was set by the user on the Process Data tab and which was expected by the slave do not match this Status Code is returned.

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

**4.2.25 Error Code 0x0025****Meaning**

Invalid Output Mapping

**Description**

The process data are described by the configuration (PdoConfig) and PDO assignment (PdoAssign).

PdoConfig: list of actual variables (usually indexes 0x6nnn for inputs and 0x7nnn for outputs). Variables are also called PDO entries. There can be one or several variables with in one list (i.e. within one PDO). The Input PDOs have the index 0x1Amm. The Output PDOs have the index 0x16mm. Example: Slave does only support one or certain PDO combinations but a different setting was made by the user. For a bus coupler the connected terminals differ from the configured terminals in the SystemManager

**Current State (or state change)**

P→S

**Resulting state**

P + E



**Solution**

n/a

**4.2.26 Error Code 0x0026****Meaning**

Inconsistent Settings

**Description**

General settings mismatch

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

**4.2.27 Error Code 0x0027****Meaning**

Freerun not supported

**Description**

n/a

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

**4.2.28 Error Code 0x0028****Meaning**

Synchronization not supported

**Description**

n/a

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

**4.2.29 Error Code 0x0029****Meaning**

Freerun needs 3 Buffer Mode

**Description**

FreeRun mode, SM has to run in 3-buffer mode

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

**4.2.30 Error Code 0x002A****Meaning**

Background Watchdog

**Description**

n/a

**Current State (or state change)**

S, O

**Resulting state**

P + E

**Solution**

n/a

### 4.2.31 Error Code 0x002B

**Meaning**

No Valid Inputs and Outputs

**Description**

n/a

**Current State (or state change)**

O, S→O

**Resulting state**

S + E

**Solution**

n/a

### 4.2.32 Error Code 0x002C

**Meaning**

Fatal Sync Error

**Description**

The hardware interrupt signal (so called Sync signal) generated by the ESC is not generated any more. The master sets and activated the cycle time of the Sync signal during state transition from PREOP to SAFEOP. If a slave was disconnected and reconnected (also due to lost frames or CRC errors) the generation of the SyncSignal may be lost.

**Current State (or state change)**

O

**Resulting state**

S + E

**Solution**

Set master to INIT and back to OP so that the DCs are initialized again

### 4.2.33 Error Code 0x002D

**Meaning**

ana

**Description**

SyncSignal not received: In SAFEOP the slave waits for the first Sync0/Sync1 events before switching to OP, if these events were not received during the SAFEOP to OP-Timeout time the slave refuses the state transition to OP

**Current State (or state change)**

n/a

**Resulting state**

n/a

**Solution**

n/a

## 4.2.34 Error Code 0x0030

**Meaning**

Invalid DC SYNC Configuration

**Description**

Distributed Clock Configuration is invalid due to application requirements

**Current State (or state change)**

O, S→O, P→S

**Resulting state**

P + E, S + E

**Solution**

n/a

## 4.2.35 Error Code 0x0031

**Meaning**

Invalid DC Latch Configuration

**Description**

DC Latch configuration is invalid due to application requirements

**Current State (or state change)**

O, S→O, P→S

**Resulting state**

P + E, S + E

**Solution**

n/a

**4.2.36 Error Code 0x0032****Meaning**

PLL Error

**Description**

Master not synchronized, at least one DC event recieved

**Current State (or state change)**

O, S→O

**Resulting state**

S + E

**Solution**

n/a

**4.2.37 Error Code 0x0033****Meaning**

DC Sync IO Error

**Description**

Multiple Synchronization Errors: At least one SyncSignal was received before. However, the PLL between slave and master is not synchronized any more. This may occur if the master application jitters too much

**Current State (or state change)**

O, S→O

**Resulting state**

S + E

**Solution**

Use specific industrial pc, standard office PCs may have power saving options, graphic accelerateds and other system services which disturb the real-time of the master.

CPU power may be too small for the PLC/NC program.

Increase EtherCAT and PLC/NC cycle time.

Use SyncUnits for the slaves using DCs.

### 4.2.38 Error Code 0x0034

**Meaning**

DC Sync Timeout Error

**Description**

Multiple Synchronization Errors, too much SM events missed

**Current State (or state change)**

O, S→O

**Resulting state**

S + E

**Solution**

n/a

### 4.2.39 Error Code 0x0035

**Meaning**

DC Invalid Sync Cycle Time

**Description**

n/a

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

### 4.2.40 Error Code 0x0036

**Meaning**

DC Sync0 Cycle Time

**Description**

DC Sync0 cycle time does not fit to the application requirements

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

**4.2.41 Error Code 0x0037****Meaning**

DC Sync1 Cycle Time

**Description**

DC Sync1 cycle time does not fit to the application requirements

**Current State (or state change)**

P→S

**Resulting state**

P + E

**Solution**

n/a

**4.2.42 Error Code 0x0041****Meaning**

MBX\_AOE

**Description**

n/a

**Current State (or state change)**

B, P, S, O

**Resulting state**

Current State + E

**Solution**

n/a

**4.2.43 Error Code 0x0042****Meaning**

MBX\_EOE

**Description**

n/a

**Current State (or state change)**

B, P, S, O

**Resulting state**

Current State + E

**Solution**

n/a

**4.2.44 Error Code 0x0043****Meaning**

MBX\_COE

**Description**

n/a

**Current State (or state change)**

B, P, S, O

**Resulting state**

Current State + E

**Solution**

n/a

**4.2.45 Error Code 0x0044****Meaning**

MBX\_FOE

**Description**

n/a

**Current State (or state change)**

B, P, S, O

**Resulting state**

Current State + E



**Solution**

n/a

**4.2.46 Error Code 0x0045****Meaning**

MBX\_SOE

**Description**

n/a

**Current State (or state change)**

B, P, S, O

**Resulting state**

Current State + E

**Solution**

n/a

**4.2.47 Error Code 0x004F****Meaning**

MBX\_VOE

**Description**

n/a

**Current State (or state change)**

B, P, S, O

**Resulting state**

Current State + E

**Solution**

n/a

**4.2.48 Error Code 0x0050****Meaning**

EEPROM No Access

**Description**

EEPROM not assigned to PDI

**Current State (or state change)**

Any

**Resulting state**

Any + E

**Solution**

n/a

**4.2.49 Error Code 0x0051****Meaning**

EEPROM Error

**Description**

EEPROM access error

**Current State (or state change)**

Any

**Resulting state**

Any + E

**Solution**

n/a

**4.2.50 Error Code 0x0060****Meaning**

Slave Requested Locally

**Description**

n/a

**Current State (or state change)**

Any

**Resulting state**

I

**Solution**

n/a

### 4.2.51 Error Code 0x0061

**Meaning**

Device Identification Value updated

**Description**

n/a

**Current State (or state change)**

P

**Resulting state**

P + E

**Solution**

n/a

### 4.2.52 Error Code 0x00F0

**Meaning**

Application Controller available

**Description**

n/a

**Current State (or state change)**

n/a

**Resulting state**

n/a

**Solution**

n/a

## **5 EtherCAT operation - control**

### **5.1 Operation**

In preparation.

## 6 EtherCAT operation - timing

### 6.1 Concept mapping

EtherCAT slaves are directly linked in the System Manager with the process image of a task. This leads to so-called synchronous mapping, i.e. write/read access to the process image takes place alternately through I/O update/fieldbus and task.

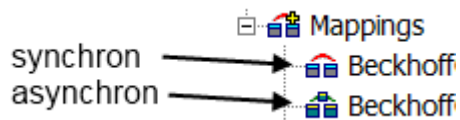


Fig. 237: System Manager icons for synchronous/asynchronous mapping

With asynchronous mapping the two processes “Task” and “Fieldbus Update” access the process image in their respective cycle time.

#### Link effect

If I/Os are linked directly with a task, the corresponding EtherCAT fieldbus update takes place synchronous with this task cycle time. No more than “max. Sync Task” I/O cycles are executed. (EtherCAT master --> advanced settings --> cyclic task).

If devices are not linked with a task, they are controlled via asynchronous mapping based on the slowest available task. This is also the case with automatically set up cross communication, for example in safety devices.

---

#### **i** Watchdog

Check the mapping settings to ensure that timings set up in this way do not exceed the watchdog times.

Watchdog examples:

- I/O watchdog: default 100 ms
  - FSoE watchdog: 100 ms (confirmed communication, cycle times  $\leq 25$  ms recommended with default setting)
-

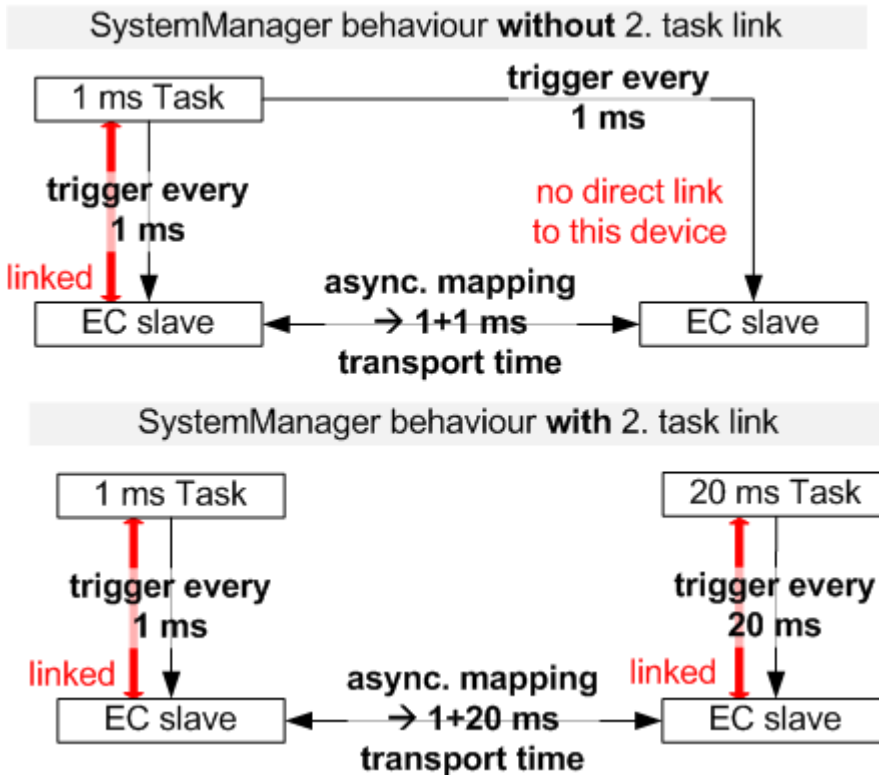


Fig. 238: Automatically set-up asynchronous mapping

## 6.2 Mapping Types and Graphical Display

### 6.2.1 Mappings

There is a list of all process image links underneath the tree entry *Assignments*.

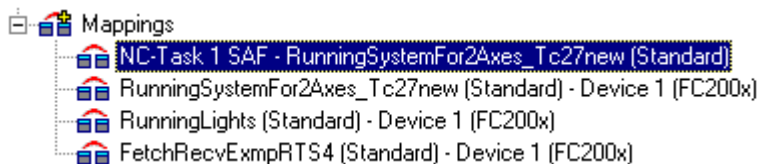


Fig. 239: TC tree: assignments, list of links

On the right-hand side the corresponding dialogue for the selected assignments will appear as shown below.

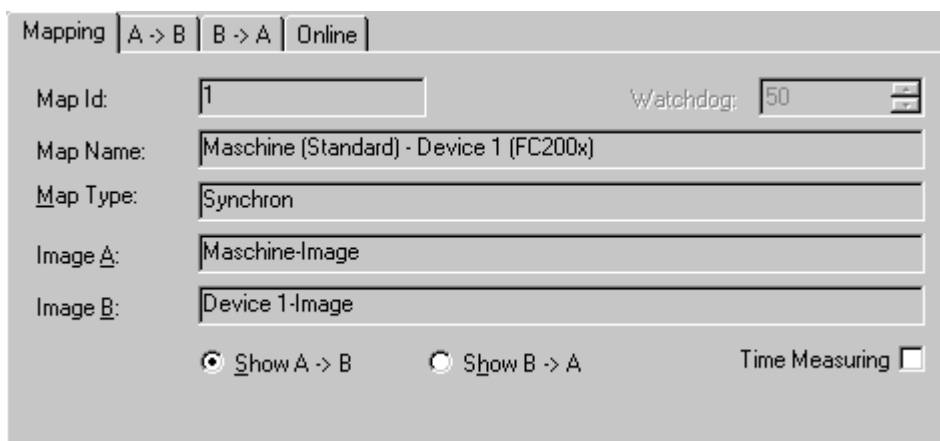


Fig. 240: "Mapping" tab

**Map ID**

Identification number for internal management of the various Assignments (mappings).

**Map Name**

Names both linked process images.

**Map Type**

**Synchronous:** A process image is master, second slave. The master actuates the outputs for writing (e.g. to fieldbus card C1220, ..) and checks that the other side has completed its I/O cycle in order to read the current inputs. The other side has no independent cycle time in this case (Engl. timekeeper).

**Asynchronous:** Is set, e.g. in the event of links from two tasks, or with devices operating on their own cycle times (e.g. COM port) and independently regulates and thereby refreshes inputs and outputs. The exchange of information between two process images thereby takes place in the case of asynchronous mapping on the basis of the three-buffer principle.

For some device types (e.g. the multitasking Profibus card FC310x) a mix of synchronous and asynchronous assignment (mappings) is used. In relation to the device the higher priority task behaves in a synchronous manner while the lower priority task is asynchronous.

**Watchdog**

In the case of asynchronous mapping (see *Assignment Type*) a task could fail to conclude correctly (endless loop), the other task is, however, processed as normal and therefore continues to read old values from the buffers. A maximum lifecycle value is specified to prevent this occurring. Once the value is reached, all buffer values are reset to '0'.

**Timing**

Activates the timing of a synchronous assignment (see *Assignment Type*).

**Process Image A**

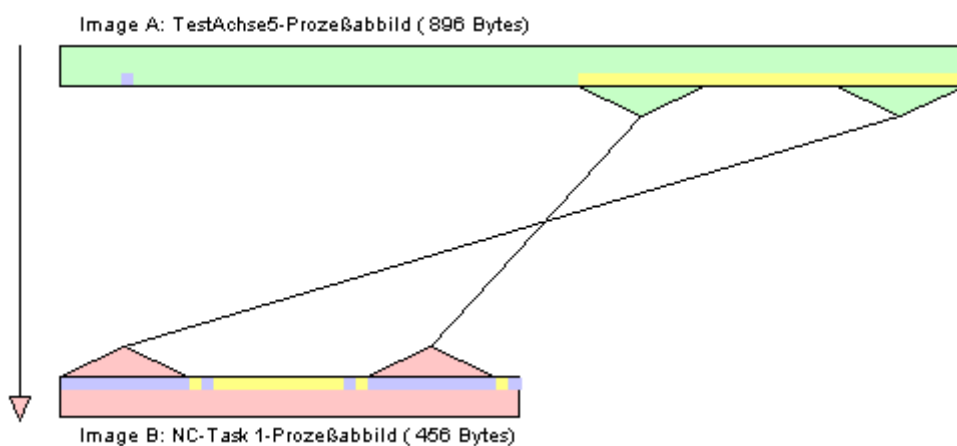
Enter the task name assigned to process image A.

**Process image B**

Enter the task name assigned to process image B.

**Show A -> B and/or B -> A**

Exchanges the displays of both process images in the displayed view.



Process images A, B

The colors green and pink represent process images A and B. The colors yellow and blue are in/outputs of the process image. If you hover with the mouse over the inputs or outputs, a so-called "tooltip" appears with the name of the variable.

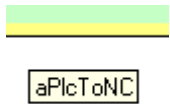


Fig. 241: Variable name

## 6.2.2 Context Menu

A right mouse click displays the context menu via which you can set the zoom resolution of the assignment graphics. The higher the pixel value per byte, the easier it is to assess the links for individual variables.

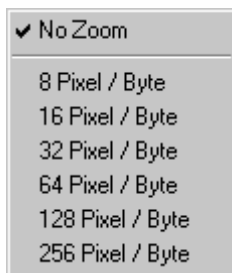


Fig. 242: Zoom factor setting

## 6.2.3 Tab "A -> B" or "B -> A"

Number	Offset above	Offset below	Size
1	0.0	8.1	0.1
2	0.2	8.2	0.1
3	0.5	8.3	0.1
4	0.1	8.5	0.1
5	0.3	8.6	0.1
6	0.6	8.7	0.1
7	1000.0	16.0	4.0
8	5.0	98.0	0.1
9	60.0	86.0	1.0

Fig. 243: Tab "B -> A"

### Number

Current Copy Action Number.

### Offset A

Gives the offset within process image A, from which the copy action is actuated.

### Offset B

Gives the offset within process image B, from which the copy action is actuated.

### Size

Length of the values to be copied from each offset (e.g. one bit would be copied at value 0.1).



### 6.2.4 Online tab

With an active configuration and the system started, you will find here a graphic illustration of the time (in nanoseconds) that is currently required for copying process image A to B or B to A. This dialog has existed since TwinCAT 2.8 with synchronous and asynchronous assignments (cf. "Assignment types" above).

**Asynchronous assignments:**

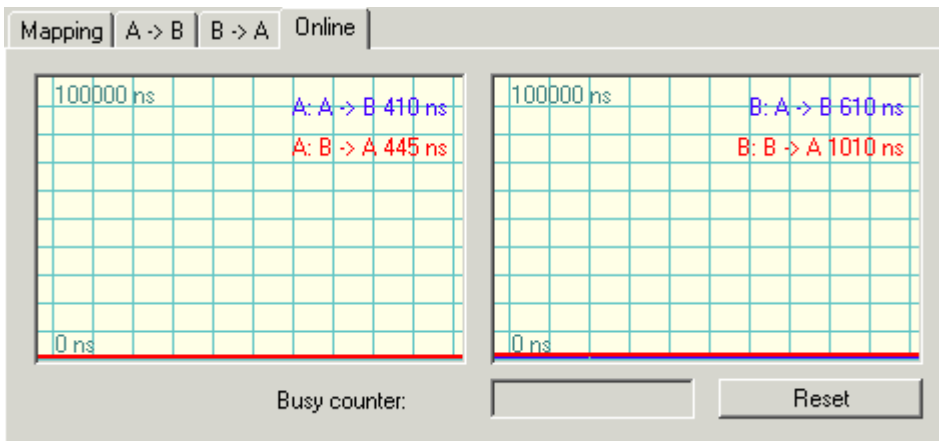


Fig. 244: Online display "asynchronous assignments"

The time shown alongside

**A: A->B** refers to the copying of all the data of this assignment of process image A into the buffer for B.

**A: B->A** refers to the copying of all the data of this assignment of process image A out of the buffer for B.

**B: A->B** refers to the copying of all the data of this assignment of process image B out of the buffer for A.

**B: B->A** refers to the copying of all the data of this assignment of process image B into the buffer for A.

**Synchronous assignments:**

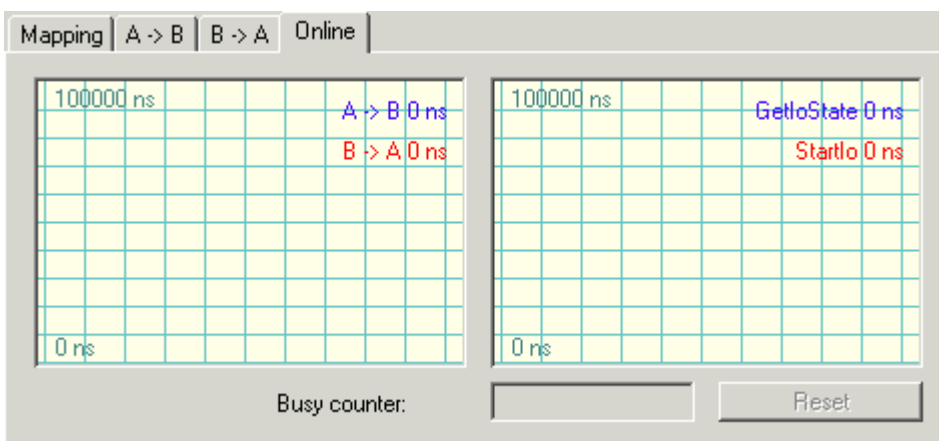


Fig. 245: Online display "synchronous assignments"

The time shown alongside

**A->B** refers to the copying of all the data of this assignment of process image A to B.

**B->A** refers to the copying of all the data of this assignment of process image A from B.

**GetIoState** refers to the duration of the testing function GetIoState(), whose complexity varies depending on the fieldbus card employed.

**StartIo** refers to the duration of the function StartIo(), which starts the bus and whose complexity varies depending on the fieldbus card employed

**Note:**

All times are measured as the difference in the time between the beginning and end of the respective action and can therefore strongly fluctuate in case of interruptions.

Details of the continuously running graphic display of the current online value can be found under: [History display settings](#).

**Busy counter**

In a synchronous assignment of process images the master process image checks whether the slave has completed its I/O cycle and offers new inputs for transmission. If this is not the case the *Busy counter* is incremented by the master and displayed in the field shown above. In case of *asynchronous assignments* the field remains empty.

**Reset**

Resets the *Busy counter* to '0'.

## 7 Distributed Clocks

### 7.1 TwinCAT & time

#### 7.1.1 TwinCAT time sources

The Beckhoff TwinCAT automation suite can analyze several independent time sources. See also the [sample program \[►\\_203\]](#).

Information regarding the architecture:

Name	BIOS motherboard	CPU time	Windows/NT time	TwinCAT/TC time	Distributed clocks/DC time
<b>Description</b>	RTC (RealTime-Clock), battery-operated on the motherboard	CPU counter from the controller hardware, not regulated Initialized by RTC	Local system time of the Windows operating system (NT) Initialized by RTC	Running TwinCAT clock Initialized by Windows	The system returns the start time of the current task cycle. Initialized by TwinCAT.
<b>Data</b>		64 bit resolution: 100 ns <sub>PC Base</sub>	from 1.1.1601 00:00 resolution: 1 ms Scope: structure with year, month, day, hour etc.	from 1.1.1601 00:00 resolution: 100 ns Scope: 64 bit	from 1.1.2000 00:00 resolution: 1 ns Scope: 64 bit
<b>Reference</b>			Local	Local time taking into account the set time zone, i.e. usually UTC	Local time taking into account the set time zone, i.e. usually UTC
<b>PLC format</b>		T_ULARGE_INTEGER	TIMESTRUCT	T_FILETIME	T_DcTime
<b>Call</b>		GetCpuCounter	NT_GetTime()	GetSystemTime()	F_GetActualDcTime() ( <i>ab</i> TwinCAT 2.11) F_GetCurDcTickTime() (= GetSystemTime) F_GetCurDcTaskTime() (from TwinCAT 2.11)
<b>Update</b>		With each call, possibly several times within a PLC cycle		with each base tick (System Manager   BaseTime)	ActualDcTime with each call, possibly several times within a PLC cycle  <i>TickTime</i> with each base tick (System Manager   BaseTime)  <i>TaskTime</i> At the start of the Sync task cycle
<b>Sample application</b>	can be used by the PLC block <i>Nt_SetTimeToRtcTime</i> for correcting the NT time	relative time measurements	Logging, time stamping at operating system level	High-precision, relative time-based tasks within one or across several task cycles	- High-precision, relative time-based tasks within the EtherCAT system - Definitive reference to the global time through external synchronization possible
<b>Manipulation options</b>			- It can be changed to the current RTC time by the PLC function block <i>Nt_SetTimeToRtcTime</i> ; this also triggers a correction of the RTC time  ATTENTION Use of this function in conjunction with EtherCAT distributed clocks systems is not recommended.  - Synchronization at network level (SNTP, NTP)		Synchronization with external reference time from TwinCAT 2.11

Table 1: time architecture, required libraries: *TcEtherCAT.lib*, *TcUtilities.lib*, *TcSystem.lib*

### Application scenario 1: Local control system without forced synchronization via the network

The local Windows clock is free-running and can be coupled to the RTC via *Nt\_SetTimeToRtcTime*. This option is not recommended when using distributed clocks components!

We recommend using the TC or DC clock if absolute time references are required.

Ideally, the DC time should be coupled to an external reference time via suitable EtherCAT components.

## Application scenario 2: Local control system with forced synchronization via the network

The local Windows clock is cyclically synchronized with the world time through a network clock/server/Internet time server.

Coupling of the Windows clock to the RTC through `Nt_SetTimeToRtcTime` is not recommended.

We recommend using the TC or DC clock. The reference to the absolute time can be established via the application through offset calculation at the NT time.

Ideally, the DC time should be coupled to an external reference time via suitable EtherCAT components.

## Application scenario 3: Local control system with external reference time via EtherCAT

A frequency- and phase-synchronized analog time is available through coupling of the DC time to an external time source (GPS, radio clock, PTP/IEEE1588, EtherCAT). The NT and TC time are not required in the application.

### ● Common time synchronization

**i** The usual time synchronization at operating system level works in discrete intervals ranging from several seconds to days. In the synchronization case this leads to an erratic/unsteady change of the subordinate time! The usual network synchronizations (SNTP, NTP and similar) or even `Nt_SetTimeToRtcTime` are affected by this. The application must be able to anticipate these sudden changes in the "absolute" time.

Beckhoff only integrates continuous synchronization techniques in EtherCAT.

## 7.1.2 Internal and external EtherCAT synchronization

### 7.1.2.1 General

In a machine control with distributed components (I/O, drives, several masters) it may be useful for the components to operate with a close time link to each other. The components must therefore have a local "time", to which the component (e.g. an I/O terminal) has access at all times.

Associated requirements may include:

1. Several outputs in a control system have to be set simultaneously, irrespective of when the respective station receives the output data.
2. Drives/axes in a control system must read their axis positions synchronized, irrespective of the topology or cycle time.

Both requirements necessitate a synchronization mechanism between the local times of the components of a control system.

1. If inputs affect the control system, the (absolute) time must be recorded. This can be helpful for subsequent analysis, if such an analysis is required for determining the sequence of events in functional chains.  
This means that time running in the components must be coupled to a globally valid time, e.g. Greenwich world time or a network clock.
2. tasks on different controllers should run synchronous and without phase shift.

The terms "close temporal reference" or "simultaneous" can be interpreted depending on requirements: for a "simultaneity" in the 10 ms range a serial communication structure may be adequate, while in other ranges 100 ns or less are required.

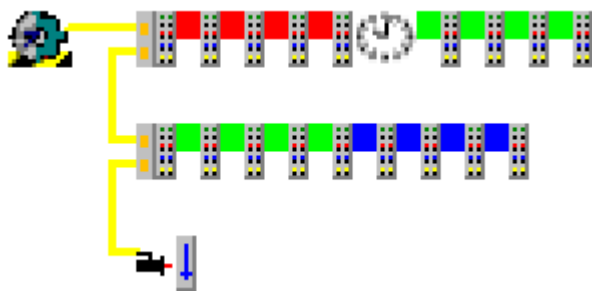


Fig. 246: Simple I/O topology

Fig. *Simple I/O topology* shows a simple EtherCAT topology consisting of a master, several I/Os and an axis. A local time is to be applied in different components. The tasks:

- synchronization of local clocks
- coupling to a higher-level reference time
- task synchronization

are discussed below.

### 7.1.2.2 Requirement 1 + 2: synchronization

In an EtherCAT system the distributed clocks concept (DC) is used for synchronization of local clocks in the EtherCAT components.

#### Synchronization of local EtherCAT devices

General:

- 1 ns time resolution corresponds to 1 digit, scope of 64 bits corresponds to approx. 584 years
- The EtherCAT master must keep the distributed clocks synchronous within the system accuracy (EtherCAT: <100 ns) using synchronization datagrams.
- Not all EtherCAT devices have to support this feature. If a slave does not support this concept, the master will not include it in the synchronization. If the EtherCAT master does not support this feature, DC is also ineffective in all slaves.
- such a clock also runs in the EtherCAT master, in this case software-based.
- in the system *one* of the existing clocks is selected as reference clock and used for synchronizing all other clocks. This reference clock is usually one of the EtherCAT slave clocks, not the EtherCAT master clock. The clock of the first EtherCAT slave in the topology that supports distributed clocks is usually automatically selected as reference clock.
- a distinction is made between
  - the EtherCAT master (the software that “manages” the EtherCAT slaves with Ethernet frames) and the EtherCAT slaves managed by it.
  - the reference clock, which is usually located in the first DC slave, and the slave clocks whose time is based on it, including the clock in the EtherCAT master.

Master:

- During the system start phase the EtherCAT master must set the local time of the reference clock and the other slave clocks to the current time and subsequently minimize deviations between the clocks through cyclic synchronization datagrams.
- in the event of topology changes the EtherCAT master must re-synchronize the clocks accordingly
- not all EtherCAT masters support this procedure
- the EtherCAT master in the Beckhoff TwinCAT automation suite fully supports distributed clocks.

Slave:

- Due to the high precision required this local clock is implemented in hardware (ASIC, FPGA).
- Distributed clocks are managed in the EtherCAT slave controller (ESC) in registers 0x0900 - 0x09FF. Specifically, the local synchronized time runs in the 8 byte from 0x0910.

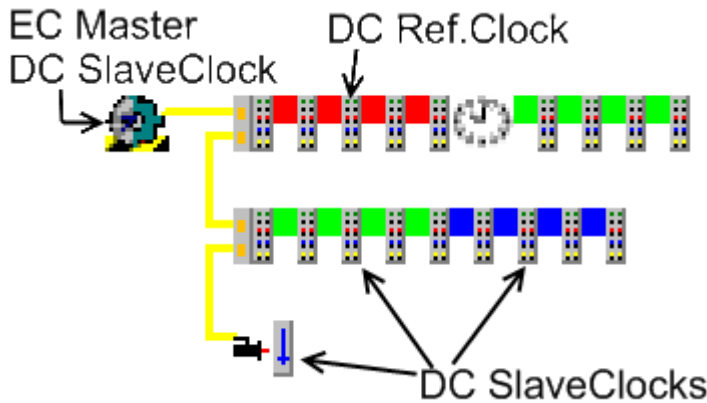


Fig. 247: mapping of DC to the topology

In Fig. *Mapping of DC to the topology* the 3<sup>rd</sup> EtherCAT slave was selected as DC reference clock as a sample. The local time of this slave is now used to adjust all other distributed clocks, i.e. all other EtherCAT slaves and the clock of the EtherCAT master. This is achieved through synchronization datagrams, which the EtherCAT master sends cyclically.

**● TwinCAT clock readjustment**

**i** The TwinCAT clock that determines the real-time must track the DC ReferenceClock to ensure that the PLC/NC tasks in the real-time context of the controller run synchronous with the distributed clocks. If more than one EtherCAT system is used on a control system, only one of these systems can provide the ReferenceClock for TwinCAT. The other EtherCAT systems have then to follow the TwinCAT clock.

See notes in section “Coupling of EtherCAT systems [▶ 154]”

This procedure ensures that all devices supporting DC always have local access to a time that is identical (within the DC synchronization precision) in all devices.

The system now operates based on the time base of the selected DC reference clock and its local clock generator/quartz with  $T_{DC}$ . Due to production tolerances this time base is rarely the same as the official sidereal time/coordinated world time UTC  $T_{UTC}$  or another reference time. This means that  $1\text{ ms}_{UTC}$  is never exactly  $1\text{ ms}_{DC}$ ,  $T_{DC} \neq T_{UTC}$ . Over longer periods also drift processes may also change the ratio. As long as DC is used for relative processes within the EtherCAT system, the deviation from the UTC is irrelevant. However, if the DC time is to be used for data logging with a global time base, for sample, the DC time base must be synchronized with the  $T_{UTC}$  time base. This is described in section Requirement 3.

**7.1.2.3 Requirement 3: higher-level global time, absolute time**

If the time base  $T_{DC}$  is to be adjusted based on a higher-level time base, the time base and the associated procedure must be selected. Generally common synchronization protocols are used for the synchronization. Samples for time sources and synchronization procedures are listed below.

- Sources: UTC world time, network time, adjacent control system, radio clocks (in Central Europe: DCF77)
- Procedures: GPS, radio clocks, NTP (NetworkTimeProtocol), SNTP (Simple NTP), PTP (IEEE1588), distributed clocks DC

The following synchronization precisions can be achieved (depending on the hardware)

- NTP/SNTP: ms range
- PTP: < 1  $\mu$ s
- DC: < 100 ns

The following two control aims must be achieved:

- The frequency of the subordinate time base must be adjusted to the higher-level time base.
- Any offset between two absolute times does not have necessarily to be controlled to 0. It is sufficient for it to be announced and kept constant. The maximum offset adjustment is  $\pm\frac{1}{2}$  cycle time.

### External EtherCAT synchronization

**i** External synchronization sources (e.g. EL6688, EL6692) can only be used from TwinCAT 2.11 used. In older versions of TwinCAT such EtherCAT slaves have no meaningful function.

If a higher-level master clock is integrated in an EtherCAT system, a special EtherCAT device is generally used for the physical connection. The device monitors both time bases and is therefore able to determine the time difference.

Please refer to [www.beckhoff.de](http://www.beckhoff.de) for suitable products currently available for this purpose.

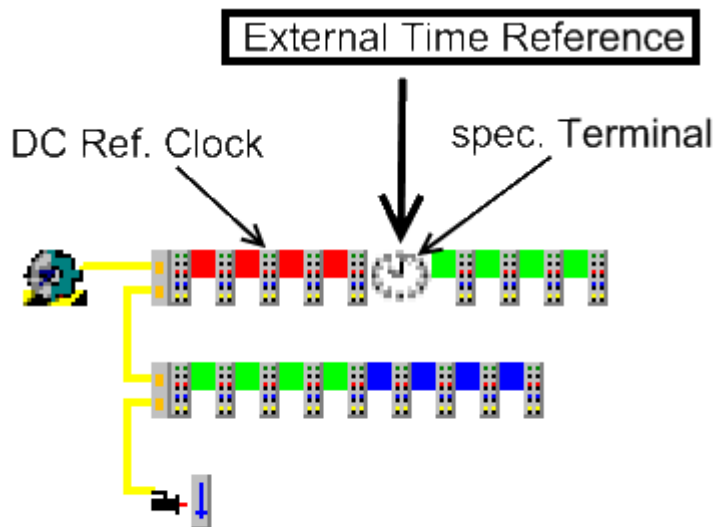


Fig. 248: EtherCAT topology with external reference clock

The different time bases can be arranged hierarchically, so that at the start of the respective system the current absolute time is taken from the subordinate system. If necessary top-down synchronization is used, if external time bases or DC components are present in the system.

### Readjustment of local time vs. higher-level absolute time

For the purpose of synchronization the local DC time is not adjusted based on the higher-level absolute time, but only to a constant offset. This offset is made available to the user as a process data. The offset is corrected by  $\pm\frac{1}{2}$  cycle time to ensure both tasks run in phase.

- When TwinCAT starts the EtherCAT master, the local DC system in the slaves is started and synchronized immediately.
- However, an external reference slave such as EL6688 (IEEE1588 PTP) takes a few seconds before it can supply a reference time that is synchronized with the higher-level clock.
- As soon as the external reference time is available, the offset to the local time is calculated and corrected by  $\pm\frac{1}{2}$  cycle time to ensure that both tasks run in phase, and the EtherCAT master Info Data are made available to the user for reconciliation with the local time values.
- From this time the offset is kept constant, depending on the selected control direction.



### 7.1.2.4 TwinCAT system behavior

#### External reference clock outage

If the external reference clock signal fails, both time bases will naturally drift apart again. Once the signal is available again, the system will once again be controlled based on the previous offset values.

TwinCAT can start without external clock signal. In this case the offset is calculated and maintained as described above, as soon as a stable external reference clock signal is received.

### 7.1.2.5 Settings in TwinCAT 2.11

External synchronization via EtherCAT is supported from TwinCAT 2.11. The synchronization direction can be set in the associated dialog.

#### Distributed clock timing settings

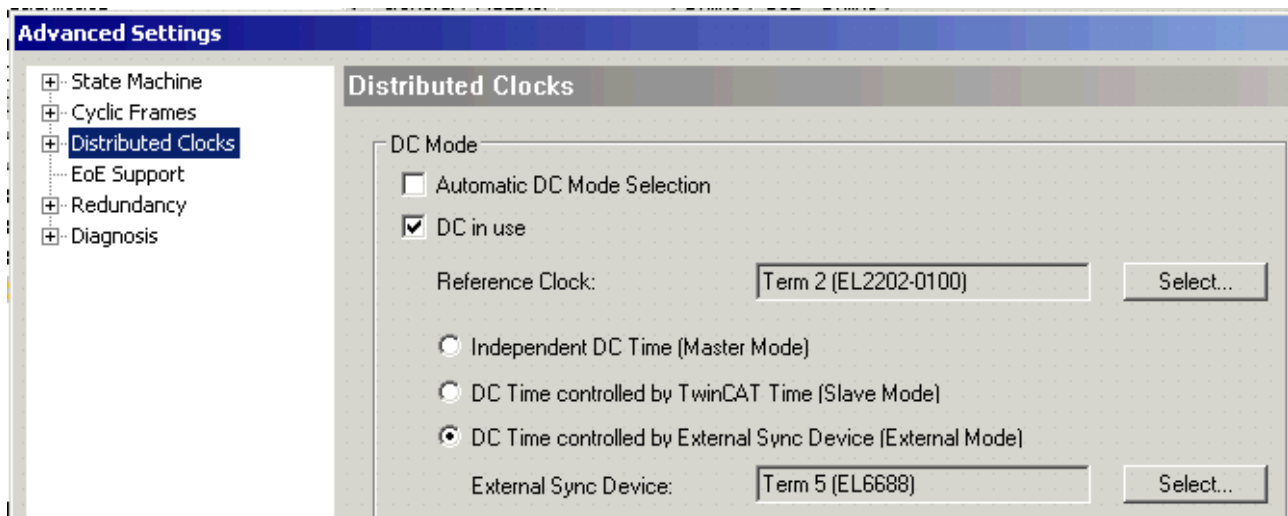


Fig. 249: TwinCAT 2.11 distributed clock settings - Sample for EL6688 in PTP slave mode as time reference for the local EtherCAT system

Fig. *Synchronization direction* shows the synchronization direction for the respective synchronization type, i.e. the source on which the synchronization time is based.

- **Independent DC Time (A):**  
One of the EL terminals (generally the first terminal supporting distributed clocks (DC)) is the reference clock to which all other DC terminals are adjusted. Selection of the reference clock in the dialog above.
- **DC Time controlled by TwinCAT (B):**  
The DC reference clock is adjusted to the local TwinCAT time. This setting is used in cases where several EtherCAT systems with distributed clocks function are operated in the same control system. This tracking mode is less accurate.  
If high accuracy is required the external CU2508 EtherCAT distributor must be used.  
Note: In the subordinate EtherCAT system a device without firmware intelligence (e.g. an EK1100 coupler) must be set as ReferenceClock.  
Please refer to section "[Coupling of EtherCAT systems](#) [[▶ 154](#)]".
- **DC Time controlled by External Sync Device (C):**  
If the EtherCAT system is to be adjusted to a higher-level clock, the external sync device can be selected here.  
Please refer to section "[External synchronization](#)" [[▶ 250](#)].

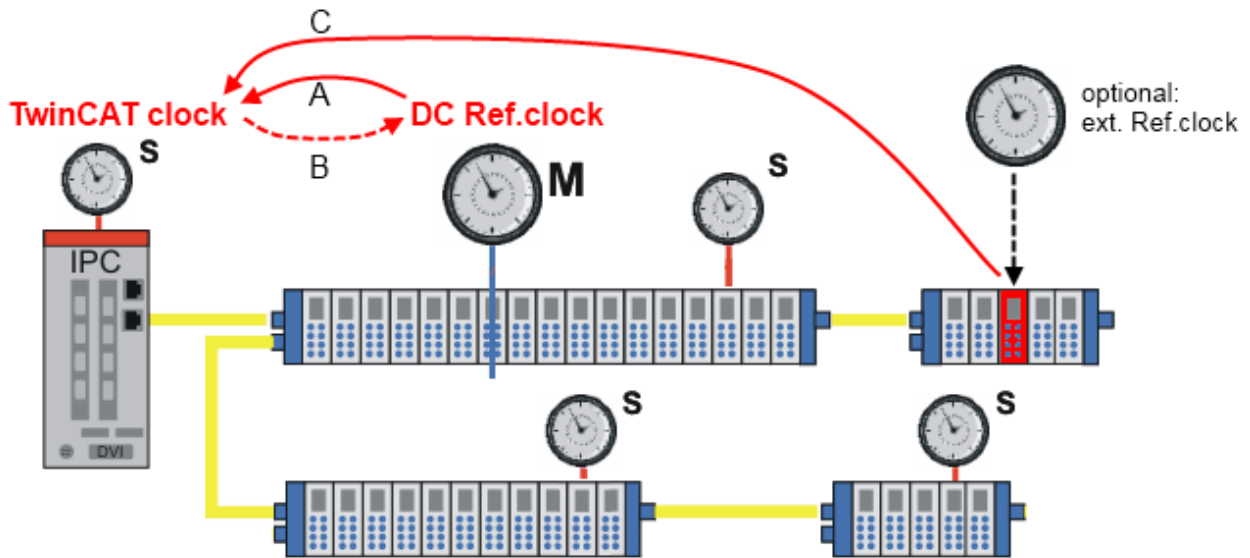


Fig. 250: synchronization direction

**Process data settings**

TwinCAT 2.11 can display the current offsets in [ns] in the EtherCAT master info data.

- These offsets are calculated once after EtherCAT has started.
- The synchronization control keeps these offsets constant.
- If local DC time values in the synchronized EtherCAT system are to be related to the absolute reference from the higher-level EtherCAT system (e.g. from EL1252 timestamp terminals), the user must adjust this offset with each local timestamp.

Sample:  $t_{EL1252 \text{ timestamp channel 1, absolute time}} = t_{EL1252 \text{ timestamp channel 1, local DC time}} + t_{ExtToDcOffset} + t_{TcToDcOffset}$

- [-] State Machine
  - Master Settings**
  - Slave Settings
- [-] Cyclic Frames
  - Sync Tasks
  - Process Image
  - VLAN Tagging
- [-] Distributed Clocks
- [-] EoE Support
- [-] Redundancy
- [-] Emergency
- [-] Diagnosis

### Master Settings

Startup State

'INIT'

'PRE-OP'

'SAFE-OP'

'OP'

Stay at 'PRE-OP' until Sync Task started

Run-Time Behaviour

Log Topology Changes

Log CRC Counters

Log Error Counters (only for testing)

Relnit after Communication Error

Show Input Toggle Information

Info Data

Enable

Include Device Id

Include Ads NetId

Include Cfg Slave Count

Include DC Time Offsets

Fig. 251: display current offsets

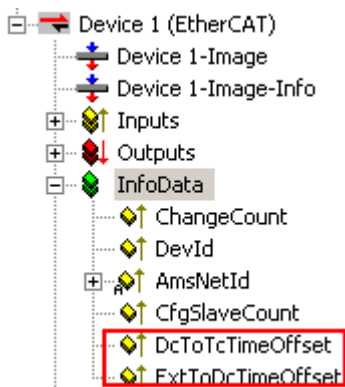


Fig. 252: Current offsets

### 7.1.3 Sample programs

#### Using the sample programs

This document contains sample applications of our products for certain areas of application. The application notes provided here are based on typical features of our products and only serve as examples. The notes contained in this document explicitly do not refer to specific applications. The customer is therefore responsible for assessing and deciding whether the product is suitable for a particular application. We accept no responsibility for the completeness and correctness of the source code contained in this document. We reserve the right to modify the content of this document at any time and accept no responsibility for errors and missing information.

#### Sample 1: Display and evaluations of the different times in TwinCAT

The sample program determines several independent local times in a TwinCAT system under Windows XPe, calculates current deviations and converts them into different representations. The function *Nt\_SetTimeToRtcTime* can be activated for testing purposes.

Notes:

- Cycle time used: 1 ms
- Determined times:
  - Local Windows NT time (shown in the taskbar)
  - Local TwinCAT time
  - Distributed Clocks time
- The example uses EtherCAT distributed clocks terminals for determining the distributed clocks time (DC).
- The individual conversion, particularly the cyclic string representations, require significant computing time. A CX1000 platform or above is recommended for testing the sample program.

Please follow the general instructions for EtherCAT synchronization.

#### Starting the sample program

The application samples have been tested with a test configuration and are described accordingly. Certain deviations when setting up actual applications are possible.

The following hardware and software were used for the test configuration:

- TwinCAT master PC with Windows XP Professional SP 3, TwinCAT version 2.10 (Build 1330) and INTEL PRO/100 VE Ethernet adapter
- Beckhoff EK1100 EtherCAT coupler, EL2202-0100, EL2252 and EL9011 terminals

### 7.1.3.1 Sample program TwinCAT 2

 <https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/2469153803.zip>

#### Preparations for starting the program

- After clicking the Download button, save the zip file locally on your hard disk, and unzip the \*.tsm (configuration) and the \*.pro (PLC program) files into a temporary working folder.
- The \*.pro file can be opened by double click or by the TwinCAT PLC Control application with menu selection "File/ Open". The \*.tsm file is provided for the TwinCAT System Manager (to review or overtake configurations).
- This example requires a PLC control with a terminal EL2202-0100. You can use either an embedded PC that has the terminal placed on the right or an IPC with an EtherCAT link of an e.g. RJ-45 connector to the EK1100 coupler with the terminal (e.g. C6915 + EK1100 + EL2202-0100).
- Within this example it's not necessary to connect the terminals outputs (as there will be the DC operation mode used only). However the Sync Master needs a link to a variable. The external "bDummyOut" variable is designated for that, so it has to be linked with one of the both channels of the terminal.

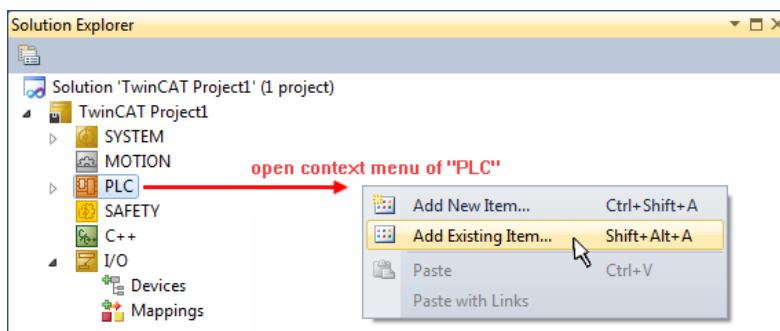
Please continue with further descriptions in section [TwinCAT Quickstart, TwinCAT 2](#) [▶ 66].

### 7.1.3.2 Sample program TwinCAT 3

 <https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/3722257291.zip>

#### Preparation to start the sample program (tpzip file/ TwinCAT 3)

- After clicking the Download button, save the zip file locally on your hard disk, and unzip the \*.tpzip - archive file into a temporary working folder.
- Create a new TwinCAT project as described in section: [TwinCAT Quickstart, TwinCAT 3, Startup](#) [▶ 76]
- Open the context menu of "PLC" within the "Solutionexplorer" and select "Add Existing Item..."



- Select the beforehand unpacked .tpzip file (sample program).
- This example requires a PLC control with a terminal EL2202-0100. You can use either an embedded PC that has the terminal placed on the right or an IPC with an EtherCAT link of an e.g. RJ-45 connector to the EK1100 coupler with the terminal (e.g. C6915 + EK1100 + EL2202-0100).
- Within this example it's not necessary to connect the terminals outputs (as there will be the DC operation mode used only). However the Sync Master needs a link to a variable. The external "bDummyOut" variable is designated for that, so it has to be linked with one of the both channels of the terminal.

Also see more hints in section:

[Commissioning, TwinCAT Quickstart, TwinCAT 3, Startup](#) [▶ 76].

## 7.2 Basic principles

### 7.2.1 EtherCAT Distributed Clocks

An introduction into distributed clock technology as a feature of the EtherCAT protocol is provided below. This section provides an overview of the main aspects from a user perspective, which should be sufficient for standard EtherCAT applications. The subsequent section provides further details and more detailed descriptions for interested users. This information is not essential for standard operation of an EtherCAT slave.

---

#### ● Contents of this documentation

**i** This introduction is limited to a description of the functions that are relevant for the user. Further and more detailed information about EtherCAT in general and distributed clocks in particular can be found under <http://www.ethercat.org/>.

The main terminology is shown in italics below.

---

The following page comprises the sections

- [The principle of distributed clocks \[► 205\]](#)
- [Optional additional ESC functions \[► 208\]](#)
- Application of distributed clocks and synchronicity with the control system in the PC

#### The principle of distributed clocks

In EtherCAT terminology the term "distributed clocks" refers to a logical network of distributed clocks. By using distributed clocks the EtherCAT real-time Ethernet protocol is able to synchronize the time in all local bus devices within a very narrow tolerance range. If an *EtherCAT slave* supports distributed clock functionality it contains its own clock that initially operates locally after switch-on, based on an independent clock generator within the EtherCAT slave (quartz, oscillator, ...). In the EtherCAT strand there is a selected EtherCAT slave that represents the *reference clock* (M, see Fig. *Distributed Clocks in the EtherCAT System*), to which the slave clocks (S) of the other devices and the controller synchronize. The reference clock therefore represents the *system time*. Adjustment and synchronization are handled automatically and consecutively by the *EtherCAT master* provided it supports distributed clock functionality, such as the Beckhoff TwinCAT EtherCAT master. To this end the EtherCAT master sends a special *EtherCAT datagram* at short intervals (with sufficient frequency to ensure that the slave clocks remain synchronized within the specified limits), in which the EtherCAT slave with the reference clock enters its current time. This information is then read from the same datagram by all other EtherCAT slaves featuring a slave clock. Due to the ring structure of EtherCAT this is possible if the reference clock is topologically located *before* all other slave clocks. The EtherCAT master therefore selects the first distributed clock-capable EtherCAT slave as reference clock.

An EtherCAT configuration therefore features an EtherCAT master that operates and manages the bus with the connected EtherCAT slaves. *One* of these EtherCAT slaves contains the reference clock, all other EtherCAT devices (i.e. including the EtherCAT master) represent slave clocks.

The *EtherCAT slave controller (ESC)* handles the EtherCAT communication and in particular the distributed clock functionality in an EtherCAT slave. This is an electronic component (chip) such as an ASIC or reprogrammable FPGA or similar. Each EtherCAT slave has such an ESC to ensure that cyclical and acyclical process data can be exchanged between master and slave via the EtherCAT fieldbus. This ESC can handle simple functions such as digital inputs and outputs directly, or it can be connected to a further processor in the EtherCAT slave via serial/parallel interfaces for handling more complex tasks such as drive control. In particular the ESC manages the local distributed clock functionality with the associated tasks, if the EtherCAT slave is required to support this feature.

The schematic of an EtherCAT slave is illustrated in Fig. *Example for ESC functionality and embedding in a typical EtherCAT slave*. In addition the embedding of the ESC in the latter with a selection of its basic functions.

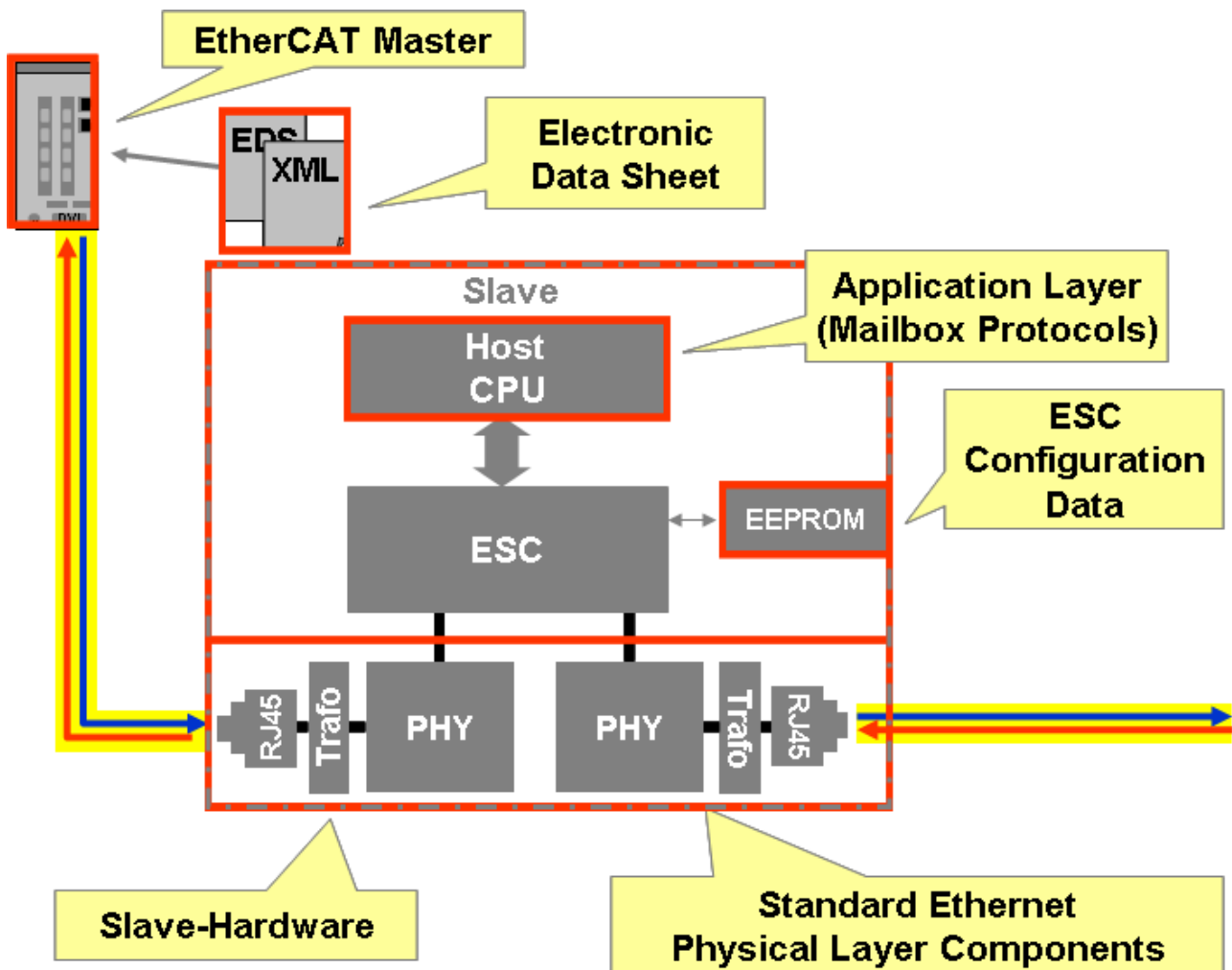


Fig. 253: Example for ESC functionality and embedding in a typical EtherCAT slave

From the RJ45 socket the electrical signals are transferred to the PHY (physical interface) via the transformer. It extracts the user data from the coded Ethernet signal and transfers them to the ESC for processing. The EtherCAT telegram is then relayed with minimum delay (due to dynamic processing) to the next EtherCAT slave via the PHY and the socket. The ESC automatically parameterizes itself with configuration data from an EEPROM when the slave starts up. If a further CPU exists in the slave, the slave can communicate with it via interfaces.

The distributed clocks unit of the ESC in the full configuration offers the following features (dependent on the device implementation):

- Clock synchronization between the EtherCAT slaves and the master
- Synchronous generation of output signals (Sync signals)
- Synchronous reading of input signals
- Precise time stamping of input signals (latch signals)
- Generation of synchronous interrupts

In Fig. *Distributed Clocks in the EtherCAT-System*, the reference clock (M) is the first slave in the EtherCAT strand with EtherCAT functionality after the EtherCAT master IPC. Since each slave causes a small delay – both in the device (S) and in the transmission link – in both directions, the run times ( $\Delta t$ ) between the reference clock and the respective slave clock must be taken into account for synchronization of the slave clocks. It is therefore not appropriate to simply copy the local time of the reference clock to all downstream slave clocks. For each slave a separate offset value should be calculated as a function of several parameters.

For measuring the offset times the EtherCAT master sends a broadcast read datagram to a special address in all ESCs during the startup phase that causes each slave to record the time when the telegram is received (based on its local clock) in both directions. The master then reads the stored times as a basis for the

calculation. These measuring cycles take place several times for all EtherCAT slaves. This enables the EtherCAT master to create a very precise map of the topology in relation to the frame delays between the EtherCAT slaves.

The actions described above ensure synchronous operation of all distributed clocks in an EtherCAT network, e.g. in a production plant, and enable relative times to be specified with high precision within the application. The absolute reference to global reality is provided via the *master clock*, which is usually based on a global world time format (DCF77, GPS, Internet time server, ...) or another time that is designated as valid across the system (network server, PC clock, BIOS clock, IEEE1588, ...). A Beckhoff TwinCAT EtherCAT master starts up using the local PC clock as master clock in order to initialize the reference clock. During system startup and subsequent operation the reference clock (M) can be readjusted based on a master clock, either via direct contact between master clock and control system (e.g. network server) or through feeding into a special EtherCAT slave (e.g. Beckhoff EL6692, EtherCAT bridge terminal).

This higher-level master clock does not violate the primacy of the reference clock, since short-term synchronization in the sub-millisecond range, which is critical for real-time control, is solely determined by the reference clock, while synchronization of the complete EtherCAT application, including the control PC with the master clock, is based on longer intervals.

The following effects must be taken into account by the distributed clock control in the EtherCAT master:

- Offset compensation of each slave relative to the reference clock. After system startup the local clocks may start with different start values.
- Offset compensation of the reference clock relative to the master clock. To be taken into account during system start-up.
- Propagation delay measurement/Measurement of the offset times - depending on the number of devices, cable lengths, dynamic changes in the configuration, etc.
- Drift compensation/Drift correction - Each slave clock usually has its own source (quartz, PLL, ...), which means that offset times do not remain constant over a prolonged period (minutes, days). Drift correction deals with this irregularity.

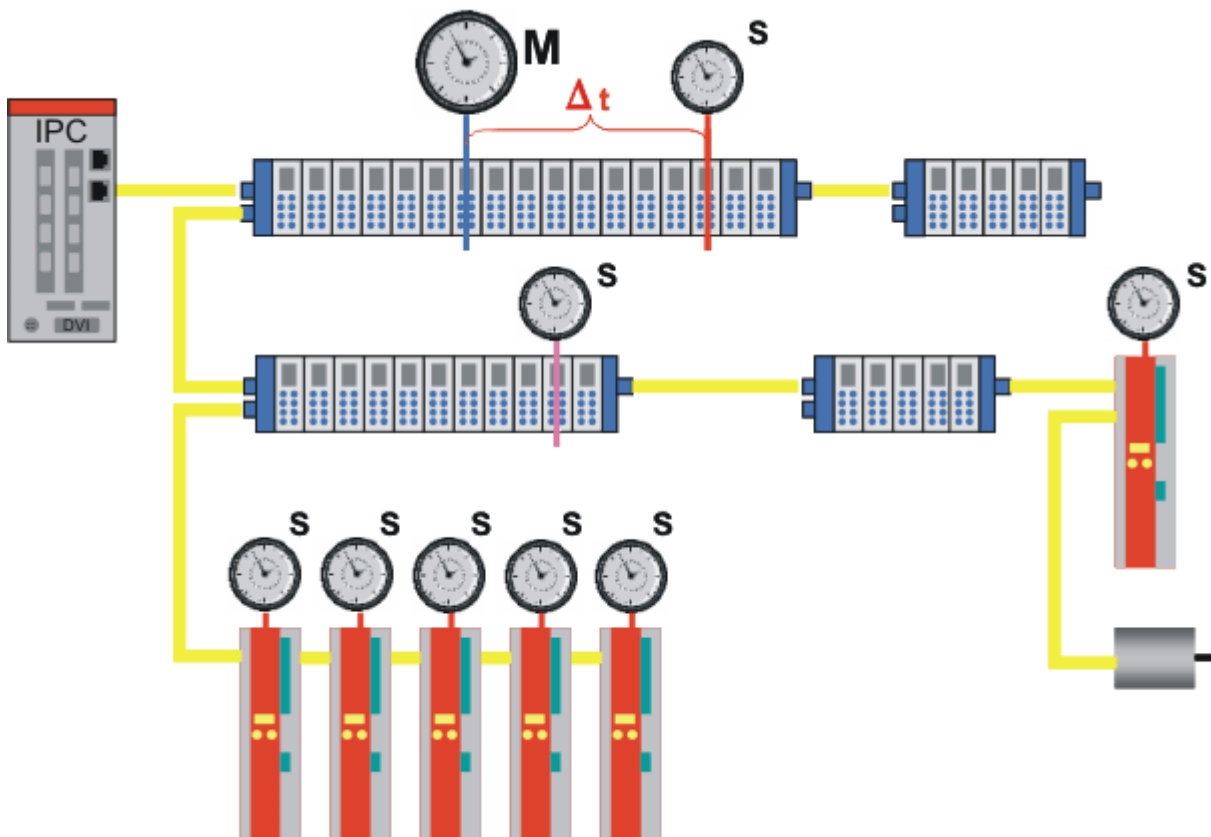


Fig. 254: Distributed clocks in the EtherCAT system

If an EtherCAT slave is no longer supplied with synchronization datagrams, because of an interruption of the bus, it can still fulfil all tasks via its local clock. Distributed clock control is jerk-free, both during normal operation and when EtherCAT traffic is reinstated.

## Summary of technical data

The distributed clock function in the EtherCAT slave controller (ESC) has the following characteristics:

- Unit  $1\text{ ns}$
- Universal zero point  $1.1.2000\ 00:00$
- Scope up to  $64\text{ bit}$  (sufficient for 584 years). However, some EtherCAT slaves only support a 32 bit scope, i.e. the register overflows locally after approx. 4.2 seconds and starts again at 0.
- The EtherCAT master automatically synchronizes each local clock with the reference clock in the EtherCAT system with a precision of  $< 100\text{ ns}$ , irrespective of the distance between individual EtherCAT slaves
- When the EtherCAT system starts up the EtherCAT master usually takes the current time from a master clock, e.g. the hardware-based BIOS clock of its own PC, thus establishing the time reference to the current world time. This time is loaded into the selected reference clock when EtherCAT starts up, which then usually automatically maintains the time based on its local clock generator. Continuous synchronization of the reference clock with the master clock is possible, if required.
- The effective step size of the local clock is generally  $10\text{ ns}$ . The remaining digit ("ones") is used for controlling the distributed clocks.

Up to now the distributed clocks function in the ESC was considered without interaction with the environment. Based on this local/global time additional functions can now be realized in the EtherCAT slave.

## Optional additional ESC functions

The distributed clock time, which is synchronized with high precision, is used by the ESC for responding to specifiable times or signals from outside the ESC via a capture/compare unit. The ESC characteristics are defined by the configuration of the EtherCAT slave during startup and can usually not be changed by the user.

### Action based on specified time: Compare - Sync0/1

The distributed clock unit in the ESC usually features 2 interrupts that can be triggered time-controlled. These interrupts are referred to as *SYNC0* and *SYNC1*. In this case the compare unit in the ESC would be active: If the local distributed clock time matches a user-defined standard time the ESC triggers an interrupt and the associated processes. The standard time can be set *once*, which leads to a *one-off* action in the ESC. This option is used in Beckhoff timestamp terminals, for example.

The ESC may also automatically load new default values, which in this case leads to a cyclical sequence of ESC actions. This option is used for Beckhoff oversampling terminals, for example. The configuration data loaded on ESC start-up from a slave EEPROM is parameterizable, for example, can be used to determine which action an ESC will carry out if a *SYNC0*/*SYNC1* signal is encountered. For example, it can write output data, read input data, or initiate communication with a connected microcontroller.

### Reaction to an external signal: Capture - Latch 0/1

If an ESC is configured accordingly it can store the current local time if an external event occurs, i.e. it can place it into a buffer without delay using a capture unit. Examples for such external events are: arrival of the EtherCAT frame, end of the EtherCAT frame, edge on a dedicated pin of the ESC, communication with a connected microcontroller, and a wide range of other options.

### Connection to an external logic - SPI/ $\mu$ C parallel/IO/IRQ

An ESC is to be used not only as a stand-alone unit, but also has interfaces for communicating with other electronic units. These could be, for example, a controller that controls a power drive or the evaluation electronics of a rotary encoder; see Fig. *Example for ESC functionality and embedding in a typical EtherCAT slave*. Communication via these interfaces can also take place under distributed clocks control. The position query to a rotary encoder electronics, for example, is thus chronologically equidistant in the ns range.



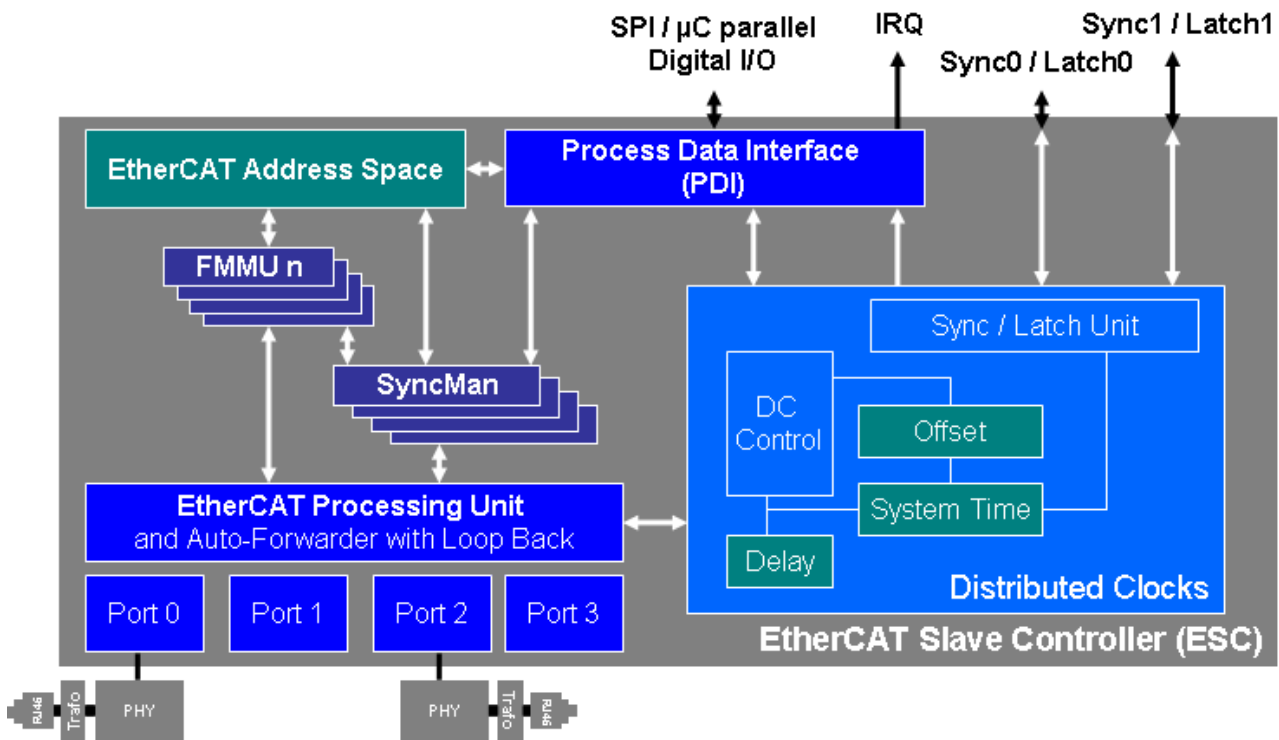


Fig. 255: Interfaces of the distributed clock unit

Fig. *Interfaces of the Distributed Clocks Unit* shows a schematic of the distributed clocks unit with its interfaces and the interaction with the EtherCAT bus.

**Application of distributed clocks and synchronicity with the control system in the PC**

A distributed clock network ensures that all EtherCAT slaves supporting this feature are operated synchronously with a tolerance of < 100 ns. An application with input channels will be used as an example to illustrate this principle, followed by an application involving output channels.

The following example involving three Beckhoff EL1202-0010 devices illustrates synchronous generation of SYNC signals [▶ 208] for simultaneous reading of the inputs. Other ESC functions (latch signals [▶ 208], connection of external logic) are described in detail in the respective slave documentation.

## Distributed clocks with input channels

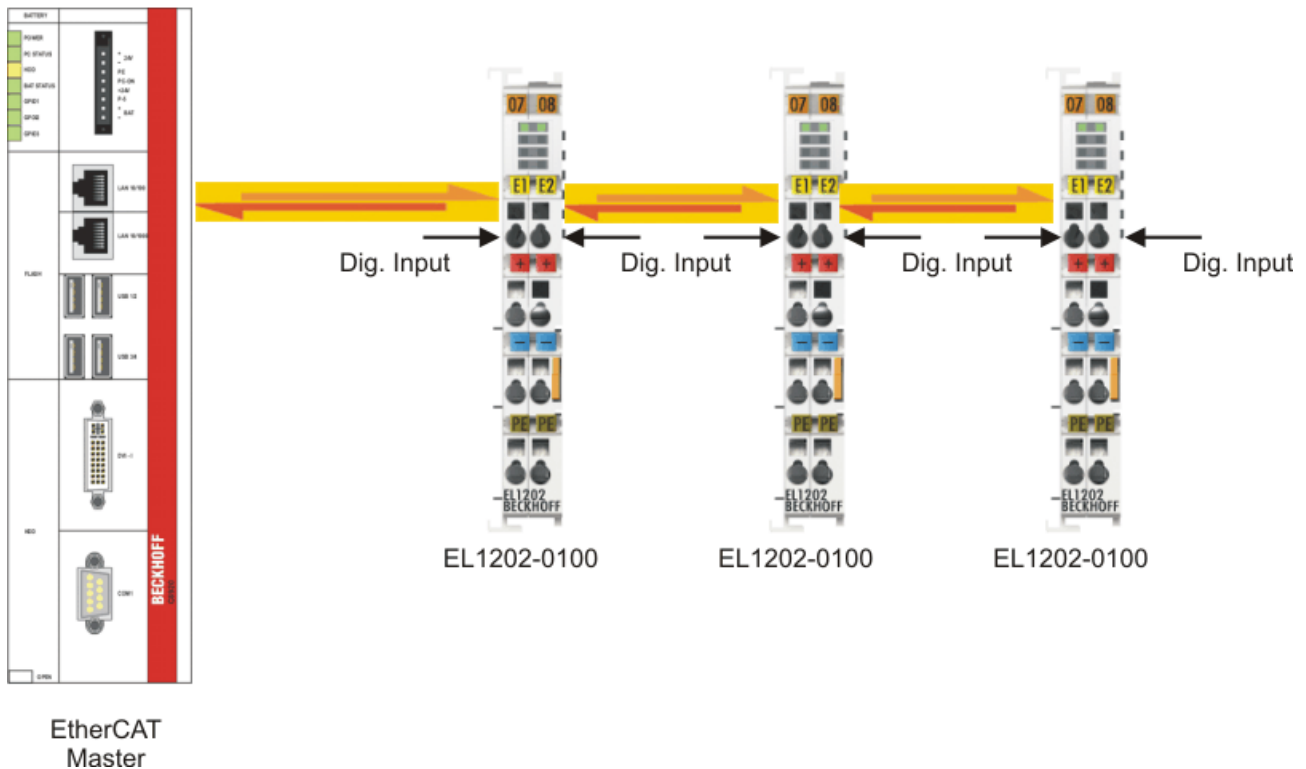


Fig. 256: Sample application with distributed clocks, cycle time 100  $\mu$ s

In Fig. *Sample application with distributed clocks, cycle time 100  $\mu$ s*, the Master PC, with its PLC on an EtherCAT master, drives an EtherCAT configuration with 3 EL1202-0100 digital two-channel input terminals with distributed clocks support (distributed clocks support is achieved through conversion of the EL1202 to the EL1202-0100 in the System Manager; see the corresponding documentation). It is irrelevant which EtherCAT slaves are used and what the distances are between the slaves. The controlling PLC and therefore the EtherCAT fieldbus are operated with a cycle time of 1 ms, i.e. inputs of the EL1202-0100 input terminals are queried at 1 ms intervals. In the EL1202-0100 the local distributed clock is used for sampling the two input channels: With each SYNC interrupt the ESC directly reads the current input value (0 or 1). When the EtherCAT frame passes the terminals shortly afterwards, each EL1202-0100 stores its two bits at the prescribed position in the frame. The distributed clock ensures that input sampling is highly consistent based on the same interval with a tolerance of < 100 ns. In addition, by default *all* EL1202-0100 in the whole EtherCAT network read their inputs at the *same* global time, irrespective of their configuration. In a timing diagram the situation looks as follows:

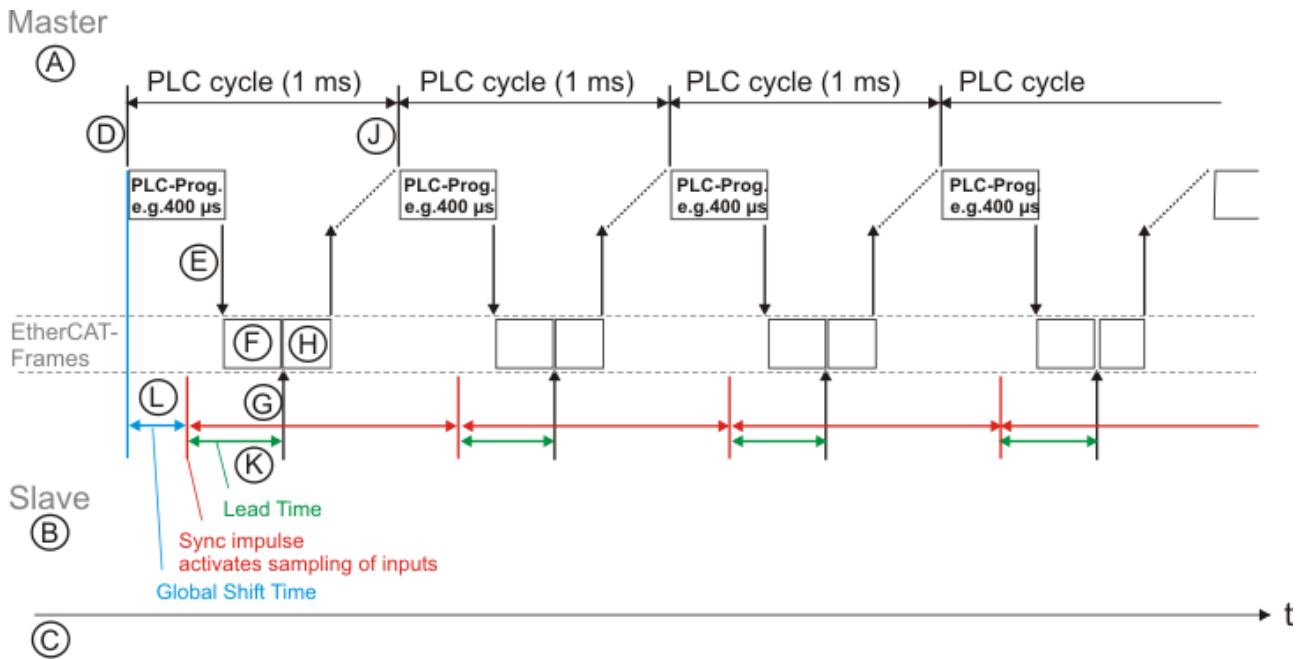


Fig. 257: EL1202-0100 read timing diagram

In Fig. *EL1202-0100 read timing diagram*, the sequence of 4 complete I/O cycles is illustrated taking an EL1202-0100 as an example (cf. Fig. *Sample application with distributed clocks, cycle time 100 μs*). Key to the processes within such a cycle:

- **A, B:** Fig. *EL1202-0100 read timing diagram* is subdivided in grey into a master and a slave side – the transport layer of the Ethernet frame moves in between them.
- **C:** The time axis in x direction enables temporal correlation of the actions described in this section.
- **D:** The PC's real-time clock starts a new processing cycle with a cycle time of 1 ms, for example. In the example the calculation takes 400 μs. The specified program code calculates the output process image based on the current input process image.
- **E:** After the calculation new output data are available for the fieldbus. The EtherCAT frame (or EtherCAT frames) is sent with the process data. Depending on the data quantity the length of an Ethernet frame may be between 7 and 128 μs. It can therefore take some time until it has been transferred to the fieldbus, has passed through all slaves, and was received back by the network card.
- **F:** The EtherCAT frame now passes through all EtherCAT slaves upstream of the current EL1202-0100.
- **G:** Depending on the number of EtherCAT slaves the frame passes the EL1202-0100 after a certain delay (several μs) in order to retrieve the input data (more precisely: it is dynamically processed by the ESC). These data must now be available for collection.
- **H:** The frame then passes through all further slaves until the complete frame is received back again at the network port of the PC.
- **J:** When the next PLC calculation cycle is started by the real-time clock, current input data from the fieldbus are therefore available.

The processes taking place in the EL1202-0100 used as an example are described in more detail below:

- **K:** So that the input data transferred to the frame at (G) is available in the ESC in time to enter the frame, the inputs must be read before the expected frame passage by means of an appropriate *lead time (green)*. Otherwise old data would be included in the frame where applicable. Reading of the physical inputs is triggered by the distributed clocks unit in the ESC through the SYNC signal. This time can be regarded as a safety margin: The shorter this safety margin, the more current the input data. If the chosen time is too short, the input data may no longer be available because the EtherCAT frame may reach the terminal too early, in which case no new process data are coupled into the frame! This lead time is therefore calculated automatically by the EtherCAT master as standard.

- **L:** From the perspective of the master with its own real-time clock the SYNC pulse locally for the terminals therefore occurs with a *global shift time* after the real-time tick. This global shift time is taken into account once when EtherCAT starts up, after which the real-time tick in the PC and the SYNC pulses in the EtherCAT slaves operate with a constant cycle time (in this case 1 ms).
- The Beckhoff TwinCAT EtherCAT master automatically calculates the global shift time based on boundary conditions such as configuration, max. program execution time, cycle time etc., to ensure that operation of all EtherCAT slaves is as safe as possible and as current as possible.
- The user can modify the automatically calculated global shift time by applying a *manual shift time* both at the global level for all EtherCAT slaves and at the local slave level for each individual EtherCAT slave.

### Correction of the PC real-time

Two time periods work alongside each other in Fig. *EL1202-0100 read timing diagram*: the PC with its real-time-tick *and* the distributed clock system with its distributed clocks in the EtherCAT slaves and the EtherCAT master (which is nevertheless located in the PC), as indicated by the grey dividing lines A/B. The reason for the introduction of a second time base for the EtherCAT slaves with derived SYNC interrupts is the jitter with which the EtherCAT frames pass through the individual EtherCAT slaves. Due to the passage through the slaves, the potentially variable PLC runtime and the quality of the real-time tick the time when a frame is sent by the PC and when it actually passes a slave may vary in the  $\mu\text{s}$  range. If local slave events were started solely based on the passage of the communication frame, the actions in the slave would be variable. For high-precision applications this is not tolerable. Through the introduction of clocks that are controlled at local slave level each EtherCAT slave has a high-precision local time base with a tolerance of  $< 100\text{ ns}$  relative to all other clocks, thereby improving the quality of all time-based actions by several orders of magnitude.

However, due to the two different time bases the "field time" (distributed clocks/system time) would deviate from the "PC-time" (operating system/BIOS clock) after a short time, with the deviation increasing continuously.

The TwinCAT EtherCAT master therefore intervenes in the PC clock and cyclically adjusts it based on the EtherCAT reference clock in order to ensure that the real-time tick of the control system and the SYNC interrupts locally for the terminals are synchronous with the set/calculated global shift time (Fig. *EL1202-0100 read timing diagram*, blue).

### ● Synchronization with external master clock

**I** In order to ensure long-term synchronicity of an automation system controlled in this way with an external reference clock (e.g. world time or local network time), suitable EtherCAT slaves must be used that relay the master clock time to the EtherCAT master, or the master clock must be connected directly to the control system. The EtherCAT master then undertakes the required steps for maintaining a low control deviation between master and reference clock.

### Calculation of global shift time

A blue "global shift time" is specified in Fig. *EL1202-0100 read timing diagram*. Its value specifies when – in relation to the overall system behavior of the ESC – a SYNC0/1 action, for example, starts in the slave. The global shift time is composed of various elements, some of which can be changed by the user; see Fig. *Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC*.

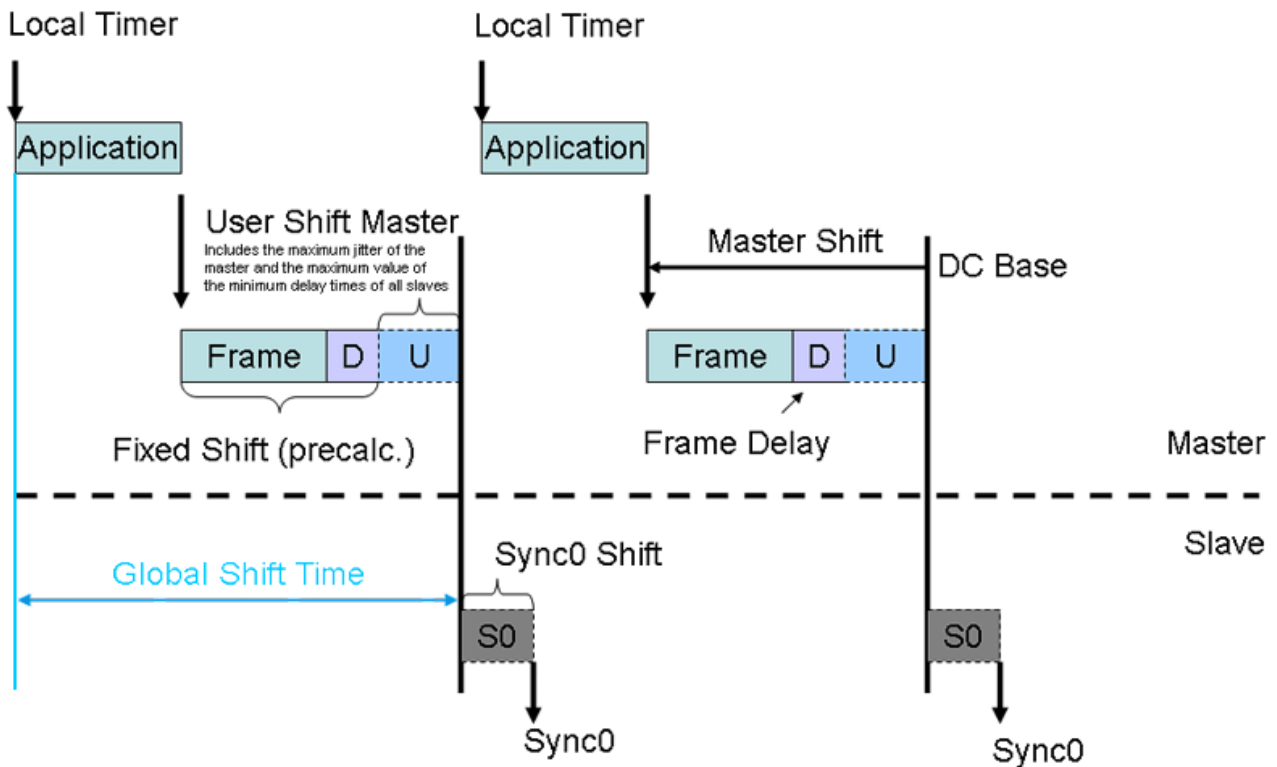


Fig. 258: Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC

The elements of the global shift time according to Fig. *Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC* are:

- **Application**  
The maximum expected calculation time for the program code (cannot be modified).
- **Frame**  
The length of the Ethernet frame (7..128 μs, perhaps several frames) (cannot be modified).
- **D**  
Total delay caused by the frames passing through the EtherCAT slaves, including jitter calculation.
- **U**  
User shift master: shift time predefined with default values by the EtherCAT master depending on the component (can be modified by the user).  
Input and output modules have different values
- **S0**  
User shift time in the slave - usually 0, although in some EtherCAT slaves values <>0 are predefined, can be modified by the user (see respective slave documentation)

Since the elements listed above can be summed to form the global shift time, it can be useful to enter *negative* values in the respective dialog – for instance in order to use a negative SlaveSync0 shift time to trigger the reading of input signals earlier than is foreseen as standard by the EtherCAT master (see Fig. *Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC*).

**NOTE**

**Attention! No plausibility check takes place!**

The mentioned notes and information should be used advisedly. The EtherCAT master automatically allocates settings that support reliable and timely process data acquisition. User intervention at this point may lead to undesired behavior. If these settings are changed in the Beckhoff TwinCAT System Manager, no plausibility checks are carried out on the software side. Correct function of the EtherCAT slaves with all conceivable setting options cannot be guaranteed! Unless specified otherwise in the associated slave documentation, we strongly advise against changing the automatic settings.

## Distributed clocks with output channels

The logic used for the output channels is essentially the same as that for the input channels. The only difference is that the local slave SYNC clock usually takes place *after* a data-carrying EtherCAT frame has passed, instead *before* the frame, as is the case with the input channels. Therefore the calculation of the global shift time specified above (Fig. *EL1202-0100 read timing diagram*, blue) takes place separately for input and output slaves. The global shift time determined in the standard way for the group of output modules is thus larger than that calculated for the group of the input modules. However, both lie within the useful range of 0 – 100% of the EtherCAT cycle time.

In the following sections, access to the distributed clocks settings from the TwinCAT System Manager will now be explained.

## 7.2.2 Distributed Clocks settings in the Beckhoff TwinCAT System Manager (2.10)

Different configuration dialogs are used for the EtherCAT master and the slaves with distributed clock support.

### NOTE

#### Attention! No plausibility check takes place!

The mentioned notes and information should be used advisedly.

The EtherCAT master automatically allocates settings that support reliable and timely process data acquisition.

User intervention at this point may lead to undesired behavior.

If these settings are changed in the Beckhoff TwinCAT System Manager, no plausibility checks are carried out on the software side.

Correct function of the EtherCAT slaves with all conceivable setting options cannot be guaranteed!

Unless specified otherwise in the associated slave documentation, we strongly advise against changing the automatic settings.

#### ● Validity of the following settings

**i** The setting options described below refer to Beckhoff TwinCAT 2.10 Build 1320. More recent editions can have a different user interface design; however, usage remains analogously the same.

### Master settings

Each EtherCAT device in the System Manager offers access (via Advanced Settings) to the Distributed Clocks settings, if EtherCAT slaves are present in the configuration:

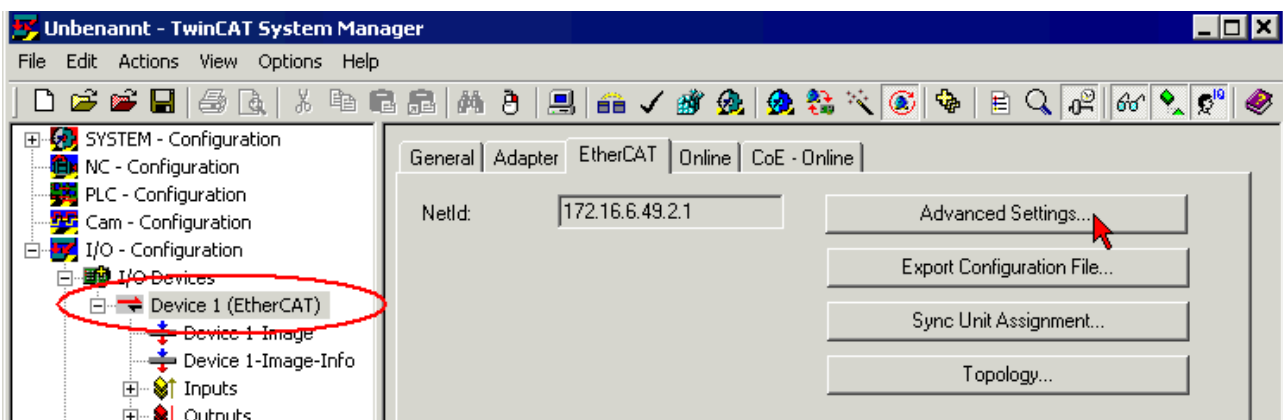


Fig. 259: EtherCAT master – advanced settings

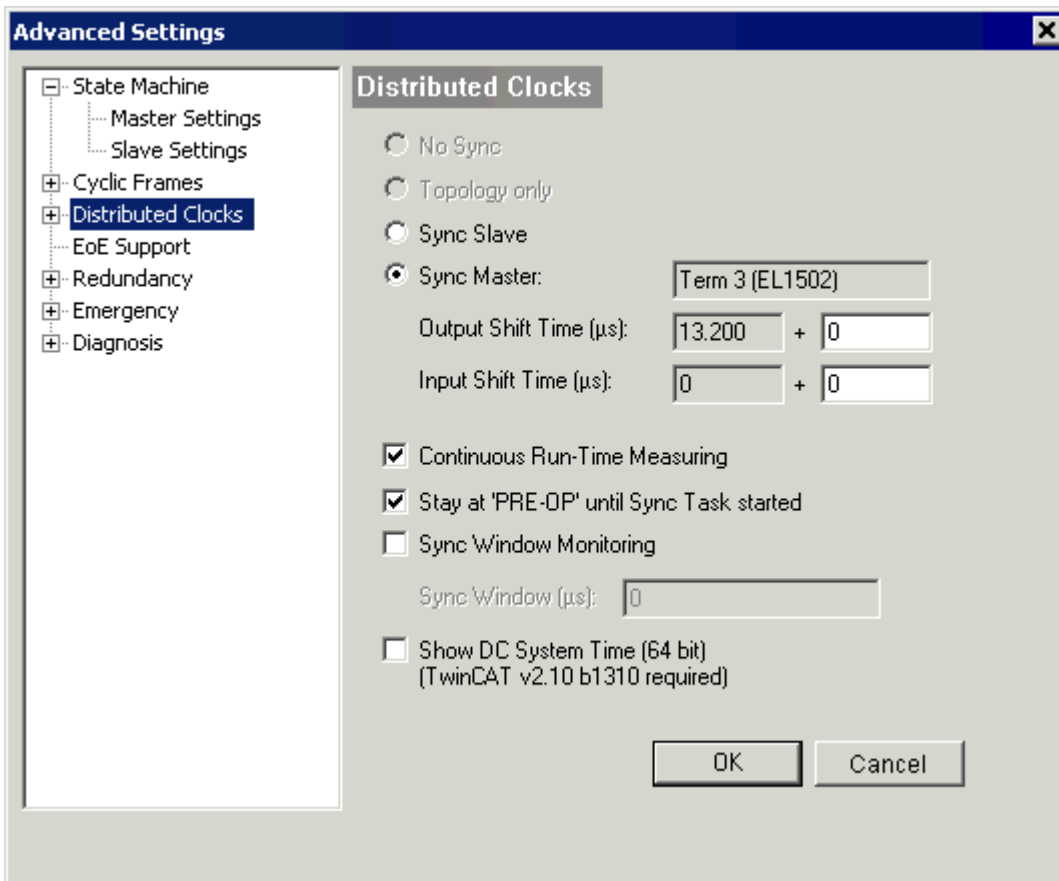


Fig. 260: EtherCAT- Master - Distributed Clocks

- **Sync Slave**

If activated, this EtherCAT device/EtherCAT strand is a Sync Slave to another EtherCAT strand in the same PC, the Sync Master – this contains the reference clock and regulates the real time on the PC. There is no reference clock in the Sync Slave EtherCAT strand.

- **Sync Master**

Default setting – this EtherCAT device (here: device 1) regulates the PC real time; the reference clock is located in this EtherCAT strand and is in this example the EtherCAT slave "Terminal 3", an EL1502.

- **Shift time (µs)**

This is the automatic shift time calculated automatically by the System Manager for all EtherCAT slaves. In this case all local Distributed Clocks in the EtherCAT slaves trigger their SYNC 13.2 µs after the real-time tick. This applies to slaves declared as input or output module.

The user can intervene with an additional value and shift the SYNC pulses in positive or negative direction.

- **Input shift time (µs)**

This shift time only applies to slaves that are declared as input module (see XML device description). Manual modification is possible.

- **Show DC system time (64-bit)**

If activated, the new 64-bit input variable *DcSysTime* appears in the process image of the EtherCAT master; see Fig. *DcSysTime input variable (can be activated)*. It is a copy of the time in the EtherCAT reference clock.

## ● Precision of the variable *DcSysTime*

**i** The time from the reference clock is read without the need for uniform sampling. It is intended as rough guidance for the user to indicate the current time range of the EtherCAT system. It is scanned cyclically, although the value may jitter by up to +/- 1 cycle time due to "soft" sampling. The time values of suitable EtherCAT slaves are to be used for high-precision relative time-based actions – e.g. the timestamp of an EL1252.

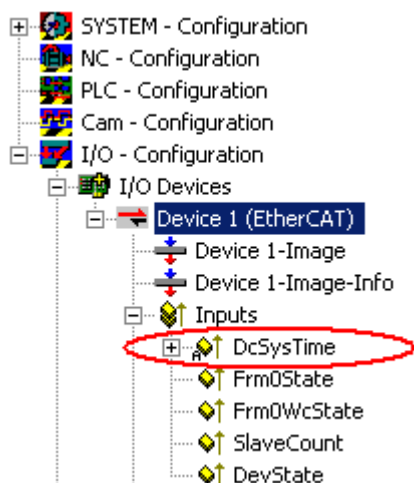


Fig. 261: DcSysTime input variable (can be activated)

### NOTE

#### Attention! No plausibility check takes place!

The mentioned notes and information should be used advisedly.

The EtherCAT master automatically allocates settings that support reliable and timely process data acquisition.

User intervention at this point may lead to undesired behavior.

If these settings are changed in the Beckhoff TwinCAT System Manager, no plausibility checks are carried out on the software side.

Correct function of the EtherCAT slaves with all conceivable setting options cannot be guaranteed!

Unless specified otherwise in the associated slave documentation, we strongly advise against changing the automatic settings.

### Slave settings

#### ● Validity of the following settings

**i** DC functionality is explained using the EL1202-0100 terminal as an example. Each EtherCAT slave with distributed clock support uses this feature on an individual basis as described in the associated documentation.

### NOTE

#### Attention! No plausibility check takes place!

The mentioned notes and information should be used advisedly.

The EtherCAT master automatically allocates settings that support reliable and timely process data acquisition.

User intervention at this point may lead to undesired behavior.

If these settings are changed in the Beckhoff TwinCAT System Manager, no plausibility checks are carried out on the software side. Correct function of the EtherCAT slaves with all conceivable setting options cannot be guaranteed!

Unless specified otherwise in the associated slave documentation, we strongly advise against changing the automatic settings.

### “DC” tab

If an EtherCAT slave supports distributed clock functionality, a "DC" tab appears for the parameterization. If an EtherCAT slave offers several operation modes, the required mode can be selected here. The EL1202-0100 can only be used in one mode, which is why no selection is possible here.

Click "Advanced Settings" to enter the advanced extended distributed clock dialog:



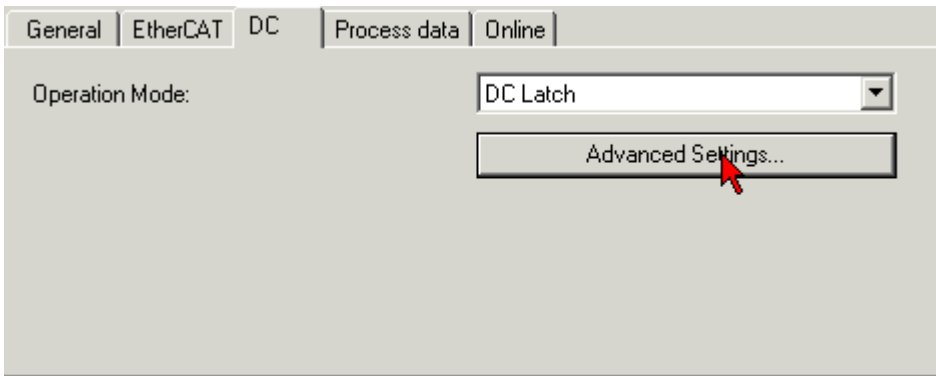


Fig. 262: DC tab

The basic side of any EtherCAT slave with distributed clocks can be seen in Fig. *Distributed clocks dialog in the EtherCAT slave* (TwinCAT 2.10, build 1320):

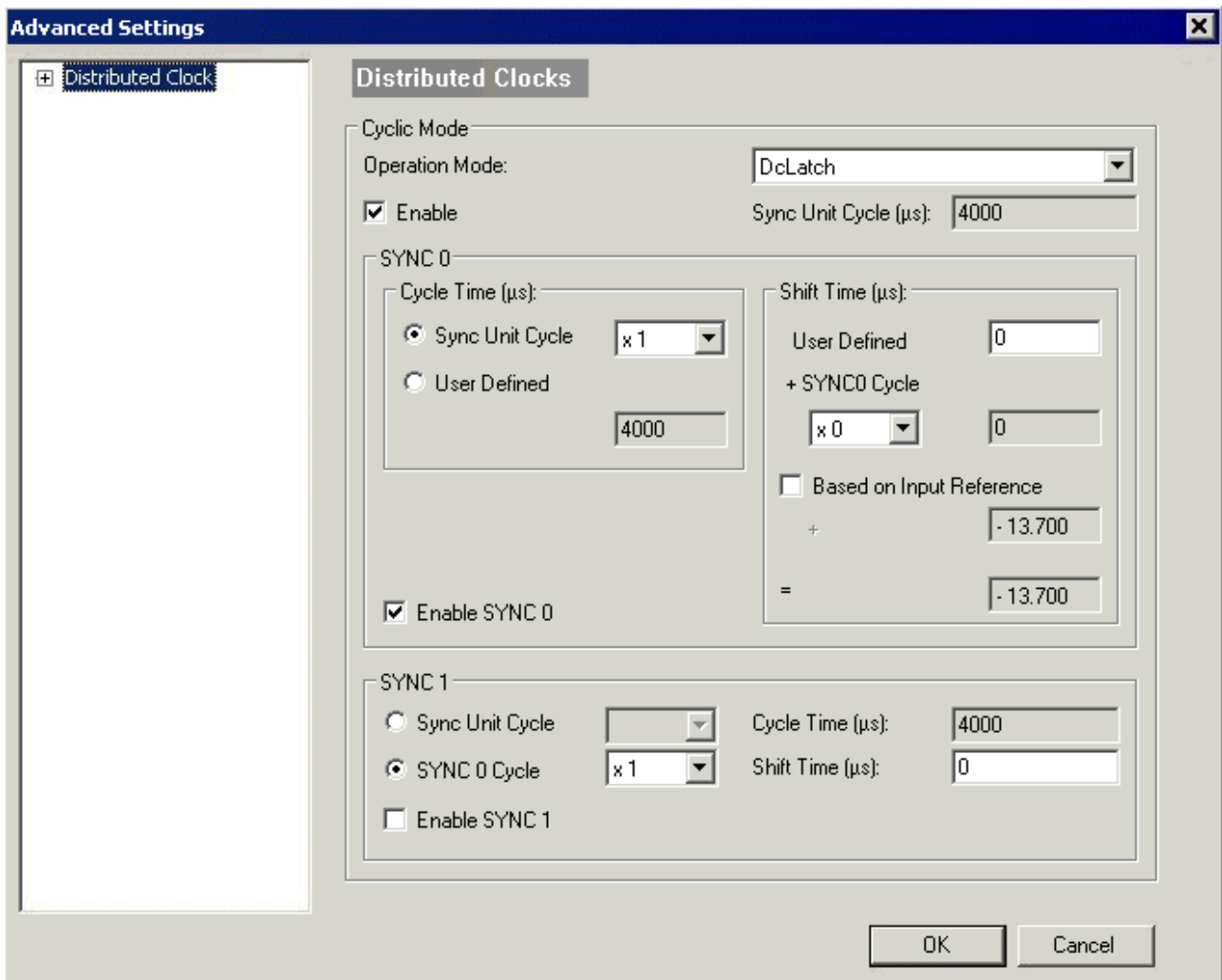


Fig. 263: Distributed clocks dialog in the EtherCAT slave

- **Operation mode**  
Same functionality as in the higher-level dialog.
- **Cyclic mode/enable**  
Activates distributed clocks.

### ● Use of an EtherCAT slave as reference clock

**I** If an EtherCAT slave supports distributed clocks on the manufacturer side, this does not necessarily have to be activated. In order to be able to use an EtherCAT slave as a reference clock, the slave's local clock must be switched on via the 'Enable' checkbox, even if distributed clocks are not required for the actual function of this slave.

- **Sync unit cycle ( $\mu\text{s}$ )**

Basic cycle of the EtherCAT slave - corresponds to the current EtherCAT cycle time of the EtherCAT slave. In this example a task with a cycle time of 4 ms (4000  $\mu\text{s}$ ) scans the EL1202-0100. If several tasks with different cycle times are operated on the same EtherCAT strand, the cycle time of the task that is currently exchanging process data with the slave is shown here. If several tasks are active on the same slave, the fastest task cycle time is shown here.

The following two sections describe the interrupt signals generated by the distributed clock unit in the ESC.

- **Enable SYNC0**

Activates the SYNC0 signal.

- **SYNC0 cycle time**

A multiple or a fraction of the basic cycle described above can be set. The result appears in the window below (here: 4000  $\mu\text{s}$  for factor 1). At these intervals the SYNC0 signal is generated by the ESC, if SYNC0 or distributed clock are activated.

- **User-defined**

Alternative any value can be specified.

- **Shift Time**

As described in the general introduction to distributed clocks, the SYNC pulse of an EtherCAT slave can be shifted forward or back by a constant time (S0 user shift time |▶ 212|). The EL1202-0100 is associated with the input modules, which is why the

- **Based on input reference**

from the global Distributed Clocks setting of the EtherCAT master is applied to this slave. Both

- **User defined and**

- **Multiple of the SYNC0 cycle time** are 0 by default. In the example the time components of this EL1202-0100 add up to a

- **Total SYNC0 shift time** of -13.7  $\mu\text{s}$ .

A somewhat reduced dialog is available for setting the SYNC1 signal:

- **Enable SYNC1**

Activates the SYNC1 signal.

- **SYNC1 cycle time**

The SYNC1 cycle time can either be derived from a multiple/a fraction of the basic cycle or the SYNC0 cycle time.

- **Shift time ( $\mu\text{s}$ )**

A constant shift time between the SYNC0 and the SYNC1 signal can be entered here (in  $\mu\text{s}$ ).

### ● Relationship between SYNC0 and SYNC1

**I** In contrast to the SYNC0 signal the SYNC1 signal is not a fully independent interrupt, as indicated by the reduced and different property dialogs. Further information can be found at [www.ethercat.org](http://www.ethercat.org), e.g. under ESC specifications.

**NOTE**

**Attention! No plausibility check takes place!**

The mentioned notes and information should be used advisedly.  
 The EtherCAT master automatically allocates settings that support reliable and timely process data acquisition.  
 User intervention at this point may lead to undesired behavior.  
 If these settings are changed in the Beckhoff TwinCAT System Manager, no plausibility checks are carried out on the software side.  
 Correct function of the EtherCAT slaves with all conceivable setting options cannot be guaranteed!  
 Unless specified otherwise in the associated slave documentation, we strongly advise against changing the automatic settings.

**Time-related cooperation with other terminals**

The final section of this introduction uses a further example to illustrate the application of distributed clocks in an EtherCAT system.

The task is to sample an analog input value in the range +/- 10 V at precise intervals of 50 µs and to transmit it to the controller; see Fig. *Sample application for a manual shift time on SYNC0*.

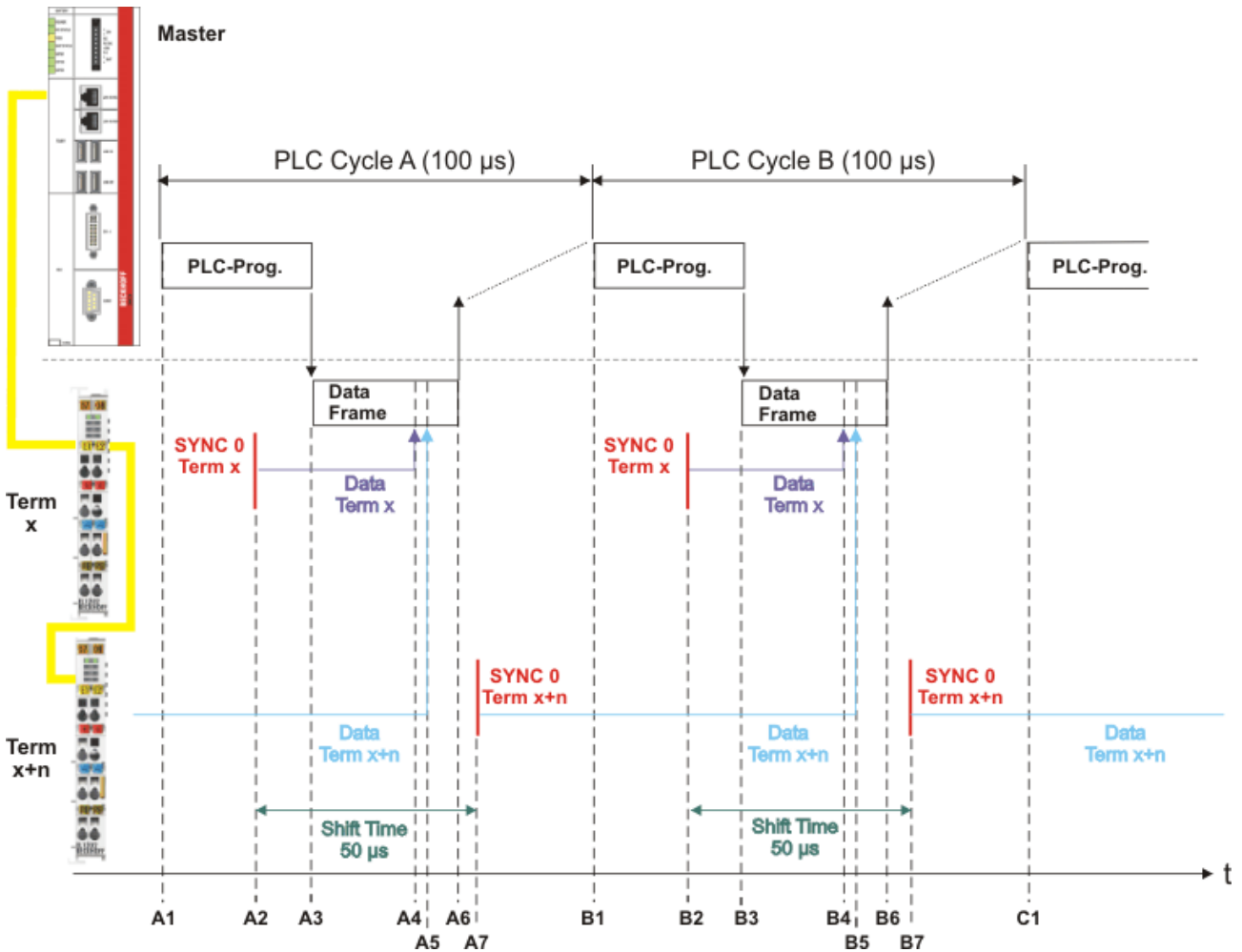


Fig. 264: Sample application for a manual shift time on SYNC0

In this example a fast analog input module (e.g. Beckhoff EL3702) is used in conjunction with an EtherCAT cycle time of 50 µs. A different analog input module with an analog/digital conversion time of 60 µs is used. An EtherCAT cycle time of 50 µs is therefore not appropriate.

The following solution is available: Two such input modules are operated side by side (referred to as "Term x" and "Term x+1" here) and subjected to the same input signal. If the cycle time is 100 µs (A1 to B1) each input module has enough time (100 µs > 60 µs) for converting the analog input value. Conversion of the SYNC0 signal in the input modules is triggered at an interval of 100 µs at time A2, B2 etc. as standard. The

situation at "Term x" is as follows: At time A2 the SYNC signal triggers conversion of the input values (the analog/digital converter is started). The lead time (A2 to A4) is dimensioned such that adequate time is available for converting the data and making them available in the slave, until at time A4 the slave "Term x" couples its input data into the passing Ethernet frame. At time A6 the frame has arrived back at master. The next data processing cycle commences at time B1.

In order to achieve the required time resolution of 50  $\mu$ s, the user manually enters a constant user shift time of 50  $\mu$ s for the SYNC0 signal in the second input module, as shown in the above dialogs. "Term x+1" always converts its input signals 50  $\mu$ s after "Term x" at time A7, B7, etc. "Term x+1" then couples the data determined at time A7 into the Ethernet frame at time B5, shortly *after* B4, since in this example "Term x+1" is located *after* "Term x".

Fig. *Sample application for a manual shift time on SYNC0* provides no information about the position of the data in the *Ethernet frame* – only the time axis is shown to the right!

Both input modules therefore cyclically convert the input value every 100  $\mu$ s with a tolerance of < 100 ns, but with a constant offset of 50  $\mu$ s. The user now has to interpret the process data generated in this way in the right time order within the PLC program. The associated terminals can support this process through a timestamp for process data, as described in the respective documentation.

### 7.2.3 Distributed Clocks settings in the Beckhoff TwinCAT System Manager (2.11)

#### NOTE

##### Attention! No plausibility check takes place!

The mentioned notes and information should be used advisedly.

The EtherCAT master automatically allocates settings that support reliable and timely process data acquisition.

User intervention at this point may lead to undesired behavior.

If these settings are changed in the Beckhoff TwinCAT System Manager, no plausibility checks are carried out on the software side. Correct function of the EtherCAT slaves with all conceivable setting options cannot be guaranteed!

Unless specified otherwise in the associated slave documentation, we strongly advise against changing the automatic settings.

#### ● Validity of the following settings

**I** The setting options described below refer to Beckhoff TwinCAT 2.11 Build 1540. More recent editions can have a different user interface design; however, usage remains analogously the same.

With the launch of TwinCAT 2.11, new features are available in the System Manager and the PLC that make the commissioning of the Distributed Clocks slaves (DC slaves) simpler and clearer. These are

- System Manager
  - individual display of the shift time for each slave
  - external synchronization is possible
- PLC
  - new function *F\_GetCurDcTaskTime*
  - new function *F\_GetActualDcTime*

Statements made in this document in the context of TwinCAT 2.10 remain valid.

#### System Manager - display of the time of action

In the System Manager, from TwinCAT 2.11 on, an optional display allows the online offsetting when outputs are set from the point of view of the controller, or the time from when read inputs originate. The following information applies exclusively to Distributed Clocks slaves.

For basic understanding, a complete EtherCAT update cycle is explained here by way of an example – in the example a digital input will be read and output to a digital output. These are in both cases DC-based slaves, therefore an EL1202-0100 (input, yellow) and an EL2202-0100 (output, red) are used.

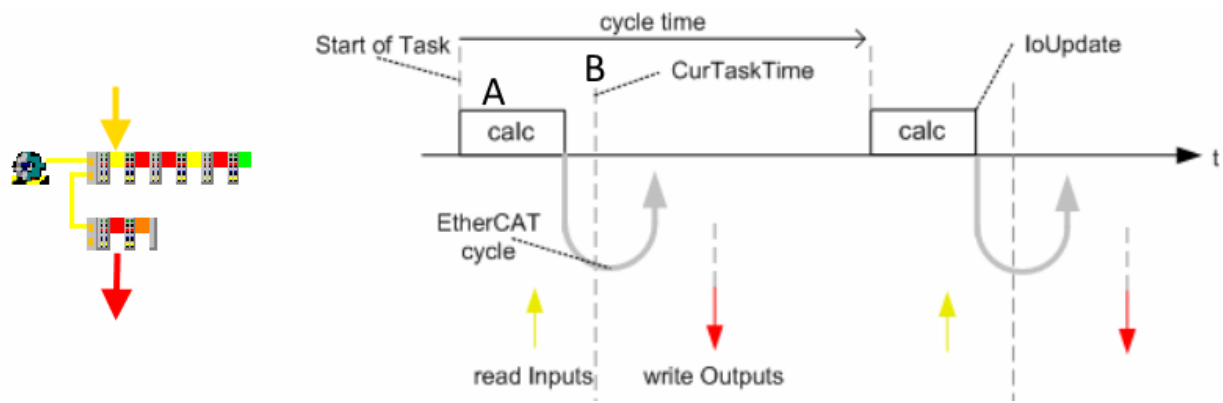


Fig. 265: Topology and principle of operation used in the example

Introductory remarks:

- EtherCAT is configured correctly and executes a fieldbus cycle at EtherCAT 'IoUpdate' with a cycle time in this example of 100 µs after each PLC 'calc' cycle.  
*SeperateInputUpdate* or "IO on task start" is not used in this example.
- In the EtherCAT cycle, the input data is collected from the slaves whilst at the same time the output data is written to the slaves.
- The slave clocks are synchronized to the EtherCAT master by the DC system.
- An external synchronization of TwinCAT to a higher-level reference clock does not take place in this example.

New in TwinCAT 2.11 is the possibility to determine the time of *CurTaskTime* in the PLC online at runtime by calling the function *F\_GetCurDcTaskTime*. The time at which the EtherCAT frame arrives at the first DC-capable device (Fig. *Topology and principle of operation used in the example*, B) is **the** central time to which the entire system is regulated. It corresponds to the **CurTaskTime** of the preceding cycle, which means: If the function *F\_GetCurDcTaskTime* is called in PLC cycle A, it reports time B. This value remains constant even after repeated determination within this task cycle. From the point of view of the running PLC program this is the "now" time. From this "now" perspective the controller sees

- its input data: these were obtained x time ago in the field.
- its output data: these will become effective y time later in the field

The x or y times respectively are constant in each cycle if the EtherCAT system with Beckhoff TwinCAT EtherCAT Master is running stably.

However, this approach is only permissible because the DC system determines exactly when input data is read and output data is output; therefore, it cannot be used for slaves without DC functionality (such as EL200x or EL100x).

The x and y times are now available as pre-calculated values in the System Manager in TwinCAT 2.11 for linking to the controller:

- *DcInputShift*: so 'old' is the input data from the point of view of the 'now' time
- *DcOutputShift*: so much later will the output data become effective in the field.

**Note**

Based on the internal real-time, TwinCAT triggers the task/NC/PLC/... The aim of the control is to always let the (first) EtherCAT frame arrive at the first DC-capable EtherCAT slave at the cycle interval (e.g. 1 ms). This is made more difficult or impossible, if the PLC/task has a very irregular/fluctuating execution time, resulting in synchronization problems at the fieldbus. Appropriate buffer can be provided through the TwinCAT setting "percent of cycle time".

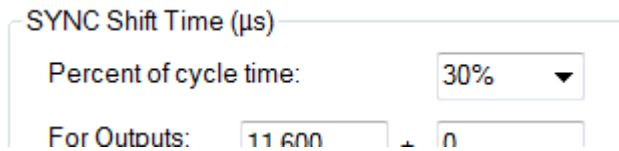


Fig. 266: TwinCAT EtherCAT master setting, section DistributedClocks

Die TwinCAT Echtzeitregelung versucht durch entsprechendes Triggern/Starten der PLC/NC/sonstige Task, trotz der evtl. schwanken Ausführungszeit

### Sample

Let us consider the EL1202-0100 in the above example at a cycle time of 100 µs (100,000 ns) in the configuration.

Name	Online	Type	Size
Input	X 1	BOOL	0.1
Input	0	BOOL	0.1
NextLatchTime	0x044985BCCFE5F650 (308925095439890000)	ULINT	8.0
WcState	0	BOOL	0.1
State	0x0008 (8)	UINT	2.0
DcOutputShift	0xFFFFD8F0 (90000)	DINT	4.0
DcInputShift	0x00033450 (110000)	DINT	4.0

Fig. 267: Process data

Since it is configured like this, the terminal informs about the following in the online state:

- DcOutputShift = 90.000 [ns]
- DcInputShift = 110.000 [ns]

### Comments

- Both time values consist of 32 bits [1 digit = 1 ns] and are therefore sufficient for approx. 4.2 seconds
- Both are to be considered 'logically':
  - positive values for outputs point to the future
  - positive values for inputs point to the past
- Each slave displays both values; which value needs to be taken into account must be judged by the presence of input and/or output variables.

The EL1202-0100 is an input terminal and thus in the 'InputBased' DC group - it works in a standard fashion, i.e. in sync with all of the DC input modules, the yellow symbols in Fig *Topology and principle of operation used in the example*. In addition, it has only input variables. Thus, especially useful is the consideration of the *DcInputShift* value:

From the point of view of the 'now' time of the controller, the input value *Input*= 1 i.e. 110,000 ns is 'old'.

Note: The EL1202-0100 also provides the exact latch time as the *NextLatchTime* process parameter, a feature that not all DC slaves support.

Sequence by way of an example

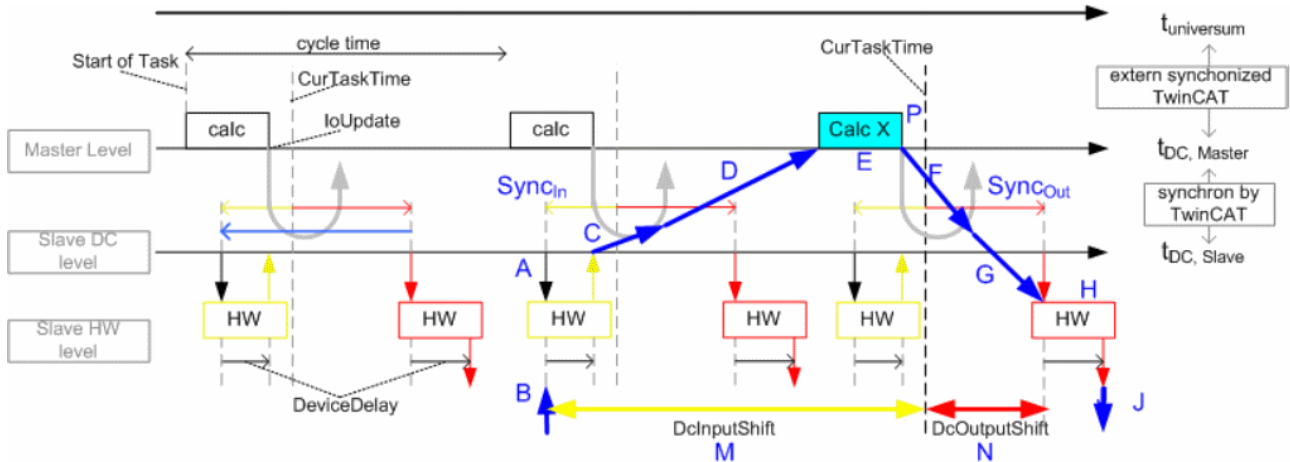


Fig. 268: EtherCAT Update (schematic)

In this example, the digital input B is to be read and immediately output again on a digital output J. We follow the sequence (blue):

- A: the DC unit in the EL1202-0100 initiates the 'latching' of the inputs with Sync<sub>In</sub> - this point in time is fixed and is defined by the shift times in the master/slave. this 'latching' must take place in such a timely manner that the data are securely available before the collecting frame despite any fluctuations.
- B: the inputs are read and a device depending delay occurs - the data is ready to be picked up at time C.  
Note: the delay in the EL 1202 is approx. 0 μs.
- D: The data are collected by the EtherCAT telegram and stored in the input process image of the controller.  
The EtherCAT frame successively passes through the slaves, at successive points in time. The FIRST DC-capable slave, the so-called reference clock, plays a special role: the time at which the (first) EtherCAT frame arrives at this slave is the central point in time *CurTaskTime*
  - The real-time control is aligned to this time: since the central, fixed reference time runs in this slave, the real-time is adjusted based on early/late arrival of the frame.
  - It is the time that is also used in the PLC as a reference point for all shift times *DcInputShift/DcOutputShift*.  
Sample: In the PLC cycle "Calc X", the function *F\_GetCurDcTaskTime* returns the time P (although from the perspective of Calc X it is slightly in the future). On this basis, *DcInputShift* and *DcOutputShift* can then be taken into account accordingly.
- After the calculation E, the output data is output with the next IO cycle or EtherCAT update to the field: F, G.
- With a sufficient safety margin to the fieldbus cycle, the data is output, triggered by the Sync<sub>Out</sub> of the DC unit in the EL2202-0100.
- H, J: Depending on the device, a device delay is still to be taken into account until the data is output.  
Note: the delay in the EL2202 is < 1 μs.

The tasks B and J, which are critical for the user, can always be set by the specifications *DcInputShift* (M) and *DcOutputShift* (N) in relation to the now-time (P).

Note: The function *SeparateInputUpdate* currently (2015-06, TwinCAT 3.1 b4018) has no automatic effect on the calculation of *DcInputShift/DcOutputShift*. The (input) shift times of the corresponding terminals should be adjusted manually, as applicable.

Checking the effectiveness of DC from the point of view of the controller

The above-mentioned values *DcInputShift* and *DcOutputShift* are theoretical values which are valid if the EtherCAT system is running stably. The default settings in the System Manager are defined such that these values are reliably adhered to in virtually all cases.

Typical reasons why they are occasionally or systematically not complied with in the application are:

- poor quality real-time e.g. when using third-party hardware
- cycle timeout due to program errors
- incorrect task prioritization
- LostFrames or otherwise incorrect Ethernet frames, for example, due to faulty wiring
- the SYNC pulses at the slave come at the wrong time due to manual DC setting

Nevertheless, it can be useful in the field to check the actual number of read/write operations that have taken place, for example, during commissioning or the manual optimization of DC slaves. As a check, the following are recommended:

- for input slaves
  - CycleCounter:
    - Some slaves (see documentation) such as the EL37xx or the EL12xx offer a continuous CycleCounter that is incremented in each slave cycle. This can be monitored in the controller.
  - InputToggle:
    - Some slaves offer this input variable, which toggles in every successful slave cycle.
  - WorkingCounter, status:
    - Monitoring this status information is obligatory
- for output slaves
  - WorkingCounter, status:
    - Monitoring this status information is obligatory
  - local error counters:
    - Some slaves (see documentation) expect on their part an incremented CycleCounter from the controller and increment an error counter locally if the controller does not follow this. This error counter in the slave can be read back by the controller. The EL2262 or EL47xx have this mechanism.

### System Manager – Shift Time settings

With respect to the shift time, very similar dialogues can be found in the advanced dialogue of EtherCAT masters and slaves, for which reason they are discussed here together.

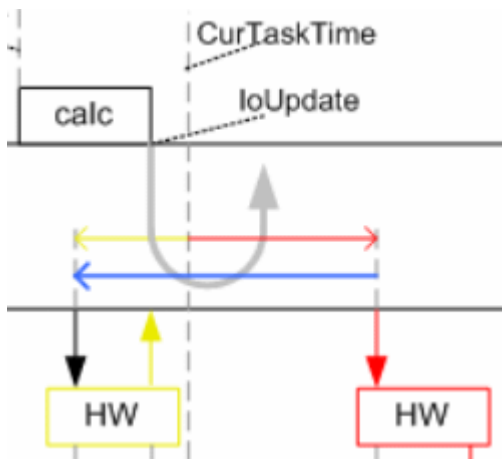


Fig. 269: Local slave shift time

The term 'shift time' in the settings dialogues differs in meaning from the above-mentioned *DcInput/OutputShift*; compare with Figs. *EtherCAT Update (schematic)* und *Local slave shift time*!

- In the TwinCAT settings dialogues, the shift time is specified from the time of the IoUpdate and, logically, originates from the range [-cycle time ... 0 ... +cycle time]  
The internal SYNC pulse in the slave is triggered this much earlier in the case of inputs (Fig. *Local slave shift time*, yellow, "A") or this much later in the case of outputs (Fig. *Local slave shift time*, red, "B") in relation to the time of the IoUpdate.  
However, this does not take into account that some time may be needed to transport the data from/to the slaves! This is only done by



- DcInput/OutputShift according to Fig. *EtherCAT Update (schematic)*, which also takes into account *SeperateInputUpdate* or "IO on task start".

The following applies for setting: effective shift time = master shift time + slave shift time

**In the EtherCAT master:**

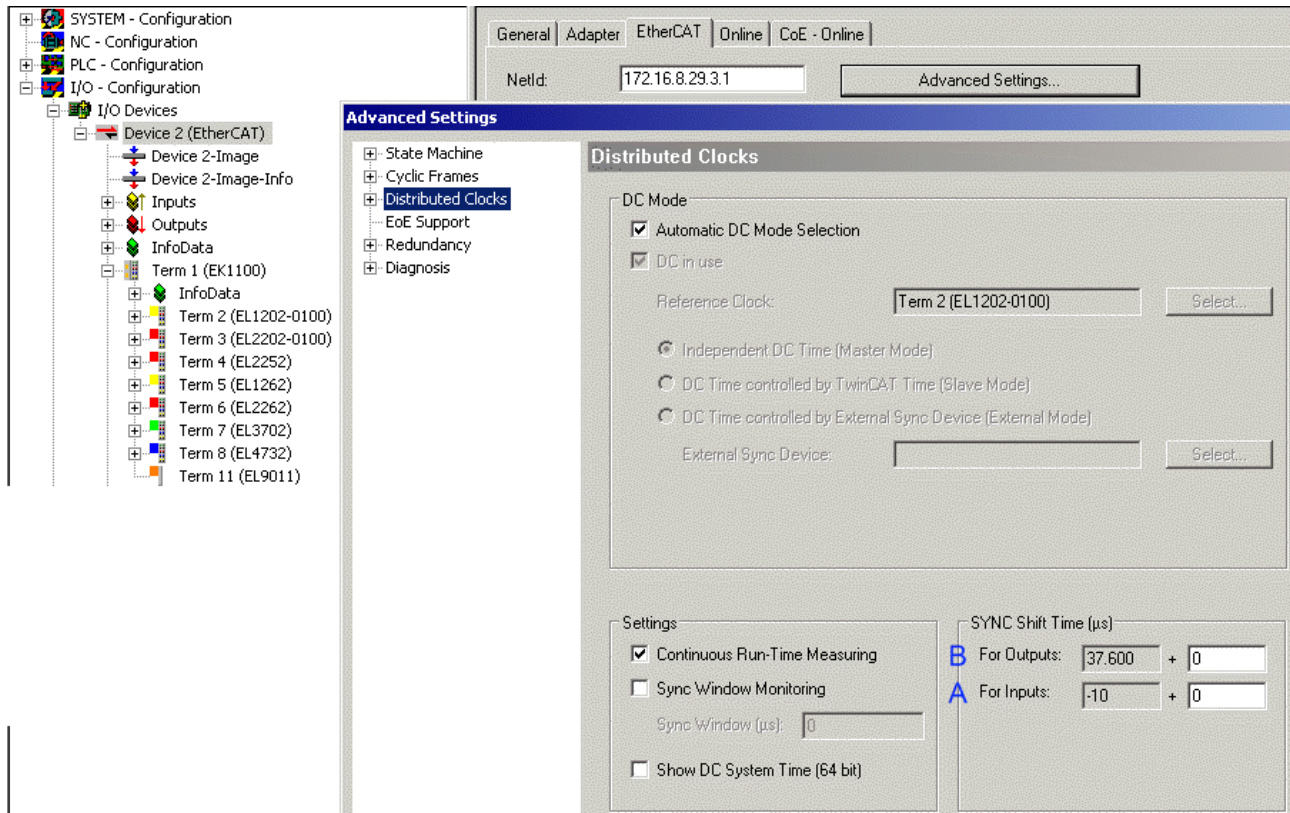


Fig. 270: EtherCAT master setting

Among other things, on the basis of the cycle time, TwinCAT has shifted the group of output modules from the point of view of *CurTaskTime* by 37.6 µs into the future here; the input modules, conversely, will read their inputs 10 µs before the *CurTaskTime*. These two times correspond to A/B from Fig. *Local slave shift time*. If necessary, they can be modified by means of additional entries.

**In the EtherCAT slave:**

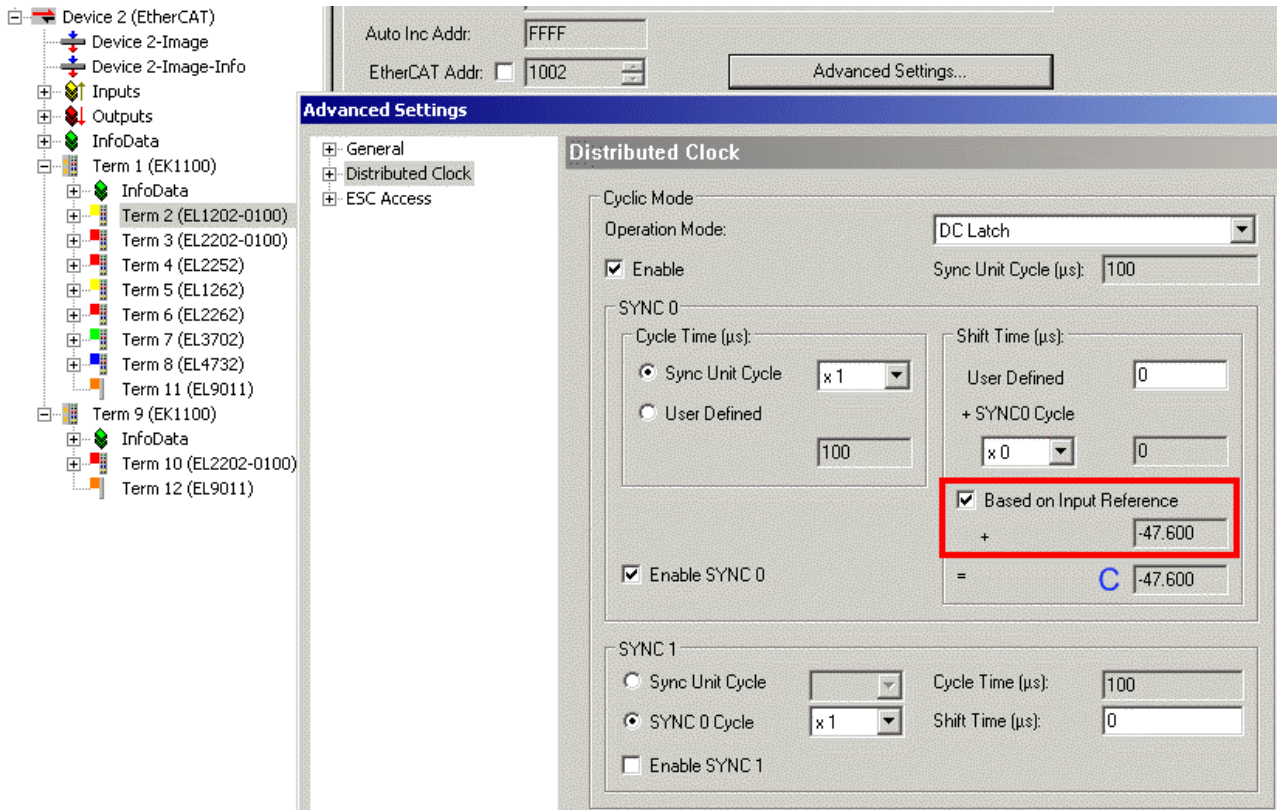


Fig. 271: EtherCAT slave setting

The shift time is displayed from the (default) point of view of the output modules in the slave dialogue - the EL1202-0100 shown here is an input module; it will therefore be automatically specified to *BasedOnInputReference* and its SYNC signal will thus be (-37.6 - 10 μs) ahead of the output modules.

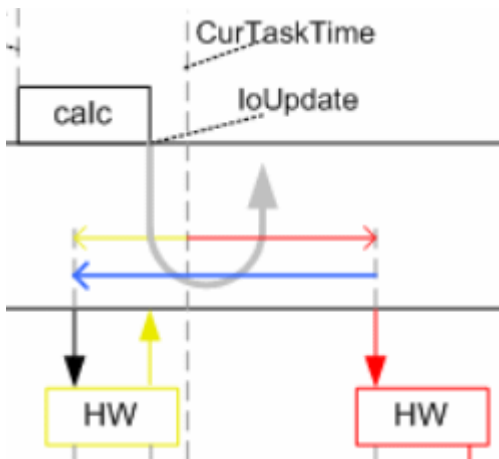


Fig. 272: Shift calculation in the slave dialogue

When a slave is to be 'shifted' manually, consistent values can be entered in the *UserDefined* field.

- Values in the range 10 – 30 % of the cycle time are consistent.
- Changes in the DC entries will become effective only after the activation of the configuration and a restart of the EtherCAT system.

**Displaying the shift time**

The display of the individual slave shift times is possible via the advanced settings of the slave.

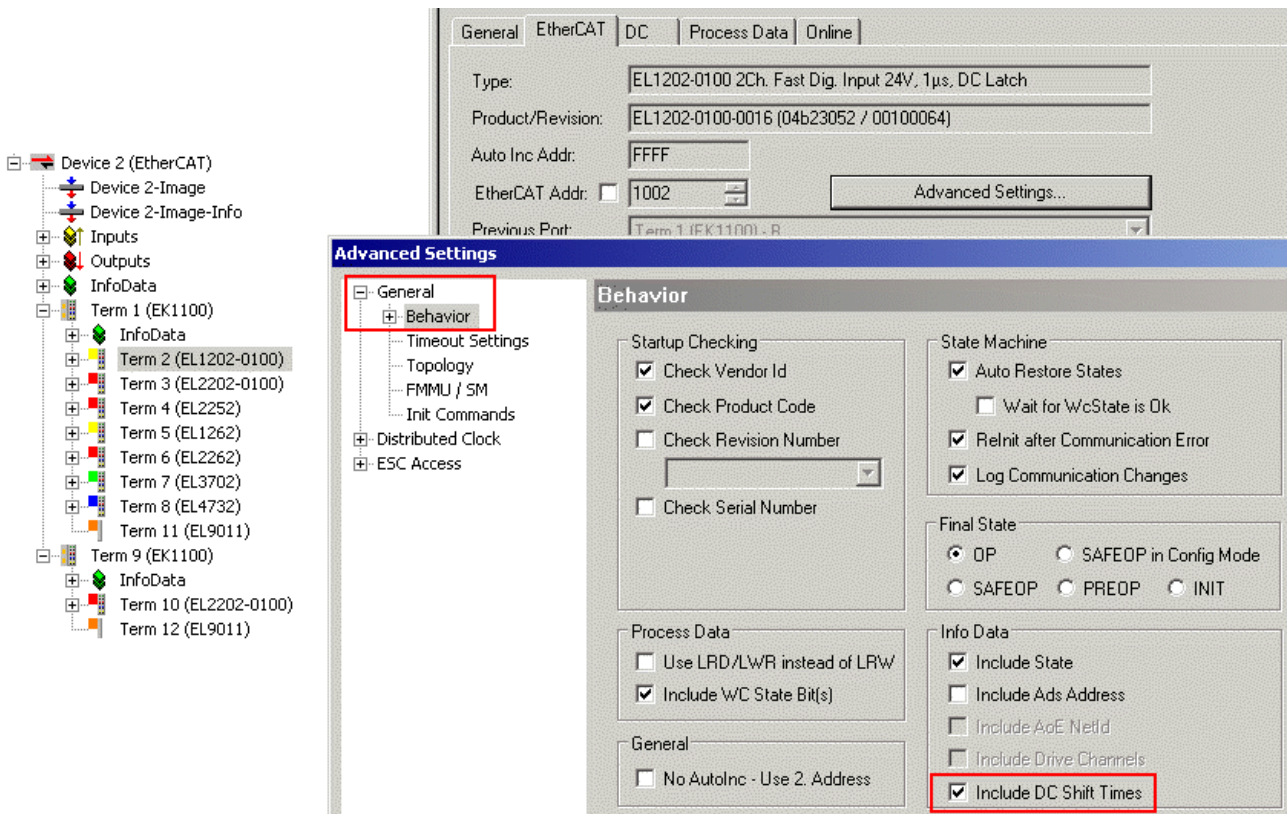


Fig. 273: Displaying the DC shift times

### Advanced Settings

- Continuous Runtime Measuring

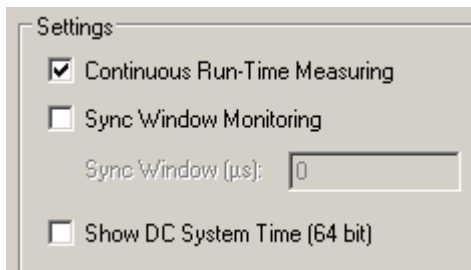


Fig. 274: Continuous setting

If activated, TwinCAT cyclically measures the runtimes between the EtherCAT devices. It is recommended to retain this setting. Deactivation can create space for cyclic data in the case of a very short cycle time, because the NOP datagram is then dispensed with.

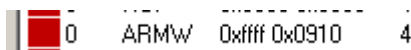


Fig. 275: Runtime Measuring

- SyncWindowMonitoring

If activated, EtherCAT *DevState* shows in bit 12 whether all DC devices are maintaining their local clocks within the specified window.

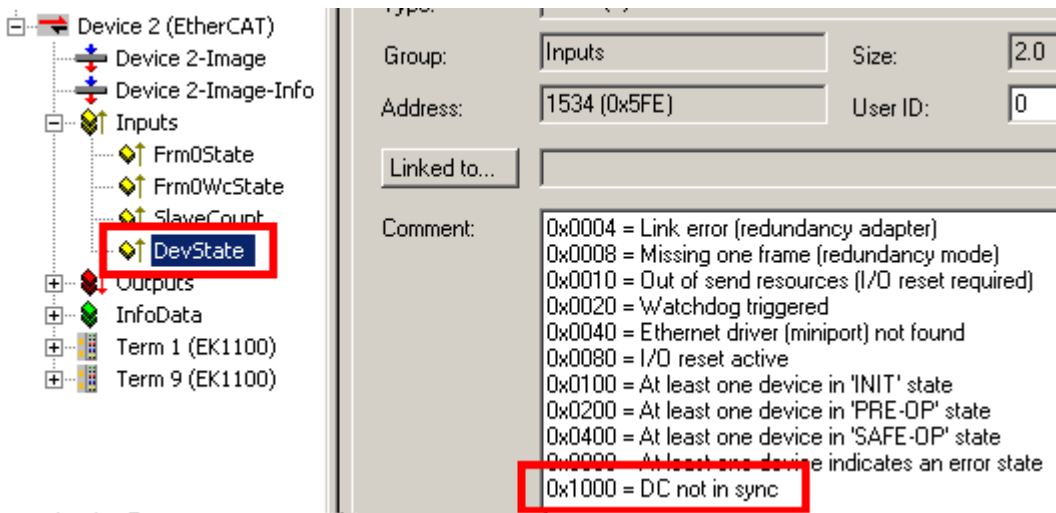


Fig. 276: DevState input

A cyclic BRD command on 0x092C (system time difference) is used for this. The information is only usable if the first EtherCAT device also contains the master clock.

- Show DC System Time

If activated, the current DC time is displayed in the inputs of the EtherCAT master as a copy from the master clock. Since the read out process is subject to the fieldbus transport, preference should be given to the PLC blocks in order to obtain the current DC system time.

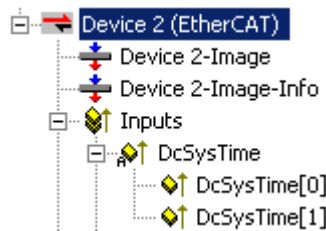


Fig. 277: DcSysTime inputs

### Distributed Clocks diagnosis

The TwinCAT System Manager offers the possibility in an online state to make a preliminary statement about the quality of the current real-time.

Sequence:

- If a task is called, it calculates the time of the next call with the current time and its own cycle time.
- If it is then called on in the next cycle, it compares this expected value with the actual time.
- The deviation is illustrated as shown below.

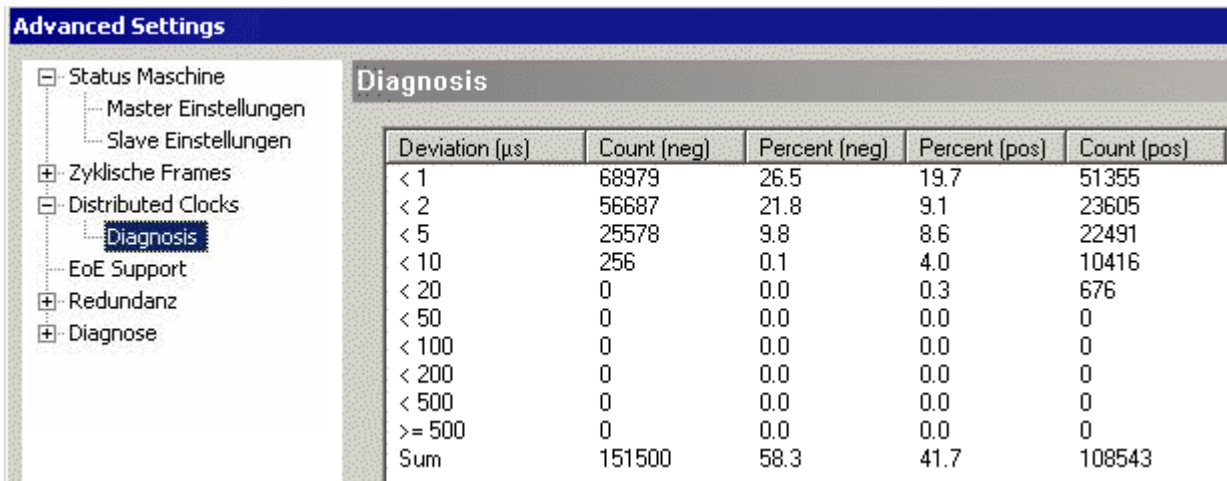


Fig. 278: Online DC diagnosis

Criterion	Good	Bad
Asymmetry	An asymmetry of positive and negative deviations is required. This represents the drift ratio between the master clock and the TwinCAT clock.	If the ratio is 0:100 or 100:0, the DC system is inoperative.
Distribution of the deviation	The deviation values should be predominantly at low levels, see Fig. <i>Online DC diagnosis - adjustment required</i>	If exclusively values in the class "> = 500 $\mu\text{s}$ " occur, the DC system is inoperative.

A system according to Fig. *Online DC diagnosis - adjustment required*, for example, is not suitable for fast EtherCAT applications with a cycle time of, say, 100  $\mu\text{s}$  and may require adjustments.

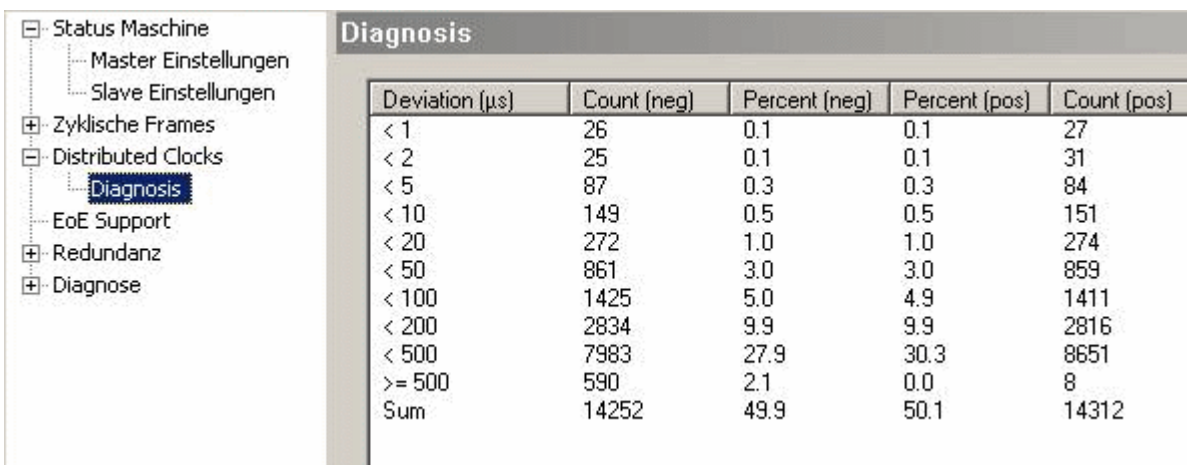


Fig. 279: Online DC diagnosis - adjustment required

## 7.2.4 Distributed Clocks & TwinCAT PLC

### Validity of the following settings

The following information is based on TwinCAT 2.11 build 1539. More recent editions can have a different user interface design; however, usage remains analogously the same. Subfunctions may already have been included in the predecessor versions, but it should be noted that the same TwinCAT version (e.g. 2.11) must be present on both the target system and the programming system in order for the latest functions to be supported.

From the point of view of the PLC, TwinCAT 2.11 is a continuous development of TwinCAT 2.10. The PLC libraries have been expanded to include more features. The following libraries are of particular interest for use with Distributed Clocks functions:

- TcUtilities.lib
  - Functions for the treatment of 64-bit values: UINT64 and INT64
  - Functions for reading different time sources
- TcEtherCAT.lib
  - Functions for the manipulation of EtherCAT devices
  - Functions for reading the DC times

These libraries are included in every TwinCAT installation.

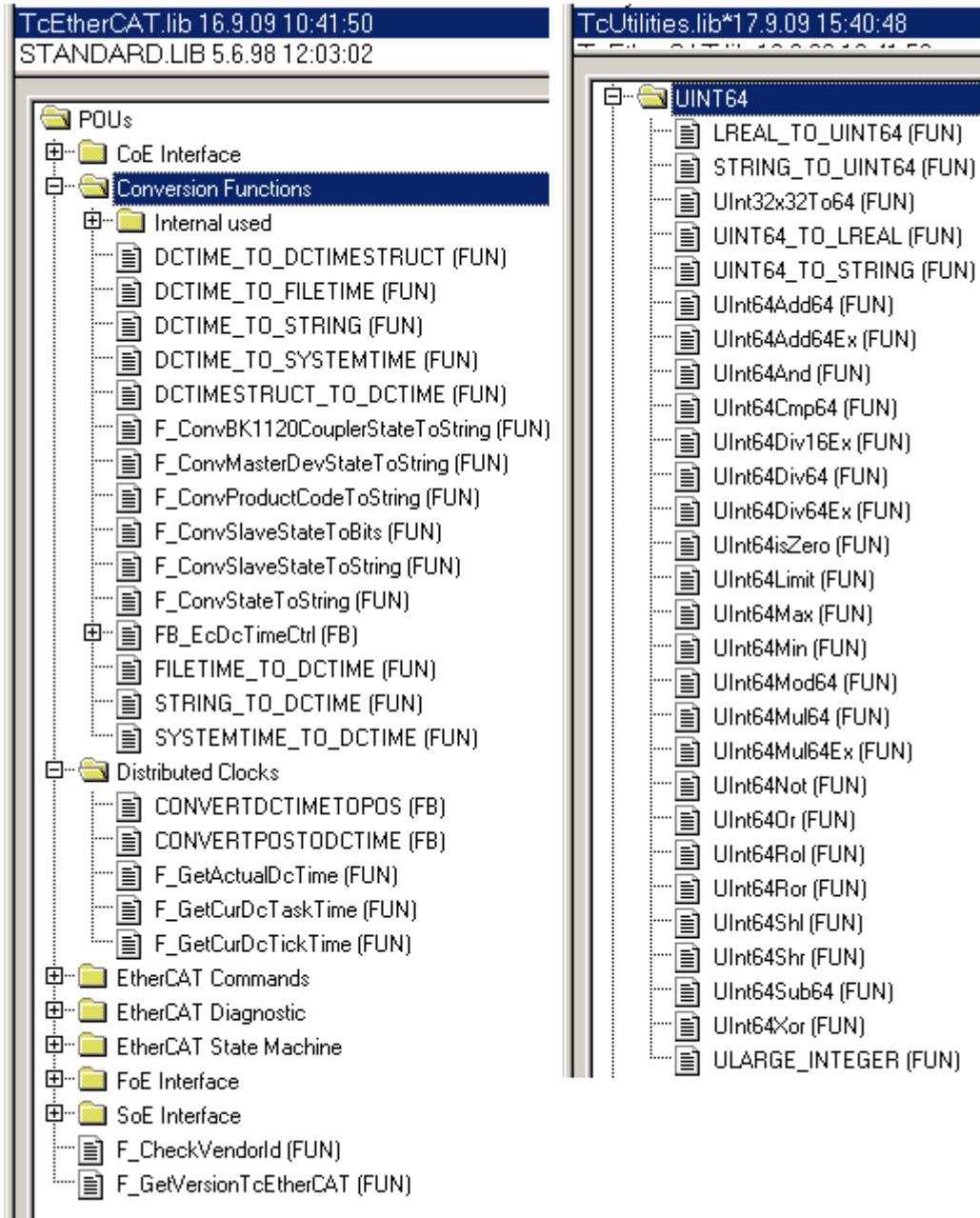


Fig. 280: PLC Libraries

### Working with DC times in the controller

From the perspective of the controller the distributed clock time has the following characteristics:

- Unit 1 ns
- Universal zero point 1.1.2000 00:00, i.e. for variable evaluations an offset of 2000 years has to be added

- Scope up to *64 bit* (sufficient for 584 years). However, some EtherCAT slaves only support a 32 bit scope, i.e. the register overflows locally after approx. 4.2 seconds and starts again at 0.

The following 3 data types are recommended for handling DC times

- **T\_DCTIME** from TcEtherCAT.lib  
This is based on T\_ULARGE\_INTEGER and is therefore unsigned. It can be used for linking with suitable hardware variables
- **T\_ULARGE\_INTEGER** from TcUtilities.lib  
Unsigned 64-bit data type
- **T\_LARGE\_INTEGER** from TcUtilities.lib  
Signed 64-bit data type, negative numbers are represented in two's complement notation (underflow below 0 --> 0xFFFF FFFF FFFF FFFF etc.)  
TcUtilities.lib (section INT64) provides numerous relevant functions. Of particular significance are the cast functions LARGE\_TO\_ULARGE and vice versa.  
This type should be used when working with time differences that may be negative.  
If TwinCAT is used for external synchronization, negative times will inevitably occur in the offset values.

### ● 64- vs. 32-bit representation

**i** Some EtherCAT slaves can only handle 32 bit values for representing the DC time or handle it as a process data. In order to prevent problems caused by overflow (every 4.2 seconds), we strongly recommend using 64-bit times in the controller.

- 32-bit times supplied to the PLC must be complemented with the current High part
- In this case only the Low part (lower 32 bit) should be transferred to the hardware

This sample project

 (<https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/2469155979.zip>) contains a function block that cyclically adds the high part to a 32-bit DC time to make 64 bits.

### Reading the DC time

As described [▶ 195], there are several time sources on a TwinCAT PC; of these, the Distributed Clock time is crucial for the EtherCAT system. Usually, the first DC-capable slave is the EtherCAT master clock; the controller, with its software form of the DC clock, is synchronized to this master clock. Hence, the current DC time is available in the controller.

In the PLC there are 4 possibilities to access the current DC time, the "now" time, during task runtime.

- Readout of the *DcSysTime* input variable in the input data of the EtherCAT device (see System Manager description, not recommended)
- TcEtherCAT.lib *F\_GetActualDcTime*
- TcEtherCAT.lib: *F\_GetCurDcTickTime*
- TcEtherCAT.lib: *F\_GetCurDcTaskTime*

### ● Compatibility

**i** The functions *F\_GetActualDcTime* and *F\_GetCurDcTaskTime* can only be used with TwinCAT version 2.11 or higher (on programming and target system).

### Sample:

A PLC task is configured in the System Manager with a cycle time of 500 µs and 100 µs base time.

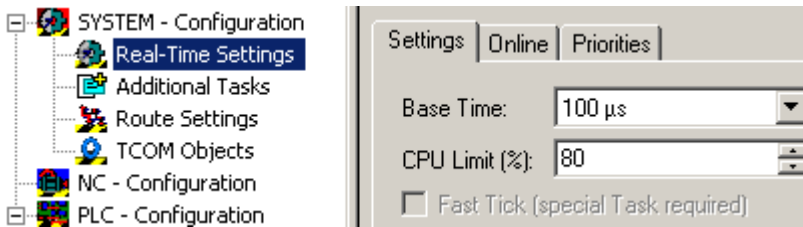


Fig. 281: Sample configuration with 500 µs cycle time

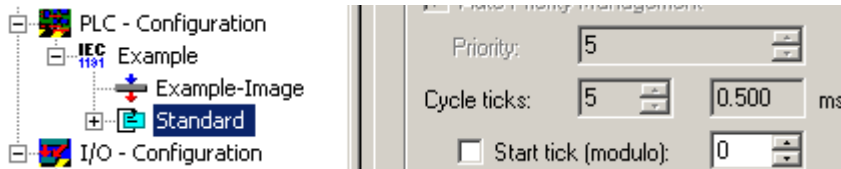


Fig. 282: Sample configuration with 500 µs cycle time

The evaluation of the above-mentioned procurement methods is enabled by Fig. *Methods of reading the DC time*:

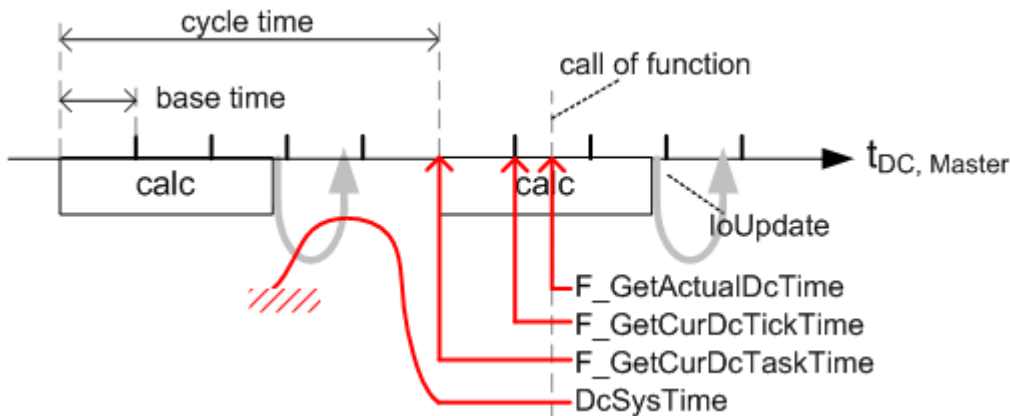


Fig. 283: Methods of reading the DC time

At runtime of the PLC program the three `F_Get...` functions are now called at any desired point. Then, with TwinCAT 2.11 or higher, they return:

- `F_GetActualDcTime`: the actual current DC time. In the case of a repeated call within a task, `F_GetActualDcTime` thus also returns different values.
- `F_GetCurDcTickTime`: Moment of the last base time. A repeated call only returns different values if there lies at least 1 base time in between. If base time = cycle time, or is called at the beginning of a task, then this function returns the same result as
- `F_GetCurDcTaskTime`: 'now' time of this task, to which the slave shift times reference. It is recommended to use this function as the basis of DC operations.

The contents of `DcSysTime` come from the field side master clock - this copying procedure is subject to fluctuations over time and therefore does not advance precisely with *cycle time*. This also applies to other local `DcSystem` times transferred as process data from the hardware.



## 7.2.5 Synchronization modes of an EtherCAT slave

### Information for advanced user

The following information is not necessary for standard applications! Extended basic knowledge of CoE (CAN over EtherCAT) and a deeper understanding of the EtherCAT protocol are required in order to utilize the following information.

#### NOTE

#### Attention! Modification of parameters can have undesired effects!

Imprudent manipulation of the parameters described below can impair the functionality of the EtherCAT slave.

An EtherCAT slave is an electronic device that carries out cyclical sequences of calculations and/or data copying tasks at a certain interval (cycle). The EtherCAT slave working cycle (e.g. in the range of a few  $\mu\text{s}$  up to several ms) can be derived from different sources. Several EtherCAT slaves and/or the EtherCAT master can be operated synchronously, e.g. via the [Distributed Clocks \[▶ 205\]](#) procedure. Associated examples were already mentioned in the introduction page for distributed clocks.

Synchronization or tuning does *not* refer to physical coupling of a processor cycle for electronic equipment in the MHz or GHz range. Instead it refers to starting of tasks at a higher logical level based on the parameters described below. Examples for such tasks are: Setting outputs, reading inputs, copying memory, initiate calculations, etc.

The following synchronization modes are described in this document:

- [Free Run \[▶ 235\]](#) - the EtherCAT slave is not synchronized with EtherCAT. The slave operates autonomously based on its own cycle and is not synchronized with the EtherCAT cycle.
- [Synchronous with SM event \[▶ 237\]](#) - the EtherCAT slave is synchronized with the SyncManager 2 (SM2) event (if the cyclical outputs are transferred) or the SM3 event (if only the cyclical inputs are transferred). The SM2/SM3 event is triggered by the SyncManager when a passing frame is processed.
- [Synchronous with SYNC event \(distributed clocks\) \[▶ 240\]](#) - the EtherCAT slave is synchronized with the SYNC0 or SYNC1 event of the distributed clock system. This type of application was already described in detail above.

All parameter described below are listed as objects in the CoE list of the EtherCAT slave. They can be read online by the slave or determined offline via XML file describing the slave.

### Presence of CoE objects

Even if an EtherCAT slave supports the procedures described below, it does not necessarily mean that the described parameters are accessible for the user or the master via a CoE list. Only EtherCAT slaves with adequate "intelligence" are able to manage a CoE list. Several Beckhoff EtherCAT terminals do not have a CoE list, which means that the parameters described below cannot be displayed or manipulated in these EtherCAT slaves. The functions may nevertheless be available, if they are managed via internal registers.

### Determining the synchronization mode

The different synchronization modes can be determined through different combinations of sub-indices 0x1C32 and 0x1C33.

"--" within the table indicates that the respective sub-index is either not used, may be "0", or does not exist.

	Sync Mode	Synchron- ization Type 0x1C32:0 1	Synchro- nization Type 0x1C33:01	Output Shift Time 0x1C32:0 3	Input Shift Time 0x1C33:03	Calc and Copy Time 0x1C32:06	Calc and Copy Time 0x1C33:0 6	Delay Time (0x1C32:0 9)	Delay Time (0x1C33:09)	Fixed SYNC0 Cycle Time
<i>1 Free Run Mode</i>										
1	Free Run	0x00	0x00	--	--	--	--	--	--	--
<i>2 SM Event Mode</i>										
2	SM2 *	0x01	0x22	--	--	--	--	--	--	--
2	SM3 *	--	0x01	--	--	--	--	--	--	--
3	SM2, Shift Input Latch *	0x01	0x22	--	!=0 **	--	!=0	--	--	--
3	SM3, Shift Input Latch *	--	0x01	--	!=0 **	--	!=0	--	--	--
<i>3 DC Mode</i>										
4	DC	0x02	0x02	--	--	!=0	!=0	!=0	--	--
5	DC, Shift Out- puts Valid and Input Latch with Shift	0x02	0x02	!=0 **	!=0 **	!=0	!=0	!=0	!=0	--
5	DC, Shift of Out- puts Valid with SYNC1	0x03	0x02	!=0 ***	!=0 **	!=0	!=0	!=0	!=0	--
5	DC, Shift of In- put Latch with SYNC1	0x02	0x03	!=0 **	!=0 ***	!=0	!=0	!=0	!=0	--
<i>4 Subordinated Application Controller Cycles</i>										
4	DC, Shift Out- puts Valid/ Input Latch	0x03	0x02 or 0x03	!=0 **	!=0 **	!=0	!=0	!=0	!=0	!=0

Table 1: Determining the synchronization mode

\* If outputs are available, the slave is generally synchronized with the SM2 event. If no outputs are available, the slave is synchronized with the SM3 event.

\*\* Can be modified, if the terminal is able to operate with variable shift times

\*\*\* Shift time can be overwritten (SYNC1 cycle time + delay time)

## Terminology

### Copy and Prepare Outputs

With a trigger event (local timer event, SM2/3 event, or SYNC0/1 event) output data are read from the SyncManager output data area and are then available for mathematical calculations, for example. Subsequently the physical output signal is generated and made available for the process with an "Outputs Valid" ID.

"Copy and Prepare Outputs" describes the total time required for copying of process data from the SyncManager into the local memories and any additional mathematical calculations and hardware delays (depending on the implementation, including software processing time). The individual times are not determined in more detail. They match the values described in SyncManager object 0x1C32:

Described time	SyncManager object 0x1C32
Copying of process data from the SyncManager and mathematical calculations	Calc and Copy Time (0x1C32:06)
Hardware delay time	Delay Time (0x1C32:09)

### "Get and Copy Inputs"

"Get and Copy Inputs" calculates the total time for hardware delays during reading of the input signal, mathematical calculations, and copying the input process data into the input data area of SyncManger 3. The individual times are not determined in more detail. They match the values described in SyncManager object 0x1C33:

Described time	SyncManager object 0x1C32
Mathematical calculations and copying of process data from the local memory to the SyncManager	Calc and Copy Time (0x1C33:06)
Hardware delay to "Input Latch"	Delay Time (0x1C33:09)

The input values are available in the input data area of SyncManger 3 after the min. cycle time (0x1C32:05).

**Outputs Valid**

With the "Outputs Valid" time the outputs are available for the process (e.g. as electrical signal).

**Start Driving Outputs**

At the "Start Driving Outputs" time the  $\mu$ C has set its outputs. The hardware "Delay Time" (0x1C32:09) is the delay between "Start Driving Outputs" and "Outputs Valid".

**Start Latch**

The "Start Latch" time indicates the start of the "Input Latch" process. Between the "Start latch" time and the "Input Latch" time a delay occurs due to the hardware, dependencies relating to the slave implementation, and software processing time, and mapped in the "Delay Time" 0x1C33:09.

**Input Latch**

At the "Input Latch" time acquisition of input data is complete. At this stage any mathematical calculations have not yet been carried out, and the data have not yet been copied into the data area of the SyncManager.

**User Shift Time**

The "User Shift Time" describes the jitter of the master.

**SYNC1 Cycle Time**

The "SYNC1 Cycle Time" can be used for shifting the "Start Input Latch" or "Start Driving Outputs". The "SYNC1 Cycle Time" is represented in register 0x0984:0x0987. It describes the shift between the SYNC0 and SYNC1 signal (SYNC0 is always the reference signal)

**Shift Time**

The "Shift Time" describes the time between the sync events (SM2 event, SM3 event, SYNC0, SYNC1) and the "Outputs Valid" or "Input Latch" times. Writeable value, if the slave supports shifting of "Outputs Valid" or "Input Latch".

**Operation mode 1 - Free Run**

In "Free Run" mode the local cycle is triggered through a local timer interrupt of the application controller. The cycle time can be modified by the master (optional) in order to change the timer interrupt. In "Free Run" mode the local cycle operates independent of the communication cycle and/or the master cycle.

**Optional features**

The slave can have a variable "Cycle Time" (0x1C32:02 can be changed). In this case the "Minimum Cycle Time" (0x1C32:05) is also variable.

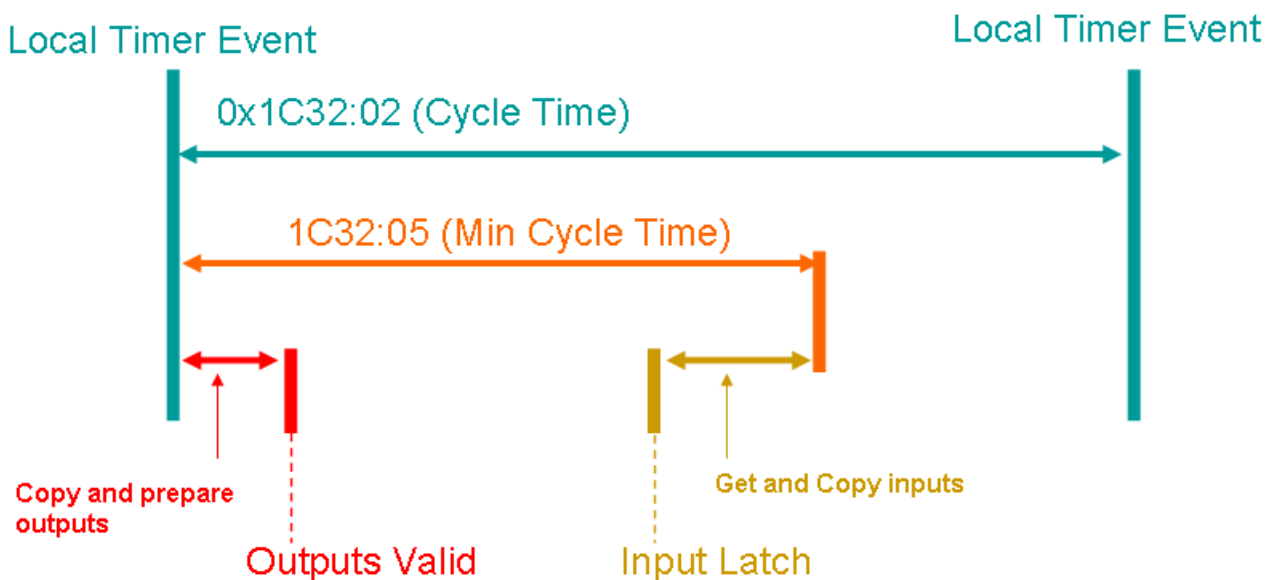


Fig. 284: Local cycle "Free Run" synchronization

Tables "0x1C32 Free Run" and "0x1C33 Free Run" explain the application of these objects in "Free Run" mode.

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x00: Free Run
2	Cycle Time	r or rw	optional	Local cycle time from application controller
3	Shift Time	--	--	
4	Synchronization Types supported	r	required	Bit 0: Free Run supported
5	Minimum Cycle Time	r	conditional	required if 0x1C32:02 variable
6	Calc and Copy Time	--	--	
7	--	--	--	
8	Get Cycle Time	--	--	
9	Delay Time	--	--	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	--	--	
12	SM-Event missed	--	--	
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	--	--	
31:15	--	--	--	
32	Sync Error	--	--	

Table 2: 0x1C32 Free Run

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x00: Free Run
2	Cycle Time	r or rw	optional	same value as 0x1C32:02
3	Shift Time	--	--	
4	Synchronization Types supported	r	required	same value as 0x1C32:04
5	Minimum Cycle Time	r	conditional	same value as 0x1C32:05
6	Calc and Copy Time	--	--	
7	--	--	--	
8	Get Cycle Time	--	--	
9	Delay Time	--	--	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	--	--	
12	SM-Event missed	--	--	
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	--	--	
31:15	--	--	--	
32	Sync Error	--	--	

Table 3: 0x1C33 Free Run

**Operation mode 2 - Synchronous with SM event**

The local cycle is started when the SM2 event [with cyclical outputs] or the SM3 event [without cyclical outputs] is received.

If the outputs are available, the slave is generally synchronized with the SM2 event. If no outputs are available, the slave is synchronized with the SM3 event, e.g. for cyclical inputs.

In this mode the following options are available:

- [Synchronous with SM2/3 event \[► 237\]](#)
- [Synchronous with SM2/3 event, shifting of the "Input Latch" time \[► 239\]](#)

**Synchronous with SM2/3 event**

The local cycle is started when the SM2/3 event is received.

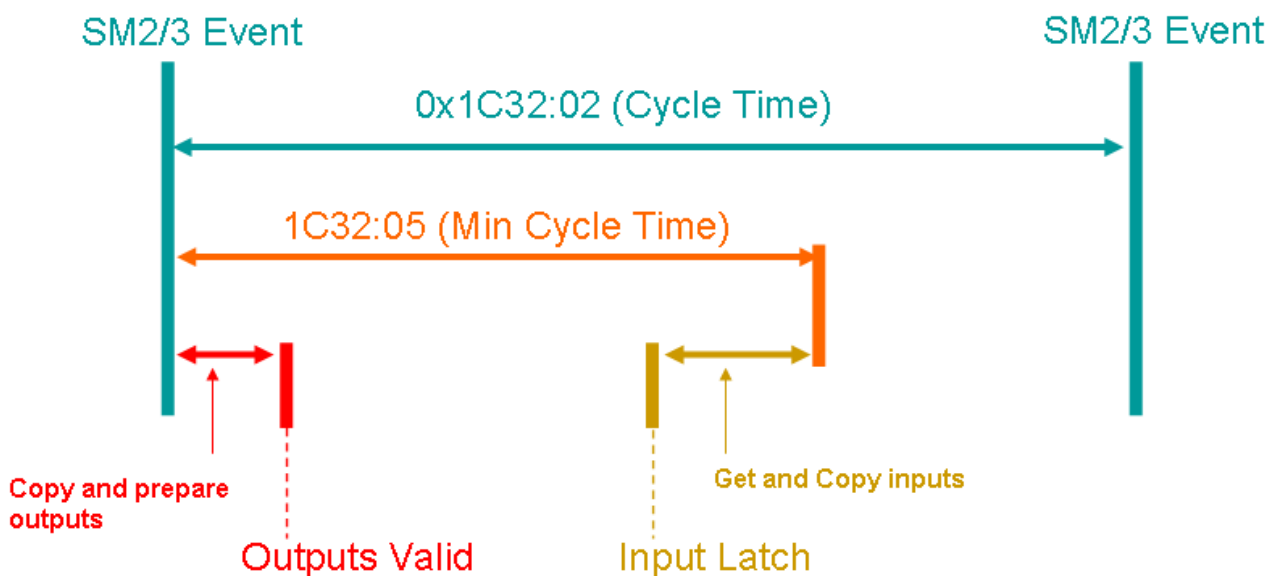


Fig. 285: Local cycle, synchronization with SM2/3 event

Tables "0x1C32 synchronous with SM 2/3 event" and "0x1C33 synchronous with SM 2/3 event" explain the application of these objects in mode "Synchronous with SM 2/3 event".

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x01: Synchronous - synchronized with SM 2 event
2	Cycle Time	r or rw	optional	Communication cycle time
3	Shift Time	--	--	
4	Synchronization Types supported	r	required	Bit 1: Synchronous SM supported
5	Minimum Cycle Time	r	required	
6	Calc and Copy Time	--	--	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	
9	Delay Time	--	--	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	
12	SM-Event Missed	r	optional	
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	r	optional	
31:15	--	--	--	
32	Sync Error	r	conditional	Supported if "SM Event Missed" Counter is used

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 4: 0x1C32 synchronous with SM 2/3 event

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x01: Synchronous - synchronized with SM 3 event (for transfer of inputs in SAFE-OP and OP status) 0x22: Synchronous - synchronized with SM 2 event (for transfer of outputs in SAFE-OP and OP status)
2	Cycle Time	r or rw	optional	same value as 0x1C32:02
3	Shift Time	--	--	
4	Synchronization Types supported	r	required	same value as 0x1C32:04
5	Minimum Cycle Time	r	required	same value as 0x1C32:05
6	Calc and Copy Time	--	--	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	same value as 0x1C32:08
9	Delay Time	--	--	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	same value as 0x1C32:0B
12	SM-Event missed	r	optional	same value as 0x1C32:0C
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	r	optional	same value as 0x1C32:0E
31:15	--	--	--	
32	Sync Error	r	conditional	same value as 0x1C32:20

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 5: 0x1C33 synchronous with SM 2/3 event

**Synchronous with SM2/3 event, shifting of the "Input Latch" time**

The input data should be logged as close as possible to the next SM2/3 event, in order to make the most current input data available to the control system (master). This can be achieved by shifting the "Input Latch" time closer to the SM2/3 event through modification of the "Shift Time" (0x1C33:03).

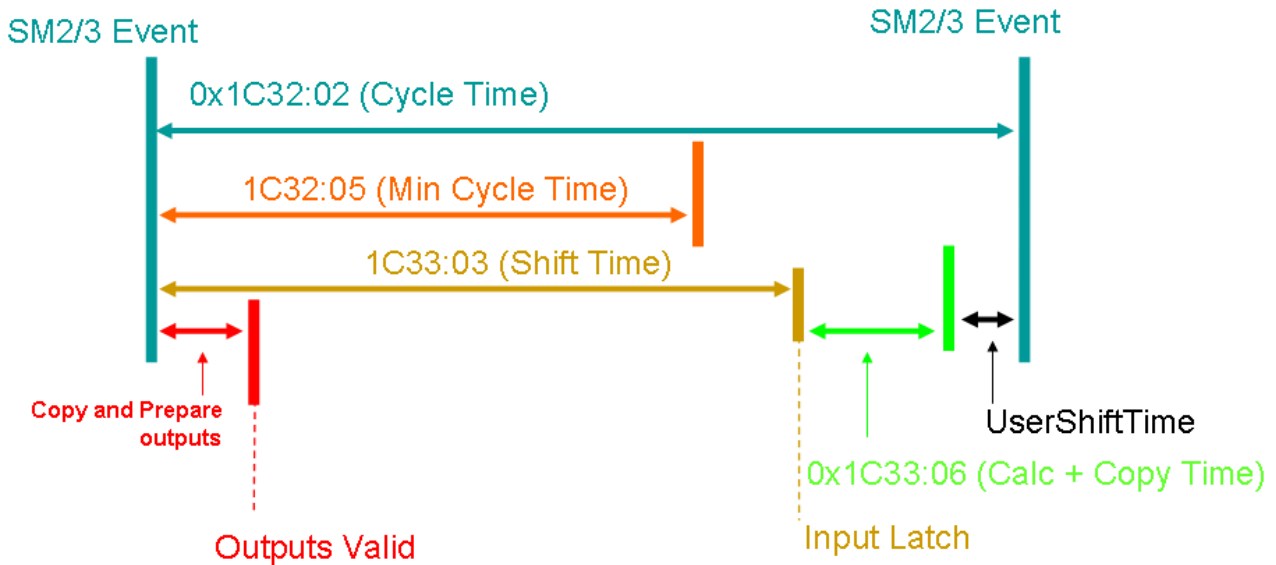


Fig. 286: Local cycle, synchronization with SM2/3 event, shifting of "Input Latch"

Tables "0x1C32 synchronous with SM 2/3 event, shifting of Input Latch" and "0x1C33 synchronous with SM 2/3 event, shifting of Input Latch" explain the application of these objects in mode "Synchronous with SM 2/3 event, if shifting of "Input Latch".

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x01: Synchronous - synchronized with SM 2/3 event
2	Cycle Time	r or rw	optional	Communication cycle time
3	Shift Time	--	--	
4	Synchronization Types supported	r	required	Bit 1: Synchronous SM supported
5	Minimum Cycle Time	r	required	
6	Calc and Copy Time	--	--	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	
9	Delay Time	--	--	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	
12	SM-Event missed	r	optional	
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	r	optional	
31:15	--	--	--	
32	Sync Error	r	conditional	Supported if SM Event Missed Counter is used

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 6: 0x1C32 Synchronous with SM 2/3 event, shifting of the "Input Latch" time"

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x01: Synchronous - synchronized with SM 3 event (only if inputs are available) 0x22: Synchronous with SM2 event (if outputs are available)
2	Cycle Time	r or rw	optional	same value as 0x1C32:02
3	Shift Time	rw	required	
4	Synchronization Types supported	r	required	same value as 0x1C32:04
5	Minimum Cycle Time	r	required	same value as 0x1C32:05
6	Calc and Copy Time	r	required	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	same value as 0x1C32:08
9	Delay Time	--	--	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	same value as 0x1C32:0B
12	SM-Event missed	r	optional	same value as 0x1C32:0C
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	r	optional	same value as 0x1C32:0E
31:15	--	--	--	
32	Sync Error	r	conditional	same value as 0x1C32:20

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 7: 0x1C33 Synchronous with SM 2/3 event, shifting of the "Input Latch" time"

### Operation mode 3 - DC mode (distributed clock mode)

Practical aspects of the synchronization of several EtherCAT slaves with each other and with the EtherCAT master were already described above. This section provides information about the internal parameter for the following operation modes:

- [DC mode \(synchronous with SYNC0 event\) \[► 240\]](#)
- [DC mode, shifting of on "Outputs Valid" and/or "Input Latch" \[► 242\]](#)
- [Distributed clocks - application with subordinate controller cycle \[► 245\]](#)

### DC mode (synchronous with SYNC0 event)

The local cycle is started when the SYNC0 event is received. The process data frame must be fully processed in the slave before the next SYNC0 event is received.



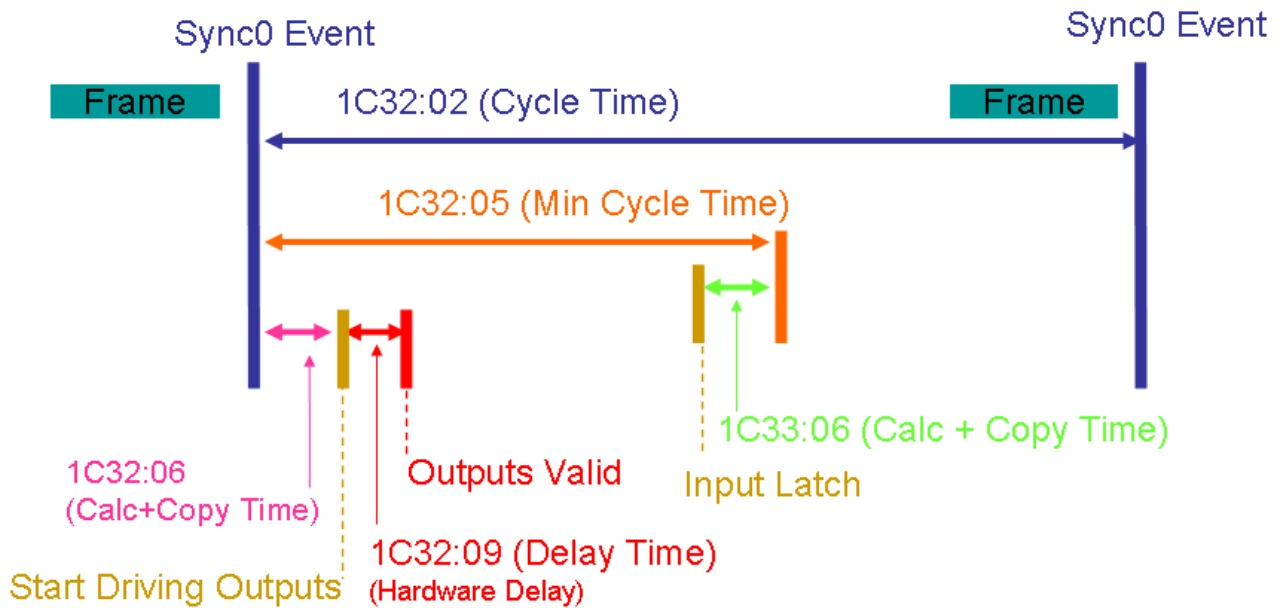


Fig. 287: Local cycle, synchronization with SYNC0 event

Tables "0x1C32 DC mode" and "0x1C33 DC mode" explain the application of these objects in DC mode.

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x02: DC SYNC0 - synchronized with SYNC0 event
2	Cycle Time	r	optional	SYNC0 cycle time (register 0x09A3:0x09A0) Time between two SYNC0 events The SYNC0 cycle time is entered in this index
3	Shift Time	--	--	
4	Synchronization Types supported	r	required	Bit 3:2: DC supported: 01 = DC
5	Minimum Cycle Time	r	required	
6	Calc and Copy Time	r	required	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	
9	Delay Time	r	required	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	
12	SM-Event missed	r	optional	
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	r	optional	
31:15	--	--	--	
32	Sync Error	r	conditional	Supported if SM Event Missed Counter is used

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 8: 0x1C32 DC mode

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r	required	0x02: DC SYNC0 - synchronized with SYNC0 event
2	Cycle Time	r	optional	same value as 0x1C32:02
3	Shift Time	--	--	
4	Synchronization Types supported	r	required	same value as 0x1C32:04
5	Minimum Cycle Time	r	required	same value as 0x1C32:05
6	Calc and Copy Time	r	required	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	same value as 0x1C32:08
9	Delay Time	--	--	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	same value as 0x1C32:0B
12	SM-Event missed	r	optional	same value as 0x1C32:0C
13	Shift Time Too Short	--	--	
14	RxPDO Toggle Failed	r	optional	same value as 0x1C32:0E
31:15	--	--	--	
32	Sync Error	r	conditional	same value as 0x1C32:20

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 9: 0x1C33 DC mode

#### DC mode, shifting of on "Outputs Valid" and/or "Input Latch"

The "Outputs Valid" time can be delayed through the "Shift Time" (0x1C32:03 = 0x02) or the SYNC1 event (0x1C32:01 = 0x03). The "Shift Time" describes the time between the SYNC0 event and the "Outputs Valid" time, while the SYNC1 cycle time describes the time until the "Start Driving Outputs" time.

The "Input Latch" time can either be delayed through the "Shift Time" (0x1C33:03 = 0x02) or the SYNC1 event (0x1C33:01 = 0x03). In this case the "Shift Time" describes the time between the SYNC0 event and the "Input Latch" time, while the SYNC1 cycle time describes the time until the "Start Latch" time.

The SYNC1 signal can either be used for shifting the "Outputs Valid" or "Input Latch" time.

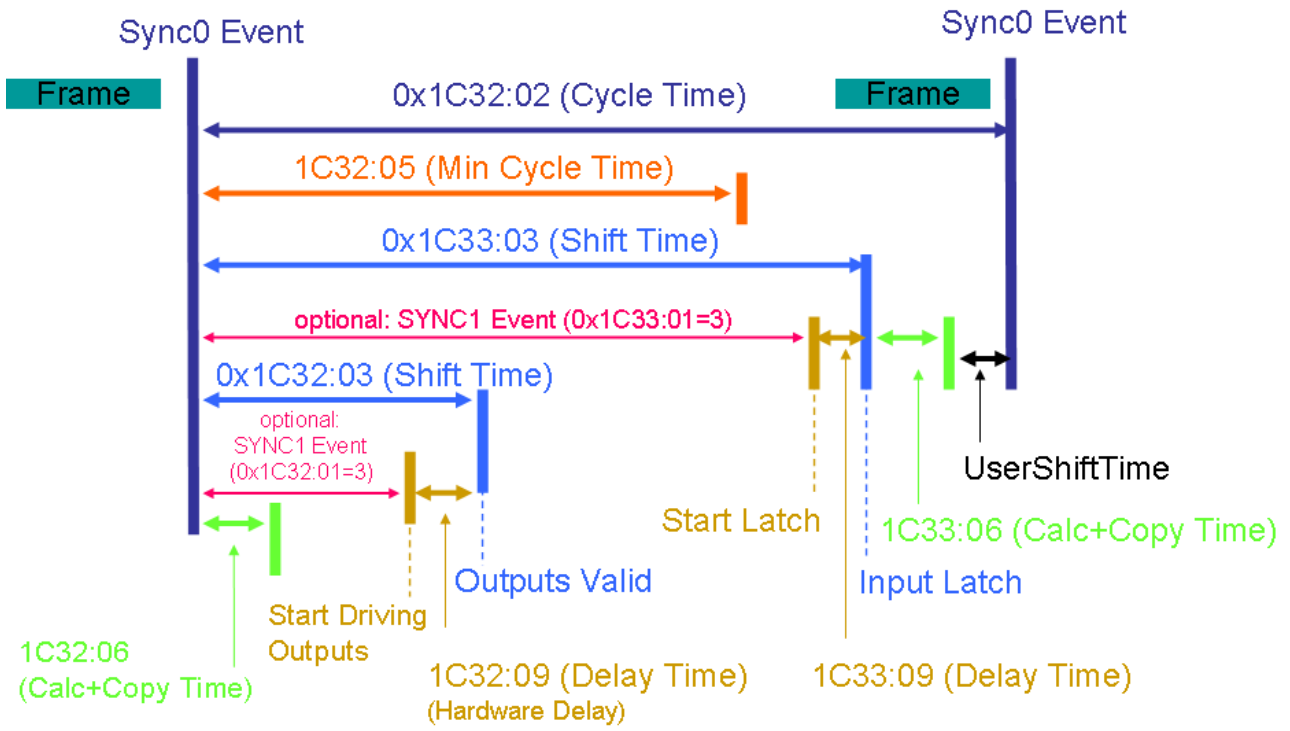


Fig. 288: Local cycle, synchronization with SYNC0 event, shifting of inputs/outputs

Tables "0x1C32 DC mode, *shifting of inputs/outputs*" and "0x1C33 DC mode, *shifting of inputs/outputs*" explain the application of these objects in DC mode with shifting of inputs/outputs.

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	If "Shift Time" is used for delaying the "Outputs Valid" time: 0x02: DC SYNC0 - synchronized with SYNC0 event  If "SYNC1" signal is used for delaying the "Outputs Valid" time: 0x03: DC SYNC1 - synchronized with SYNC1 event
2	Cycle Time	r	required	SYNC0 cycle time (register 0x09A3:0x09A0)  Time between two SYNC0 events The SYNC0 cycle time is entered in this index
3	Shift Time	r or rw	required	The input is writeable if the "Shift Time" is used for shifting the "Outputs Valid" time  If SYNC1 is used for shifting the "Outputs Valid" time, the SYNC1 cycle time + delay time (0x1C32:09) can be copied by the slave application into this entry.
4	Synchronization Types supported	r	required	Bit 3:2: DC supported: 01 = normal DC  Bit 5:4: shift settings 00 = no output time shift  If "Shift Time" is used for shifting the "Outputs Valid" time: 01 = output time shift with local timer (Shift Time)  If "SYNC1" signal is used for shifting the "Outputs Valid" time: 10 = output time shift with SYNC1
5	Minimum Cycle Time	r	required	
6	Calc and Copy Time	r	required	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	
9	Delay Time	r	required	
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	
12	SM-Event missed	r	optional	
13	Shift Time Too Short	r	optional	
14	RxPDO Toggle Failed	r	optional	
31:15	--	--	--	
32	Sync Error	r	conditional	Supported, if "SM Event Missed" or "Shift Time Too Short" counter is used

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time  
 Table 10: 0x1C32 DC mode, shifting of inputs/outputs

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	If "Shift Time" is used for delaying the "Input Latch" time: 0x02: DC SYNC0 - synchronized with SYNC0 event  If "SYNC1" signal is used for delaying the "Input Latch" time: 0x03: DC SYNC1 - synchronized with SYNC1 event
2	Cycle Time	r	required	same value as 0x1C32:02
3	Shift Time	r or rw	required	The input is writeable if the "Shift Time" is used for shifting the "Input Latch" time  If SYNC1 is used for shifting the "Input Latch" time, the SYNC1 cycle time + delay time (0x1C33:09) can be copied by the slave application into this entry.
4	Synchronization Types supported	r	required	Bit 3:2: DC supported: 01 = DC  Bit 5:4: shift settings 00 = no input time shift  If "Shift Time" is used for shifting the "Input Latch" time: 01 = input time shift with local timer (Shift Time)  If "SYNC1" signal is used for shifting the "Outputs Valid" time: 10 = input time shift with SYNC1
5	Minimum Cycle Time	r	required	same value as 0x1C32:05
6	Calc and Copy Time	r	required	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	same value as 0x1C32:08
9	Delay Time	r	conditional	This is used if the "SYNC1" signal is used for shifting the "Input Latch" time
10	SYNC0 Cycle Time	--	--	
11	Cycle Time Too Small	r	required	same value as 0x1C32:0B
12	SM-Event missed	r	optional	same value as 0x1C32:0C
13	Shift Time Too Short	r	optional	same value as 0x1C32:0D
14	RxPDO Toggle Failed	r	optional	same value as 0x1C32:0E
31:15	--	--	--	
32	Sync Error	r	conditional	same value as 0x1C32:20

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 11: 0x1C33 DC mode, shifting of inputs/outputs

**Distributed clocks - application with subordinate controller cycle**

For slaves with fast local cycle times (e.g. control loops) the cycle time of the application controller may be significantly faster than the communication cycle time and/or the SYNC0 cycle time. This is the case with Beckhoff XFC oversampling terminals, for example.

In this case two synchronization features are used:

1. Shifting of the "Outputs Valid" time
2. Shifting of the "Input Latch" time

**DC, subordinate  $\mu$ C cycle, shifting of "Outputs Valid" and/or "Input Latch"**

The "SYNC0" signal is used as a trigger for the local  $\mu$ C cycle. The "Outputs Valid" and "Input Latch" time is triggered by the SYNC1 event and can only be shifted by the "Output Shift Time" or "Input Shift Time".

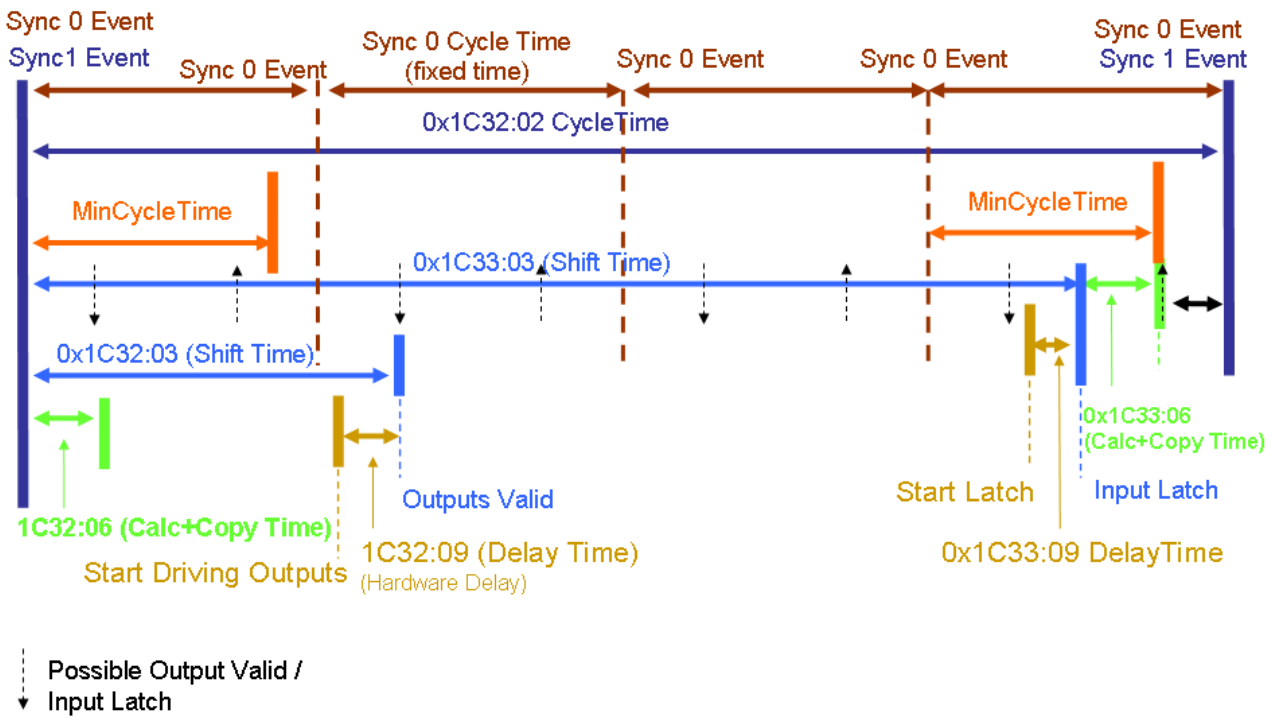


Fig. 289: DC, subordinate  $\mu$ C cycles, inputs/outputs delayed

Tables "01x32C32 DC mode, subordinate  $\mu$ C cycles, shifting of inputs/outputs" and "01x33C32 DC mode, subordinate  $\mu$ C cycles, shifting of inputs/outputs" explain the application of these objects in DC mode for subordinate  $\mu$ C cycles and shifting of the "Outputs Valid"/"Input Latch" time.

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x03: DC SYNC1 - synchronized with SYNC1 event
2	Cycle Time	r or rw	required	SYNC1 cycle time (register 0x09A7:0x09A4) Time between two SYNC1 events SYNC1 cycle time can be copied by the slave application to this entry.
3	Shift Time	rw	optional	
4	Synchronization Types supported	r	required	Bit 3:2: DC supported: 10 = subordinate application  Bit 5:4: shift settings 00 = no output time shift 01 = output time shift with local timer (Shift Time)
5	Minimum Cycle Time	r	required	
6	Calc and Copy Time	r	required	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	
9	Delay Time	r	required	
10	SYNC0 Cycle Time	r	required	
11	Cycle Time Too Small	r	required	
12	SM-Event missed	r	optional	
13	Shift Time Too Short	r	optional	
14	RxPDO Toggle Failed	r	optional	
31:15	--	--	--	
32	Sync Error	r	conditional	Supported, if "SM Event Missed" or "Shift Time Too Short" counter is used

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time  
 Table 12: 0x1C32 DC mode, subordinate  $\mu$ C cycles, shifting of inputs/outputs

Subindex	Description	Flag	Use	Description/default value
1	Synchronization Type	r or rw	required	0x01: DC SYNC1 - synchronized with SYNC1 event
2	Cycle Time	r	required	same value as 0x1C32:02
3	Shift Time	r	optional	
4	Synchronization Types supported	r	required	Bit 3:2: DC supported: 10 = subordinate application  Bit 5:4: shift settings 00 = no input time shift 01 = input time shift with local timer (Shift Time)
5	Minimum Cycle Time	r	required	same value as 0x1C32:05
6	Calc and Copy Time	r	required	
7	--	--	--	
8	Get Cycle Time	rw	conditional****	same value as 0x1C32:08
9	Delay Time	r	required	
10	SYNC0 Cycle Time	r	required	same value as 0x1C32:0A
11	Cycle Time Too Small	r	required	same value as 0x1C32:0B
12	SM-Event missed	r	optional	same value as 0x1C32:0C
13	Shift Time Too Short	r	optional	same value as 0x1C32:0D
14	RxPDO Toggle Failed	r	optional	same value as 0x1C32:0E
31:15	--	--	--	
32	Sync Error	r	conditional	same value as 0x1C32:20

\*\*\*\* used in synchronous mode or in DC mode with variable cycle time

Table 13: 0x1C33 DC mode, subordinate  $\mu$ C cycles, shifting of inputs/outputs

## 7.2.6 EKxxxx - Optional Distributed Clocks support

### Basic principles Distributed Clocks (DC)

The EtherCAT Distributed Clocks system comprises local clocks that are integrated in the EtherCAT slaves and are synchronized by the EtherCAT master via special datagrams. Not all EtherCAT slaves support the Distributed Clocks procedure. It is only supported by slaves whose function requires it. In the TwinCAT System Manager a slave indicates its DC capability by showing "DC" in the settings dialog.

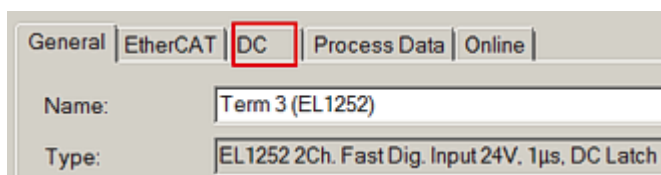


Fig. 290: DC tab for indicating the Distributed Clocks function

Once of these local clocks is the reference clock, based on which all other clocks are synchronized. See also explanatory notes in the [Basic EtherCAT documentation](#). The reference clock must be the first DC-capable EtherCAT slave. By default TwinCAT therefore selects the first DC-capable device as reference clock. This is shown (and can be modified by the user) under advanced properties of the EtherCAT master. The standard setting should not be changed, except in cases where external synchronization is recommended in the relevant documentation, for example.



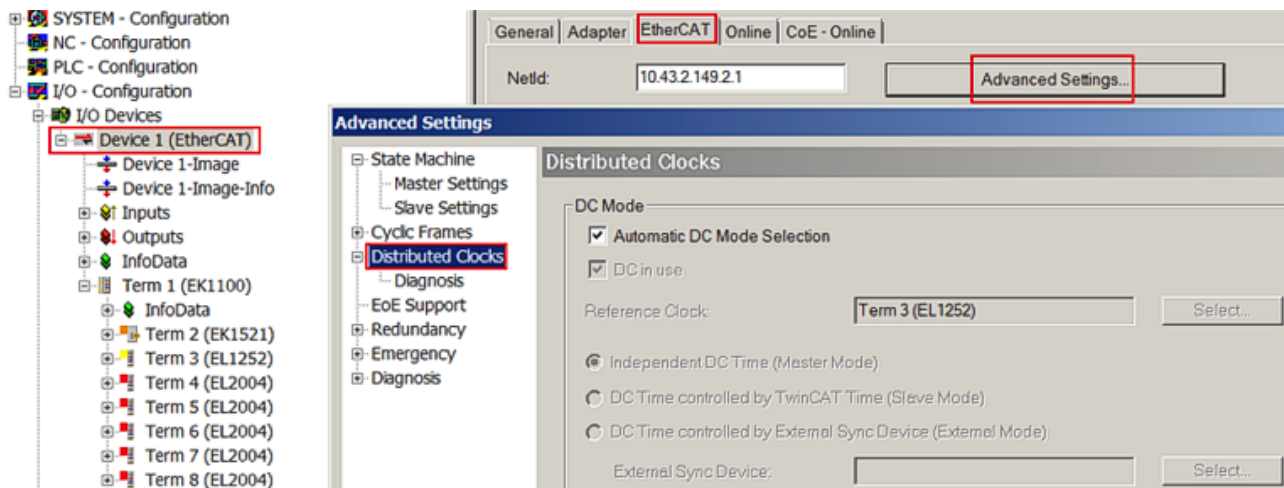


Fig. 291: Advanced Distributed Clocks settings in the EtherCAT master

Fig. *Advanced distributed clocks settings in the EtherCAT master* illustrates how TwinCAT selects the EL1252 as reference clock by default, since the preceding components do not support DC.

### Settings EtherCAT device

System and infrastructure devices such as EK1100 or EK1122 couplers and junction etc. do not require Distributed Clocks to function properly. Nevertheless, it may be topologically expedient to designate the first coupler in an EtherCAT system as reference clock, for example. For this reason, from a certain level the infrastructure components are able to operate as reference clocks, based on special configuration settings.

The components support activation of Distributed Clocks, based on the following table:

Device	XML revision in the configuration	Serial number of the component
BK1150	from BK1150-0000-0016	from firmware 01: xxxx01yy
CU1128	from CU1128-0000-0000	from firmware 00: xxxx00yy
EK1100	from EK1100-0000-0017	from firmware 06: xxxx06yy
EK1101	from EK1101-0000-0017	from firmware 01: xxxx01yy
EK1501	from EK1501-0000-0017	from firmware 01: xxxx01yy
EK1501-0010	from EK1501-0010-0017	from firmware 02: xxxx02yy
EK1122	from EK1122-0000-0017	from firmware 01: xxxx02yy
EK1521	from EK1521-0000-0018	from firmware 03: xxxx03yy
EK1541	from EK1541-0000-0016	from firmware 01: xxxx01yy
EK1561	from EK1561-0000-0016	from firmware 01: xxxx01yy
EK1521-0010	from EK1521-0010-0018	from firmware 03: xxxx03yy
EK1814	from EK1814-0000-0016	from firmware 00: xxxx00yy

Table 1: DC support from rev/firmware version

To ensure that TwinCAT uses such a component as DC reference clock, a manual intervention during the configuration setup is required, as shown here using the EK1100 as an example.

The checkboxes “Cyclic Mode Enable” and “Use as potential Reference Clock” must be set.

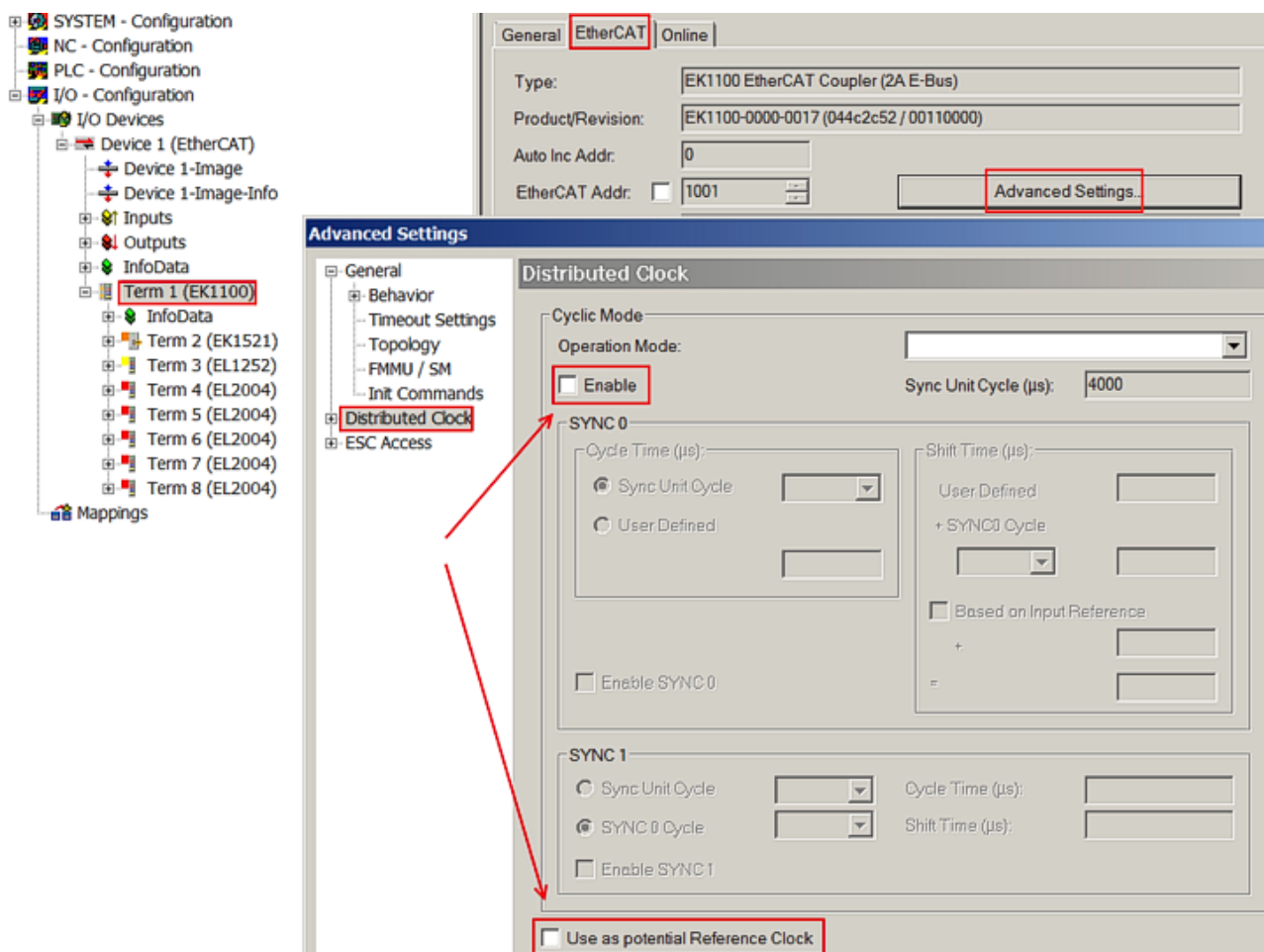


Fig. 292: TwinCAT setting for using this component as reference clock

**i** **Activation of Distributed Clocks support**

The (synchronization) procedure described here is only successful for the components described above. The checkboxes can be set for other components, too, although the hardware does not support this function, unless specified in the respective documentation. In particular, please note that after commissioning the component may not be replaced with a previous version without DC support.

## 7.3 External synchronization

### 7.3.1 Basic principles

**i** **Using the sample programs**

This document contains sample applications of our products for certain areas of application. The application notices provided here are based on typical features of our products and only serve as samples. The notices contained in this document explicitly do not refer to specific applications. The customer is therefore responsible for assessing and deciding whether the product is suitable for a particular application. We accept no responsibility for the completeness and correctness of the source code contained in this document. We reserve the right to modify the content of this document at any time and accept no responsibility for errors and missing information.

**TwinCAT clock hierarchy**

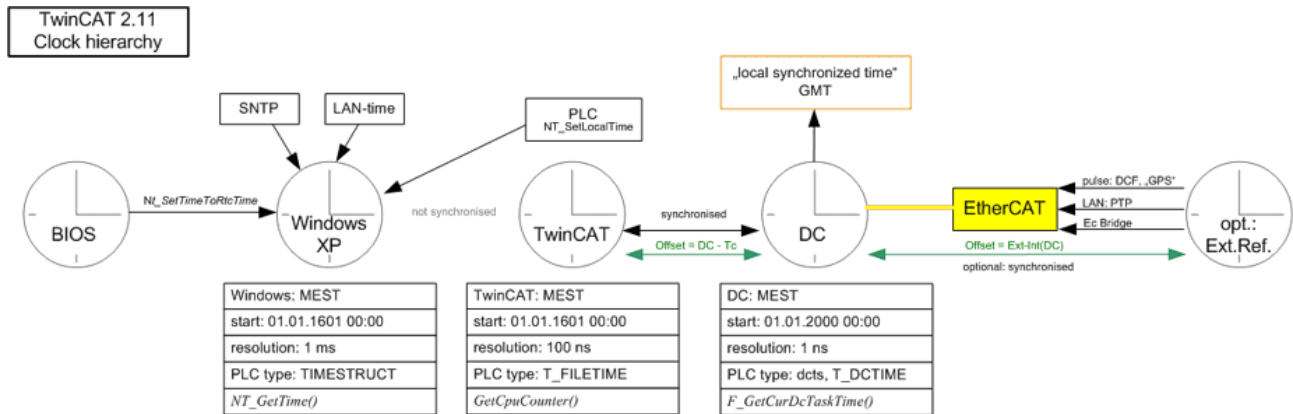


Fig. 293: TwinCAT 2.11 clock hierarchy (subject to change)

The CE operating system uses a different system time. Using the DC clock is therefore recommended.

**Basic principles**

If EtherCAT components are used for external synchronization of TwinCAT, a time is provided for the local controller that matches the higher-level time. As the fieldbus, EtherCAT makes the necessary operating resources available, in particular EtherCAT’s own synchronization mechanism, distributed clocks. In other words,

- in the TwinCAT controller the EtherCAT slaves and the EtherCAT master are synchronized locally in TwinCAT (see also preceding pages).
- The controller is then adjusted based on the higher-level clock total as a slave clock with DC clock.

The procedure is as follows:

- The frequencies of the two time bases are synchronized
- The offset between the two time bases is determined and announced

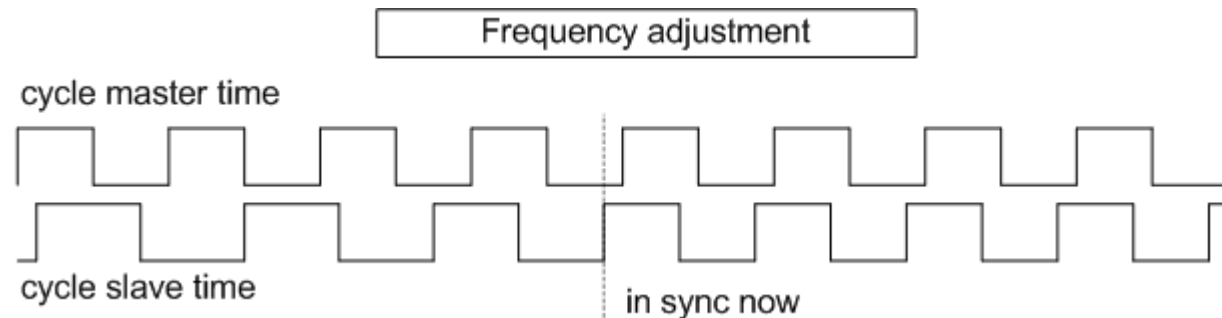


Fig. 294: Frequency synchronicity

Note the following:

- After TwinCAT has started, the slave clock controller adjusts the frequency of its distributed clock time based on the higher-level time:
  - at the start of EtherCAT, the initial offset between the two times is determined.
  - the subsequent adjustment keeps this offset constant and makes it known.
  - the readjustment takes place continuously.
- In the case of failure of the synchronization (interruption of the connection, restart of one of the systems), the behavior is as follows:
  - once slave clock control resumes, a new offset is calculated and announced.
  - the application must therefore continuously observe this offset.
- A new offset is also calculated, if the control limit of  $\pm 1$  cycle time is exceeded.
- This does not affect the BIOS clock (motherboard) or the operating system clock (Windows).

- The TwinCAT clock also remains unchanged.
- The local DC time must still be used for tasks related to the respective station hardware (EtherCAT slaves, terminals).
- If the TwinCAT time is used in the application, the TcToDc offset between the TwinCAT clock and the DC clock must be taken into account.

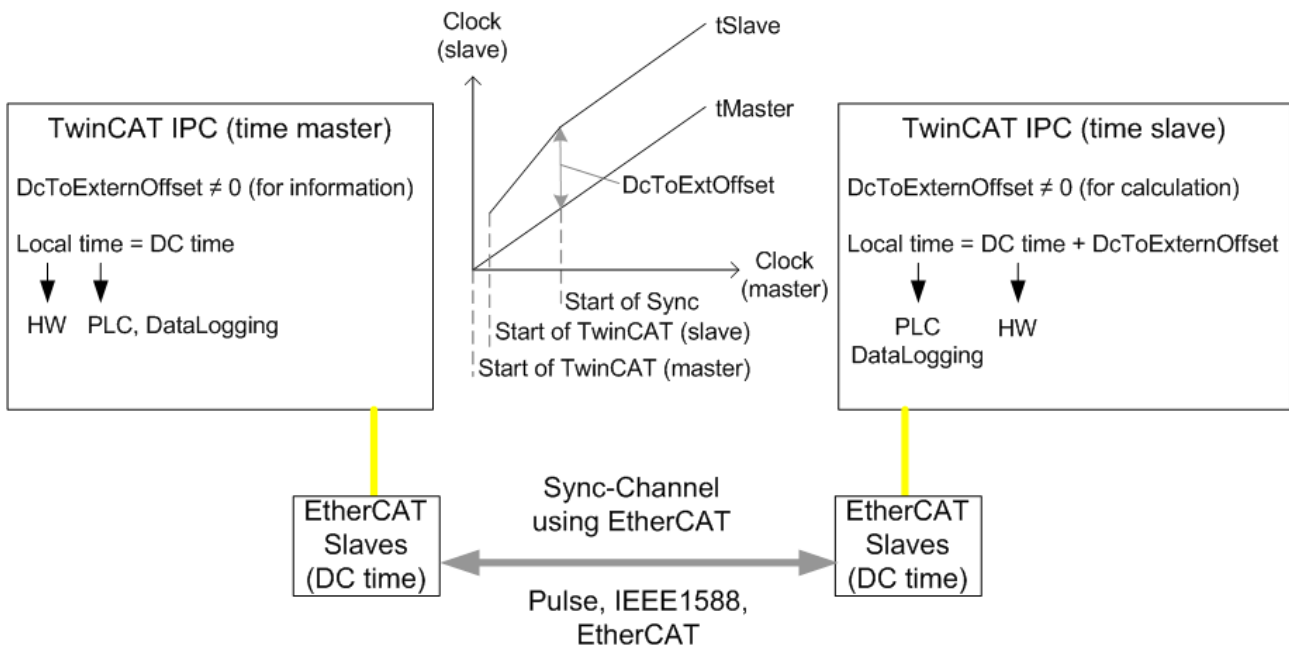


Fig. 295: Synchronization of 2 TwinCAT IPCs with the aid of EtherCAT components

**i Using the synchronized time**

In the adjusted station the "other" time from the master PC is known through:

**Synchronized DC time = local DC time + offset**

This synchronized time can now be used for data logging. The local DC time must still be used for tasks related to the respective station hardware (EtherCAT slaves, terminals).

**Cascading of synchronized TwinCAT systems**

We advise against cascading of several time-synchronized TwinCAT systems. However, please note that simple cascading already occurs if a TwinCAT system is controlled by an external clock based on GPS and transfers its local time to a subordinate EtherCAT system via an EL6692 bridge terminal.

In the subordinate systems the respective DcToExt offset relative to the higher-level systems must be taken into account.

**Synchronized DC time = local DC time + DcToExtOffset<sub>local</sub> + Σ DcToExtOffset<sub>higher-level</sub>**

The respective higher-level DcToExtOffsets can be transported through network variables, ADS, via the EL6692 bridge terminal or any other channels. The subordinate system must take these offsets into account.

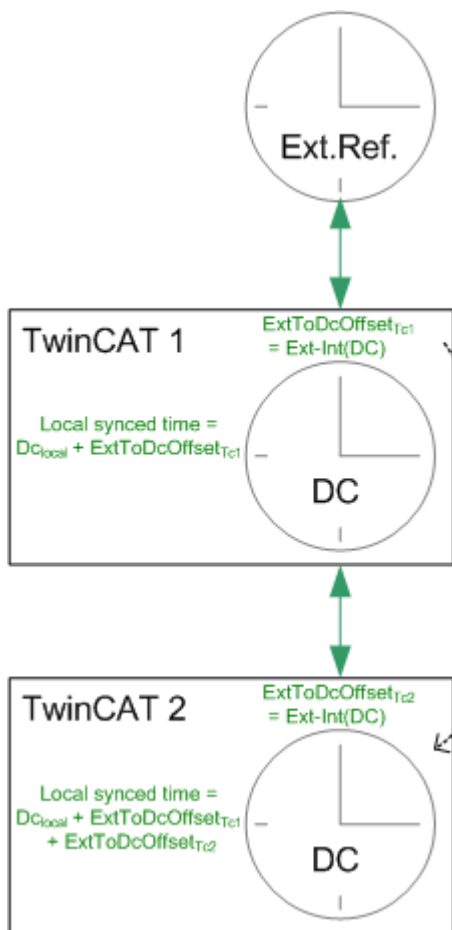


Fig. 296: Cascade consisting of controlled TwinCAT systems

### 7.3.2 Sample: EL6692 bridge terminal

**i** Using the sample programs

This document contains sample applications of our products for certain areas of application. The application notices provided here are based on typical features of our products and only serve as samples. The notices contained in this document explicitly do not refer to specific applications. The customer is therefore responsible for assessing and deciding whether the product is suitable for a particular application. We accept no responsibility for the completeness and correctness of the source code contained in this document. We reserve the right to modify the content of this document at any time and accept no responsibility for errors and missing information.

In this example, two Beckhoff IPCs with TwinCAT 2.11, b1539, will be synchronized with each other. One PC is the master clock, the second (slave clock) synchronizes its 'time' to the first. As the fieldbus, EtherCAT makes the necessary operating resources available, in particular EtherCAT's own synchronization mechanism, distributed clocks.

The procedure is as explained in the previous chapter.

The following must be observed:

- The master PC works autonomously on the basis of its DC time
- Following the start of TwinCAT, the slave PC readjusts its distributed clocks time to the master IPC:
  - at the start of EtherCAT, the initial offset between the two times is determined.
  - the subsequent adjustment keeps this offset constant and makes it known.
  - the readjustment takes place continuously.
- In the case of failure of the synchronization (interruption of the connection, restart of one of the systems), the behavior is as follows:


- If the adjustment restarts in the slave PC, a new offset is calculated there and made known.
- the application must therefore continuously observe this offset.
- The local DC time must still be used for tasks related to the respective station hardware (EtherCAT slaves, terminals).
- The EtherCAT cycle time must be identical in both systems.
- If different configurations are used in the two systems, i.e. the number, type and/or order of the slaves is different, the automatically calculated shift times will also differ.  
Sample: Both systems use an EL2202-0100, which are both supposed to switch their output at the same time. A constant difference is measured, since different output shift times were calculated. In the system with the smaller output shift time the shift time of the other system should be entered.

### NOTE

#### Effect on devices when changing the shift times

Side effects relating to the function of the other slaves when the shift times are changed should be taken into account!

- In a controlled system the time offset between the systems is subject to certain fluctuations.

 Sample program (<https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/2469158155.zip>), TwinCAT 2.11

Pay attention in the program to the use of 'signed' and 'unsigned' 64-bit variables as required.

#### Topology

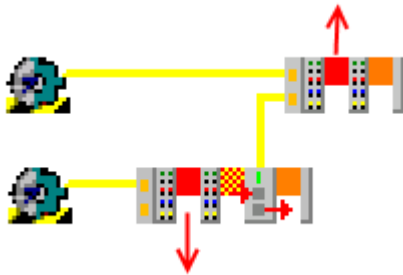


Fig. 297: Topology of the sample program

Station Master: EK1100, EL2202, EL6692

Station Slave: EK1100, EL2202

In this example, the EL6692 is synchronized in the direction *PrimarySide* --> *SecondarySide* (RJ45 connection). Synchronization in the other direction is also possible.

#### EL6692 documentation

Please note the information in the EL6692 documentation regarding the system behavior of this terminal.

#### Demo program

In this demo program, the slave's own local DC time from the ReferenceClock in the EtherCAT strand is offset by the time difference to the external synchronization device. This offset therefore only makes sense on a platform that is a synchronization slave to one master.

The synchronization route can be

- another EtherCAT system; means: Beckhoff EL6692 bridge terminal (this example)
- an IEEE1588 system; means: Beckhoff EL6688 PTP terminal
- an arbitrary timer with time information (GPS, radio clock); means: TwinCAT 'External Synchronization' supplement

The principle:

TwinCAT cyclically receives (e.g. every second) a pair (64-bit, unit 1 ns) made up of an internal (DC) and an external time stamp. These two time stamps are originally obtained simultaneously. The offset between the two time bases is calculated from the initial difference and made known in the System Manager | EtherCAT device | InfoData. Furthermore, the slave TwinCAT readjusts its own local DC time from the course of the two time stamps with respect to each other.

Calculations:

- current control deviation =  $DcToExtOffset - (external\ time\ stamp - internal\ time\ stamp)$   
This value ('signed', 64-bit) is compared to an application-specific limit; the validity of the time is output if within the limit
- local synchronized time = local DC time +  $DcToExtOffset$   
This 'nuLocalTime' ('unsigned', 64-bit) can now be used for data logging and events with a time reference to the master PC clock.

### Setting up TwinCAT 2.11

In the following procedure the complete system is set up as follows:

- EL6692 primary side (E-bus): Sync Master (i.e. reference clock)
- EL6692 secondary side (RJ45 sockets): Sync Slave (i.e. synchronized side)

The synchronization direction of the time can be set up the other way around; the instructions must then be followed analogously.

#### Sync Master side

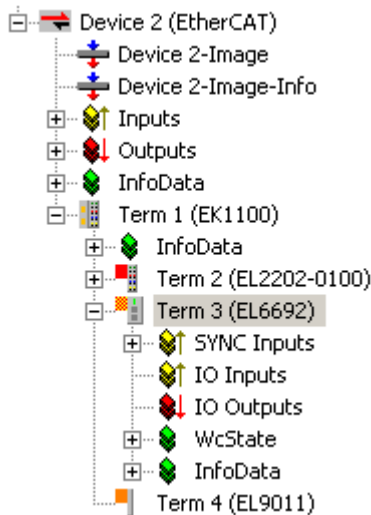


Fig. 298: Device on the master side

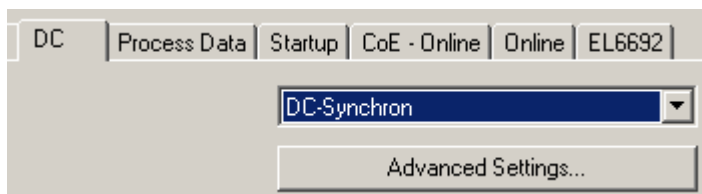


Fig. 299: Set the EL6692 PrimarySide to DC

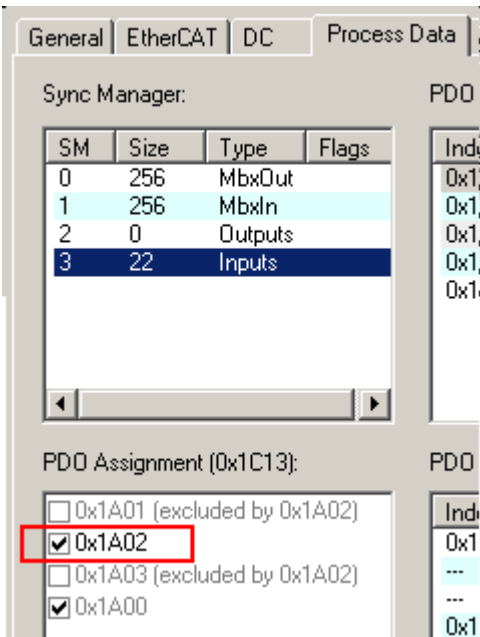


Fig. 300: Activate PDO 0x1A02 to display the time stamps

**Timestamp PDO**

**i** The activation of the timestamp PDO indicates to the TwinCAT software on the respective side that this side is to be synchronized, i.e. that it is the Sync Slave. It is not necessary to activate the timestamp PDO on the Sync Master side (i.e. the side that represents the reference clock).

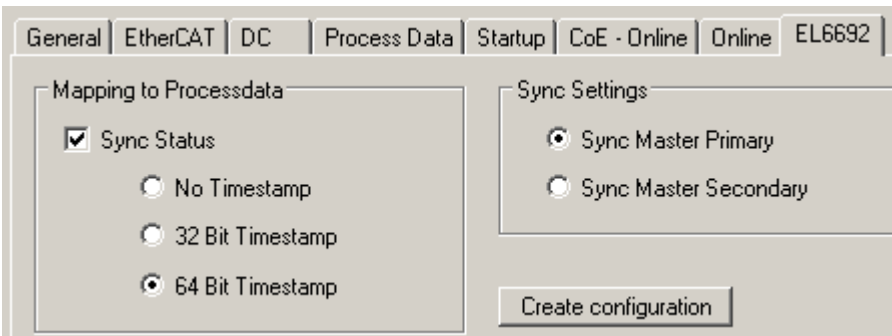


Fig. 301: Set the synchronization direction on the PrimarySide; in this case SyncSettings: Primary --> Secondary



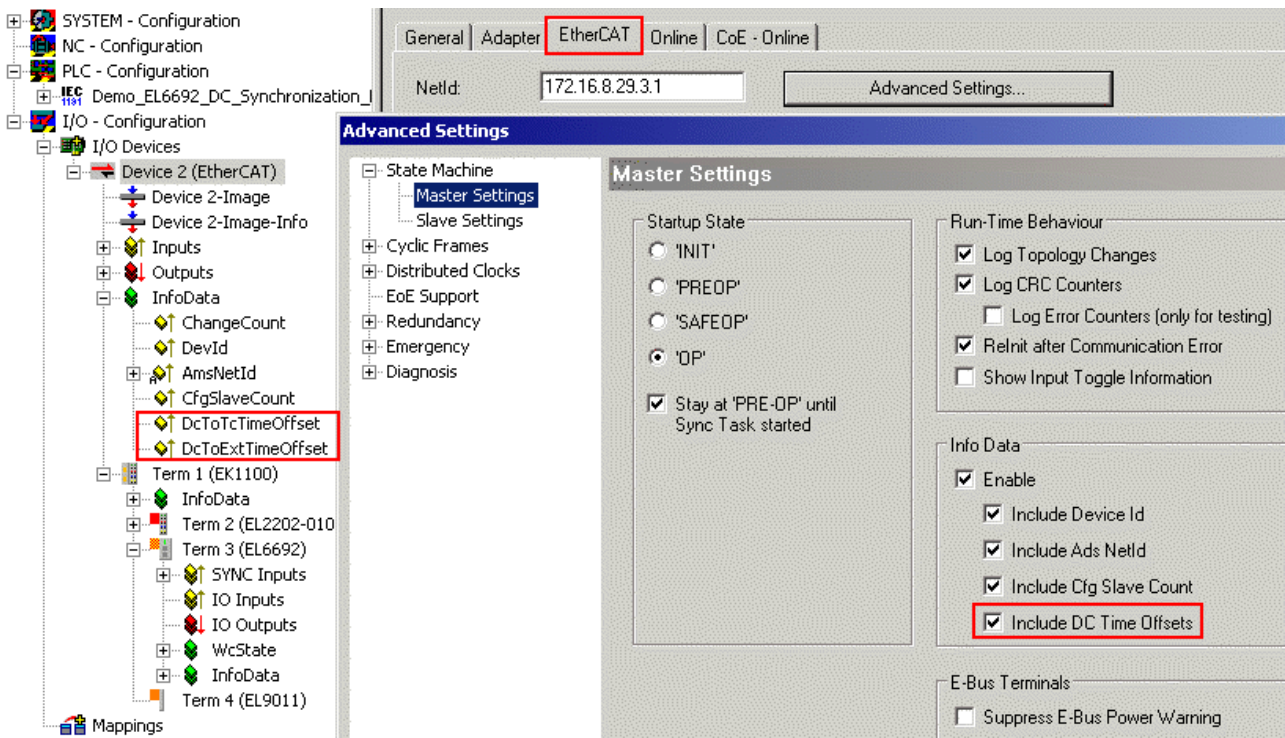


Fig. 302: Activate the display of the DC offsets in the EtherCAT master; can be evaluated on the master side

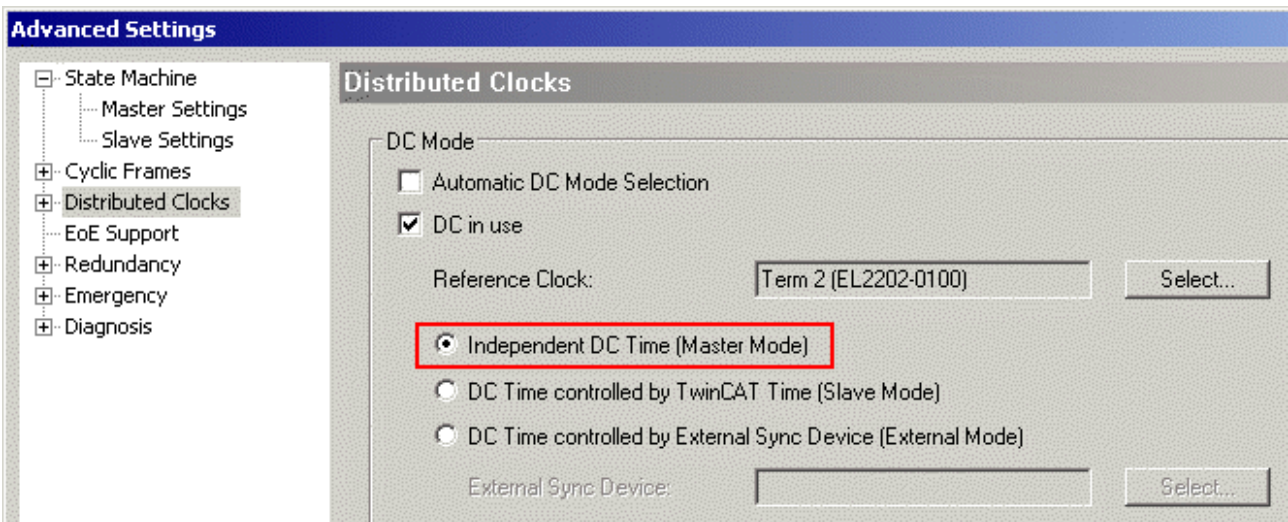


Fig. 303: Master PC works with its own ReferenceClock as a basis

TwinCAT can now be activated and started on this side. All devices must be in OP, WorkingCounter = 0, no LostFrames. The EL6692 time stamps on the PrimarySide remain at 0, because the SecondarySide has not yet been configured.

**Sync Slave side**

The EL6692, *SecondarySide* is set to DC and 0x1A02 according to Figs. *Set the EL6692 PrimarySide to DC and Activate PDO 0x1A02 to display the time stamps.*

After reloading the configuration (or restarting in *ConfigMode*, *FreeRun*), the synchronization direction can be read out by means of *GetConfiguration* on the *SecondarySide*, see the fig. *SecondarySide of the EL6692.*

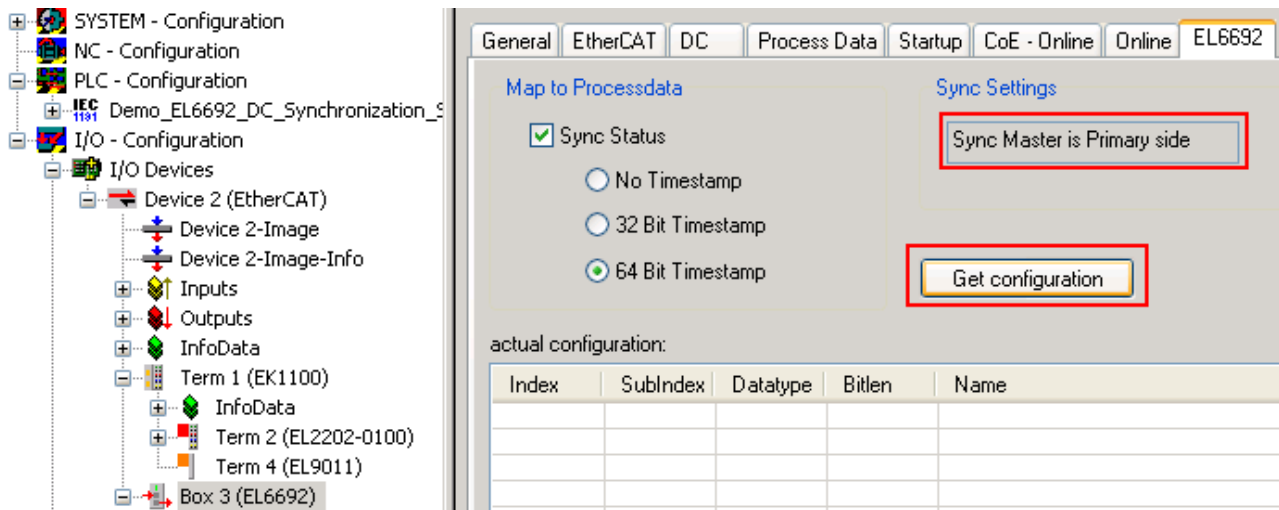


Fig. 304: SecondarySide of the EL6692

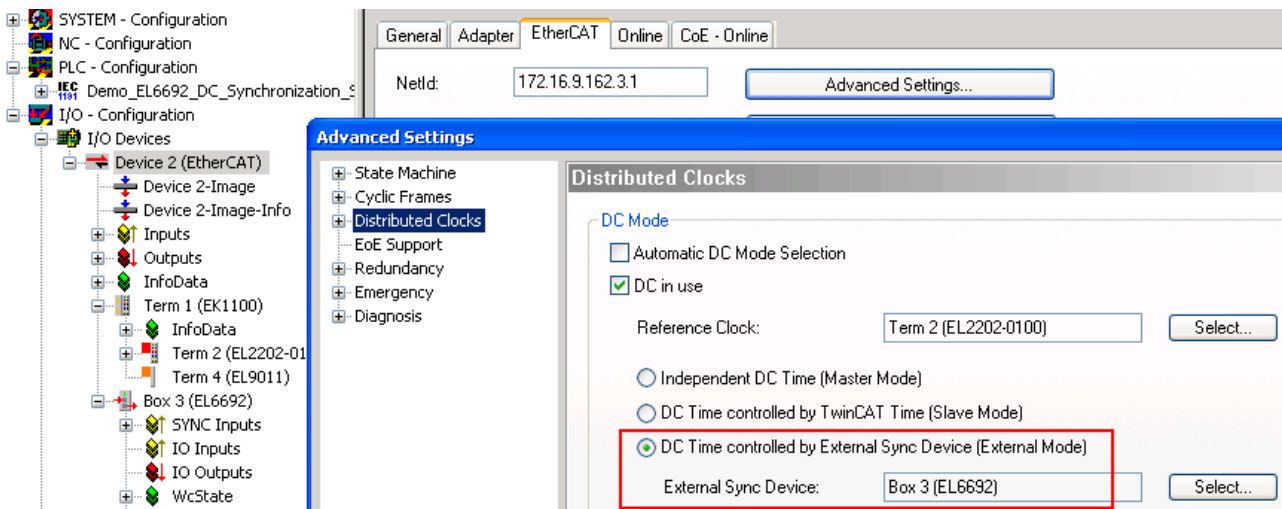


Fig. 305: EtherCAT master settings, slave side

After the restart, the DC function of the EL6692 is known to the EtherCAT master; therefore, it now offers this EL6692 as an *ExternalSyncDevice* in the DC dialogue.

The linking of the following variables is necessary for the evaluations; see the fig. *Slave side*.

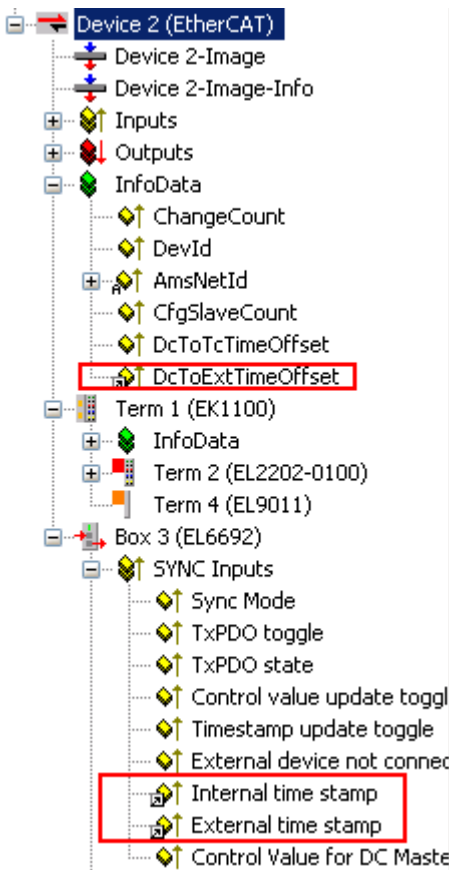


Fig. 306: Slave side

**NOTE**

**Demo program**

The following screenshots and information refer exclusively to the PLC demo program discussed here and the sample code it contains, and not to the analysis functions of the system manager.  
 See also the [note \[▶ 253\]](#).

On the slave side, the start of the synchronization can be observed with the incorporated visualization.

**Synchronization started**

local DC time (world time format):	2009-11-05-13:53:20.220000000
local synchronized time (world time format): 2009-11-05-13:53:20.220000000	

**Synchronization in Window  
window: 10000 ns**

**Synchronization  
in Progress**

Fig. 307: Start slave side

Only the local DC time is available on the slave side immediately after the start.

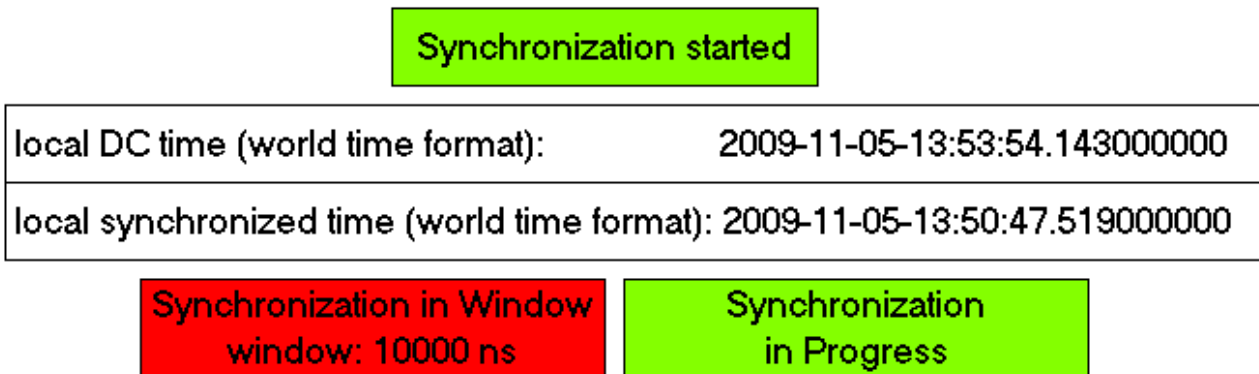


Fig. 308: Time stamp known

Following receipt of the first time stamps via the EL6692, the offset is known; in this case it is around 3 minutes different to the time of the IPC used. Synchronization has begun; in this sample a window of  $\pm 10 \mu\text{s}$  is to be achieved.

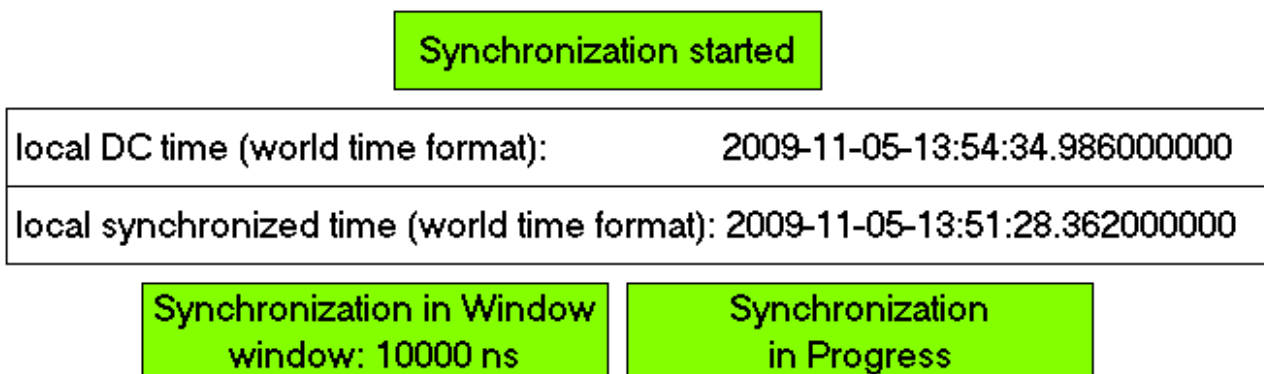


Fig. 309: Synchronization successful

# 8 Configuration of the terminals

## 8.1 General configuration

### 8.1.1 EtherCAT subscriber configuration

In the left-hand window of the TwinCAT 2 System Manager or the Solution Explorer of the TwinCAT 3 Development Environment respectively, click on the element of the terminal within the tree you wish to configure (in the example: EL3751 Terminal 3).

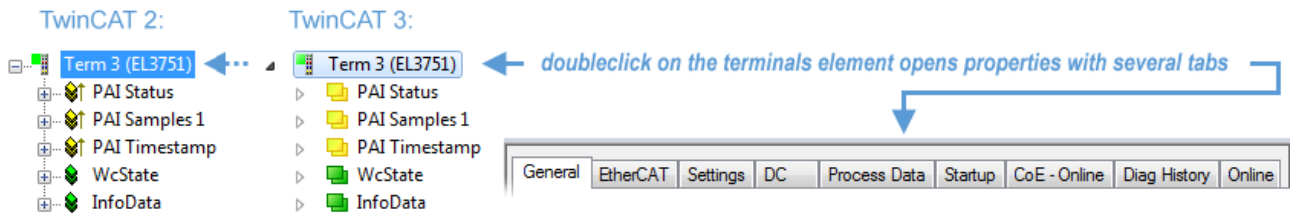


Fig. 310: Branch element as terminal EL3751

In the right-hand window of the TwinCAT System Manager (TwinCAT 2) or the Development Environment (TwinCAT 3), various tabs are now available for configuring the terminal. And yet the dimension of complexity of a subscriber determines which tabs are provided. Thus as illustrated in the example above the terminal EL3751 provides many setup options and also a respective number of tabs are available. On the contrary by the terminal EL1004 for example the tabs “General”, “EtherCAT”, “Process Data” and “Online“ are available only. Several terminals, as for instance the EL6695 provide special functions by a tab with its own terminal name, so “EL6695” in this case. A specific tab “Settings” by terminals with a wide range of setup options will be provided also (e.g. EL3751).

#### “General” tab

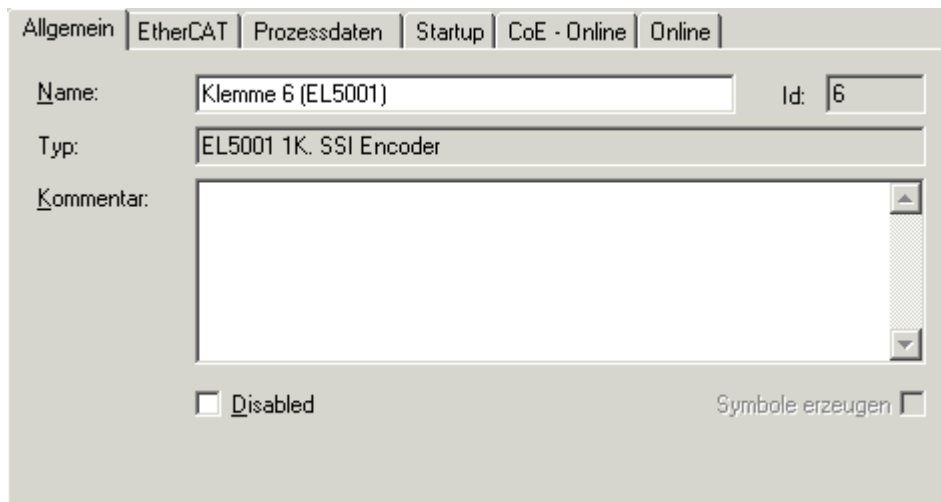


Fig. 311: “General” tab

<b>Name</b>	Name of the EtherCAT device
<b>Id</b>	Number of the EtherCAT device
<b>Type</b>	EtherCAT device type
<b>Comment</b>	Here you can add a comment (e.g. regarding the system).
<b>Disabled</b>	Here you can deactivate the EtherCAT device.
<b>Create symbols</b>	Access to this EtherCAT slave via ADS is only available if this control box is activated.

**“EtherCAT” tab**

Fig. 312: “EtherCAT” tab

<b>Type</b>	EtherCAT device type
<b>Product/Revision</b>	Product and revision number of the EtherCAT device
<b>Auto Inc Addr.</b>	Auto increment address of the EtherCAT device. The auto increment address can be used for addressing each EtherCAT device in the communication ring through its physical position. Auto increment addressing is used during the start-up phase when the EtherCAT master allocates addresses to the EtherCAT devices. With auto increment addressing the first EtherCAT slave in the ring has the address $0000_{\text{hex}}$ . For each further slave the address is decremented by 1 ( $FFFF_{\text{hex}}$ , $FFFE_{\text{hex}}$ etc.).
<b>EtherCAT Addr.</b>	Fixed address of an EtherCAT slave. This address is allocated by the EtherCAT master during the start-up phase. Tick the control box to the left of the input field in order to modify the default value.
<b>Previous Port</b>	Name and port of the EtherCAT device to which this device is connected. If it is possible to connect this device with another one without changing the order of the EtherCAT devices in the communication ring, then this combination field is activated and the EtherCAT device to which this device is to be connected can be selected.
<b>Advanced Settings</b>	This button opens the dialogs for advanced settings.

The link at the bottom of the tab points to the product page for this EtherCAT device on the web.

**“Process Data” tab**

Indicates the configuration of the process data. The input and output data of the EtherCAT slave are represented as CANopen process data objects (**P**rocess **D**ata **O**bjects, PDOs). The user can select a PDO via PDO assignment and modify the content of the individual PDO via this dialog, if the EtherCAT slave supports this function.

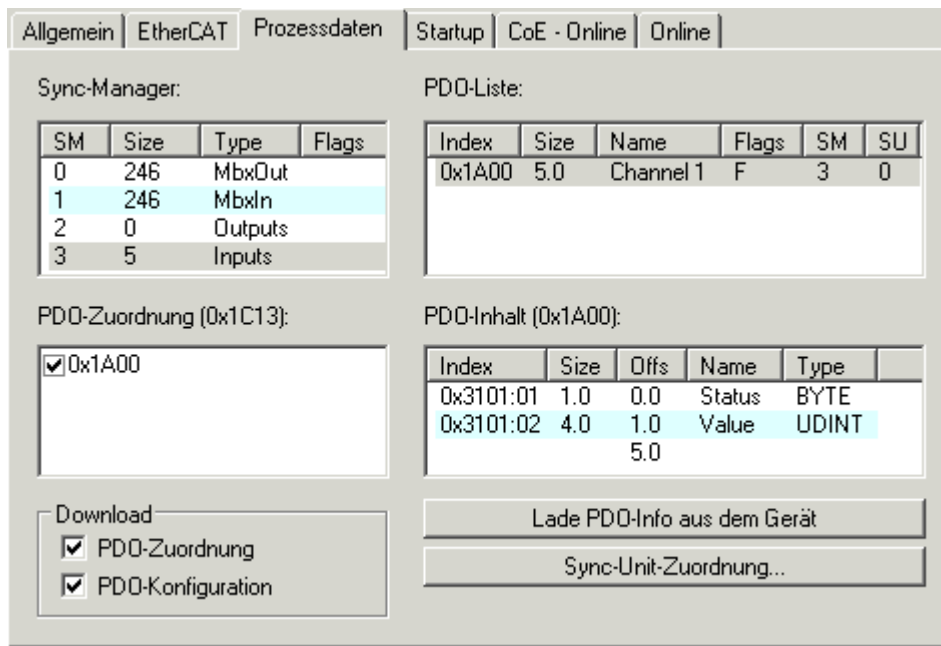


Fig. 313: "Process Data" tab

The process data (PDOs) transferred by an EtherCAT slave during each cycle are user data which the application expects to be updated cyclically or which are sent to the slave. To this end the EtherCAT master (Beckhoff TwinCAT) parameterizes each EtherCAT slave during the start-up phase to define which process data (size in bits/bytes, source location, transmission type) it wants to transfer to or from this slave. Incorrect configuration can prevent successful start-up of the slave.

For Beckhoff EtherCAT EL, ES, EM, EJ and EP slaves the following applies in general:

- The input/output process data supported by the device are defined by the manufacturer in the ESI/XML description. The TwinCAT EtherCAT Master uses the ESI description to configure the slave correctly.
- The process data can be modified in the System Manager. See the device documentation. Examples of modifications include: mask out a channel, displaying additional cyclic information, 16-bit display instead of 8-bit data size, etc.
- In so-called "intelligent" EtherCAT devices the process data information is also stored in the CoE directory. Any changes in the CoE directory that lead to different PDO settings prevent successful startup of the slave. It is not advisable to deviate from the designated process data, because the device firmware (if available) is adapted to these PDO combinations.

If the device documentation allows modification of process data, proceed as follows (see Figure *Configuring the process data*).

- A: select the device to configure
- B: in the "Process Data" tab select Input or Output under SyncManager (C)
- D: the PDOs can be selected or deselected
- H: the new process data are visible as linkable variables in the System Manager  
The new process data are active once the configuration has been activated and TwinCAT has been restarted (or the EtherCAT master has been restarted)
- E: if a slave supports this, Input and Output PDO can be modified simultaneously by selecting a so-called PDO record ("predefined PDO settings").

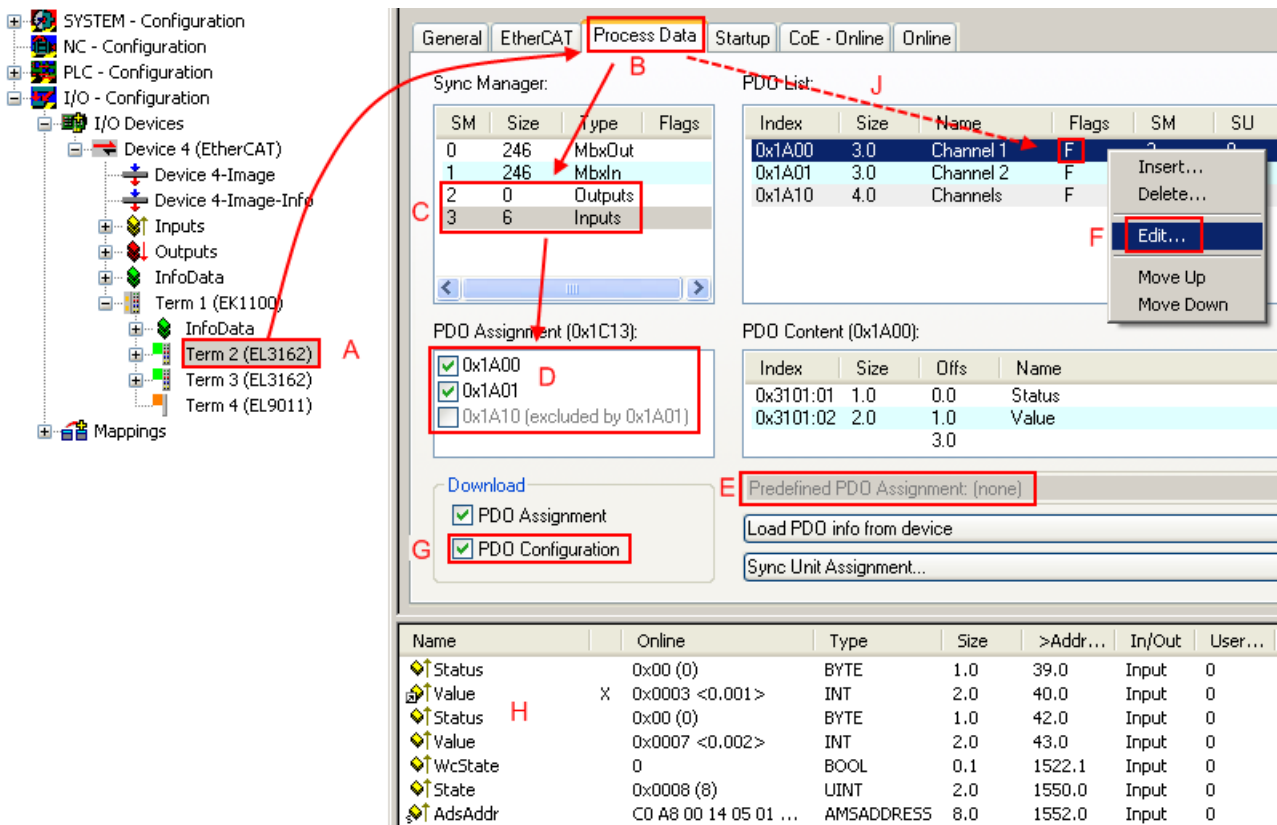


Fig. 314: Configuring the process data

### Manual modification of the process data

According to the ESI description, a PDO can be identified as “fixed” with the flag “F” in the PDO overview (Fig. *Configuring the process data*, J). The configuration of such PDOs cannot be changed, even if TwinCAT offers the associated dialog (“Edit”). In particular, CoE content cannot be displayed as cyclic process data. This generally also applies in cases where a device supports download of the PDO configuration, “G”. In case of incorrect configuration the EtherCAT slave usually refuses to start and change to OP state. The System Manager displays an “invalid SM cfg” logger message: This error message (“invalid SM IN cfg” or “invalid SM OUT cfg”) also indicates the reason for the failed start.

A detailed description [▶ 269] can be found at the end of this section.

### “Startup” tab

The *Startup* tab is displayed if the EtherCAT slave has a mailbox and supports the *CANopen over EtherCAT* (CoE) or *Servo drive over EtherCAT* protocol. This tab indicates which download requests are sent to the mailbox during startup. It is also possible to add new mailbox requests to the list display. The download requests are sent to the slave in the same order as they are shown in the list.



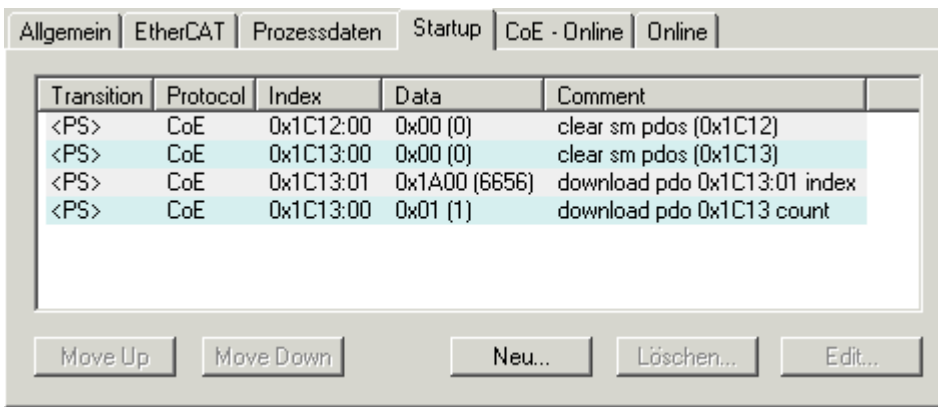


Fig. 315: “Startup” tab

Column	Description
Transition	Transition to which the request is sent. This can either be <ul style="list-style-type: none"> <li>the transition from pre-operational to safe-operational (PS), or</li> <li>the transition from safe-operational to operational (SO).</li> </ul> If the transition is enclosed in “<>” (e.g. <PS>), the mailbox request is fixed and cannot be modified or deleted by the user.
Protocol	Type of mailbox protocol
Index	Index of the object
Data	Date on which this object is to be downloaded.
Comment	Description of the request to be sent to the mailbox

- Move Up**      This button moves the selected request up by one position in the list.
- Move Down**      This button moves the selected request down by one position in the list.
- New**      This button adds a new mailbox download request to be sent during startup.
- Delete**      This button deletes the selected entry.
- Edit**      This button edits an existing request.

**“CoE - Online” tab**

The additional *CoE - Online* tab is displayed if the EtherCAT slave supports the *CANopen over EtherCAT* (CoE) protocol. This dialog lists the content of the object list of the slave (SDO upload) and enables the user to modify the content of an object from this list. Details for the objects of the individual EtherCAT devices can be found in the device-specific object descriptions.

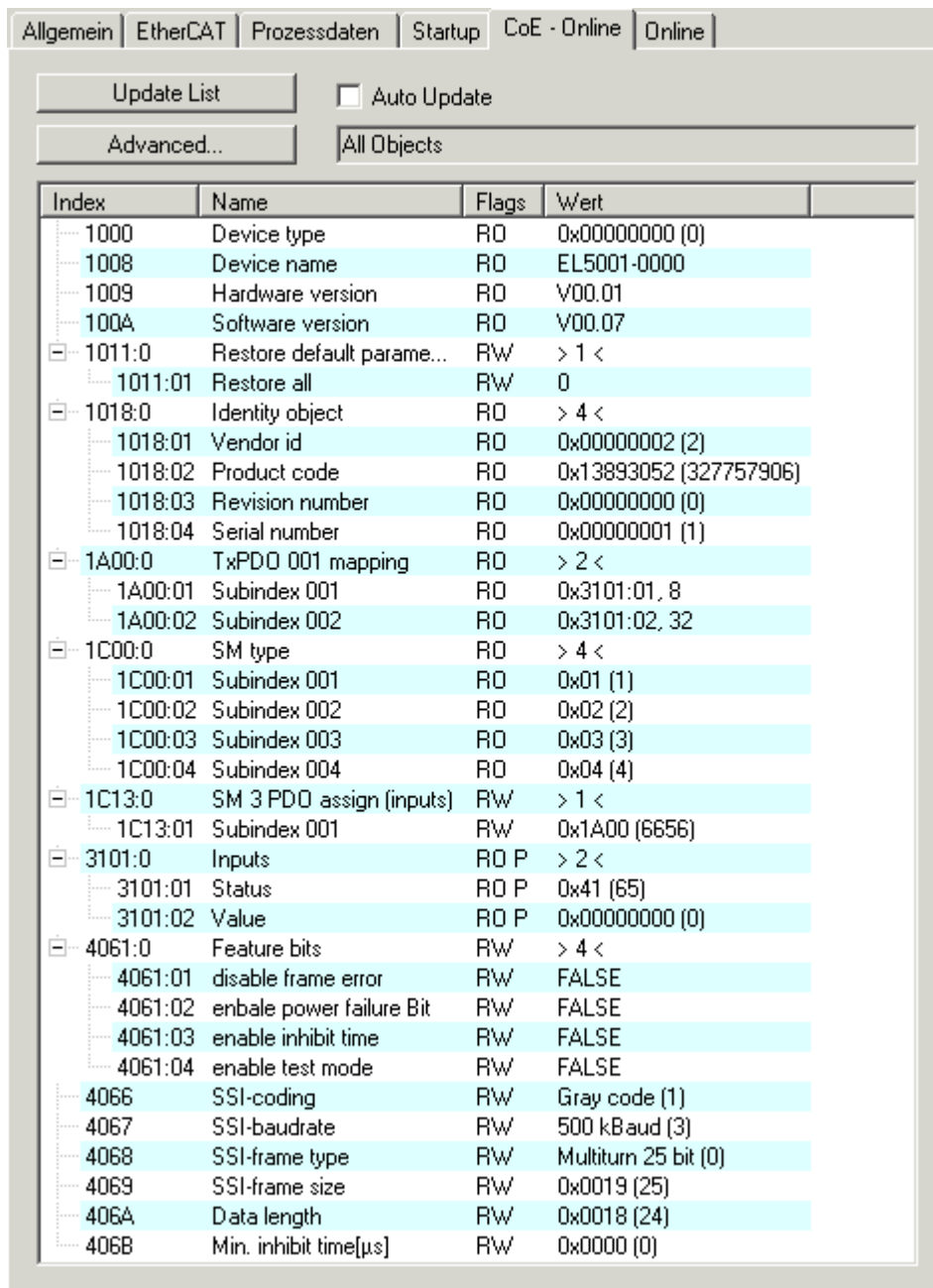


Fig. 316: “CoE - Online” tab

**Object list display**

Column	Description
Index	Index and sub-index of the object
Name	Name of the object
Flags	RW The object can be read, and data can be written to the object (read/write)
	RO The object can be read, but no data can be written to the object (read only)
	P An additional P identifies the object as a process data object.
Value	Value of the object

**Update List** The *Update list* button updates all objects in the displayed list

**Auto Update** If this check box is selected, the content of the objects is updated automatically.

**Advanced** The *Advanced* button opens the *Advanced Settings* dialog. Here you can specify which objects are displayed in the list.

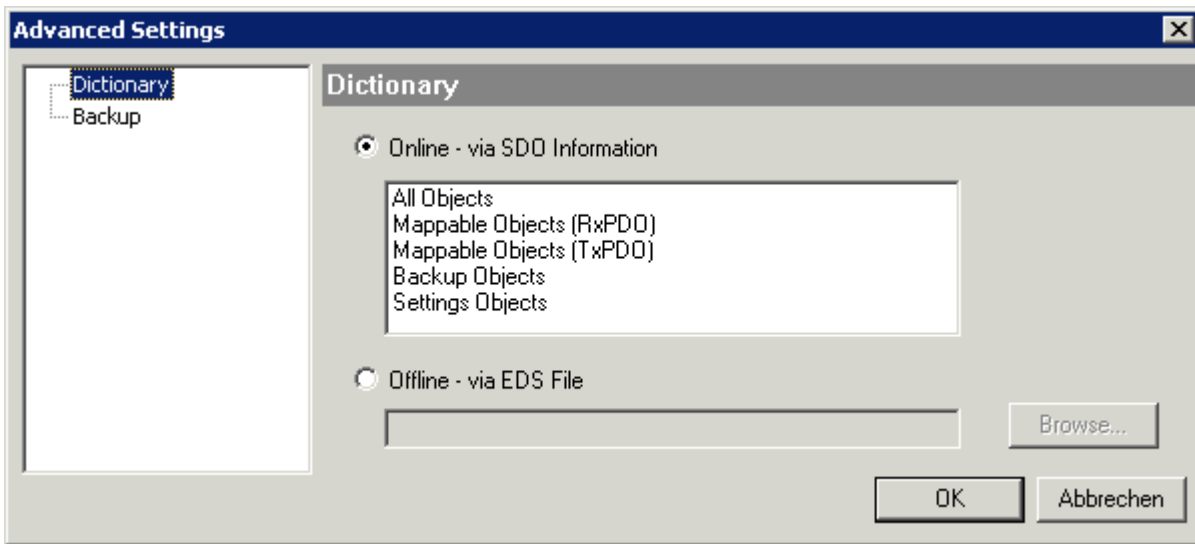


Fig. 317: Dialog “Advanced settings”

**Online - via SDO Information** If this option button is selected, the list of the objects included in the object list of the slave is uploaded from the slave via SDO information. The list below can be used to specify which object types are to be uploaded.

**Offline - via EDS File** If this option button is selected, the list of the objects included in the object list is read from an EDS file provided by the user.

“Online” tab

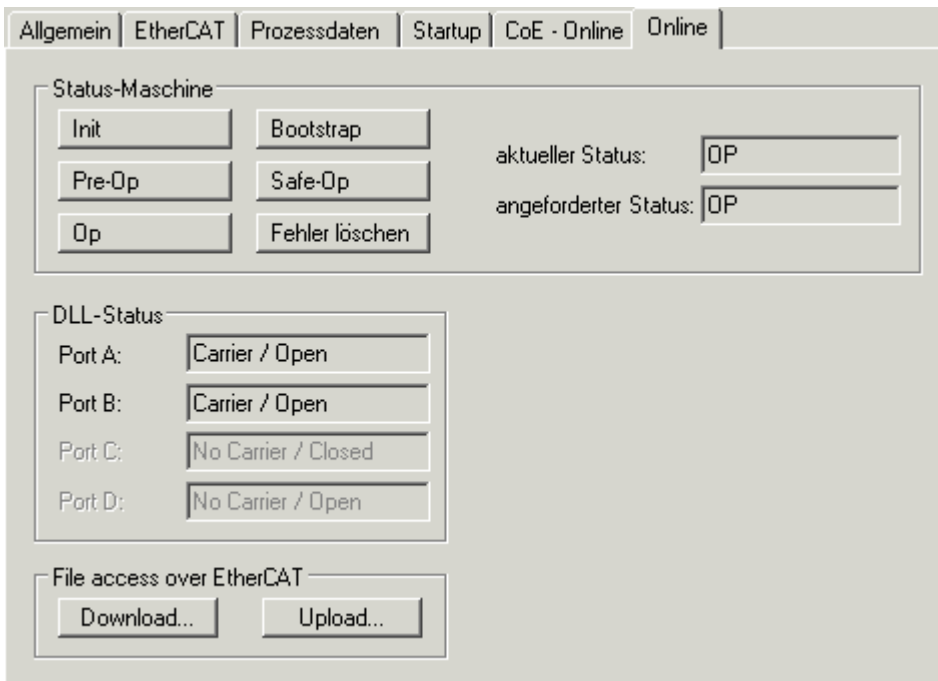


Fig. 318: “Online” tab

**State Machine**

- Init** This button attempts to set the EtherCAT device to the *Init* state.
- Pre-Op** This button attempts to set the EtherCAT device to the *pre-operational* state.
- Op** This button attempts to set the EtherCAT device to the *operational* state.
- Bootstrap** This button attempts to set the EtherCAT device to the *Bootstrap* state.
- Safe-Op** This button attempts to set the EtherCAT device to the *safe-operational* state.
- Clear Error** This button attempts to delete the fault display. If an EtherCAT slave fails during change of state it sets an error flag.  
  
Example: An EtherCAT slave is in PREOP state (pre-operational). The master now requests the SAFEOP state (safe-operational). If the slave fails during change of state it sets the error flag. The current state is now displayed as ERR PREOP. When the *Clear Error* button is pressed the error flag is cleared, and the current state is displayed as PREOP again.
- Current State** Indicates the current state of the EtherCAT device.
- Requested State** Indicates the state requested for the EtherCAT device.

**DLL Status**

Indicates the DLL status (data link layer status) of the individual ports of the EtherCAT slave. The DLL status can have four different states:

Status	Description
No Carrier / Open	No carrier signal is available at the port, but the port is open.
No Carrier / Closed	No carrier signal is available at the port, and the port is closed.
Carrier / Open	A carrier signal is available at the port, and the port is open.
Carrier / Closed	A carrier signal is available at the port, but the port is closed.

**File Access over EtherCAT**

- Download** With this button a file can be written to the EtherCAT device.
- Upload** With this button a file can be read from the EtherCAT device.

**“DC” tab (Distributed Clocks)**

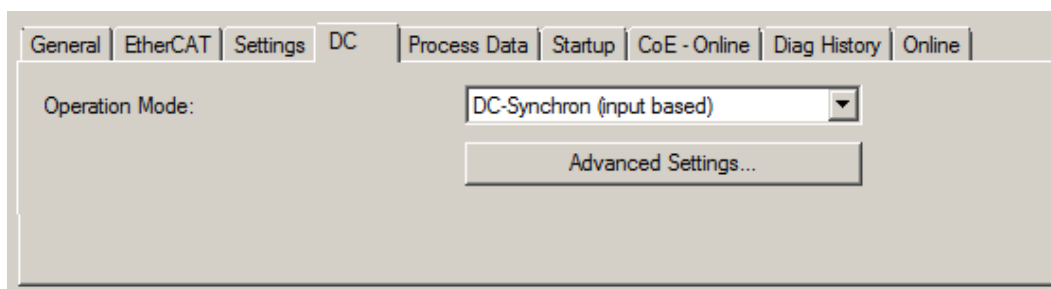


Fig. 319: “DC” tab (Distributed Clocks)

- Operation Mode** Options (optional):
  - FreeRun
  - SM-Synchron
  - DC-Synchron (Input based)
  - DC-Synchron
- Advanced Settings...** Advanced settings for readjustment of the real time determinant TwinCAT-clock

Detailed information to Distributed Clocks is specified on <http://infosys.beckhoff.com>:

**Fieldbus Components** → EtherCAT Terminals → EtherCAT System documentation → EtherCAT basics → Distributed Clocks

### 8.1.1.1 Detailed description of Process Data tab

#### Sync Manager

Lists the configuration of the Sync Manager (SM).

If the EtherCAT device has a mailbox, SM0 is used for the mailbox output (MbxOut) and SM1 for the mailbox input (MbxIn).

SM2 is used for the output process data (outputs) and SM3 (inputs) for the input process data.

If an input is selected, the corresponding PDO assignment is displayed in the *PDO Assignment* list below.

#### PDO Assignment



PDO assignment of the selected Sync Manager. All PDOs defined for this Sync Manager type are listed here:

- If the output Sync Manager (outputs) is selected in the Sync Manager list, all RxPDOs are displayed.
- If the input Sync Manager (inputs) is selected in the Sync Manager list, all TxPDOs are displayed.

The selected entries are the PDOs involved in the process data transfer. In the tree diagram of the System Manager these PDOs are displayed as variables of the EtherCAT device. The name of the variable is identical to the *Name* parameter of the PDO, as displayed in the PDO list. If an entry in the PDO assignment list is deactivated (not selected and greyed out), this indicates that the input is excluded from the PDO assignment. In order to be able to select a greyed out PDO, the currently selected PDO has to be deselected first.

#### **i** Activation of PDO assignment

- ✓ If you have changed the PDO assignment, in order to activate the new PDO assignment,
  - a) the EtherCAT slave has to run through the PS status transition cycle (from pre-operational to safe-operational) once (see [Online tab \[▶ 267\]](#)),
  - b) and the System Manager has to reload the EtherCAT slaves

(  button for TwinCAT 2 or  button for TwinCAT 3)

#### PDO list

List of all PDOs supported by this EtherCAT device. The content of the selected PDOs is displayed in the *PDO Content* list. The PDO configuration can be modified by double-clicking on an entry.

Column	Description	
Index	PDO index.	
Size	Size of the PDO in bytes.	
Name	Name of the PDO. If this PDO is assigned to a Sync Manager, it appears as a variable of the slave with this parameter as the name.	
Flags	F	Fixed content: The content of this PDO is fixed and cannot be changed by the System Manager.
	M	Mandatory PDO. This PDO is mandatory and must therefore be assigned to a Sync Manager! Consequently, this PDO cannot be deleted from the <i>PDO Assignment</i> list
SM	Sync Manager to which this PDO is assigned. If this entry is empty, this PDO does not take part in the process data traffic.	
SU	Sync unit to which this PDO is assigned.	

#### PDO Content

Indicates the content of the PDO. If flag F (fixed content) of the PDO is not set the content can be modified.

## Download

If the device is intelligent and has a mailbox, the configuration of the PDO and the PDO assignments can be downloaded to the device. This is an optional feature that is not supported by all EtherCAT slaves.

## PDO Assignment

If this check box is selected, the PDO assignment that is configured in the PDO Assignment list is downloaded to the device on startup. The required commands to be sent to the device can be viewed in the [Startup \[► 264\]](#) tab.

## PDO Configuration

If this check box is selected, the configuration of the respective PDOs (as shown in the PDO list and the PDO Content display) is downloaded to the EtherCAT slave.

## 8.1.2 Fast mode

The Fast Mode in Beckhoff EtherCAT Terminals has developed historically and is an operation mode with which EL Terminals, primarily the EL3xxx and EL4xxx groups (analog input/output terminals), can be operated with a considerably faster conversion time. Hence, an analog input value can be converted more quickly/more often or output via the controller accordingly. This is achieved at the expense of other features, therefore consideration is required.

If an EL Terminal supports this mode, this is indicated in the relevant documentation.

There are 2 groups of terminals that support the FastMode:

- Prototypes of EL3xx2 and EL4xx2, with product launch before 2009:  
You can switch the two-channel analog input and output terminals into *Fast Mode* by **switching off the second channel**. When operating with a single channel (*Fast Mode*) the terminal's conversion time is about one third less than it is when operating two channels. You can find the precise figures for the conversion times in single-channel and two-channel operation in the technical data for the each particular terminal.
- EL3xxx and EL4xxx from year of manufacture 2009  
**CoE access can be deactivated** in these terminals (if this is possible according to the documentation). All existing channels can convert faster as a result of this.

Here is an example for each group.

### FastMode by means of channel deactivation

#### Sample 1

You can switch the second input channel on and off on the *Process data* tab of the EL3101 under *PDO assignment* with the aid of the check box (see the red arrow).

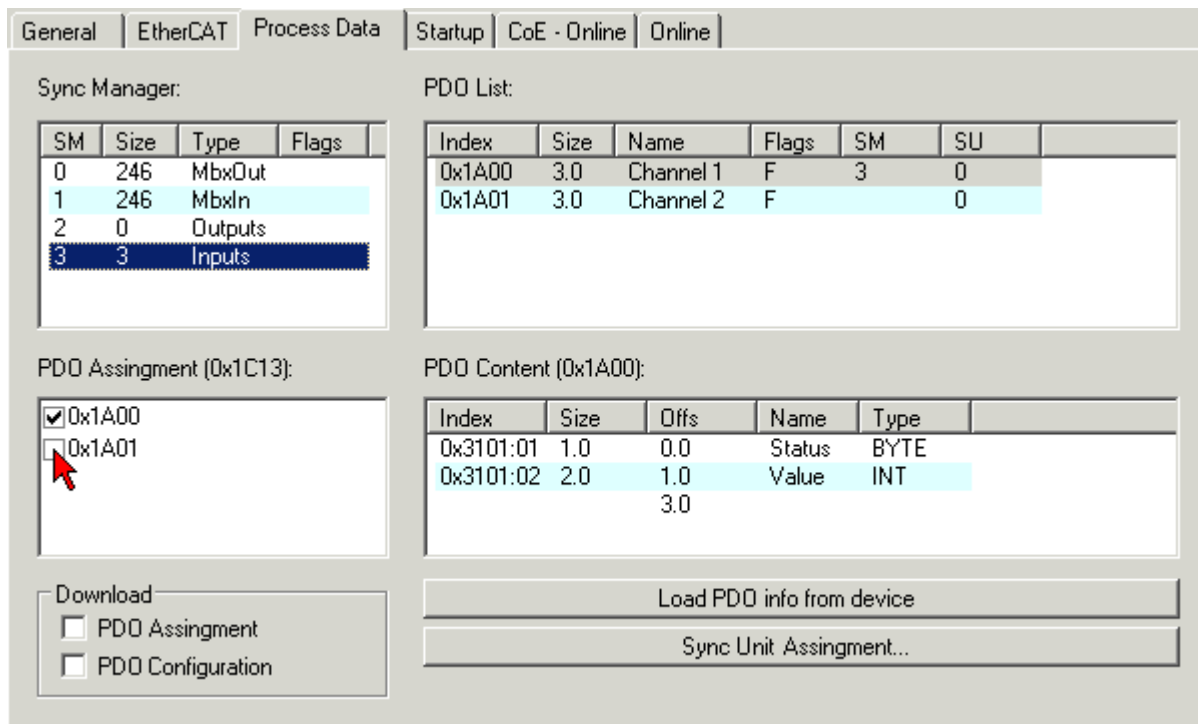


Fig. 320: "Process data" tab

**Sample 2**

You can switch the second output channel on and off on the *Process data* tab of the EL4101 under *PDO assignment* with the aid of the check box (see the red arrow).

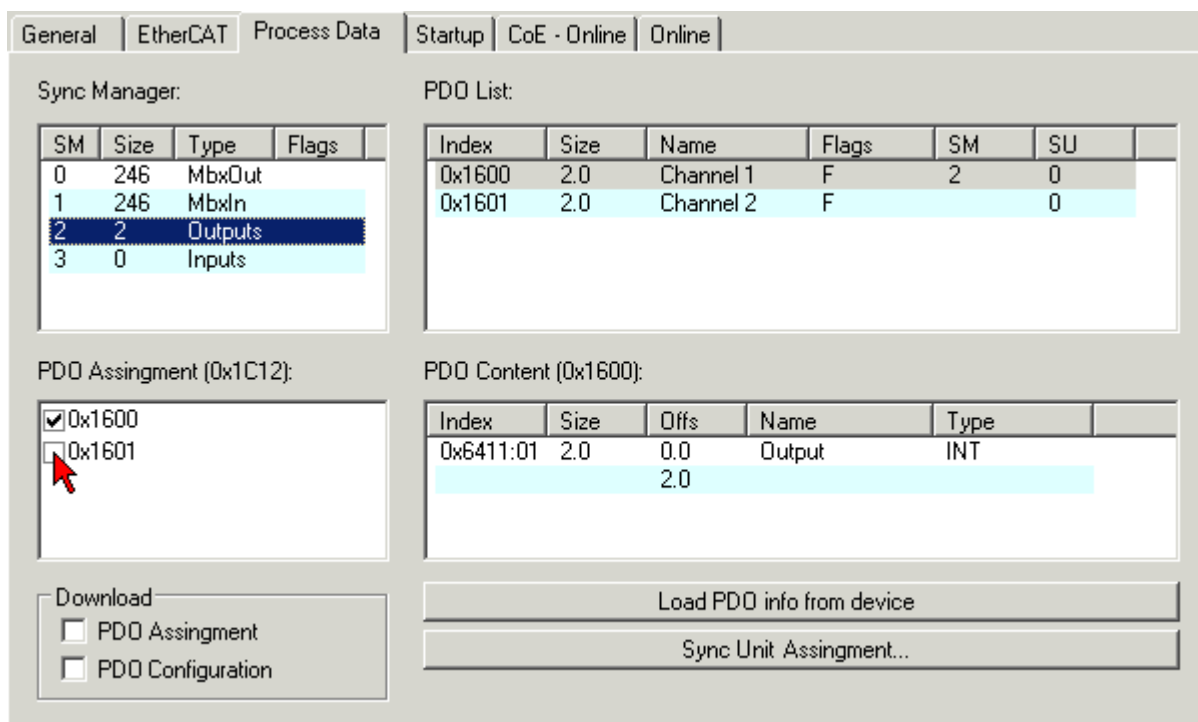


Fig. 321: "Process data" tab

**FastMode by means of CoE deactivation**

In order to deactivate the CoE support, an

PS CoE 0x1C33:01 0x8001 (32769) Sync mode

Fig. 322: entry

must be entered next to the terminal in the start-up list in the System Manager. This deactivates the CoE later in SAFEOP and OP.

More generally, the mailbox traffic of this terminal is turned off by this FastMode.

The CoE access can be reactivated by writing the original value or, for example, 0x00 in the PREOP phase after CoE 0x1C33:01. See the entries in the [Synchronization modes \[► 197\]](#) overview regarding this.

## 8.2 Advanced configuration

### 8.2.1 Behavior

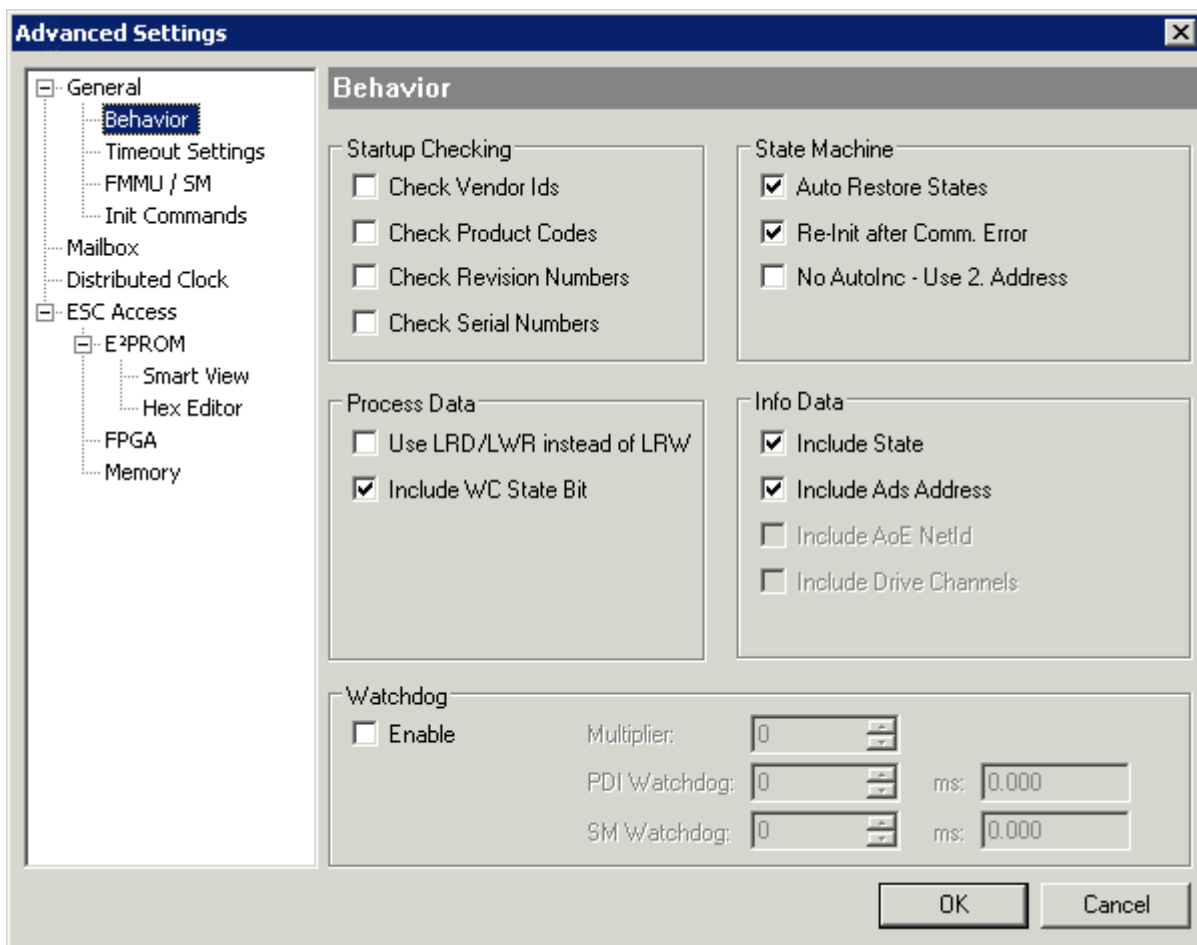


Fig. 323: Advanced Settings: "Behavior" dialog

#### Startup Checking

Here you can specify which slave information the EtherCAT master should check during startup.

#### Check Vendor IDs

Here you can specify that the EtherCAT master should compare the vendor ID of each slave with the configured vendor ID (default: inactive).



**Check Product Codes**

Here you can specify that the EtherCAT master should compare the product code of each slave with the configured product code (default: inactive).

**Check Revision Numbers**

Here you can specify that the EtherCAT master should compare the revision number of each slave with the configured revision number (default: inactive).

**Check Serial Numbers**

Here you can specify that the EtherCAT master should compare the serial number of each slave with the configured serial number (default: inactive).

**State Machine****Auto Restore States**

Here you can specify that the EtherCAT master should try and restore the state of an EtherCAT slave automatically (default: active). If an EtherCAT slave changes its state from an error state (ERR SAVE-OP, ERR OP etc.) to a regular state (SAFE-OP, OP etc.), the master tries to change the slave state to the master state.

**Re-Init after Communication Error**

Here you can specify that the EtherCAT master should reset the slave to state *Init* after a communication error (default: active).

**No AutoInc - Use 2<sup>nd</sup> address**

If this checkbox is activated, the EtherCAT master addresses this EtherCAT slave during the start-up phase not via the position in the EtherCAT ring, but reads a fixed address from the slave (default: inactive).

**Process data****Use LRD/LWR instead of LRW**

Here you can specify that an LRD command (Logical Read) should be used for reading the inputs and an LWR command (Logical Write) for setting the outputs (default: active). Otherwise an LRW command (Logical Read/Write) is used for reading the inputs and setting the outputs.

**Include WC State Bit**

Here you can specify that an input variable that displays the working counter state of the EtherCAT slave is added to its process image (default: active).

**Info Data**

To use these options, Info Data must be activated in the master settings.

**Include state**

Here you can specify that the input variable *State* is added to the Info Data entry for each EtherCAT slave (default: active). This variable contains the current EtherCAT state and the link state of an EtherCAT slave.

**Include ADS address**

Here you can specify that the input variable *AdsAddress* should be added to the Info Data entry for each EtherCAT slave (default: active for all EtherCAT slaves that support mailbox protocols such as CoE (CANopen over EtherCAT) or SoE (servo over EtherCAT)).

**Include AoE NetID**

If this checkbox is activated, the NetID for ADS over EtherCAT is added (default: inactive).

**Include Drive Channels**

Here you can specify that the input variable *ChnX* (X = channel number) is added to the Info Data entry for each EtherCAT slave (default: inactive).

**Watchdog**

Here you can configure the watchdog behavior.

**Enable**

If this checkbox is activated, the watchdogs are switched on.

**Multiplier**

Multiplier

**PDI Watchdog**

Watchdog for the process data interface

**SM Watchdog**

Watchdog for the Sync Manager

**8.2.1.1 Displaying channels in InfoData (Include Channels)**

It may be helpful if the controller/PLC knows which channel of an IO module (terminal, box) it is using. Examples:

- A 4-channel EL3104 analog input terminal is used. In the controller, a function block (FB) accepts the data. The FB expects the information on which channel of the EL3104 it is linked to, e.g. the 3rd or 4th channel.
- A module has technologically different channels and the control expects the information to which technology channel it is linked.  
Example: In the standard setting, the EL3356 analog bridge terminal operates as a 1-channel load terminal and outputs the measured weight in kg. Alternatively, it can be reconfigured to 2-channel voltage measurement (supply voltage and bridge voltage). The terminal thus offers a total of three channels for use (although not all of them simultaneously).

Solution: In TwinCAT 2.11 and 3, the channels of a module can be displayed in the info data.

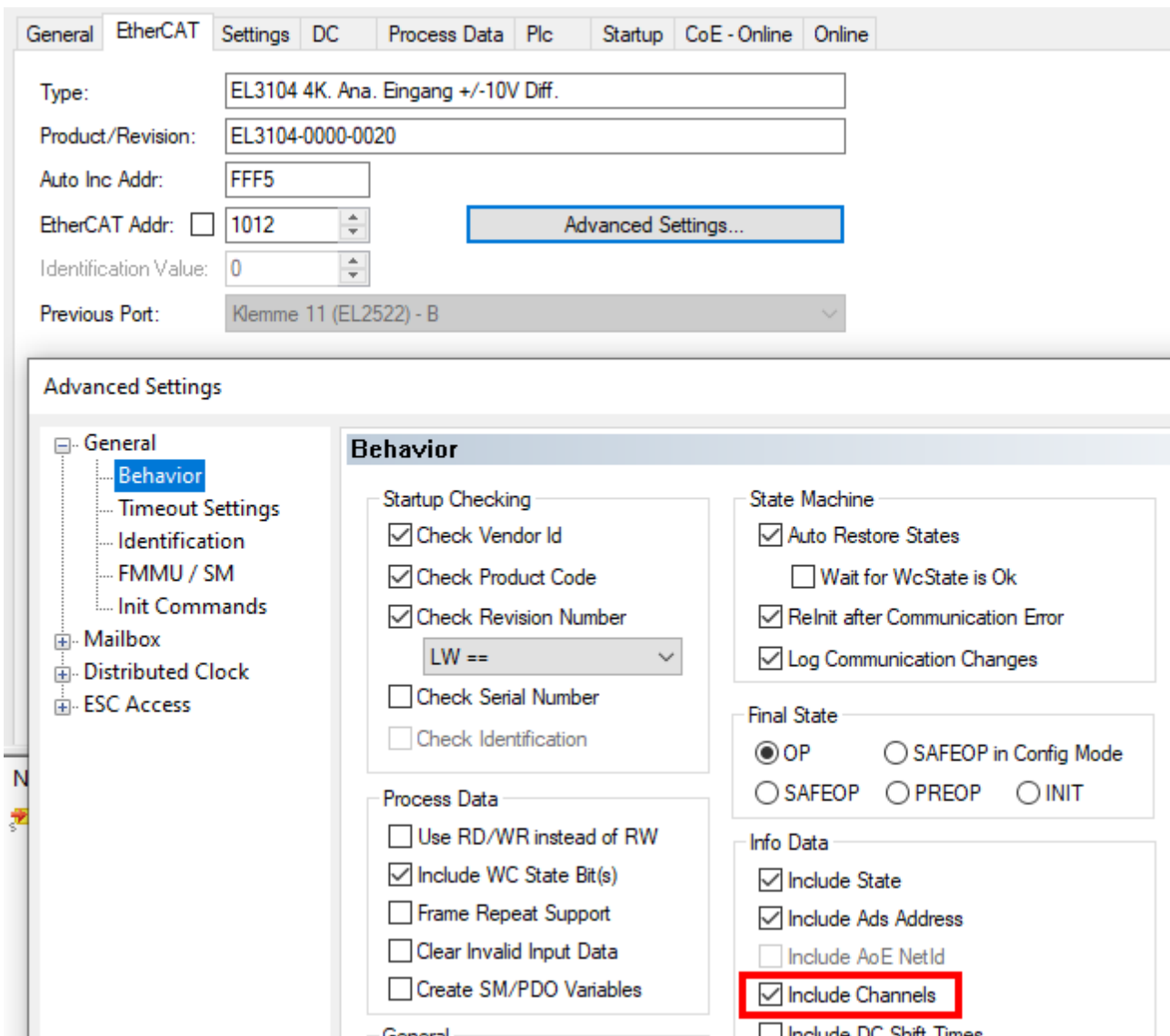


Fig. 324: Display the channels in InfoData via "Advanced settings" -> "Behavior" "Include Channels"

TwinCAT then shows linkable channels:

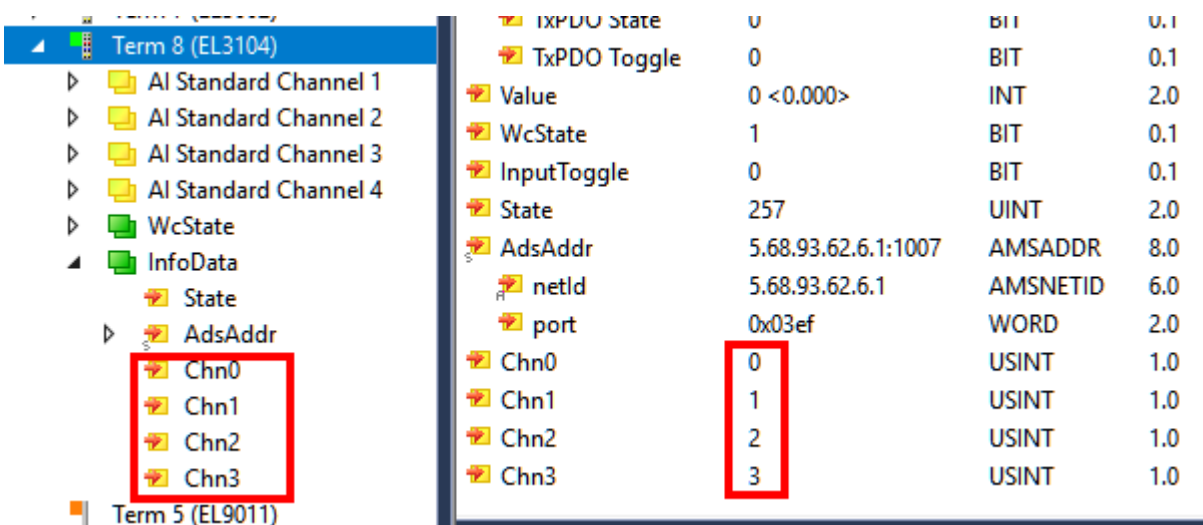
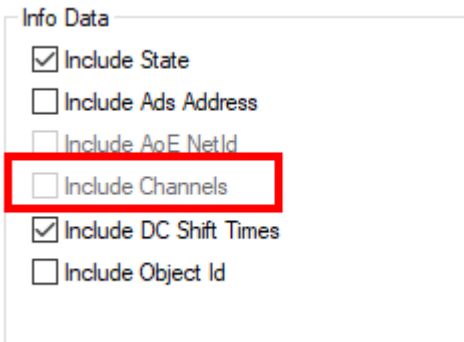


Fig. 325: Display of linkable channels in TwinCAT

The content of the bytes is fixed and cannot be changed, which is the consecutive channel number.

Background:

- TwinCAT is orientated on the profile information in the ESI file → "Profiles". For modules that do not (yet) have this information in the ESI, the channels cannot be activated.



In the Advanced Settings of the EtherCAT Master the following information is displayed, if the channels are activated in one of the slaves.

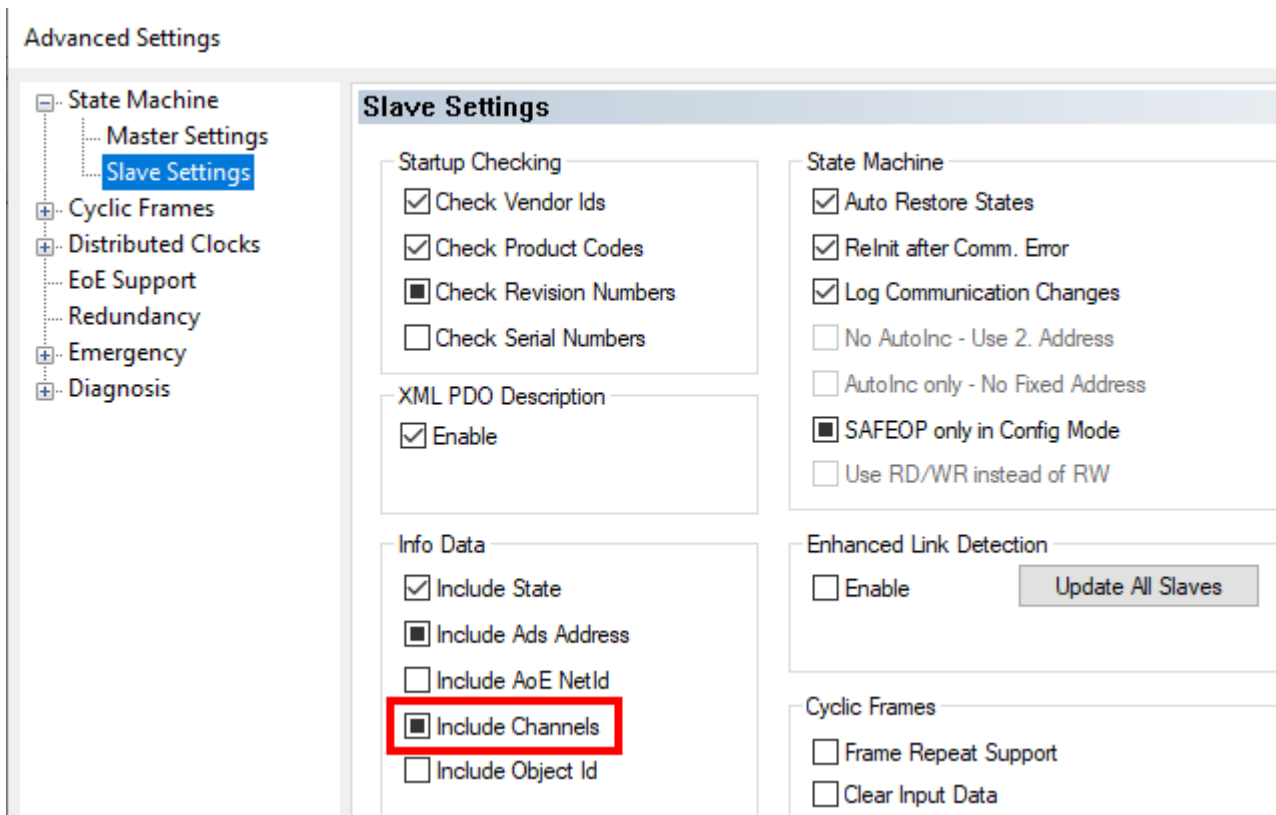


Fig. 326: Display of the activation of "Include Channels" in the EtherCAT Master

## 8.2.2 Timeout Settings

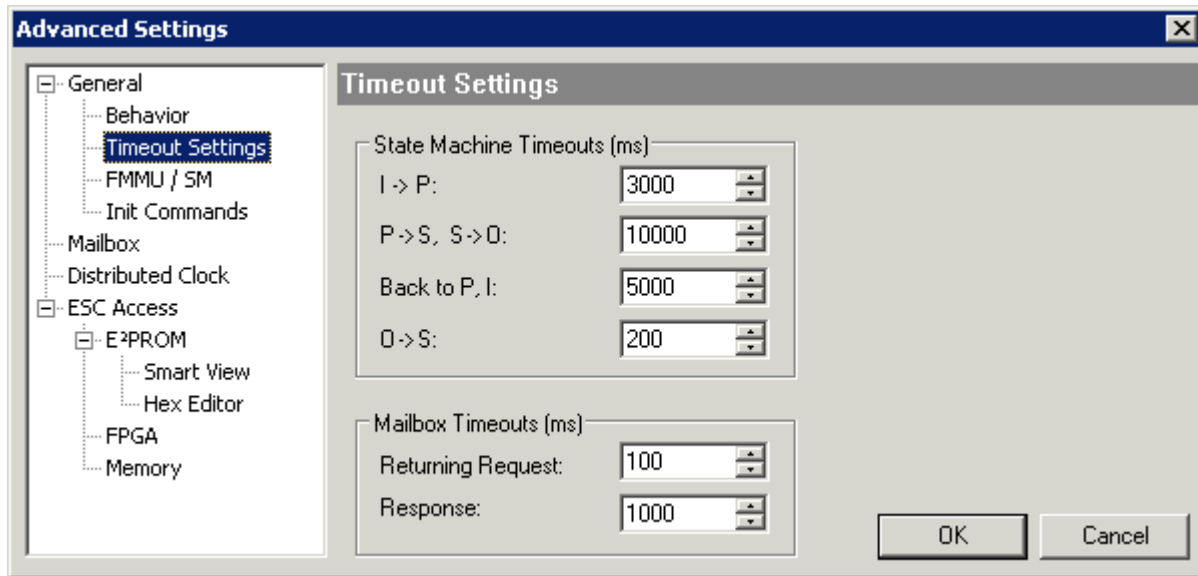


Fig. 327: Advanced Settings: "Timeout Settings" dialog

### State-Machine Timeouts (ms)

Here you can specify the times for the timeouts during transition from one state of the state machine to another.

#### I -> P

Timeout for the transition from state **Init** to state **Pre-Operational**.

#### P -> S, S -> O

Timeout for the transition

- from state **Pre-Operational** to state **Safe-Operational** or
- from state **Safe-Operational** to state **Operational**.

#### Back to P, I

Timeout for return to state **Pre-Operational** or **Init**.

#### O -> S

Timeout for the transition from state **Operational** to state **Safe-Operational**.

### Mailbox Timeouts (ms)

Here you can specify the timeouts for acyclic commands for the mailbox (mailbox interface).

#### Returning Request

Timeout for returning a request from the EtherCAT ring.

#### Response

Timeout for the response from the addressed EtherCAT device to the request.

### 8.2.3 FMMU / SM

This dialog shows the current configuration of the Fieldbus Memory Management Unit (FMMU) and the Sync Managers (SM) and enables you to modify them.

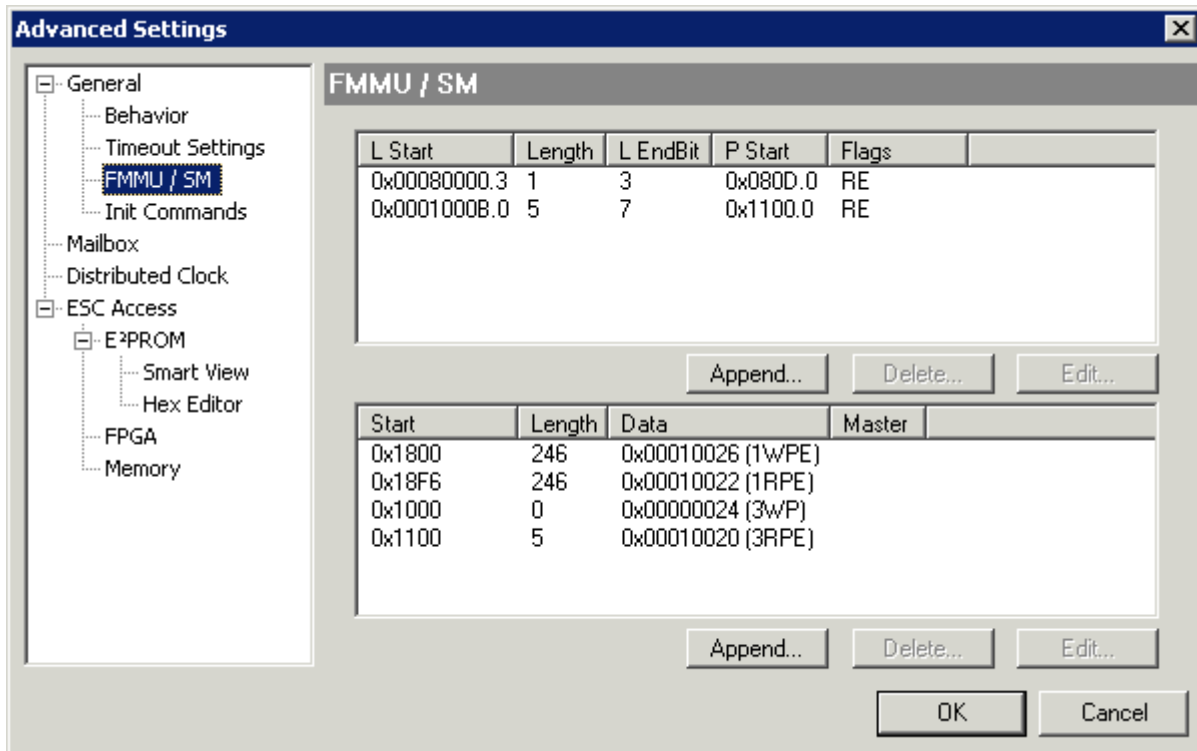


Fig. 328: Advanced Settings: "FMMU/SM" dialog

#### Configuration if the FMMUs (list at the top)

##### L Start

Specified from which logical address the FMMU starts mapping the data. The start bit is set to match the number following the dot (0xnnnnnnnn.**start bit**)

##### Length

Specifies how many bytes are mapped by the logical addressing.

##### L EndBit

End bit of the logical address. If the logical address for a byte is to be configured, the start bit must be configured as 0 (L start = 0xnnnnnnnn.**0**).

##### P Start

Specifies the physical address to which the logical address points.

##### Flags

RE: Read Enabled  
WE: Write enabled

**Configuration of the Sync Managers (list at the bottom)**

**Start**

Specifies from which address the Sync Channel is active.

**Length**

Length of the Sync Channel in bytes. If the Sync Channel is not activated, the length is 0.

**Data**

Configuration data to be written to the Sync Channel.

**Master**

(in preparation)

**8.2.4 Mailbox**

If the EtherCAT slave supports one or several mailbox protocols, the additional tab *Mailbox* is displayed. This dialog lists the mailbox protocols supported by the EtherCAT slave and enables their configuration to be modified.

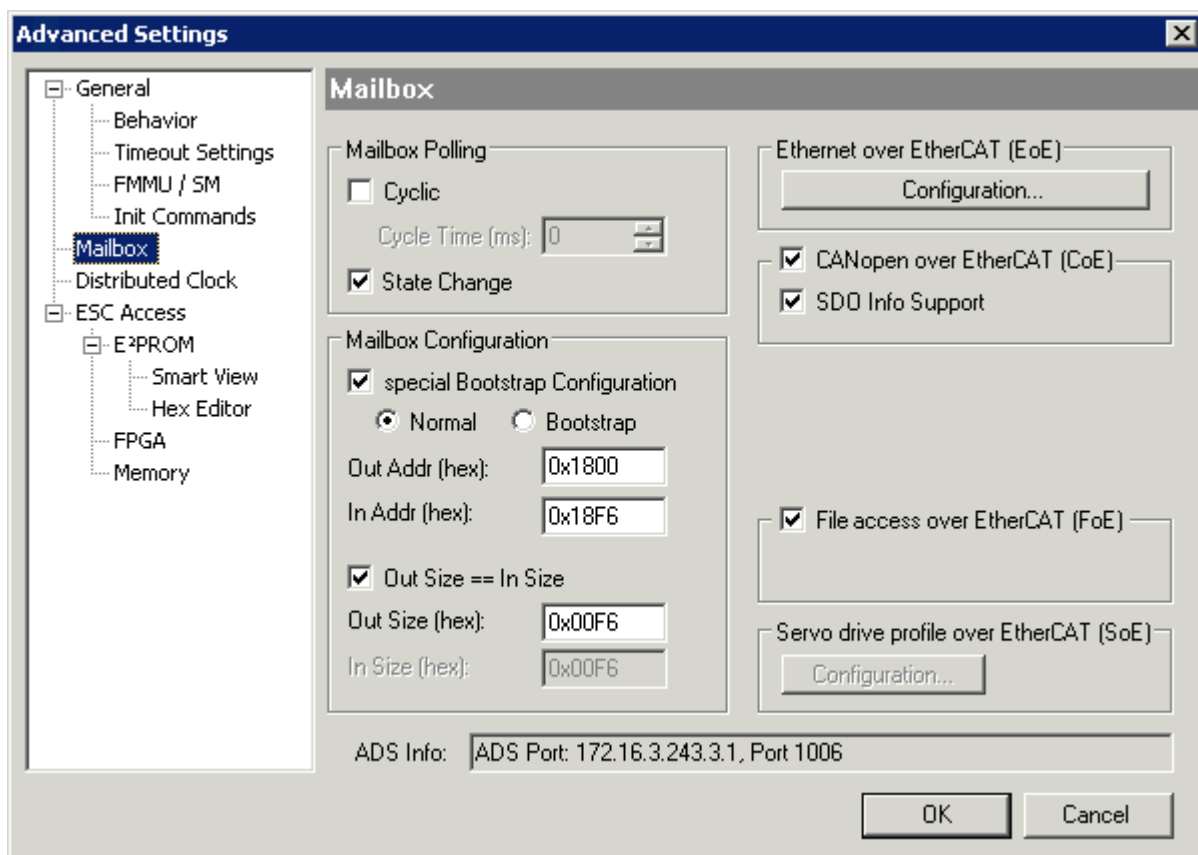


Fig. 329: Advanced Settings: "Mailbox" dialog

**Mailbox Polling**

- **Cyclical**  
If this checkbox is selected, the master cyclically reads the mailbox of the EtherCAT slave.
- **Cycle time (ms)**  
If the checkbox *Cyclic* is selected, this value specifies how often the master reads the mailbox of the EtherCAT slave.

- **State Change**  
If this checkbox is selected, the master checks a state bit of the slave in order to determine whether unread data are available in the mailbox. In this case the master reads the mailbox. This mode is more efficient than cyclic mode, because the master can check the state of the mailboxes of several EtherCAT slaves with a single EtherCAT command (LRD).

### Mailbox Configuration

- **Special Bootstrap Configuration**  
Normal: for process data traffic or mailbox protocols  
Bootstrap: for firmware download
- **Out-Adr.**  
Physical start address of the output mailbox in the slave controller.
- **In-Adr.**  
Physical start address of the input mailbox in the slave controller.
- **Out-Size == In Size**  
If you select this checkbox, Out Size and In Size are identical.
- **Out Size**  
Size of the output mailbox in bytes.
- **In Size**  
Size of the input mailbox in bytes.

### Mailbox protocols

- **Ethernet over EtherCAT (EoE)**
  - **Ethernet over EtherCAT (EoE)**  
If this group is enabled, the EtherCAT slave supports the mailbox protocol *Ethernet over EtherCAT* (EoE).
  - **Configuration**  
This button opens a dialog for configuring the mailbox protocol *Ethernet over EtherCAT*.
- **CANopen over EtherCAT (CoE)**
  - **CANopen over EtherCAT (CoE)**  
If this checkbox is activated, the EtherCAT slave supports the mailbox protocol *CANopen over EtherCAT* (CoE).
  - **SDO Info Support**  
If this checkbox is activated, the master can load the object directory of the EtherCAT slave.
- **File access over EtherCAT (FoE)**
  - **File access over EtherCAT (FoE)**  
If this checkbox is activated, the EtherCAT slave supports the mailbox protocol *File access over EtherCAT* (FoE).
- **Servo drive profile over EtherCAT (SoE)**
  - **Servo drive profile over EtherCAT**  
If this group is enabled, the EtherCAT slave supports the mailbox protocol *Servo drive over EtherCAT* (SoE).
  - **Configuration**  
This button opens a dialog for configuring the mailbox protocol *Servo drive over EtherCAT*.

### ADS-Info

ADS identification of an EtherCAT slave.

- The ADS Net ID is the same as the NetID of an EtherCAT device.
- The ADS port is the same as the fixed address of an EtherCAT device (see EtherCAT Adr).

ADS enables communication with the mailbox of the EtherCAT slave (e.g. SDO Upload Request).



## 8.2.5 Smart View

This dialog shows the settings that are stored in the EEPROM of the EtherCAT Slave Controller (ESC).

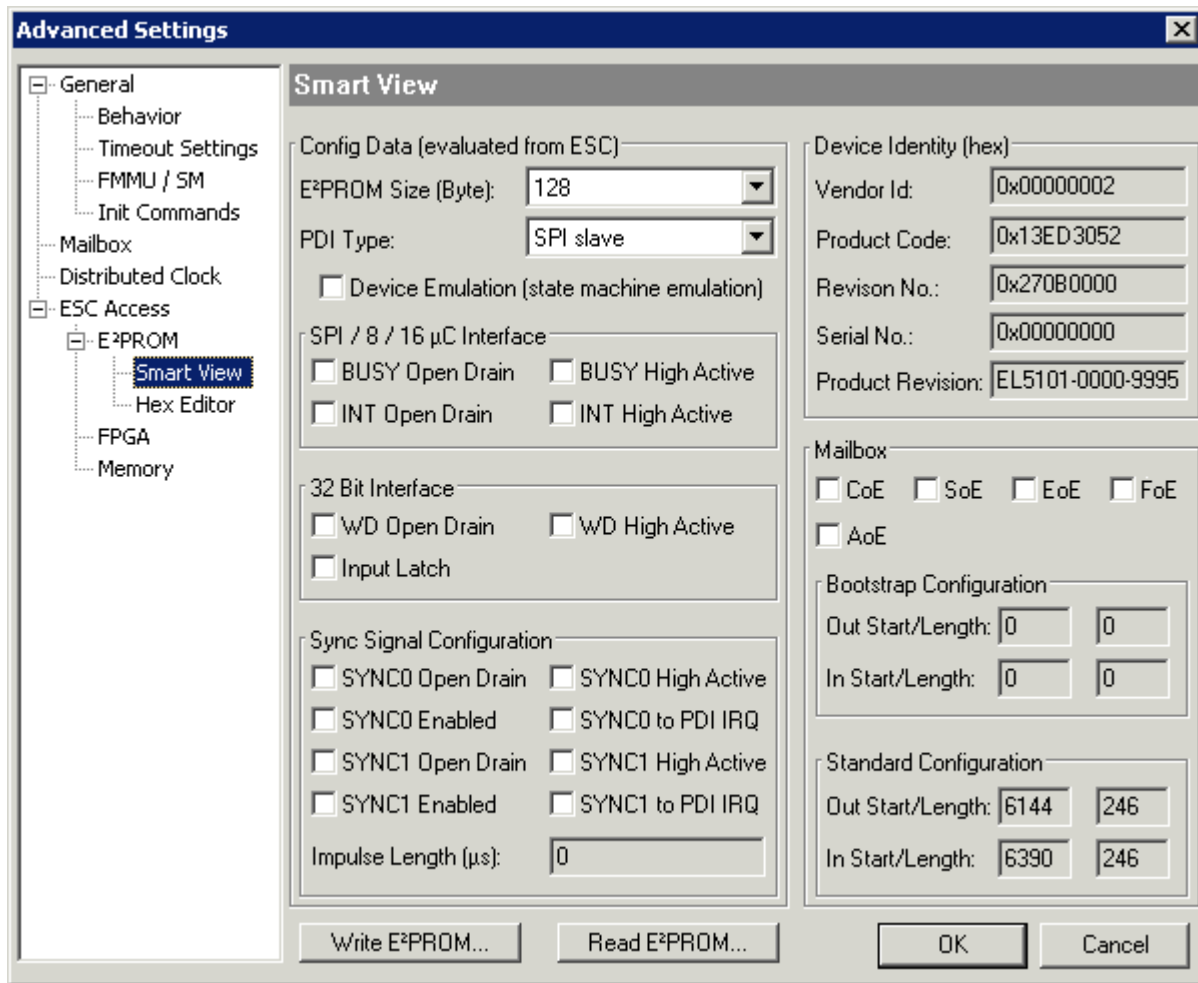


Fig. 330: Advanced Settings: "Smart View" dialog

### Config Data (evaluated from ESC)

This section shows internal parameters of the EtherCAT Slave Controller (ESC).

- EEPROM Size (Byte): Size of the EEPROM in bytes
- PDI Type: Type of process data interface
- Device Emulation (state machine emulation)

### SPI / 8 / 18 µC Interface

This section shows internal parameters of the 16 bit process data interface:

- BUSY Open Drain
- BUSY High Active
- INT Open Drain
- INT High Active

### 32 Bit Interface

This section shows internal parameters of the 32 bit process data interface:

- WD Open Drain: Watchdog Open Drain
- WD High Active: Watchdog Open High Active

- Input Latch

### **Sync Signal Configuration**

This section shows internal parameters for configuring the Sync signal:

- Sync 0 Open Drain
- Sync 0 High Active
- Sync 0 Enabled
- Sync 0 to PDI IRQ
- Sync 1 Open Drain
- Sync 1 High Active
- Sync 1 Enabled
- Sync 1 to PDI IRQ

### **EEPROM buttons**

- Write EEPROM: Button for writing to the EEPROM.
- Read EEPROM: Button for reading the EEPROM.

### **Device Identity (hex)**

This section shows information for the EtherCAT device.

- Vendor ID: Identification number of the device manufacturer.
- Product Code: Product code of the EtherCAT device.
- Revision No.: Revision number of the EtherCAT device.
- Serial No.: Serial number of the EtherCAT device.
- Product Revision: Product revision of the EtherCAT device.

### **Mailbox**

Indicates which mailbox communication variants are supported.

- CoE: CanOpen over EtherCAT
- SoE: Servo-Profile over EtherCAT
- EoE: Ethernet via EtherCAT
- FoE: Servo-Profile over EtherCAT
- AoE: ADS over EtherCAT

### **Bootstrap Configuration**

Configuration for Bootstrap mode (e.g. firmware update)

- Out Start/Length
- In Start/Length

### **Standard Configuration**

Configuration for regular operation (process data transfer or mailbox operation)

- Out Start/Length
- In Start/Length

## 8.2.6 Memory

The Memory dialog enables the user to read data from the memory (DPRAM) of the EtherCAT slave controller or save them there. The list below shows the memory of the EtherCAT slave controller. The Start Offset corresponds to the value entered in the *Start Offset* box. Each entry shows a register (2 bytes). If the System Manager knows a description for a register, it is displayed next to the *Offset*. To change a register value, enter the required value in column Dec, Hex or Char. Once the value has been changed it is shown in red, and the *Write* button is activated. You can now click the *Write* button to transfer the modified values to the EtherCAT slave.

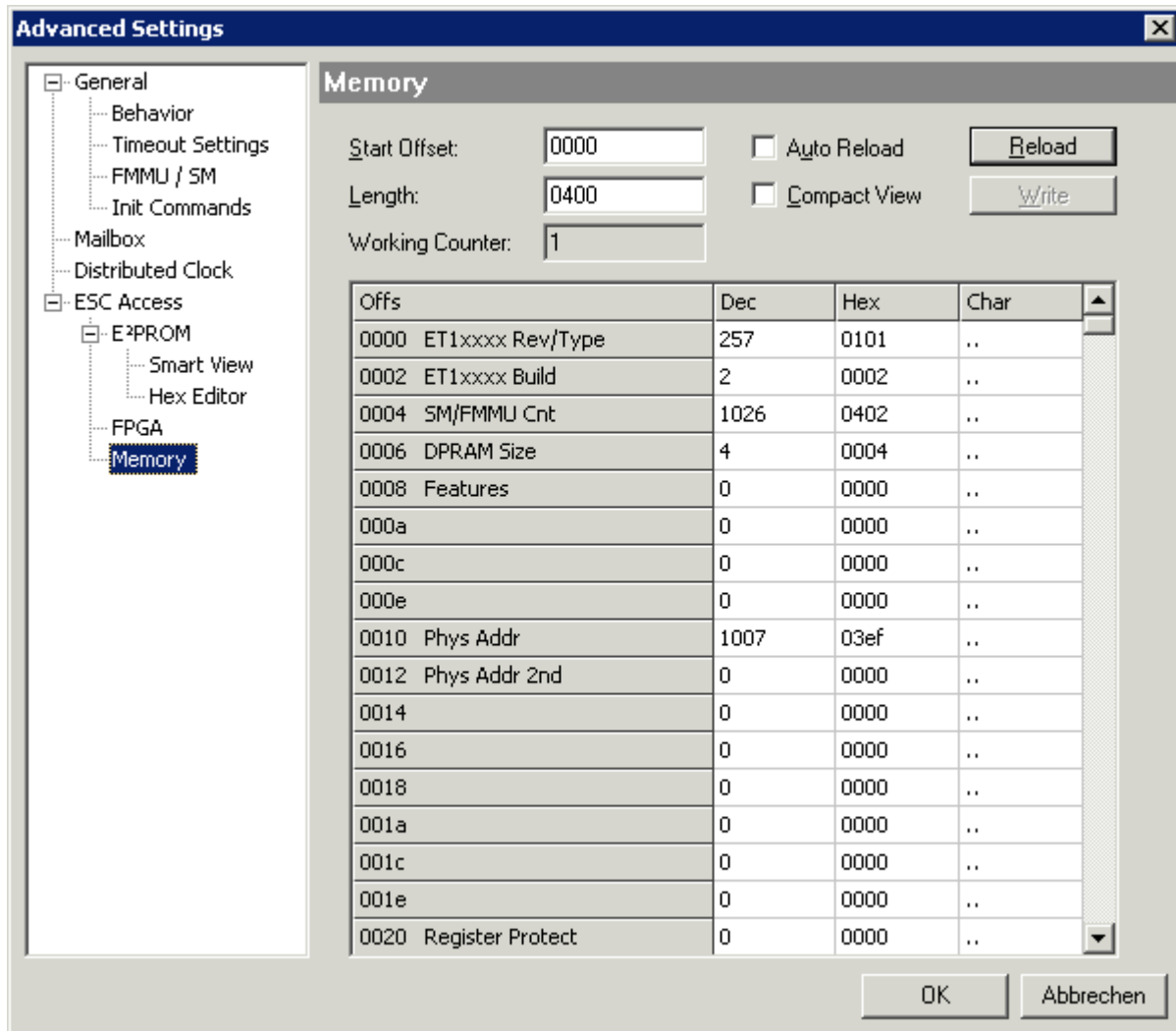


Fig. 331: Advanced Settings: "Memory" dialog

### Start Offset

Start address (hexadecimal) of first register to be displayed in the list.

### Length

Length (hexadecimal) of the data to be displayed in bytes. The maximum permitted length is 0400<sub>hex</sub> (1024<sub>dec</sub>).

### Working Counter

If the master succeeded in writing to or reading from the slave, the working counter is 1, otherwise the working counter is 0.

### Auto Reload

Here you can activate cyclic reading of the memory (default: inactive).

### Compact View

Here you can specify that only those parameter are displayed, for which the System Manager has a description (default: not active).

### Reload

Reads the current values from the EtherCAT slave.

### Write

Saves modified values (shown in red) in the EtherCAT slave.

## 8.3 Firmware Update EL/ES/EM/ELM/EPxxxx

This section describes the device update for Beckhoff EtherCAT slaves from the EL/ES, ELM, EM, EK and EP series. A firmware update should only be carried out after consultation with Beckhoff support.

### Storage locations

An EtherCAT slave stores operating data in up to three locations:

- Depending on functionality and performance EtherCAT slaves have one or several local controllers for processing I/O data. The corresponding program is the so-called **firmware** in \*.efw format.
- In some EtherCAT slaves the EtherCAT communication may also be integrated in these controllers. In this case the controller is usually a so-called **FPGA** chip with \*.rbf firmware.
- In addition, each EtherCAT slave has a memory chip, a so-called **ESI-EEPROM**, for storing its own device description (ESI: EtherCAT Slave Information). On power-up this description is loaded and the EtherCAT communication is set up accordingly. The device description is available from the download area of the Beckhoff website at (<https://www.beckhoff.de>). All ESI files are accessible there as zip files.

Customers can access the data via the EtherCAT fieldbus and its communication mechanisms. Acyclic mailbox communication or register access to the ESC is used for updating or reading of these data.

The TwinCAT System Manager offers mechanisms for programming all three parts with new data, if the slave is set up for this purpose. Generally the slave does not check whether the new data are suitable, i.e. it may no longer be able to operate if the data are unsuitable.

### Simplified update by bundle firmware

The update using so-called **bundle firmware** is more convenient: in this case the controller firmware and the ESI description are combined in a \*.efw file; during the update both the firmware and the ESI are changed in the terminal. For this to happen it is necessary

- for the firmware to be in a packed format: recognizable by the file name, which also contains the revision number, e.g. ELxxx-xxx\_REV0016\_SW01.efw
- for password=1 to be entered in the download dialog. If password=0 (default setting) only the firmware update is carried out, without an ESI update.
- for the device to support this function. The function usually cannot be retrofitted; it is a component of many new developments from year of manufacture 2016.

Following the update, its success should be verified

- ESI/Revision: e.g. by means of an online scan in TwinCAT ConfigMode/FreeRun – this is a convenient way to determine the revision
- Firmware: e.g. by looking in the online CoE of the device

**NOTE**

**Risk of damage to the device!**

- ✓ Note the following when downloading new device files
- a) Firmware downloads to an EtherCAT device must not be interrupted
- b) Flawless EtherCAT communication must be ensured. CRC errors or LostFrames must be avoided.
- c) The power supply must adequately dimensioned. The signal level must meet the specification.

⇒ In the event of malfunctions during the update process the EtherCAT device may become unusable and require re-commissioning by the manufacturer.

### 8.3.1 Device description ESI file/XML

**NOTE**

**Attention regarding update of the ESI description/EEPROM**

Some slaves have stored calibration and configuration data from the production in the EEPROM. These are irretrievably overwritten during an update.

The ESI device description is stored locally on the slave and loaded on start-up. Each device description has a unique identifier consisting of slave name (9 characters/digits) and a revision number (4 digits). Each slave configured in the System Manager shows its identifier in the EtherCAT tab:

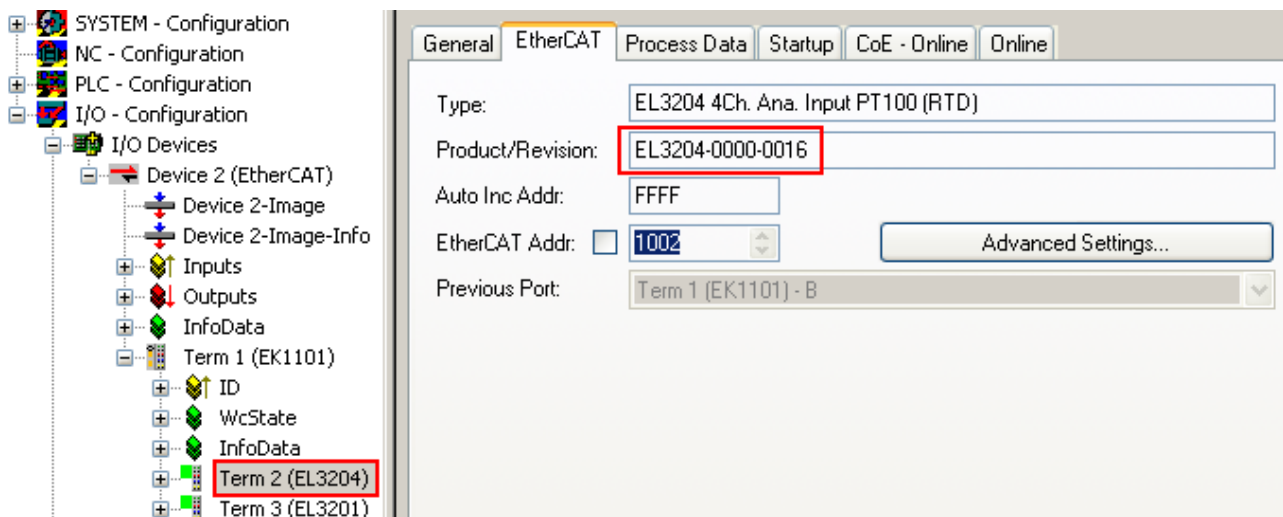


Fig. 332: Device identifier consisting of name EL3204-0000 and revision -0016

The configured identifier must be compatible with the actual device description used as hardware, i.e. the description which the slave has loaded on start-up (in this case EL3204). Normally the configured revision must be the same or lower than that actually present in the terminal network.

For further information on this, please refer to the [EtherCAT system documentation](#).

**● Update of XML/ESI description**

**i** The device revision is closely linked to the firmware and hardware used. Incompatible combinations lead to malfunctions or even final shutdown of the device. Corresponding updates should only be carried out in consultation with Beckhoff support.

**Display of ESI slave identifier**

The simplest way to ascertain compliance of configured and actual device description is to scan the EtherCAT boxes in TwinCAT mode Config/FreeRun:

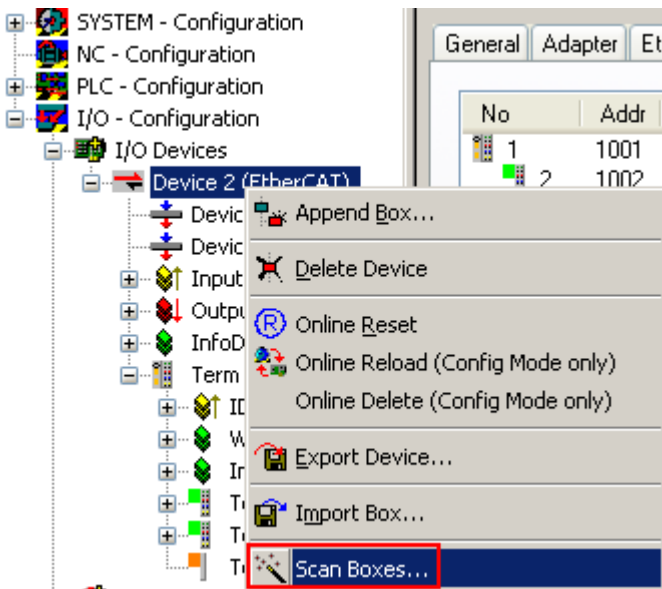


Fig. 333: Scan the subordinate field by right-clicking on the EtherCAT device

If the found field matches the configured field, the display shows



Fig. 334: Configuration is identical

otherwise a change dialog appears for entering the actual data in the configuration.

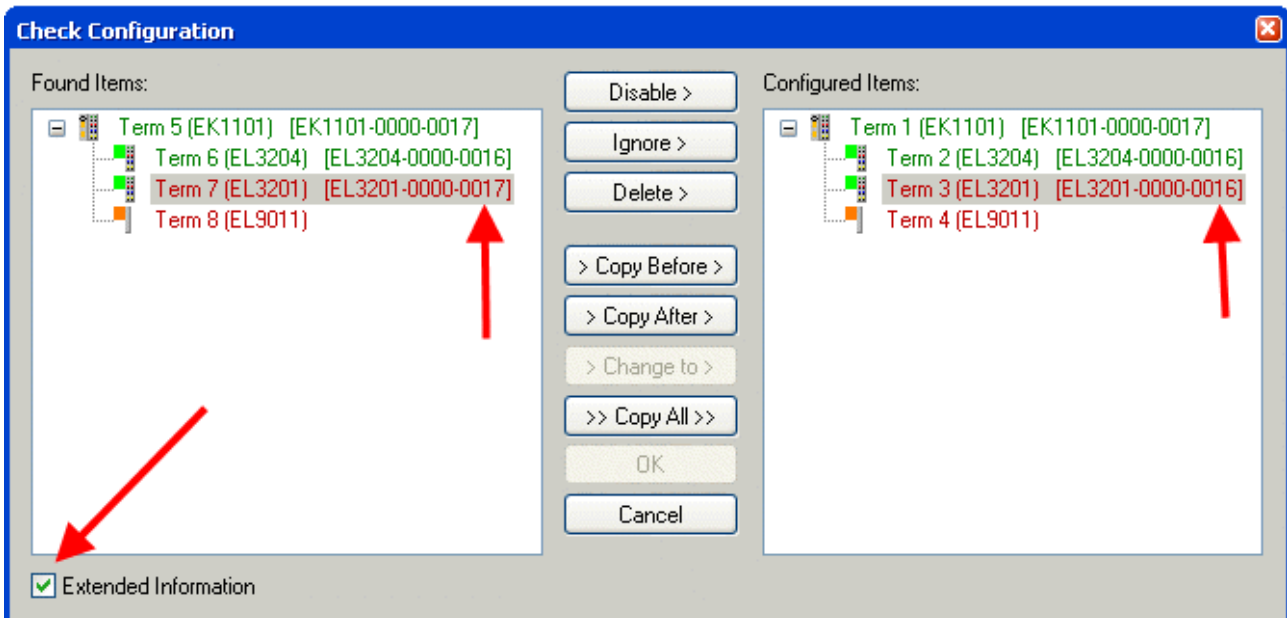


Fig. 335: Change dialog

In this example in Fig. *Change dialog*, an EL3201-0000-0017 was found, while an EL3201-0000-0016 was configured. In this case the configuration can be adapted with the *Copy Before* button. The *Extended Information* checkbox must be set in order to display the revision.

### Changing the ESI slave identifier

The ESI/EEPROM identifier can be updated as follows under TwinCAT:

- Trouble-free EtherCAT communication must be established with the slave.
- The state of the slave is irrelevant.
- Right-clicking on the slave in the online display opens the *EEPROM Update* dialog, Fig. *EEPROM Update*

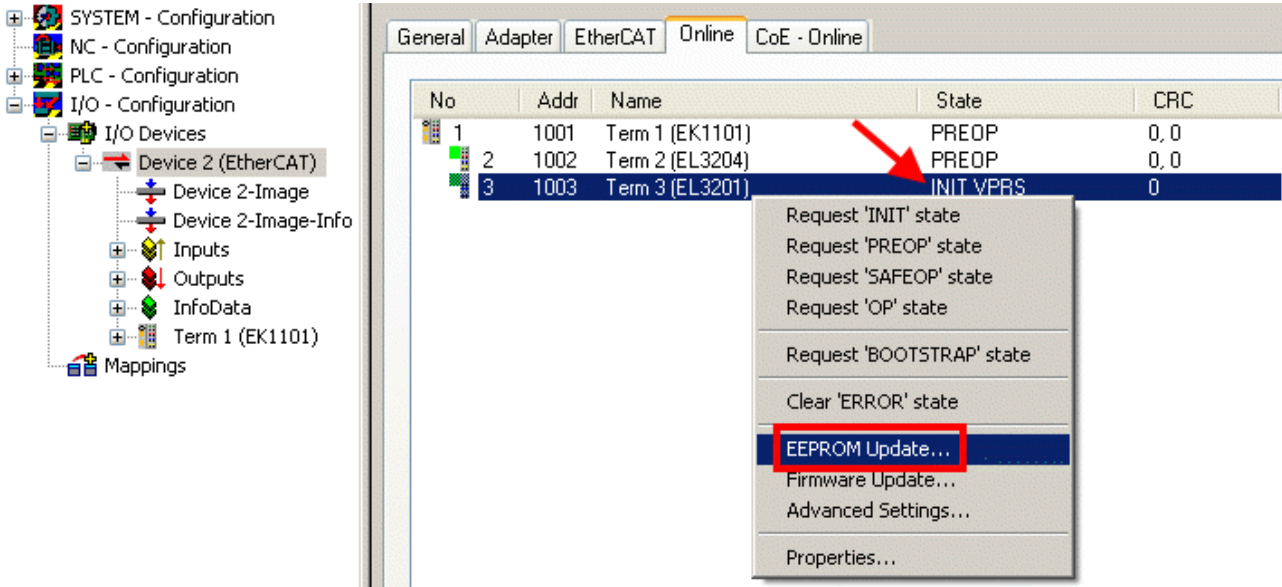


Fig. 336: EEPROM Update

The new ESI description is selected in the following dialog, see Fig. *Selecting the new ESI*. The checkbox *Show Hidden Devices* also displays older, normally hidden versions of a slave.

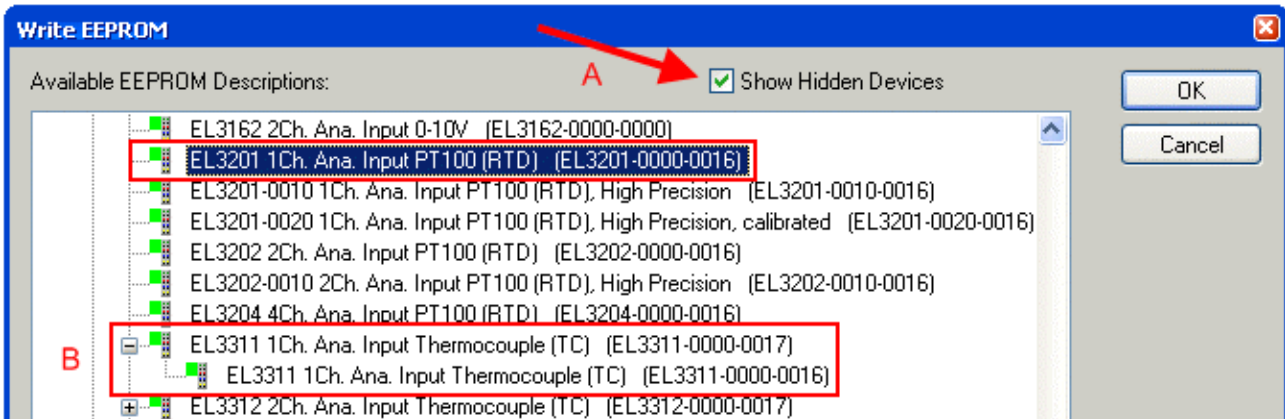


Fig. 337: Selecting the new ESI

A progress bar in the System Manager shows the progress. Data are first written, then verified.

**i The change only takes effect after a restart.**

Most EtherCAT devices read a modified ESI description immediately or after startup from the INIT. Some communication settings such as distributed clocks are only read during power-on. The EtherCAT slave therefore has to be switched off briefly in order for the change to take effect.

### 8.3.2 Firmware explanation

#### Determining the firmware version

##### Determining the version on laser inscription

Beckhoff EtherCAT slaves feature serial numbers applied by laser. The serial number has the following structure: **KK YY FF HH**

- KK - week of production (CW, calendar week)
- YY - year of production
- FF - firmware version
- HH - hardware version

Example with ser. no.: 12 10 03 02:

- 12 - week of production 12
- 10 - year of production 2010
- 03 - firmware version 03
- 02 - hardware version 02

##### Determining the version via the System Manager

The TwinCAT System Manager shows the version of the controller firmware if the master can access the slave online. Click on the E-Bus Terminal whose controller firmware you want to check (in the example terminal 2 (EL3204)) and select the tab *CoE Online* (CAN over EtherCAT).

#### ● CoE Online and Offline CoE

**i**

Two CoE directories are available:

- **online**: This is offered in the EtherCAT slave by the controller, if the EtherCAT slave supports this. This CoE directory can only be displayed if a slave is connected and operational.
- **offline**: The EtherCAT Slave Information ESI/XML may contain the default content of the CoE. This CoE directory can only be displayed if it is included in the ESI (e.g. "Beckhoff EL5xxx.xml").

The Advanced button must be used for switching between the two views.

In Fig. *Display of EL3204 firmware version* the firmware version of the selected EL3204 is shown as 03 in CoE entry 0x100A.

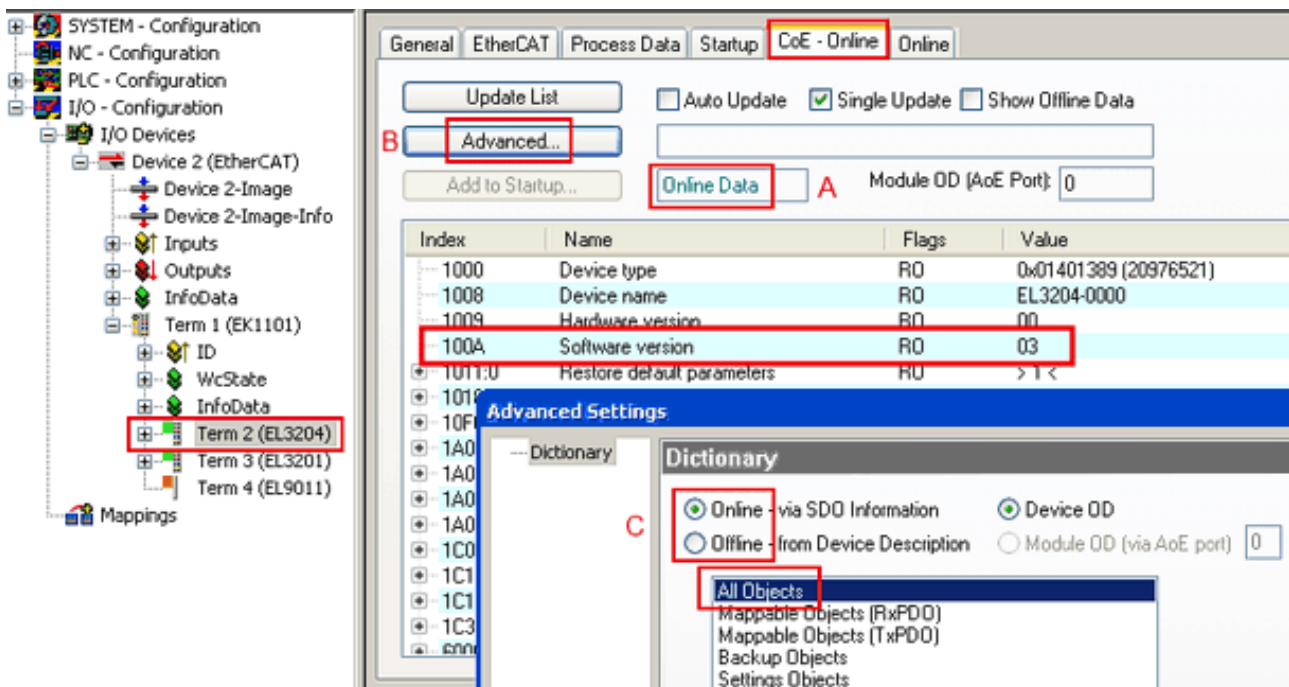


Fig. 338: Display of EL3204 firmware version



In (A) TwinCAT 2.11 shows that the Online CoE directory is currently displayed. If this is not the case, the Online directory can be loaded via the *Online* option in Advanced Settings (B) and double-clicking on *AllObjects*.

### 8.3.3 Updating controller firmware \*.efw

**● CoE directory**

**i** The Online CoE directory is managed by the controller and stored in a dedicated EEPROM, which is generally not changed during a firmware update.

Switch to the *Online* tab to update the controller firmware of a slave, see Fig. *Firmware Update*.

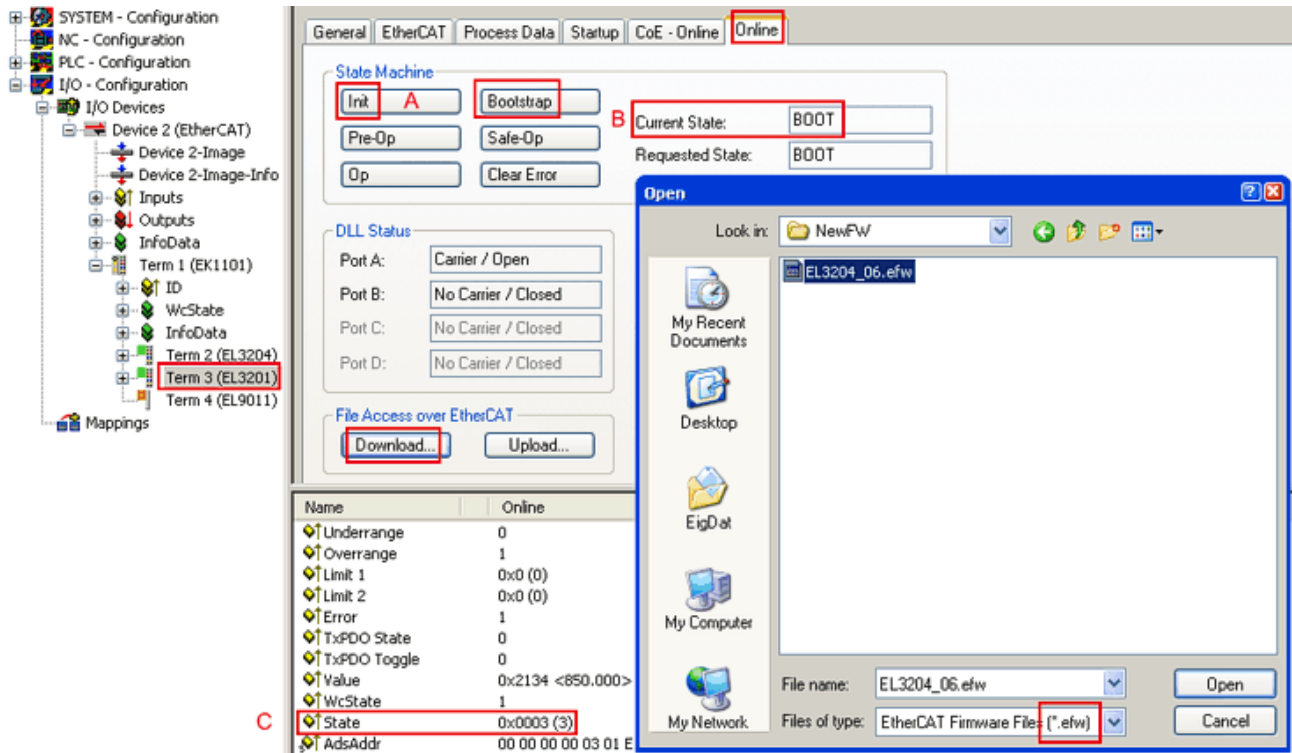
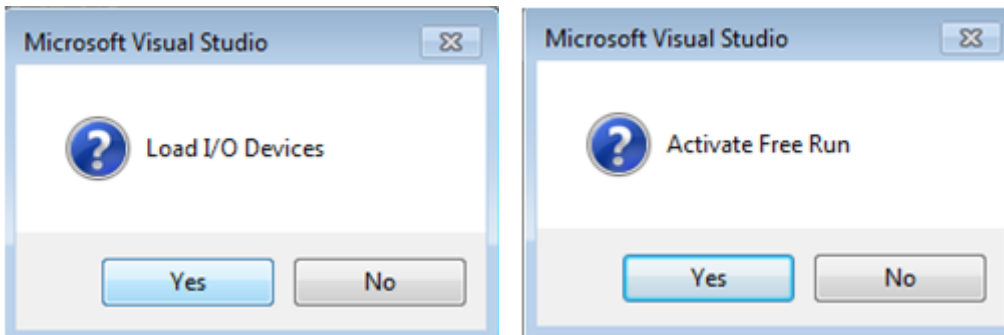


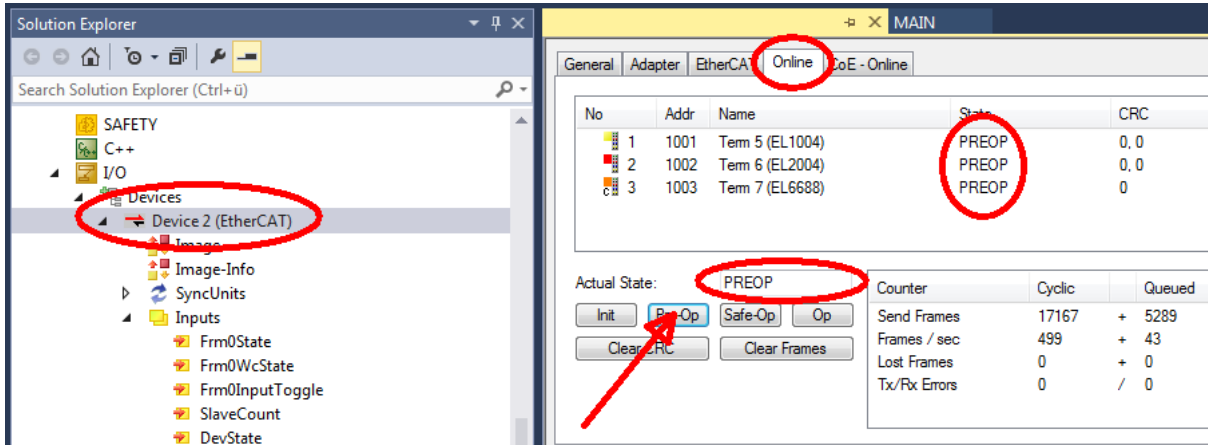
Fig. 339: Firmware Update

Proceed as follows, unless instructed otherwise by Beckhoff support. Valid for TwinCAT 2 and 3 as EtherCAT master.

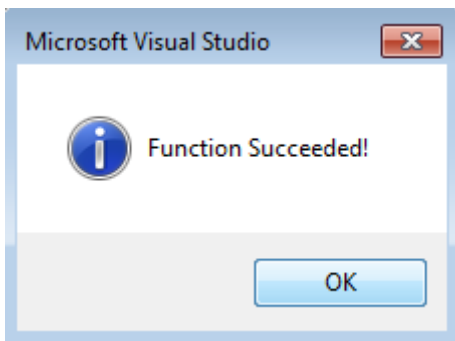
- Switch TwinCAT system to ConfigMode/FreeRun with cycle time  $\geq 1$  ms (default in ConfigMode is 4 ms). A FW-Update during real time operation is not recommended.



- Switch EtherCAT Master to PreOP



- Switch slave to INIT (A)
- Switch slave to BOOTSTRAP
- Check the current status (B, C)
- Download the new \*efw file (wait until it ends). A pass word will not be necessary usually.



- After the download switch to INIT, then PreOP
- Switch off the slave briefly (don't pull under voltage!)
- Check within CoE 0x100A, if the FW status was correctly overtaken.

### 8.3.4 FPGA firmware \*.rbf

If an FPGA chip deals with the EtherCAT communication an update may be accomplished via an \*.rbf file.

- Controller firmware for processing I/O signals
- FPGA firmware for EtherCAT communication (only for terminals with FPGA)

The firmware version number included in the terminal serial number contains both firmware components. If one of these firmware components is modified this version number is updated.

#### Determining the version via the System Manager

The TwinCAT System Manager indicates the FPGA firmware version. Click on the Ethernet card of your EtherCAT strand (Device 2 in the example) and select the *Online* tab.

The *Reg:0002* column indicates the firmware version of the individual EtherCAT devices in hexadecimal and decimal representation.

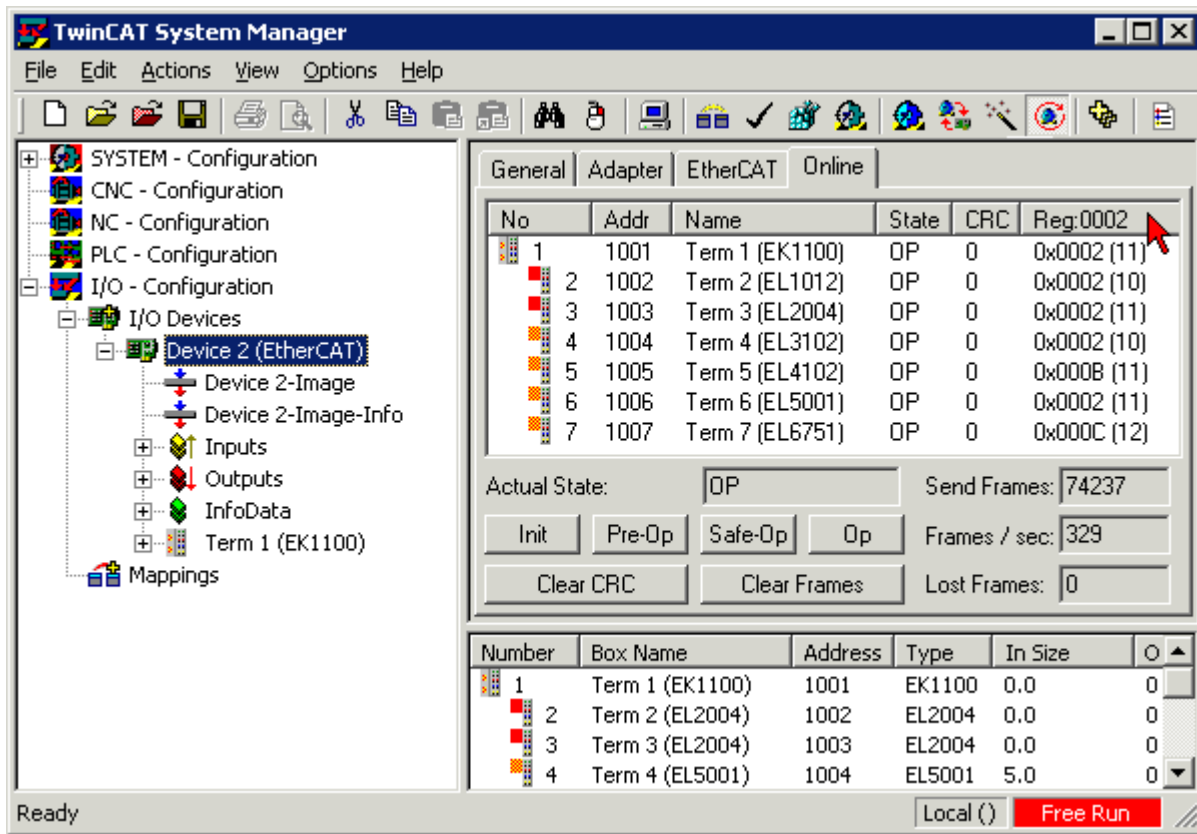


Fig. 340: FPGA firmware version definition

If the column *Reg:0002* is not displayed, right-click the table header and select *Properties* in the context menu.

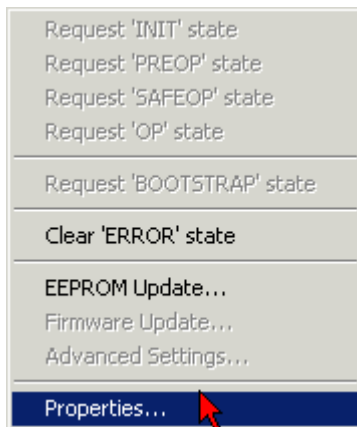
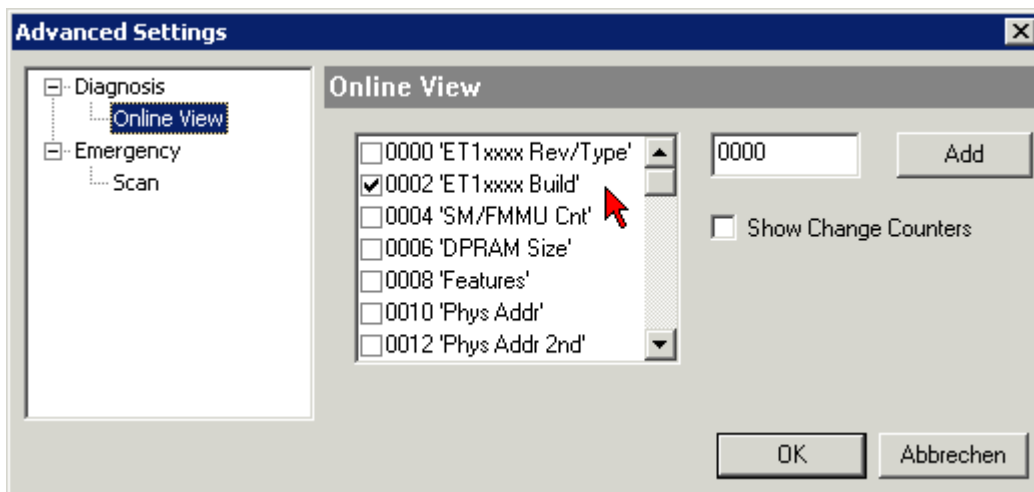


Fig. 341: Context menu *Properties*

The *Advanced Settings* dialog appears where the columns to be displayed can be selected. Under *Diagnosis/Online View* select the *'0002 ETxxxx Build'* check box in order to activate the FPGA firmware version display.

Fig. 342: Dialog *Advanced Settings*

### Update

For updating the FPGA firmware

- of an EtherCAT coupler the coupler must have FPGA firmware version 11 or higher;
- of an E-Bus Terminal the terminal must have FPGA firmware version 10 or higher.

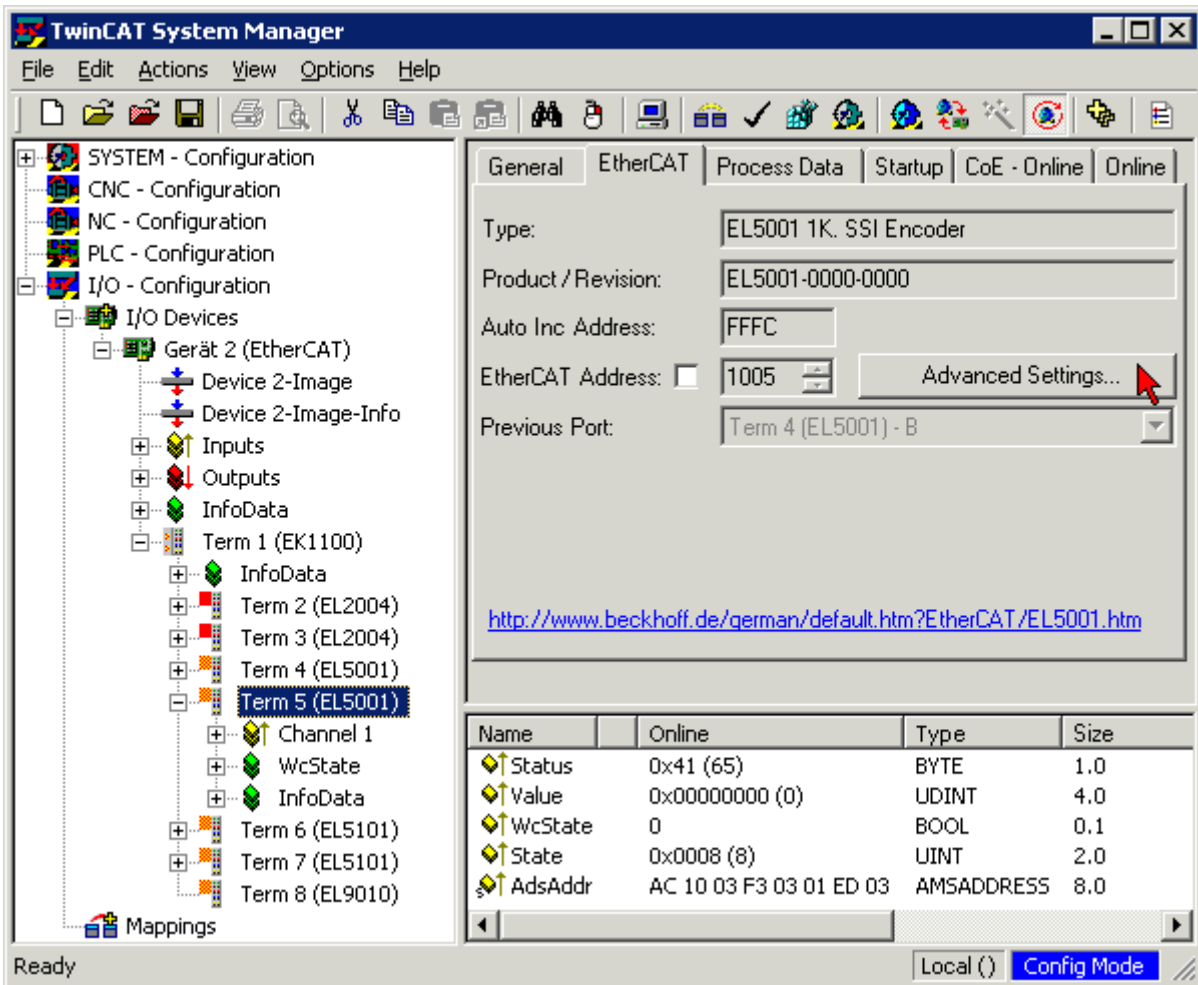
Older firmware versions can only be updated by the manufacturer!

### Updating an EtherCAT device

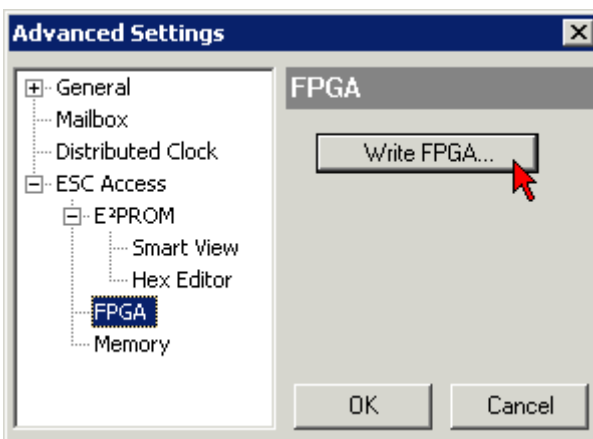
The following sequence order have to be met if no other specifications are given (e.g. by the Beckhoff support):

- Switch TwinCAT system to ConfigMode/FreeRun with cycle time  $\geq 1$  ms (default in ConfigMode is 4 ms). A FW-Update during real time operation is not recommended.

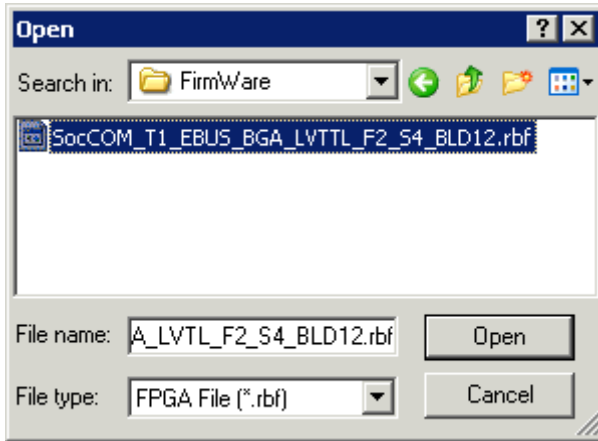
- In the TwinCAT System Manager select the terminal for which the FPGA firmware is to be updated (in the example: Terminal 5: EL5001) and click the *Advanced Settings* button in the *EtherCAT* tab:



- The *Advanced Settings* dialog appears. Under *ESC Access/E<sup>2</sup>PROM/FPGA* click on *Write FPGA* button:



- Select the file (\*.rbf) with the new FPGA firmware, and transfer it to the EtherCAT device:



- Wait until download ends
- Switch slave current less for a short time (don't pull under voltage!). In order to activate the new FPGA firmware a restart (switching the power supply off and on again) of the EtherCAT device is required.
- Check the new FPGA status

**NOTE**

**Risk of damage to the device!**

A download of firmware to an EtherCAT device must not be interrupted in any case! If you interrupt this process by switching off power supply or disconnecting the Ethernet link, the EtherCAT device can only be recommissioned by the manufacturer!

### 8.3.5 Simultaneous updating of several EtherCAT devices

The firmware and ESI descriptions of several devices can be updated simultaneously, provided the devices have the same firmware file/ESI.

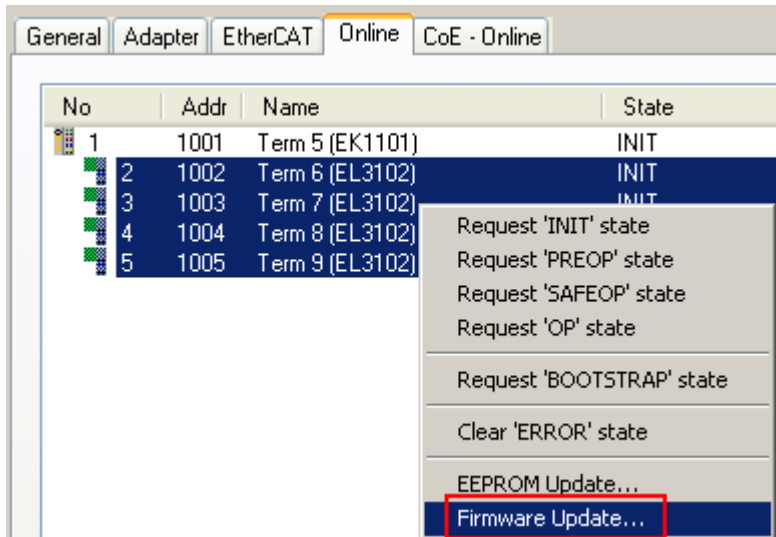


Fig. 343: Multiple selection and firmware update

Select the required slaves and carry out the firmware update in BOOTSTRAP mode as described above.

## 9 FAQ

### 9.1 Windows Memory Dump

#### Situation

In rare cases it can be useful to know the internal software state of an IPC systems before a failure. To this end Windows can write an image of the RAM at the time of the restart to the hard disk. This is referred to as a memory dump.

#### Solution

The correct writing of the memory dump must be set before the restart commences.

---

#### ● Windows operating system



Memory dumps are only available under Windows NT/XP/XPe/Vista/7/WES, not under WEC/CE.

---

#### ● Swap file



Successful writing of a dump may require activation of the Windows swap file (see below). The operating system sees the swap file on the hard disk as additional RAM and uses it regularly. This results in increased hard disk access activity. If a CompactFlash (CF) is used as the operating system disk it may reduce its service life. It is generally advisable to monitor CF cards using suitable MDP diagnostic functions from the application (e.g. FB\_SiliconDrive\_Read). The swap file then only has to be activated temporarily for diagnostic purposes.

---

#### ● Write protection



Active write protection of the hard disk may need to be removed for successful writing. Examples of write protect filters include:

- EWF (Enhanced Write Filter)
  - FBWF (File Based Write Filter)
- 

There are different types of memory dump, depending on the range and size of the monitored memory area. The corresponding settings are implemented in the Windows system administration.

- Small memory dump (64 kB)
- Kernel
- Complete

Beckhoff may analyze the core image available as a file on the hard disk after a system restart, if the restart was caused by TwinCAT, for example.

A “complete dump” offers the most comprehensive analysis option, although the size may be several MB. In the settings please ensure that:

- the hard disk/CF card has enough memory for the dump.
- the set file path is valid.

The following settings apply for a minimum dump:

1. first activate the swap file.  
Windows System Properties --> Advanced --> Performance Settings

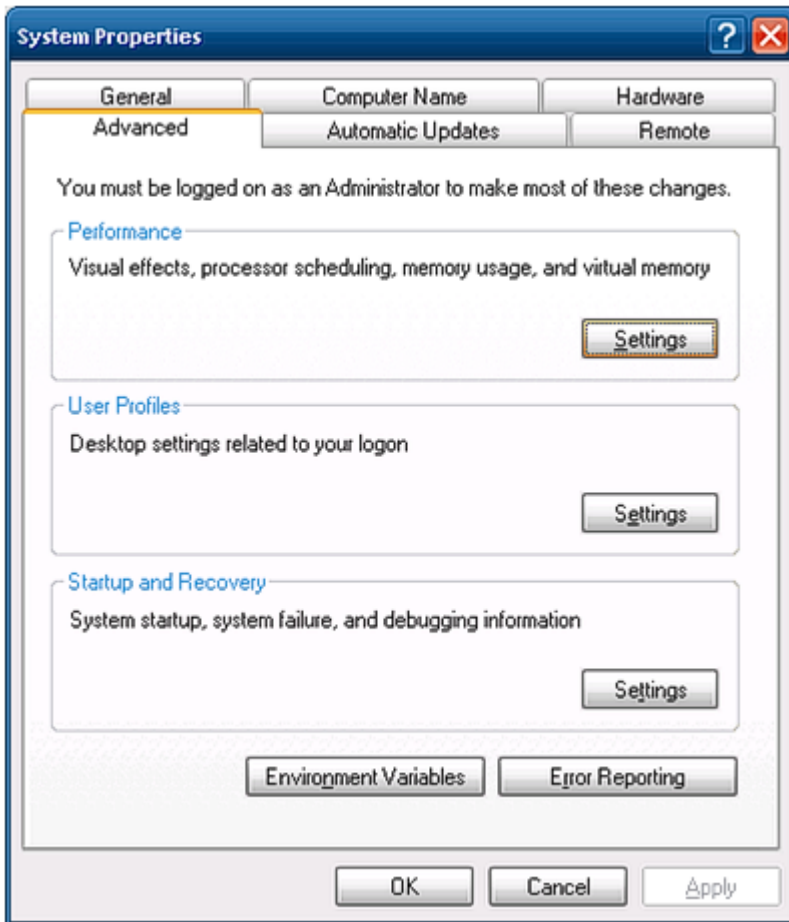


Fig. 344: System Properties

2. Performance Options --> Virtual Memory --> Change



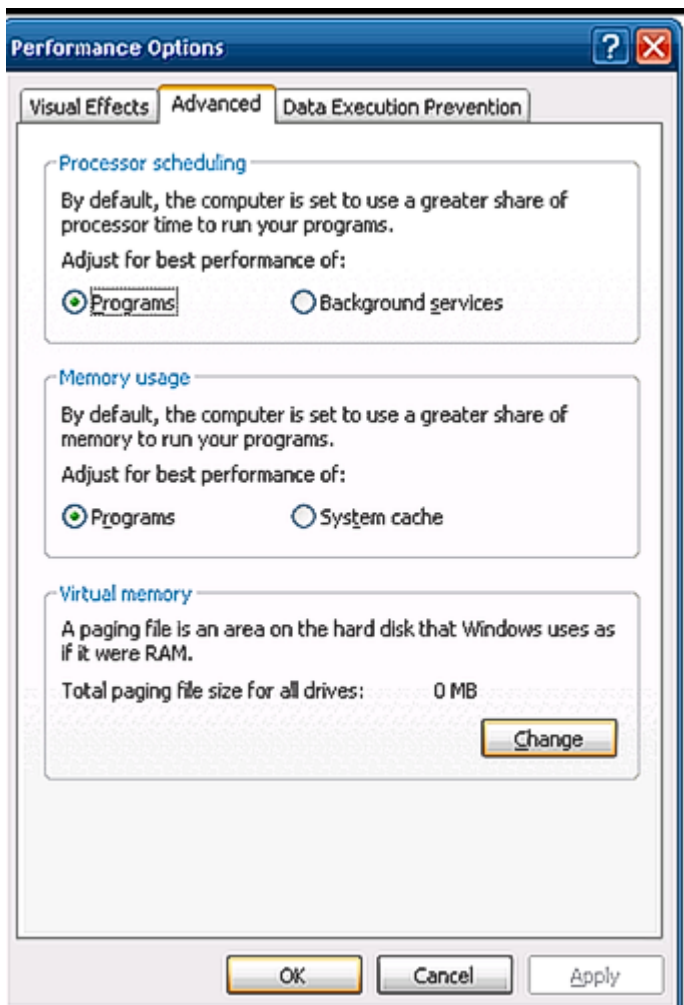


Fig. 345: virtual memory settings

### 3. Swap file specification

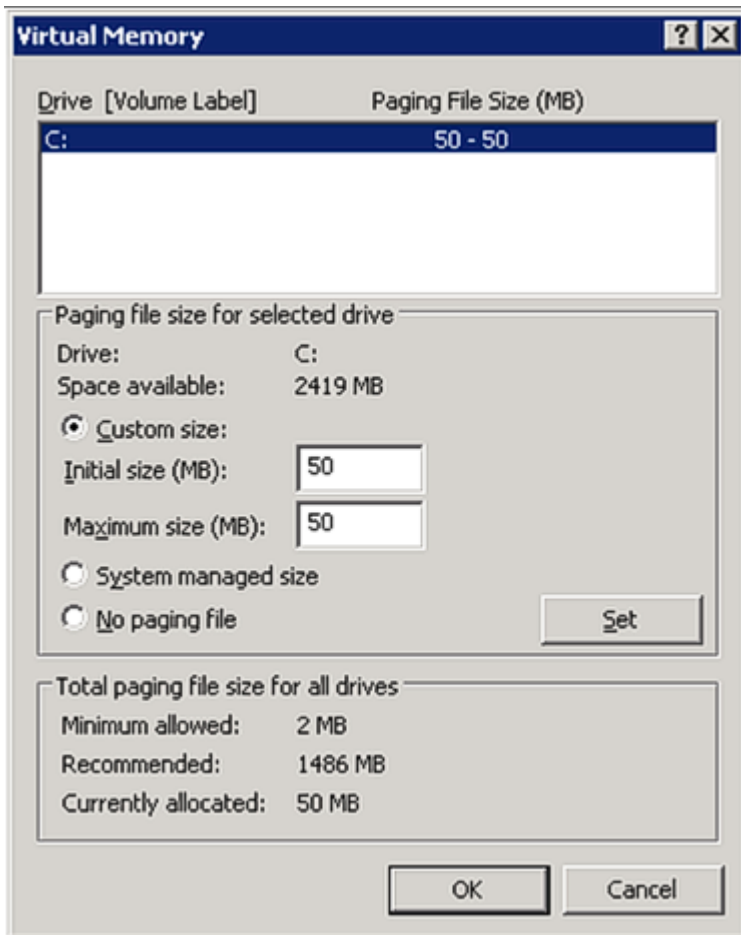


Fig. 346: swap file settings

“Initial Size” and “Maximum Size” should be at least 50 MB. Confirm with SET. Clicking OK is not sufficient! Confirm with OK.

1. In the second step writing of the dump is activated in the Windows system settings (Fig. *System Properties*).
  - > Startup and Recovery
  - > Select “Small memory dump” under “Write debugging information”

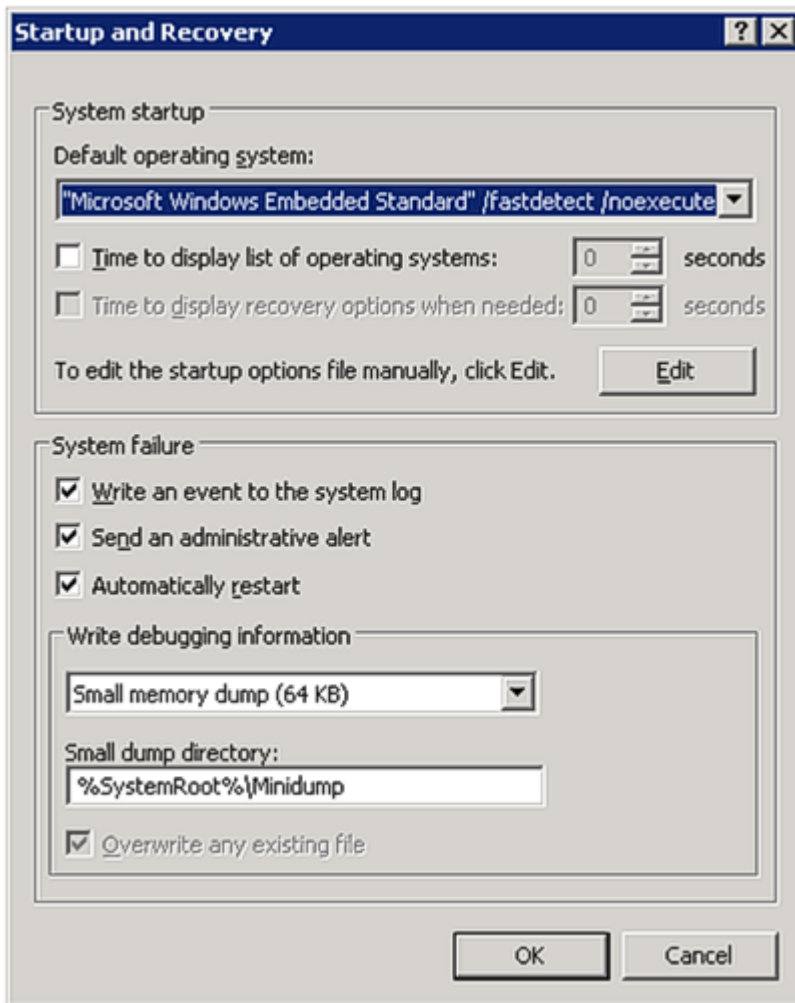


Fig. 347: dump settings

2. You then need to restart the IPC.

After the restart a dump file with current time stamp can be found at the location specified above, if writing of the dump was successful.

Depending on the path settings, complete dumps are saved as "Memory.dmp" under C:\Windows\, small dumps as "Mini.dmp" in subfolder C:\Windows\Minidump\, for example.

## 9.2 MessageBox in the target system

### Situation

On the target system (with MS XP/XPe/7/WES) TwinCAT message boxes ("PopUp windows") are displayed during operation which require confirmation, for example by clicking "OK". If target system is in production mode without HMI (visualization, monitor, keyboard, mouse) and/or no machine operator is present, the message box cannot be confirmed. In this case logger outputs are no longer displayed on the programming device/TwinCAT linked via ADS.

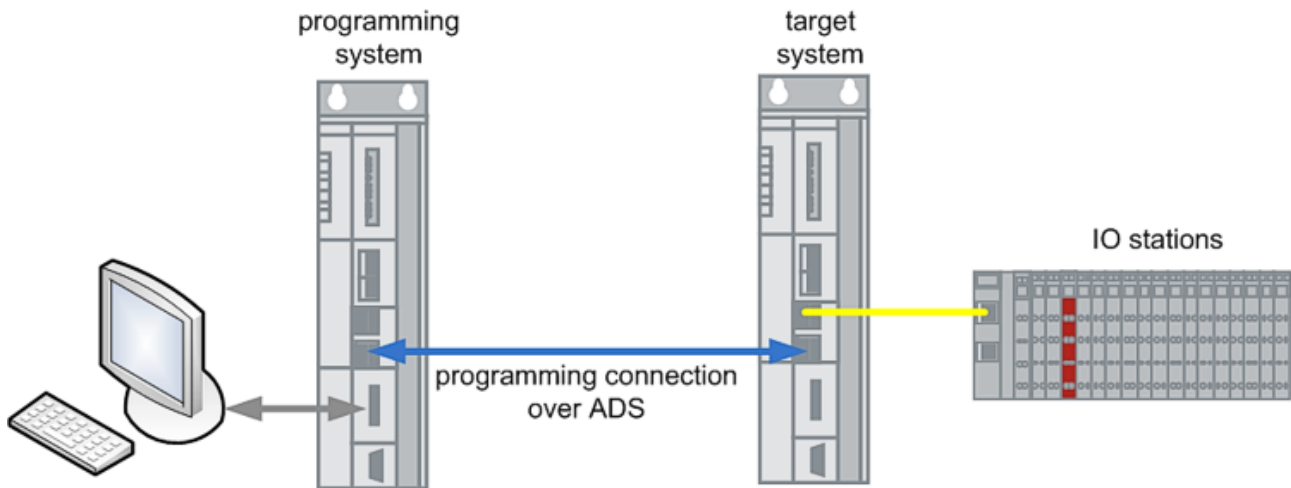


Fig. 348: programming system / target system topology

### Solution




Set a RegistryKey on the target system to suppress the TwinCAT information window.

 RegistryKey (<https://infosys.beckhoff.com/content/1033/ethercatsystem/Resources/zip/2469160331.zip>)

It is *not* advisable to set this key as default, because TwinCAT information windows would no longer be displayed at all.

# 10 Appendix

## 10.1 UL notice

	<p><b>Application</b>                  Beckhoff EtherCAT modules are intended for use with Beckhoff's UL Listed EtherCAT System only.</p>
	<p><b>Examination</b>                  For cULus examination, the Beckhoff I/O System has only been investigated for risk of fire and electrical shock (in accordance with UL508 and CSA C22.2 No. 142).</p>
	<p><b>For devices with Ethernet connectors</b>                  Not for connection to telecommunication circuits.</p>

### Basic principles

UL certification according to UL508. Devices with this kind of certification are marked by this sign:



## 10.2 Training

### Beckhoff

Beckhoff offers training courses on the following subjects, amongst others:

- TwinCAT training for users
- EtherCAT training for users (TR8020)  
The commissioning and diagnostics of EtherCAT devices are dealt with here.
- Training courses for developers on the basis of the Beckhoff development products for EtherCAT implementation

Contact: <http://www.beckhoff.de/training>

### ETG ([www.ethercat.org](http://www.ethercat.org))

[www.ethercat.org](http://www.ethercat.org)

The ETG offers, for example:

- Developer Basics
- Customized trainings

## 10.3 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

### Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for [local support and service](#) on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages:

<http://www.beckhoff.com>

You will also find further [documentation](#) for Beckhoff components there.

### Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20  
33415 Verl  
Germany

Phone: +49 5246 963 0  
Fax: +49 5246 963 198  
e-mail: [info@beckhoff.com](mailto:info@beckhoff.com)

### Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline: +49 5246 963 157  
Fax: +49 5246 963 9157  
e-mail: support@beckhoff.com

**Beckhoff Service**

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline: +49 5246 963 460  
Fax: +49 5246 963 479  
e-mail: service@beckhoff.com

## Table of figures

Fig. 1	Dynamic telegram processing .....	10
Fig. 2	Full duplex Ethernet in the ring, one frame for many devices. The EtherCAT system architecture increases the communication efficiency. ....	11
Fig. 3	Operating system-compatible TwinCAT Y driver: .....	12
Fig. 4	EtherCAT protocol structure .....	13
Fig. 5	Sample: Ethernet topology .....	14
Fig. 6	Decentralization through intelligent interface terminals on the EtherCAT fieldbus .....	16
Fig. 7	Switch port terminals tunnel Ethernet frames through the EtherCAT protocol .....	18
Fig. 8	States of the EtherCAT State Machine .....	19
Fig. 9	EL3152 CoE directory .....	22
Fig. 10	Online/Offline display .....	22
Fig. 11	Manual insertion of a startup entry .....	24
Fig. 12	Edit .....	25
Fig. 13	Changed startup list .....	25
Fig. 14	Opening the *. tzip archive .....	27
Fig. 15	Selecting the "Restore default parameters" PDO .....	29
Fig. 16	CoE reset as a startup entry .....	29
Fig. 17	Entering a restore value in the Set Value dialog .....	30
Fig. 18	The cable redundancy principle .....	30
Fig. 19	Special solution for cable redundancy .....	31
Fig. 20	Setting up the device .....	32
Fig. 21	Parameterizing the redundancy .....	33
Fig. 22	Structure of an Ethernet frame with EtherCAT protocol data .....	33
Fig. 23	TwinCAT view, frame structure - small topology .....	34
Fig. 24	TwinCAT view, frame structure - larger topology .....	34
Fig. 25	WC = 1 indicates invalid data .....	35
Fig. 26	Entering the SyncUnits in a default frame structure .....	36
Fig. 27	New frame structure with SyncUnits .....	36
Fig. 28	Available information .....	37
Fig. 29	Typical error patterns in the redundancy case .....	38
Fig. 30	Star topology .....	40
Fig. 31	Star topology with cable redundancy – presently not permitted .....	40
Fig. 32	Line topology .....	40
Fig. 33	Line topology with cable redundancy is not possible .....	41
Fig. 34	Stacked groups .....	41
Fig. 35	Free ports .....	41
Fig. 36	Identification of FHC port at EK1122-0080 and EK1101-0080 .....	42
Fig. 37	Recommended combination of Ethernet ports .....	42
Fig. 38	Detection of incorrect port allocation in the TwinCAT logger .....	43
Fig. 39	Configuration of a Fast Hot Connect group .....	43
Fig. 40	Marking in the TwinCAT System Manager .....	44
Fig. 41	DC master setting .....	44
Fig. 42	Default setting of an EtherCAT slave that supports the DataWord identification mode .....	45
Fig. 43	EK1101 with ID switch .....	46



Fig. 44	setting the SSA.....	47
Fig. 45	Checking the SSA.....	47
Fig. 46	Setting SSA under TwinCAT 2.11 .....	48
Fig. 47	Creating the configuration.....	49
Fig. 48	Creating the configuration.....	49
Fig. 49	Assign HotConnect group.....	49
Fig. 50	Assign address .....	50
Fig. 51	Hot Connect group.....	50
Fig. 52	Topology view.....	50
Fig. 53	Topology view.....	50
Fig. 54	Online topology view.....	51
Fig. 55	Online display .....	51
Fig. 56	HotConnect tab.....	52
Fig. 57	PreviousPort at bus station.....	52
Fig. 58	Calling up the HotConnect dialog .....	53
Fig. 59	HotConnect features.....	53
Fig. 60	HotConnect features.....	54
Fig. 61	Setting detail .....	54
Fig. 62	Status messages .....	55
Fig. 63	Group A in operation.....	55
Fig. 64	Process data of the ID module .....	55
Fig. 65	Offline view .....	56
Fig. 66	Online view .....	56
Fig. 67	Online view .....	56
Fig. 68	Current connection status.....	57
Fig. 69	Demo structure .....	57
Fig. 70	AdsAddr .....	58
Fig. 71	Switching on the "ADS Address" in the Advanced Settings .....	58
Fig. 72	Information structure.....	59
Fig. 73	Topologies for different data exchange methods.....	61
Fig. 74	TwinSAFE application manual.....	62
Fig. 75	Relationship between user side (commissioning) and installation.....	64
Fig. 76	Control configuration with Embedded PC, input (EL1004) and output (EL2008) .....	65
Fig. 77	Initial TwinCAT 2 user interface.....	66
Fig. 78	Selection of the target system .....	67
Fig. 79	Specify the PLC for access by the TwinCAT System Manager: selection of the target system ..	67
Fig. 80	Select "Scan Devices..." .....	68
Fig. 81	Automatic detection of I/O devices: selection the devices to be integrated.....	68
Fig. 82	Mapping of the configuration in the TwinCAT 2 System Manager.....	69
Fig. 83	Reading of individual terminals connected to a device.....	69
Fig. 84	TwinCAT PLC Control after startup .....	70
Fig. 85	Sample program with variables after a compile process (without variable integration) .....	71
Fig. 86	Appending the TwinCAT PLC Control project .....	71
Fig. 87	PLC project integrated in the PLC configuration of the System Manager .....	72
Fig. 88	Creating the links between PLC variables and process objects.....	72
Fig. 89	Selecting PDO of type BOOL .....	73

Fig. 90	Selecting several PDOs simultaneously: activate "Continuous" and "All types" .....	73
Fig. 91	Application of a "Goto Link" variable, using "MAIN.bEL1004_Ch4" as a sample .....	74
Fig. 92	Choose target system (remote) .....	75
Fig. 93	PLC Control logged in, ready for program startup .....	76
Fig. 94	Initial TwinCAT 3 user interface .....	77
Fig. 95	Create new TwinCAT project .....	77
Fig. 96	New TwinCAT3 project in the project folder explorer .....	78
Fig. 97	Selection dialog: Choose the target system .....	78
Fig. 98	Specify the PLC for access by the TwinCAT System Manager: selection of the target system ..	79
Fig. 99	Select "Scan" .....	79
Fig. 100	Automatic detection of I/O devices: selection the devices to be integrated .....	80
Fig. 101	Mapping of the configuration in VS shell of the TwinCAT3 environment .....	80
Fig. 102	Reading of individual terminals connected to a device .....	81
Fig. 103	Adding the programming environment in "PLC" .....	82
Fig. 104	Specifying the name and directory for the PLC programming environment .....	82
Fig. 105	Initial "Main" program of the standard PLC project .....	83
Fig. 106	Sample program with variables after a compile process (without variable integration) .....	84
Fig. 107	Start program compilation .....	84
Fig. 108	Creating the links between PLC variables and process objects .....	85
Fig. 109	Selecting PDO of type BOOL .....	85
Fig. 110	Selecting several PDOs simultaneously: activate "Continuous" and "All types" .....	86
Fig. 111	Application of a "Goto Link" variable, using "MAIN.bEL1004_Ch4" as a sample .....	86
Fig. 112	Creating a PLC data type .....	87
Fig. 113	Instance_of_struct .....	87
Fig. 114	Linking the structure .....	88
Fig. 115	Reading a variable from the structure of the process data .....	88
Fig. 116	TwinCAT development environment (VS shell): logged-in, after program startup .....	89
Fig. 117	EtherCAT device in the configuration tree .....	90
Fig. 118	EtherCAT master in the IPC environment .....	90
Fig. 119	synchronous execution of task and EtherCAT communication .....	91
Fig. 120	Distinction between fixed (cyclic) and variable EtherCAT telegrams (acyclic) .....	91
Fig. 121	Configuration-dependent time division in the System Manager .....	92
Fig. 122	Priority list in the TwinCAT System Manager .....	93
Fig. 123	Tasks with own Ethernet frames (Task 1 ms: "red" frame, task 10 ms: "yellow" frame) .....	93
Fig. 124	System Manager view – current calculation .....	94
Fig. 125	Components of the EtherCAT device .....	96
Fig. 126	Simplified illustration of the entire EtherCAT device .....	97
Fig. 127	Context menu "Append Box" .....	99
Fig. 128	Selecting and appending an EtherCAT device .....	99
Fig. 129	Activate checkbox "Extended Information" to display revision .....	99
Fig. 130	Context menu "Scan Boxes" .....	100
Fig. 131	Context menu "Scan Boxes" .....	100
Fig. 132	Configuration comparison .....	101
Fig. 133	Reading product information from EEPROM .....	101
Fig. 134	Exporting the information .....	101
Fig. 135	Reading the information from identity object 1018 .....	102

Fig. 136 EL5021 EL terminal, standard IP20 IO device with serial/ batch number and revision ID (since 2014/01)..... 103

Fig. 137 EK1100 EtherCAT coupler, standard IP20 IO device with serial/ batch number..... 104

Fig. 138 CU2016 switch with serial/ batch number..... 104

Fig. 139 EL3202-0020 with serial/ batch number 26131006 and unique ID-number 204418 ..... 104

Fig. 140 EP1258-00001 IP67 EtherCAT Box with batch number/ date code 22090101 and unique serial number 158102..... 105

Fig. 141 EP1908-0002 IP67 EtherCAT Safety Box with batch number/ date code 071201FF and unique serial number 00346070 ..... 105

Fig. 142 EL2904 IP20 safety terminal with batch number/ date code 50110302 and unique serial number 00331701..... 105

Fig. 143 ELM3604-0002 terminal with unique ID number (QR code) 100001051 and serial/ batch number 44160201..... 105

Fig. 144 BIC as data matrix code (DMC, code scheme ECC200)..... 106

Fig. 145 System manager Online View -> Properties..... 109

Fig. 146 Activate "Show Production Info" ..... 109

Fig. 147 Online display ..... 109

Fig. 148 TwinCAT 3.1: Activate "Show Production Info"..... 110

Fig. 149 Export ..... 110

Fig. 150 Sample of the contents of the csv file ..... 110

Fig. 151 Visualization in the sample program..... 111

Fig. 152 Filtering in the spreadsheet program ..... 111

Fig. 153 System Manager "Options" (TwinCAT 2)..... 112

Fig. 154 Call up under VS Shell (TwinCAT 3) ..... 112

Fig. 155 Overview of network interfaces ..... 112

Fig. 156 EtherCAT device properties(TwinCAT 2): click on "Compatible Devices..." of tab "Adapte" ..... 113

Fig. 157 Windows properties of the network interface..... 113

Fig. 158 Exemplary correct driver setting for the Ethernet port ..... 114

Fig. 159 Incorrect driver settings for the Ethernet port ..... 115

Fig. 160 TCP/IP setting for the Ethernet port ..... 116

Fig. 161 Identifier structure ..... 117

Fig. 162 OnlineDescription information window (TwinCAT 2) ..... 118

Fig. 163 Information window OnlineDescription (TwinCAT 3)..... 118

Fig. 164 File OnlineDescription.xml created by the System Manager ..... 119

Fig. 165 Indication of an online recorded ESI of EL2521 as an example..... 119

Fig. 166 Information window for faulty ESI file (left: TwinCAT 2; right: TwinCAT 3)..... 119

Fig. 167 Using the ESI Updater (>= TwinCAT 2.11)..... 121

Fig. 168 Using the ESI Updater (TwinCAT 3)..... 121

Fig. 169 Append EtherCAT device (left: TwinCAT 2; right: TwinCAT 3) ..... 122

Fig. 170 Selecting the EtherCAT connection (TwinCAT 2.11, TwinCAT 3)..... 122

Fig. 171 Selecting the Ethernet port ..... 122

Fig. 172 EtherCAT device properties (TwinCAT 2) ..... 123

Fig. 173 Appending EtherCAT devices (left: TwinCAT 2; right: TwinCAT 3)..... 123

Fig. 174 Selection dialog for new EtherCAT device ..... 124

Fig. 175 Display of device revision ..... 124

Fig. 176 Display of previous revisions ..... 125

Fig. 177 Name/revision of the terminal..... 125

Fig. 178 EtherCAT terminal in the TwinCAT tree (left: TwinCAT 2; right: TwinCAT 3).....	126
Fig. 179 Differentiation local/target system (left: TwinCAT 2; right: TwinCAT 3).....	127
Fig. 180 Scan Devices (left: TwinCAT 2; right: TwinCAT 3).....	127
Fig. 181 Note for automatic device scan (left: TwinCAT 2; right: TwinCAT 3).....	127
Fig. 182 Detected Ethernet devices .....	128
Fig. 183 Example default state .....	128
Fig. 184 Installing EtherCAT terminal with revision -1018 .....	129
Fig. 185 Detection of EtherCAT terminal with revision -1019 .....	129
Fig. 186 Scan query after automatic creation of an EtherCAT device (left: TwinCAT 2; right: TwinCAT 3) .....	129
Fig. 187 Manual triggering of a device scan on a specified EtherCAT device (left: TwinCAT 2; right: TwinCAT 3).....	130
Fig. 188 Scan progress exemplary by TwinCAT 2 .....	130
Fig. 189 Config/FreeRun query (left: TwinCAT 2; right: TwinCAT 3).....	130
Fig. 190 Displaying of “Free Run” and “Config Mode” toggling right below in the status bar .....	130
Fig. 191 TwinCAT can also be switched to this state by using a button (left: TwinCAT 2; right: TwinCAT 3) .....	130
Fig. 192 Online display example .....	131
Fig. 193 Faulty identification .....	131
Fig. 194 Identical configuration (left: TwinCAT 2; right: TwinCAT 3).....	132
Fig. 195 Correction dialog .....	132
Fig. 196 Name/revision of the terminal.....	133
Fig. 197 Correction dialog with modifications .....	134
Fig. 198 Dialog “Change to Compatible Type...” (left: TwinCAT 2; right: TwinCAT 3).....	134
Fig. 199 TwinCAT 2 Dialog Change to Alternative Type .....	134
Fig. 200 online diagnostics for EtherCAT device.....	137
Fig. 201 Process data.....	137
Fig. 202 Master settings .....	138
Fig. 203 Slave settings .....	140
Fig. 204 Sync Task.....	142
Fig. 205 Distributed Clocks.....	143
Fig. 206 Ethernet over EtherCAT support .....	143
Fig. 207 Activating the VirtualEthernetSwitch.....	144
Fig. 208 EtherCAT cable redundancy.....	145
Fig. 209 Sync Unit assignment.....	146
Fig. 210 topology of the EtherCAT system with DC-capable devices .....	147
Fig. 211 distributed clock master settings.....	148
Fig. 212 "DcSysTime" display in the TwinCAT tree.....	149
Fig. 213 DevState with SyncWindow display monitoring.....	150
Fig. 214 Manual selection of the reference clock .....	150
Fig. 215 Distributed Clocks tab (DC) .....	150
Fig. 216 checkbox for manual selection .....	150
Fig. 217 Event logger message: no support by reference clock.....	151
Fig. 218 Event logger message: no synchronization .....	151
Fig. 219 Default slave setting – slave with and without DC capability.....	152
Fig. 220 logger window showing information relating to invalid DC configuration of the slave.....	153
Fig. 221 display of additional register values from the slaves .....	153

Fig. 222 DC register selection .....	153
Fig. 223 display of DC register values .....	154
Fig. 224 distributed clocks system topology .....	155
Fig. 225 synchronization direction .....	155
Fig. 226 DC-coupling of EtherCAT systems .....	156
Fig. 227 EtherCAT topology with CU2508 and 2 EtherCAT systems .....	157
Fig. 228 Selection of the diagnostic information of an EtherCAT Slave .....	158
Fig. 229 Basic EtherCAT Slave Diagnosis in the PLC.....	159
Fig. 230 EL3102, CoE directory .....	161
Fig. 231 Example of commissioning aid for a EL3204 .....	162
Fig. 232 Default behaviour of the System Manager .....	163
Fig. 233 Default target state in the Slave .....	163
Fig. 234 PLC function blocks .....	164
Fig. 235 Illegally exceeding the E-Bus current .....	165
Fig. 236 Warning message for exceeding E-Bus current .....	165
Fig. 237 System Manager icons for synchronous/asynchronous mapping .....	189
Fig. 238 Automatically set-up asynchronous mapping .....	190
Fig. 239 TC tree: assignments, list of links .....	190
Fig. 240 "Mapping" tab .....	190
Fig. 241 Variable name.....	192
Fig. 242 Zoom factor setting .....	192
Fig. 243 Tab "B -> A" .....	192
Fig. 244 Online display "asynchronous assignments" .....	193
Fig. 245 Online display "synchronous assignments" .....	193
Fig. 246 Simple I/O topology .....	198
Fig. 247 mapping of DC to the topology .....	199
Fig. 248 EtherCAT topology with external reference clock.....	200
Fig. 249 TwinCAT 2.11 distributed clock settings - Sample for EL6688 in PTP slave mode as time reference for the local EtherCAT system .....	201
Fig. 250 synchronization direction .....	202
Fig. 251 display current offsets.....	202
Fig. 252 Current offsets .....	203
Fig. 253 Example for ESC functionality and embedding in a typical EtherCAT slave .....	206
Fig. 254 Distributed clocks in the EtherCAT system.....	207
Fig. 255 Interfaces of the distributed clock unit .....	209
Fig. 256 Sample application with distributed clocks, cycle time 100 µs.....	210
Fig. 257 EL1202-0100 read timing diagram .....	211
Fig. 258 Time specification of the SYNC pulse for the local slave with regard to the real-time tick of the master PC.....	213
Fig. 259 EtherCAT master – advanced settings .....	214
Fig. 260 EtherCAT- Master - Distributed Clocks.....	215
Fig. 261 DcSysTime input variable (can be activated) .....	216
Fig. 262 DC tab.....	217
Fig. 263 Distributed clocks dialog in the EtherCAT slave .....	217
Fig. 264 Sample application for a manual shift time on SYNC0 .....	219
Fig. 265 Topology and principle of operation used in the example .....	221

Fig. 266 TwinCAT EtherCAT master setting, section DistributedClocks .....	222
Fig. 267 Process data .....	222
Fig. 268 EtherCAT Update (schematic) .....	223
Fig. 269 Local slave shift time .....	224
Fig. 270 EtherCAT master setting .....	225
Fig. 271 EtherCAT slave setting .....	226
Fig. 272 Shift calculation in the slave dialogue .....	226
Fig. 273 Displaying the DC shift times .....	227
Fig. 274 Continuous setting .....	227
Fig. 275 Runtime Measuring .....	227
Fig. 276 DevState input .....	228
Fig. 277 DcSysTime inputs .....	228
Fig. 278 Online DC diagnosis .....	229
Fig. 279 Online DC diagnosis - adjustment required .....	229
Fig. 280 PLC Libraries .....	230
Fig. 281 Sample configuration with 500 $\mu$ s cycle time .....	232
Fig. 282 Sample configuration with 500 $\mu$ s cycle time .....	232
Fig. 283 Methods of reading the DC time .....	232
Fig. 284 Local cycle "Free Run" synchronization .....	236
Fig. 285 Local cycle, synchronization with SM2/3 event .....	237
Fig. 286 Local cycle, synchronization with SM2/3 event, shifting of "Input Latch" .....	239
Fig. 287 Local cycle, synchronization with SYNC0 event .....	241
Fig. 288 Local cycle, synchronization with SYNC0 event, shifting of inputs/outputs .....	243
Fig. 289 DC, subordinate $\mu$ C cycles, inputs/outputs delayed .....	246
Fig. 290 DC tab for indicating the Distributed Clocks function .....	248
Fig. 291 Advanced Distributed Clocks settings in the EtherCAT master .....	249
Fig. 292 TwinCAT setting for using this component as reference clock .....	250
Fig. 293 TwinCAT 2.11 clock hierarchy (subject to change) .....	251
Fig. 294 Frequency synchronicity .....	251
Fig. 295 Synchronization of 2 TwinCAT IPCs with the aid of EtherCAT components .....	252
Fig. 296 Cascade consisting of controlled TwinCAT systems .....	253
Fig. 297 Topology of the sample program .....	254
Fig. 298 Device on the master side .....	255
Fig. 299 Set the EL6692 PrimarySide to DC .....	255
Fig. 300 Activate PDO 0x1A02 to display the time stamps .....	256
Fig. 301 Set the synchronization direction on the PrimarySide; in this case SyncSettings: Primary --> Secondary .....	256
Fig. 302 Activate the display of the DC offsets in the EtherCAT master; can be evaluated on the master side .....	257
Fig. 303 Master PC works with its own ReferenceClock as a basis .....	257
Fig. 304 SecondarySide of the EL6692 .....	258
Fig. 305 EtherCAT master settings, slave side .....	258
Fig. 306 Slave side .....	259
Fig. 307 Start slave side .....	259
Fig. 308 Time stamp known .....	260
Fig. 309 Synchronization successful .....	260

Fig. 310 Branch element as terminal EL3751.....	261
Fig. 311 "General" tab.....	261
Fig. 312 "EtherCAT" tab.....	262
Fig. 313 "Process Data" tab.....	263
Fig. 314 Configuring the process data.....	264
Fig. 315 "Startup" tab.....	265
Fig. 316 "CoE - Online" tab.....	266
Fig. 317 Dialog "Advanced settings".....	267
Fig. 318 "Online" tab.....	267
Fig. 319 "DC" tab (Distributed Clocks).....	268
Fig. 320 "Process data" tab.....	271
Fig. 321 "Process data" tab.....	271
Fig. 322 entry.....	272
Fig. 323 Advanced Settings: "Behavior" dialog.....	272
Fig. 324 Display the channels in InfoData via "Advanced settings" -> "Behavior" "Include Channels.....	275
Fig. 325 Display of linkable channels in TwinCAT.....	275
Fig. 326 Display of the activation of "Include Channels" in the EtherCAT Master.....	276
Fig. 327 Advanced Settings: "Timeout Settings" dialog.....	277
Fig. 328 Advanced Settings: "FMMU/SM" dialog.....	278
Fig. 329 Advanced Settings: "Mailbox" dialog.....	279
Fig. 330 Advanced Settings: "Smart View" dialog.....	281
Fig. 331 Advanced Settings: "Memory" dialog.....	283
Fig. 332 Device identifier consisting of name EL3204-0000 and revision -0016.....	285
Fig. 333 Scan the subordinate field by right-clicking on the EtherCAT device.....	286
Fig. 334 Configuration is identical.....	286
Fig. 335 Change dialog.....	286
Fig. 336 EEPROM Update.....	287
Fig. 337 Selecting the new ESI.....	287
Fig. 338 Display of EL3204 firmware version.....	288
Fig. 339 Firmware Update.....	289
Fig. 340 FPGA firmware version definition.....	291
Fig. 341 Context menu Properties.....	291
Fig. 342 Dialog Advanced Settings.....	292
Fig. 343 Multiple selection and firmware update.....	294
Fig. 344 System Properties.....	296
Fig. 345 virtual memory settings.....	297
Fig. 346 swap file settings.....	298
Fig. 347 dump settings.....	299
Fig. 348 programming system / target system topology.....	300





Beckhoff Automation GmbH & Co. KG  
Hülshorstweg 20  
33415 Verl  
Germany  
Phone: +49 5246 9630  
[info@beckhoff.com](mailto:info@beckhoff.com)  
[www.beckhoff.com](http://www.beckhoff.com)