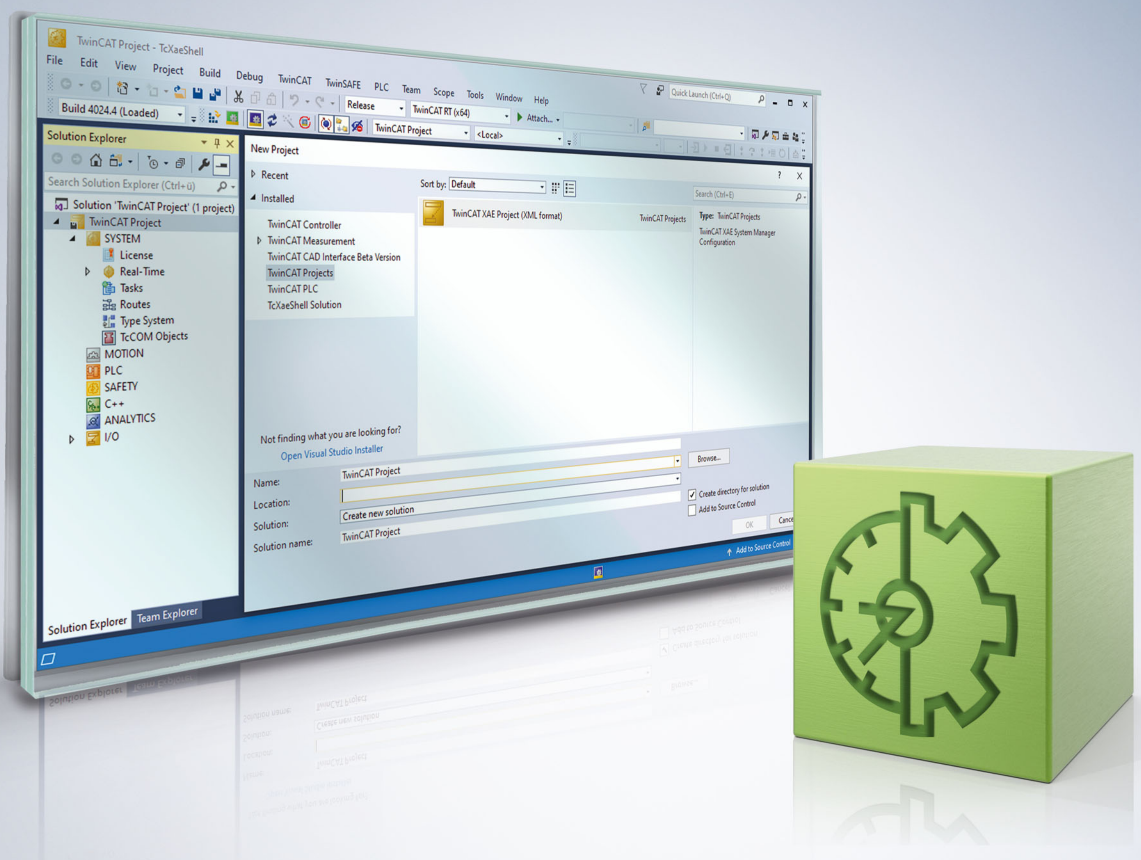
BECKHOFF New Automation Technology

Manual | EN

TE1111

TwinCAT 3 | EtherCAT Simulation



2023-02-06 | Version: 1.5.2

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the 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

Trademarks

Patent Pending

Copyright

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 symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

New machines and plants are becoming ever more complex and must be realized cost-effectively under considerable deadline pressure. Beyond that, the control software portion is continuously increasing, among other things due to the demand for flexibility with regard to the products that can be produced on a plant. A reduction of the engineering expenditure for the generation of the control code as well as the commissioning of this code in the actual plant thus promises a clear economic benefit. One of the methods aimed at achieving this is called “virtual commissioning”. The objective is to test and optimize the generated control code at an early stage of the engineering, even without any real existing hardware, so that the actual commissioning can proceed much faster.

The **TE1111 TwinCAT EtherCAT Simulation** function has been created in order to fulfill these requirements. If, in addition, models of the machine/plant are already available in Matlab Simulink or a simulation tool that offers FMU (code) export, it is possible to perform HIL (Hardware-In-the-Loop) simulations together with the TE1400 Target for Matlab®/Simulink® or **TE1420 Target for FMI** functions with manageable effort.

The **TE1111 TwinCAT EtherCAT Simulation** function simulates an EtherCAT segment. An already created I/O configuration of the real plant is exported and can be imported into a second system as an ‘EtherCAT Simulation’ device. A mirrored process image is thus available on this system that can be linked with corresponding TwinCAT modules (e.g. written in the IEC 61131-3 languages or generated from Matlab®/ Simulink®). The desired behavior of the machine must be implemented with sufficient accuracy in these modules. If the TwinCAT real-time is now started on both systems, one has an HIL simulation without having to adapt the original project in order to achieve this.

Since the project for the control system to be tested should not be changed, the EtherCAT Simulation Device operates unsynchronized. This means that the task that drives the Simulation Device has to run twice as fast on account of Shannon’s theorem (sampling theorem). On standard IPC hardware the shortest cycle time of the simulation device is currently 50µs. HIL simulations of the control system to be tested are thus possible with a cycle time of 100µs.

Features supported:

Basic functionality

- Digital slaves are supported along with slaves with CoE parameters
- A mirrored process image is created and can be linked
- Simple configuration of the EtherCAT Simulation Device directly in the TwinCAT XAE
- Support for distributed clocks
- Support for AoE and SoE mailbox commands (these are forwarded to the PLC)
- Mixed simulation of real and simulated slaves
- HotConnect simulation (from TwinCAT 3.1.4024)
- Fault simulation (Lost Frames, Link Lost, Working Counter) (from TwinCAT 3.1.4024)
- CU2508 Port multiplier support (from TwinCAT 3.1.4024)

3 Installation/ Licensing

The TE1111 | TwinCAT EtherCAT Simulation function is already installed with the TwinCAT 3 installation and is included as a release version since TwinCAT version 3.1 Build 4018.0. It therefore only needs to be licensed. Please refer to chapter Ordering and activating TwinCAT 3 standard licenses. The TE1111 function has an instance-based license, i.e. it is licensed per instantiated EtherCAT Simulation Device.

Required licenses:

TE1111 EtherCAT Simulation

Although this license must be activated on the target system, it is an engineering license. For test purposes, the EtherCAT Simulation can also be used in demo mode without a license.

A fully functional 7-day trial license is not available for this product.

Restrictions in the demo version

If the EtherCAT Simulation is used in demo mode, all simulated I/Os are released after 30 min.

4 Basic principles

The **TE1111 TwinCAT EtherCAT Simulation** function simulates an EtherCAT segment. An already created I/O configuration of the real plant is exported and can be imported into a second system as an ‘EtherCAT Simulation’ device. A mirrored process image is thus available on this system that can be linked with corresponding TwinCAT modules (e.g. written in the IEC 61131-3 languages or generated from Matlab®/ Simulink®). The desired behavior of the machine must be implemented with sufficient accuracy in these modules. If the TwinCAT real-time is now started on both systems, one has an HIL simulation without having to adapt the original project in order to achieve this.

Since the project for the control system to be tested should not be changed, the EtherCAT Simulation Device operates unsynchronized. This means that the task that drives the Simulation Device has to run twice as fast on account of Shannon’s theorem (sampling theorem). On standard IPC hardware the shortest cycle time of the simulation device is currently 50µs. HIL simulations of the control system to be tested are thus possible with a cycle time of 100µs.

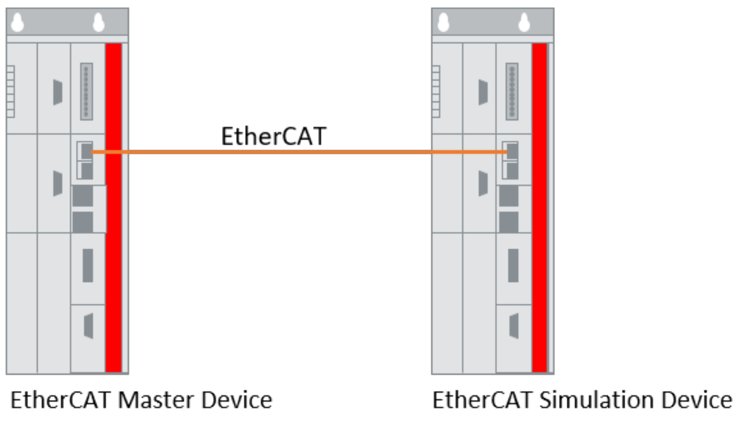
In addition to HIL simulation, EtherCAT simulation can also be used to simulate one or more EtherCAT segments on the same computer, as long as sufficient free network interfaces are available. Depending on the application, various cabling options are available, which are described in the chapter Cabling [▶ 10].

The EtherCAT simulation is essentially limited to the EtherCAT-relevant aspects of the bus devices. The internal behavior of the bus devices themselves is neglected. For some bus devices, however, it is necessary to simulate at least part of the behavior, as otherwise the EtherCAT master cannot be started correctly (e.g. EL6224). To simulate this, the EtherCAT simulation offers the possibility to activate theMailbox Auto Response [▶ 13] option, which automatically answers such requests without having to implement anything. If these parameters are still required, this option can also be deactivated. In this case it is possible to redirect the CoE, SoE and AoE communication into a user PLC module and to react there to these telegrams (see CoE / SoE / AoE [▶ 14]).

Another example in which the pure EtherCAT consideration of the devices is not sufficient are drives. Here it is necessary to implement at least part of the drive state machine so that TwinCAT NC can start correctly. For the drive profiles CoE and SoE, these are implemented as function blocks as examples (see). These function blocks can be instantiated in a PLC module for each drive contained in the configuration and linked to the axes. They are thus switched between the EtherCAT simulation and the process simulation to be implemented.

4.1 Cabling

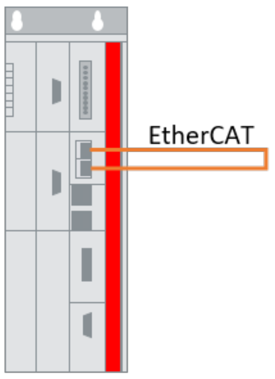
As already described in the chapter Overview [▶ 8], the EtherCAT simulation was designed to support a virtual commissioning of a plant. The normal cabling for this purpose will look like this:



On the left side you can see the control computer, which contains the original control project, which should be used later to control the machine/plant. On the right side you can see a simulation IPC, which contains an EtherCAT simulation device. Both systems are connected by a standard EtherCAT cable.

Use of EtherCAT and EtherCAT simulation on the same IPC:

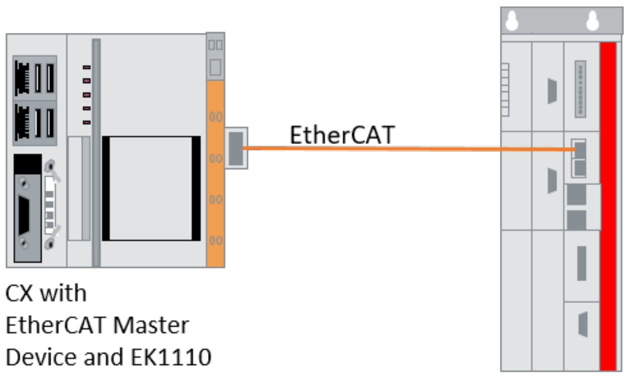
For testing partial functions of the control program (e.g. EtherCAT diagnosis) at an earlier time (before virtual commissioning), the control computer can also be used to simulate the EtherCAT, as long as the control computer has sufficient free network interfaces. Just like every EtherCAT master, every EtherCAT simulation device also requires its own network interface.



EtherCAT Master Device and
EtherCAT Simulation Device

Using a CX as controller

If a CX is used as controller that is to be coupled with a simulation IPC, there may still be EtherCAT devices available depending on the cabling, which must not be taken into account. The following figure shows, for example, the intelligent power supply unit of the CX and an EK1110 EtherCAT extension. Here in the Settings tab [▶ 24] a manual device correction can be set, i.e. devices that should not be considered. If this setting is not set, this could result in the EtherCAT master not starting up or the working counter not being calculated correctly.

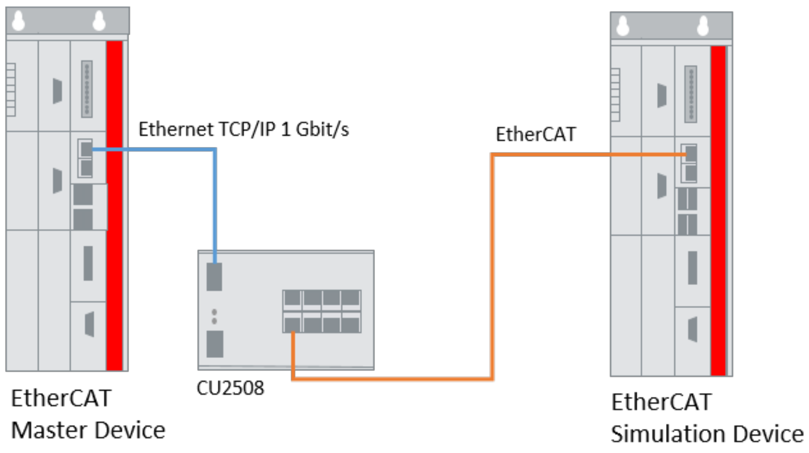


CX with
EtherCAT Master
Device and EK1110

IPC with
EtherCAT Simulation Device

Use of the port multiplier on the original project page:

If a CU2508 port multiplier is used in the real-time configuration of the control computer, the outputs of the port multiplier can be connected to normal network ports on a simulation computer. For these network ports, a corresponding number of EtherCAT simulation devices must be created on the simulation computer.



NOTE

Use of the port multiplier on the EtherCAT simulation side not supported

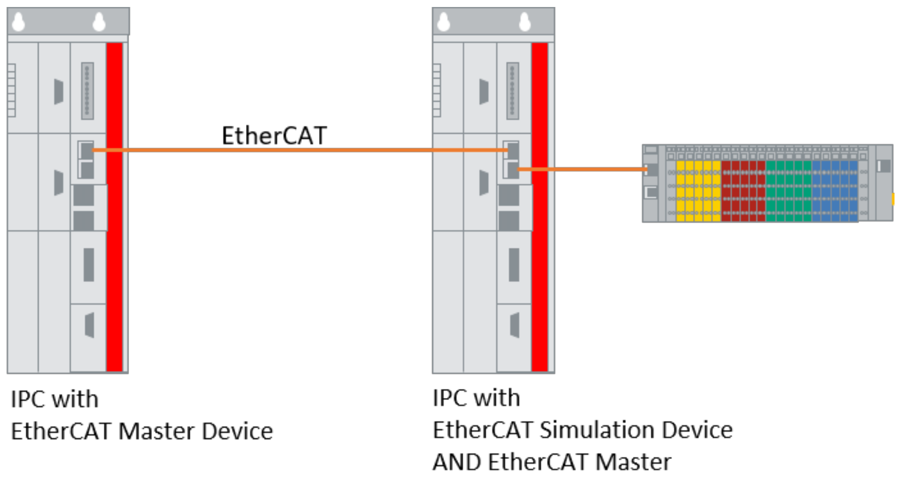
The use of a CU2508 port multiplier on the EtherCAT simulation side is not supported. If a CU2508 is used on the slave side, it is not contained in the TwinCAT project that exports the configuration file. This CU2508 also only receives the EtherCAT frame after the EtherCAT simulation device. It is thus “invisible” to the EtherCAT simulation. Consequently, neither the Device State nor the Slave Count on the EtherCAT master side display correct values. An EtherCAT diagnosis programmed in the original project would thus display incorrect results.

Use of the port multiplier on the EtherCAT simulation side:

The use of a CU2508 port multiplier on the EtherCAT simulation side is not supported. If a CU2508 is used on the slave side, it is not contained in the TwinCAT project that exports the configuration file. This CU2508 also only receives the EtherCAT frame after the EtherCAT simulation device. It is thus “invisible” to the EtherCAT simulation. Consequently, neither the Device State nor the Slave Count on the EtherCAT master side display correct values. An EtherCAT diagnosis programmed in the original project would thus display incorrect results.

Operation of the EtherCAT simulation in mixed mode

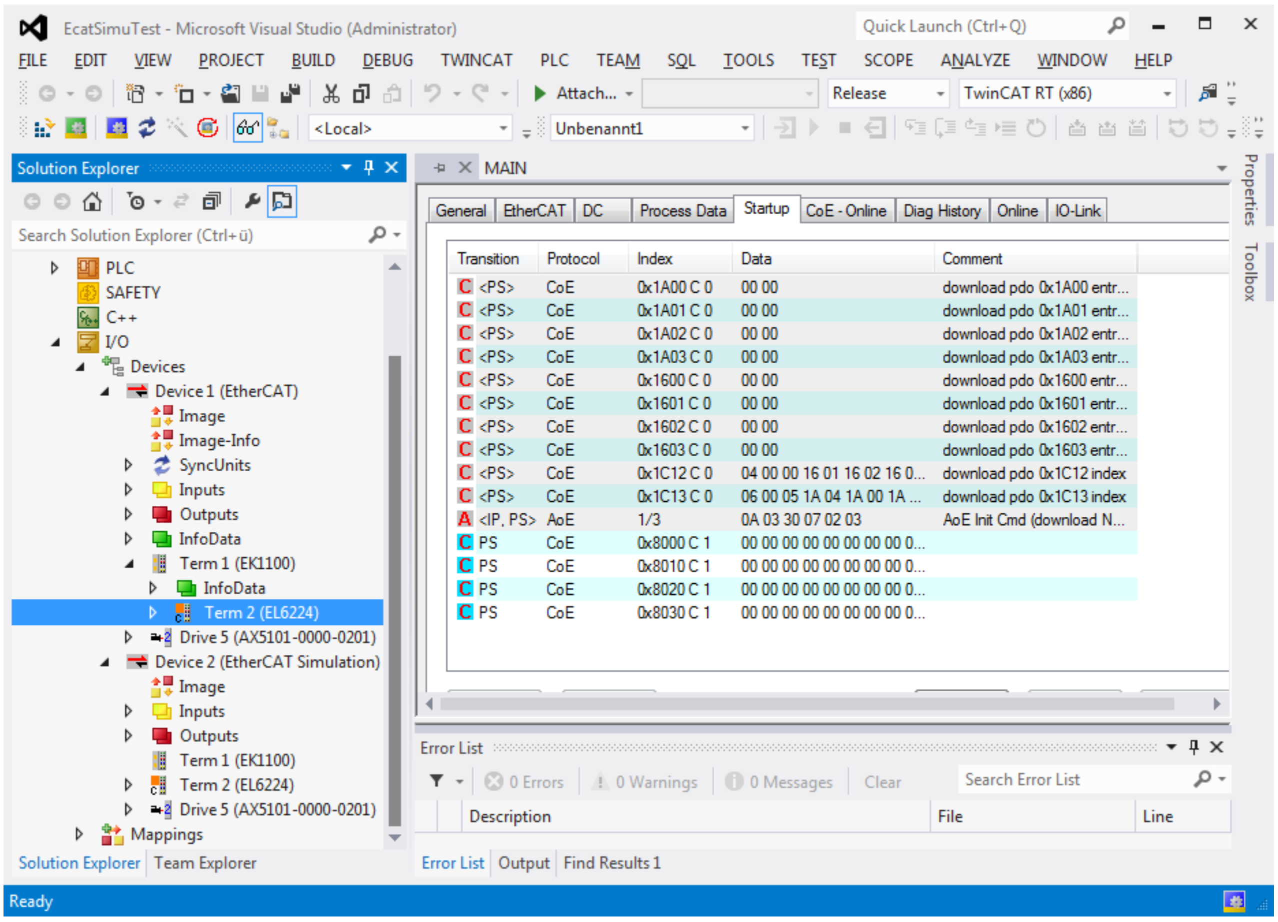
If a machine/plant is to be put into operation step by step or only parts of the machine/plant are to be simulated, cabling can be carried out as shown in the following figure. For this it is necessary that the simulation IPC has 2 free network interfaces. The first network interface is used as an EtherCAT simulation device and is connected with the EtherCAT cable to the control IPC. The second network interface is used as EtherCAT master and connected to the real existing bus devices (see also Mixed operation of real and virtual slaves [▶ 23]).



4.2 Mailbox Auto Response

For some terminals there are startup commands on the master side, which have to be sent to the terminal during each EtherCAT startup (typically from PreOP after SaveOP).

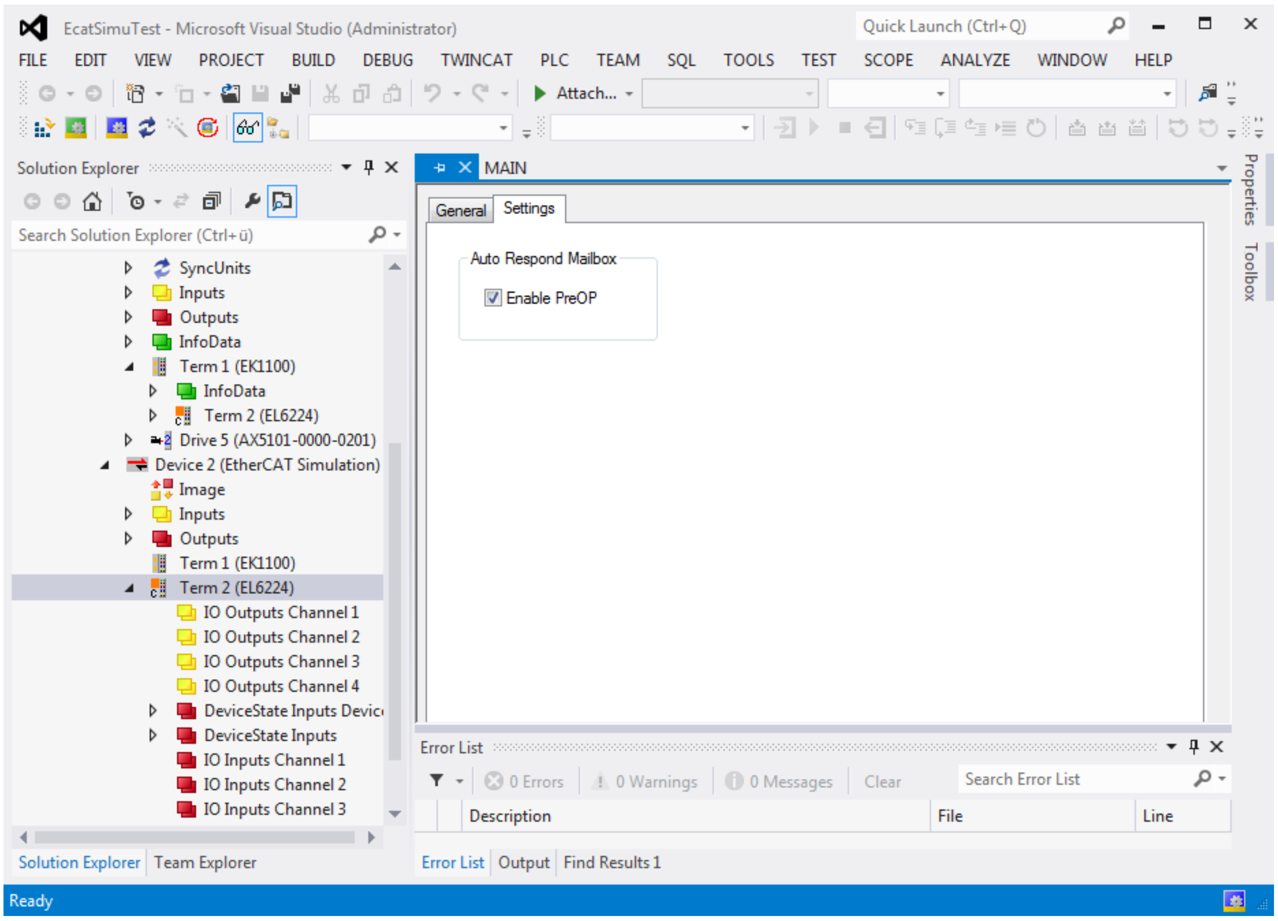
Example EL6224 (IO-Link):



Transition	Protocol	Index	Data	Comment
C <PS>	CoE	0x1A00 C 0	00 00	download pdo 0x1A00 entr...
C <PS>	CoE	0x1A01 C 0	00 00	download pdo 0x1A01 entr...
C <PS>	CoE	0x1A02 C 0	00 00	download pdo 0x1A02 entr...
C <PS>	CoE	0x1A03 C 0	00 00	download pdo 0x1A03 entr...
C <PS>	CoE	0x1600 C 0	00 00	download pdo 0x1600 entr...
C <PS>	CoE	0x1601 C 0	00 00	download pdo 0x1601 entr...
C <PS>	CoE	0x1602 C 0	00 00	download pdo 0x1602 entr...
C <PS>	CoE	0x1603 C 0	00 00	download pdo 0x1603 entr...
C <PS>	CoE	0x1C12 C 0	04 00 00 16 01 16 02 16 0...	download pdo 0x1C12 index
C <PS>	CoE	0x1C13 C 0	06 00 05 1A 04 1A 00 1A ...	download pdo 0x1C13 index
A <IP, PS>	AoE	1/3	0A 03 30 07 02 03	AoE Init Cmd (download N...
C PS	CoE	0x8000 C 1	00 00 00 00 00 00 00 00 0...	
C PS	CoE	0x8010 C 1	00 00 00 00 00 00 00 00 0...	
C PS	CoE	0x8020 C 1	00 00 00 00 00 00 00 00 0...	
C PS	CoE	0x8030 C 1	00 00 00 00 00 00 00 00 0...	

These startup commands must be answered by the EtherCAT simulator, to ensure that the EtherCAT master starts up correctly. The EtherCAT simulation device behaves accordingly, i.e. startup parameters are answered automatically between PreOp and SafeOp, without the need for intervention in the application.

In cases where these parameter are required anyhow, the option “Auto Respond Mailbox” can be disabled (see figure below). If this option is disabled, the queries have to be processed in the application program as required, to ensure that the master starts up correctly.



4.3 CoE / SoE / AoE

Using the EtherCAT simulation it is possible to route the protocols CoE (Can over EtherCAT), SoE (Sercos over EtherCAT) and AoE (ADS over EtherCAT) further (e.g. to the PLC). In this way data can be read or written from the PLC via these protocols.

Routing of CoE and SoE to a PLC

The following two steps are required:

1. The EtherCAT simulation device must be notified of the NetId and the port to which the data are to be relayed. This is done via an ADS-Write to the NetID of the Simulation Device, PortNr.: 0xFFFF, IndexGroup: 0xFF01, IndexOffset:1

Example: Registering the mailbox recipient

```
//AMSNetID of Simulation Device
netIdSimu := '5.35.2.224.2.1';

//AMSNetID of PLC Project on Sim. Dev.
arrNetId[0]:= 5;
arrNetId[1]:= 35;
arrNetId[2]:= 2;
arrNetId[3]:= 224;
arrNetId[4]:= 1; //system ID 1.1
arrNetId[5]:= 1;
arrNetId[6]:= 16#53; //Port of PLC runtime (1)
arrNetId[7]:= 16#03; //Port of PLC runtime (2)  (Port 851 = 16#353)

// sent registration to Simulation driver
fbAdsWrite(WRITE:= FALSE);
fbAdsWrite(NETID:= netIdSimu, PORT:= 16#FFFF, IDXGRP:= 16#FF01, IDXOFFS:= 1, SRCADDR:= ADR(arrNetId)
, LEN:= 8, WRITE:= TRUE);
```

2. Receiving and responding to the ADS indications. The mailbox service is stored in the IndexGroup, the PortID identifies the sending or receiving device, and the data to be accessed is identified through index and subindex.

Example: Intercepting and answering indications:

```
//capture all write requests
fbAdsWriteInd();
IF fbAdsWriteInd.VALID = TRUE THEN //true if a write command has been sent
    IF fbAdsWriteInd.IDXGRP = 16#8000F302 THEN //F302 = CoE request, F420 = SoE request
        //fill in your custom CoE Object Dictionary
        // example reaction
        IF fbAdsWriteInd.IDXOFFS = 16#80000000 THEN (*error*)
            fbAdsWriteRes(
                          NETID:= fbAdsWriteInd.NETID,
                          PORT:= fbAdsWriteInd.PORT,
                          INVOKEID:= fbAdsWriteInd.INVOKEID,
                          RESULT:=  0,
                          RESPOND:= TRUE);
            fbAdsWriteRes(RESPOND:= FALSE);
        ELSE
            fbAdsWriteRes(
                          NETID:= fbAdsWriteInd.NETID,
                          PORT:= fbAdsWriteInd.PORT,
                          INVOKEID:= fbAdsWriteInd.INVOKEID,
                          RESULT:= 0,
                          RESPOND:= TRUE);
            fbAdsWriteRes(RESPOND:= FALSE);
        END_IF
    END_IF
    fbAdsWriteInd(CLEAR:= TRUE);
    fbAdsWriteInd(CLEAR:= FALSE);
END_IF

//capture all read requests
fbAdsReadInd();
IF fbAdsReadInd.VALID = TRUE THEN
    IF fbAdsReadInd.IDXGRP = 16#8000F302 THEN (*CoE*)
        // fill in your CoE Object Dictionary
        // example reaction
        IF fbAdsReadInd.IDXOFFS = 16#80000000 THEN (*error*)
            fbAdsReadRes(
                          NETID:= fbAdsReadInd.NETID,
                          PORT:= fbAdsReadInd.PORT,
                          INVOKEID:= fbAdsReadInd.INVOKEID,
                          LEN:= 0,
                          RESULT:= 16#34567890,
                          RESPOND:= TRUE);
        ELSE
            IF fbAdsReadInd.IDXOFFS = 16#60000000 THEN (*short response*)
                fbAdsReadRes(
                          NETID:= fbAdsReadInd.NETID,
                          PORT:= fbAdsReadInd.PORT,
                          INVOKEID:= fbAdsReadInd.INVOKEID,
                          DATAADDR:= ADR(arrNetId),
                          LEN:= 2,
                          RESULT:= 0,
                          RESPOND:=  TRUE);
            ELSE
                fbAdsReadRes(
                          NETID:= fbAdsReadInd.NETID,
                          PORT:= fbAdsReadInd.PORT,
                          INVOKEID:= fbAdsReadInd.INVOKEID,
                          DATAADDR:= ADR(arrNetId),
                          LEN:= 6,
                          RESULT:= 0,
                          RESPOND:= TRUE);
            END_IF
        END_IF
        fbAdsReadRes(RESPOND:= FALSE);
    END_IF
    fbAdsReadInd(CLEAR:= TRUE);
    fbAdsReadInd(CLEAR:= FALSE);
END_IF
```

As shown in the sample, a check takes place to ascertain whether the indication is valid and if yes, whether it was sent by the simulation device. If both conditions are met, it is a relayed mailbox message that requires a corresponding response. The IndexGroup indicates the mailbox protocol (16#8000F302 for CoE, 16#8000F420 for SoE).

The sample described above can be downloaded here: https://infosys.beckhoff.com/content/1033/TE1111_EtherCAT_Simulation/Resources/3268846987/.zip

Sending CoE Emergency Error Messages

The sending of CoE Emergency error messages is carried out analogously as shown above via ADS Write to the **NetId of** the EtherCAT simulation device, **PortNr.**: Port number of the terminal that is to send the emergency, **IndexGroup**: 0x00FF41, **IndexOffset**:0.

Sending SoE Notifications

The sending of SoE Notifications error messages takes place analogously as shown above via ADS Write to the **NetId of** the EtherCAT simulation device, **PortNr.**: Port number of the terminal that is to send the notification, **IndexGroup**: 0x00FF42, **IndexOffset**: results from the type definition TETHERCAT_SOE_ADS_IOFFS:

```
typedef struct TETHERCAT_SOE_ADS_IOFFS
{
       USHORT IDN;
       union
       {
              struct
              {
                     BYTE          DataState     : 1;
                     BYTE          Name          : 1;
                     BYTE          Attribute     : 1;
                     BYTE          Unit          : 1;
                     BYTE          Min           : 1;
                     BYTE          Max           : 1;
                     BYTE          Value         : 1;
                     BYTE          Default       : 1;
              };
              BYTE   Elements;
       };
       BYTE          DriveNo       : 3;
       BYTE          Reserved2     : 4;
       BYTE          Command       : 1;
} ETHERCAT_SOE_ADS_IOFFS
```

Example: 0x00401234 -> For the SoE IDN 1234 the value flag is addressed here.

Routing of AoE to a PLC

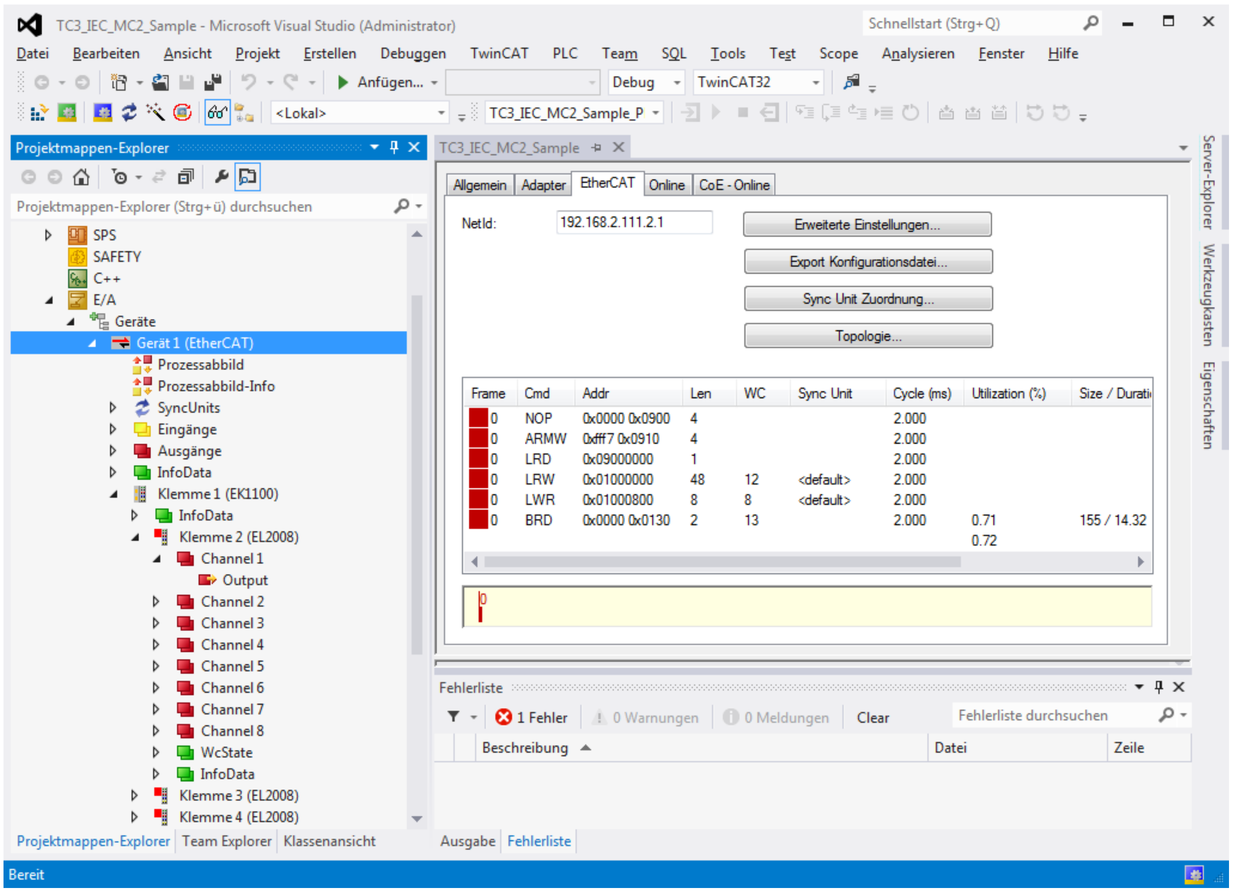
Registration of the mailbox recipient is identical to the description under CoE and SoE. The handling of AoE in the PLC also takes place in a similar structure, although several queries are involved, due to the functionality of AoE. A sample is shown below, which can be downloaded here: https://infosys.beckhoff.com/content/1033/TE1111_EtherCAT_Simulation/Resources/3268845323/.zip

```
fbAdsReadWriteInd();
IF fbAdsReadWriteInd.VALID = TRUE THEN
    MEMCPY(ADR(arrTmp), fbAdsReadWriteInd.DATAADDR, fbAdsReadWriteInd.WRTLENGTH);
    //copy of data in temp variable
    //evaluate destination of AeE command (e.g. terminal)-first 6 byte in arrTmp
    FOR Idx := 0 TO 5 DO
        arrDest[Idx] := arrTmp[Idx];
    END_FOR
    IF smyDestId = F_CreateAmsNetId(arrDest) THEN
        //evaluate port (different physical ports or different services of terminal may be possible)
        IF fbAdsReadWriteInd.PORT = 16#1000 THEN //Io-Link Port 1
            // evaluate command type - byte 16 : write = 3, read = 2
            IF arrTmp[16] = 3 THEN // write command
                //evaluate index group and index offset as shown in CoE and SoE
                //index group: evaluate service  (e.g. object register of Io-Link Terminal)
                IF fbAdsReadWriteInd.IDXGRP = 16#8000F302 THEN
                    IF fbAdsReadWriteInd.IDXOFFS = 16#80000000 THEN //adressing of service
                        MEMCPY(ADR(arrTmp), fbAdsReadWriteInd.DATAADDR + 8, 8);
                        //change destination net ID and source net ID
                        MEMCPY(ADR(arrTmp) + 8, fbAdsReadWriteInd.DATAADDR, 8);
                        arrTmp[18].0:= 1; (*set Respond Bit*)
                        arrTmp[20]:= 4; (*Length*)
                        arrTmp[21]:= 0;
                        arrTmp[22]:= 0;
                        arrTmp[23]:= 0;
                        arrTmp[32]:= 0;  (*Response Code*)
                        arrTmp[33]:= 0;
                        arrTmp[34]:= 0;
                        arrTmp[35]:= 0;
                        fbAdsReadWriteRes(
                                          NETID:= fbAdsReadWriteInd.NETID,
                                          PORT:= fbAdsReadWriteInd.PORT,
                                          INVOKEID:= fbAdsReadWriteInd.INVOKEID,
                                          DATAADDR:= ADR(arrTmp),
                                          LEN:= 36, RESPOND:= TRUE);
                    END_IF
                END_IF
            ELSIF arrTmp[16] = 2 THEN //read command
                IF fbAdsReadWriteInd.IDXGRP = 16#8000F302 THEN
                    IF fbAdsReadWriteInd.IDXOFFS = 16#80000000 THEN //adressing of service
                        MEMCPY(ADR(arrTmp), fbAdsReadWriteInd.DATAADDR + 8, 8);
                        //change destination net ID and source net ID
                        MEMCPY(ADR(arrTmp) + 8, fbAdsReadWriteInd.DATAADDR, 8);
                        arrTmp[18].0:= 1; (* set Respond Bit*)
                        arrTmp[20]:= 14; (*Len*)
                        arrTmp[21]:= 0;
                        arrTmp[22]:= 0;
                        arrTmp[23]:= 0;
                        arrTmp[32]:= 0; (*Response Code*)
                        arrTmp[33]:= 0;
                        arrTmp[34]:= 0;
                        arrTmp[35]:= 0;
                        arrTmp[36]:= 6; (*Len*)
                        arrTmp[37]:= 0;
                        arrTmp[38]:= 0;
                        arrTmp[39]:= 0;
                        arrTmp[40]:= arrNetId[0]; (*Daten*)
                        arrTmp[41]:= arrNetId[1];
                        arrTmp[42]:= arrNetId[2];
                        arrTmp[43]:= arrNetId[3];
                        arrTmp[44]:= arrNetId[4];
                        arrTmp[45]:= arrNetId[5];
                        fbAdsReadWriteRes(
                                          NETID:= fbAdsReadWriteInd.NETID,
                                          PORT:= fbAdsReadWriteInd.PORT,
                                          INVOKEID:= fbAdsReadWriteInd.INVOKEID,
                                          DATAADDR:= ADR(arrTmp),
                                          LEN:= 46, RESPOND:= TRUE);
                    END_IF;
                END_IF;
            END_IF
        END_IF
    END_IF;
    fbAdsReadWriteRes(
                     NETID:= fbAdsReadWriteInd.NETID,
                     PORT:= fbAdsReadWriteInd.PORT,
                     INVOKEID:= fbAdsReadWriteInd.INVOKEID,
                     DATAADDR:= ADR(arrTmp),
                     LEN:= fbAdsReadWriteInd.WRTLENGTH,
                     RESPOND:= TRUE);
    fbAdsReadWriteRes(RESPOND:= FALSE);
    fbAdsReadWriteInd(CLEAR:= TRUE);
    fbAdsReadWriteInd(CLEAR:= FALSE);
END_IF
```

5 Quickstart

Creation of a simulation project with the hardware that is to be used in the control project.

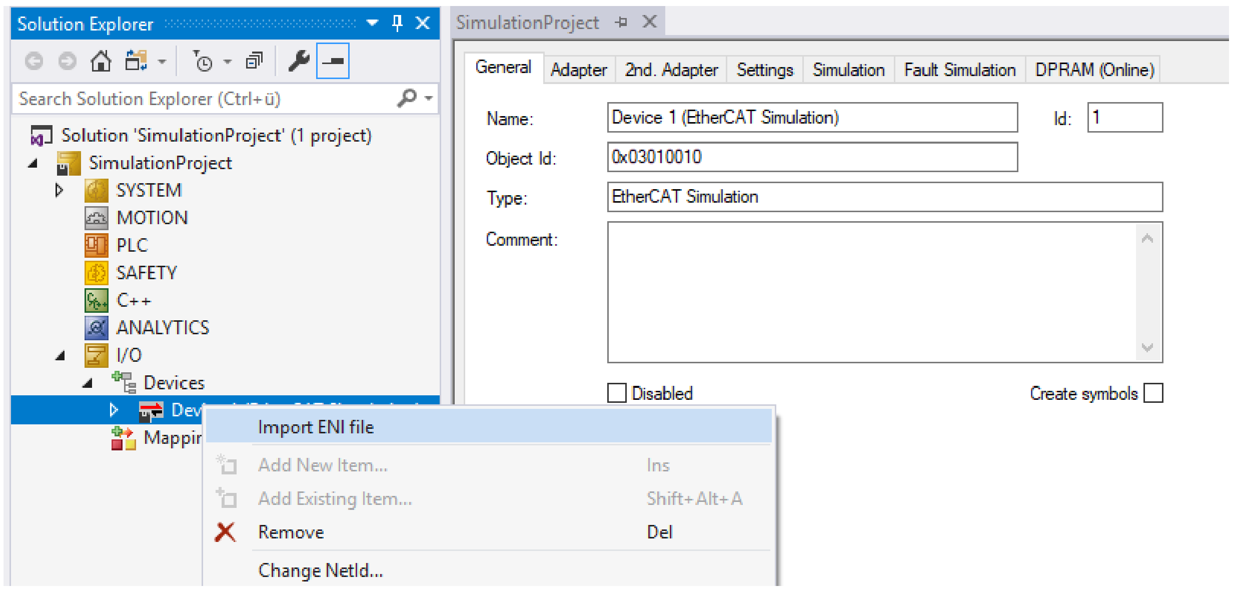
1. To export the hardware configuration of the control project click in the Solution Explorer on the EtherCAT Master Device, then on the tab **EtherCAT** and finally on the button **Export Configuration File...** .



NetId: 192.168.2.111.2.1

Frame	Cmd	Addr	Len	WC	Sync Unit	Cycle (ms)	Utilization (%)	Size / Durati
🟥 0	NOP	0x0000 0x0900	4			2.000		
🟥 0	ARMW	0xfff7 0x0910	4			2.000		
🟥 0	LRD	0x09000000	1			2.000		
🟥 0	LRW	0x01000000	48	12	<default>	2.000		
🟥 0	LWR	0x01000800	8	8	<default>	2.000		
🟥 0	BRD	0x0000 0x0130	2	13		2.000	0.71	155 / 14.32
							0.72	

2. Then create the simulation project on the simulation IPC
3. click on **Add New Item** in order to create an EtherCAT simulation device.
4. Open the context menu of the EtherCAT simulation device and click on **Import ENI file** in order to import the EtherCAT configuration of the EtherCAT master.


Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'SimulationProject' (1 project)
SimulationProject
SYSTEM
MOTION
PLC
SAFETY
C++
ANALYTICS
I/O
Devices
Dev
Mappir
Import ENI file
Add New Item... Ins
Add Existing Item... Shift+Alt+A
Remove Del
Change NetId...
SimulationProject
General Adapter 2nd. Adapter Settings Simulation Fault Simulation DPRAM (Online)

Name: Device 1 (EtherCAT Simulation) Id: 1
Object Id: 0x03010010
Type: EtherCAT Simulation
Comment:
☐ Disabled Create symbols ☐

⇨ The simulation project is created and filled with the hardware configuration of the control project.

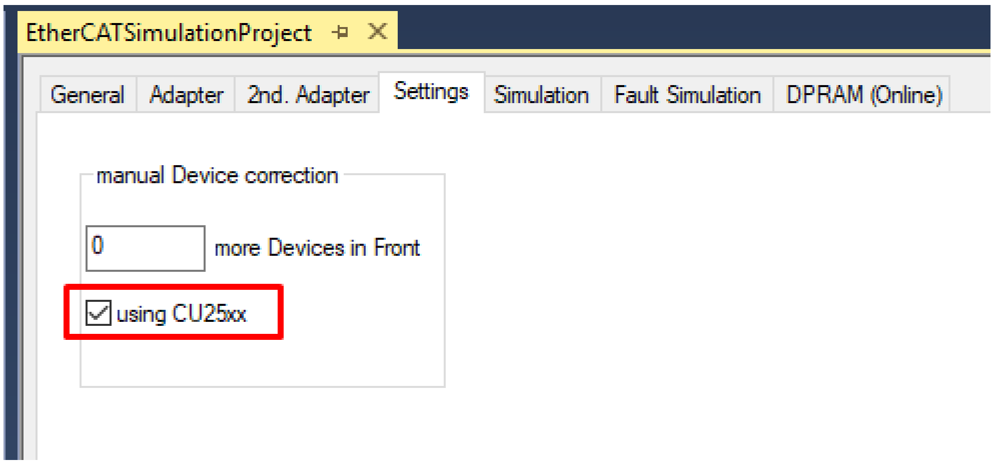
What else needs to be done?

- Creation or instancing of the “simulation” TwinCAT modules
- Creation of a task that drives the simulation modules (with half the cycle time on account of the sampling theorem)
- Linking of the variables of the process image of the simulation models with those of the mirrored process image of the EtherCAT simulation device

6 Use of CU2508 port multipliers

TwinCAT versions < TwinCAT 3.1.4024

The CU2508 may only be used on the EtherCAT Master side (see also Cabling [▶ 11]). If this is the case, the option **Using CU2508** must be activated in the Settings tab.



manual Device correction

0 more Devices in Front

☑ using CU25xx

TwinCAT versions > TwinCAT 3.1.4024

The CU2508 may only be used on the EtherCAT Master side (see also Cabling [▶ 11]). The EtherCAT simulation device automatically detects that a CU2508 was used on the master side. The option “using CU2508” is therefore no longer necessary

NOTE

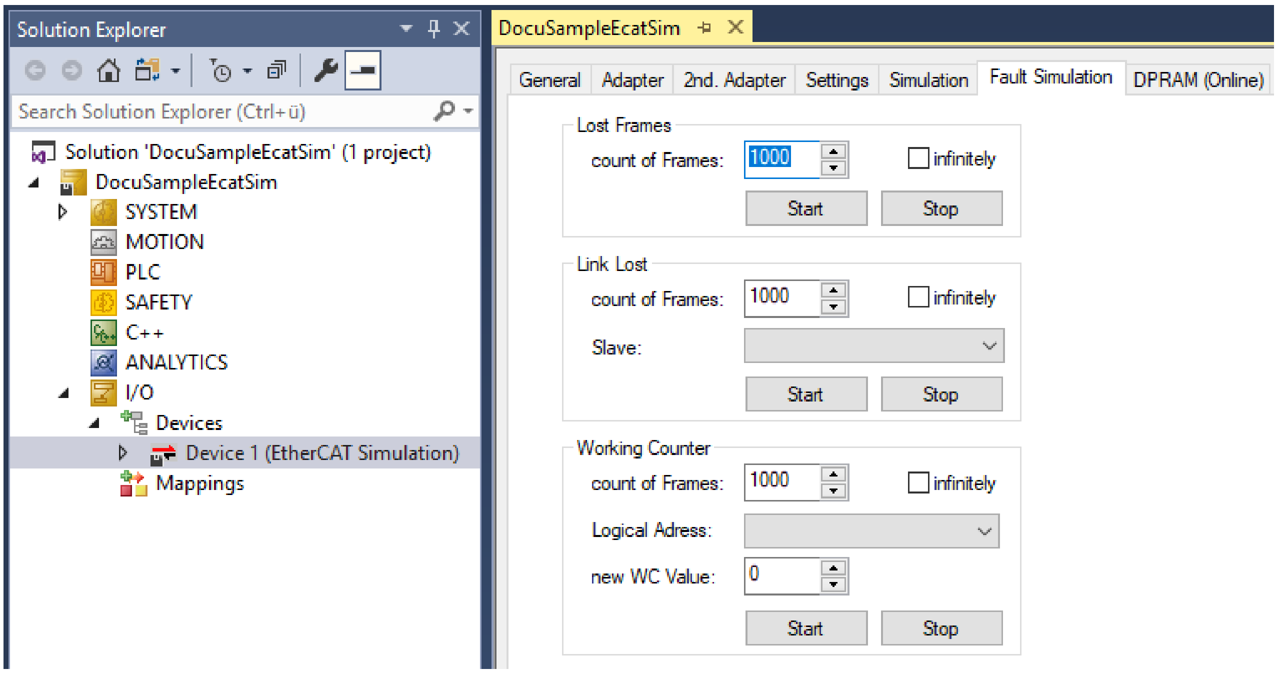
Incorrect values in the EtherCAT diagnosis

If a CU2508 is used on the slave side, it is not contained in the TwinCAT project that exports the configuration file. In addition, this CU2508 only receives the EtherCAT frame after the EtherCAT simulation device. It is thus “invisible” to the EtherCAT simulation device. Consequently, neither the Device State nor the Slave Count on the EtherCAT master side display correct values. An EtherCAT diagnosis programmed in the original project thus displays incorrect results.

7 Fault simulation

The following errors can be simulated using the EtherCAT simulation device:

- Lost Frames
- Link Lost
- Working Counter Error



Lost Frames
count of Frames: 1000 ☐ infinitely

Link Lost
count of Frames: 1000 ☐ infinitely
Slave:

Working Counter
count of Frames: 1000 ☐ infinitely
Logical Adress:
new WC Value: 0

Simulation of lost frames:

1. Change to the Fault Simulation tab on the EtherCAT Simulation Device
2. In the **count of Frames** input field, set the number of frames to be lost (if this is not limited, check **infinitely** instead).
3. Press the **Start** button
   ⇨ The simulation of lost frames starts
4. To stop the simulation press the **Stop** button

Simulation of Link Losts:

5. Change to the Fault Simulation tab on the EtherCAT Simulation Device
6. In the **count of Frames** input field, set the number of frames for which a link-lost error should be reported (if this is not limited, check **infinitely** instead).
7. **In the Slave** selection box, set the EtherCAT device for which the Lost Error link is to be reported
8. Press the **Start** button
   ⇨ The simulation of the link-lost error starts
9. To stop the simulation press the **Stop** button

Simulation of working counter errors:

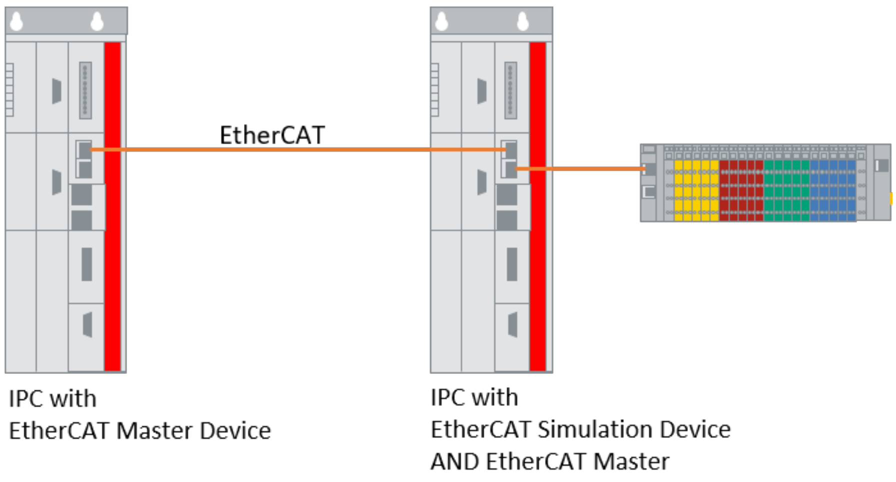
10. Change to the Fault Simulation tab on the EtherCAT Simulation Device
11. In the **count of Frames** input field, set the number of frames for which a working counter error should be reported (if this is not limited, check **infinitely** instead).
12. **In the Logical Address** selection box, select the logical address that is to add an incorrect value to the working counter.
13. In the **WC Value** field, enter the new (absolute) value to be assigned to the working counter.

14. Press the **Start** button
    ⇨ The working counter fault simulation starts
15. To stop the simulation press the **Stop** button

See also the chapter Fault Simulation tab [▶ 25]

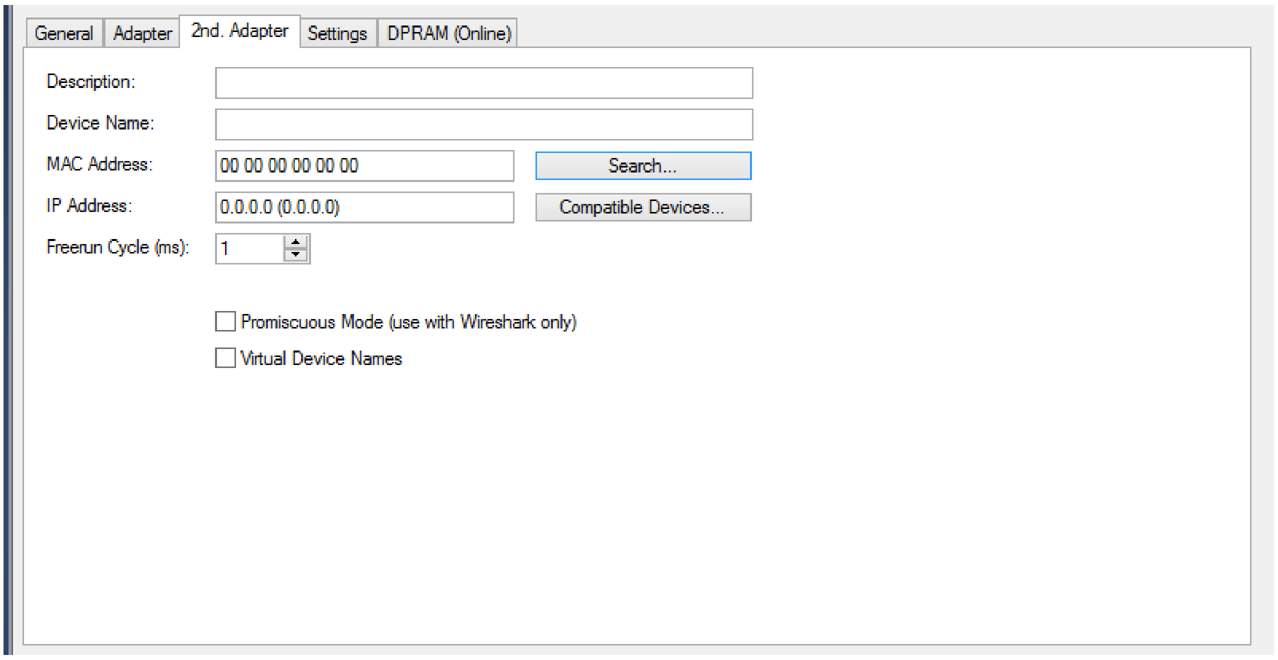
8 Mixed operation of real and virtual slaves

The EtherCAT simulation enables mixing of real and virtual slaves. This can be used, for example, to gradually put a machine/plant into operation. The following figure illustrates this schematically.



For mixed operation, 2 free network interfaces at the simulation IPC are required. The first interface serves as an EtherCAT simulation device and is connected to the control computer via the EtherCAT cable. This is set in the Adapter tab of the EtherCAT simulation device (see also Adapter).

For the connection with the real existing slaves the second adapter on the EtherCAT simulation device is used (tab 2nd Adapter). This is selected in the same way as the first adapter.



Description:	
Device Name:	
MAC Address:	00 00 00 00 00 00
IP Address:	0.0.0.0 (0.0.0.0)
Freerun Cycle (ms):	1

☐ Promiscuous Mode (use with Wireshark only)

☐ Virtual Device Names

The slaves that actually exist must be deactivated in the imported project of the EtherCAT simulation device and in exactly the same order on the EtherCAT master of the 2nd. adapter must be connected. The figure above is intended to illustrate this.

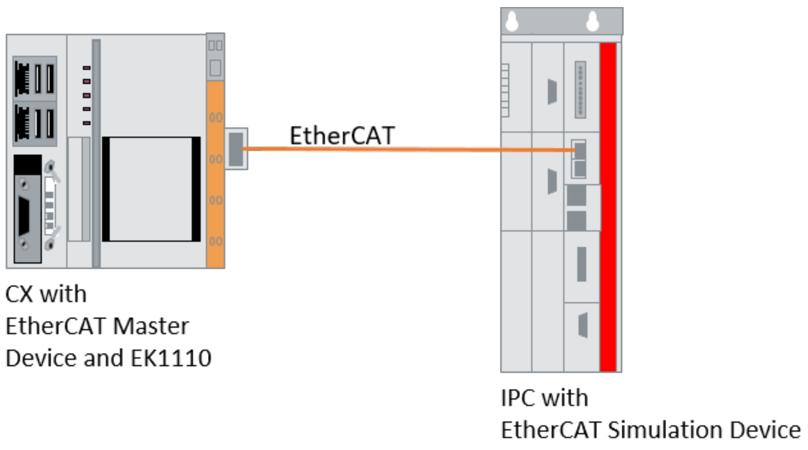
9 Reference, user interface

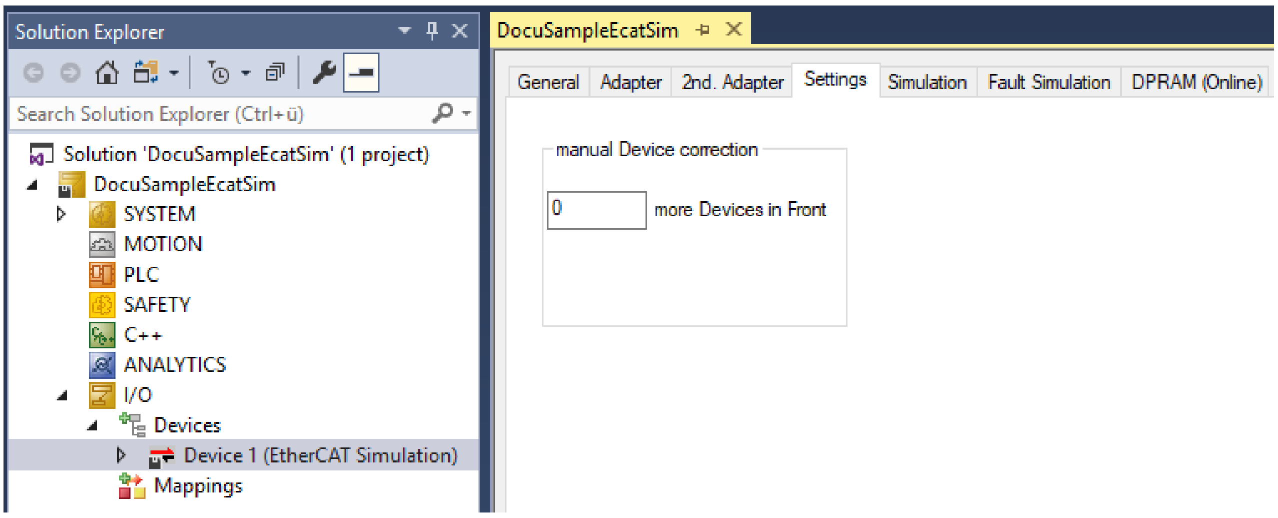
The following chapter explains the user interface of the EtherCAT Simulation Device. The General and Adapter/ 2nd Adapter tabs are the same as for a standard EtherCAT master. For this reason, we would like to refer to this at this point (see Adapter).

The 2nd adapter of the EtherCAT Simulation Device is used to configure a second used interface if the EtherCAT simulation is to be operated in mixed operation of real and virtual slaves (see Mixed operation of real and virtual slaves [▶ 23]).

9.1 Settings tab

The Settings tab contains the manual Device correction setting. This can be used if EtherCAT devices are connected before the actual EtherCAT simulation. These could be, for example, infrastructure devices (e.g. the CU2508) or intelligent power supply units (with CX).





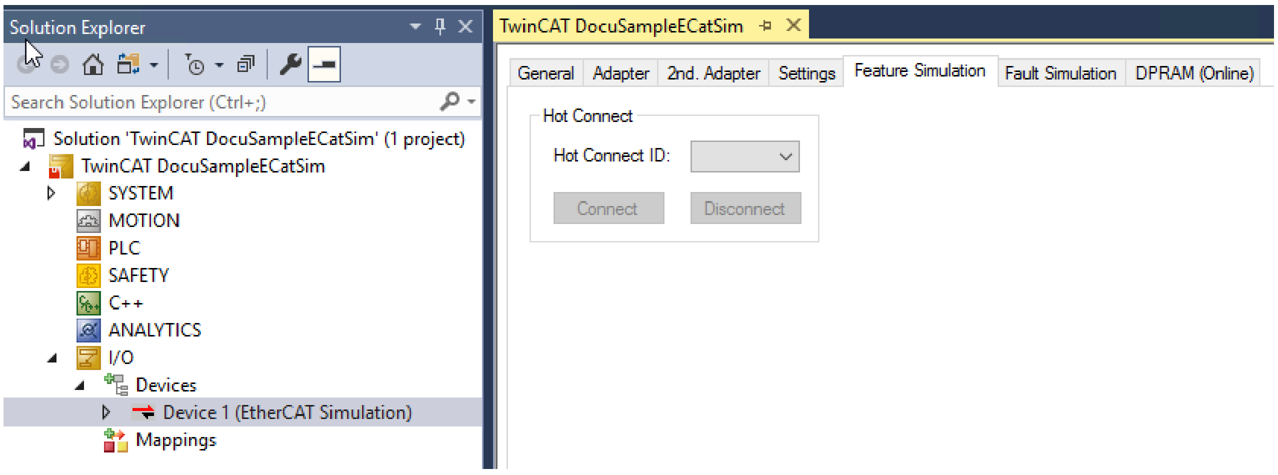
manual Device correction

0 more Devices in Front

From the TwinCAT version TwinCAT 3.1.4024.0 the CU2508 is automatically detected. For this device NO manual device correction is necessary from this version on.

9.2 Feature Simulation tab

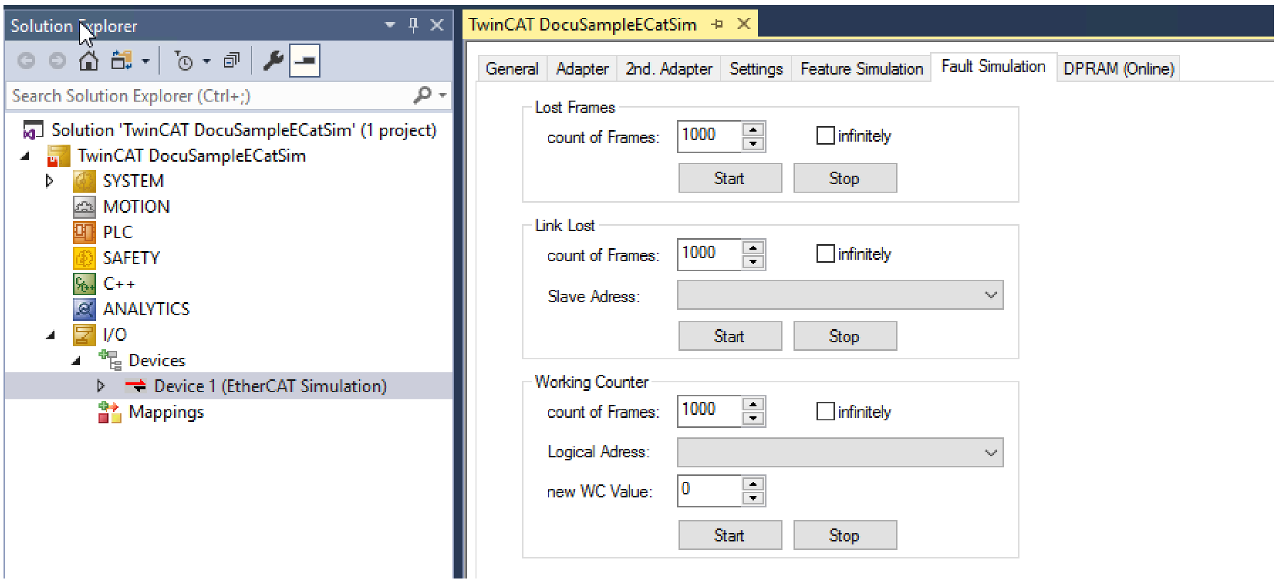
In the **Simulation** tab, HotConnect groups can be simulated and logged on or off. The group to be logged on or off can be selected from the dropbox. The selected HotConnect group is then logged on or off using the **Connect** or **Disconnect** buttons.



9.3 Fault Simulation tab

The Fault Simulation tab is used to simulate EtherCAT errors. The options provided here can be used to test the robustness of the set EtherCAT configuration or the diagnosis of the machine/plant. The following errors can be simulated:

- Lost Frames
- Link Lost
- Working Counter Error



See also the chapter Fault simulation [▶ 21].

The settings in the Fault Simulation tab are explained below.

Setting	Meaning
Lost Frames	
Count of Frames	Number of EtherCAT frames to be lost
infinitely	Setting the number of EtherCAT frames to infinity
Start	Start of Lost Frames Simulation
Stop	Stopping the Lost Frames Simulation
Link Lost	
Count of Frames	Number of EtherCAT frames in which a Link Lost should be reported
infinitely	Setting the number of EtherCAT frames to infinity.
Slave	Selection of the EtherCAT slave that should report the Lost Error link
Start	Start of the Link Lost Simulation
Stop	Stopping the Link Lost Simulation
Working Counter	
Count of Frames	Number of EtherCAT frames in which a wrong working counter is to be reported
infinitely	Setting the number of EtherCAT frames to infinity.
Logical Address	Logical address at which the working counter is to be calculated incorrectly
New WC Value	New (absolute) WC value that is to be assigned
Start	Start of the Working Counter Simulation
Stop	Stopping the Working Counter Simulation

10 PLC API

10.1 ADS commands

The options provided in the Feature Simulation tab [▶ 25] or Fault Simulation tab [▶ 25] regarding the simulation of Hot Connect groups or EtherCAT errors can also be called via ADS (e.g. from the PLC).

ADS Write commands to the NetId of the EtherCAT simulation device must be called as follows:

Hot Connect:

Defined index groups:

Disconnect Device: 0x0010

Connect Device: 0x0011

Index offsets to be used:

The high word must be 0.

The low word contains the Identification Value (Hot Connect ID) of the device you want to connect or disconnect.

Simulation of lost frames:

Index-Group: 0x0100

Index offset: 0x0001

Data: 4 byte counter in which it is specified how many frames are to be omitted. 0xFFFFFFFF means infinite

Simulation of link lost errors:

Index-Group: 0x0100

Index offset: 0x0002

Data: 4 byte counter in which it is specified how many frames are to be omitted. 0xFFFFFFFF means infinite

Simulation of working counter errors:

Index-Group: 0x0100

Index offset: 0x0003

Data: a total of 10 bytes:

- 4 byte counter in which it is specified how many frames are to be omitted. 0xFFFFFFFF means infinite
- 4 byte logical address. This addresses the telegram which is to be manipulated
- 2 bytes of new working counter. This value is then entered into the telegram

More Information:
www.beckhoff.com/te1111

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com