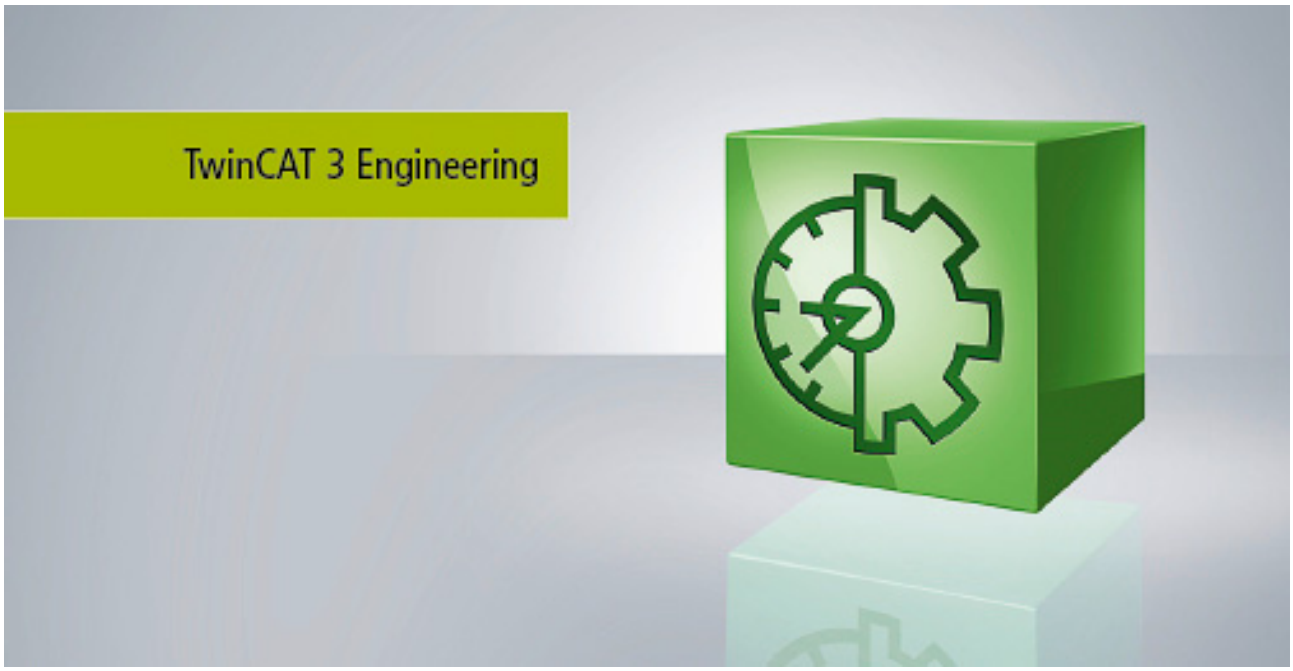


Manual

TC3 EtherCAT Simulation

TwinCAT 3

Version: 1.2
Date: 2018-08-23
Order No.: TE1111

BECKHOFF

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 the 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

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®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® 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, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

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

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.

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 already exist in Matlab Simulink, it is possible together with the TE1400 Target for Matlab®/ Simulink® function to perform HIL (Hardware-In-the-Loop) simulations with no great 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

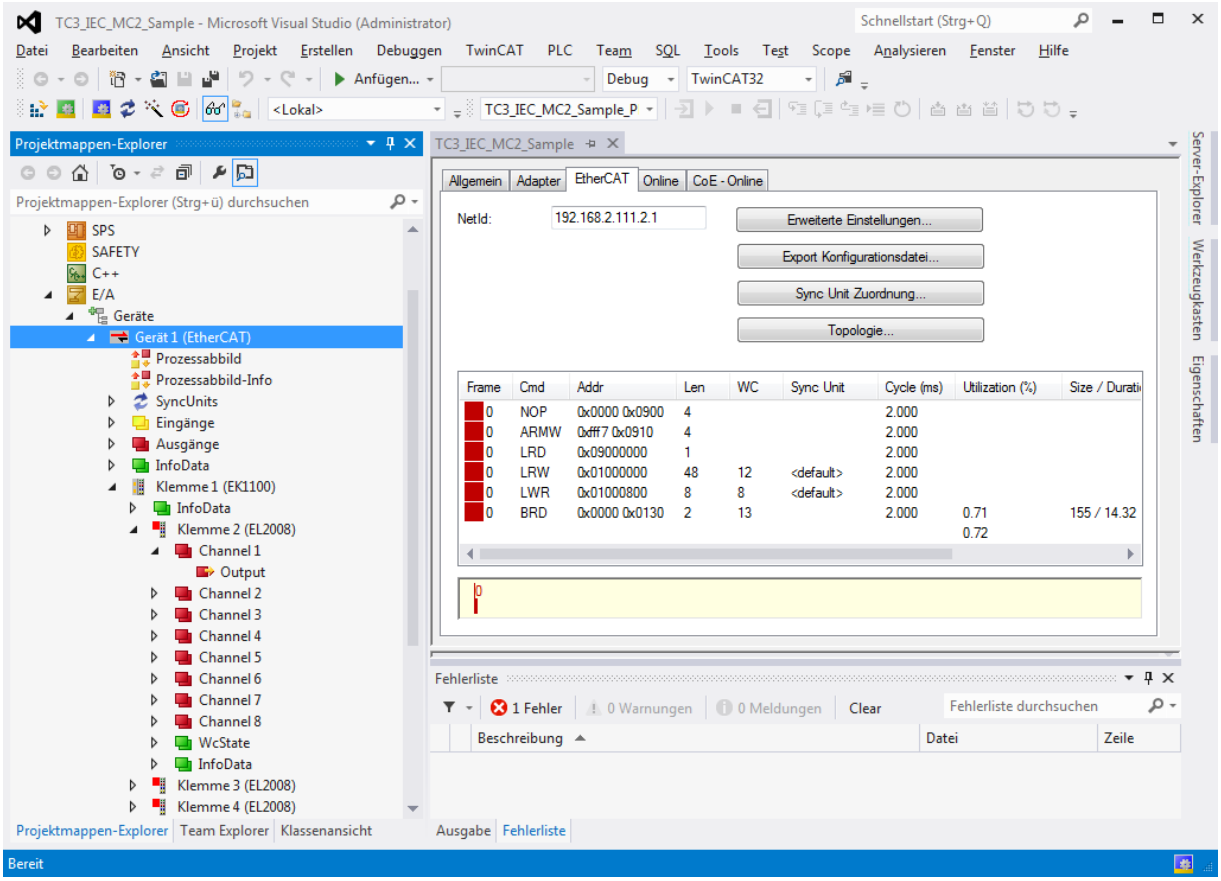
3 Installation

The TE1111 function | TwinCAT EtherCAT simulation is already installed together with the TwinCAT installation and is available as release version since TwinCAT 3.1 build 4018.0. Therefore it only needs to be licensed. See also ‘Ordering and activation of TwinCAT 3 standard licenses’.

4 Quickstart

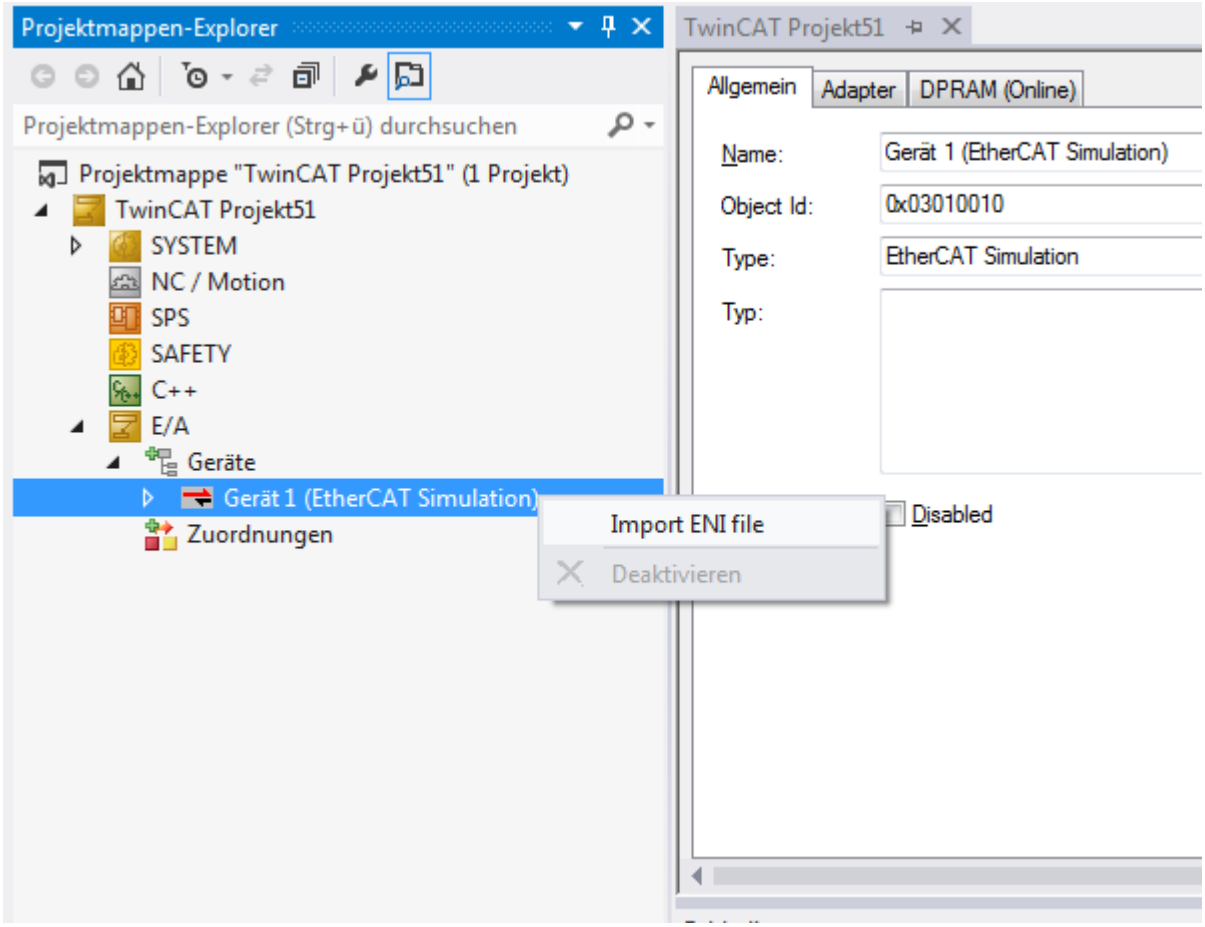
Creation of a simulation project with the hardware that is to be used in the control project.

1. To export the hardware configuration of the control project, click on the EtherCAT master device in the Project Folder Explorer, then on the ‘EtherCAT’ tab and then on the ‘Export configuration file...’ button.



2. Then create the simulation project on the simulation IPC
3. click on ‘Add new element’ 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.



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

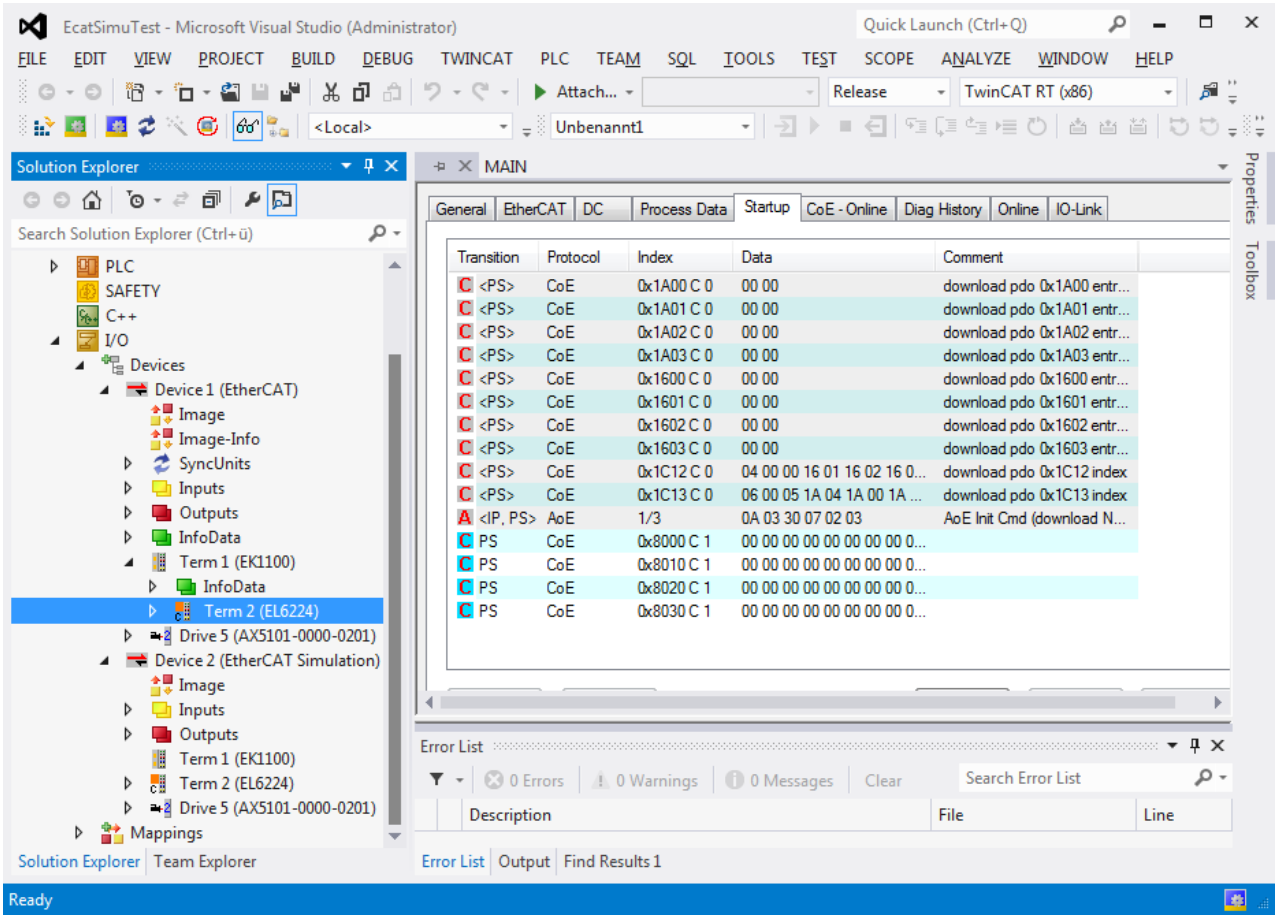
What else needs to be done?

- 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

5 Mailbox Auto Response

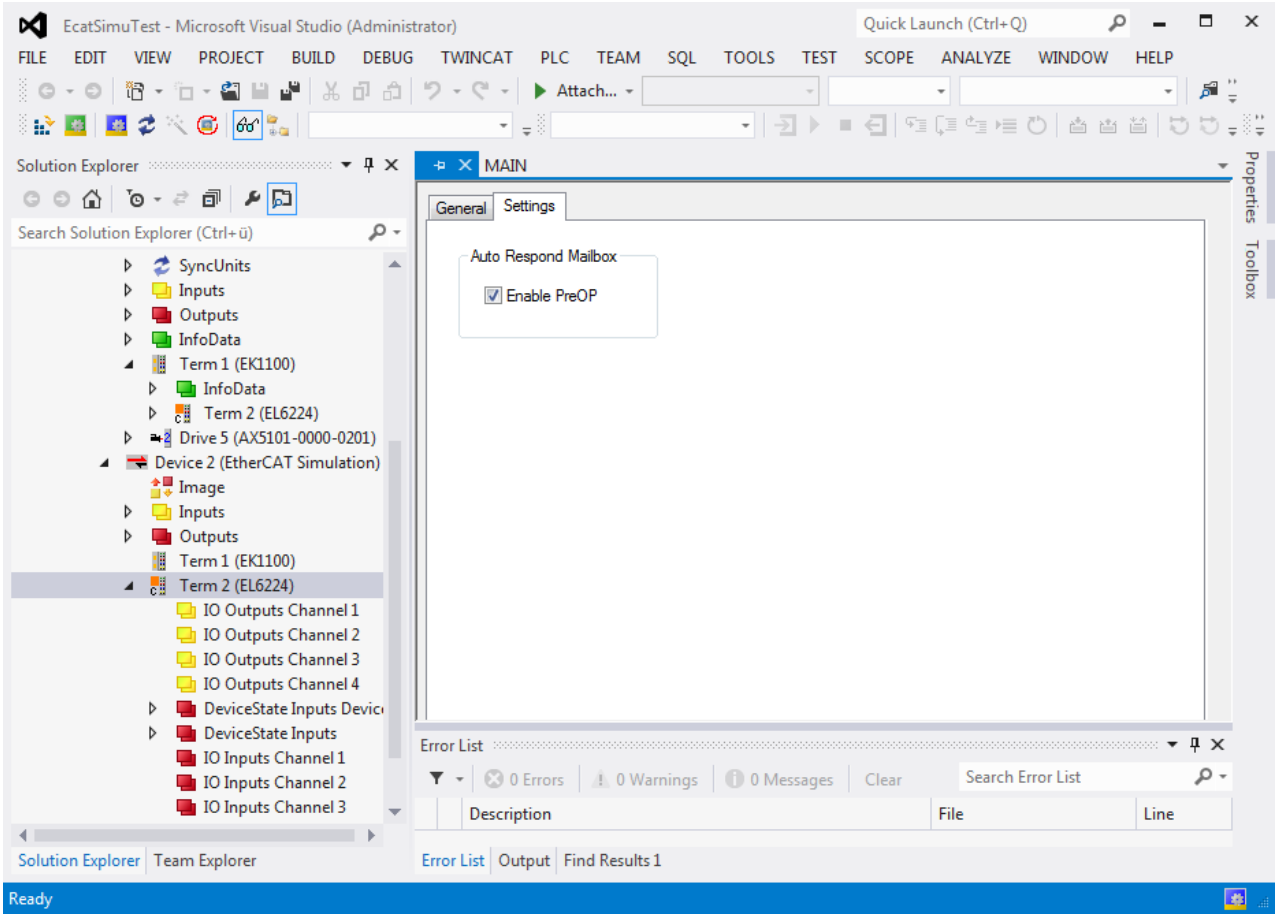
For some terminals there are startup commands on the master side, which have to be sent to the terminal during each EtherCAT startup (typically from PreOP after SaveOP).

Example EL6224 (IO-Link):



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.


EcatSimuTest - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TWINCAT PLC TEAM SQL TOOLS TEST SCOPE ANALYZE WINDOW HELP
Attach...
Solution Explorer
Search Solution Explorer (Ctrl+ü)
SyncUnits
Inputs
Outputs
InfoData
Term 1 (EK1100)
InfoData
Term 2 (EL6224)
Drive 5 (AX5101-0000-0201)
Device 2 (EtherCAT Simulation)
Image
Inputs
Outputs
Term 1 (EK1100)
Term 2 (EL6224)
IO Outputs Channel 1
IO Outputs Channel 2
IO Outputs Channel 3
IO Outputs Channel 4
DeviceState Inputs Devic
DeviceState Inputs
IO Inputs Channel 1
IO Inputs Channel 2
IO Inputs Channel 3
Solution Explorer
Team Explorer
MAIN
General
Settings
Auto Respond Mailbox
Enable PreOP
Properties
Toolbox
Error List
0 Errors
0 Warnings
0 Messages
Clear
Search Error List
Description
File
Line
Error List
Output
Find Results 1
Ready

6 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 command to the NetID of the simulation device, PortNo.: 0xFFFF, IndexGroup: 0xFF01, IndexOffset:1

Sample: 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(arr-
NetId), 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.

Sample: 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 a certain 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/0033/TE1111_EtherCAT_Simulation/Resources/zip/9007202523587979.zip

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/0033/TE1111_EtherCAT_Simulation/Resources/zip/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
```

7 Mixed operation of real and virtual slaves

The EtherCAT simulation enables mixing of real and virtual slaves. In cases where this is desirable, the real slaves should be deactivated in the simulation project and added at the end of the EtherCAT strand in exactly the same order as they are stored in the project.

Mixed operation of real and virtual slaves therefore requires a further network card. This can be set on the EtherCAT simulation device in the 2nd Adapter tab.