# Table of contents

1 Foreword .................................................................................................................................................. 5  
1.1 Notes on the documentation ........................................................................................................... 5  
1.2 Safety instructions .......................................................................................................................... 6  
1.3 Notes on information security .......................................................................................................... 7  
2 Introduction ........................................................................................................................................ 8  
3 EnOcean ............................................................................................................................................. 9  
3.1 Range planning ............................................................................................................................... 9  
3.2 Approval of EnOcean wireless technology ..................................................................................... 10  
4 Programming ....................................................................................................................................... 11  
4.1 POUs .............................................................................................................................................. 12  
4.1.1 KL6021-0023 ........................................................................................................................... 14  
4.1.2 KL6581 .................................................................................................................................. 23  
4.2 DUTs ............................................................................................................................................... 35  
4.2.1 KL6021-0023 ........................................................................................................................... 36  
4.2.2 KL6581 .................................................................................................................................. 38  
4.3 Integration into TwinCAT ................................................................................................................. 44  
4.3.1 KL6581 with CX5120 ............................................................................................................... 44  
5 Appendix ............................................................................................................................................ 48  
5.1 Support and Service ........................................................................................................................ 48
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

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®, TwinCAT/BSD®, TC/BSD®, 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:

EtherCAT® is a 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 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!

<table>
<thead>
<tr>
<th>⚠️ DANGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serious risk of injury!</td>
</tr>
<tr>
<td>Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>⚠️ WARNING</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk of injury!</td>
</tr>
<tr>
<td>Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>⚠️ CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal injuries!</td>
</tr>
<tr>
<td>Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Damage to the environment or devices</td>
</tr>
<tr>
<td>Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

 ромб •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](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](https://www.beckhoff.com/secinfo).
2 Introduction

The Tc2_EnOcean library is a TwinCAT PLC library for data exchange with EnOcean devices. The library should only be used in conjunction with a KL6021-0023 or KL6581.

The user of this library requires basic knowledge of the following:

- TwinCAT XAE
- PC and network knowledge
- Structure and properties of the Beckhoff Embedded PC and its Bus Terminal system
- Technology of EnOcean devices
- Relevant safety regulations for building technical equipment

This software library is intended for building automation system partners of Beckhoff Automation GmbH & Co. KG. The system partners operate in the field of building automation and are concerned with the installation, commissioning, expansion, maintenance and service of measurement, control and regulating systems for the technical equipment of buildings.

The Tc2_EnOcean library is usable on all hardware platforms that support TwinCAT 3.1 or higher.

Hardware documentation for KL6021-0023 and KL6581 in the Beckhoff Information System.
3 EnOcean

The EnOcean radio technology makes a far-reaching signal with low quantities of ambient energy possible. With 50 µWs, a standard EnOcean radio module can easily transmit a signal over a distance of 300 m (in the free field). The signal period for an EnOcean telegram is approx. 1 thousandth of second.

- Licence-free 868 MHz frequency band with 1% duty cycle
- Multiple telegram transmission with checksum
- Short telegrams (approx. 1 ms) lead to a small probability of collision
- Long range: 30 m inside buildings or 300 m in the free field
- Repeater available for extensions
- Unidirectional and bidirectional communication
- High data transmission rates of 125 kbit/s
- Low ‘data overhead’
- ASK modulation
- Radio protocol is defined and integrated in modules
- Sensor profiles specified and adhered to by users
- Unique transmission ID (32-bit)
- No interference with DECT, WLAN, PMR systems etc.
- System design verified in industrial environment

Protocol structure

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Description</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>Telegram type</td>
<td>1 byte</td>
</tr>
<tr>
<td>DB_3</td>
<td>Data byte 3</td>
<td>1 byte</td>
</tr>
<tr>
<td>DB_2</td>
<td>Data byte 2</td>
<td>1 byte</td>
</tr>
<tr>
<td>DB_1</td>
<td>Data byte 1</td>
<td>1 byte</td>
</tr>
<tr>
<td>DB_0</td>
<td>Data byte 0</td>
<td>1 byte</td>
</tr>
<tr>
<td>ID_3</td>
<td>Transmitter ID byte 3</td>
<td>1 byte</td>
</tr>
<tr>
<td>ID_2</td>
<td>Transmitter ID byte 2</td>
<td>1 byte</td>
</tr>
<tr>
<td>ID_1</td>
<td>Transmitter ID byte 1</td>
<td>1 byte</td>
</tr>
<tr>
<td>ID_0</td>
<td>Transmitter ID byte 0</td>
<td>1 byte</td>
</tr>
<tr>
<td>STATUS</td>
<td>Information status</td>
<td>1 byte</td>
</tr>
</tbody>
</table>

3.1 Range planning

Please follow the recommendations of the EnOcean Alliance (see www.enocean.org) when placing the EnOcean devices. Adherence to the recommendations is conducive to an optimum range and high noise immunity.

Attenuation of different materials

<table>
<thead>
<tr>
<th>Material</th>
<th>Attenuation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wood, plaster, uncoated glass (without metal)</td>
<td>0...10 %</td>
</tr>
<tr>
<td>Brick, chipboard</td>
<td>5...35 %</td>
</tr>
<tr>
<td>Concrete with iron reinforcement</td>
<td>10...90 %</td>
</tr>
<tr>
<td>Metal, aluminum cladding</td>
<td>90..100 %</td>
</tr>
</tbody>
</table>
Range

<table>
<thead>
<tr>
<th>Material</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line of sight</td>
<td>Typically 30 m in corridors, up to 100 m in halls</td>
</tr>
<tr>
<td>Plasterboard walls/wood</td>
<td>Typically 30 m, through max. 5 walls</td>
</tr>
<tr>
<td>Brick walls/aerated concrete</td>
<td>Typically 20 m, through max. 3 walls</td>
</tr>
<tr>
<td>Reinforced concrete walls/ceilings</td>
<td>Typically 10 m, through max. 1 wall/ceiling</td>
</tr>
</tbody>
</table>

Placement of the KL6583 module

The KL6583 module contains transmitter, receiver and antenna.

Distances

The distance to a reinforced concrete ceiling should be at least 50 cm and to a wall 10 cm.

Do not attach or screw the KL6583 module to a metal plate!

Environmental conditions

Furthermore, the environmental conditions are to be adhered to:

• Maximum air humidity 95% without condensation
• Ambient temperature 0 - 55°C

3.2 Approval of EnOcean wireless technology

Check the admissibility of the operation in your country

The KL6583 EnOcean transceivers can be operated in following countries without registration or fee: KL6583-0000: European Union and Switzerland
KL6583-0100: USA and Canada

Permission for use in other countries must be clarified explicitly!

KL6583-0100 for USA and Canada

Includes IC: 5731A-TCM320C

Includes FCC ID: SZV-TCM320C

The affected device complies with part 15 of the FCC rules.

The operation is subject to the following conditions:

• this device must not cause adverse interference, and
• this device must absorb all received interference, including interference that would impair the operation.
# 4 Programming

## POUs/KL6021-0023

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOceanReceive</td>
<td>Communication with KL6021-0023</td>
</tr>
</tbody>
</table>

## POUs/KL6021-0023/Read

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOceanPTM100</td>
<td>Receives the signals of a PTM100 module</td>
</tr>
<tr>
<td>FB_EnOceanPTM200</td>
<td>Receives the signals of a PTM200 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM100</td>
<td>Receives the signals of a STM100 module (obsolete)</td>
</tr>
<tr>
<td>FB_EnOceanSTM100Generic</td>
<td>Receives the signals of a STM100 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM250</td>
<td>Receives the signals of a STM250 module</td>
</tr>
</tbody>
</table>

## POUs/KL6581

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_KL6581</td>
<td>Communication with a KL6581</td>
</tr>
</tbody>
</table>

## POUs/KL6581/Read

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_Rec_Generic</td>
<td>Receives all types of EnOcean telegrams</td>
</tr>
<tr>
<td>FB_Rec_1BS</td>
<td>Receives data with ORG telegram 6. Typical EnOcean device: Window contact.</td>
</tr>
<tr>
<td>FB_Rec_RPS_Switch</td>
<td>Receives data with ORG telegram 5. Typical EnOcean device: Buttons.</td>
</tr>
<tr>
<td>FB_Rec_RPS_Window_Handle</td>
<td>Receives data with ORG telegram 5. Typical EnOcean device: Window handle.</td>
</tr>
</tbody>
</table>

## POUs/KL6581/Send

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_Send_Generic</td>
<td>All kinds of EnOcean data telegrams can be sent with this block.</td>
</tr>
<tr>
<td>FB_Send_4BS</td>
<td>Sends EnOcean telegrams in the 4BS format.</td>
</tr>
<tr>
<td>FB_Send_RPS_Switch</td>
<td>Sends EnOcean telegrams in the format of a button.</td>
</tr>
<tr>
<td>FB_Send_RPS_SwitchAuto</td>
<td>This function block sends data such as those from a switch.</td>
</tr>
</tbody>
</table>

## POUs/KL6581/Other

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOcean_Search</td>
<td>This function block recognizes all EnOcean devices within its range and displays them.</td>
</tr>
<tr>
<td>FB_Rec_Teach_In</td>
<td>This function block indicates if the LRN bit in an EnOcean telegram is set, independent of its EnOcean ID.</td>
</tr>
<tr>
<td>FB_Rec_Teach_In_Ex</td>
<td>This function block indicates pressing of the Learn button at an EnOcean device.</td>
</tr>
</tbody>
</table>
### POUs/KL6581/Function

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_Byte_To_Temp [33]</td>
<td>This function converts a byte raw value into a REAL variable.</td>
</tr>
<tr>
<td>F_Byte_To_TurnSwitch [34]</td>
<td>This function converts a raw byte value to a Boolean array.</td>
</tr>
</tbody>
</table>

### DUTs/KL6021-0023/Hardware Types

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_EnOceanInData [38]</td>
<td>Process image of the KL6021-0023 inputs</td>
</tr>
<tr>
<td>ST_EnOceanOutData [38]</td>
<td>Process image of the KL6021-0023 outputs</td>
</tr>
</tbody>
</table>

### DUTs/KL6021-0023

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_EnOceanRotarySwitch [37]</td>
<td>State of the rotary-switch on the transmitting-module</td>
</tr>
<tr>
<td>E_EnOceanSensorType [36]</td>
<td>Sensor type</td>
</tr>
<tr>
<td>ST_EnOceanReceivedData [37]</td>
<td>Internal structure</td>
</tr>
</tbody>
</table>

### DUTs/KL6581/hardware types

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL6581_Input [40]</td>
<td>Process image of the KL6581 inputs</td>
</tr>
<tr>
<td>KL6581_Output [41]</td>
<td>Process image of the KL6581 outputs</td>
</tr>
</tbody>
</table>

### DUTs/KL6581

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR_EnOceanWindow [42]</td>
<td>State of the window</td>
</tr>
<tr>
<td>E_ENOCEAN_Org [39]</td>
<td>EnOcean telegram type</td>
</tr>
<tr>
<td>E_KL6581_Err [39]</td>
<td>Error messages</td>
</tr>
<tr>
<td>STR_EnOceanSwitch [42]</td>
<td>State of the buttons</td>
</tr>
<tr>
<td>STR_KL6581 [42]</td>
<td>Internal structure</td>
</tr>
<tr>
<td>STR_Teach_In [43]</td>
<td>Data structure manufacturer ID, type and profile</td>
</tr>
<tr>
<td>STREnOceanTurnSwitch [44]</td>
<td>Position of the rotary switch at the room control unit</td>
</tr>
</tbody>
</table>

### 4.1 POUs

#### KL6021-0023

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOceanReceive [14]</td>
<td>Communication with KL6021-0023</td>
</tr>
</tbody>
</table>
### KL6021-0023/Read

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOceanPTM100</td>
<td>Receives the signals of a PTM100 module</td>
</tr>
<tr>
<td>FB_EnOceanPTM200</td>
<td>Receives the signals of a PTM200 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM100</td>
<td>Receives the signals of a STM100 module (obsolete)</td>
</tr>
<tr>
<td>FB_EnOceanSTM100Generic</td>
<td>Receives the signals of a STM100 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM250</td>
<td>Receives the signals of a STM250 module</td>
</tr>
</tbody>
</table>

### KL6581

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_KL6581</td>
<td>Main block for the communication with the KL6581 and the connected KL6583 modules</td>
</tr>
</tbody>
</table>

### KL6581/Read

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_Rec_Generic</td>
<td>Receives all types of EnOcean telegrams</td>
</tr>
<tr>
<td>FB_Rec_1BS</td>
<td>Receives data with ORG telegram 6. Typical EnOcean device: Window contact.</td>
</tr>
<tr>
<td>FB_Rec_RPS_Switch</td>
<td>Receives data with ORG telegram 5. Typical EnOcean device: Buttons.</td>
</tr>
<tr>
<td>FB_Rec_RPS_Window_Handle</td>
<td>Receives data with ORG telegram 5. Typical EnOcean device: Window handle.</td>
</tr>
</tbody>
</table>

### KL6581/Send

<table>
<thead>
<tr>
<th>POUs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_Send_Generic</td>
<td>All kinds of EnOcean data telegrams can be sent with this block.</td>
</tr>
<tr>
<td>FB_Send_4BS</td>
<td>Sends EnOcean telegrams in the 4BS format.</td>
</tr>
<tr>
<td>FB_Send_RPS_Switch</td>
<td>Sends EnOcean telegrams in the format of a button.</td>
</tr>
<tr>
<td>FB_Send_RPS_SwitchAuto</td>
<td>This function block sends data such as those from a switch.</td>
</tr>
</tbody>
</table>

### KL6581/Other

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOcean_Search</td>
<td>This function block recognizes all EnOcean devices within its range and displays them.</td>
</tr>
<tr>
<td>FB_Rec_Teach_In</td>
<td>This function block indicates if the LRN bit in an EnOcean telegram is set, independent of its EnOcean ID.</td>
</tr>
<tr>
<td>FB_Rec_Teach_In_Ex</td>
<td>This function block indicates pressing of the Learn button at an EnOcean device.</td>
</tr>
</tbody>
</table>

### KL6581/Function

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_Byte_To_Temp</td>
<td>This function converts a byte raw value into a REAL variable.</td>
</tr>
<tr>
<td>F_Byte_To_TurnSwitch</td>
<td>This function converts a raw byte value to a Boolean array.</td>
</tr>
</tbody>
</table>
### 4.1.1 KL6021-0023

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOceanReceive</td>
<td>Communication with a KL6021-0023</td>
</tr>
</tbody>
</table>

#### Read

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOceanPTM100</td>
<td>Receives the signals of a PTM100 module</td>
</tr>
<tr>
<td>FB_EnOceanPTM200</td>
<td>Receives the signals of a PTM200 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM100</td>
<td>Receives the signals of a STM100 module (outdated)</td>
</tr>
<tr>
<td>FB_EnOceanSTM100Generic</td>
<td>Receives the signals of a STM100 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM250</td>
<td>Receives the signals of a STM250 module</td>
</tr>
</tbody>
</table>

#### 4.1.1.1 FB_EnOceanReceive

The function block `FB_EnOceanReceive()` is a receive block, which makes the telegrams sent by the EnOcean modules available in the structure `stEnOceanReceivedData`. This structure can then be analyzed with further blocks. The documentation for this blocks also includes sample programs, which illustrate the operating principle.

**VAR_INPUT**

```
VAR_INPUT
  bEnable : BOOL := FALSE;
```

- **bEnable**: A positive signal at this input activates the block. A negative signal at the input disables the block functionality.

**VAR_OUTPUT**

```
VAR_OUTPUT
  bError : BOOL := FALSE;
  nErrorId : UDINT := 0;
  stEnOceanReceivedData : ST_EnOceanReceivedData;
```

- **bError**: This output goes TRUE as soon as an error occurs. This error is described via the variable `nErrorId`.
- **nErrorId**: Describes the error type (see error codes [22]).
- **stEnOceanReceivedData**: This structure contains the received data (see `ST_EnOceanReceivedData` [37]).

**VAR_IN_OUT**

```
VAR_IN_OUT
  stEnOceanInData : ST_EnOceanInData;
  stEnOceanOutData : ST_EnOceanOutData;
```

- **stEnOceanInData**: Is linked with the input addresses of the KL6021-0023 in the System Manager (see `ST_EnOceanInData` [38]).
- **stEnOceanOutData**: Is linked with the output addresses of the KL6021-0023 in the System Manager (see `ST_EnOceanOutData` [38]).
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

## 4.1.1.2 Read

### Function blocks

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_EnOceanPTM100 (15)</td>
<td>Receives the signals of a PTM100 module</td>
</tr>
<tr>
<td>FB_EnOceanPTM200 (16)</td>
<td>Receives the signals of a PTM200 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM100 (18)</td>
<td>Receives the signals of a STM100 module (outdated)</td>
</tr>
<tr>
<td>FB_EnOceanSTM100Generic (20)</td>
<td>Receives the signals of a STM100 module</td>
</tr>
<tr>
<td>FB_EnOceanSTM250 (21)</td>
<td>Receives the signals of a STM250 module</td>
</tr>
</tbody>
</table>

### 4.1.1.2.1 FB_EnOceanPTM100

The function block **FB_EnOceanPTM100()** provides a user-friendly evaluation of the state of an EnOcean PTM100 module. The function block **FB_EnOceanReceive()** (14) is required for this purpose.

In contrast to the PTM200 and PTM250 modules, only one button at a time can be pressed in the PTM100 module. In addition, the PTM100 module supports eight buttons, not four.

A new instance of this function block must be created for each button module used.

### VAR_INPUT

- **bEnable**: A positive signal at this input activates the block. A negative signal at the input disables the block functionality, and all outputs are set to 0 or FALSE.
- **tWatchdog**: Monitoring time. Within this time, new information must reach this block via the input **stEnOceanReceivedData** described below. If this time is set to t#0s, the watchdog function is inactive.
- **nTransmitterId**: ID of the EnOcean module, to which the block should respond.
- **stEnOceanReceivedData**: Information and required connection to the EnOcean receive block **FB_EnOceanReceive()** (14). This information is stored in a structure (see **ST_EnOceanReceivedData** (37)).

### VAR_OUTPUT

- **bSwitches**: This field of 8 Boolean values describes the states of the 8 buttons on the button module.
- **bError**: this output goes TRUE as soon as an error occurs. This error is described via the variable **nErrorId**.
nErrorId: Describes the error type (see error codes [»22]).

The following sample program illustrates the operating principle of this block:

```plaintext
PROGRAM MAIN
VAR
    fbEnOceanReceive : FB_EnOceanReceive;
    fbEnOceanPTM100_1 : FB_EnOceanPTM100;
    fbEnOceanPTM100_2 : FB_EnOceanPTM100;
    bSwitches1 : ARRAY [0..7] OF BOOL;
    bSwitches2_1 : BOOL;
    bSwitches2_2 : BOOL;
    bSwitches2_3 : BOOL;
    bSwitches2_4 : BOOL;
    bSwitches2_5 : BOOL;
    bSwitches2_6 : BOOL;
    bSwitches2_7 : BOOL;
    bSwitches2_8 : BOOL;
END_VAR

fbEnOceanReceive(
    bEnable := TRUE,
    stEnOceanInData := stEnOceanInData,
    stEnOceanOutData := stEnOceanOutData);

fbEnOceanPTM100_1(
    bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.bEnable,
    nTransmitterId := 16#000000C4,
    tWatchdog:=t#0s,
    stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData);

bSwitches1 := fbEnOceanPTM100_1.bSwitches;

fbEnOceanPTM100_2(
    bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.bEnable,
    nTransmitterId := 16#000000C5,
    tWatchdog:=t#0s,
    stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData);

bSwitches2_1 := fbEnOceanPTM100_2.bSwitches[0];
bSwitches2_2 := fbEnOceanPTM100_2.bSwitches[1];
bSwitches2_3 := fbEnOceanPTM100_2.bSwitches[2];
bSwitches2_4 := fbEnOceanPTM100_2.bSwitches[3];
bSwitches2_5 := fbEnOceanPTM100_2.bSwitches[4];
bSwitches2_6 := fbEnOceanPTM100_2.bSwitches[5];
bSwitches2_7 := fbEnOceanPTM100_2.bSwitches[6];

In this example program two transmitter modules (PTM100) are queried: a transmitter module with the transmitter ID 16#C4 and another module with the transmitter ID 16#C5. A function block FB_EnOceanPTM100 is created for both transmitter modules. Both function blocks obtain their information from the upstream receive block FB_EnOceanReceive and are only active (input bEnable) if the receive block is active and not in error. The buttons of the first transmitter module are assigned to a Boolean array bSwitches1 for further analysis, while the buttons of the second transmitter module are assigned to individual Boolean variables bSwitches2_1 to bSwitches2_8; both options are conceivable.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.1.2.2 FB_EnOceanPTM200

The function block FB_EnOceanPTM200() provides a user-friendly evaluation of the state of an EnOcean PTM200 or PTM250 module. The function block FB_EnOceanReceive() [»14] is required for this purpose.
In contrast to the PTM100 module, in the PTM200/250 module two buttons can be pressed simultaneously. In addition, the PTM200/250 module supports four, not eight buttons.

A new instance of this function block must be created for each button module used.

**VAR_INPUT**

- \( bEnable \) : BOOL := FALSE;
- \( tWatchdog \) : TIME;
- \( nTransmitterId \) : UDINT;
- \( stEnOceanReceivedData \) : ST_EnOceanReceivedData;

\( bEnable \): A positive signal at this input activates the block. A negative signal at the input disables the block functionality, and all outputs are set to 0 or FALSE.

\( tWatchdog \): Monitoring time. Within this time, new information must reach this block via the input \( stEnOceanReceivedData \) described below. If this time is set to \( #0s \), the watchdog function is inactive.

\( nTransmitterId \): ID of the EnOcean module, to which the block should respond.

\( stEnOceanReceivedData \): Information and required connection to the EnOcean receive block \( FB_EnOceanReceive() \). This information is stored in a structure (see \( ST_EnOceanReceivedData \)).

**VAR_OUTPUT**

- \( bSwitches \) : ARRAY [0..3] OF BOOL;
- \( bError \) : BOOL := FALSE;
- \( nErrorId \) : UDINT := 0;

\( bSwitches \): This field of 4 Boolean values describes the states of the 4 buttons on the button module.

\( bError \): This output goes TRUE as soon as an error occurs. This error is described via the variable \( nErrorId \).

\( nErrorId \): Describes the error type (see \( error codes \)).

The following sample program illustrates the operating principle of this block:

```plaintext
PROGRAM MAIN
VAR
  fbEnOceanReceive : FB_EnOceanReceive;
  fbEnOceanPTM100_1 : FB_EnOceanPTM200;
  fbEnOceanPTM100_2 : FB_EnOceanPTM200;
  bSwitches1 : ARRAY [0..3] OF BOOL;
  bSwitches2_1 : BOOL;
  bSwitches2_2 : BOOL;
  bSwitches2_3 : BOOL;
  bSwitches2_4 : BOOL;
END_VAR

fbEnOceanReceive(
  bEnable := TRUE,
  stEnOceanInData := stEnOceanInData
  stEnOceanOutData := stEnOceanOutData);

fbEnOceanPTM200_1(
  bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.bEnable,
  nTransmitterId := 16#000000C6,
  tWatchdog:=t#0s,
  stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData);

fbEnOceanPTM200_2(
  bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.bEnable,
  nTransmitterId := 16#000000C7,
  tWatchdog:=t#0s,
  stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData);

bSwitches2_1 := fbEnOceanPTM200_2.bSwitches[0];
  bSwitches2_2 := fbEnOceanPTM200_2.bSwitches[1];
  bSwitches2_3 := fbEnOceanPTM200_2.bSwitches[2];
  bSwitches2_4 := fbEnOceanPTM200_2.bSwitches[3];
```

VAR_OUTPUT
In this example program two transmitter modules (PTM200/PTM250) are queried; one with the transmitter ID 16#C6, another one with the transmitter ID 16#C7. A function block `FB_EnOceanPTM200` is created for both transmitter modules. Both function blocks obtain their information from the upstream receive block `FB_EnOceanReceive` and are only active (input `bEnable`) if the receive block is active and not in error. The buttons of the first transmitter module are assigned to a Boolean array `bSwitches1` for further analysis, while the buttons of the second transmitter module are assigned to individual Boolean variables `bSwitches2_1` to `bSwitches2_8`; both options are conceivable.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.1.2.3  **FB_EnOceanSTM100**

The function block `FB_EnOceanSTM100()` provides a user-friendly evaluation of the data of an EnOcean STM100 module. The function block `FB_EnOceanReceive()` is required for this purpose.

A new instance of this function block must be created for each button module used.

**VAR_INPUT**

- `bEnable` : BOOL := FALSE;
- `tWatchdog` : TIME;
- `nTransmitterId` : UDINT;
- `stEnOceanReceivedData` : ST_EnOceanReceivedData;

- **bEnable**: A positive signal at this input activates the block. A negative signal at the input disables the block functionality, and all outputs are set to 0 or FALSE.

- **tWatchdog**: Monitoring time. Within this time, new information must reach this block via the input `stEnOceanReceivedData` described below. If this time is set to t#0s, the watchdog function is inactive.

- **nTransmitterId**: ID of the EnOcean module, to which the block should respond.

- **stEnOceanReceivedData**: Information and required connection to the EnOcean receive block `FB_EnOceanReceive()` [14]. This information is stored in a structure (see `ST_EnOceanReceivedData` [37]).

**VAR_OUTPUT**

- `eEnOceanRotarySwitch` : E_EnOceanRotarySwitch;
- `nSetpoint` : INT;
- `nTemperature` : INT;
- `bPresentSwitch` : BOOL;

For new projects the block `FB_EnOceanSTM100Generic()` [20] should be used!
bLearnSwitch         : BOOL;
bError             : BOOL := FALSE;
nErrorId         := UDINT := 0;

**nErrorId:** Describes the error type (see error codes [22]).

The following sample program illustrates the operating principle of this block:

```plaintext
PROGRAM MAIN
VAR
  fbEnOceanReceive : FB_EnOceanReceive;
  fbEnOceanSTM100_1 : FB_EnOceanSTM100;
  fbEnOceanSTM100_2 : FB_EnOceanSTM100;
  nTemperature : ARRAY [1..2] OF INT;
  nSetpoint : ARRAY [1..2] OF INT;
  nStateRotarySwitch : ARRAY [1..2] OF E_EnOceanRotarySwitch;
  bPresentSwitch : ARRAY [1..2] OF BOOL;
END_VAR

fbEnOceanReceive(
  bEnable := TRUE,
  stEnOceanInData := stEnOceanInData,
  stEnOceanOutData := stEnOceanOutData);

fbEnOceanSTM100_1(
  bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceve.bEnable,
  nTransmitterId := 16#000000C4,
  tWatchdog := t#1h,
  stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData
  nTemperature => Temperature[1],
  nSetpoint => nSetpoint[1],
  eEnOceanRotarySwitch => nStateRotarySwitch[1],
  bPresentSwitch => bPresentSwitch[1]);

fbEnOceanSTM100_2(
  bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceve.bEnable,
  nTransmitterId := 16#000000C5,
  tWatchdog := t#0s,
  stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData
  nTemperature => Temperature[2],
  nSetpoint => nSetpoint[2],
  eEnOceanRotarySwitch => nStateRotarySwitch[2],
  bPresentSwitch => bPresentSwitch[2]);

In this example program two room control units are queried; one with the transmitter ID 16#000000C4 and another one with the transmitter ID 16#000000C5. A function block FB_EnOceanSTM100 is created for both modules. Both function blocks obtain their information from the upstream receive block FB_EnOceanReceive [14] and are only active (input bEnable) if the receive block is active and not in error. The first device monitored with the watchdog function. New values have to be transferred to the controller within 1 hour; the second device is programmed without watchdog monitoring. The output values at the function blocks are assigned flags for further evaluation.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
### FB_EnOceanSTM100Generic

The function block **FB_EnOceanSTM100Generic** provides a user-friendly evaluation of the data of an EnOcean STM100 module. The function block **FB_EnOceanReceive()** is required for this purpose.

![Function Block Diagram](image)

**VAR_INPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL := TRUE;</td>
<td>A positive signal at this input activates the block. A negative signal at the input disables the block functionality, and all outputs are set to 0 or FALSE.</td>
</tr>
<tr>
<td>tWatchdog</td>
<td>TIME;</td>
<td>Monitoring time. Within this time, new information must reach this block via the input stEnOceanReceivedData described below. If this time is set to t#0s, the watchdog function is inactive.</td>
</tr>
<tr>
<td>nTransmitterId</td>
<td>UDINT;</td>
<td>ID of the EnOcean module, to which the block should respond.</td>
</tr>
<tr>
<td>stEnOceanReceivedData</td>
<td></td>
<td>Information and required connection to the EnOcean receive block FB_EnOceanReceive(). This information is stored in a structure (see ST_EnOceanReceivedData).</td>
</tr>
</tbody>
</table>

**VAR_OUTPUT**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nDataBytes</td>
<td>ARRAY [0..3] OF BYTE;</td>
<td>4-byte array with the user data sent by the STM100 module. The purpose of the individual bytes is manufacturer-specific.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL := FALSE;</td>
<td>this output goes TRUE as soon as an error occurs. This error is described via the variable nErrorId.</td>
</tr>
<tr>
<td>nErrorId</td>
<td>UDINT := 0;</td>
<td>Describes the error type (see error codes).</td>
</tr>
</tbody>
</table>

The following sample program illustrates the operating principle of this block:

```plaintext
PROGRAM MAIN
VAR
    fbEnOceanReceive : FB_EnOceanReceive;
    fbEnOceanSTM100_1 : FB_EnOceanSTM100Generic;
    fbEnOceanSTM100_2 : FB_EnOceanSTM100Generic;
    nTemperature : ARRAY [1..2] OF BYTE;
    nSetpoint : ARRAY [1..2] OF BYTE;
    nStateRotarySwitch : ARRAY [1..2] OF BYTE;
    nPresentSwitch : ARRAY [1..2] OF BYTE;
END_VAR

fbEnOceanReceive(
    bEnable := TRUE,
    stEnOceanInData := stEnOceanInData,
    stEnOceanOutData := stEnOceanOutData);

fbEnOceanSTM100_1(
    bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.bEnable,
    nTransmitterId := 16#000000C4,
    tWatchdog := t#1h,
    stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData;
    nTemperature[1] := fbEnOceanSTM100_1.nDataBytes[0];
```

A new instance of this function block must be created for each button module used.
In this example program two EnOcean transmitter modules are queried; one with the transmitter ID 16#000000C4, another one with the transmitter ID 16#000000C5. A function block FB_EnOceanSTM100Generic is created for both transmitters. Both function blocks obtain their information from the upstream receive block FB_EnOceanReceive and are only active (input bEnable) if the receive block is active and not in error. The first device monitored with the watchdog function. New values have to be transferred to the controller within 1 hour; the second device is programmed without watchdog monitoring. The output values at the function blocks are assigned variables for further evaluation. Before the values can be used further, they have to be scaled to physical values. Details of the conversion can be found in the data sheet for the sensor.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.1.2.5 FB_EnOceanSTM250

The function block FB_EnOceanSTM250() provides a user-friendly evaluation of the data of an EnOcean STM250 module. The function block FB_EnOceanReceive() is required for this purpose.

A new instance of this function block must be created for each STM100 module used.

VAR_INPUT

<table>
<thead>
<tr>
<th>bEnable</th>
<th>bState</th>
</tr>
</thead>
<tbody>
<tr>
<td>tWatchdog</td>
<td>bLearn</td>
</tr>
<tr>
<td>nTransmitterId</td>
<td>bError</td>
</tr>
<tr>
<td>stEnOceanReceivedData</td>
<td>nErroId</td>
</tr>
</tbody>
</table>

bEnable: A positive signal at this input activates the block. A negative signal at the input disables the block functionality, and all outputs are set to 0 or FALSE.

tWatchdog: Monitoring time. Within this time, new information must reach this block via the input stEnOceanReceivedData described below. If this time is set to t#0s, the watchdog function is inactive.

nTransmitterId: ID of the EnOcean module, to which the block should respond.

stEnOceanReceivedData: Information and required connection to the EnOcean receive block FB_EnOceanReceive(). This information is stored in a structure (see ST_EnOceanReceivedData).
VAR_OUTPUT

bState : BOOL;
bLearn : BOOL;
bError : BOOL := FALSE;
nErrorId : UDINT := 0;

bState: Upon activation of the reed contact in the STM250 module, this output becomes TRUE (contact closed).

bLearn: This output becomes FALSE if the teach-in button at the STM250 module is activated.

bError: this output goes TRUE as soon as an error occurs. This error is described via the variable nErrorId.

nErrorId: Describes the error type (see error codes [22]).

The following sample program illustrates the operating principle of this block:

```plaintext
PROGRAM MAIN
VAR
  fbEnOceanReceive : FB_EnOceanReceive;
  fbEnOceanSTM250 : FB_EnOceanSTM250;
  bState : BOOL;
  bLearn : BOOL;
END_VAR

fbEnOceanReceive(
  bEnable := TRUE,
  stEnOceanInData := stEnOceanInData,
  stEnOceanOutData := stEnOceanOutData);

fbEnOceanSTM250(
  bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.bEnable,
  nTransmitterId := 16#000008CA,
  tWatchdog:=t#0s,
  stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData
  bState => bState,
  bLearn => bLearn);
```

In this example program an STM250 module with the transmitter ID 16#000008CA is queried. To this end the function block FB_EnOceanSTM250 is created. This function blocks obtains its information from the upstream receive block FB_EnOceanReceive [14] and is only active (input bEnable) if the receive block is active and not in error. The output values at the function blocks are assigned variables for further evaluation.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.1.3 Error codes

<table>
<thead>
<tr>
<th>Value (hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>No error.</td>
</tr>
<tr>
<td>0x0001</td>
<td>Checksum error.</td>
</tr>
<tr>
<td>0x0002</td>
<td>Watchdog monitoring.</td>
</tr>
<tr>
<td>0x0003</td>
<td>Buffer overflow (in the KL6023)</td>
</tr>
<tr>
<td>0x0004</td>
<td>No data received yet from sensor receive.</td>
</tr>
</tbody>
</table>
4.1.2KL6581

4.1.2.1FB_KL6581

This function block takes care of communication with the KL6581 EnOcean Bus Terminal. The KL6581 is configured and data exchange with the EnOcean network is started via this block.

Restrictions:
- This block may only be called once per KL6581!
- In the PLC project this block may only be called once per cycle!
- A PLC project may contain a maximum of 64 KL6581 terminals!

VAR_INPUT

<table>
<thead>
<tr>
<th>bInit</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>nIdx</td>
<td>USINT</td>
</tr>
</tbody>
</table>

bInit: Activates the block that configures the KL6301 and then activates the data exchange.

nIdx: The idx number must be unique for each KL6581, if more than one Bus Terminal per PLC program is used (valid values: 1...64).

VAR_OUTPUT

<table>
<thead>
<tr>
<th>bReady</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>bBusy</td>
<td>BOOL</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
</tr>
<tr>
<td>iErrorID</td>
<td>E_KL6581_Err</td>
</tr>
<tr>
<td>str_KL6581</td>
<td>STR_KL6581</td>
</tr>
</tbody>
</table>

bReady: The block is ready for sending and receiving data.

bBusy: The block is active.

bError: this output goes TRUE as soon as an error occurs. This error is described via the iErrorID variable.

iErrorID: Describes the error type (see E_KL6581_Err [39]).

str_KL6581: Is linked to the send and receive blocks (see STR_KL6581 [42]).

VAR_IN_OUT

<table>
<thead>
<tr>
<th>stKL6581_in</th>
<th>KL6581_Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>stKL6581_out</td>
<td>KL6581_Output</td>
</tr>
</tbody>
</table>

stKL6581_in: Is linked to the input addresses of the KL6581 in the System Manager (see KL6581_Input [40]).

stKL6581_out: Is linked to the output addresses of the KL6581 in the System Manager (see KL6581_Output [41]).
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

### 4.1.2.2 Read

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_Rec_Generic [24]</td>
<td>Receives all types of EnOcean telegrams</td>
</tr>
<tr>
<td>FB_Rec_1BS [25]</td>
<td>Receives data with ORG telegram 6. Typical EnOcean device: Window contact</td>
</tr>
<tr>
<td>FB_Rec_RPS_Window_Handle [26]</td>
<td>Receives data with ORG telegram 5. Typical EnOcean device: Window handle</td>
</tr>
</tbody>
</table>

#### 4.1.2.2.1 FB_Rec_Generic

This function block receives all data that were received via EnOcean. This block can be used for all kinds of EnOcean telegrams.

The user must interpret the data himself. The manufacturer’s documentation for the sending EnOcean device is necessary for this.

**VAR_INPUT**

- str_KL6581 : STR_KL6581;
- byNode : BYTE;
- dw_ID : DWORD;

**VAR_OUTPUT**

- ar_Value : ARRAY [0..3] OF BYTE;
- by_Node : BYTE;
- by_STATE : BYTE;
- bReceive : BOOL := TRUE;
- EnOceanTyp : E_EnOcean_Org;

**str_KL6581**: Is linked with the data structure of block **FB_KL6581** [23] (see **STR_KL6581** [42]).

**byNode**: Filter – if the value is zero the EnOcean telegrams from all KL6583s are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

**dw_ID**: EnOcean ID to be received.

**ar_Value**: 4-byte EnOcean data.

**by_Node**: Node number of the KL6583 that has received the EnOcean telegram.

**by_STATE**: EnOcean STATUS field.

**bReceive**: On receiving an EnOcean telegram this value is set to FALSE for one cycle.

**EnOceanTyp**: EnOcean ORG field (see **E_EnOcean_Org** [39]).
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.2.2.2 FB_Rec_1BS

This function block receives data that were received via EnOcean. This block is used for integration of window contacts, for example (ORG field 6).

VAR_INPUT

```
str_KL6581 : STR_KL6581;
byNode    : BYTE;
dw_ID     : DWORD;
```

- `str_KL6581`: Is linked with the data structure of block FB_KL6581() [23] (see STR_KL6581 [42]).
- `byNode`: Filter - if the value is zero the EnOcean telegrams from all KL6583s are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.
- `dw_ID`: EnOcean ID to be received.

VAR_OUTPUT

```
bOpen    : BOOL;
bClose   : BOOL;
bLRN     : BOOL;
by_Node  : BYTE;
bReceive : BOOL := TRUE;
```

- `bOpen`: Contact open.
- `bClose`: Contact closed.
- `bLRN`: LRN button pressed.
- `by_Node`: Node number of the KL6583 that has received the EnOcean telegram.
- `bReceive`: On receiving an EnOcean telegram this value is set to FALSE for one cycle.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.2.2.3 FB_Rec_RPS_Switch

```
str_KL6581 str_EnOceanSwitch
byNode     by_Node
dw_ID      bReceive
```

This function block receives data that were received via EnOcean. This block is used for integration of window contacts, for example (ORG field 6).
This function block receives data from a switch that were received via EnOcean. The block outputs the data in a data structure (ORG field 5).

**VAR_INPUT**
```
str_KL6581 : STR_KL6581;
byNode : BYTE;
dw_ID : DWORD;
```

*str_KL6581*: Is linked with the data structure of block `FB_KL6581()` [23] (see `STR_KL6581` [42]).

*byNode*: Filter - if the value is zero the EnOcean telegrams from all KL6583s are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

*dw_ID*: EnOcean ID to be received.

**VAR_OUTPUT**
```
str_EnOceanSwitch : STR_EnOceanSwitch;
by_Node : BYTE;
bReceive : BOOL := TRUE;
```

*str_EnOceanSwitch*: Switch data (see `STR_EnOceanSwitch` [42]).

*by_Node*: Node number of the KL6583 that has received the EnOcean telegram.

*bReceive*: On receiving an EnOcean telegram this value is set to FALSE for one cycle.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

### 4.1.2.2.4 FB_Rec_RPS_Window_Handle

```
FB_Rec_RPS_Window_Handle

- str_KL6581 ar_Data
- byNode by_Node
- dw_ID bReceive
```

This function block receives data from a window handle that were received via EnOcean. The block outputs the data in a data structure (ORG field 5).

**VAR_INPUT**
```
str_KL6581 : STR_KL6581;
byNode : BYTE;
dw_ID : DWORD;
```

*str_KL6581*: Is linked with the data structure of block `FB_KL6581()` [23] (see `STR_KL6581` [42]).

*byNode*: Filter - if the value is zero the EnOcean telegrams from all KL6583s are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

*dw_ID*: EnOcean ID to be received.

**VAR_OUTPUT**
```
ar_Data : AR_EnOceanWindow;
by_Node : BYTE;
bReceive : BOOL := TRUE;
```

*ar_Data*: window handle data (see `set_EnOceanWindow` [42]).

*by_Node*: Node number of the KL6583 that has received the EnOcean telegram.
**bReceive**: On receiving an EnOcean telegram this value is set to FALSE for one cycle.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

### 4.1.2.3 Send

#### Function blocks

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB_Send_Generic</td>
<td>Sends arbitrary EnOcean telegrams</td>
</tr>
<tr>
<td>FB_Send_4BS</td>
<td>Sends EnOcean telegrams in the 4BS format</td>
</tr>
<tr>
<td>FB_Send_RPS_Switch</td>
<td>Sends EnOcean telegrams in the format of a button</td>
</tr>
<tr>
<td>FB_Send_RPS_SwitchAuto</td>
<td>Sends EnOcean telegrams in the format of a button</td>
</tr>
</tbody>
</table>

#### 4.1.2.3.1 FB_Send_Generic

```plaintext
FB_Send_Generic

- bStart
- by_Node
- by_ORG
- pt_SendData
- by_STATE
- nEnOceanID
- str_KL6581
```

This function block sends data via EnOcean. The type and the data contents are arbitrary. All kinds of EnOcean data telegrams can be sent with this block.

#### VAR_INPUT

```plaintext
bStart : BOOL;
by_Node : BYTE;
by_ORG  : E_EnOcean_Org;
pt_SendData : DWORD;
by_STATE  : BYTE;
nEnOceanID : BYTE;
str_KL6581 : STR_KL6581;
```

- **bStart**: A rising edge sends the data.

- **by_Node**: Address of the KL6583 module to which the telegram is to be sent (valid values: 1...8).

- **by_ORG**: ORG field of the EnOcean telegram (see E_EnOcean_Org [39]).

- **pt_SendData**: Pointer to the data to be sent. ADR is used to determine the pointer address. The pointer must point to 4-byte variable.

- **by_STATE**: EnOcean STATE. Can be changed by the TCM module.

- **nEnOceanID**: Virtual EnOcean ID. A value of 0...127 is added to the real EnOcean ID (valid values: 0...127).

- **str_KL6581**: Is linked with the data structure of block FB_KL6581 () (see STR_KL6581 [42]).
VAR_OUTPUT
bBusy : BOOL;
setError : BOOL;
iErrorID : E_KL6581_Err;

bBusy: The block is active. No new data can be sent at this stage.

setError: this output goes TRUE as soon as an error occurs. This error is described via the iErrorID variable.

iErrorID: Describes the error type (see E_KL6581_Err [39]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.2.3.2 FB_Send_4BS

This function block sends data via EnOcean. The ORG field is set permanently to 7.

VAR_INPUT
bStart : BOOL;
by_Node : BYTE;
pt_SendData : DWORD;
nEnOceanID : BYTE;
str_KL6581 : STR_KL6581;

bStart: A rising edge sends the data.

by_Node: Address of the KL6583 module to which the telegram is to be sent (valid values: 1...8).

pt_SendData: Pointer to the data to be sent. ADR is used to determine the pointer address. The pointer must point to 4-byte variable.

nEnOceanID: Virtual EnOcean ID. A value of 0...127 is added to the real EnOcean ID (valid values: 0...127).

str_KL6581: Is linked with the data structure of block FB_KL6581() [23] (see STR_KL6581 [42]).

VAR_OUTPUT
bBusy : BOOL;
setError : BOOL;
iErrorID : E_KL6581_Err;

bBusy: The block is active. No new data can be sent at this stage.

setError: this output goes TRUE as soon as an error occurs. This error is described via the iErrorID variable.

iErrorID: Describes the error type (see E_KL6581_Err [39]).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
4.1.2.3.3  FB_Send_RPS_Switch

This block sends EnOcean telegrams in the format of a button. The value of bData is sent with a positive edge of bStart. In order to simulate a keystroke, the block usually has to be started twice, once with bData = TRUE, once with bData = FALSE. For simpler handling the block FB_Send_RPS_SwitchAuto() can be used.

VAR_INPUT

bStart : BOOL;
by_Node : BYTE;
bData : BOOL;
nRockerID : INT;
nEnOceanID : BYTE;
str_KL6581 : STR_KL6581;

bStart: A rising edge sends the data.
by_Node: Address of the KL6583 module to which the telegram is to be sent (valid values: 1...8).
bData: Value to be transferred.
nRockerID: Button number, valid values 0 to 3.
nEnOceanID: Virtual EnOcean ID. A value of 0...127 is added to the real EnOcean ID (valid values: 0...127).
str_KL6581: Is linked with the data structure of block FB_KL6581() (see STR_KL6581).

VAR_OUTPUT

bBusy : BOOL;
bError : BOOL;
iErrorID : E_KL6581_Err;

bBusy: The block is active. No new data can be sent at this stage.
bError: this output goes TRUE as soon as an error occurs. This error is described via the iErrorID variable.
iErrorID: Describes the error type (see E_KL6581_Err).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
### 4.1.2.3.4 FB_Send_RPS_SwitchAuto

This block sends EnOcean telegrams in the format of a button. The value of `bData` is sent with a positive edge of `bStart`. The signal "Release button" is sent once the time `t_SwitchDelay` has elapsed.

#### VAR_INPUT

- `bStart` : BOOL;
- `bData` : BOOL;
- `by_Node` : BYTE;
- `t_SwitchDelay` : TIME := T#100ms;
- `nRockerID` : INT;
- `nEnOceanID` : BYTE;
- `str_KL6581` : STR_KL6581;

`bStart`: A rising edge sends the data.

`bData`: Value to be transmitted.

`by_Node`: Address of the KL6583 module to which the telegram is to be sent (valid values: 1...8).

`t_SwitchDelay`: How long the button has to be pressed.

`nRockerID`: Button number, valid values 0 to 3.

`nEnOceanID`: Virtual EnOcean ID. A value of 0...127 is added to the real EnOcean ID (valid values: 0...127).

`str_KL6581`: Is linked with the data structure of block `FB_KL6581` [23] (see `STR_KL6581` [42]).

#### VAR_OUTPUT

- `bBusy` : BOOL;
- `bError` : BOOL;
- `iErrorID` : E_KL6581_Err;

`bBusy`: The block is active. No new data can be sent at this stage.

`bError`: this output goes TRUE as soon as an error occurs. This error is described via the `iErrorID` variable.

`iErrorID`: Describes the error type (see `E_KL6581_Err` [39]).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
4.1.2.4 Other

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB EnOcean Search [31]</td>
<td>Function block recognizes all EnOcean devices within its range and displays them.</td>
</tr>
<tr>
<td>FB Rec Teach In [32]</td>
<td>This function block indicates if the LRN bit in an EnOcean telegram is set, independent of its EnOcean ID.</td>
</tr>
<tr>
<td>FB Rec Teach In Ex [32]</td>
<td>This function block indicates pressing of the Learn button at an EnOcean device.</td>
</tr>
</tbody>
</table>

4.1.2.4.1 FB EnOcean_Search

This function block displays all EnOcean IDs that it has received and enters them in a reception array, (ar_ID). Up to 256 EnOcean devices can be recognized. Alternatively the block can also be created separately for each KL6583. This allows you to recognize whether an EnOcean device is received by several KL6583s.

**VAR_INPUT**

```plaintext
VAR_INPUT
bStart : BOOL;
str_KL6581 : STR_KL6581;
byNode : BYTE;
```

- **bStart**: If TRUE the block is activated, if FALSE it is deactivated.
- **str_KL6581**: Is linked with the data structure of block FB_KL6581 [23] (see STR_KL6581 [42]).
- **byNode**: Filter - if the value is zero the EnOcean telegrams from all KL6583s are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

**VAR_OUTPUT**

```plaintext
VAR_OUTPUT
bReceive : BOOL := TRUE;
iDevices : INT;
ar_ID : ARRAY [0..255] OF DWORD;
```

- **bReceive**: On receiving an EnOcean telegram this value is set to FALSE for one cycle.
- **iDevices**: Number of EnOcean devices found.
- **ar_ID**: EnOcean IDs that were found.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
4.1.2.4.2  FB_Rec_Teach_In

This function block indicates when a learn button is pressed on an EnOcean device. If the flag bLearnType is set, further information about the EnOcean device can be read. This function must be provided by the EnOcean device. So far, however, it is only supported by very few EnOcean devices.

VAR_INPUT

bStart : BOOL;
byNode : BYTE;
str_KL6581 : STR_KL6581;

bStart: If TRUE the block is activated, if FALSE it is deactivated.

byNode: Filter - if the value is zero the EnOcean telegrams from all KL6583s are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

str_KL6581: Is linked with the data structure of block FB_KL6581() ([23]) (see STR_KL6581 ([42]))

VAR_OUTPUT

bLearnType : BOOL;
by_Node : BYTE;
dw_ID : DWORD;
str_Teach_In : STR_Teach_In;
bReceive := BOOL := TRUE;

bLearnType: If the bit is set you will find further data in the str_Teach_In structure.

by_Node: Number of EnOcean devices found.

dw_ID: EnOcean ID for which the teach-in button was pressed.

str_Teach_In: Data structure, profile, type and manufacturer ID (see STR_Teach_In ([42]))

bReceive: On receiving an EnOcean telegram this value is set to FALSE for one cycle.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.2.4.3  FB_Rec_Teach_In_Ex

This function block indicates when a learn button is pressed on an EnOcean device. If the flag bLearnType is set, further information about the EnOcean device can be read. This function must be provided by the EnOcean device. So far, however, it is only supported by very few EnOcean devices.
In addition to the FB_Rec_Teach_In() function block, the system checks for an EEP telegram.

**VAR_INPUT**

```plaintext
bStart : BOOL;
byNode : BYTE;
str_KL6581 : STR_KL6581;
```

*bStart*: If TRUE the block is activated, if FALSE it is deactivated.

*byNode*: Filter - if the value is zero the EnOcean telegrams from all KL6583s are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

*str_KL6581*: Is linked with the data structure of block FB_KL6581() (see STR_KL6581).

**VAR_OUTPUT**

```plaintext
bLearnType : BOOL;
by_Node : BYTE;
dw_ID : DWORD;
str_Teach_In : STR_Teach;
bReceive : BOOL := TRUE;
```

*bLearnType*: If the bit is set you will find further data in the str_Teach_In structure.

*by_Node*: Number of EnOcean devices found.

*dw_ID*: EnOcean ID for which the teach-in button was pressed.

*str_Teach_In*: Data structure, function, type and manufacturer ID (see STR_Teach).

*bReceive*: On receiving an EnOcean telegram this value is set to FALSE for one cycle.

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.32</td>
<td>Tc2_EnOcean from v3.4.6.0</td>
</tr>
</tbody>
</table>

**4.1.2.5 Function**

<table>
<thead>
<tr>
<th>Function blocks</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_Byte_To_Temp</td>
<td>This function converts a byte raw value into a REAL variable.</td>
</tr>
<tr>
<td>F_Byte_To_TurnSwitch</td>
<td>This function converts a raw byte value to a Boolean array.</td>
</tr>
</tbody>
</table>

**4.1.2.5.1 F_Byte_to_Temp : REAL**

```plaintext
F_Byte_to_Temp(byData, minTemp, maxTemp)
```

This function converts a byte raw value into a REAL variable.

In EnOcean, temperature data are transmitted in a certain format, which is one byte in size. These data are usually scaled to a certain temperature value.

For example, a value is transmitted from a range of values from 0 to 40°C. The minimum and maximum data value and the raw value are transferred to the function. The output of the function then outputs the temperature as REAL variable.
VAR_INPUT
byData : BYTE;
minTemp : REAL := 0;
maxTemp : REAL := 40;

byData: Raw data.
minTemp: Minimum temperature.
maxTemp: Maximum temperature.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.1.2.5.2 F_Byte_to_TurnSwitch

This function converts a raw byte value to a Boolean array in the form of a data structure (see STREnOceanTurnSwitch) [44].

VAR_INPUT
byData : BYTE;

byData: Raw data.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
### 4.1.2.6 Error codes

<table>
<thead>
<tr>
<th>Value (hex)</th>
<th>Value (dec)</th>
<th>Value (enum)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0</td>
<td>NO_ERROR</td>
<td>No error at the block.</td>
</tr>
<tr>
<td>0x000A</td>
<td>10</td>
<td>KL6581_WrongTerminal</td>
<td>Incorrect terminal connected.</td>
</tr>
<tr>
<td>0x0010</td>
<td>16</td>
<td>KL6581_WatchdogError</td>
<td>Timeout during initialization of function block FB_KL6581().</td>
</tr>
<tr>
<td>0x0011</td>
<td>17</td>
<td>KL6581_NoComWithKL6581</td>
<td>This message usually means that there is no connection to the terminal. Terminal linked to the variables in the System Manager? Terminal plugged in incorrectly? Everything corrected, everything translated and re-read into the System Manager?</td>
</tr>
<tr>
<td>0x0012</td>
<td>18</td>
<td>KL6581_idx_number_not_OK</td>
<td>The input variable nIdx of block FB_KL6581() is greater than 64.</td>
</tr>
<tr>
<td>0x0013</td>
<td>19</td>
<td>KL6581_Switch_to_Stopp</td>
<td>The terminal has exited the data exchange with the EnOcean transmitter and receiver KL6583-0000, no EnOcean data was sent or received.</td>
</tr>
<tr>
<td>0x0014</td>
<td>20</td>
<td>KL6581_not_ready</td>
<td>Internal message for the function blocks connected to the FB_KL6581().</td>
</tr>
<tr>
<td>0x0015</td>
<td>21</td>
<td>KL6581_No_KL6853_Found</td>
<td>No KL6583 is connected to the EnOcean master terminal KL6581, or the communication does not exist.</td>
</tr>
<tr>
<td>0x0016</td>
<td>22</td>
<td>KL6581_TransmissionError</td>
<td>Data could not sent; check the address of the KL6583, or KL6583 not ready for operation.</td>
</tr>
</tbody>
</table>

### 4.2 DUTs

#### KL6021-0023/Hardware Types

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_EnOceanInData</td>
<td>Process image of the inputs</td>
</tr>
<tr>
<td>ST_EnOceanOutData</td>
<td>Process image of the outputs</td>
</tr>
</tbody>
</table>

#### KL6021-0023

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_EnOceanRotarySwitch</td>
<td>State of the rotary-switch on the transmitting-module</td>
</tr>
<tr>
<td>E_EnOceanSensorType</td>
<td>Sensor type</td>
</tr>
<tr>
<td>ST_EnOceanReceivedData</td>
<td>Internal structure</td>
</tr>
</tbody>
</table>

#### KL6581/Hardware Types

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL6581_INPUT</td>
<td>Process image of the inputs</td>
</tr>
<tr>
<td>KL6581_OUTPUT</td>
<td>Process image of the outputs</td>
</tr>
</tbody>
</table>
KL6581

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR_EnOceanWindow [42]</td>
<td>State of the window</td>
</tr>
<tr>
<td>E_ENOCEAN_Org [39]</td>
<td>EnOcean telegram type</td>
</tr>
<tr>
<td>E_KL6581_Err [39]</td>
<td>Error messages</td>
</tr>
<tr>
<td>STR_EnOceanSwitch [42]</td>
<td>State of the buttons</td>
</tr>
<tr>
<td>STR_KL6581 [42]</td>
<td>Internal structure</td>
</tr>
<tr>
<td>STR_Teach [43]</td>
<td>Data structure manufacturer ID, type and function</td>
</tr>
<tr>
<td>STR_Teach_In [43]</td>
<td>Data structure manufacturer ID, type and profile</td>
</tr>
<tr>
<td>STREnOceanTurnSwitch [44]</td>
<td>Position of the rotary switch at the room control unit</td>
</tr>
</tbody>
</table>

4.2.1 KL6021-0023

Hardware types

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_EnOceanInData [38]</td>
<td>Process image of the KL6021-0023 inputs</td>
</tr>
<tr>
<td>ST_EnOceanOutData [38]</td>
<td>Process image of the KL6021-0023 outputs</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_EnOceanRotarySwitch [37]</td>
<td>Position of the rotary switch at the room control unit</td>
</tr>
<tr>
<td>E_EnOceanSensorType [36]</td>
<td>Sensor type</td>
</tr>
<tr>
<td>ST_EnOceanReceivedData [37]</td>
<td>Internal structure</td>
</tr>
</tbody>
</table>

4.2.1.1 Enums

4.2.1.1.1 E_EnOceanSensorType

Sensor type.

```plaintext
TYPE E_EnOceanSensorType :
{
  eEnOceanSensorTypePTM := 5,
  eEnOceanSensorTypeSTM1Byte := 6,
  eEnOceanSensorTypeSTM4Byte := 7,
  eEnOceanSensorTypeCTM := 8
}
END_TYPE
```

eEnOceanSensorTypePTM: PTM.
eEnOceanSensorTypeSTM1Byte: STM 1 Byte.
eEnOceanSensorTypeSTM4Byte: STM 4 Byte.
eEnOceanSensorTypeCTM: CTM.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
4.2.1.2 E_EnOceanRotarySwitch

E_EnOceanRotarySwitch describes the position of the rotary switch at the room control unit.

```plaintext
TYPE E_EnOceanRotarySwitch :
{  
eEnOceanRotarySwitchStep0 := 0,  
eEnOceanRotarySwitchStep1 := 1,  
eEnOceanRotarySwitchStep2 := 2,  
eEnOceanRotarySwitchStep3 := 3,  
eEnOceanRotarySwitchAuto := 4  
}
END_TYPE
```

eEnOceanRotarySwitchStep0: Switch in position “0”.
eEnOceanRotarySwitchStep1: Switch in position “1”.
eEnOceanRotarySwitchStep2: Switch in position “2”.
eEnOceanRotarySwitchStep3: Switch in position “3”.
eEnOceanRotarySwitchAuto: Switch in posture “Auto”.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.1.2 Structures

4.2.1.2.1 ST_EnOceanReceivedData

Internal structure.

This structure is used to link the block FB_EnOceanReceive() [14] with the receive blocks.

```plaintext
TYPE ST_EnOceanReceivedData :
  STRUCT  
  bReceived : BOOL;  
  nLength : BYTE;  
  eEnOceanSensorType : E_EnOceanSensorType;  
  nData : ARRAY[0..3] OF BYTE;  
  nStatus : BYTE;  
  nTransmitterId : UDINT;  
END_STRUCT
END_TYPE
```

bReceived: Data received.
nLength: Length.

eEnOceanSensorType: Sensor type (see E_EnOceanSensorType [36]).
nData: Data bytes.
nStatus: Status.
nTransmitterId: Transmitter ID.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
4.2.1.2.2  ST_EnOceanInData

Process image of the KL6021-0023 inputs.

Linked to the terminals in the System Manager.

```
TYPE ST_EnOceanInData : 
  STRUCT 
    nStatus : BYTE;
    nData  : ARRAY[0..10] OF BYTE;
  END_STRUCT
END_TYPE
```

**nStatus:** Status byte.

**nData:** 11 bytes for the input data.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.1.2.3  ST_EnOceanOutData

Process image of the KL6021-0023 outputs.

Linked to the terminals in the System Manager.

```
TYPE ST_EnOceanOutData : 
  STRUCT 
    nCtrl  : BYTE;
    nData  : ARRAY[0..10] OF BYTE;
  END_STRUCT
END_TYPE
```

**nCtrl:** Control byte.

**nData:** 11 bytes for the output data.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.2  KL6581

Hardware types

<table>
<thead>
<tr>
<th>Data types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL6581_Input</td>
<td>Process image of the KL6581 inputs</td>
</tr>
<tr>
<td>KL6581_Output</td>
<td>Process image of the KL6581 outputs</td>
</tr>
</tbody>
</table>
### Data types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR_EnOceanWindow [42]</td>
<td>State of the window</td>
</tr>
<tr>
<td>E_ENOCEAN_Org [39]</td>
<td>EnOcean telegram type</td>
</tr>
<tr>
<td>E_KL6581_Err [39]</td>
<td>Error messages</td>
</tr>
<tr>
<td>STR_EnOceanSwitch [42]</td>
<td>State of the buttons</td>
</tr>
<tr>
<td>STR_KL6581 [42]</td>
<td>Internal structure</td>
</tr>
<tr>
<td>STR_Teach [43]</td>
<td>Data structure manufacturer ID, type and function</td>
</tr>
<tr>
<td>STR_Teach_In [43]</td>
<td>Data structure manufacturer ID, type and profile</td>
</tr>
<tr>
<td>STREnOceanTurnSwitch [44]</td>
<td>Position of the rotary switch at the room control unit</td>
</tr>
</tbody>
</table>

#### 4.2.2.1 Enums

#### 4.2.2.1.1 E_ENOCEAN_ORG

EnOcean telegram type.

```lxml
TYPE E_ENOCEAN_Org :
  
  PTM_TELEGRAM := 5,
  STM_1BYTE_TELEGRAM := 6,
  STM_4BYTE_TELEGRAM := 7,
  CTM_TELEGRAM := 8,
  MODEM_TELEGRAM := 16#A,
  MODEM_ACK_TELEGRAM := 16#B
END_TYPE
```

**PTM_TELEGRAM**: PTM telegram.

**STM_1BYTE_TELEGRAM**: 1-byte telegram.

**STM_4BYTE_TELEGRAM**: 4-byte telegram.

**CTM_TELEGRAM**: CTM telegram.

**MODEM_TELEGRAM**: Modem telegram.

**MODEM_ACK_TELEGRAM**: Modem telegram with acknowledgement.

#### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

#### 4.2.2.1.2 E_KL6581_Err

Error messages.

```lxml
TYPE E_KL6581_Err :
  
  NO_ERROR := 16#0,
  KL6581_WrongTerminal := 16#A,
  KL6581_WatchdogError := 16#10,
  KL6581_NoComWithKL6581 := 16#11,
  KL6581_idx_number_not_OK := 16#12,
  KL6581_Switch_to_Stopp := 16#13,
  KL6581_not_ready := 16#14,
  KL6581_No_KL6853_Found := 16#15,
  KL6581_TransmissionError := 16#16
END_TYPE
```
NO_ERROR: No error at the block.

KL6581_WrongTerminal: Incorrect terminal connected.

KL6581_WatchdogError: Timeout during initialization of block "FB_KL6581".

KL6581_NoComWithKL6581: This message usually means that there is no connection to the terminal. Terminal linked to the variables in the System Manager? Terminal plugged in incorrectly? Everything corrected, everything translated and re-read into the System Manager?

KL6581_idx_number_not_OK: The input variable nIdx of block FB_KL6581() \[23\] is greater than 64.

KL6581_Switch_to_Stopp: The terminal has exited the data exchange with the KL6583. No EnOcean data was sent or received.

KL6581_not_ready: Internal message for the function blocks connected to the FB_KL6581().

KL6581_No_KL6853_Found: No KL6583 is connected to the KL6581, or the communication does not exist.

KL6581_TransmissionError: Data could not sent; check the address of the KL6583, or KL6583 not ready for operation.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.2.2 Structures

4.2.2.2.1 KL6581_INPUT

Process image of the KL6581 inputs.

Linked to the terminal in the System Manager.

```plaintext
TYPE KL6581_Input :
STRUCT
  nStatus : BYTE;
  CNODE  : BYTE;
  ORG    : BYTE;
  DB0    : BYTE;
  DB1    : BYTE;
  DB2    : BYTE;
  DB3    : BYTE;
  ID0    : BYTE;
  ID1    : BYTE;
  ID2    : BYTE;
  ID3    : BYTE;
  STATUS : BYTE;
END_STRUCT
END_TYPE
```

nStatus: Status byte.

CNODE: Data byte.

ORG: Data byte.

DB0: Data byte.

DB1: Data byte.

DB2: Data byte.

DB3: Data byte.

ID0: Data byte.
ID1: Data byte.
ID2: Data byte.
ID3: Data byte.

**STATUS:** Data byte.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.2.2.2 **KL6581_Output**

Process image of the KL6581 outputs.

Linked to the terminal in the System Manager.

```plaintext
TYPE KL6581_Output :
  STRUCT
    nControl  : BYTE;
    CNODE     : BYTE;
    ORG       : BYTE;
    DB0       : BYTE;
    DB1       : BYTE;
    DB2       : BYTE;
    DB3       : BYTE;
    ID0       : BYTE;
    ID1       : BYTE;
    ID2       : BYTE;
    ID3       : BYTE;
    STATUS    : BYTE;
  END_STRUCT
END_TYPE
```

**nControl:** Control byte.

**CNODE:** Data byte.

**ORG:** Data byte.

**DB0:** Data byte.

**DB1:** Data byte.

**DB2:** Data byte.

**DB3:** Data byte.

**ID0:** Data byte.

**ID1:** Data byte.

**ID2:** Data byte.

**ID3:** Data byte.

**STATUS:** Data byte.

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>
4.2.2.2.3  ar_EnOceanWindow

This structure indicates the state of the window.

```plaintext
TYPE AR_EnOceanWindow :
  STRUCT
    bUp   : BOOL;
    bOpen : BOOL;
    bClose: BOOL;
  END_STRUCT
END_TYPE
```

- **bUp**: The window is tilted.
- **bOpen**: The window is open.
- **bClose**: The window is closed.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.2.2.4  str_EnOceanSwitch

State of the buttons.

```plaintext
TYPE STR_EnOceanSwitch :
  STRUCT
    bT1_ON  : BOOL;
    bT1_OFF : BOOL;
    bT2_ON  : BOOL;
    bT2_OFF : BOOL;
    bT3_ON  : BOOL;
    bT3_OFF : BOOL;
    bT4_ON  : BOOL;
    bT4_OFF : BOOL;
  END_STRUCT
END_TYPE
```

- **bT1_ON**: Button 1 on.
- **bT1_OFF**: Button 1 off.
- **bT2_ON**: Button 2 on.
- **bT2_OFF**: Button 2 off.
- **bT3_ON**: Button 3 on.
- **bT3_OFF**: Button 3 off.
- **bT4_ON**: Button 4 on.
- **bT4_OFF**: Button 4 off.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.2.2.5  str_KL6581

Internal structure.

This structure is used to link the block `FB_KL6581()` [23] with the send/receive blocks.
TYPE STR_KL6581 :
STRUCT
  by_Status : BYTE;
  by_Node : BYTE;
  by_ORG : BYTE;
  ar_DB : ARRAY[0..3] OF BYTE;
  _Dummy : BYTE;
  dw_ID : DWORD;
  ptData : PVOID;
  iErrorId : E_KL6581_Err;
  by_STATE : BYTE;
  bError : BOOL;
  idx : USINT;
END_STRUCT
END_TYPE

by_Status: Status.

by_Node: Node number of the KL6583 that has received the EnOcean telegram.

by_ORG: EnOcean telegram type.

ar_DB: Data bytes.

 Dummy: Placeholder, no other purpose.

dw_ID: Transmitter ID.

ptData: Pointer.

iErrorId: Describes the error type (see E_KL6581_Err [39]).

by_STATE: State.

bError: this output goes TRUE as soon as an error occurs. This error is described via the iErrorId variable.

idx: Index.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.2.2.6 STR_Teach

Data structure manufacturer ID, type and function.

STRUCT
  nManufacturerID : WORD;
  nTYPE : BYTE;
  nFunc : BYTE;
END_STRUCT
END_TYPE

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.32</td>
<td>Tc2_EnOcean from v3.4.6.0</td>
</tr>
</tbody>
</table>

4.2.2.2.7 str_Teach_In

Data structure manufacturer ID, type and profile.

TYPE STR_Teach_In :
STRUCT
  nManufacturerID : WORD;
  nTYPE : BYTE;
END_STRUCT
END_TYPE
nProfile: Profile.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.2.2.2.8 STREnOceanTurnSwitch

STREnOceanTurnSwitch describes the position of the rotary switch at the room control unit.

```c
TYPE STREnOceanTurnSwitch :
STRUCT
  bStageAuto : BOOL;
  bStage_0   : BOOL;
  bStage_1   : BOOL;
  bStage_2   : BOOL;
  bStage_3   : BOOL;
END_STRUCT
END_TYPE
```

- **bStageAuto**: Switch in posture "Auto".
- **bStage_0**: Switch in position "0".
- **bStage_1**: Switch in position "1".
- **bStage_2**: Switch in position "2".
- **bStage_3**: Switch in position "3".

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>required TC3 PLC library</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4020.14</td>
<td>Tc2_EnOcean from v3.3.5.0</td>
</tr>
</tbody>
</table>

4.3 Integration into TwinCAT

4.3.1 KL6581 with CX5120

This example describes how a simple PLC program for EnOcean can be written in TwinCAT and how it is linked with the hardware. The task is to receive four probe signals of an EnOcean wireless switch module.

Example: https://infosys.beckhoff.com/content/1033/tcpclib_tc2_enocean/Resources/zip/6200373771.zip

Hardware

Setting up the components

The following hardware is required:

- 1x CX5120 Embedded PC
- 1x KL6581 EnOcean master terminal
- 1x KL6583-0000 EnOcean transmitter and receiver
1x KL9010 end terminal

Set up the hardware and the EnOcean components as described in the associated documentation. This example assumes that the ID of the wireless switch module is known.

**Software**

**Creation of the PLC program**

Create a new "TwinCAT XAE project" and a "Standard PLC project". Add the library Tc2_EnOcean in the PLC project under "References". Generate a global variable list with the name GVL_EnOcean and create the following variables:

```plc
VAR_GLOBAL
    stKL6581Input AT %I* : KL6581_Input;
    stKL6581Output AT %Q* : KL6581_Output;
    stKL6581 : STR_KL6581;
END_VAR
```

**stKL6581Input**: Input variable for the EnOcean terminal (see KL6581_INPUT [40]).

**stKL6581Output**: Output variable for the EnOcean terminal (see KL6581_Output [41]).

**stKL6581**: Required for communication with EnOcean (see str_KL6581 [42]).

All EnOcean function blocks must be called in the same task.

Create a MAIN program (CFC) in which the function blocks FB_KL6581 [23] and FB_Rec_RPS_Switch [25] are called. Ensure that the communication block is linked with the structures stKL6581Input, stKL6581Output and stKL6581.

The input dw_ID of the receive block is linked with the local variable dwId (ID from wireless switch module) and str_KL6581 with the global variable stKL6581.

Go to the task configuration and give the task a lower interval time.

Further conditions can be found in the description of the function block FB_KL6581 [23].
I/O configuration

Select the CX as target system and initiate a search for its hardware. In the project instance within the PLC section, you can see that the input and output variables are assigned to the task.

Now link the global variables with the inputs and outputs of the Bus Terminals.

The linking of the EnOcean variables is described in detail below.

Right-click the structure stKL6581Input and select “Change link”.

Under “I/O Configuration” select the terminal, select “All Types” and “Continuous”, then select “Status” to “InputStatus” with the left mouse button and the >SHIFT< key. Then press "OK".
You can now check the connection. To do this go onto the KL6581 and open it. All terminal data should now show an arrow. If that is the case, then proceed in exactly the same way with the outputs.
5 Appendix

5.1 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: https://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

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

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
web: https://www.beckhoff.com