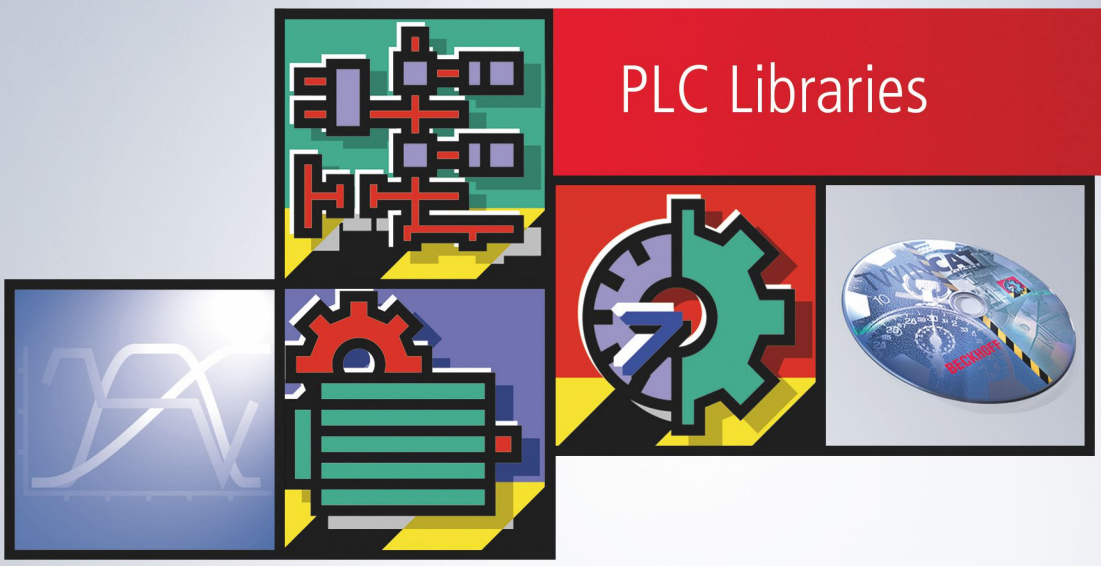
BECKHOFF New Automation Technology

Manual | EN

TX1200

TwinCAT 2 | PLC Library: TcEnOcean



2023-08-31 | Version: 1.3

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

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

Disclaimer

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:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

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

Copyright

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are 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 technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

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

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

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

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

2 Introduction

The TcEnOcean library is a TwinCAT PLC library for data exchange with EnOcean devices.

This library is to be used only in conjunction with a EnOcean bus terminal KL6021-0023 or a EnOcean master terminal KL6581.

3 Target groups

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

- TwinCAT PLC-Control
- TwinCAT System Manager
- PCs and networks
- 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.

4 EnOcean

Technology

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.

- License-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

Protocol	Description	Length
ORG	Telegram type	1 bytes
DB_3	Data byte 3	1 byte
DB_2	Data byte 2	1 byte
DB_1	Data byte 1	1 byte
DB_0	Data byte 0	1 byte
ID_3	Transmitter ID byte 3	1 byte
ID_2	Transmitter ID byte 2	1 byte
ID_1	Transmitter ID byte 1	1 byte
ID_0	Transmitter ID byte 0	1 byte
STATUS	Information status	1 byte

4.1 Range planning

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

Attenuation of different materials

Material	Attenuation
Wood, plaster, uncoated glass (without metal)	0...10 %
Brick, chipboard	5...35 %
Concrete with iron reinforcement	10...90 %
Metal, aluminum cladding	90..100 %

Range

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

Placement of the KL6583 module

The EnOcean transmitter and receiver KL6583-0000 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

4.2 Approval of EnOcean wireless technology

Check that operation is permitted in your country

- **KL6583-0000**: **European Union** and **Switzerland**
- **KL6583-0100**: **USA** and **Canada**

KL6583-0100 for USA and Canada

Contains IC: 5731A-TCM320C

Contains FCC ID: SZV-TCM320C

This device complies with Part 15 of the FCC regulations.
Operation is subject to the following conditions:
(i.) this device may not cause harmful interference, and
(ii.) this device must accept any interference received, including interference that may cause undesired operation
.

5 Integration into TwinCAT

5.1 KL6581 - Linking to the System Manager

1. Link the PLC program and click with the right mouse button on the data structure.

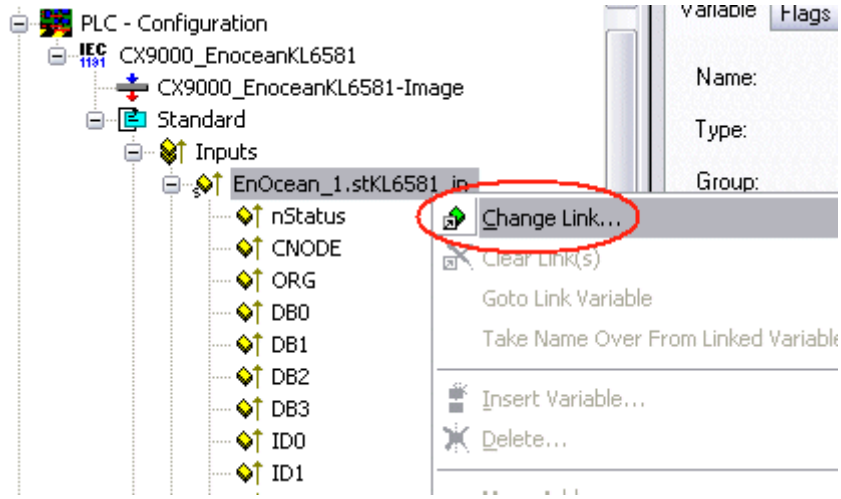


Figure 1

2. Select "All Types" and "Continuous" (see Figure 2).

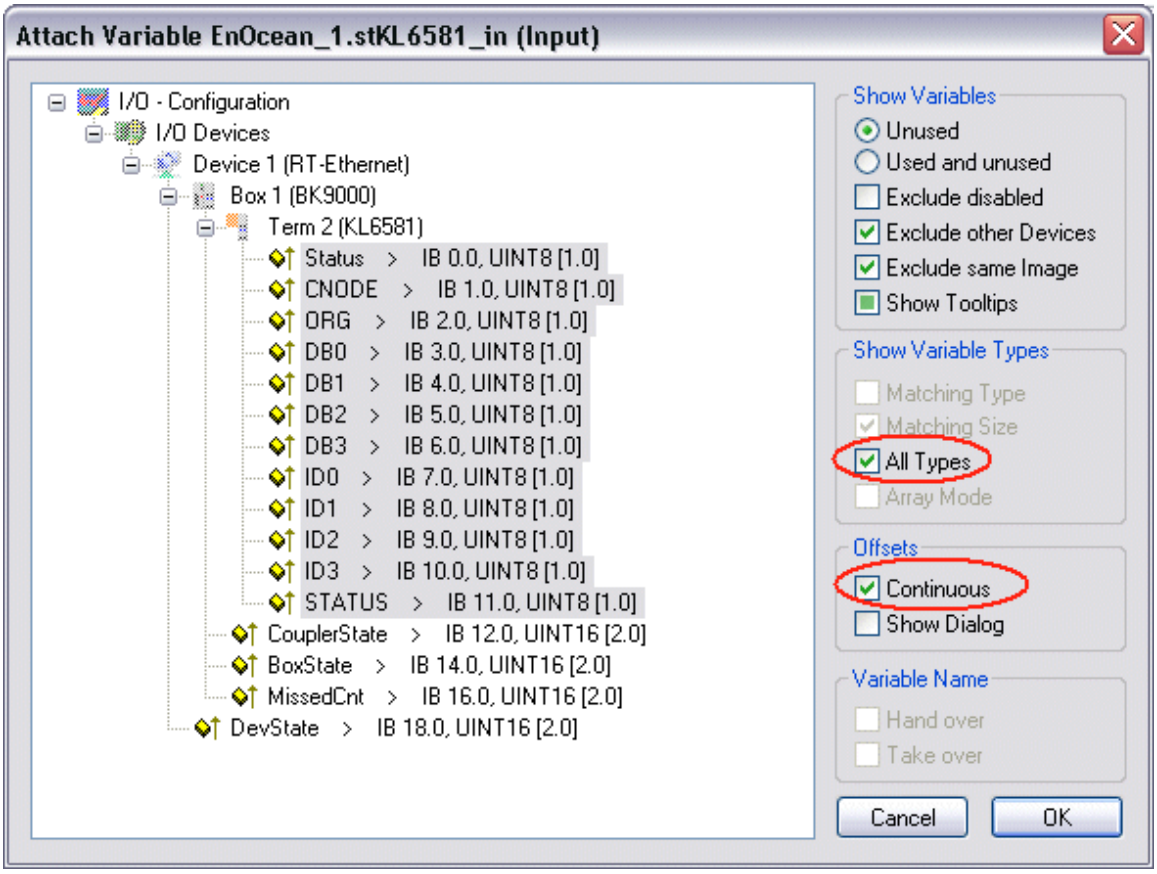


Figure 2

3. Click with the mouse on the first variable of the EnOcean master terminal KL6581 ‘Status’. Then press the <SHIFT> key, and hold it down.

4. Move the mouse pointer over the last variable of the KL6581 ‘STATUS’ and click again with the left mouse button.
5. Now release the <SHIFT> key again. All the terminal's data should now be highlighted (see Figure 2).
6. Press button OK.
7. Check the links
   To do this go onto the KL6581 and open it. All the terminal's data should now be marked by a small arrow (see Figure 3). If that is the case, then proceed in exactly the same way with the outputs.

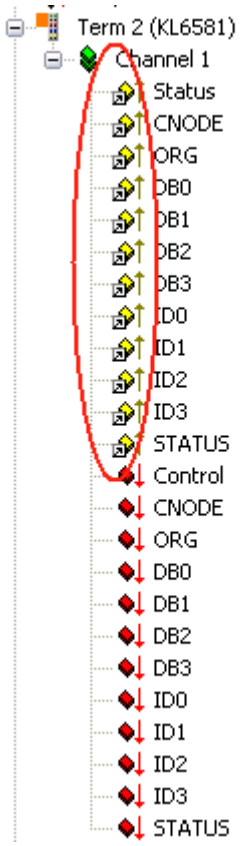


Figure 3

5.2 KL6021-0023 - Linking to the System Manager

1. Link the PLC program and click with the right mouse button on the data structure.

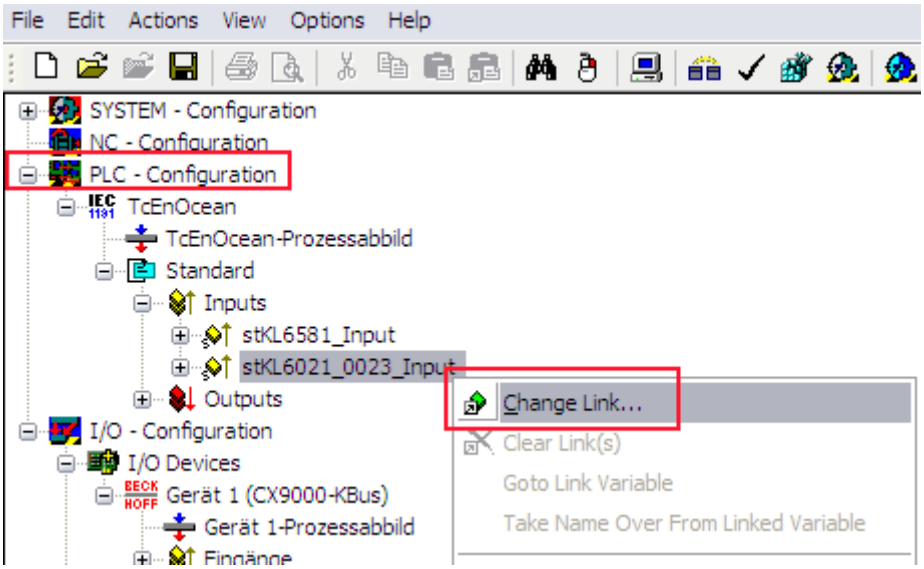


Figure 1

2. Select "All Types" and "Continuous" (see Figure 2).

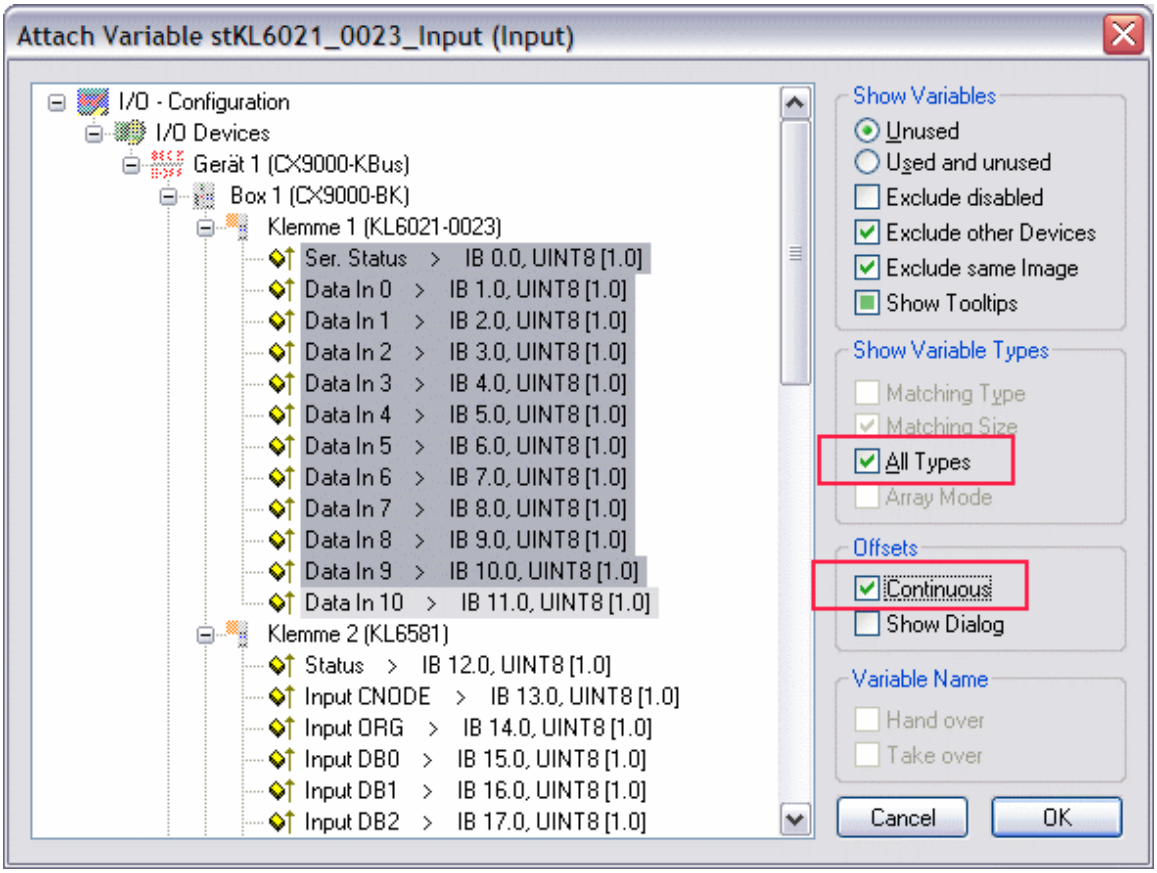


Figure 2

3. Click with the mouse on the first variable of the EnOcean bus terminal KL6021-0023 ‘Ser.Status’. Then press the <SHIFT> key, and hold it down.
4. Move the mouse pointer over the last variable of the KL6021-0023 ‘Data In 10' and click again with the left mouse button.
5. Now release the <SHIFT> key again. All the terminal's data should now be highlighted (see Figure 2).
6. Press button "OK"

7. Check the links
   To do this go onto the EnOcean master terminal KL6581 and open it. All the terminal's data should now be marked by a small arrow (see Figure 3). If that is the case, then proceed in exactly the same way with the outputs.

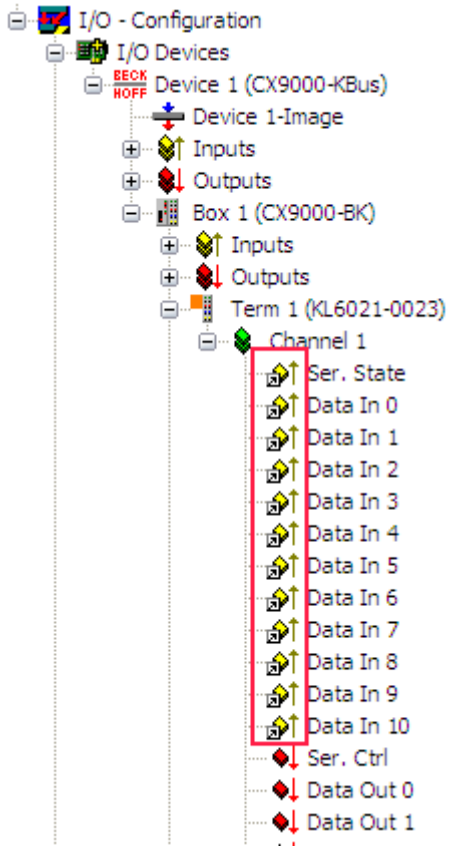


Figure 3

5.3 Integration in TwinCAT (CX9020)

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 the four probe signals of an EnOcean wireless switch module.

https://infosys.beckhoff.com/content/1033/tcplclibenocean/Resources/11985702027/.zip https://infosys.beckhoff.com/content/1033/tcplclibenocean/Resources/11985702027/.zip

Hardware

Setting up the components

The following hardware is required:

- 1x Embedded PC CX9020
- 1x EnOcean master terminal KL6581
- 1x EnOcean transmitter and receiver KL6583-0000
- 1x end terminal KL9010

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 PLC project for PC-based systems (ARM) and add the *TcEnOcean.lib* library.

Next, generate the following global variables:

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

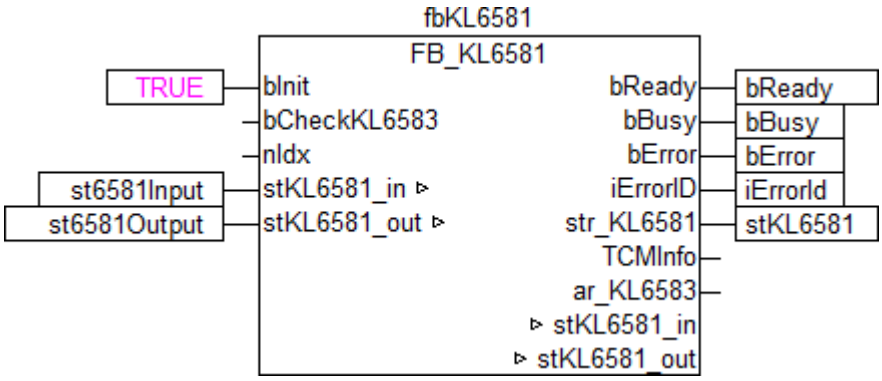
stKL6581Input : Input variable [▶ 46] for the EnOcean terminal.

stKL6581Output : Output variable [▶ 46] for the EnOcean terminal.

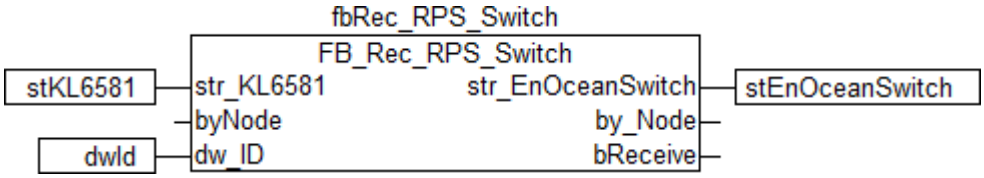
stKL6581 : Required for the communication [▶ 47] with EnOcean.

All blocks with EnOcean must be executed in one task.

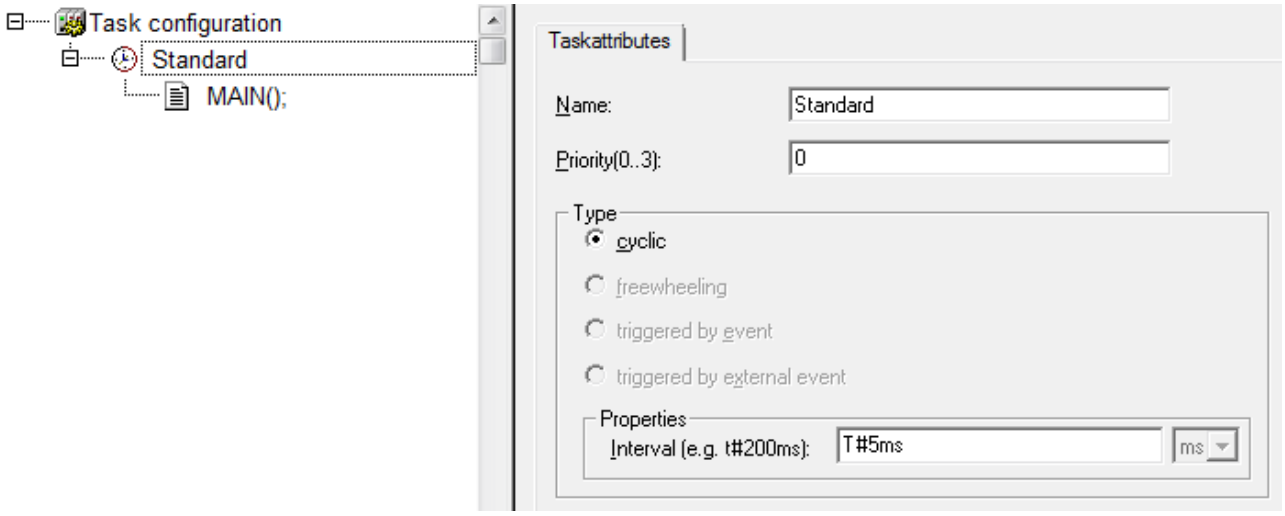
Therefore, create a MAIN program (CFC) in which the FB_KL6581() [▶ 24] and FB_Rec_RPS_Switch() [▶ 26] function blocks are called. Make sure to link the communication block with *stKL6581Input*, *stKL6581Output* and *stKL6581*.



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. More detailed information can be found in the FB_KL6581() [▶ 24] block description.

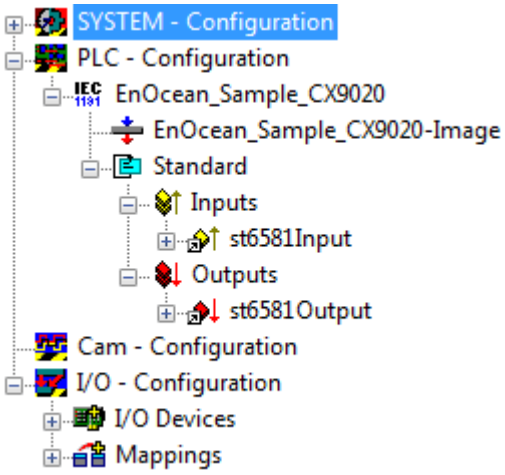


Load the project to the CX as the boot project and save it.

Configuration in the System Manager

Create a new System Manager project, select the CX as the target system, and search for the associated hardware.

Add the PLC program created above under PLC configuration.



Now link the global variables of the PLC program with the Bus Terminal inputs and outputs, create the allocations, and activate the configuration. Then start the device in run mode.
Your CX is now ready for use.

After pressing the button on the wireless switch module, the received radio signals can be viewed in the structure *stEnOceanSwitch*.

5.4 Integration into TwinCAT (BC9191)

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 the four probe signals of an EnOcean wireless switch module.

https://infosys.beckhoff.com/content/1033/tcplclibenocean/Resources/11985703435/.zip https://infosys.beckhoff.com/content/1033/tcplclibenocean/Resources/11985703435/.zip

Hardware

Setting up the components

The following hardware is required:

- 1x Bus Terminal Controller BC9191
- 1x EnOcean transmitter and receiver KL6583-0000

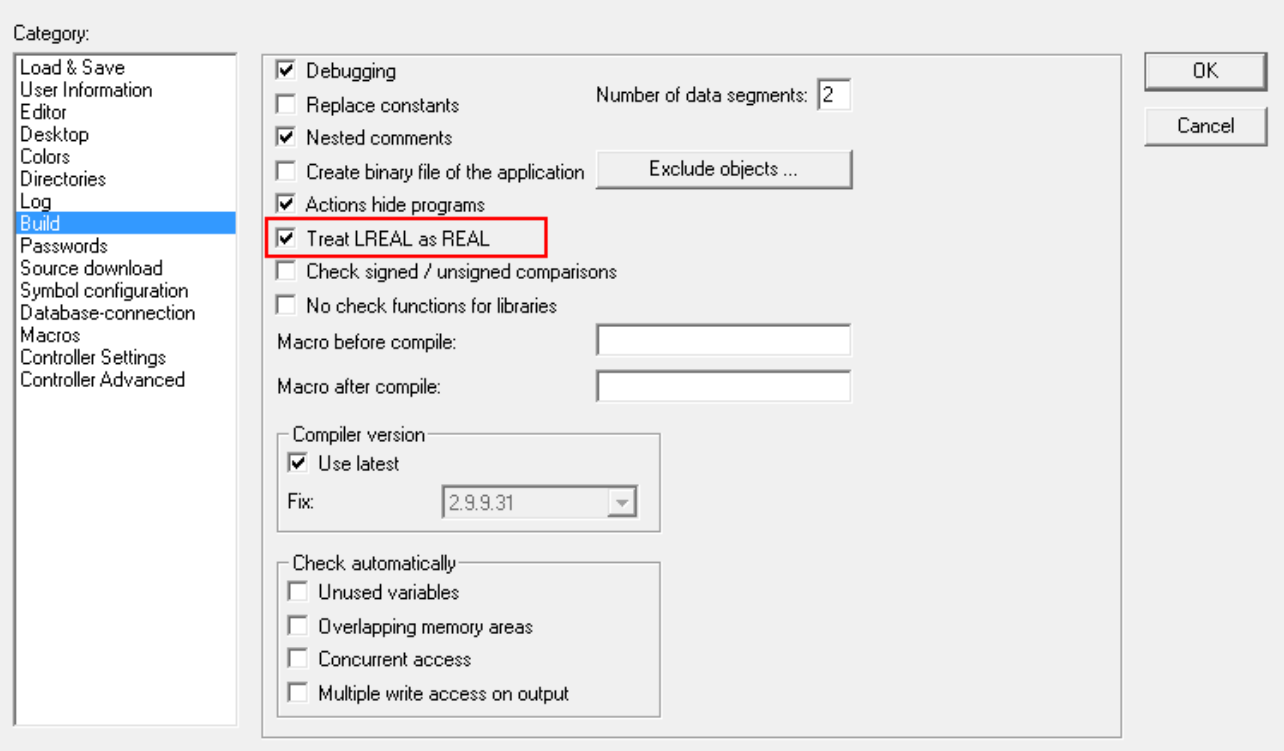
Set up the hardware and the EnOcean components as described in the associated documentation.

This example assumes that the ID of the wireless switch module is known.

Software

Creation of the PLC program

Create a new PLC project for BC-based systems (BCxx50 via AMS) and add the libraries *TcEnOcean.lbx* and *TcSystemBCxx50.lbx*. Then navigate to *Project→Options... →Build* and select *TreatLREAL as REAL*.



Next, generate the following global variables:

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

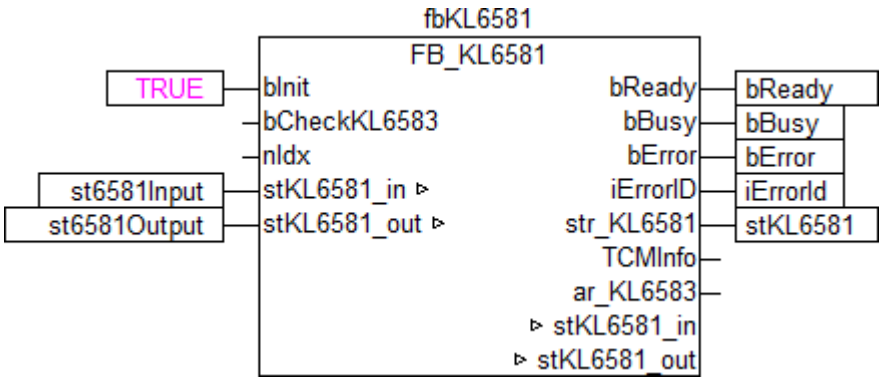
stKL6581Input: Input variable [▶ 46] for EnOcean.

stKL6581Output: Output variable [▶ 46] for EnOcean.

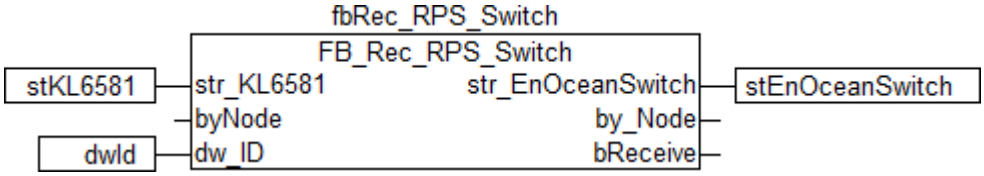
stKL6581: Required for the communication [▶ 47] with EnOcean.

All blocks with EnOcean must be executed in one task.

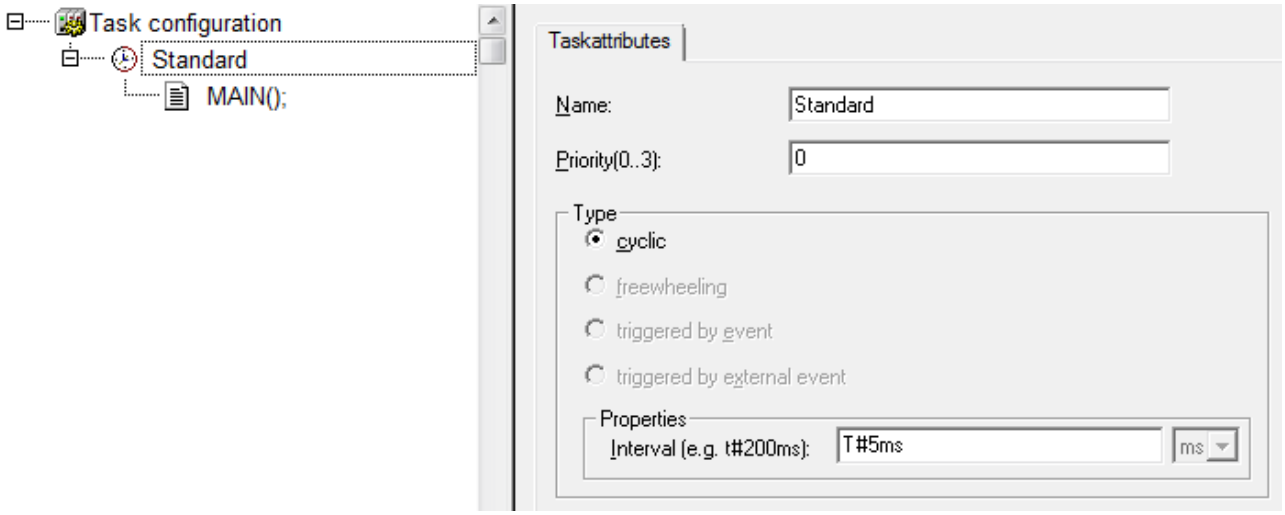
Therefore, create a MAIN program (CFC) in which the FB_KL6581() [▶ 24] and FB_Rec_RPS_Switch() [▶ 26] function blocks are called. Make sure to link the communication block with *stKL6581Input*, *stKL6581Output* and *stKL6581*.



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. More detailed information can be found in the FB_KL6581() [▶ 24] block description.

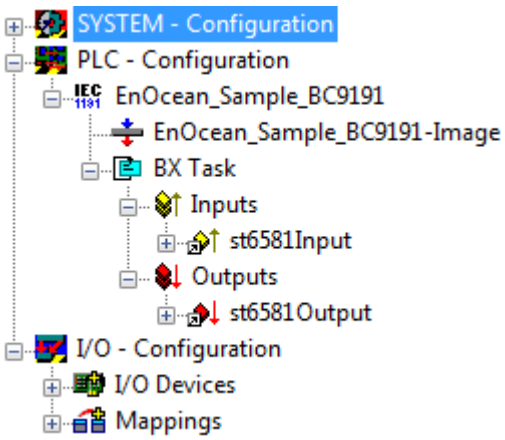


Now load the project as a boot project to the BC and save it.

Configuration in the System Manager

Create a new System Manager project, select the BC as the target system, and search for the associated hardware.

Add the PLC program created above under PLC configuration.



Now link the global variables of the PLC program with the inputs and outputs of the EnOcean master terminal KL6581 from BC9191, create the allocations, and activate the configuration. Then start the device in run mode.

Your BC is now ready for use.

After pressing the button on the wireless switch module, the received radio signals can be viewed in the structure *stEnOceanSwitch*.

6 Programming

Content
General information [▶ 22]
KL6581 - Linking to the System Manager [▶ 12]
KL6021-0023 - Linking to the System Manager [▶ 13]

KL6581

POUs	Description
FB_KL6581 [▶ 24]	Main block for the communication with the EnOcean master terminal KL6581

Function

POUs	Description
F_Byte_to_Temp [▶ 33]	Converts a byte into a REAL value.
F_Byte_to_TurnSwitch [▶ 34]	Converts a hand switch, for example for ventilation (AUTO, 0, I, II, III), into a BOOL array.

Other

POUs	Description
FB_EnOcean_Search [▶ 31]	Block recognizes all EnOcean devices within its range and displays them.
FB_Rec_Teach_In [▶ 32]	This block indicates if the LRN bit in an EnOcean telegram is set, independent of its EnOcean ID.
FB_Rec_Teach_In_Ex [▶ 32]	This block indicates if the LRN bit in an EnOcean telegram is set, independent of its EnOcean ID.

Read

POUs	Description
FB_Rec_1BS [▶ 26]	Receives data with ORG telegram 6.Typical EnOcean device: Window contact.
FB_Rec_Generic [▶ 25]	Receives all types of EnOcean telegrams.
FB_Rec_RPS_Switch [▶ 26]	Receives data with ORG telegram 5. Typical EnOcean device: Buttons.
FB_Rec_RPS_Window_Handle [▶ 27]	Receives data with ORG telegram 5.Typical EnOcean device: Window handle.

Send

POUs	Description
FB_Send_Generic [▶ 28]	All kinds of EnOcean data telegrams can be sent with this block.
FB_Send_4BS [▶ 29]	Sends EnOcean telegrams in the 4BS format.
FB_Send_RPS_Switch [▶ 29]	Sends EnOcean telegrams in the format of a button.
FB_Send_RPS_SwitchAuto [▶ 30]	This function block sends data such as those from a switch.

Enums

Data types	Description
E_EnOcean_Org [▶ 45]	Type of EnOcean telegram.
E_KL6581_Err [▶ 45]	Error massage.

Structs

Data types	Description
KL6581_Input [▶ 46]	Process image of the inputs.
KL6581_Output [▶ 46]	Process image of the outputs.
AR_EnOceanWindow [▶ 49]	State of the window.
STR_EnOceanSwitch [▶ 47]	State of the buttons.
STR_KL6581 [▶ 47]	Internal structure.
STR_Teach [▶ 48]	Data structure - manufacturer ID, type and function.
STR_Teach_In [▶ 48]	Data structure - manufacturer ID, type and profile.
STREnOceanTurnSwitch [▶ 49]	State of the rotary-switch on the transmitting-module.

KL6021-0023

POUs	Description
FB_EnOceanReceive [▶ 35]	Communication with EnOcean bus terminal KL6021-0023

Read

POUs	Description
FB_EnOceanPTM100 [▶ 36]	Receives the signals of a PTM100 module.
FB_EnOceanPTM200 [▶ 38]	Receives the signals of a PTM200 module.
FB_EnOceanSTM100 [▶ 40]	Receives the signals of a STM100 module (obsolete).
FB_EnOceanSTM100Generic [▶ 42]	Receives the signals of a STM100 module.
FB_EnOceanSTM250 [▶ 43]	Receives the signals of a STM250 module.

Enums

Data types	Description
E_EnOceanRotarySwitch [▶ 49]	State of the rotary-switch on the transmitting-module.
E_EnOceanSensorType [▶ 50]	Sensor type.

Structs

Data types	Description
ST_EnOceanInData [▶ 50]	Process image of the inputs.
ST_EnOceanOutData [▶ 50]	Process image of the outputs.
ST_EnOceanReceivedData [▶ 50]	Internal structure.

6.1 General information

Installation

Beginning with TwinCAT 2.11 Build 2229 (R3 and x64 Engineering), the libraries "TcEnOcean.lib/.lb6/.lbx" will be installed automatically.

Name of the library

This library replaces the "TcKL6581.lib/.lb6./.lbx". Only the name of the libraries has changed. The modules are still compatible.

Hardware documentation in Beckhoff Information System: KL6021-0023, KL6023 - EnOcean bus terminal

Hardware documentation in Beckhoff Information System: KL6581, KL6583 - EnOcean bus terminal

Further libraries are required

For PC systems (x86) and Embedded-PCs (CXxxxx):

- Standard.lib
- TcBase.lib
- TcSystem.lib

For Bus Terminal Controller of BCxx00 series:

- Standard.lb6
- PlcHelperBC.lb6

For Bus Terminal Controller of BCxx50, BCxx20 and BC9191 series:

- Standard.lbx
- TcBaseBCxx50.lbx
- TcSystemBCxx50.lbx

For Bus Terminal Controller of BXxx00 series:

- Standard.lbx
- TcBaseBX.lbx
- TcSystemBX.lbx

Memory usage

By linking the library PLC program memory is already consumed. Depending on the application program the remaining memory can not be sufficient.

6.2 KL6581

POUs	Description
FB_KL6581 [▶ 24]	Main block for the communication with the EnOcean master terminal KL6581

Function

POUs	Description
F_Byte_to_Temp [▶ 33]	Converts a byte into a REAL value.
F_Byte_to_TurnSwitch [▶ 34]	Converts a hand switch, for example for ventilation (AUTO, 0, I, II, III), into a BOOL array.

Other

POUs	Description
FB_EnOcean_Search [▶ 31]	Block recognizes all EnOcean devices within its range and displays them.
FB_Rec_Teach_In [▶ 32]	This block indicates if the LRN bit in an EnOcean telegram is set, independent of its EnOcean ID.
FB_Rec_Teach_In_Ex [▶ 32]	This block indicates if the LRN bit in an EnOcean telegram is set, independent of its EnOcean ID.

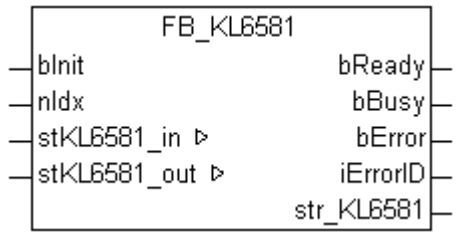
Read

POUs	Description
FB_Rec_1BS [▶ 26]	Receives data with ORG telegram 6.Typical EnOcean device: Window contact.
FB_Rec_Generic [▶ 25]	Receives all types of EnOcean telegrams.
FB_Rec_RPS_Switch [▶ 26]	Receives data with ORG telegram 5. Typical EnOcean device: Buttons.
FB_Rec_RPS_Window_Handle [▶ 27]	Receives data with ORG telegram 5.Typical EnOcean device: Window handle.

Send

POUs	Description
FB_Send_Generic [▶ 28]	All kinds of EnOcean data telegrams can be sent with this block.
FB_Send_4BS [▶ 29]	Sends EnOcean telegrams in the 4BS format.
FB_Send_RPS_Switch [▶ 29]	Sends EnOcean telegrams in the format of a button.
FB_Send_RPS_SwitchAuto [▶ 30]	This function block sends data such as those from a switch.

6.2.1 FB_KL6581



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

Restrictions

- Only one call per instance
- Call must be made once per PLC cycle
- Instance must be called in the same PLC task as the send and receive blocks assigned to it
- Maximum 64 instances per PLC project allowed

VAR_INPUT

```
bInit          : BOOL;
nIdx           : USINT := 1;
```

bInit: Activates the block that configures the KL6581 first and then activates the data exchange, if the input remains TRUE.

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

VAR_OUTPUT

```
bReady         : BOOL;
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : E_KL6581_Err;
str_KL6581     : STR_KL6581;
```

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

bBusy: The block is active.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *iErrorID*.

iErrorID: Type of error (see E_KL6581_Err [▶ 45]).

str_KL6581: Data structure that is connected to the send and receive function blocks (see STR_KL6581 [▶ 47]).

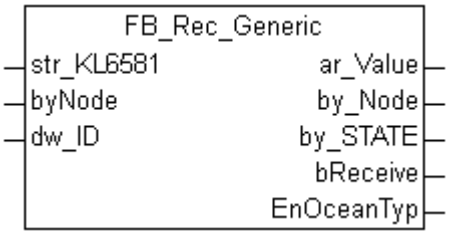
VAR_IN_OUT

```
stKL6581_in       : KL6581_Input;
stKL6581_out      : KL6581_Output;
```

stKL6581_in: Is linked in System Manager to the input addresses of the KL6581 (see KL6581_Input [▶ 46]).

stKL6581_out: Is linked in System Manager to the output addresses of the KL6581 (see KL6581_Output [▶ 46]).

6.2.2 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;
```

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

byNode: Filter - if the value is zero the EnOcean telegrams from all EnOcean transmitter and receiver KL6583-0000 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_Value     : ARRAY [0..3] OF BYTE;
by_Node      : BYTE;
by_STATE     : BYTE;
bReceive     : BOOL := TRUE;
EnOceanTyp   : E_EnOcean_Org;
```

ar_Value: 4-byte EnOcean data.

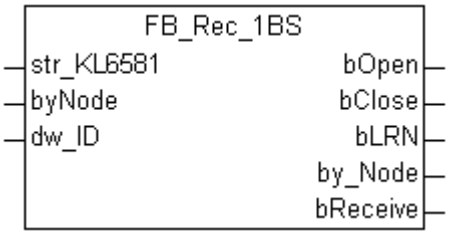
by_Node: Node number of the KL6583 that has received the EnOcean telegram.

by_STATE: EnOcean STATUS field.

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

EnOceanTyp: EnOcean ORG field.

6.2.3 FB_Rec_1BS



This function block receives data that were received via EnOcean. This block is used, for example, for the connection of window contacts.

ORG FIELD 6

VAR_INPUT

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

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

byNode: Filter - if the value is zero the EnOcean telegrams from all EnOcean transmitter and receiver KL6583-0000 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.

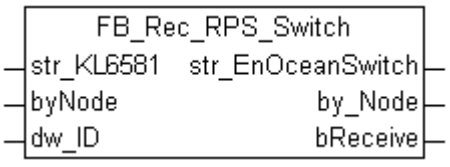
bClose: Contact closed.

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.

6.2.4 FB_Rec_RPS_Switch



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: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

byNode: Filter - if the value is zero the EnOcean telegrams from all EnOcean transmitter and receiver KL6583-0000 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

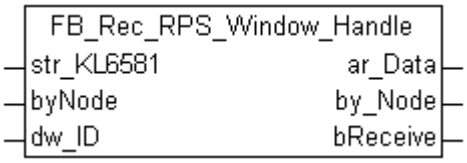
```
str_EnOceanSwitch   : STR_EnOceanSwitch;
by_Node             : BYTE;
bReceive            : BOOL := TRUE;
```

str_EnOceanSwitch: Data from the switch (see STR_EnOceanSwitch [▶ 47]).

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.

6.2.5 FB_Rec_RPS_Window_Handle



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: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

byNode: Filter - if the value is zero the EnOcean telegrams from all EnOcean transmitter and receiver KL6583-0000 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

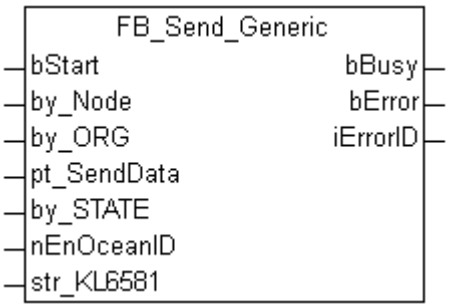
```
ar_Data      : AR_EnOceanWindow;
by_Node      : BYTE;
bReceive     : BOOL := TRUE;
```

dw_ID: Data from the window handle (see AR_EnOceanWindow [▶ 49]).

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.

6.2.6 FB_Send_Generic



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

```
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 EnOcean transmitter and receiver KL6583-0000 to which the telegram is to be sent (valid values: 1 to 8).

by_ORG: ORG field of the EnOcean telegram.

pt_SendData: Pointer to the data which are to be sent; the pointer address is determined with ADR. The pointer must point to a 4-byte variable.

by_STATE: EnOcean STATE, can be changed by the TCM module.

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

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

VAR_OUTPUT

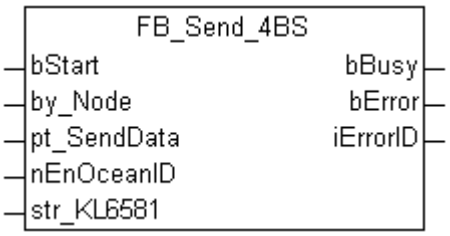
```
bBusy        : BOOL;
bError       : BOOL;
iErrorID     : E_KL6581_Err;
```

bBusy: Block is is busy, no new data can be sent at the moment.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *iErrorID*.

iErrorID: Type of error (see E_KL6581_Err [▶ 45]).

6.2.7 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 EnOcean transmitter and receiver KL6583-0000 to which the telegram is to be sent (valid values: 1 to 8).

pt_SendData: Pointer to the data which are to be sent; the pointer address is determined with ADR. The pointer must point to a 4-byte variable.

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

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

VAR_OUTPUT

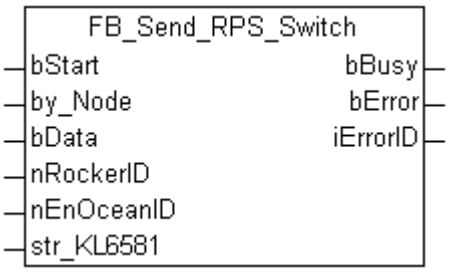
```
bBusy        : BOOL;
bError       : BOOL;
iErrorID     : E_KL6581_Err;
```

bBusy: Block is is busy, no new data can be sent at the moment.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *iErrorID*.

iErrorID: Type of error (see E_KL6581_Err [▶ 45]).

6.2.8 FB_Send_RPS_Switch



This module sends EnOcean-telegrams in the EnOcean-pushbutton-format. With a rising edge at *bStart*, the entered value at *bData* will be sent. After expiration of tSwitchDelay the signal “pushbutton” released will be sent. In order to simulate an activation of a pushbutton, the module has normally to be started twice: once with *bData* = TRUE and another time with *bData* = FALSE. For an easier handling it is possible to use the module FB_Send_RPS_SwitchAuto() [▶ 30].

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 EnOcean transmitter and receiver KL6583-0000 to which the telegram is to be sent (valid values: 1 to 8).

bData: Value to be transmitted.

nRockerID: Button number, valid values 0 to 3.

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

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

VAR_OUTPUT

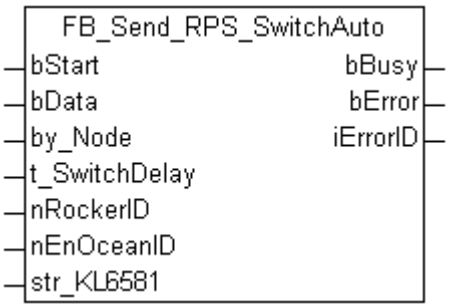
```
bBusy       : BOOL;
bError      : BOOL;
iErrorID    : E_KL6581_Err;
```

bBusy: Block is is busy, no new data can be sent at the moment.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *iErrorID*.

iErrorID: Type of error (see E_KL6581_Err [▶ 45]).

6.2.9 FB_Send_RPS_SwitchAuto



This module sends EnOcean-telegrams in the EnOcean-pushbutton-format. With a rising edge at *bStart*, the entered value at *bData* will be sent. After expiration of *tSwitchDelay* the signal “pushbutton” released will be sent. After expiration of *tSwitchDelay* the signal “pushbutton” released will be sent.

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 EnOcean transmitter and receiver KL6583-0000 to which the telegram is to be sent (valid values: 1 to 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 to 127 is added to the real EnOcean ID (valid values: 0 to 127).

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

VAR_OUTPUT

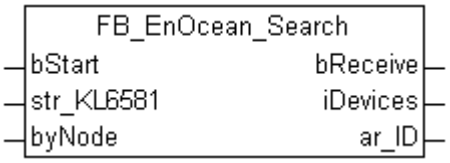
```
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : E_KL6581_Err;
```

bBusy: Block is is busy, no new data can be sent at the moment.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *iErrorID*.

iErrorID: Type of error (see E_KL6581_Err [▶ 45]).

6.2.10 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 EnOcean transmitter and receiver KL6583-0000. This allows you to recognize whether an EnOcean device is received by several KL6583s.

VAR_INPUT

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

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

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

byNode: Filter - if the value is zero the EnOcean telegrams from all EnOcean transmitter and receiver KL6583-0000 are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

VAR_OUTPUT

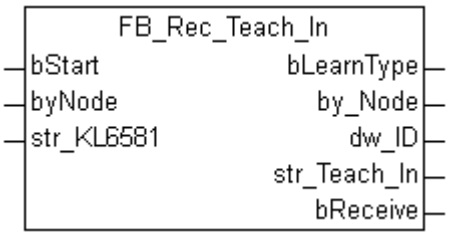
```
bReceive    : BOOL:=TRUE;
iDevices    : INT;
ar_ID       : ARRAY [0..255] OF DWORD;
```

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.

6.2.11 FB_Rec_Teach_In



This function block indicates when a learn button is pressed on an EnOcean device. If the *bLearnType* flag is set, then further information can be read out from the EnOcean device. This function must be provided by the EnOcean device. (So far, 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 active, if FALSE it is deactivated.

byNode: Filter - if the value is zero the EnOcean telegrams from all EnOcean transmitter and receiver KL6583-0000 are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

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.

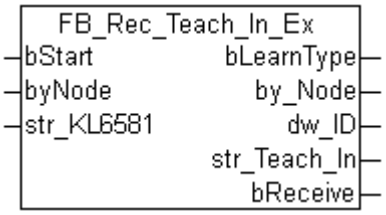
by_Node: Number of EnOcean devices found.

dw_ID: EnOcean IDs where a learn button was pressed.

str_Teach_In: Data structure - Manufacturer ID, type and profile (see STR_Teach_In [▶ 48]).

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

6.2.12 FB_Rec_Teach_In_Ex



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

VAR_INPUT

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

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

byNode: Filter - if the value is zero the EnOcean telegrams from all EnOcean transmitter and receiver KL6583-0000 are received. If a value of 1 to 8 is entered, only the data from the corresponding KL6583 are received.

str_KL6581: Data structure that is connected to the FB_KL6581() [▶ 24] function block (see STR_KL6581 [▶ 47]).

VAR_OUTPUT

```
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 IDs where a learn button was pressed.

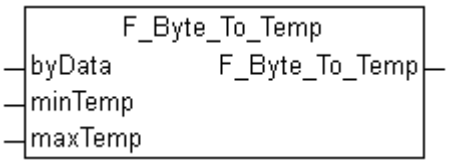
str_Teach_In: Data structure - Manufacturer ID, type and function (see STR_Teach [▶ 48]).

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

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2251	PC/CX, BX or BC	TcEnOcean library from V2.0.60

6.2.13 F_Byte_to_Temp : REAL



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 values are now input to the function along with the raw value. 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.

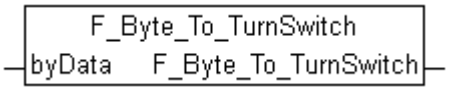
minTemp: Minimum temperature.

maxTemp: Maximum temperature.

F_Byte_To_Temp: Temperature value, scaled over *minTemp* and *maxTemp*.

6.2.14 F_Byte_to_TurnSwitch : STREnOceanTurnSwitch

STREnOceanTurnSwitch [▶ 49]



This function converts a byte raw value into a bool array, which is present as a data structure.

VAR_INPUT

```
byData      : BYTE;
```

byData: Raw data.

F_Byte_To_TurnSwitch: data structure (AUTO, 0, I, II, III).

The values are interpreted as follows:

210. .255: f_Byte_to_TurnSwitch.bStageAuto := TRUE;

190 ..209: f_Byte_to_TurnSwitch.bStage_0 := TRUE;

165. .189: f_Byte_to_TurnSwitch.bStage_1 := TRUE;

145. .164: f_Byte_to_TurnSwitch.bStage_2 := TRUE;

0.. 144: f_Byte_to_TurnSwitch.bStage_3 := TRUE;

6.2.15 Error codes KL6581

Value (hex)	Value (dez)	Value (enum)	Description
0x0000	0	NO_ERROR	No error is present at the block.
0x000A	10	KL6581_WrongTerminal	Wrong terminal connected.
0x0010	16	KL6581_WatchdogError	Timeout during initialization of the block FB_KL6581() [▶ 24].
0x0011	17	KL6581_NoComWithKL6581	No connection to the terminal. Terminal variables linked in the System Manager? Terminal plugged wrong? Clean all, Rebuild all and ReScan in the System Manager?
0x0012	18	KL6581_idx_number_not_OK	The input variable *nIdx* of the block FB_KL6581() is greater than 64.
0x0013	19	KL6581_Switch_to_Stopp	The terminal is gone from the data exchange with the EnOcean transmitter and receiver KL6583-0000
0x0014	20	KL6581_not_ready	Internal message to the function blocks, which are connected to the FB_KL6581().
0x0015	21	KL6581_No_KL6853_Found	There is no KL6583 to the EnOcean master terminal KL6581
0x0016	22	KL6581_TransmissionError	Data could not be sent, check address of the KL6583 or KL6583 is non-operational.

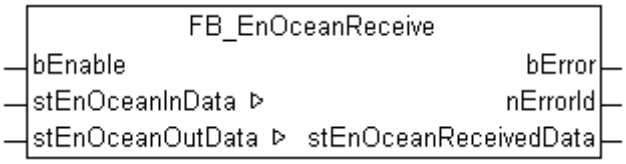
6.3 KL6021-0023

POUs	Description
FB_EnOceanReceive [▶ 35]	Communication with EnOcean bus terminal KL6021-0023

Read

POUs	Description
FB_EnOceanPTM100 [▶ 36]	Receives the signals of a PTM100 module.
FB_EnOceanPTM200 [▶ 38]	Receives the signals of a PTM200 module.
FB_EnOceanSTM100 [▶ 40]	Receives the signals of a STM100 module (obsolete).
FB_EnOceanSTM100Generic [▶ 42]	Receives the signals of a STM100 module.
FB_EnOceanSTM250 [▶ 43]	Receives the signals of a STM250 module.

6.3.1 FB_EnOceanReceive



The function-block FB_EnOceanReceive() is a receiving-unit, which collects the sent data of EnOcean-transmitters and then puts the gathered information into a structure called *stEnOceanReceivedData*. The structure is then analysed with the function-blocks FB_EnOceanPTM100() [▶ 36] and FB_EnOceanSTM100() [▶ 40]. Programming-examples are given in the documentation of these two units.

VAR_INPUT

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

bEnable: A positive signal at this input sets the function-block to the active-state. With a negative signal at the *bEnable* input the unit is without function and all outputs will be set to 0 or FALSE.

VAR_OUTPUT

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

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *nErrorId*.

nErrorId: Type of error (see Errorcodes [▶ 45]).

stEnOceanReceivedData: Into this structure the received data is written (see ST_EnOceanReceivedData [▶ 50]).

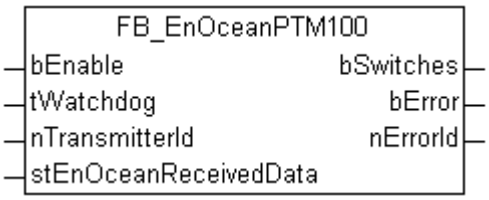
VAR_IN_OUT

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

stEnOceanInData: Is linked in System Manager to the input addresses of the EnOcean bus terminal KL6021-0023 (see ST_EnOceanInData [▶ 50]).

stEnOceanOutData: Is linked in System Manager to the output addresses of the EnOcean bus terminal KL6021-0023 (see ST_EnOceanOutData [▶ 50]).

6.3.2 FB_EnOceanPTM100



The function-block *FB_EnOceanPTM100* allows a user-friendly anaysis of the data sent by an EnOcean PTM100-transmitter-module. It gets its information from the receiver-unit FB_EnOceanReceive() [▶ 35].

In the difference of the PTM200- and PTM250-module, it is only possible to press one pushbuttons at the same time. Besides, the PTM100-module supports eight pushbuttons, instead of four pushbuttons.

It is very important, that for every new transmitting-module a new instance of the FB_EnOceanSTM100() must be programmed.

VAR_INPUT

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

bEnable: A positive signal at this input sets the function-block to the active-state. With a negative signal at the *bEnable* input the unit is without function and all outputs will be set to 0 or FALSE.

tWatchdog: Within this time the information at the input *stEnOceanReceivedData* has to be updated. The watchdog is disabled, if the entered time is set to t#0s.

nTransmitterId: The ID of the EnOcean-Transmitter, the function-block is assigned to.

stEnOceanReceivedData: Raw-information and therefore neccessary connection to the EnOcean receive function block FB_EnOceanReceive() [▶ 35]. This information is assemled in a structure of the type ST_EnOceanReceivedData [▶ 50].

VAR_OUTPUT

```
bSwitches           : ARRAY [0..7] OF BOOL;
bError              : BOOL := FALSE;
nErrorId            : UDINT := 0;
```

bSwitches: This array of 8 boolean variables contains the actual information about the button-states on the transmitter.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *nErrorId*.

nErrorId: Type of error (see Errorcodes [▶ 45]).

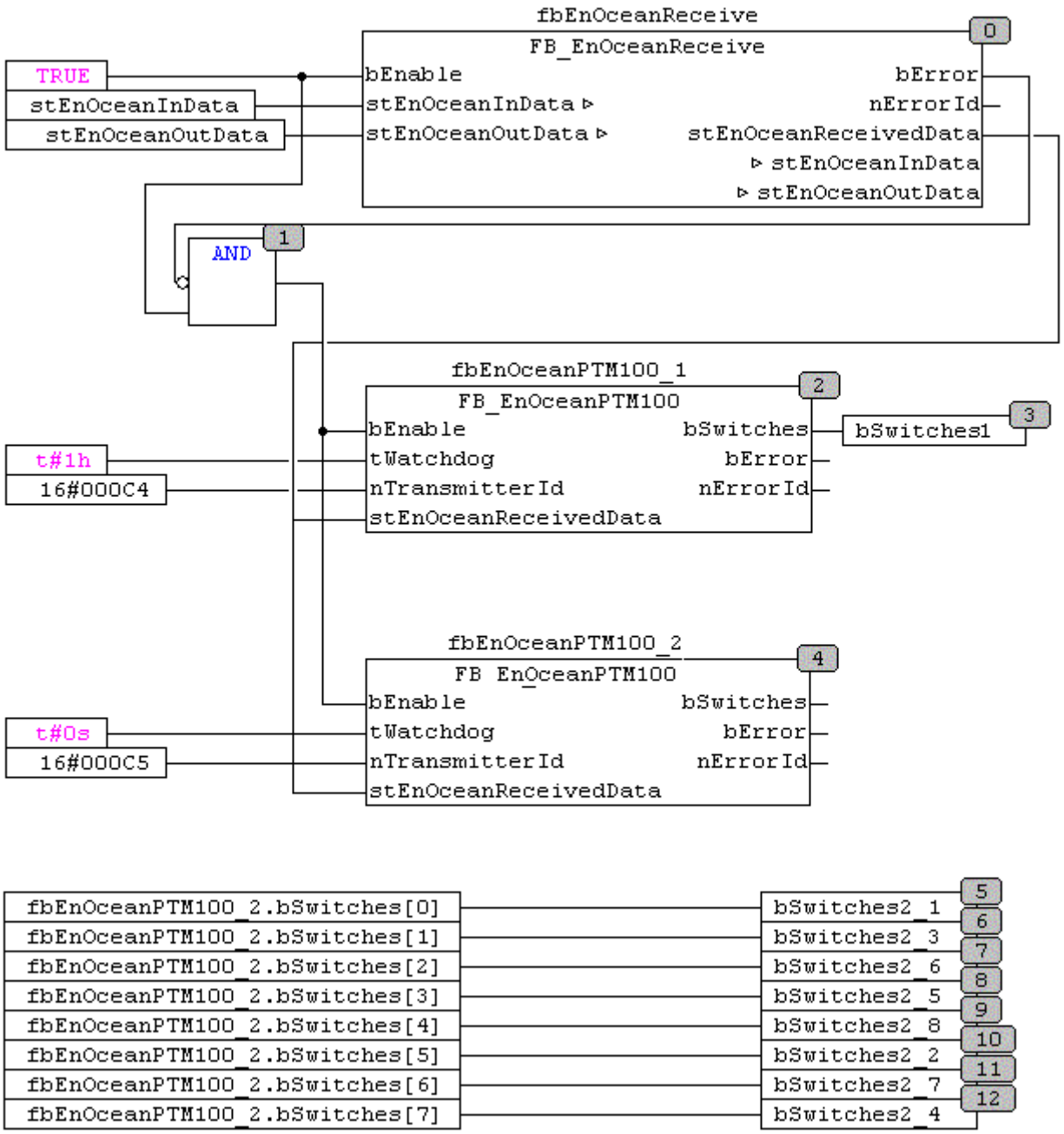
The following example should give some idea, how the function-block has to be used:

```
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,
```

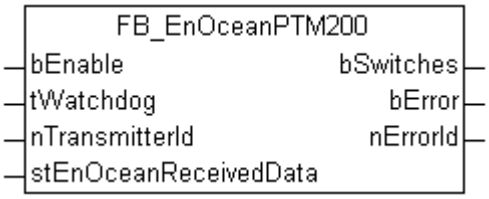
```
         stEnOceanInData := stEnOceanInData,
         stEnOceanOutData := stEnOceanOutData);

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_3 := fbEnOceanPTM100_2.bSwitches[1];
bSwitches2_6 := fbEnOceanPTM100_2.bSwitches[2];
bSwitches2_5 := fbEnOceanPTM100_2.bSwitches[3];
bSwitches2_8 := fbEnOceanPTM100_2.bSwitches[4];
bSwitches2_2 := fbEnOceanPTM100_2.bSwitches[5];
bSwitches2_7 := fbEnOceanPTM100_2.bSwitches[6];
bSwitches2_4 := fbEnOceanPTM100_2.bSwitches[7];
```

In this program two sensor-modules are observed - one has the ID 16#000000C4 and the other one 16#000000C5. Each transmitter has its own instance of the function-block FB_EnOceanSTM100()- these two units are getting their information of the pre-posted receiver FB_EnOceanReceive() and are only active (input *bEnable*), if the receiver is active and not in an error-state. The switches of the first transmitter is then assigned to the boolean array *bSwitches1* of the same size, those of the second transmitter are each assigned to different boolean variables (*bSwitches2_1*...*bSwitches2_8*) - both solutions are possible.

In continuous function chart editor (CFC) this example would look like the following:



6.3.3 FB_EnOceanPTM200



The function-block FB_EnOceanPTM200() allows a user-friendly analysis of the data sent by an EnOcean PTM200- or PTM250-transmitter-module. It gets its information from the receiver-unit FB_EnOceanReceive() [▶ 35].

In the difference of the PTM100-module, it is possible to press up to two pushbuttons simultaneous. Besides, the PTM200-module supports four pushbuttons, instead of eight pushbuttons.

It is very important, that for every new transmitting-module a new instance of the *FB_EnOceanSTM200* has to be programmed.

VAR_INPUT

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

bEnable: A positive signal at this input sets the function-block to the active-state. With a negative signal at the *bEnable* input the unit is without function and all outputs will be set to 0 or FALSE.

tWatchdog: Within this time the information at the input *stEnOceanReceivedData* has to be updated. The watchdog is disabled, if the entered time is set to t#0s.

nTransmitterId: The ID of the EnOcean-Transmitter, the function-block is assigned to.

stEnOceanReceivedData: Raw-information and therefore neccessary connection to the EnOcean receive function block FB_EnOceanReceive() [▶ 35]. This information is assemled in a structure of the type ST_EnOceanReceivedData [▶ 50].

VAR_OUTPUT

```
bSwitches          : ARRAY [0..3] OF BOOL;
bError             : BOOL := FALSE;
nErrorId           : UDINT := 0;
```

bSwitches: This array of 4 boolean variables contains the actual information about the pushbutton-states on the transmitter.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *nErrorId*.

nErrorId: Type of error (see Errorcodes [▶ 45]).

The following example should give some idea, how the function-block has to be used:

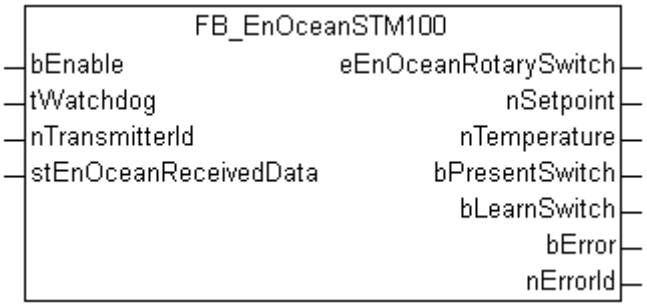
```
PROGRAM MAIN
VAR
  fbEnOceanReceive   : FB_EnOceanReceive;
  fbEnOceanPTM200_1  : FB_EnOceanPTM200;
  fbEnOceanPTM200_2  : FB_EnOceanPTM200;
  bSwitches1         : ARRAY [0..3] OF BOOL;
  bSwitches2_1       : BOOL;
  bSwitches2_2       : BOOL;
  bSwitches2_3       : BOOL;
  bSwitches2_4       : BOOL;
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);
bSwitches1 := fbEnOceanPTM200_1.bSwitches;
bSwitches2_1 := fbEnOceanPTM200_2.bSwitches[0];
bSwitches2_2 := fbEnOceanPTM200_2.bSwitches[1];
bSwitches2_3 := fbEnOceanPTM200_2.bSwitches[2];
bSwitches2_4 := fbEnOceanPTM200_2.bSwitches[3];
```

In this program two transmitter-modules (PTM200/PTM250) are observed - one has the ID 16#000000C6 and the other one 16#000000C7. Each transmitter has its own instance of the function-block FB_EnOceanSTM200()- these two units are getting their information of the pre-posted receiver FB_EnOceanReceive() [▶ 35] and are only active (input *bEnable*), if the receiver is active and not in an error-

state. The switches of the first transmitter is then assigned to the boolean array *bSwitches1* of the same size, those of the second transmitter are each assigned to different boolean variables (*bSwitches2_1*...*bSwitches2_4*) - both solutions are possible.

An Example in continuous function chart editor (CFC) is under the discription of the FB_EnOceanPTM100() [▶ 36].

6.3.4 FB_EnOceanSTM100



Obsolete! Please use in new projects the function block FB_EnOceanSTM100Generic() [▶ 42].

The function-block FB_EnOceanSTM100() allows a user-friendly anaysis of the data sent by an EnOcean STM100-module. It gets its information from the receiver-unit FB_EnOceanReceive() [▶ 35].

It is very important , that for every new EnOcean STM100-module a new instance of the FB_EnOceanSTM100() has to be programmed.

VAR_INPUT

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

bEnable: A positive signal at this input sets the function-block to the active-state. With a negative signal at the *bEnable* input the unit is without function and all outputs will be set to 0 or FALSE.

tWatchdog: Within this time the information at the input *stEnOceanReceivedData* has to be updated. The watchdog is disabled, if the entered time is set to t#0s.

nTransmitterId: The ID of the EnOcean-Transmitter, the function-block is assigned to.

stEnOceanReceivedData: Raw-information and therefore neccessary connection to the EnOcean receive function block FB_EnOceanReceive() [▶ 35]. This information is assemled in a structure of the type ST_EnOceanReceivedData [▶ 50].

VAR_OUTPUT

```
eEnOceanRotarySwitch  : E_EnOceanRotarySwitch;
nSetpoint             : INT;
nTemperature          : INT;
bPresentSwitch        : BOOL;
bLearnSwitch          : BOOL;
bError                : BOOL := FALSE;
nErrorId              : UDINT := 0;
```

eEnOceanRotarySwitch: This output describes the state of the rotary-switch on the transmitting-module. The information is defined by an ENUM called E_EnOceanRotarySwitch [▶ 49].

nSetpoint: The setpoint on the transmitter can be read at this output-variable. The range goes from -100 up to 100.

nTemperature: At this variable the measured temperature is given in 1/10 deg C. The lower limit is 0 deg C and the upper limit is 40 deg C. If the function-block is in a Watchdog-error-state, the temperature will be set to 850 deg C.

bPresentSwitch: This output will be TRUE, if the present-button on the transmitter is pressed.

bLearnSwitch: This output will be TRUE, if the learn-button on the transmitter is pressed.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *nErrorId*.

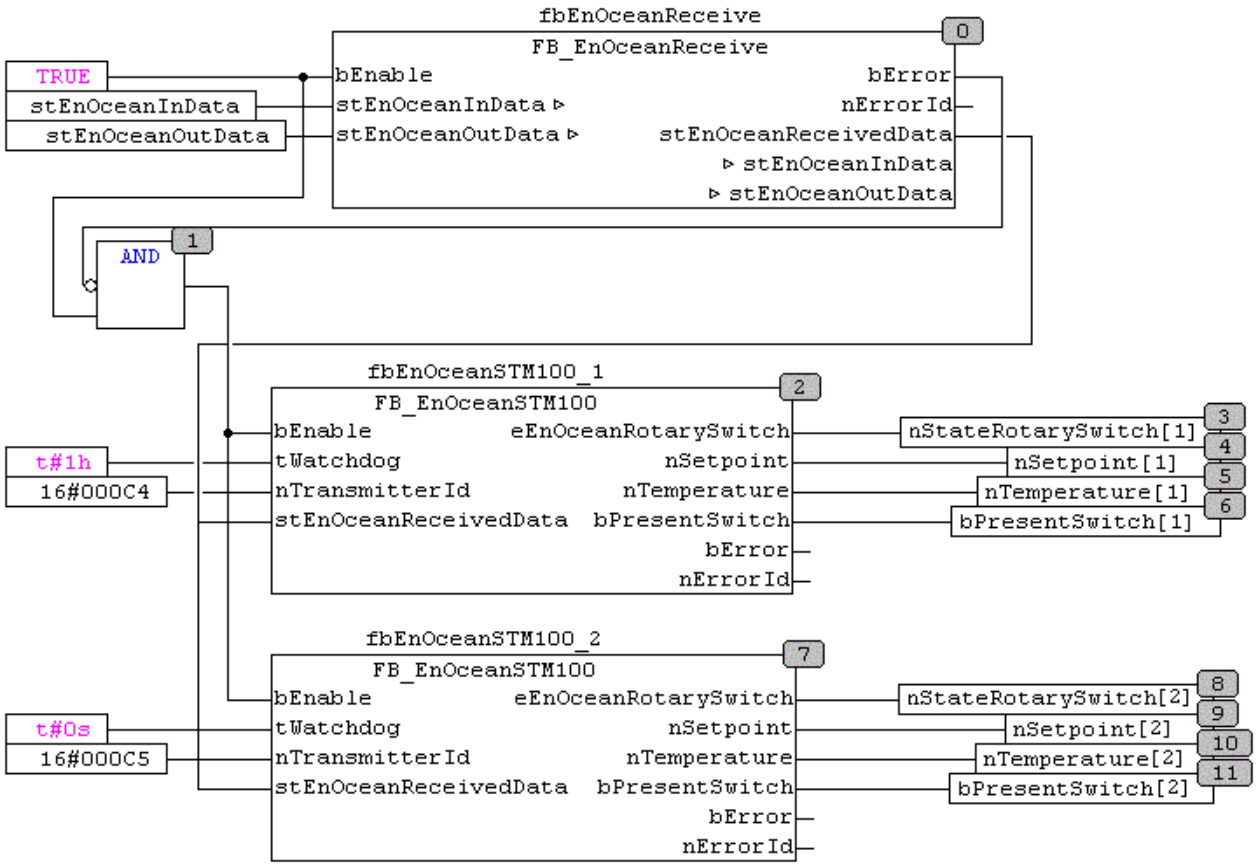
nErrorId: Type of error (see Errorcodes [▶ 45]).

The following example should give some idea, how the function-block has to be used:

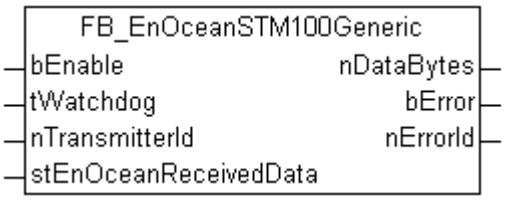
```
PROGRAM MAIN
VAR
  fbEnOceanReceive   : FB_EnOceanReceive;
  fbEnOceanSTM100_1  : FB_EnOceanSTM100;
  fbEnOceanSTM100_2  : FB_EnOceanSTM100;
  nTemperature       : ARRAY [1..2] OF INT;
  nSetpoint          : ARRAY [1..2] OF INT;
  nStateRotarySwitch : ARRAY [1..2] OF E_EnOceanRotarySwitch;
  bPresentSwitch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

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

fbEnOceanSTM100_1(bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.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 fbEnOceanReceive.bEnable,
          nTransmitterId := 16#000000C5,
          tWatchdog := t#0s,
          stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData,
          nTemperature => Temperature[2],
          nSetpoint => nSetpoint[2] ,
          eEnOceanRotarySwitch => nStateRotarySwitch[2],
          bPresentSwitch => bPresentSwitch[2]);
```

In this program two sensor-modules are observed - one has the ID 16#000000C4 and the other one 16#000000C5. Each transmitter has its own instance of the function-block FB_EnOceanSTM100() - these two units are getting their information of the pre-posted receiver FB_EnOceanReceive() and are only active (Input *bEnable*), if the receiver is active and not in an error-state. The first transmitter is observed by the watchdog-function, in this case the information has to be updated every hour. The second transmitter works without watchdog-observation. The resulting information is then assigned to variables for further usage in the program.

In continuous function chart editor (CFC) this example would look like the following:



6.3.5 FB_EnOceanSTM100Generic



The function-block FB_EnOceanSTM100Generic() allows a user-friendly anaysis of the data sent by an EnOcean STM100-module. It gets its information from the receiver-unit FB_EnOceanReceive().

It is very important , that for every new EnOcen STM100-module a new instance of the FB_EnOceanSTM100() has to be programmed.

VAR_INPUT

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

bEnable: A positive signal at this input sets the function-block to the active-state. With a negative signal at the *bEnable* input the unit is without function and all outputs will be set to 0 or FALSE.

tWatchdog: Within this time the information at the input *stEnOceanReceivedData* has to be updated. The watchdog is disabled, if the entered time is set to t#0s.

nTransmitterId: The ID of the EnOcean-Transmitter, the function-block is assigned to.

stEnOceanReceivedData: Raw-information and therefore neccessary connection to the EnOcean receive function block FB_EnOceanReceive() [▶ 35]. This information is assemled in a structure of the type ST_EnOceanReceivedData [▶ 50].

VAR_OUTPUT

```
nDataBytes          : ARRAY [0..3] OF BYTE;
bError              : BOOL := FALSE;
nErrorId            : UDINT := 0;
```

nDataBytes: Array of four bytes that contains the data of the STM100-module. The meaning of the data dependens on the manufacturer.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *nErrorId*.

nErrorId: Type of error (see Errorcodes [▶ 45]).

The following example should give some idea, how the function-block has to be used:

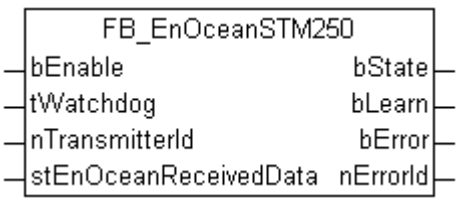
```
PROGRAM MAIN
VAR
  fbEnOceanReceive    : FB_EnOceanReceive;
  fbEnOceanSTM100_1   : FB_EnOceanSTM100Generic;
  fbEnOceanSTM100_2   : FB_EnOceanSTM100Generic;
  nTemperature        : ARRAY [1..2] OF BYTE;
  nSetpoint           : ARRAY [1..2] OF BYTE;
  nStateRotarySwitch  : ARRAY [1..2] OF BYTE;
  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];
nSetpoint[1] := fbEnOceanSTM100_1.nDataBytes[1];
nStateRotarySwitch[1] := fbEnOceanSTM100_1.nDataBytes[2];
nPresentSwitch[1] := fbEnOceanSTM100_1.nDataBytes[3];

fbEnOceanSTM100_2(bEnable := NOT fbEnOceanReceive.bError AND fbEnOceanReceive.bEnable,
          nTransmitterId := 16#000000C5,
          tWatchdog := t#0s,
          stEnOceanReceivedData := fbEnOceanReceive.stEnOceanReceivedData);
nTemperature[2] := fbEnOceanSTM100_2.nDataBytes[0];
nSetpoint[2] := fbEnOceanSTM100_2.nDataBytes[1];
nStateRotarySwitch[2] := fbEnOceanSTM100_2.nDataBytes[2];
nPresentSwitch[2] := fbEnOceanSTM100_2.nDataBytes[3];
```

In this program two sensor-modules are observed - one has the ID 16#000000C4 and the other one 16#000000C5. Each transmitter has its own instance of the function-block FB_EnOceanSTM100Generic() - these two units are getting their information of the pre-posted receiver FB_EnOceanReceive() [▶ 35] and are only active (Input *bEnable*), if the receiver is active and not in an error-state. The first transmitter is observed by the watchdog-function, in this case the information must be updated every hour. The second transmitter works without watchdog-observation. The resulting information is then assigned to variables for further usage in the program. You should scale the data to real physical units. Usually, in the datasheet of the sensor are more information about the data conversion.

Also see about this

- FB_EnOceanReceive [▶ 35]

6.3.6 FB_EnOceanSTM250



The function-block FB_EnOceanSTM250() allows a user-friendly anaysis of the data sent by an EnOcean STM250-module. It gets its information from the receiver-unit FB_EnOceanReceive() [▶ 35].

It is very important , that for every new EnOcean STM250-module a new instance of the *FB_EnOceanSTM250* has to be programmed.

VAR_INPUT

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

bEnable: A positive signal at this input sets the function-block to the active-state. With a negative signal at the *bEnable* input the unit is without function and all outputs will be set to 0 or FALSE.

tWatchdog: Within this time the information at the input *stEnOceanReceivedData* has to be updated. The watchdog is disabled, if the entered time is set to t#0s.

nTransmitterId: The ID of the EnOcean-Transmitter, the function-block is assigned to.

stEnOceanReceivedData: Raw-information and therefore neccessary connection to the EnOcean receive function block FB_EnOceanReceive() [▶ 35]. This information is assemled in a structure of the type ST_EnOceanReceivedData [▶ 50].

VAR_OUTPUT

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

bState: This output will be TRUE, if the contact of the Reed-relay on the STM250-transmitter-modul is activ (contact closed).

bLearn: This output will be FALSE, if the learn-button on the STM250-transmitter-modul is pressed.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *nErrorId*.

nErrorId: Type of error (see Errorcodes [▶ 45]).

The following example should give some idea, how the function-block has to be used:

```
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 program one EnOcean STM250-modules with the ID 16#000000C4 is observed. The transmitter has its own instance of the function-block FB_EnOceanSTM250(). This unit is getting this information of the pre-posted receiver FB_EnOceanReceive() [▶ 35] and is only active (Input *bEnable*), if the receiver is active and not in an error-state. The transmitter is observed by the watchdog-function, in this case the information has to be updated every hour. The resulting information is then assigned to variables for further usage in the program.

6.3.7 Error codes KL6021-0023

Value (hex)	Description
0x0000	No error.
0x0001	Checksum-error.
0x0002	Watchdog-error.
0x0003	Bufferoverflow (in the KL6023).
0x0004	No data received yet.

6.4 Data types

6.4.1 E_EnOcean_Org

Type of EnOcean telegram.

```
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 acknowledgment.

6.4.2 E_KL6581_Err

Error massage.

```
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 is present at the block.

KL6581_WrongTerminal**:** Wrong terminal connected.

KL6581_WatchdogError**:** Timeout during initialization of the block FB_KL6581() [▶ 24].

KL6581_NoComWithKL6581**:** No connection to the terminal. Terminal variables linked in the System Manager? Terminal plugged wrong? Clean all, Rebuild all and ReScan in the System Manager?

KL6581_idx_number_not_OK**:** The input variable *nIdx* of the block FB_KL6581() is greater than 64.

KL6581_Switch_to_Stopp**:** The terminal is gone from the data exchange with the EnOcean transmitter and receiver KL6583-0000. No EnOcean data were sent or received.

KL6581_not_ready**:** Internal message to the function blocks, which are connected to the FB_KL6581().

KL6581_No_KL6853_Found**:** There is no KL6583 to the KL6581 connected or the communication does not exist!

KL6581_TransmissionError**:** Data could be sent, address the KL6583 check or KL6583 non-operational.

6.4.3 KL6581_Input

Process image of the inputs.

Is linked in the System Manager to the terminal.

```
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.

6.4.4 KL6581_Output

Process image of the outputs.

Is linked in the System Manager to the terminal.

```
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.

6.4.5 STR_EnOceanSwitch

State of the buttons.

```
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.

6.4.6 STR_KL6581

Internal structure.

About this structure, the block FB_KL6581() [▶ 24] is connected to the read / send function 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     : DWORD;
  iErrorId   : E_KL6581_Err;
  by_STATE   : BYTE;
  bError     : BOOL;
  idx    : USINT;
END_STRUCT
END_TYPE
```

by_Status: Status.

by_Node: Node number of the EnOcean transmitter and receiver KL6583-0000, which has received the EnOcean telegram.

by_ORG: Type of EnOcean telegram.

ar_DB: Data bytes.

_Dummy: Placeholder, without further significance.

dw_ID: Transmitter Id.

ptData: Pointer.

iErrorId: Type of error (see E_KL6581_Err [▶ 45]).

by_STATE: State.

bError: If the function-block is in an error-state, this output will be set to TRUE. This error is described by the variable *iErrorId*.

idx: Index.

6.4.7 STR_Teach

Data structure - Manufacturer ID, type and function.

```
TYPE STR_Teach :
STRUCT
  nManufacturerID  : WORD;
  nTYPE        : BYTE;
  nFunc        : BYTE;
END_STRUCT
END_TYPE
```

nManufacturerID: Manufacturer ID.

nTYPE: Type.

nFunc: Function.

6.4.8 STR_Teach_In

Data structure - Manufacturer ID, type and profile.

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

nManufacturerID: Manufacturer ID.

nTYPE: Type.

nProfile: Profile.

6.4.9 STREnOceanTurnSwitch

strEnOceanTurnSwitch describes the state of the rotary-switch on the transmitting-module.

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

bStageAuto: Switch is set to „Auto“.

bStage_0: Switch is set to „0“.

bStage_1: Switch is set to „1“.

bStage_2: Switch is set to „2“.

bStage_3: Switch is set to „3“.

6.4.10 AR_EnOceanWindow

This structure shows the state of the window.

```
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.

6.4.11 E_EnOceanRotarySwitch

E_EnOceanRotarySwitch describes the state of the rotary-switch on the transmitting-module.

```
TYPE E_EnOceanRotarySwitch :
(
  eEnOceanRotarySwitchStep0  := 0,
  eEnOceanRotarySwitchStep1  := 1,
  eEnOceanRotarySwitchStep2  := 2,
  eEnOceanRotarySwitchStep3  := 3,
  eEnOceanRotarySwitchAuto   := 4,
)
END_TYPE
```

eEnOceanRotarySwitchStep0**:** Switch is set to „0“.

eEnOceanRotarySwitchStep1**:** Switch is set to „1“.

eEnOceanRotarySwitchStep2**:** Switch is set to „2“.

eEnOceanRotarySwitchStep3**:** Switch is set to „3“.

eEnOceanRotarySwitchAuto**:** Switch is set to „Auto“.

6.4.12 E_EnOceanSensorType

Sensor type.

```
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.

6.4.13 ST_EnOceanInData

Process image of the inputs of the EnOcean bus terminal KL6021-0023.

Is linked in the System Manager to the terminal.

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

nStatus: Status byte.

nData: 11 bytes for input data.

6.4.14 ST_EnOceanOutData

Process image of the outputs of the EnOcean bus terminal KL6021-0023.

Is linked in the System Manager to the terminal.

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

nCtrl: Control byte.

nData: 11 bytes for output data.

6.4.15 ST_EnOceanReceivedData

Internal structure

About this structure, the block FB_EnOceanReceive() [▶ 35] is connected to the meter function blocks.

```
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 from EnOcean.

nLength: Length.

eEnOceanSensorType: Sensor type (see E_EnOceanSensorType [▶ 50]).

nData: Data bytes.

nStatus: Status.

nTransmitterId: Transmitter Id.

7 Appendix

7.1 Examples

Requirements

Example	Description
https://infosys.beckhoff.com/content/1033/tcplclibenocean/Resources/11985704843/.zip	TwinCAT-PLC-Project for the KL6581.
Micropelt MVA002	Communication with the thermostatic radiator valve MVA-002 of the micropelt company.
HORA SmartDrive MX	Communication with the thermostatic radiator valve SmartDrive MX of the HORA company.

7.2 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

More Information:
www.beckhoff.com/tx1200

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