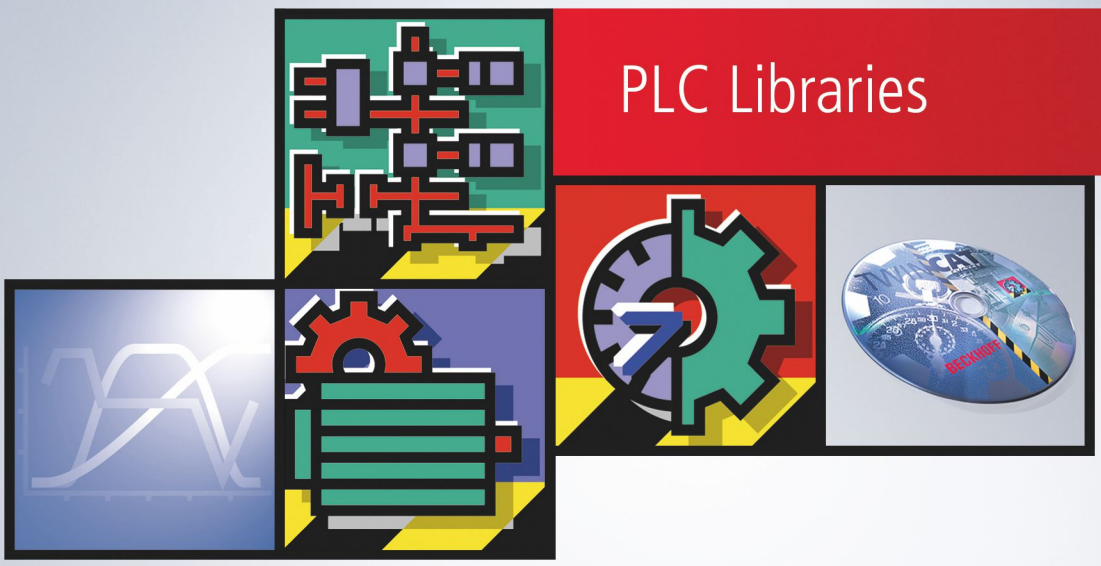
BECKHOFF New Automation Technology

Manual | EN

# TX1200

## TwinCAT 2 | PLC Library: TcEIB




2023-09-13 | 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

© 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 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 EIB library is a TwinCAT PLC library for data exchange with EIB devices.

All function blocks from the library must be called in the same task.

This library is to be used only in conjunction with a KL6301 (EIB master terminal).

# 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 EIB 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 Function of the KL6301

Working with the EIB bus terminal requires function blocks, which are described in this documentation.

From firmware version B1 and library version V3.000.000 there are three different modes that can be activated in the KL6301() [▶ 26] function block.

Mode 0: 4 filters, each with 64 group entries (compatible with firmware B0)

Mode 1: 8 filters, each with 32 group entries

Mode 2: 8 filters, each with 32 inverted group entries

Mode 100: Monitor function (all group address telegrams can be received, the KL6301 sends no ACK). Sending is disabled in this mode.

### Sending

The KL6301 sends data individually. This means that an Data variable sent to the KL6301 is sent to the EIB network individually. Subsequent EIB data can only be transferred to the KL6301 after a successful transfer. Two types of EIB telegrams can be sent:

- WRITE_GROUP for writing data to other EIB devices
- READ_GROUP_REQ for requesting data from other EIB devices

### Receiving

The KL6301 has a maximum of 8 filter addresses. These filters filter the EIB group addresses. Only EIB telegrams entered in the filter are visible in the process image and are acknowledged.

A filter may contain up to 64 group addresses. With filters a total of 256 group addresses are available. For 8 filters multiplied by 32 entries this makes a total of 256 group addresses to receive data. The system is configured via a function block. The group addresses are loaded and are immediately active when the Bus Terminal is initialized.

At lease one filter has to be parameterized. The data type is not significant for the filter setting.

### Monitor function

If mode 100 is activated no filters must be set. The filters EIB_GROUP_FILTER just have to be empty and not to be written.

# 5 Integration into TwinCAT

## 5.1 KL6301 - Linking to the TwinCAT System Manager

**How do I link the KL6771 to the System Manager?**

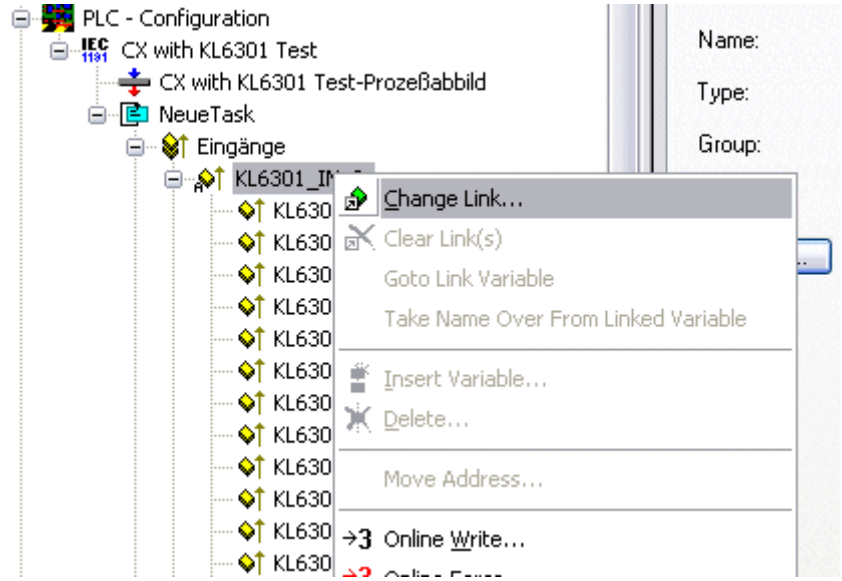



*Figure 1*

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

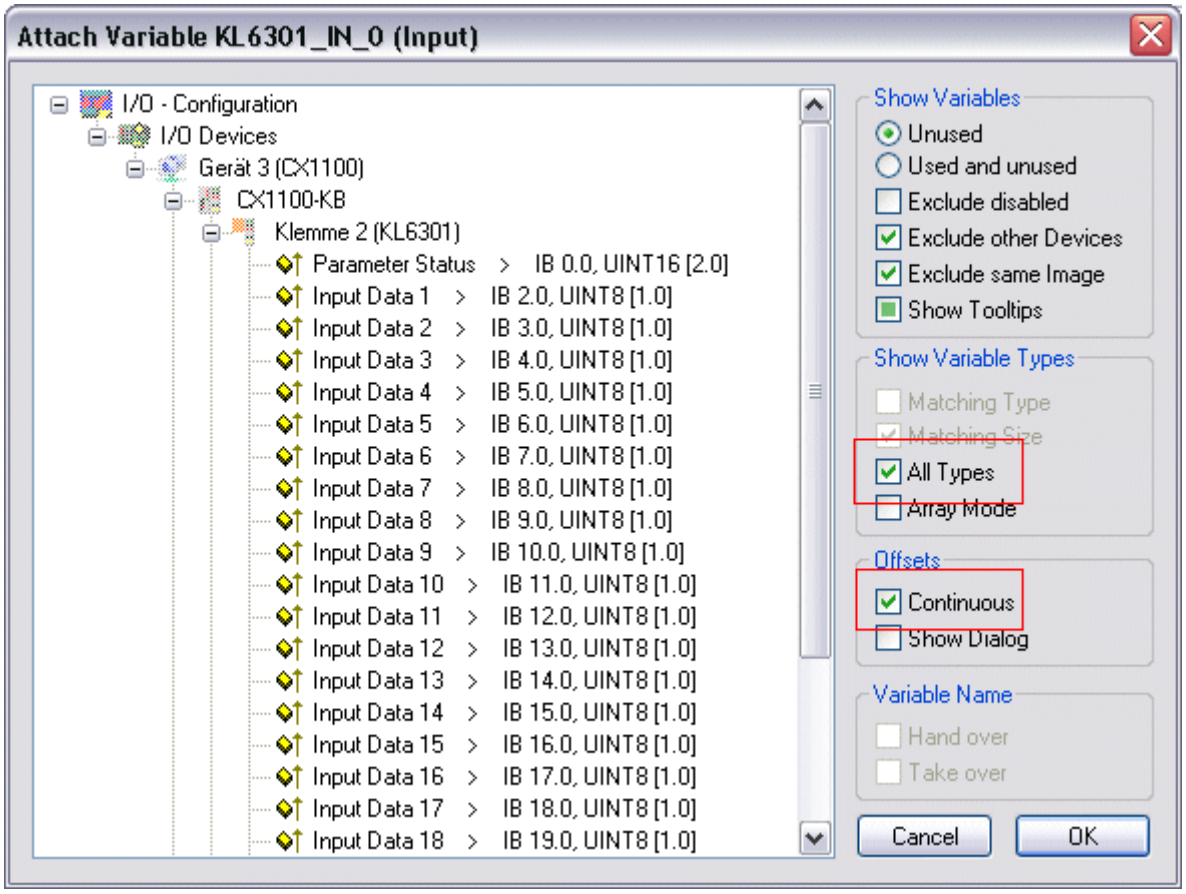



*Figure 2*

Click the mouse on the first variable in the KL6301 (Parameter Status). Then press the <SHIFT> key, and hold it down. Take the mouse pointer to the last variable in the KL6301 (Input Data 22), and again click the left mouse button. Now release the <SHIFT> key again. All the terminal's data should now be highlighted (see Figure 3). Then press the "OK" button.

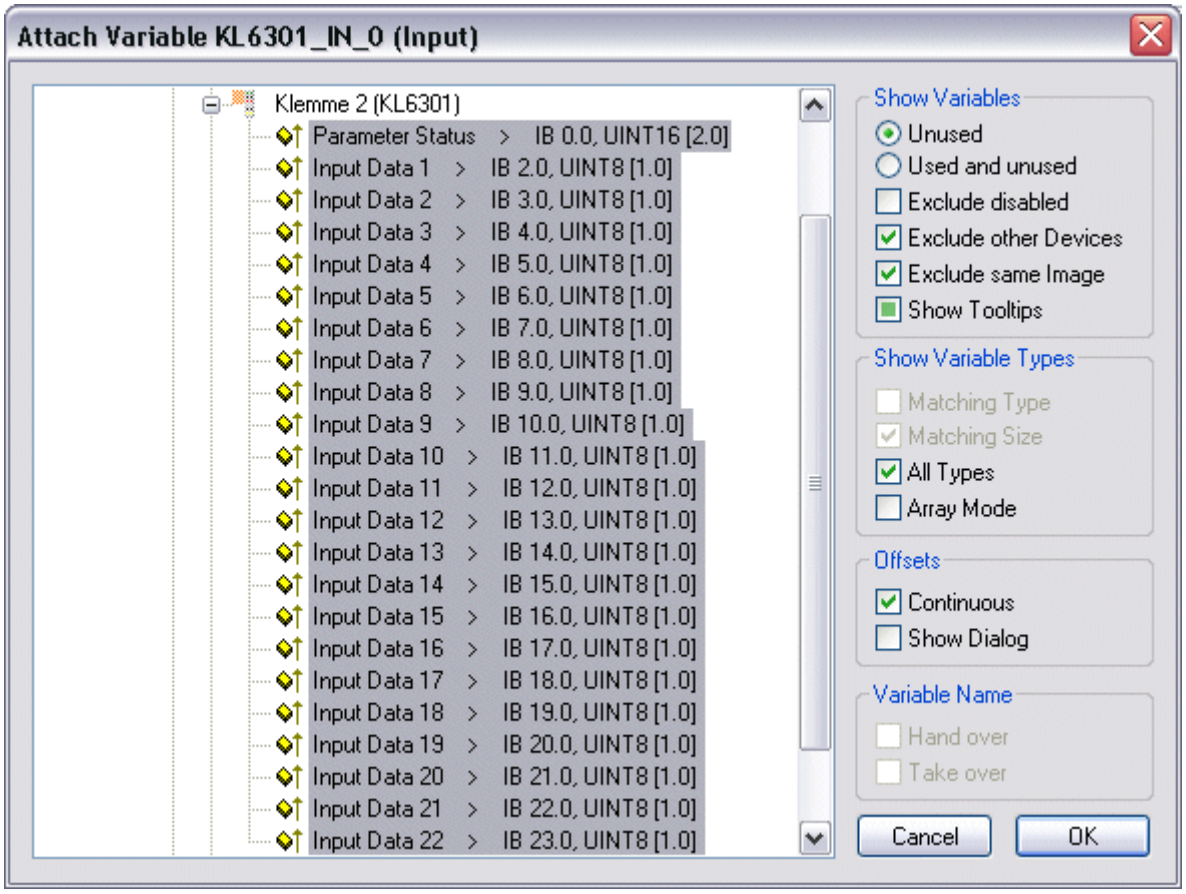



*Figure 3*

You can now check the connection. To do this, go to the KL6301 and open it. All the terminal's data should now be marked by a small arrow (see Figure 4). If that is the case, then proceed in exactly the same way with the outputs.

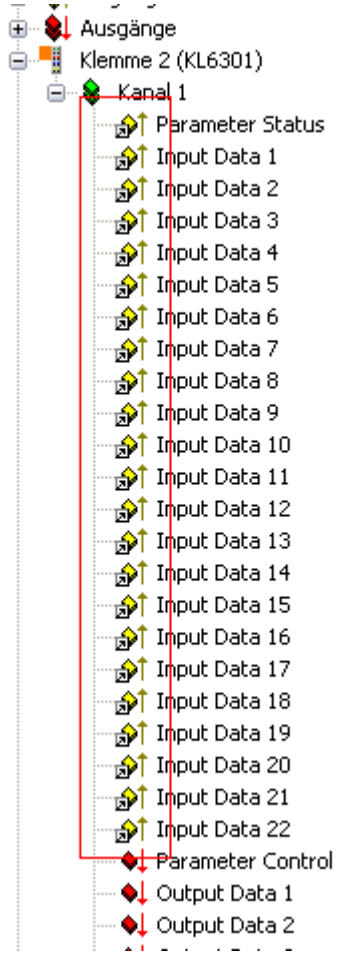



*Figure 4*

## 5.2 Integration in TwinCAT (CX9020)

This example describes how a simple PLC program for EIB can be written in TwinCAT and how it is linked with the hardware. The task is to change the state of a switching output with a button.

https://infosys.beckhoff.com/content/1033/tcplclibeib/Resources/11993060235/.zip https://infosys.beckhoff.com/content/1033/tcplclibeib/Resources/11993060235/.zip

**Hardware**

**Setting up the components**

The following hardware is required:

- 1x Embedded PC CX9020
- 1x digital 2-channel input terminal KL1002 (for the set and reset functions)
- 1x EIB terminal KL6301
- 1x end terminal KL9010

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

This example assumes that a Set button was connected to the first KL1002 input and a Reset button to the second, and that the EIB group address of the switching output is known.

### Software

### Creation of the PLC program

Create a new PLC project for PC-based systems (ARM) and add the *TcEIB.lib* library.

Next, generate the following global variables:

```
VAR_GLOBAL
    bSet        AT %I*     : BOOL;
    bReset      AT %I*     : BOOL;
    arrKL6301_in    AT %I* : ARRAY[1..24] OF BYTE;
    arrKL6301_out   AT %Q* : ARRAY[1..24] OF BYTE;
    stDataRec              : EIB_REC;
END_VAR
```

**bSet:** Input variable for the Set button.

**bReset:** Input variable for the Reset button.

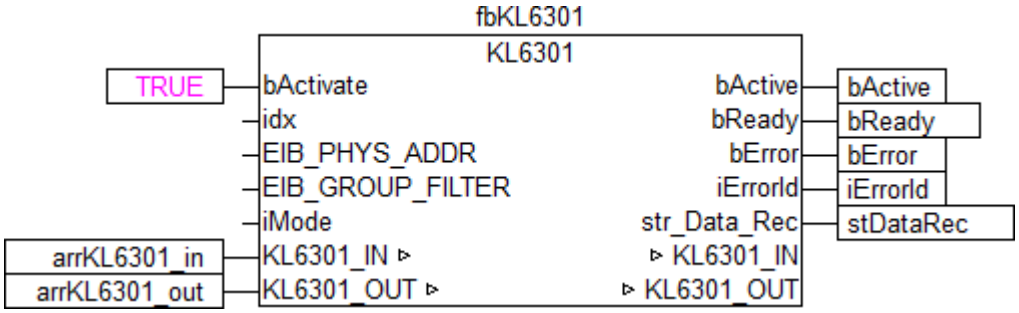
**arrKL6301_in:** Input variable for the EIB terminal.

**arrKL6301_out:** Output variable for the EIB terminal.

**stDataRec:** Needed for communication [▶ 67] with EIB.

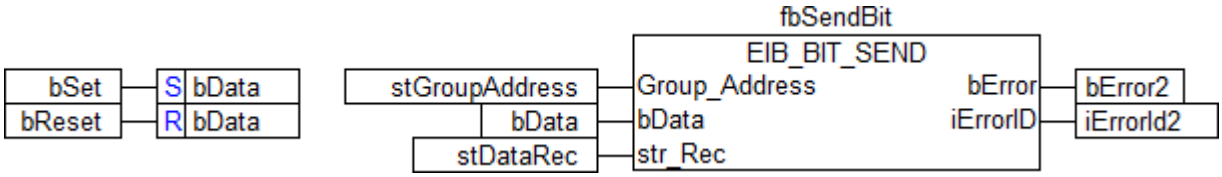
All EIB function blocks must be called in the same task.

Therefore, create a MAIN program (CFC) in which the function blocks KL6301() [▶ 26] and EIB_BIT_SEND() [▶ 57] are called. Make sure to link the communication block with *arrKL6301_in*, *arrKL6301_out* and *stDataRec*.



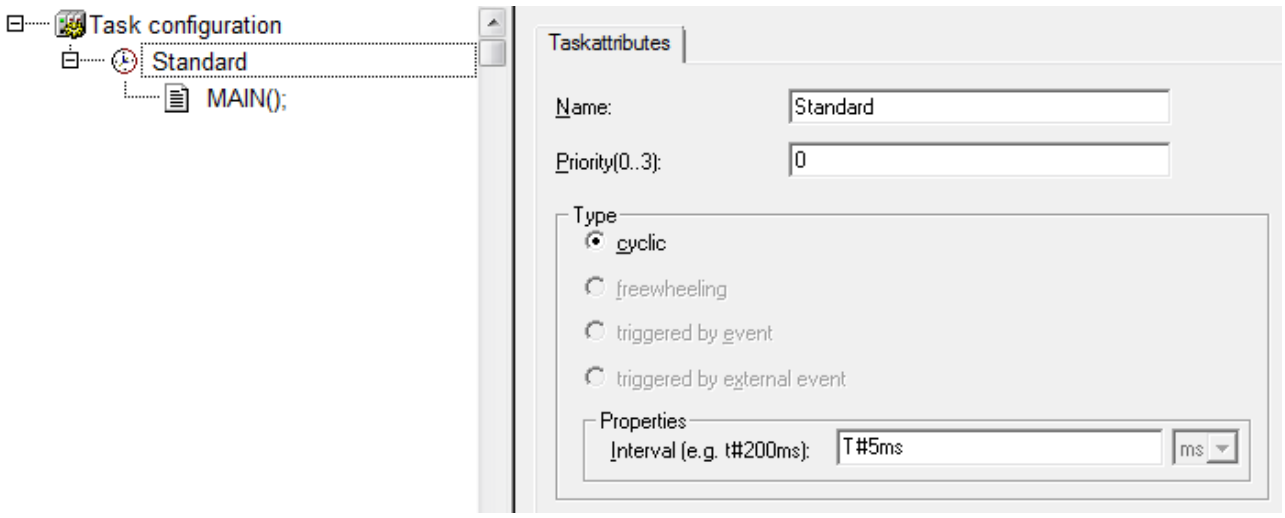

Sample-PC-Comm

Link the local variable *bData* with the global variables *bSet* and *bReset*, then with the input *bData* of the send block. Link the global variable *stDataRec* with *st_Rec*.




Sample-PC-MAIN

Go to the task configuration and give the task a lower interval time. More detailed information can be found in the KL6301() [▶ 26] function block description.



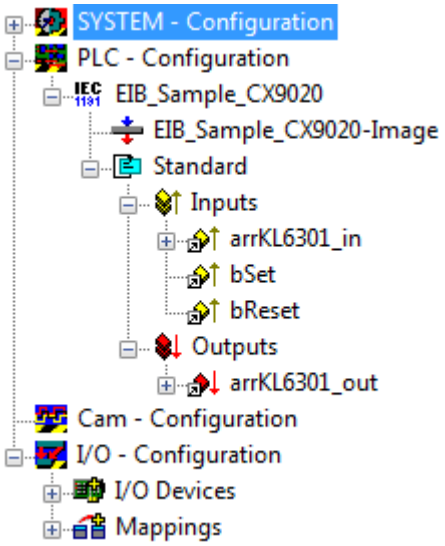

Sample_BC_Task

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.

The switching output can be set or reset by pressing the button.

## 5.3 Integration into TwinCAT (BC9191)

This example describes how a simple PLC program for EIB can be written in TwinCAT and how it is linked with the hardware. The task is to change the state of a switching output with a button.

https://infosys.beckhoff.com/content/1033/tcplclibeib/Resources/11993061643/.zip https://infosys.beckhoff.com/content/1033/tcplclibeib/Resources/11993061643/.zip

### Hardware

### Setting up the components

The following hardware is required:

- 1x Bus Terminal Controller BC9191
- 1x potential feed terminal 24V DC
- 1x digital 2-channel input terminal KL1002 (for the set and reset functions)
- 1x EIB terminal KL6301
- 1x end terminal KL9010

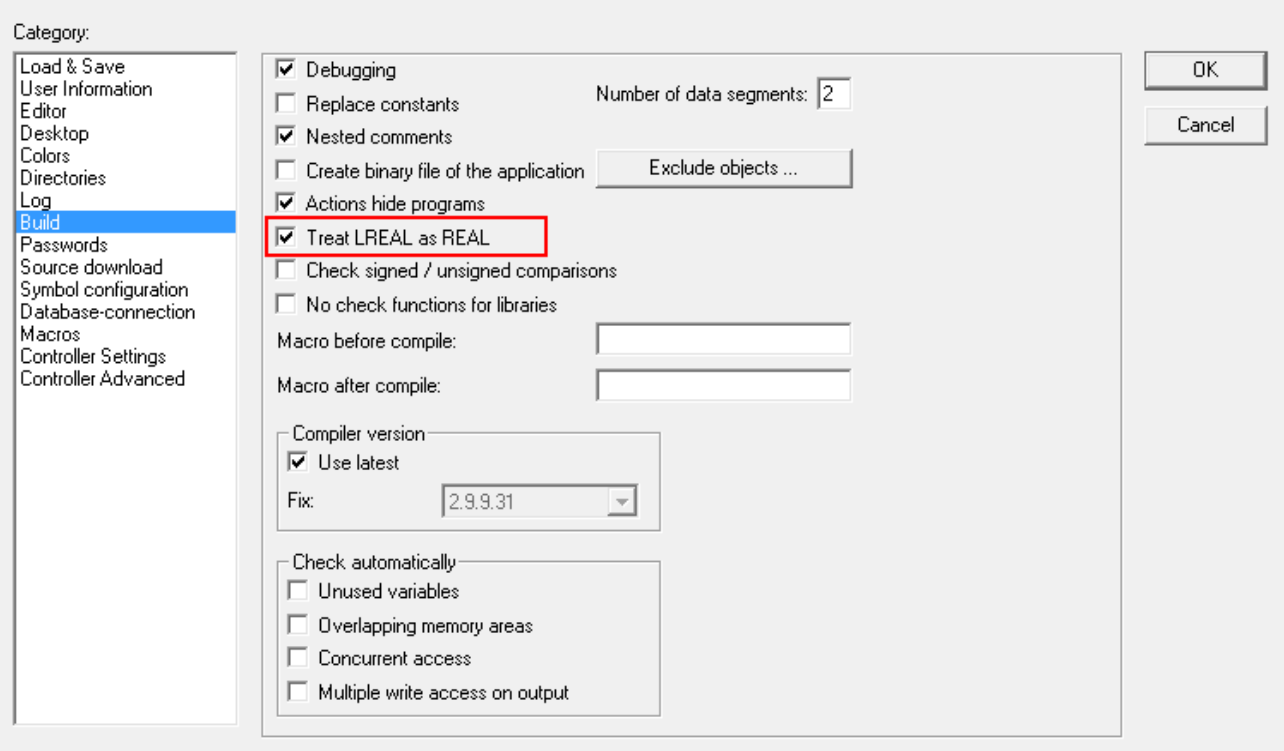
Set up the hardware and the EIB components as described in the associated documentation.

This example assumes that a Set button was connected to the first KL1002 input and a Reset button to the second, and that the EIB group address of the switching output is known.

### Software

### Creation of the PLC program

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




Next, generate the following global variables:

```
VAR_GLOBAL
    bSet         AT %I*      : BOOL;
    bReset       AT %I*      : BOOL;
    arrKL6301_in     AT %I* : ARRAY[1..24] OF BYTE;
    arrKL6301_out    AT %Q* : ARRAY[1..24] OF BYTE;
    stDataRec               : EIB_REC;
END_VAR
```

**bSet :** Input variable for the Set button.

**bReset:** Input variable for the Reset button.

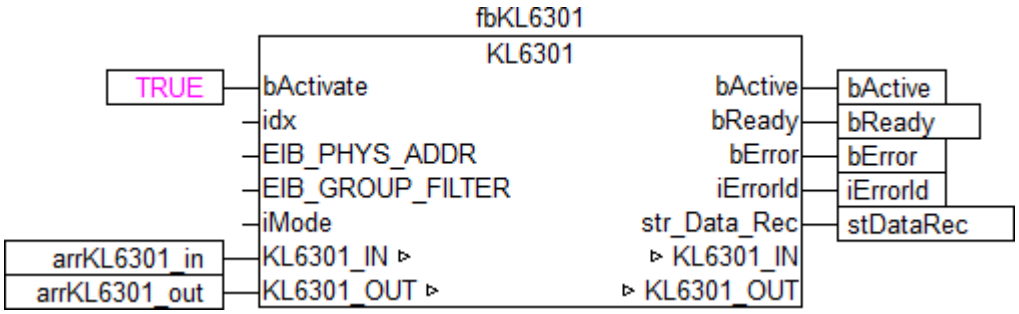
**arrKL6301_in :** Input variable for the EIB terminal.

**arrKL6301_out :** Output variable for the EIB terminal.

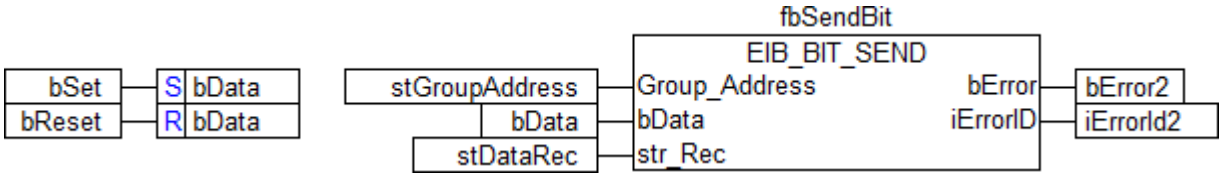
**stDataRec :** required for communication [▶ 67] with EIB.

Since BC devices can only process one task, communication with EIB cannot run separately.

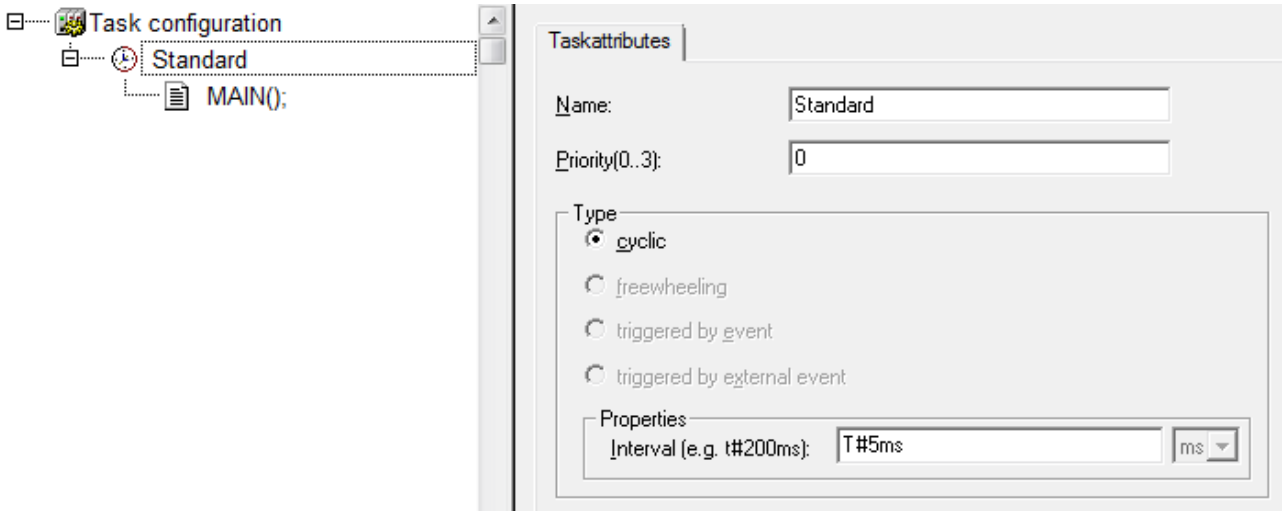
Therefore, create a MAIN program (CFC) in which the KL6301() [▶ 26] and EIB_BIT_SEND() [▶ 57] function blocks are called. Make sure to link the communication block with *arrKL6301*, *arrKL6301* and *stDataRec*.




Link the local variable *bData* with the global variables *bSet* and *bReset*, then with the input *bData* of the send block. Link the global variable *stDataRec* with *st_Rec*.




Go to the task configuration and give the task a lower interval time. More detailed information can be found in the KL6301() [▶ 26] block description.

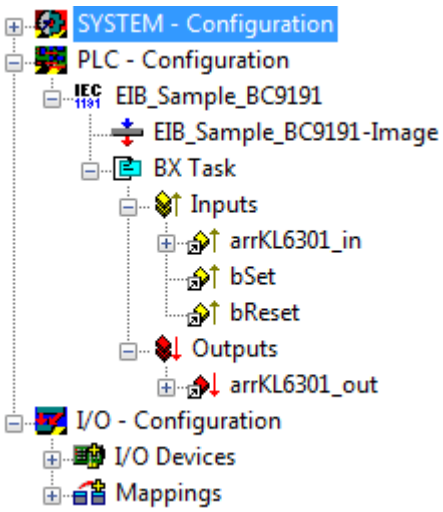



Now load the project as a boot project to the BC and save it.

### Configuration in the System Manager

Create a new TwinCAT 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 Bus Terminal inputs and outputs, create the allocations, and activate the configuration. Then start the device in run mode.
Your BC is now ready for use.

The switching output can be set or reset by pressing the button.

# 6 Programming

| Contents |
|---|
| General Information [▶ 22] |
| KL6301 - Link with the TwinCAT System Manager [▶ 11] |
| EIB group filter [▶ 22] |

### General

| Contents | Description |
|---|---|
| KL6301 [▶ 26] | Communication with a KL6301 |
| KL6301_EX [▶ 27] | Communication with a KL6301 |

### Read

| Function blocks | Description |
|---|---|
| EIB_2OCTET_FLOAT_REC [▶ 29] | Receiving of 2-byte Float EIB data and conversion to REAL |
| EIB_2OCTET_SIGN_REC [▶ 29] | Receiving of 2-byte Sign EIB data and conversion to INT |
| EIB_2OCTET_UNSIGN_REC [▶ 30] | Receiving of 2-byte Unsign EIB data and conversion to UINT |
| EIB_3BIT_CONTROL_REC [▶ 30] | Receiving of a "3 Bit Controlled" data type |
| EIB_4OCTET_FLOAT_REC [▶ 31] | Receiving of 4-byte Float EIB data and conversion to REAL |
| EIB_4OCTET_SIGN_REC [▶ 31] | Receiving of 4-byte Sign EIB data and conversion to DINT |
| EIB_4OCTET_UNSIGN_REC [▶ 32] | Receiving of 4-byte Unsign EIB data and conversion to UDINT |
| EIB_8BIT_SIGN_REC [▶ 32] | Receiving of 8 BIT EIB data and conversion to INT |
| EIB_8BIT_UNSIGN_REC [▶ 33] | Receiving of 8 BIT EIB data and conversion to BYTE |
| EIB_ALL_DATA_TYPES_REC [▶ 33] | Receives any EIB data |
| EIB_ALL_DATA_TYPES_REC_EX [▶ 34] | Receives any EIB data |
| EIB_BIT_CONTROL_REC [▶ 34] | Receiving of a "1 Bit Controlled" data type |
| EIB_BIT_REC [▶ 35] | Receiving of 1 BIT EIB data and conversion to BOOL |
| EIB_DATE_REC [▶ 36] | Receiving a date |
| EIB_TIME_REC [▶ 36] | Receiving a time |

### Send

| Function blocks | Description |
|---|---|
| EIB_2OCTET_FLOAT_SEND [▶ 37] | Sending a REAL value (conversion to 2 byte Float EIB) |
| EIB_2OCTET_FLOAT_SEND_EX [▶ 38] | Sending a REAL value (conversion to 2 byte Float EIB) |
| EIB_2OCTET_SIGN_SEND [▶ 39] | Sending an INT value (conversion to 2 byte Sign EIB) |
| EIB_2OCTET_SIGN_SEND_EX [▶ 40] | Sending an INT value (conversion to 2 byte Sign EIB) |
| EIB_2OCTET_UNSIGN_SEND [▶ 41] | Sending an UINT value (conversion to 2 byte Unsign EIB) |
| EIB_2OCTET_UNSIGN_SEND_EX [▶ 42] | Sending an UINT value (conversion to 2 byte Unsign EIB) |

| Function blocks | Description |
|---|---|
| EIB_3BIT_CONTROL_SEND [▶ 43] | Sending a "3 bit Controlled" data type |
| EIB_4OCTET_FLOAT_SEND [▶ 44] | Sending a REAL value (conversion to 4 byte Float EIB) |
| EIB_4OCTET_FLOAT_SEND_EX [▶ 45] | Sending a REAL value (conversion to 4 byte Float EIB) |
| EIB_4OCTET_SIGN_SEND [▶ 46] | Sending a DINT value (conversion to 4 byte Sign EIB) |
| EIB_4OCTET_SIGN_SEND_EX [▶ 47] | Sending a DINT value (conversion to 4 byte Sign EIB) |
| EIB_4OCTET_UNSIGN_SEND [▶ 48] | Sending a UDINT value (conversion to 4 byte Unsign EIB) |
| EIB_8BIT_SIGN_SEND [▶ 49] | Sending an INT value (conversion to 1 byte Sign EIB) |
| EIB_8BIT_SIGN_SEND_EX [▶ 50] | Sending an INT value (conversion to 1 byte Sign EIB) |
| EIB_8BIT_UNSIGN_SEND [▶ 51] | Sending a BYTE value (conversion to 1 byte Unsign EIB) |
| EIB_8BIT_UNSIGN_SEND_EX [▶ 52] | Sending a BYTE value (conversion to 1 byte Unsign EIB) |
| EIB_ALL_DATA_TYPES_SEND [▶ 54] | Sending any EIB data |
| EIB_BIT_CONTROL_SEND [▶ 56] | Sending a "1 bit Controlled" data type |
| EIB_BIT_SEND [▶ 57] | Sending a BOOL value |
| EIB_BIT_SEND_EX [▶ 58] | Sending a BOOL value |
| EIB_BIT_SEND_MANUAL [▶ 59] | Sending a BOOL value |
| EIB_DATE_SEND [▶ 60] | Sending a date |
| EIB_READ_SEND [▶ 60] | Sending a *Read_Group_Req* |
| EIB_TIME_SEND [▶ 61] | Sending a time |

### Functions

| Function blocks | Description |
|---|---|
| F_CONV_2GROUP_TO_3GROUP [▶ 63] | Conversion of a 2-stage group address to a 3-stage group address |
| F_CONV_3GROUP_TO_2GROUP [▶ 64] | Conversion of a 3-stage group address to a 2-stage group address |

### Enums

| Data types | Description |
|---|---|
| EIB_ERROR_CODE [▶ 64] | Error messages |
| EIB_PRIORITY [▶ 66] | EIB telegram priority |

### Structs

| Data types | Description |
|---|---|
| EIB_GROUP_ADDR [▶ 66] | 3-stage group address |
| EIB_GROUP_ADDR_2GROUP [▶ 66] | 2-stage group address |
| EIB_GROUP_FILTER [▶ 66] | Group filter |
| EIB_PHYS_ADDR [▶ 67] | Physical address |
| EIB_REC [▶ 67] | Links the send and receive blocks with the function block *KL6301* |

## 6.1 General information

**Installation**

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

**Name of the library**

This library replaces the "TcKL6301.lib/.lb6./.lbx". Only the name of the libraries has changed. The modules are still compatible.

**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.

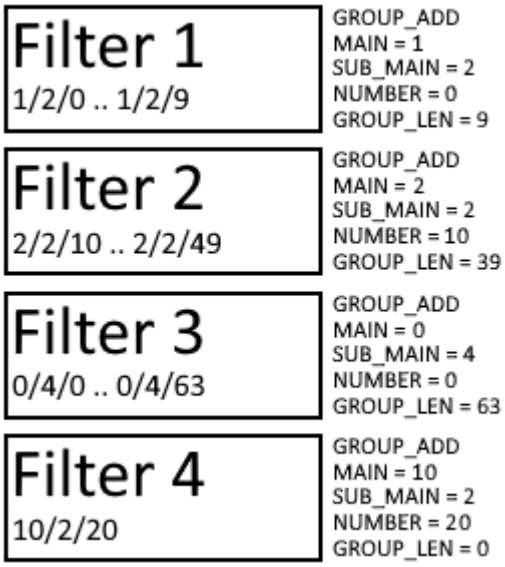
## 6.2 EIB group filter

The EIB group filters have to be parameterized before the KL6301 can enter data exchange mode. The filters are required for all data with a group address sent to the KL6301. Each group telegram that is also included in the filters is acknowledged and entered in the process data, i.e. it is visible in the function blocks. The KL6301 discards EIB telegrams with group addresses that are not included in the filter.

**Sample**

Filter 1 Group address 1/2/0 Length: 20
All EIB telegrams with group address 1/2/0 - 1/2/19 pass the filter

At least one filter must always be activated.
The selected mode determines the number and length of the group filters. The length specification starts at 0, which corresponds to exactly one entry.




**Change to firmware B1 and library version V3**

With firmware version B1 and the TwinCAT library TcEIB (V3.000.000) you can parameterize **instead of 4 filters also 8 filters**. However, the maximum length of the individual filters is reduced from 64 entries to 32 entries per filter group. Thus, the sum of the maximum receiving group filters remains the same at 256.

## 6.3 Function blocks

**General**

| Contents | Description |
|---|---|
| KL6301 [▶ 26] | Communication with a KL6301 |
| KL6301_EX [▶ 27] | Communication with a KL6301 |

**Read**

| Function blocks | Description |
|---|---|
| EIB_2OCTET_FLOAT_REC [▶ 29] | Receiving of 2-byte Float EIB data and conversion to REAL |
| EIB_2OCTET_SIGN_REC [▶ 29] | Receiving of 2-byte Sign EIB data and conversion to INT |
| EIB_2OCTET_UNSIGN_REC [▶ 30] | Receiving of 2-byte Unsign EIB data and conversion to UINT |
| EIB_3BIT_CONTROL_REC [▶ 30] | Receiving of a "3 Bit Controlled" data type |
| EIB_4OCTET_FLOAT_REC [▶ 31] | Receiving of 4-byte Float EIB data and conversion to REAL |
| EIB_4OCTET_SIGN_REC [▶ 31] | Receiving of 4-byte Sign EIB data and conversion to DINT |
| EIB_4OCTET_UNSIGN_REC [▶ 32] | Receiving of 4-byte Unsign EIB data and conversion to UDINT |
| EIB_8BIT_SIGN_REC [▶ 32] | Receiving of 8 BIT EIB data and conversion to INT |
| EIB_8BIT_UNSIGN_REC [▶ 33] | Receiving of 8 BIT EIB data and conversion to BYTE |
| EIB_ALL_DATA_TYPES_REC [▶ 33] | Receives any EIB data |
| EIB_ALL_DATA_TYPES_REC_EX [▶ 34] | Receives any EIB data |
| EIB_BIT_CONTROL_REC [▶ 34] | Receiving of a "1 Bit Controlled" data type |
| EIB_BIT_REC [▶ 35] | Receiving of 1 BIT EIB data and conversion to BOOL |

| Function blocks | Description |
|---|---|
| EIB_DATE_REC [▶ 36] | Receiving a date |
| EIB_TIME_REC [▶ 36] | Receiving a time |

**Send**

| Function blocks | Description |
|---|---|
| EIB_2OCTET_FLOAT_SEND [▶ 37] | Sending a REAL value (conversion to 2 byte Float EIB) |
| EIB_2OCTET_FLOAT_SEND_EX [▶ 38] | Sending a REAL value (conversion to 2 byte Float EIB) |
| EIB_2OCTET_SIGN_SEND [▶ 39] | Sending an INT value (conversion to 2 byte Sign EIB) |
| EIB_2OCTET_SIGN_SEND_EX [▶ 40] | Sending an INT value (conversion to 2 byte Sign EIB) |
| EIB_2OCTET_UNSIGN_SEND [▶ 41] | Sending an UINT value (conversion to 2 byte Unsign EIB) |
| EIB_2OCTET_UNSIGN_SEND_EX [▶ 42] | Sending an UINT value (conversion to 2 byte Unsign EIB) |
| EIB_3BIT_CONTROL_SEND [▶ 43] | Sending a "3 bit Controlled" data type |
| EIB_4OCTET_FLOAT_SEND [▶ 44] | Sending a REAL value (conversion to 4 byte Float EIB) |
| EIB_4OCTET_FLOAT_SEND_EX [▶ 45] | Sending a REAL value (conversion to 4 byte Float EIB) |
| EIB_4OCTET_SIGN_SEND [▶ 46] | Sending a DINT value (conversion to 4 byte Sign EIB) |
| EIB_4OCTET_SIGN_SEND_EX [▶ 47] | Sending a DINT value (conversion to 4 byte Sign EIB) |
| EIB_4OCTET_UNSIGN_SEND [▶ 48] | Sending a UDINT value (conversion to 4 byte Unsign EIB) |
| EIB_8BIT_SIGN_SEND [▶ 49] | Sending an INT value (conversion to 1 byte Sign EIB) |
| EIB_8BIT_SIGN_SEND_EX [▶ 50] | Sending an INT value (conversion to 1 byte Sign EIB) |
| EIB_8BIT_UNSIGN_SEND [▶ 51] | Sending a BYTE value (conversion to 1 byte Unsign EIB) |
| EIB_8BIT_UNSIGN_SEND_EX [▶ 52] | Sending a BYTE value (conversion to 1 byte Unsign EIB) |
| EIB_ALL_DATA_TYPES_SEND [▶ 54] | Sending any EIB data |
| EIB_BIT_CONTROL_SEND [▶ 56] | Sending a "1 bit Controlled" data type |
| EIB_BIT_SEND [▶ 57] | Sending a BOOL value |
| EIB_BIT_SEND_EX [▶ 58] | Sending a BOOL value |
| EIB_BIT_SEND_MANUAL [▶ 59] | Sending a BOOL value |
| EIB_DATE_SEND [▶ 60] | Sending a date |
| EIB_READ_SEND [▶ 60] | Sending a *Read_Group_Req* |
| EIB_TIME_SEND [▶ 61] | Sending a time |

## 6.3.1 Function blocks details

| Description | _Rec | _Send | | | | |
|---|---|---|---|---|---|---|
| | | _Send | First Cycle | Delta, min. Send Time | Polling | Auto/manual |
| EIB_BIT | yes [▶ 35] | yes [▶ 57] | no | 200 msec | no | Auto |
| EIB_BIT_EX | no | yes [▶ 58] | yes | 1 sec, variable | 10 sec, variable | Auto/manual |

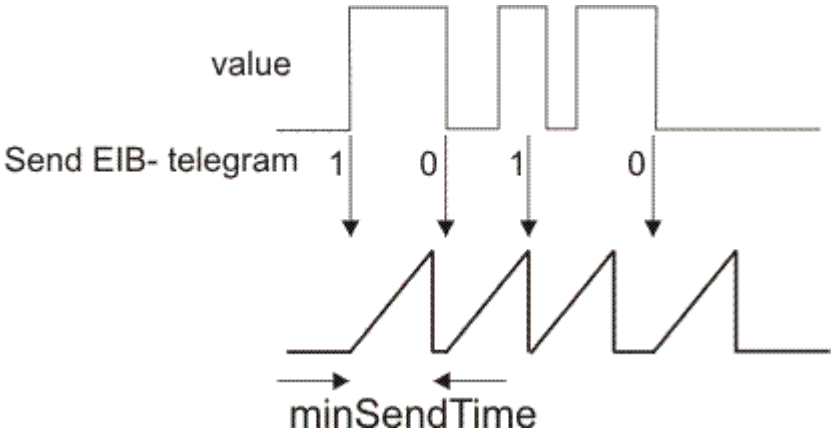
| Description | _Rec | _Send | | | | |
|---|---|---|---|---|---|---|
| | | _Send | First Cycle | Delta, min. Send Time | Polling | Auto/manual |
| EIB_BIT_MANUAL | no | yes [▶ 59] | no | no | no | Manual |
| EIB_BIT_CONTROL | yes [▶ 34] | yes [▶ 56] | no | 200 msec | no | Auto |
| EIB_3BIT_CONTROL | yes [▶ 30] | yes [▶ 43] | no | 200 msec | no | Auto |
| EIB_8BIT_SIGN | yes [▶ 32] | yes [▶ 49] | no | 1 sec | no | Auto |
| EIB_8BIT_SIGN_EX | no | yes [▶ 50] | yes | 1 sec, variable | 500 msec, variable | Auto/manual |
| EIB_8BIT_UNSIGN | yes [▶ 33] | yes [▶ 51] | no | 1 sec | no | Auto |
| EIB_8BIT_UNSIGN_EX | no | yes [▶ 52] | yes | 1 sec, variable | 500 msec, variable | Auto/manual |
| EIB_2OCTET_SIGN | yes [▶ 29] | yes [▶ 39] | no | 1 sec | no | Auto |
| EIB_2OCTET_SIGN_EX | no | yes [▶ 40] | yes | 1 sec, variable | 500 msec, variable | Auto/manual |
| EIB_2OCTET_UNSIGN | yes [▶ 30] | yes [▶ 41] | no | 1 sec | no | Auto |
| EIB_2OCTET_UNSIGN_EX | no | yes [▶ 42] | yes | 1 sec, variable | 500 msec, variable | Auto/manual |
| EIB_2OCTET_FLOAT | yes [▶ 29] | yes [▶ 37] | no | 1 sec | no | Auto |
| EIB_2OCTET_FLOAT_EX | no | yes [▶ 38] | yes | 1 sec, variable | 500 msec, variable | Auto/manual |
| EIB_TIME | yes [▶ 36] | yes [▶ 61] | yes | no | 5 min | Auto |
| EIB_DATE | yes [▶ 36] | yes [▶ 60] | yes | no | 5 min | Auto |
| EIB_4OCTET_SIGN | yes [▶ 31] | yes [▶ 46] | no | 1 sec | no | Auto |
| EIB_4OCTET_SIGN_EX | no | yes [▶ 47] | yes | 1 sec, variable | 500 msec, variable | Auto/manual |
| EIB_4OCTET_UNSIGN | yes [▶ 32] | yes [▶ 48] | no | 1 sec | no | Auto |
| EIB_4OCTET_FLOAT | yes [▶ 31] | yes [▶ 44] | no | 1 sec | no | Auto |
| EIB_4OCTET_FLOAT_EX | no | yes [▶ 45] | yes | 1 sec, variable | 10 min, variable | Auto/manual |
| EIB_READ | no | yes [▶ 60] | no | no | no | Manual |
| EIB_ALL_DATA_TYPES | yes [▶ 33] | yes [▶ 54] | no | 1 sec, variable | 100 msec, variable | Auto/manual |
| EIB_ALL_DATA_TYPES_EX | yes [▶ 34] | no | no | no | no | no |

*_Rec:* yes - receiving is supported, no - receiving is not supported

*_Send:* yes - sending is supported, no - sending is not supported

**First Cycle:** An EIB telegram is sent when the function block is called for the first time

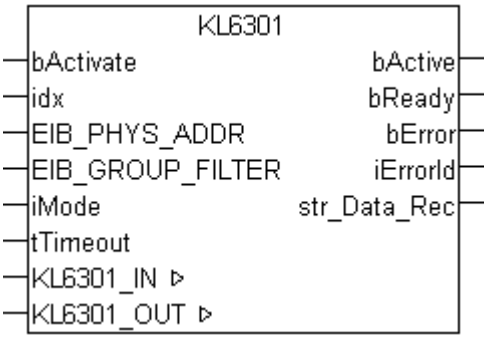
*Delta, min. Send Time:* An EIB telegram is only sent when the data is changed. The parameter "min. Send Time" is used to activate a "send filter". It does not extend the response time for the first relative change in input, but is active for subsequent changes. The min. send time (send delay time) describes the minimum interval required between sending of two telegrams. The min. send time reduces bus communication and ensures that send instructions from other function blocks can also transmit their EIB telegrams.




**Polling:** The data are automatically sent at the specified interval, even if the data did not change

*Auto/Manual:* Auto - Data is sent automatically when the function block is called, Manual - Data is only sent when requested by the function block

## 6.3.2 KL6301




This function block deals with the KL6301 EIB Bus Terminal communication. It is used for configuring the KL6301 and for starting the data exchange with the EIB network.

**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

```
bActivate          : BOOL;
idx                : INT := 1;
EIB_PHYS_ADDR      : EIB_PHYS_ADDR;
EIB_GROUP_FILTER   : ARRAY [1..8] OF EIB_GROUP_FILTER;
iMode              : INT;
tTimeout           : TIME := t#5s;
```

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

**idx:** If more than one Bus Terminal per PLC program is used, each KL6301 must have a unique idx number. Valid values 1...64.

**EIB_PHYS_ADDR:** Physical EIB address (see EIB_PHYS_ADDR [▶ 67]). The default address is 1.2.3. This address must be unique in the EIB network!

**EIB_GROUP_FILTER:** Group address filter (see EIB_GROUP_FILTER [▶ 66]). A maximum of 8 filters are possible.

**iMode:**
0 - for firmware B0 and higher - 4 Filter, each with 64 entries
1 - for firmware B1 and higher - 8 Filter, each with 32 entries
2 - for firmware B3 and higher - 8 Filter, each with 32 entries inverted
100 - for firmware B1 and higher - monitor function, all group address telegrams can be received. The telegrams are not acknowledged (no ACK is being sent). At monitor operation transmission is not possible.

**tTimeout:** Time that a send function block has to transmit an EIB telegram until a timeout is signaled.

### VAR_OUTPUT

```
bActive             : BOOL;
bReady              : BOOL;
bError              : BOOL;
iErrorId            : EIB_Error_Code;
str_Data_Rec        : EIB_REC;
```

**bActive:** The function block has been activated.

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

**bError:** The output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

**iErrorId:** An error code is available at this output in the event of an error (see EIB_ERROR_CODE [▶ 64]). *bError* goes TRUE at the same time.

**str_Data_Rec:** Data structure that is connected to the send and receive blocks (see EIB_REC [▶ 67]).

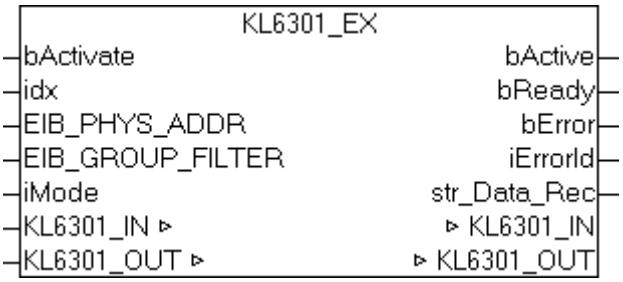
### VAR_IN_OUT

```
KL6301_IN           : ARRAY [1..24] OF BYTE;
KL6301_OUT          : ARRAY [1..24] OF BYTE;
```

**KL6301_IN:** Linked with the KL6301 input addresses.

**KL6301_OUT:** Linked with the KL6301 output addresses.

## 6.3.3 KL6301_EX




This function block deals with the KL6301 EIB Bus Terminal communication. It is used for configuring the KL6301 and for starting the data exchange with the EIB network.

BETA: ETS support for search and LED flashing.

**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

```
bActivate          : BOOL;
idx                : INT := 1;
EIB_PHYS_ADDR      : EIB_PHYS_ADDR;
EIB_GROUP_FILTER   : ARRAY [1..8] OF EIB_GROUP_FILTER;
iMode              : INT;
```

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

**idx:** If more than one Bus Terminal per PLC program is used, each KL6301 must have a unique idx number. Valid values 1...64.

**EIB_PHYS_ADDR:** Physical EIB address (see EIB_PHYS_ADDR [▶ 67]). The default address is 1.2.3. This address must be unique in the EIB network!

**EIB_GROUP_FILTER:** Group address filter (see EIB_GROUP_FILTER [▶ 66]). A maximum of 8 filters are possible.

**iMode:**
0 - for firmware B0 and higher - 4 Filter, each with 64 entries
1 - for firmware B1 and higher - 8 Filter, each with 32 entries
2 - for firmware B3 and higher - 8 Filter, each with 32 entries inverted
100 - for firmware B1 and higher - monitor function, all group address telegrams can be received. The telegrams are not acknowledged (no ACK is being sent). At monitor operation transmission is not possible.

### VAR_OUTPUT

```
bActive            : BOOL;
bReady             : BOOL;
bError             : BOOL;
iErrorId           : EIB_Error_Code;
str_Data_Rec       : EIB_REC;
```

**bActive:** The function block has been activated.

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

**bError:** The output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

**iErrorId:** An error code is available at this output in the event of an error (see EIB_ERROR_CODE [▶ 64]). *bError* goes TRUE at the same time.

**str_Data_Rec:** Data structure that is connected to the send and receive blocks (see EIB_REC [▶ 67]).

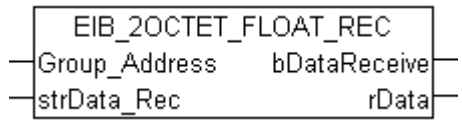
### VAR_IN_OUT

```
KL6301_IN          : ARRAY [1..24] OF BYTE;
KL6301_OUT         : ARRAY [1..24] OF BYTE;
```

**KL6301_IN:** Linked with the KL6301 input addresses.

**KL6301_OUT:** Linked with the KL6301 output addresses.

## 6.3.4 EIB_2OCTET_FLOAT_REC




This function block receives 2 bytes of float EIB data on the set group address and converts them into an IEC61131-3 REAL variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

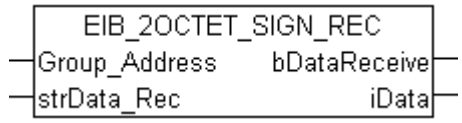
### VAR_OUTPUT

```
bDataReceive    : BOOL;
rData           : REAL;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**rData:** Contains the user data of the received EIB telegram.

## 6.3.5 EIB_2OCTET_SIGN_REC




This function block receives 2 bytes of sign EIB data on the set group address and converts them into an IEC61131-3 INT variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

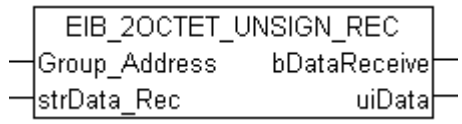
### VAR_OUTPUT

```
bDataReceive    : BOOL;
iData           : INT;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received

**iData:** Contains the user data of the received EIB telegram.

## 6.3.6 EIB_2OCTET_UNSIGN_REC




This function block receives 2 bytes of unsign EIB data on the set group address and converts them into an IEC61131-3 UINT variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bDataReceive    : BOOL;
uiData          : UINT;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**uiData:** Contains the user data of the received EIB telegram.

## 6.3.7 EIB_3BIT_CONTROL_REC


This function block receives 4 bits of EIB data on the set group address and converts them into an IEC61131-3 BOOL variable and a byte variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bDataReceive    : BOOL;
bControl        : BOOL;
byRange         : BYTE;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

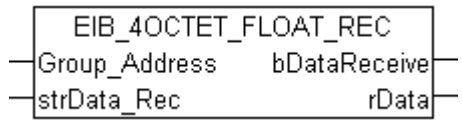
**bControl:** Valid values (TRUE/FALSE).

**byRange:** Valid values (000b..111b).

Allocation of the 4 bits to the variables bControl and byRange.

| | | | |
|---|---|---|---|
| bControl | byRange.2 | byRange.1 | byRange.0 |

## 6.3.8 EIB_4OCTET_FLOAT_REC




This function block receives 4 bytes of float EIB data on the set group address and converts them into an IEC61131-3 REAL variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

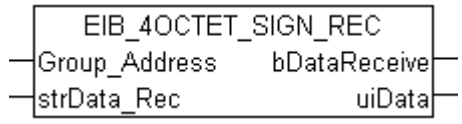
### VAR_OUTPUT

```
bDataReceive    : BOOL;
rData           : REAL;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**rData:** Contains the user data of the incoming EIB telegram.

## 6.3.9 EIB_4OCTET_SIGN_REC




This function block receives 4 bytes of sign EIB data on the set group address and converts them into an IEC61131-3 DINT variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

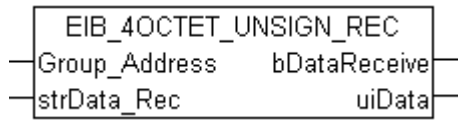
### VAR_OUTPUT

```
bDataReceive    : BOOL;
uiData          : DINT;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**uiData:** Contains the user data of the received EIB telegram.

## 6.3.10 EIB_4OCTET_UNSIGN_REC




This function block receives 4 bytes of unsign EIB data on the set group address and converts them into an IEC61131-3 UDINT variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

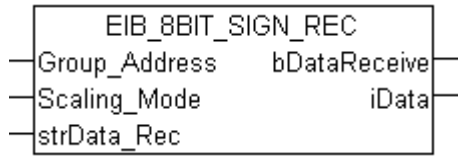
### VAR_OUTPUT

```
bDataReceive    : BOOL;
uiData          : UDINT;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**uiData:** Contains the user data of the received EIB telegram.

## 6.3.11 EIB_8BIT_SIGN_REC




This function block receives 8 bits of EIB data on the set group address and converts them into an IEC61131-3 INT variable. In addition the value may be converted automatically.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
Scaling_Mode    : INT;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**Scaling_Mode:**
0 - The 8 bit value is output as a percentage value 0...100%
1 - The 8 bit value is output as an angle 0...360°
2 - The 8 bit value is output as a byte value 0...255

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

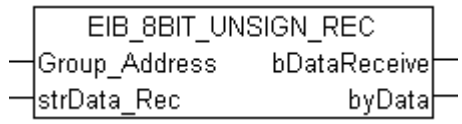
### VAR_OUTPUT

```
bDataReceive    : BOOL;
iData           : INT;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**iData:** Scaled value, see *Scaling_Mode* (-1 - an invalid scaling mode was entered).

## 6.3.12 EIB_8BIT_UNSIGN_REC




This function block receives 8 bits of EIB data on the set group address and converts them into an IEC61131-3 BYTE variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data is sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

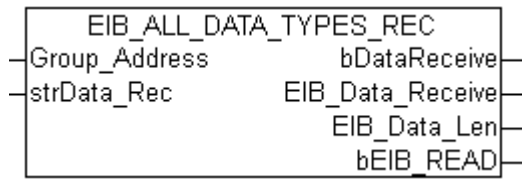
### VAR_OUTPUT

```
bDataReceive    : BOOL;
byData          : BYTE;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**byData:** Contains the user data of the received EIB telegram.

## 6.3.13 EIB_ALL_DATA_TYPES_REC




This function block receives variable EIB data sizes on the set group address and outputs the raw data as a byte ARRAY.

### VAR_INPUT

```
Group_Address      : EIB_GROUP_ADDR;
strData_Rec        : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bDataReceive       : BOOL;
EIB_Data_Receive   : ARRAY [1..14] OF BYTE;
EIB_Data_Len       : USINT;
bEIB_READ          : BOOL;
```

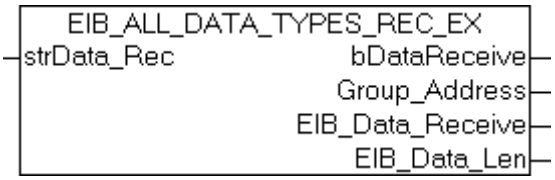
**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**EIB_Data_Receive:** Contains the user data of the received EIB telegram.

**EIB_Data_Len:** Contains the user data length of the incoming EIB telegram.
Data < 8 bit specified length 1
Data >=) 8 bit specified length +1
Example: If 1 bit of data is received, the length in EIB_Data_Len is 1. If 2 bytes of data are received, the length in EIB_Data_Len is 3.

**bEIB_READ:** TRUE = EIB read command. FALSE = normal EIB telegram (ab V5.2.5).

## 6.3.14 EIB_ALL_DATA_TYPES_REC_EX




This function block receives variable EIB data sizes of all group addresses and outputs the raw data as a byte ARRAY.

### VAR_INPUT

```
strData_Rec       : EIB_REC;
```

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bDataReceive      : BOOL;
Group_Address     : EIB_GROUP_ADDR;
EIB_Data_Receive  : ARRAY [1..14] OF BYTE;
EIB_Data_Len      : USINT;
```

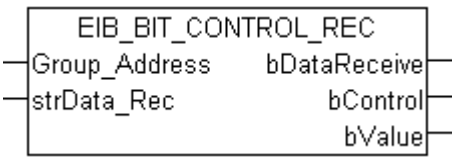
**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**EIB_Data_Receive:** Contains the user data of the received EIB telegram.

**EIB_Data_Len:** Contains the user data length of the incoming EIB telegram.
Data < 8 bit specified length 1
Data >=) 8 bit specified length +1
Example: If 1 bit of data is received, the length in EIB_Data_Len is 1. If 2 bytes of data are received, the length in EIB_Data_Len is 3.

## 6.3.15 EIB_BIT_CONTROL_REC




This function block receives 2 bits of EIB data on the set group address and converts them into two IEC61131-3 BOOL variables.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bDataReceive    : BOOL;
bControl        : BOOL;
bValue          : BOOL;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

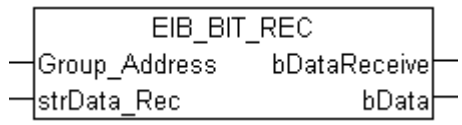
**bControl:** Valid values (TRUE/FALSE).

**bValue:** Valid values (TRUE/FALSE).

Allocation of the 2 bits to the variables bControl and bValue.

| bControl | bValue |
|---|---|

## 6.3.16 EIB_BIT_REC




This function block receives 1 bit of EIB data on the set group address and converts them into an IEC61131-3 BOOL variable.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

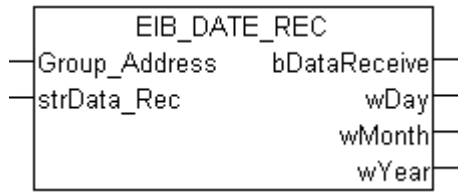
### VAR_OUTPUT

```
bDataReceive    : BOOL;
bData           : BOOL;
```

**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**bData:** Valid values (TRUE/FALSE).

## 6.3.17 EIB_DATE_REC




This function block receives 3 bytes of EIB data on the set group address and converts them into three IEC61131-3 WORD variables.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bDataReceive    : BOOL;
wDay            : WORD;
wMonth          : WORD;
wYear           : WORD;
```

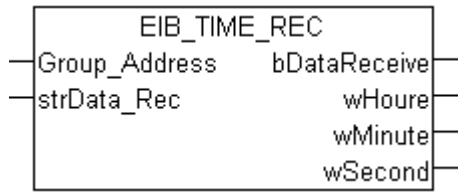
**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**wDay:** date, days [1...31].

**wMonth:** date, month [1...12].

**wYear:** date, year [0...99].

## 6.3.18 EIB_TIME_REC




### Application

This function block receives 3 bytes of EIB data on the set group address and converts them into three IEC61131-3 WORD variables.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
strData_Rec     : EIB_REC;
```

**Group_Address:** Group address from which the data were sent (see EIB_GROUP_ADDR [▶ 66]). This group address must be entered in the filters!

**strData_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bDataReceive    : BOOL;
wHoure          : WORD;
wMinute         : WORD;
wSecond         : WORD;
```

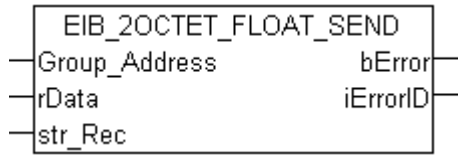
**bDataReceive:** This bit is set to FALSE for exactly one cycle when an EIB telegram with the group address is received.

**wHoure:** Time, hours [0...23].

**wMinute:** Time, minutes [0...59].

**wSecond:** Time, seconds [0...59].

## 6.3.19 EIB_2OCTET_FLOAT_SEND




This function block sends a 2-byte float EIB value to the set group address. An IEC61131-3 real value is available as input value. The data are only transferred if there is a change. If the value changes again within 1 second, new data are only sent to the EIB device after another second has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
rData           : REAL;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**rData:** The data value in REAL. This is automatically converted to an EIB 2OCTET FLOAT value.

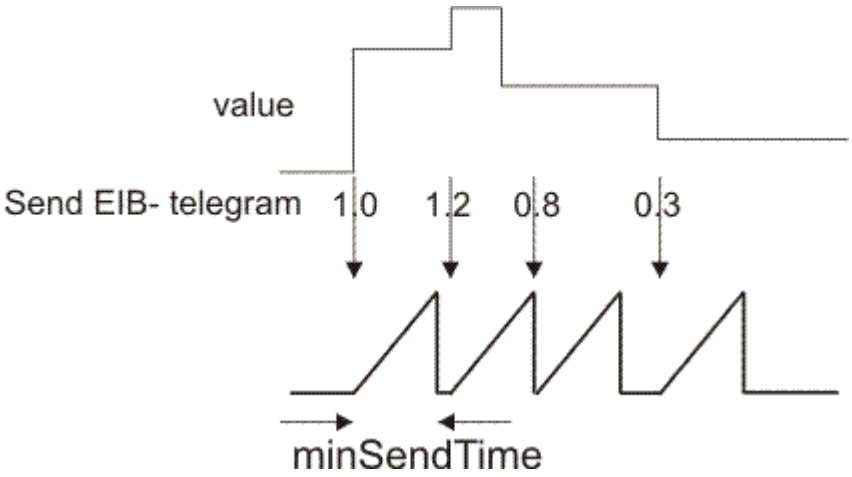
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

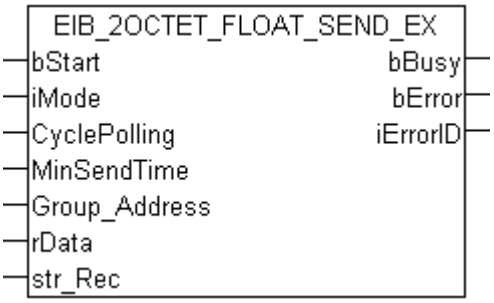
```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.20 EIB_2OCTET_FLOAT_SEND_EX




This function block sends a 2-byte float EIB value to the set group address. An IEC61131-3 real value is available as input value. In dependence of the mode (*iMode*) the data can be sent manually, by polling or on change.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
CyclePolling    : TIME := t#500ms;
MinSendTime     : TIME := t#1s;
Group_Address   : EIB_GROUP_ADDR;
rData           : REAL;
str_Rec         : EIB_REC;
bEnableReadReq  : BOOL;
```

**bStart:** Activates the block. The block starts to work in dependence of the parameterized mode (see *iMode*).

**iMode:**
0 - At rising edge of *bStart* an EIB telegram is sent. If the output *bBusy* is FALSE again then the command was executed.
1 - Polling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling*.
2 - OnChange Mode: If *bStart* is TRUE at change of data an EIB telegram is sent automatically. With *MinSendTime* the minimum interval between two EIB messages can be parameterized to avoid unnecessary load to the EIB network.
3 - OnChangePolling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling* or automatically at change of data. The minimum interval between two EIB messages is set with *MinSendTime*.

**CyclePolling:** Polling time for *iMode* = 1 (polling mode). The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**rData:** The data value in REAL. This is automatically converted to an EIB 2OCTET FLOAT value.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**bEnableReadReq:** Allows the execution of read commands.

### VAR_OUTPUT

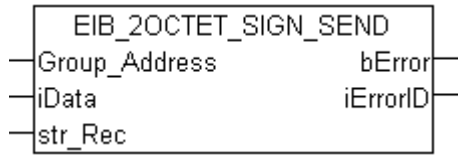
```
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.21 EIB_2OCTET_SIGN_SEND




This function block sends a 2-byte sign EIB value to the set group address. An IEC61131-3 INT value is available as input value. The data are only transferred if there is a change. If the value changes again within 1 second, new data are only sent to the EIB device after another second has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address  : EIB_GROUP_ADDR;
iData          : INT;
str_Rec        : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**iData:** The data value in INT. This is automatically converted to an EIB 2OCTET SIGN value.

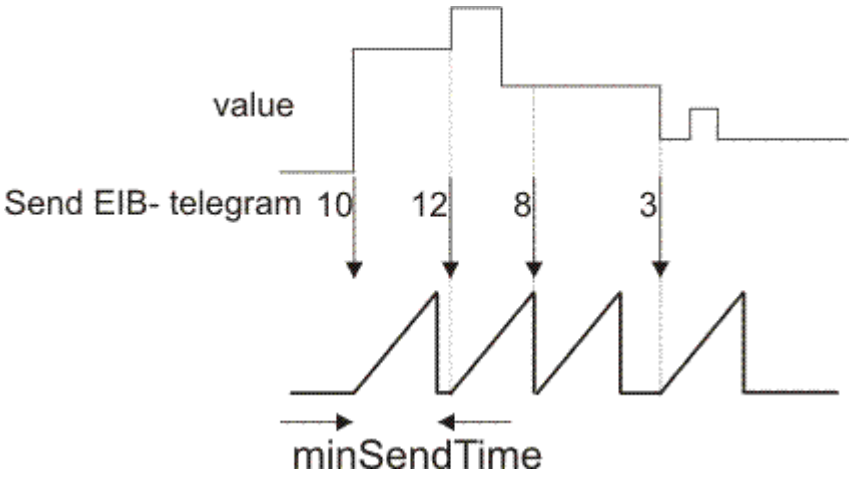
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

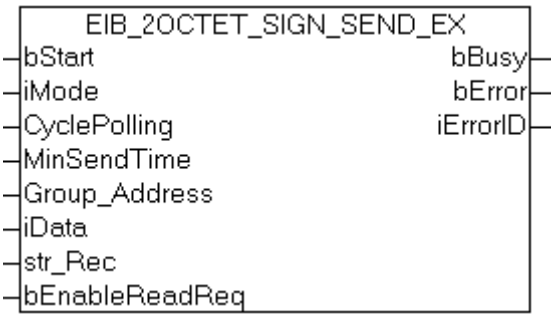
```
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.22 EIB_2OCTET_SIGN_SEND_EX




This function block sends a 2-byte sign EIB value to the set group address. An IEC61131-3 int value is available as input value. In dependence of the mode (*iMode*) the data can be sent manually, by polling or on change.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
CyclePolling    : TIME := t#500ms;
MinSendTime     : TIME := t#1s;
Group_Address   : EIB_GROUP_ADDR;
iData           : INT;
str_Rec         : EIB_REC;
bEnableReadReq : BOOL;
```

**bStart:** Activates the block. The block starts to work in dependence of the parameterized mode (see *iMode*).

**iMode:**
0 - At rising edge of *bStart* an EIB telegram is sent. If the output *bBusy* is FALSE again then the command was executed.
1 - Polling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling*.
2 - OnChange Mode: If *bStart* is TRUE at change of data an EIB telegram is sent automatically. With *MinSendTime* the minimum interval between two EIB messages can be parameterized to avoid unnecessary load to the EIB network.
3 - OnChangePolling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling* or automatically at change of data. The minimum interval between two EIB messages is set with *MinSendTime*.

**CyclePolling:** Polling time for *iMode* = 1 (polling mode). The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**iData:** The data value in INT. This is automatically converted to an EIB 2OCTET SIGN value.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**bEnableReadReq:** Allows the execution of read commands.

### VAR_OUTPUT

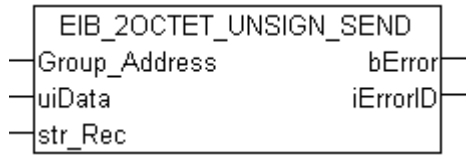
```
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.23 EIB_2OCTET_UNSIGN_SEND




This function block sends a 2-byte unsign EIB value to the set group address. An IEC61131-3 UINT value is available as input value. The data are only transferred if there is a change. If the value changes again within 1 second, new data are only sent to the EIB device after another second has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address  : EIB_GROUP_ADDR;
uiData         : UINT;
str_Rec        : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**uiData:** The data value in UINT. This is automatically converted to an EIB 2OCTET UNSIGN value.

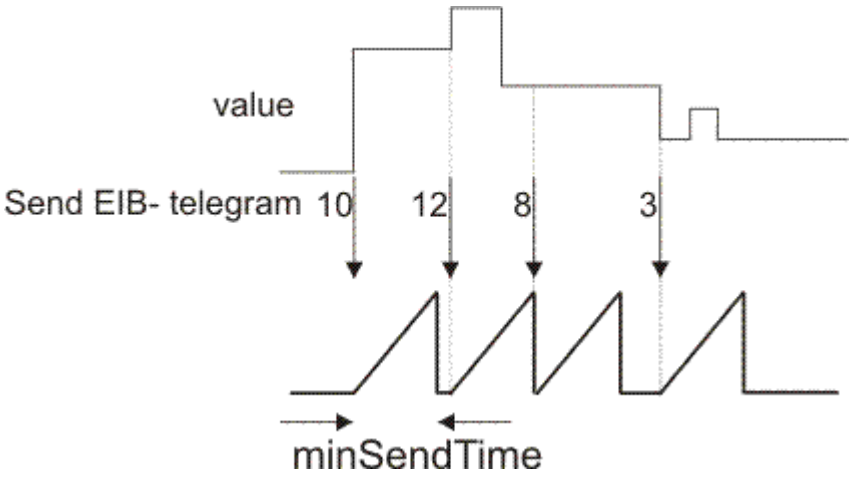
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

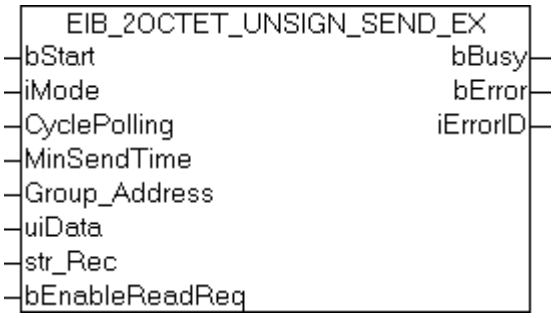
```
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.24 EIB_2OCTET_UNSIGN_SEND_EX




This function block sends a 2-byte Unsign EIB value to the set group address. An IEC61131-3 UINT value is available as input value. The data can be sent in Manual, Polling or OnChange depending on the set mode (*iMode*).

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
CyclePolling    : TIME := t#500ms;
MinSendTime     : TIME := t#1s;
Group_Address   : EIB_GROUP_ADDR;
uiData          : UINT;
str_Rec         : EIB_REC;
bEnableReadReq : BOOL;
```

**bStart:** Activates the function block, so that the function block starts to work depending on the set mode (see *iMode*).

**iMode:**
0 - With a positive edge at *bStart*, an EIB telegram is sent. If the output *bBusy* is FALSE again, the command is processed.
1 - Polling mode: If *bStart* is TRUE, EIB telegrams are sent at intervals of *CyclePolling*.
2 - OnChange mode: If *bStart* is TRUE, an EIB telegram is automatically sent when the data changes. *MinSendTime* can be used to parameterize the minimum interval between two EIB messages, in order to avoid excessive EIB network load.
3 - OnChangePolling mode: If *bStart* is TRUE, EIB telegrams are sent at intervals of *CyclePolling* or automatically when the data changes. The minimum interval between two EIB messages is set with *MinSendTime*.

**CyclePolling:** Polling time for *iMode* = 1 (Polling mode). The minimum time is 200 ms.

**MinSendTime:** Minimum interval time, which has to elapse before a telegram is sent in OnChange mode. The minimum time is 200 ms.

**Group_Address:** Group address to which the data is sent (see EIB_GROUP_ADDR [▶ 66]).

**uiData:** The data value in UINT; it is automatically converted into an EIB 2OCTET UNSIGN value.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**bEnableReadReq:** Enables the execution of read commands.

### VAR_OUTPUT

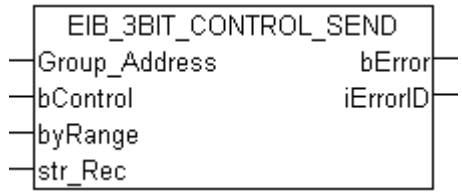
```
bBusy           : BOOL;
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bBusy:** The function block is active. Wait for new functions, until *bBusy* is FALSE again.

**bError:** The output becomes TRUE as soon as an error occurs. This error is described via the *iErrorID* variable.

**iErrorID:** An error code is available at this output in the event of an error (see EIB_ERROR_CODE [▶ 64]). *bError* goes TRUE at the same time.

## 6.3.25 EIB_3BIT_CONTROL_SEND




This function block sends a 4-bit EIB value to the set group address. An IEC61131-3 BOOL and a BYTE value are available as input value. The data are only transferred if there is a change in one of the two data types. If the value changes again within 200 milliseconds, new data are only sent to the EIB device after another 200 millisecond has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
bControl        : BOOL;
byRange         : BYTE;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**bControl:** Range of values TRUE/FALSE.

**byRange:** Range 000b..111b.

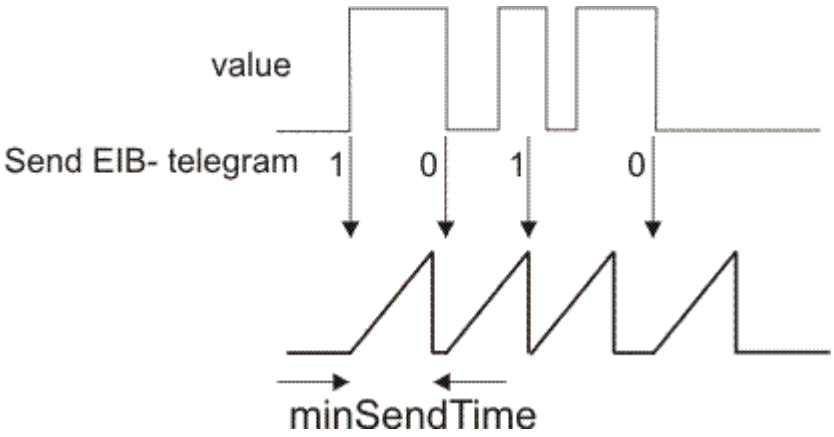
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

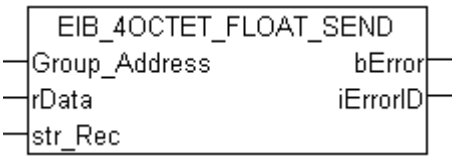
```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.26 EIB_4OCTET_FLOAT_SEND




This function block sends a 4-byte float EIB value to the set group address. An IEC61131-3 REAL value is available as input value. The data are only transferred if there is a change. If the value changes again within 1 second, new data are only sent to the EIB device after another second has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
rData           : REAL;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**rData:** The data value in REAL. This is automatically converted to an EIB 2OCTET FLOAT value.

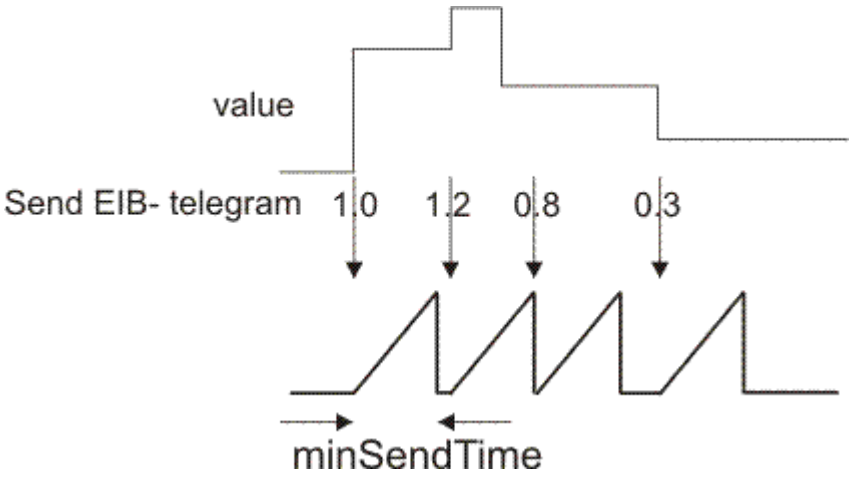
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

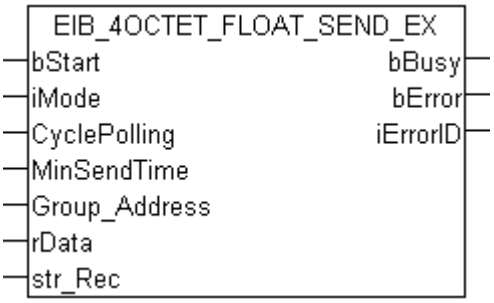
```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.27 EIB_4OCTET_FLOAT_SEND_EX




This function block sends a 4-byte float EIB value to the set group address. An IEC61131-3 real value is available as input value. In dependence of the mode (*iMode*) the data can be sent manually, by polling or on change.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
CyclePolling    : TIME := t#10m;
MinSendTime     : TIME := t#1s;
Group_Address   : EIB_GROUP_ADDR;
rData           : REAL;
str_Rec         : EIB_REC;
bEnableReadReq  : BOOL;
```

**bStart:** Activates the block. The block starts to work in dependence of the parameterized mode (see *iMode*).

**iMode:**
0 - At rising edge of *bStart* an EIB telegram is sent. If the output *bBusy* is FALSE again then the command was executed.
1 - Polling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling*.
2 - OnChange Mode: If *bStart* is TRUE at change of data an EIB telegram is sent automatically. With *MinSendTime* the minimum interval between two EIB messages can be parameterized to avoid unnecessary load to the EIB network.
3 - OnChangePolling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling* or automatically at change of data. The minimum interval between two EIB messages is set with *MinSendTime*.

**CyclePolling:** Polling time for the polling mode. The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**rData:** The data value in REAL. This is automatically converted to an EIB 2OCTET FLOAT value.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**bEnableReadReq:** Allows the execution of read commands.

### VAR_OUTPUT

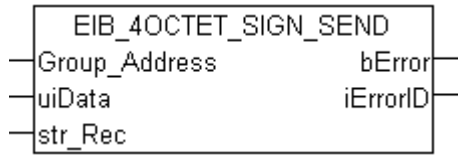
```
bBusy           : BOOL;
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.28 EIB_4OCTET_SIGN_SEND




This function block sends a 4-byte sign EIB value to the set group address. An IEC61131-3 DINT value is available as input value. The data are only transferred if there is a change. If the value changes again within 1 second, new data are only sent to the EIB device after another second has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
uiData          : DINT;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**uiData:** The data value in DINT. This is automatically converted to an EIB 4OCTET SIGN value.

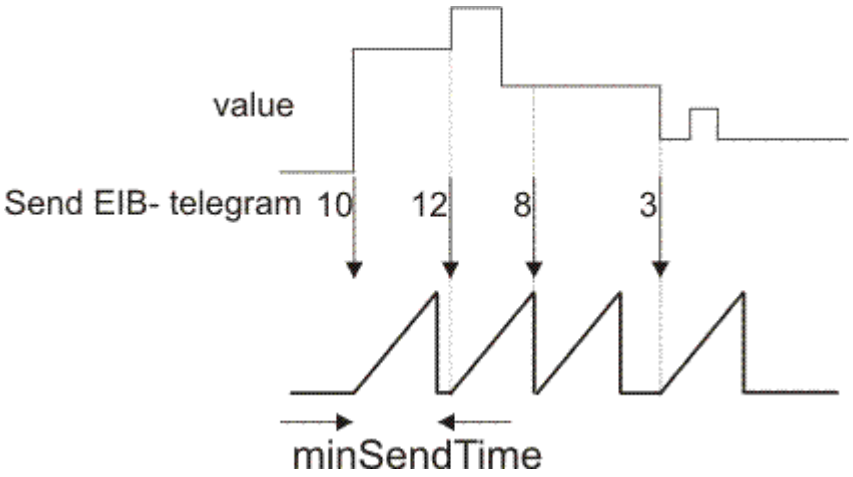
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

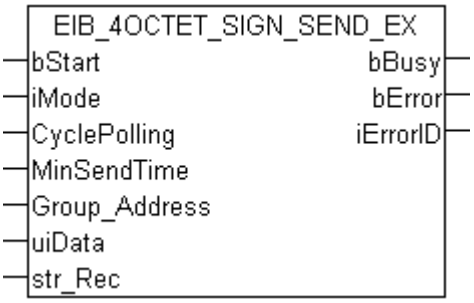
```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.29 EIB_4OCTET_SIGN_SEND_EX




This function block sends a 4-byte sign EIB value to the set group address. An IEC61131-3 DINT value is available as input value. In dependence of the mode (*iMode*) the data can be sent manually, by polling or on change.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
CyclePolling    : TIME := t#500ms;
MinSendTime     : TIME := t#1s;
Group_Address   : EIB_GROUP_ADDR;
uiData          : DINT;
str_Rec         : EIB_REC;
bEnableReadReq  : BOOL;
```

**bStart:** Activates the block. The block starts to work in dependence of the parameterized mode (see *iMode*).

**iMode:**
0 - At rising edge of *bStart* an EIB telegram is sent. If the output *bBusy* is FALSE again then the command was executed.
1 - Polling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling*.
2 - OnChange Mode: If *bStart* is TRUE at change of data an EIB telegram is sent automatically. With *MinSendTime* the minimum interval between two EIB messages can be parameterized to avoid unnecessary load to the EIB network.
3 - OnChangePolling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling* or automatically at change of data. The minimum interval between two EIB messages is set with *MinSendTime*.

**CyclePolling:** Polling time for the polling mode. The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**uiData:** The data value in DINT. This is automatically converted to an EIB 4OCTET SIGN value.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**bEnableReadReq:** Allows the execution of read commands.

### VAR_OUTPUT

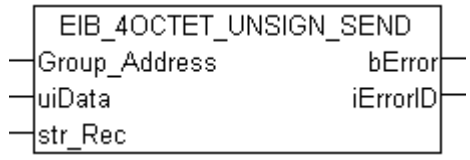
```
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.30 EIB_4OCTET_UNSIGN_SEND




This function block sends a 4-byte unsign EIB value to the set group address. An IEC61131-3 UDINT value is available as input value. The data are only transferred if there is a change. If the value changes again within 1 second, new data are only sent to the EIB device after another second has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address  : EIB_GROUP_ADDR;
uiData         : UDINT;
str_Rec        : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**uiData:** The data value in UDINT. This is automatically converted to an EIB 4OCTET SIGN value.

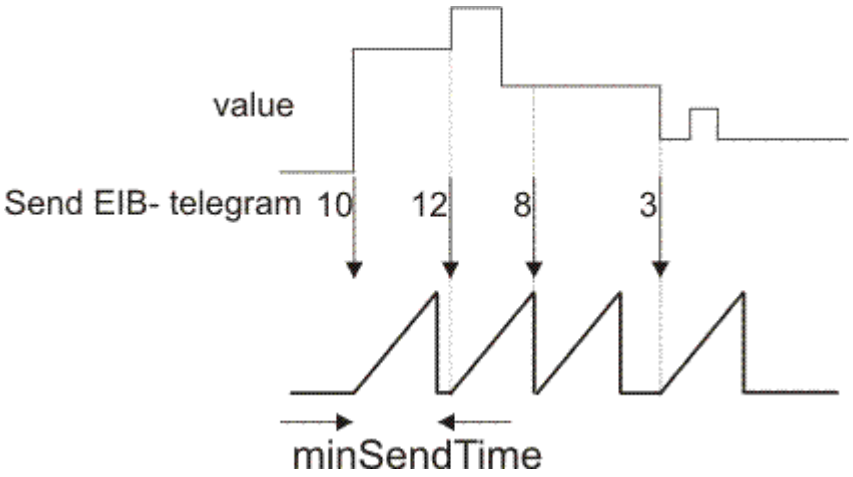
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.31 EIB_8BIT_SIGN_SEND


This function block sends a 8-bit EIB value to the set group address. An IEC61131-3 INT is available as input value. Scaling_Mode can be used to scale the input data value. The data are only transferred if there is a change in the data value. If the value changes again within 1 second, new data are only sent to the EIB device after minSendTime has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
iData           : INT;
Scaling_Mode    : INT;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**iData:** Data to be sent. Range of values depending on *Scaling_Mode*.

**Scaling_Mode:**
0 - 0...100 [%]
1 - 0...360 [°]
2 - 0...255

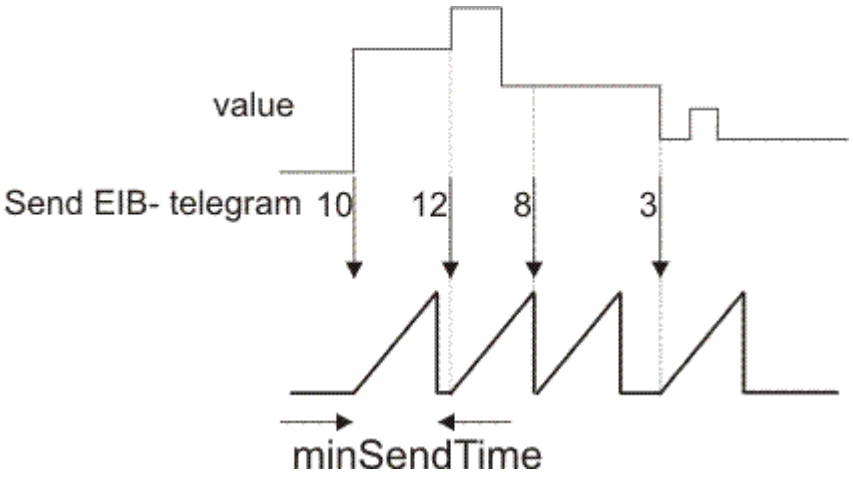
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

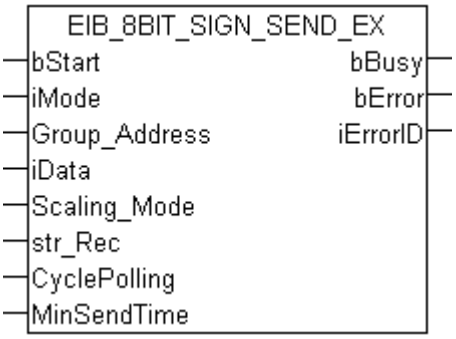
```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.32 EIB_8BIT_SIGN_SEND_EX




This function block sends an 8 Bit EIB value to the parameterized group address. As entry value there is an IEC61131-3 INT. By *Scaling_Mode* the entry data value can be scaled. In dependence of the mode (*iMode*) the data can be sent manually, by polling or on change.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
Group_Address   : EIB_GROUP_ADDR;
iData           : INT;
Scaling_Mode    : INT;
str_Rec         : EIB_REC;
CyclePolling    : TIME := t#500ms;
MinSendTime     : TIME := t#1s;
bEnableReadReq : BOOL;
```

**bStart:** Activates the block. The block starts to work in dependence of the parameterized mode (see *iMode*).

**iMode:**
0 - At rising edge of *bStart* an EIB telegram is sent. If the output *bBusy* is FALSE again then the command was executed.
1 - Polling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling*.
2 - OnChange Mode: If *bStart* is TRUE at change of data an EIB telegram is sent automatically. With *MinSendTime* the minimum interval between two EIB messages can be parameterized to avoid unnecessary load to the EIB network.
3 - OnChangePolling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling* or automatically at change of data. The minimum interval between two EIB messages is set with *MinSendTime*.

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**iData:** Data that has to be sent. The value range depends on the *Scaling_Mode*.

**Scaling_Mode:**
0 - 0...100 [%]
1 - 0...360 [°]
2 - 0...255

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**CyclePolling:** Polling time for the polling mode. The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**bEnableReadReq:** Allows the execution of read commands.

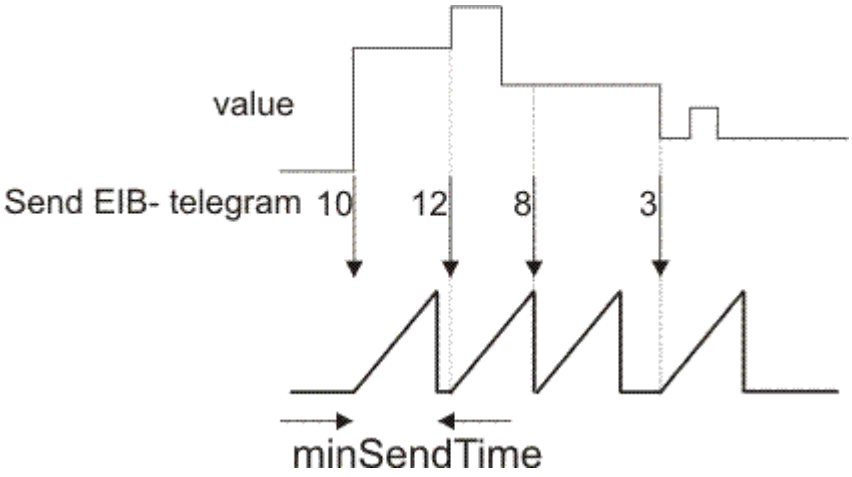
### VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

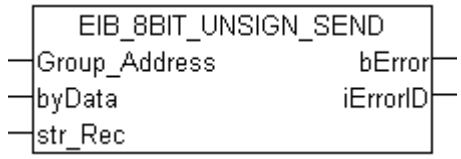
**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.33 EIB_8BIT_UNSIGN_SEND




This function block sends a 8-bit EIB value to the set group address. An IEC61131-3 byte variable is available as input value. The data are only transferred if there is a change in the data value. If the value changes again within 1 second, new data are only sent to the EIB device after minSendTime has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
byData          : BYTE;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**byData:** Data to be sent. Range of values 0x00...0xFF.

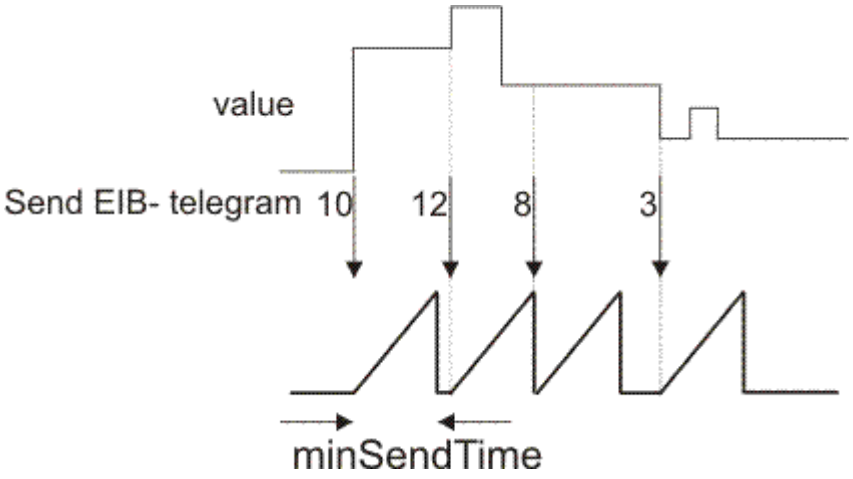
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

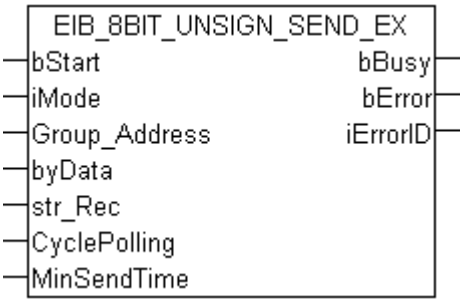
```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.34 EIB_8BIT_UNSIGN_SEND_EX




This function block sends an 8 Bit EIB value to the parameterized group address. As entry value there is an IEC61131--3 Byte-Variable. In dependence of the mode (*iMode*) the data can be sent manually, by polling or on change.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
Group_Address   : EIB_GROUP_ADDR;
byData          : BYTE;
str_Rec         : EIB_REC;
CyclePolling    : TIME := t#500ms;
MinSendTime     : TIME := t#1s;
bEnableReadReq  : BOOL;
```

**bStart:** Activates the block. The block starts to work in dependence of the parameterized mode (see *iMode*).

**iMode:**
0 - At rising edge of *bStart* an EIB telegram is sent. If the output *bBusy* is FALSE again then the command was executed.
1 - Polling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling*.
2 - OnChange Mode: If *bStart* is TRUE at change of data an EIB telegram is sent automatically. With *MinSendTime* the minimum interval between two EIB messages can be parameterized to avoid unnecessary load to the EIB network.
3 - OnChangePolling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling* or automatically at change of data. The minimum interval between two EIB messages is set with *MinSendTime*.

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**byData:** Data that has to be sent. Value range 0x00...0xFF.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**CyclePolling:** Polling time for the polling mode. The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**bEnableReadReq:** Allows the execution of read commands.

### VAR_OUTPUT

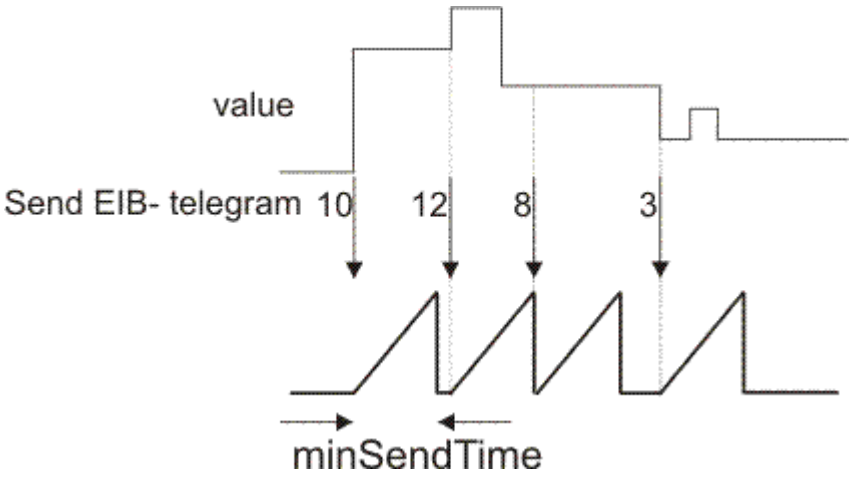
```
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.

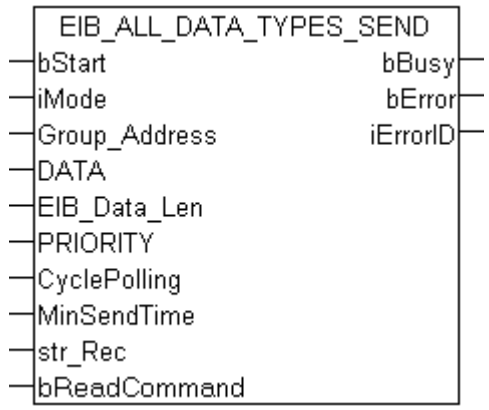
**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

OnChange Mode:



## 6.3.35 EIB_ALL_DATA_TYPES_SEND




This function block sends a freely selectable EIB value to the set group address. An IEC61131-3 byte ARRAY variable is available as input value. The data are sent depending on the set mode.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
Group_Address   : EIB_GROUP_ADDR;
DATA            : ARRAY [1..14] OF OF BYTE;
EIB_Data_Len    : USINT := 1;
PRIORITY        : EIB_PRIORITY := EIB_PRIORITY_LOW;
CyclePolling    : TIME := t#100ms;
MinSendTime     : TIME := t#1s;
str_Rec         : EIB_REC;
bReadCommand    : BOOL;
```

**bStart:** If the mode is set to 0, an EIB telegram with rising edge is sent to bStart .

**iMode:**
0 - manual (Fig. 1)
1 - polling (Fig. 2)
2 - OnChange (Fig. 3)

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**DATA:** EIB data value.

**EIB_Data_Len:** Length of EIB data,
EIB values >=) 1 byte: use length +1,
EIB values < 1 byte: use length = 1

**PRIORITY:** EIB priority, low, high, alarm.

**CyclePolling:** Polling time for the polling mode. The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**bReadCommand:** An response to a EIB READ COMMAND can be sent.

### VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bBusy:** If the bit is set, the block is still active. As long as the *bBusy* bit is set, no new data can be transferred!

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

**Transfer mode**

**Mode 0 manual**

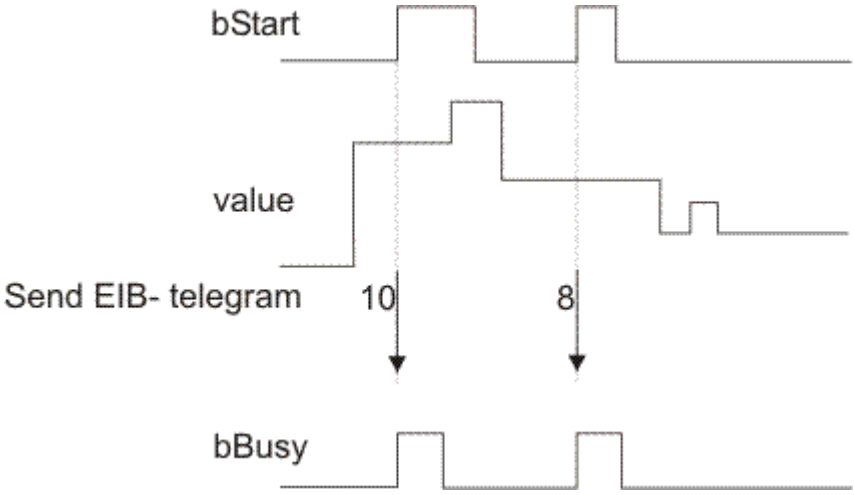



Figure 1

**Mode 1 Polling**

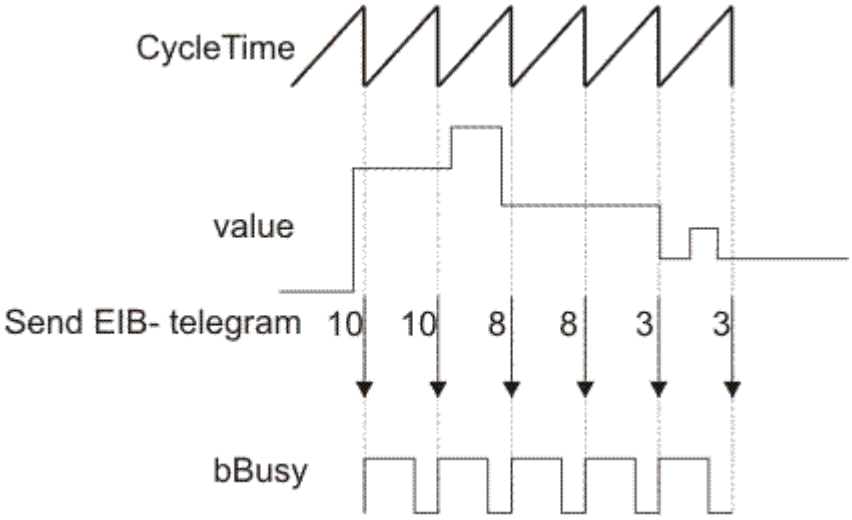



Figure 2

**Mode 2 OnChange**

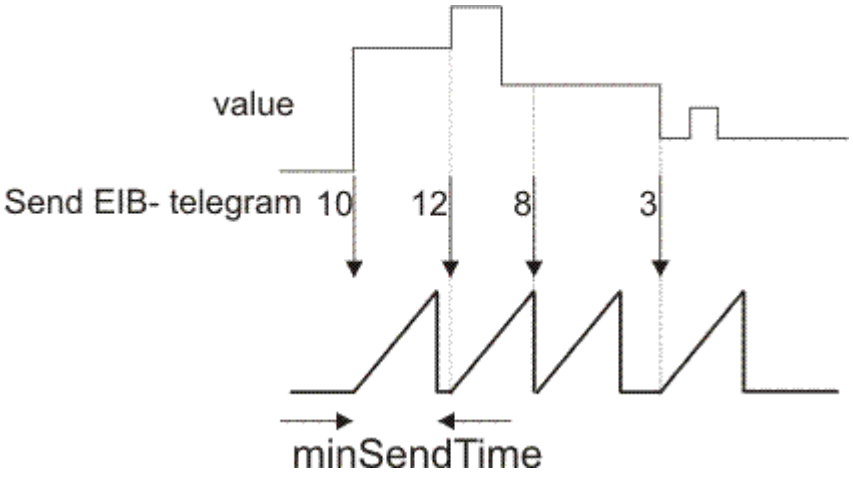
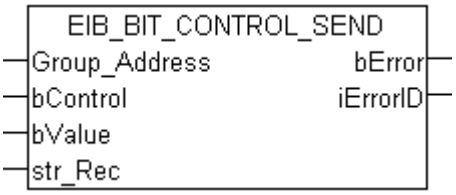



Figure 3

## 6.3.36 EIB_BIT_CONTROL_SEND




This function block sends a 2-bit EIB value to the set group address. Two IEC61131-3 BOOL variables are available as input values. The data are only transferred if there is a change in one of the two data types. If the value changes again within 200 milliseconds, new data are only sent to the EIB device after another 200 millisecond has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address  : EIB_GROUP_ADDR;
bControl       : BOOL;
bValue         : BOOL;
str_Rec        : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**bControl:** Range of values TRUE/FALSE.

**bValue:** Range of values TRUE/FALSE.

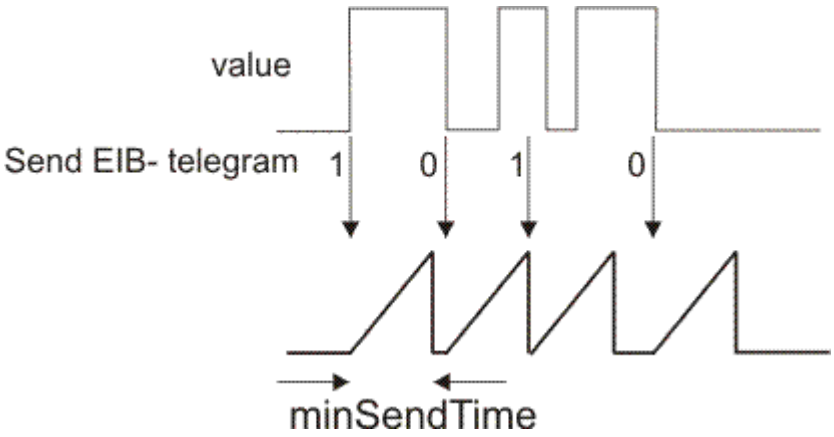
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs.

**iErrorID:** The *iErrorID* output issues an error code when an error occurs (EIB_ERROR_CODE [▶ 64])




## 6.3.37 EIB_BIT_SEND


This function block sends a 1-bit EIB value to the set group address. An IEC61131-3 BOOL variable is available as input value. The data are only transferred if there is a change in the data value. If the value changes again within 200 milliseconds, new data are only sent to the EIB device after another 200 millisecond has passed (see diagram). No new EIB telegram is sent if the value changes within the "min. send time" but falls back to the old, already sent value within the "min. send time".

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
bData           : BOOL;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**bData:** Range of values TRUE/FALSE.

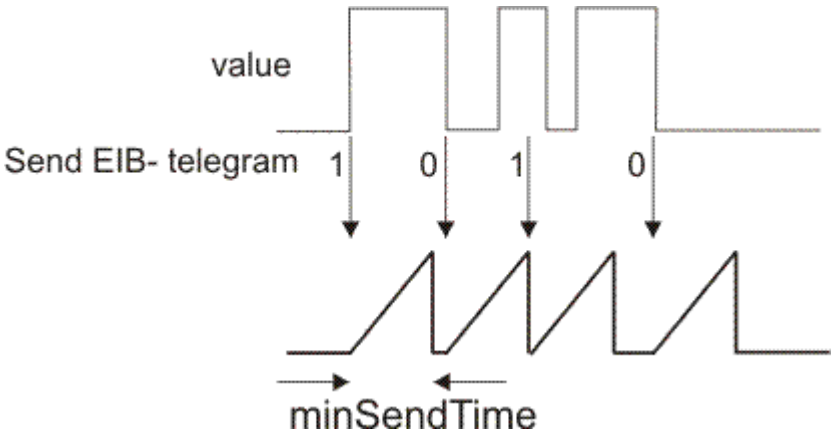
**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

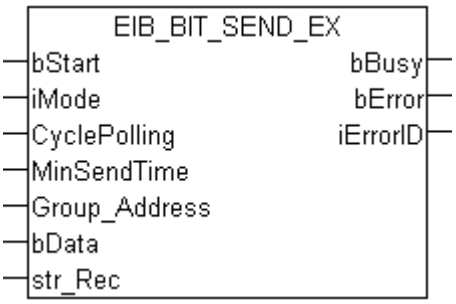
```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.




## 6.3.38 EIB_BIT_SEND_EX




This function block sends an 1 Bit EIB value to the parameterized group address. In dependence of the mode (*iMode*) the data can be sent manually, by polling or on change.

### VAR_INPUT

```
bStart          : BOOL;
iMode           : INT;
CyclePolling    : TIME := t#10s;
MinSendTime     : TIME := t#1s;
Group_Address   : EIB_GROUP_ADDR;
bData           : BOOL;
str_Rec         : EIB_REC;
bEnableReadReq : BOOL;
```

**bStart:** Activates the block. The block starts to work in dependence of the parameterized mode (see *iMode*).

**iMode:**
0 - At rising edge of *bStart* an EIB telegram is sent. If the output *bBusy* is FALSE again then the command was executed.
1 - Polling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling*.
2 - OnChange Mode: If *bStart* is TRUE at change of data an EIB telegram is sent automatically. With *MinSendTime* the minimum interval between two EIB messages can be parameterized to avoid unnecessary load to the EIB network.
3 - OnChangePolling Mode: If *bStart* is TRUE EIB telegrams are sent with a time interval of *CyclePolling* or automatically at change of data. The minimum interval between two EIB messages is set with *MinSendTime*.

**CyclePolling:** Polling time for the polling mode. The minimum time is 200ms.

**MinSendTime:** Interval time that has to be last at minimum until another telegram is changed in OnChange mode. The minimum time is 200ms.

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**bData:** Range of values TRUE/FALSE.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

**bEnableReadReq:** Allows the execution of read commands.

### VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.39 EIB_BIT_SEND_MANUAL


This function block sends a 1-bit EIB value to the parameterized group address. As entry value there is an EC61131-3 BOOL-Variable. The data is sent at a rising edge of bSend. As long as the block is active, the bit bBusy is set. bBusy is set back to FALSE if the EIB command is sent or an error occurs. An error is displayed by setting the variable bError. In this case the error code is given in iErrorID.

### VAR_INPUT

```
Group_Address  : EIB_GROUP_ADDR;
bSend          : BOOL;
bData          : BOOL;
str_Rec        : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**bSend:** Positive edge sends EIB telegram.

**bData:** Range of values TRUE/FALSE.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

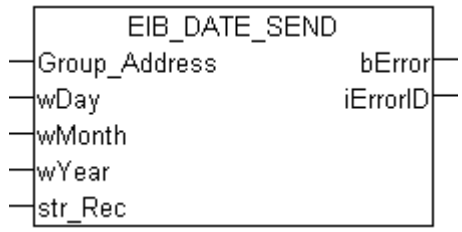
```
bBusy          : BOOL;
bError         : BOOL;
iErrorID       : EIB_ERROR_CODE;
```

**bBusy:** If the block is active bBusy is TRUE, if the EIB telegram is sent it is set back to FALSE.

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.40 EIB_DATE_SEND




This function block sends a 3-byte EIB value to the set group address. Three IEC61131-3 word variables are available as input values. The data are sent when the block is called for the first time the and then every 5 minutes.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
wDay            : WORD;
wMonth          : WORD;
wYear           : WORD;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**wDay:** Range of values 1...31.

**wMonth:** Range of values 1...12.

**wYear:** Range of values 0...99. If a value greater 2000 entered, 2000 is automatically subtracted. For the year 2005, for example, only the 5 is transferred to the EIB node.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

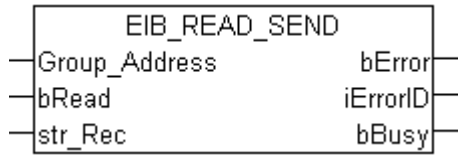
### VAR_OUTPUT

```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.41 EIB_READ_SEND




### Application

This function block sends a *Read_Group_Req* to the set group address. For receiving a *Read_Group_Res* the group address filter of the KL6301 must be parameterized accordingly.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
bRead           : BOOL;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data are sent (see EIB_GROUP_ADDR [▶ 66]).

**bRead:** Rising edge starts the block and sends a *Read_Group_Req* to the EIB device.

**For receiving a response the group address must be entered in the filter!**

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

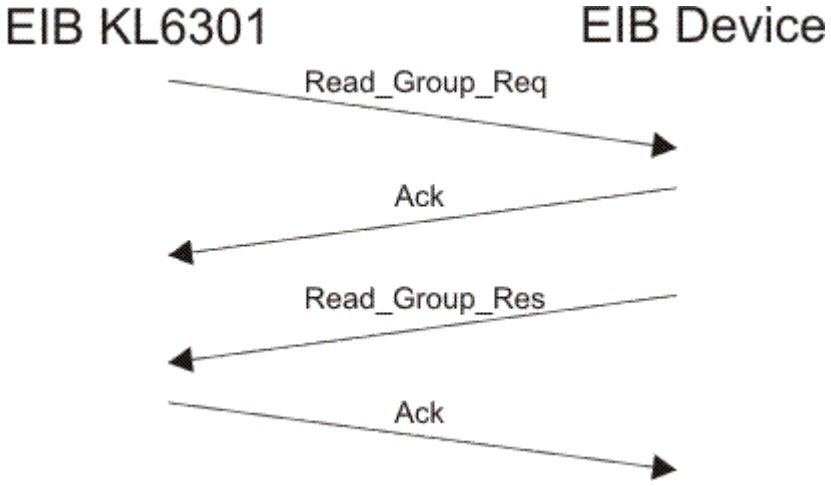
### VAR_OUTPUT

```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
bBusy           : BOOL;
```

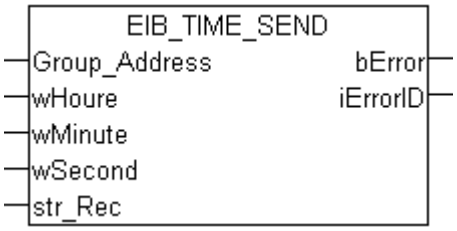
**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

**bBusy:** The block is active. For new functions wait until *bBusy* is set back to FALSE.




## 6.3.42 EIB_TIME_SEND




### Application

This function block sends a 3-byte EIB value to the set group address. Three IEC61131-3 word variables are available as input values. The data are sent when the block is called for the first time the and then every 5 minutes.

### VAR_INPUT

```
Group_Address   : EIB_GROUP_ADDR;
wHoure          : WORD;
wMinute         : WORD;
wSecond         : WORD;
str_Rec         : EIB_REC;
```

**Group_Address:** Group address to which the data is sent (see EIB_GROUP_ADDR [▶ 66]).

**wHoure:** Value range 0..23.

**wMinute:** Value range 0..59.

**wSecond:** Value range 0..59.

**str_Rec:** The data structure with which the KL6301() [▶ 26] function block must be linked (see EIB_REC [▶ 67]).

### VAR_OUTPUT

```
bError          : BOOL;
iErrorID        : EIB_ERROR_CODE;
```

**bError:** The *bError* output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorID*.

**iErrorID:** The output issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *bError* is TRUE.

## 6.3.43 Error codes

| Value (hex) | Value (dec) | Value (enum) | Description |
|---|---|---|---|
| 0x0000 | 0 | NO_EIB_ERROR | No error. |
| 0x0001 | 1 | WRONG_EIB_PHYS_ADDR | Outdated, no longer used. |
| 0x0002 | 2 | WRONG_EIB_GROUP_ADDR | The input variable *EIB_GROUP_FILTER.GROUP_ADDR* is faulty. Check *GROUP_ADDR* of your filters. *MAIN* must be less than 16, *SUB_MAIN* less than 8. |
| 0x0003 | 3 | WRONG_EIB_GROUP_LEN | The input variable *EIB_GROUP_FILTER.GROUP_LEN* is faulty. Incorrect filter length. Check the mode and the length of the filters. |
| 0x0004 | 4 | WRONG_EIB_NO_FILTER | No filter detected. Check your filter in *EIB_GROUP_FILTER* and the mode. |
| 0x0005 | 5 | WRONG_EIB_IDX_RANGE | The input variable *idx* has an incorrect value. |
| 0x000A | 10 | WRONG_EIB_FIRMWARE | The mode is not supported with this firmware. |
| 0x000B | 11 | WRONG_EIB_MODE | Unsupported mode during parameterization. Check *iMode*. Permitted values are 0, 1 and 100. |
| 0x000C | 12 | WRONG_MODE | The input variable *iMode* has an incorrect value. |
| 0x000E | 14 | WRONG_EIB_FIRMWARE_B1_NECESSARY | Firmware B1 or higher required. |
| 0x000F | 15 | WRONG_EIB_FIRMWARE_B3_NECESSARY | Firmware B3 or higher required. |
| 0x0014 | 20 | WRONG_EIB_DATA_LEN | Expected data length of the EIB telegram is wrong. Telegram is discarded. Check the EIB group addresses and/or the data type used. |
| 0x0015 | 21 | ERROR_EIB_SERVICE_NOT_SUPPORT | An EIB telegram is not supported. |

| Value (hex) | Value (dec) | Value (enum) | Description |
|---|---|---|---|
| 0x001E | 30 | KL6301_TP_TOGGLE_ERROR | Terminal did not respond for one second. Check the connection with the KL6301. Is it still busy with data exchange? |
| 0x001F | 31 | TIME_OUT | The terminal fails to respond during parameterization. Check the connection with the KL6301. |
| 0x0020 | 32 | KL6301_NO_RESPONSE_FROM_TERMINAL | No connection to KL6301. Either terminal not available or mapping incorrect. |
| 0x0028 | 40 | ERROR_SEND_8BIT_WRONG_Scaling_Mode | Incorrect or unsupported Scaling mode. |
| 0x0064 | 100 | ERROR_EIB_PHY_ADDR_NOT_SUPPORT | Physical addressing not permitted. |
| 0x0065 | 101 | ERROR_EIB_WRITE_DATA | Outdated. No longer used. |
| 0x0066 | 102 | MONITOR_MODE_LEN_IS_NOT_OK_MUST_0 | For Monitor mode the filter length must be 0. |
| 0x0067 | 103 | MONITOR_MODE_ADDR_IS_NOT_OK_MUST_0 | For Monitor mode the addresses must be 0. |
| 0x0068 | 104 | WATCHDOG_ERROR_NO_SEND | Data transfer not possible. The group address for which the data transfer has failed can be found in the local variable "NotSendGroup" of function block KL6301. |
| 0x0BBB | 3003 | ERROR_EIB_NO_ACK | No ACK received. |
| 0xFAFB | 64251 | ERROR_EIB_NO_COM_TO_TP | No communication with the EIB hardware. |
| 0x0FCC | 4044 | ERROR_TP_TEMP_WARNING | Temperature in KL6301 exceeded. |
| 0x17CC | 6092 | ERROR_TP_PROTOCOL_ERROR | Protocol error in EIB physics. |
| 0x27CC | 10188 | ERROR_TP_TRANSMITTER_ERROR | Protocol error in EIB physics. |
| 0x47CC | 18380 | ERROR_TP_RECEIVE_ERROR | Protocol error in EIB physics. |
| 0x87CC | 34764 | ERROR_TP_SLAVE_COLLISION | Too many collisions in the EIB physics. Reduce the EIB load. |

## 6.4 Functions

| Function blocks | Description |
|---|---|
| F_CONV_2GROUP_TO_3GROUP [▶ 63] | Conversion of a 2-stage group address to a 3-stage group address |
| F_CONV_3GROUP_TO_2GROUP [▶ 64] | Conversion of a 3-stage group address to a 2-stage group address |

### 6.4.1 F_CONV_2GROUP_TO_3GROUP : EIB_GROUP_ADDR

EIB_GROUP_ADDR [▶ 66]


Conversion of a 2-level group address in a 3-level group address.

**VAR_INPUT**

```
IN      : EIB_GROUP_ADDR_2GROUP;
```

**IN:** 2-level group address (see EIB_GROUP_ADDR_2GROUP [▶ 66]).

### 6.4.2 F_CONV_3GROUP_TO_2GROUP : EIB_GROUP_ADDR_2GROUP

EIB_GROUP_ADDR_2GROUP [▶ 66]


Conversion of a 3-level group address in a 2-level group address.

**VAR_INPUT**

```
IN      : EIB_GROUP_ADDR;
```

**IN:** 3-level group address (see EIB_GROUP_ADDR [▶ 66]).

## 6.5 Data types

| Data types | Description |
|---|---|
| EIB_ERROR_CODE [▶ 64] | Error messages |
| EIB_PRIORITY [▶ 66] | EIB telegram priority |

| Data types | Description |
|---|---|
| EIB_GROUP_ADDR [▶ 66] | 3-stage group address |
| EIB_GROUP_ADDR_2GROUP [▶ 66] | 2-stage group address |
| EIB_GROUP_FILTER [▶ 66] | Group filter |
| EIB_PHYS_ADDR [▶ 67] | Physical address |
| EIB_REC [▶ 67] | Links the send and receive blocks with the function block *KL6301* |

### 6.5.1 EIB_ERROR_CODE

Library error messages.

```
TYPE EIB_ERROR_CODE :
(
  NO_EIB_ERROR                       := 0,
  WRONG_EIB_PHYS_ADDR                := 1,
  WRONG_EIB_GROUP_ADDR               := 2,
  WRONG_EIB_GROUP_LEN                := 3,
  WRONG_EIB_NO_FILTER                := 4,
  WRONG_EIB_IDX_RANGE                := 5,
  WRONG_EIB_FIRMWARE                 := 10,
  WRONG_EIB_MODE                     := 11,
  WRONG_MODE                         := 12,
  WRONG_EIB_FIRMWARE_B1_NECESSARY    := 14,
  WRONG_EIB_FIRMWARE_B3_NECESSARY    := 15,
  WRONG_EIB_DATA_LEN                 := 20,
  ERROR_EIB_SERVICE_NOT_SUPPORT      := 21,
  KL6301_TP_TOGGLE_ERROR             := 30,
  TIME_OUT                           := 31,
  KL6301_NO_RESPONSE_FROM_TERMINAL   := 32,
  ERROR_SEND_8BIT_WRONG_Scaling_Mode := 40,
  ERROR_EIB_PHY_ADDR_NOT_SUPPORT     := 100,
  ERROR_EIB_WRITE_DATA               := 101,
  MONITOR_MODE_LEN_IS_NOT_OK_MUST_0  := 102,
  MONITOR_MODE_ADDR_IS_NOT_OK_MUST_0 := 103,
```

```
  WATCHDOG_ERROR_NO_SEND              := 104,
  ERROR_EIB_NO_ACK                    := 16#0BBB,
  ERROR_EIB_NO_COM_TO_TP              := 16#FAFB,
  ERROR_TP_TEMP_WARNING               := 16#0FCC,
  ERROR_TP_PROTOCOL_ERROR             := 16#17CC,
  ERROR_TP_TRANSMITTER_ERROR          := 16#27CC,
  ERROR_TP_RECEIVE_ERROR              := 16#47CC,
  ERROR_TP_SLAVE_COLLISION            := 16#87CC
)
END_TYPE
```

NO_EIB_ERROR: No error.

WRONG_EIB_PHYS_ADDR: Outdated, no longer used.

WRONG_EIB_GROUP_ADDR: The input variable *EIB_GROUP_FILTER.GROUP_ADDR* is faulty. Check *GROUP_ADDR* of your filters. *MAIN* must be less than 16, *SUB_MAIN* less than 8.

WRONG_EIB_GROUP_LEN: The input variable *EIB_GROUP_FILTER.GROUP_LEN* is faulty. Incorrect filter length. Check the mode and the length of the filters.

WRONG_EIB_NO_FILTER: No filters detected. Check your filter in *EIB_GROUP_FILTER* and the mode.

WRONG_EIB_IDX_RANGE: The input variable *idx* has an incorrect value.

WRONG_EIB_FIRMWARE: The mode is not supported with this firmware.

WRONG_EIB_MODE: Unsupported mode during parameterization. Check *iMode*. Permitted values are 0, 1 and 100.

WRONG_MODE: The input variable *iMode* has an incorrect value.

WRONG_EIB_FIRMWARE_B1_NECESSARY: At least firmware B1 or higher required.

WRONG_EIB_FIRMWARE_B3_NECESSARY: At least firmware B3 or higher required.

WRONG_EIB_DATA_LEN: Expected data length of the EIB telegram is wrong. Telegram is discarded. Check the EIB group addresses and/or the data type used.

ERROR_EIB_SERVICE_NOT_SUPPORT: This EIB telegram is not supported.

KL6301_TP_TOGGLE_ERROR: Terminal did not respond for one second. Check the connection with the KL6301. Is it still busy with data exchange?

TIME_OUT: The terminal failed to respond during parameterization. Check the connection with the KL6301.

KL6301_NO_RESPONSE_FROM_TERMINAL: No connection to KL6301. Either terminal not available or mapping incorrect.

ERROR_SEND_8BIT_WRONG_Scaling_Mode: Wrong or not supported Scaling mode.

ERROR_EIB_PHY_ADDR_NOT_SUPPORT: Physical addressing not allowed.

ERROR_EIB_WRITE_DATA: Outdated. No longer used.

MONITOR_MODE_LEN_IS_NOT_OK_MUST_0: For Monitor operation the length of the filters must be 0.

MONITOR_MODE_ADDR_IS_NOT_OK_MUST_0: For Monitor operation the addresses must be 0.

WATCHDOG_ERROR_NO_SEND: Transmission of data not possible. The group address for which the data transfer has failed can be found in the local variable "NotSendGroup" of function block KL6301.

ERROR_EIB_NO_ACK: No ACK received.

ERROR_EIB_NO_COM_TO_TP: No communication with the EIB hardware.

ERROR_TP_TEMP_WARNING: Temperature exceeded in the KL6301.

ERROR_TP_PROTOCOL_ERROR: Protocol error on EIB physics.

ERROR_TP_TRANSMITTER_ERROR: Protocol error on EIB physics.

ERROR_TP_RECEIVE_ERROR: Protocol error on EIB physics.

ERROR_TP_SLAVE_COLLISION: Too many collisions on EIB physics. Reduce the EIB load.

## 6.5.2 EIB_PRIORITY

Priority of the EIB telegram.

```
TYPE EIB_PRIORITY :
(
  EIB_PRIORITY_LOW    := 1,
  EIB_PRIORITY_HIGH   := 2,
  EIB_PRIORITY_ALARM  := 3,
)
END_TYPE
```

EIB_PRIORITY_LOW: Priority low.

EIB_PRIORITY_HIGH: Priority high.

EIB_PRIORITY_ALARM: Priority alarm.

## 6.5.3 EIB_GROUP_ADDR

3-level group address.

```
TYPE EIB_GROUP_ADDR :
STRUCT
  MAIN      : BYTE;
  SUB_MAIN  : BYTE;
  NUMBER    : BYTE;
END_STRUCT
END_TYPE
```

**MAIN:** Main group (range 0..31).

**SUB_MAIN:** Middle group (range 0..7).

**NUMBER:** Sub group (range 0..255).

## 6.5.4 EIB_GROUP_ADDR_2GROUP

2-level group address.

```
TYPE EIB_GROUP_ADDR_2GROUP :
STRUCT
  MAIN      : BYTE;
  SUB_MAIN  : WORD;
END_STRUCT
END_TYPE
```

**MAIN:** Main group (range 0..15).

**SUB_MAIN:** Sub group (range 0..2048).

## 6.5.5 EIB_GROUP_FILTER

Group filter.

```
TYPE EIB_GROUP_FILTER :
STRUCT
  GROUP_ADDR  : EIB_GROUP_ADDR;
  GROUP_LEN   : WORD;
END_STRUCT
END_TYPE
```

**GROUP_ADDR:** Group address (see EIB_GROUP_ADDR [▶ 66]).

**GROUP_LEN:** iMode 0 - 0..63. iMode 1 - 0..31.

## 6.5.6 EIB_PHYS_ADDR

Physical address.

```
TYPE EIB_PHYS_ADDR :
STRUCT
  Area    : BYTE := 1;
  Line    : BYTE := 2;
  Device  : BYTE := 3;
END_STRUCT
END_TYPE
```

**Area:** 0..15.

**Line:** 0..15.

**Device:** 0..255.

## 6.5.7 EIB_REC

Connection between *KL6301* and the read / send function blocks.

```
TYPE EIB_REC :
STRUCT
  Rec_Group       : EIB_GROUP_ADDR;
  Rec_Len         : INT;
  Rec_Idx         : INT := 1;
  Rec_Data        : ARRAY[1..15] OF BYTE;
  Rec_bWriteBusy  : BOOL;
  Rec_bReadBusy   : BOOL;
  Rec_bReady      : BOOL;
  Rec_bError      : BOOL;
  Rec_iErrorID    : EIB_Error_Code;
  pStr_Send       : DWORD;
  Rec_Data_rec    : BOOL;
  Rec_Typ         : EIB_Read_Typ;
END_STRUCT
END_TYPE
```

**Rec_Group:** Group address (see EIB_GROUP_ADDR [▶ 66]).

**Rec_Len:** Length.

**Rec_Idx:** Index.

**Rec_Data:** Data bytes.

**Rec_bWriteBusy:** Data is sent.

**Rec_bReadBusy:** Data is read.

**Rec_bReady:** Ready.

**Rec_bError:** Becomes TRUE as soon as an error occurs. The error is described via the variable *Rec_iErrorID*.

**Rec_iErrorID:** Issues an error code when an error occurs (see EIB_ERROR_CODE [▶ 64]). Simultaneously *Rec_bError* is TRUE.

**pStr_Send:** Pointer to the data to be sent.

**Rec_Data_rec:** Signals data reception.

**Rec_Typ:** Type of telegram.

# 7 Appendix

## 7.1 Examples

**Requirements**

| Example | Description |
|---|---|
| https://infosys.beckhoff.com/content/1033/tcplclibeib/Resources/11993063051/.zip | TwinCAT PLC project for the KL6301. |
| https://infosys.beckhoff.com/content/1033/tcplclibeib/Resources/11993064459/.zip | Example for a Bus Terminal Controller of BCxx00 series. |

## 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.

**Download finder**

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

**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 our internet page: 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 |
| 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 |
| e-mail: | service@beckhoff.com |

**Beckhoff Headquarters**

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

| | |
|---|---|
| Phone: | +49 5246 963-0 |
| e-mail: | info@beckhoff.com |
| web: | 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