

## Documentation

# BX9000

## Bus Terminal Controller for Ethernet

Version: 2.0.0  
Date: 2017-03-02

**BECKHOFF**



# Table of Contents

<b>1 Foreword .....</b>	<b>6</b>
1.1 Notes on the documentation .....	6
1.2 Safety instructions .....	7
1.3 Documentation issue status .....	8
<b>2 Product overview .....</b>	<b>9</b>
2.1 Bus Terminal Controllers of the BX series .....	9
2.2 BX9000 - Introduction .....	10
2.3 Technical data .....	11
2.3.1 Technical data - BX .....	11
2.3.2 Technical Data - PLC .....	13
2.3.3 Technical data - Ethernet .....	13
2.4 The principle of the Bus Terminal .....	14
2.5 The Beckhoff Bus Terminal system .....	14
<b>3 Mounting and wiring .....</b>	<b>17</b>
3.1 Mounting .....	17
3.1.1 Dimensions .....	17
3.1.2 Installation on mounting rails .....	18
3.2 Wiring .....	19
3.2.1 Potential groups, insulation testing and PE .....	19
3.2.2 Power supply .....	22
3.2.3 Programming cable for COM1 .....	23
3.2.4 SSB and COM interface .....	25
3.2.5 Ethernet connection .....	26
<b>4 Parameterization and commissioning .....</b>	<b>28</b>
4.1 Start-up behavior of the Bus Terminal Controller .....	28
4.2 Setting the IP address .....	29
4.2.1 Address Configuration via TwinCAT System Manager .....	29
4.2.2 Setting the address via BootP server .....	30
4.2.3 Setting the address via DHCP server .....	31
4.2.4 Subnet mask .....	32
4.3 Configuration .....	32
4.3.1 Overview .....	32
4.3.2 Creating a TwinCAT configuration .....	34
4.3.3 Downloading a TwinCAT configuration .....	35
4.3.4 Uploading a TwinCAT configuration .....	37
4.3.5 Resources in the Bus Terminal Controller .....	38
4.3.6 ADS connection via serial interface .....	40
4.3.7 ADS connection via Ethernet .....	42
4.3.8 K-bus .....	46
4.3.9 PLC .....	49
4.3.10 SSB .....	52
4.3.11 Real-Time Clock (RTC) .....	83
4.3.12 COM port .....	85
4.4 Menu .....	86
4.4.1 BX menu settings .....	86
4.4.2 Creating own menus .....	90
4.5 Configuration software KS2000 .....	90
<b>5 Programming .....</b>	<b>91</b>

5.1	PLC features of the BX controllers .....	91
5.2	TwinCAT PLC .....	91
5.3	TwinCAT PLC - Error codes .....	92
5.4	Remanent data .....	94
5.5	Persistent data .....	95
5.6	Allocated flags .....	97
5.7	Local process image in delivery state (default config) .....	97
5.8	Mapping the Bus Terminals .....	98
5.9	Local process image in the TwinCAT configuration .....	99
5.10	Creating a boot project .....	101
5.11	Communication between TwinCAT and BX/BCxx50 .....	101
5.12	Up- and downloading of programs .....	103
5.13	Libraries .....	106
5.13.1	Libraries overview .....	106
5.13.2	Overview of libraries for BX9000, BC9020, BC9050, BC9120 .....	109
5.13.3	TcBaseBX .....	110
5.13.4	TcSystemBX .....	121
5.13.5	TcComPortBX .....	124
5.13.6	TcTwinSAFE .....	135
5.13.7	TcBaseBX9000 .....	141
5.13.8	TcSystemBX9000 .....	167
5.14	Program transfer .....	169
5.14.1	Program Transfer via Ethernet .....	169
5.14.2	Program transfer via the serial interface .....	170
5.15	Process image .....	172
5.15.1	Fieldbus Process Image .....	172
<b>6</b>	<b>Ethernet .....</b>	<b>173</b>
6.1	Introduction to the system .....	173
6.1.1	Ethernet .....	173
6.1.2	Topology .....	174
6.1.3	Ethernet cable .....	175
6.2	ModbusTCP .....	176
6.2.1	ModbusTCP Protocol .....	176
6.2.2	Modbus TCP interface .....	178
6.2.3	ModbusTCP slave error answer (BK9000, BX/BC9xx0, IP/ILxxxx-B/C900, EK9000) ...	179
6.2.4	ModbusTCP functions .....	180
6.3	ADS-Communication .....	183
6.3.1	ADS-Communication .....	183
6.3.2	ADS protocol .....	184
6.3.3	ADS services .....	185
<b>7</b>	<b>Error handling and diagnosis .....</b>	<b>187</b>
7.1	Diagnosis .....	187
7.2	Diagnostic LEDs .....	188
7.3	Diagnostics display .....	192
<b>8</b>	<b>Appendix .....</b>	<b>193</b>
8.1	BX9000 - First steps .....	193
8.2	Switching between controllers .....	198
8.3	Firmware-Update BX9000 .....	200
8.4	Sample programs - overview .....	201

8.5 Sample programs for BX9000 - Overview ..... 201

8.6 General operating conditions ..... 202

8.7 Test standards for device testing ..... 203

8.8 Bibliography ..... 203

8.9 List of Abbreviations ..... 204

8.10 Support and Service ..... 205

# 1 Foreword

## 1.1 Notes on the documentation

### Intended audience

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning these components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

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

### Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.

We reserve the right to revise and change the documentation at any time and without prior announcement.

No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

### Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

### Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents: EP1590927, EP1789857, DE102004044764, DE102007017835 with corresponding applications or registrations in various other countries.

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



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

### Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

## 1.2 Safety instructions

### Safety regulations

Please note the following safety instructions and explanations!

Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

### Exclusion of liability






All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

### Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

### Description of symbols

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

 <b>DANGER</b>	<b>Serious risk of injury!</b> Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.
 <b>WARNING</b>	<b>Risk of injury!</b> Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.
 <b>CAUTION</b>	<b>Personal injuries!</b> Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.
 <b>Attention</b>	<b>Damage to the environment or devices</b> Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.
 <b>Note</b>	<b>Tip or pointer</b> This symbol indicates information that contributes to better understanding.

## 1.3 Documentation issue status

Version	Modifications
2.0.0	<ul style="list-style-type: none"> <li>• Migration</li> <li>• Update structure</li> </ul>
1.2.0	<ul style="list-style-type: none"> <li>• Update to firmware version 1.20, from hardware 3.5</li> </ul>
1.1.7	<ul style="list-style-type: none"> <li>• Update to firmware version 1.17</li> </ul>
1.1.5	<ul style="list-style-type: none"> <li>• Update to firmware version 1.15</li> </ul>
1.1.2	<ul style="list-style-type: none"> <li>• Update to firmware version 1.12</li> </ul>
1.0.8	<ul style="list-style-type: none"> <li>• Update to firmware version 1.08</li> </ul>
1.0.0	<ul style="list-style-type: none"> <li>• First version</li> </ul>

### BX9000 firmware version

The current firmware version is displayed for approx. three seconds when the BX controller is switched on. Use the TwinCAT System Manager to update the firmware on your BX9000 [Firmware with TWINCAT System Manager](#) [► 200].

Firmware	Description
1.25	<ul style="list-style-type: none"> <li>• SSB reset problem in the event of short-circuited CAN line resolved</li> </ul>
1.24	<ul style="list-style-type: none"> <li>• SSB fix if the CAN is not connected when the BX starts up</li> </ul>
1.22	<ul style="list-style-type: none"> <li>• Large model implemented</li> </ul>
1.20	<ul style="list-style-type: none"> <li>• Firmware for BX controllers from hardware version 3.5</li> <li>• Switching of the COM 2 interface between RS232 and RS485 modified</li> <li>• In DHCP and BootP mode a private IP address is generated if the DHCP or BootP server does not respond.</li> </ul> <p><b>Caution</b> Firmware version 1.20 does not run on older hardware versions (lower than 3.5). The hardware version of your BX controller can be found on its sticker.</p>
1.17	<ul style="list-style-type: none"> <li>• New: Support for TwinSAFE Bus Terminals: A maximum of one logic terminal at the K-bus with a maximum of 7 connections permitted (<a href="#">further information</a> [► 135])</li> <li>• New: In addition to the 2 kbytes of remanent data, 1000 bytes of <a href="#">persistent data</a> [► 95] is incorporated</li> </ul>
1.15	<ul style="list-style-type: none"> <li>• New: <a href="#">Modbus client</a> [► 155]</li> <li>• New: <a href="#">TCP/IP and UDP/IP function blocks implemented</a> [► 142]</li> <li>• New: HTML Page</li> <li>• The display illumination can be switched on and off</li> <li>• CRC of the program can displayed during booting</li> </ul>
1.12	<ul style="list-style-type: none"> <li>• New: SSB sync support</li> <li>• New: ADS indication</li> <li>• Larger TwinCAT files are possible for the TwinCAT configuration</li> <li>• ADS read client 2 byte offset rectified</li> <li>• New <a href="#">function blocks implemented</a> [► 109]</li> </ul>
1.08	<ul style="list-style-type: none"> <li>• TwinCAT 2.10 from build 1251 is required for this firmware version</li> </ul>
1.00	<ul style="list-style-type: none"> <li>• First version</li> </ul>



## 2 Product overview

### 2.1 Bus Terminal Controllers of the BX series

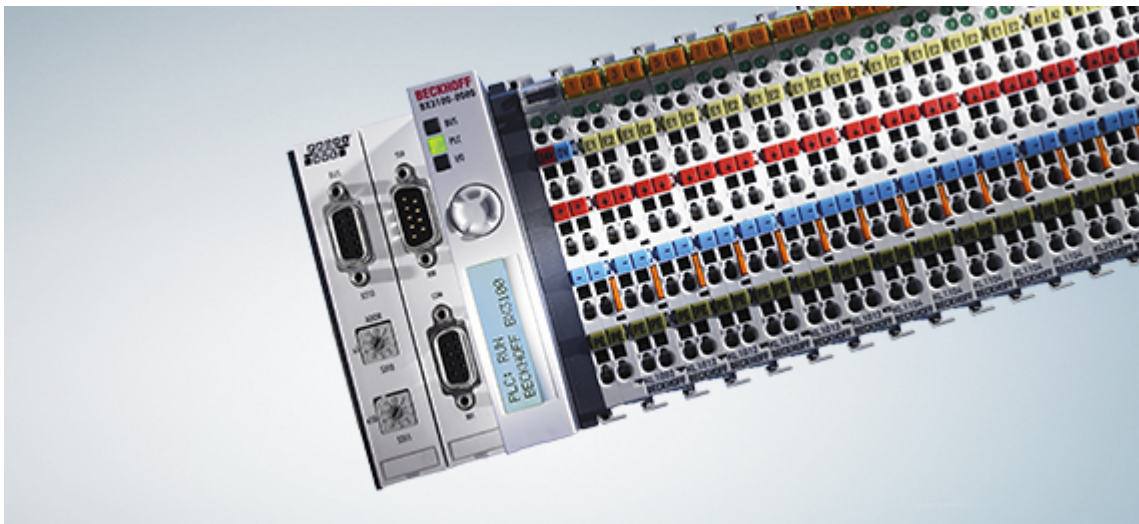


Fig. 1: Bus Terminal Controllers of the BX series

The Bus Terminal Controllers of the BX series (BX controllers) offer a high degree of flexibility. In terms of the equipment and performance range, the BX series is positioned between the BC series Bus Terminal controller and the CX1000 Embedded PC. The concept of a stand-alone controller in combination with a link to a higher-level fieldbus system is based on the BC series. The housing design originates from the CX1000. The main features distinguishing the BC and BX series are the larger memory and the expanded interfaces of the BX series.

The BX controllers consist of a programmable IEC 61131-3 controller, a connection to the higher-level fieldbus system and the K-bus interface for connecting the Beckhoff Bus Terminals. In addition, the BX controllers have two serial interfaces: one for programming, the other for free utilization. The device itself includes an illuminated LC display (2 rows with 16 characters each) with joystick switch and a real-time clock. Further peripheral devices, e.g. displays, can be connected via the integrated Beckhoff Smart System Bus (SSB).

The Bus Terminals are connected on the right side of the BX controller, as usual. The comprehensive range of different I/Os enables any input signal to be read and any output signal that may be required to be generated. The BX controllers can be used for a wide range of automation tasks, from garage door control to autonomous temperature control at injection molding machines. The BX controllers are also eminently suitable for a modular machine concept. Within a network, the BX controllers can exchange data with other machine components via the fieldbus interfaces. The real-time clock also enables decentralized applications, for which the day of the week or the time play an important role.

The areas of application of this series are similar to that of the BC series, but due to the larger memory the BX can execute significantly more complex and larger programs and can manage more data locally (e.g. history and trend data recording), which are then successively fetched over the fieldbus.



**Note**

#### **Bus Terminal and end terminal required**

To operate a BX controller, at least one Bus Terminal with process image and the end terminal must be connected to its K-bus.

#### **Fieldbus interface**

The variants of the BX series Bus Terminal Controllers differ in terms of their fieldbus interfaces. Additionally, two serial interfaces are integrated for programming and for the connection of further serial devices. Five different versions cover the main fieldbus systems:

- BX3100: PROFIBUS DP

- [BX5100](#): CANopen
- [BX5200](#): DeviceNet
- [BX8000](#): RS232 or RS485 (without fieldbus interface)
- [BX9000](#): Ethernet ModbusTCP/ADS-TCP/UDP

## Programming

The BX controllers are programmed based on the effective IEC 61131-3 standard. As with all other Beckhoff controllers, the TwinCAT automation software is the basis for parameterization and programming. Users therefore have the familiar TwinCAT tools available, e.g. PLC programming interface, System Manager and TwinCAT Scope. Data is exchanged optionally via the serial port (COM1) or via the fieldbus through Beckhoff PC FCxxx fieldbus cards.

## Configuration

The configuration is also carried out using TwinCAT. The fieldbus interface, the SSB bus and the real-time clock can be configured and parameterized via the System Manager. The System Manager can read all connected devices and Bus Terminals. After the parameterization, the configuration is saved on the BX via the serial interface. The configuration thus created can be accessed again later.

## 2.2 BX9000 - Introduction

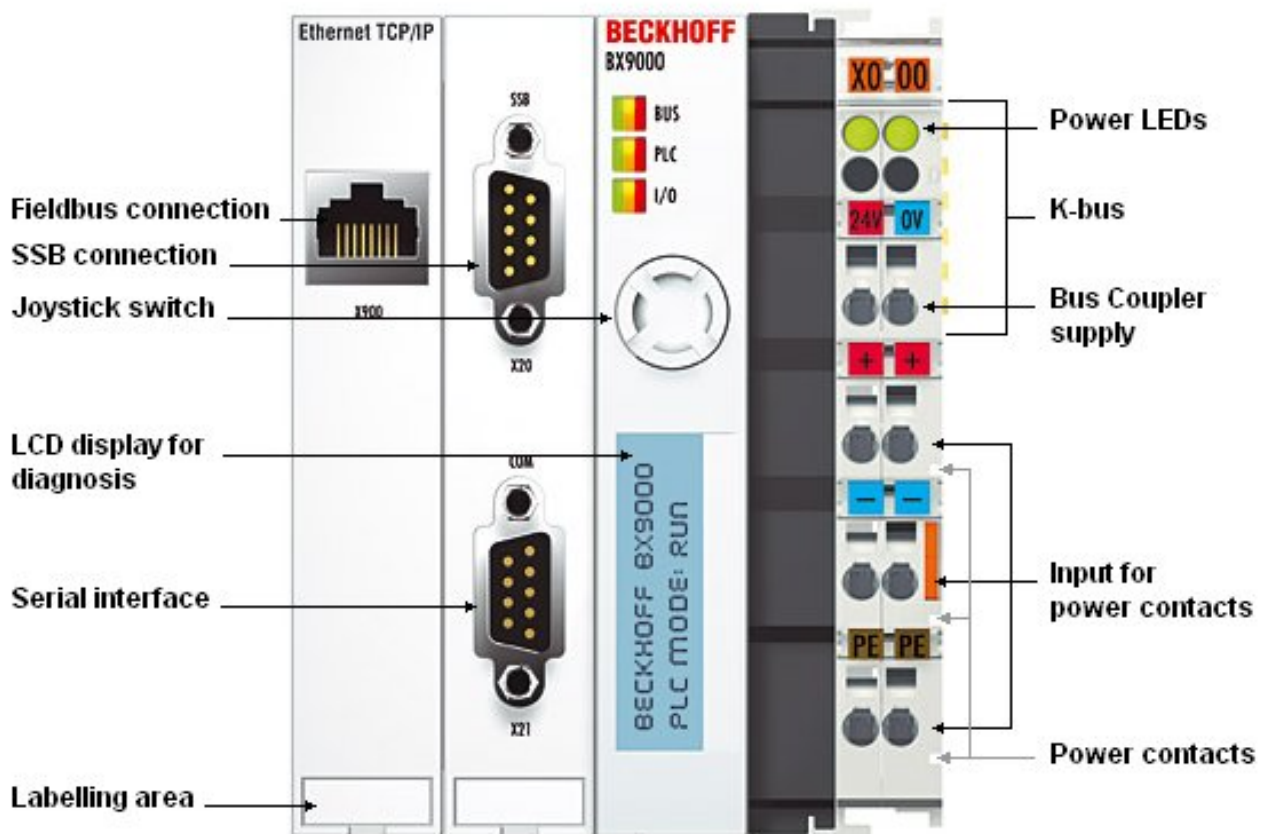


Fig. 2: BX9000 - Ethernet Modbus TCP/ADS-TCP/UDP

The BX9000 Bus Terminal Controller has an Ethernet slave/master interface and features automatic baud rate detection up to 100 Mbaud. The address can be set via DHCP, BootP, ARP entry or the joystick switch. Inputs and outputs up to 2 kbytes can be exchanged with the controller. The ModbusTCP, ADS/TCP and the ADS/UDP protocols are implemented.



## 2.3 Technical data

### 2.3.1 Technical data - BX

Technical data	BX3100	BX5100	BX5200	BX8000	BX9000
Processor	16 bit micro-controller				
Diagnostic LEDs	2 x power supply, 2 x K-Bus				
Display	FSTN 2 x 16 lines display for diagnosis or own texts, illuminated				
Switch	Joystick switch for parameterization and diagnosis				
Clock	battery-powered internal clock for time and date				
Configuration and programming software	TwinCAT PLC				

Fieldbus interface	BX3100	BX5100	BX5200	BX8000	BX9000
Fieldbus	PROFIBUS DP	CANopen	DeviceNet	-	Ethernet

Interfaces	BX3100	BX5100	BX5200	BX8000	BX9000
Serial interface	COM1 (RS232 for configuration and programming, automatic baud rate detection 9600/19200/38400/57600 baud) COM2 (RS232 or RS485) for connecting serial devices				
SSB	CANopen-based sub-bus interface				
Terminal Bus (K-Bus)	64 (255 with K-bus extension)				

Technical data	BX3100	BX5100	BX5200	BX8000	BX9000
Digital peripheral signals	2040 inputs/outputs				
Analog peripheral signals	1024 inputs/outputs				
Configuration possibility	via TwinCAT or the controller				
max. number of bytes, fieldbus	depending on fieldbus				
max. number of bytes, PLC	2048 bytes of input data, 2048 bytes of output data				
Fieldbus connection	D-sub, 9-pin	Open Style Connector, 5 pin	-	RJ45	
Power supply (Us)	24 V <sub>DC</sub> (-15% /+20%) Use a 4 A fuse or an <i>NEC Class 2</i> power supply to meet the UL requirements!				
<div><p>Use 4 Amp. fuse or Class 2 power supply. See instructions.</p></div>					
Input current (Us)	180 mA + (total K-bus current)/4				
Starting current (Us)	approx. 2.5 x continuous current				
K-bus current (5 V)	maximum 1450 mA				
Power contact voltage (Up)	24 V <sub>DC</sub> max. Use a 4 A fuse or an <i>NEC Class 2</i> -compliant power supply to meet the UL requirements.				
<div><p>Use 4 Amp. fuse or Class 2 power supply. See instructions.</p></div>					
Power contact current load (Up)	maximum 10 A				
Dielectric strength	500 V (power contact/supply voltage/Ethernet/fieldbus)				
Permissible ambient temperature range during operation	-25°C ... +60 °C extended temperature range from hardware 4.4	-25°C ... +60 °C extended temperature range from hardware 4.4	-25°C ... +60 °C extended temperature range from hardware 4.4	-25°C ... +60 °C extended temperature range from hardware 4.4	0°C ... +55 °C
Permissible ambient temperature range during storage	-40°C ... +85 °C extended temperature range from hardware 4.4	-40°C ... +85 °C extended temperature range from hardware 4.4	-40°C ... +85 °C extended temperature range from hardware 4.4	-40°C ... +85 °C extended temperature range from hardware 4.4	-20°C ... +85 °C
Relative humidity	95 % no condensation				
Vibration / shock resistance	conforms to EN 60068-2-6 / EN 60068-2-27				
EMC immunity / emission	conforms to EN 61000-6-2 / EN 61000-6-4				
Protection class	IP20				

Mechanical data	BX3100	BX5100	BX5200	BX8000	BX9000
Weight	approx. 170 g				
Dimensions (W x H x D)	approx. 83 mm x 100 mm x 90 mm (BX8000: approx. 65 mm x 100 mm x 90 mm)				
Mounting	with latch, on mounting rail (35 mm DIN rail)				
Installation position	Any				
Connection cross-section	0.08 mm <sup>2</sup> .. 2.5 mm <sup>2</sup> AWG 28-14 8-9 mm strip length				

### 2.3.2 Technical Data - PLC

PLC data	BXxxxx
Programmability	via programming interface (COM1 or COM2) or fieldbus
Program memory	256 kbyte
Source code memory	256 kbyte
Data memory	256 kbyte
Remanent flags	2 kbyte
PLC cycle time	Approx. 0.85 ms for 1000 IL commands (without I/O cycle)
Programming languages	IEC 6-1131-3 (IL, LD, FBD, ST, SFC)
Propagation delay	1 PLC task (second task in preparation)
Online change	Yes
Up/Down Load Code	Yes/Yes

### 2.3.3 Technical data - Ethernet

System data	Ethernet (BX9000)
Number of I/O modules	depending on controller
Number of I/O points	depending on controller
Transmission medium	4 x 2 twisted pair copper cable category 5 (100 Mbaud)
Cable length	100 m between switch and BX9000
Data transfer rate	10/100 Mbaud
Topology	star wiring

## 2.4 The principle of the Bus Terminal

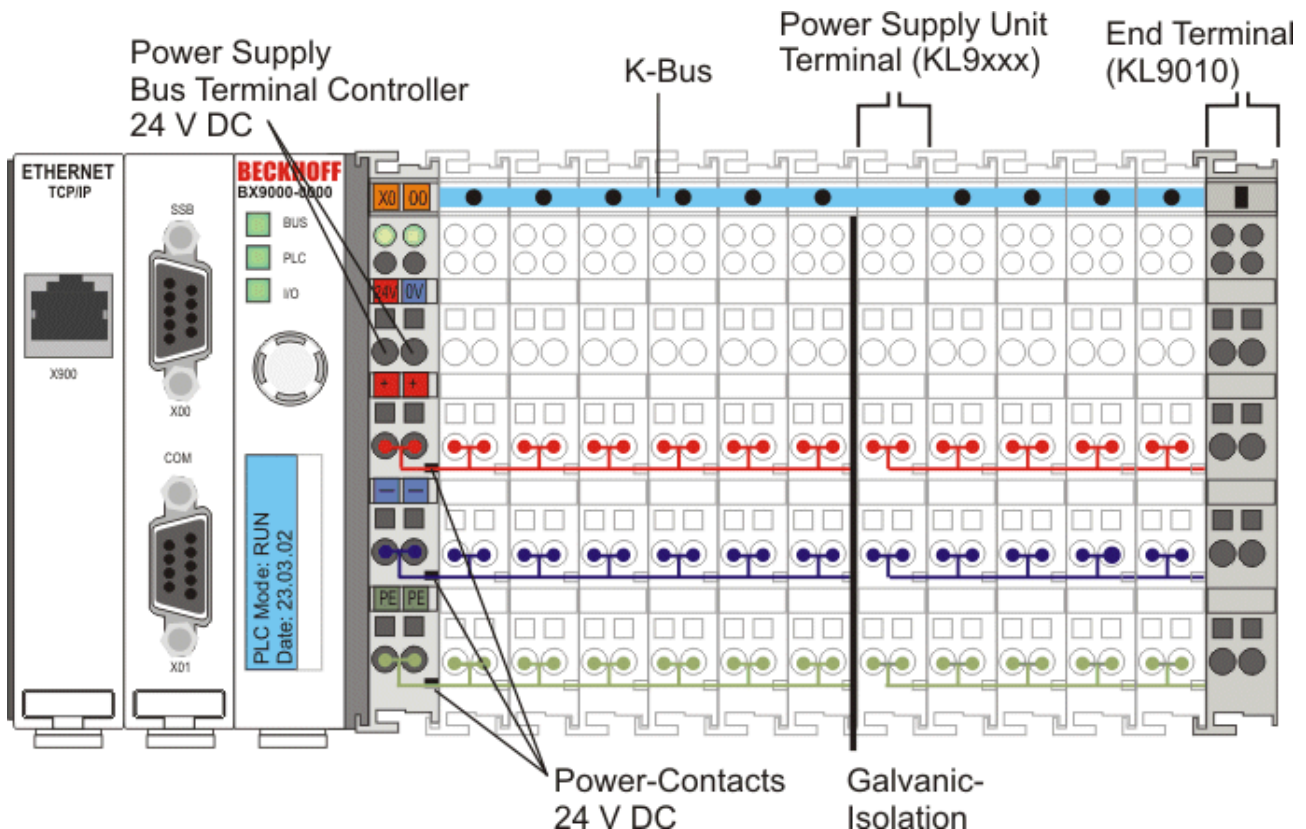


Fig. 3: The principle of the Bus Terminal

## 2.5 The Beckhoff Bus Terminal system

### Up to 256 Bus Terminals, with 1 to 16 I/O channels per signal form

The Bus Terminal system is the universal interface between a fieldbus system and the sensor / actuator level. A unit consists of a Bus Coupler as the head station, and up to 64 electronic series terminals, the last one being an end terminal. Up to 255 Bus Terminals can be connected via the K-bus extension. For each technical signal form, terminals are available with one, two, four or eight I/O channels, which can be mixed as required. All the terminal types have the same mechanical construction, so that difficulties of planning and design are minimized. The height and depth match the dimensions of compact terminal boxes.

### Decentralised wiring of each I/O level

Fieldbus technology allows more compact forms of controller to be used. The I/O level does not have to be brought to the controller. The sensors and actuators can be wired decentrally, using minimum cable lengths. The controller can be installed at any location within the plant.

### Industrial PCs as controllers

The use of an Industrial PC as the controller means that the operating and observing element can be implemented in the controller's hardware. The controller can therefore be located at an operating panel, in a control room, or at some similar place. The Bus Terminals form the decentralised input/output level of the controller in the control cabinet and the subsidiary terminal boxes. The power sector of the plant is also controlled over the bus system in addition to the sensor/actuator level. The Bus Terminal replaces the conventional series terminal as the wiring level in the control cabinet. The control cabinet can have smaller dimensions.



## **Bus Couplers for all usual bus systems**

The Beckhoff Bus Terminal system unites the advantages of a bus system with the possibilities of the compact series terminal. Bus Terminals can be driven within all the usual bus systems, thus reducing the controller parts count. The Bus Terminals then behave like conventional connections for that bus system. All the performance features of the particular bus system are supported.

## **Mounting on standardized mounting rails**

The installation is standardized thanks to the simple and space-saving mounting on a standardized mounting rail (EN 60715, 35 mm) and the direct wiring of actuators and sensors, without cross connections between the terminals. The consistent labelling scheme also contributes.

The small physical size and the great flexibility of the Bus Terminal system allow it to be used wherever a series terminal is also used. Every type of connection, such as analog, digital, serial or the direct connection of sensors can be implemented.

## **Modularity**

The modular assembly of the terminal strip with Bus Terminals of various functions limits the number of unused channels to a maximum of one per function. The presence of two channels in one terminal is the optimum compromise of unused channels and the cost of each channel. The possibility of electrical isolation through potential feed terminals also helps to keep the number of unused channels low.

## **Display of the channel state**

The integrated LEDs show the state of the channel at a location close to the sensors and actuators.

## **K-bus**

The K-bus is the data path within a terminal strip. The K-bus is led through from the Bus Coupler through all the terminals via six contacts on the terminals' side walls. The end terminal terminates the K-bus. The user does not have to learn anything about the function of the K-bus or about the internal workings of the terminals and the Bus Coupler. Many software tools that can be supplied make project planning, configuration and operation easy.

## **Potential feed terminals for isolated groups**

The operating voltage is passed on to following terminals via three power contacts. You can divide the terminal strip into arbitrary isolated groups by means of potential feed terminals. The potential feed terminals play no part in the control of the terminals, and can be inserted at any locations within the terminal strip.

Up to 64 Bus Terminals can be used in a terminal block, with optional K-bus extension for up to 256 Bus Terminals. This count does include potential feed terminals, but not the end terminal.

## **Bus Couplers for various fieldbus systems**

Various Bus Couplers can be used to couple the electronic terminal strip quickly and easily to different fieldbus systems. It is also possible to convert to another fieldbus system at a later time. The Bus Coupler performs all the monitoring and control tasks that are necessary for operation of the connected Bus Terminals. The operation and configuration of the Bus Terminals is carried out exclusively by the Bus Coupler. Nevertheless, the parameters that have been set are stored in each Bus Terminal, and are retained in the event of voltage drop-out. Fieldbus, K-bus and I/O level are electrically isolated.

If the exchange of data over the fieldbus is prone to errors or fails for a period of time, register contents (such as counter states) are retained, digital outputs are cleared, and analog outputs take a value that can be configured for each output when commissioning. The default setting for analog outputs is 0 V or 0 mA. Digital outputs return in the inactive state. The timeout periods for the Bus Couplers correspond to the usual settings for the fieldbus system. When converting to a different bus system it is necessary to bear in mind the need to change the timeout periods if the bus cycle time is longer.

**The interfaces**

A Bus Coupler has six different methods of connection. These interfaces are designed as plug connectors and as spring-loaded terminals.



## 3 Mounting and wiring

### 3.1 Mounting

#### 3.1.1 Dimensions

The Beckhoff Bus Terminal system is characterized by low physical volume and high modularity. When planning a project it must be assumed that at least one Bus Coupler and a number of Bus Terminals will be used. The dimensions of the Bus Terminal Controllers are independent of the fieldbus system.

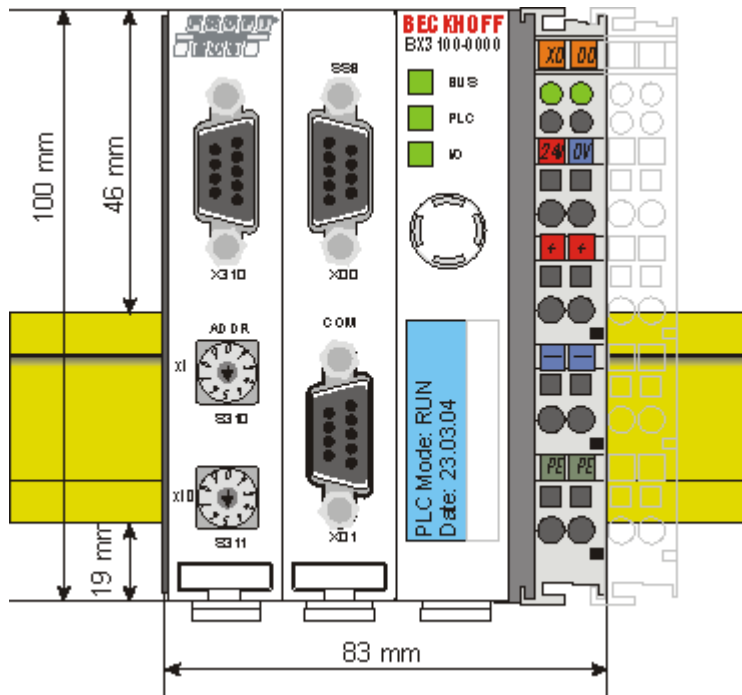


Fig. 4: BX3100, BX5100, BX5200, BX9000

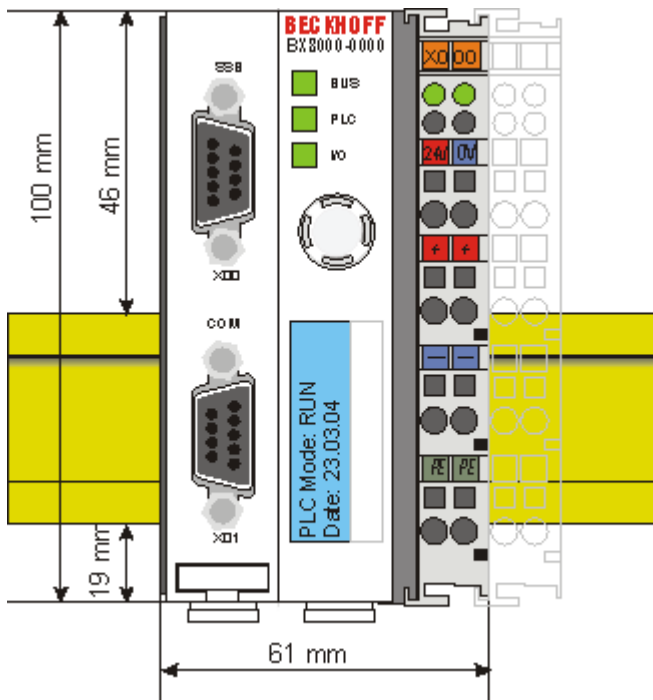


Fig. 5: BX8000

The overall width of the fieldbus station is the width of the Bus Terminal Controller plus the widths of the individual Bus Terminals (including the KL9010 bus end terminal). Depending on design, the Bus Terminals are 12 mm or 24 mm wide. The height is 100 mm.

The BX series Bus Terminal Controllers are up to 83 mm wide and 91 mm deep.

**Note****Pay attention to the total depth**

Note that a Bus Terminal Controller with DIN rail and connected plug connectors is usually higher than the specified value of 91 mm. Example:  
 BX3100 + ZB3100 + DIN rail = 105 mm

### 3.1.2 Installation on mounting rails

#### Mounting

1. The white pull-tabs on the underside of the BX controller are connected to a latching mechanism. Pull the tabs downwards before pushing the BX controller onto the mounting rail.

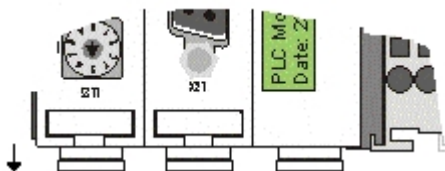


Fig. 6: Released BX controller

**Attention****Avoid damaging the display during the installation!**

Avoid pressing on the display when you push the BX controller onto the mounting rail, in order to avoid damaging the display.

2. Now press the BX controller onto the mounting rail.
3. Once it has snapped onto the mounting rail, push the tabs back into their initial position.

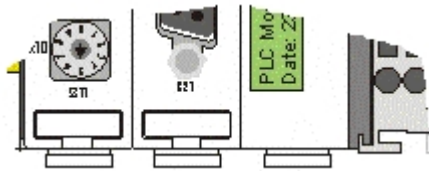


Fig. 7: Latched BX controller

### Disassembly

1. First release all pull tabs on the underside of the BX controller.
2. Then pull the orange tab next to the power supply for the power contacts.

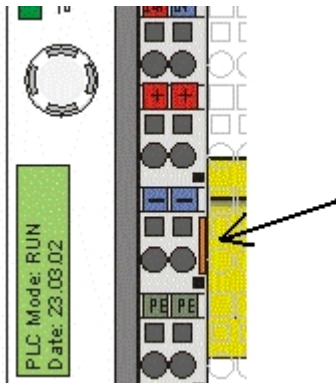


Fig. 8: Disassembly

## 3.2 Wiring

### 3.2.1 Potential groups, insulation testing and PE

#### Potential groups

A Beckhoff Bus Terminal block usually has three different potential groups:

- The fieldbus interface is electrically isolated (except for individual Low Cost couplers) and forms the first potential group.
- Bus Coupler / Bus Terminal Controller logic, K-bus and terminal logic form a second electrically isolated potential group.
- The inputs and outputs are supplied via the power contacts and form further potential groups.

Groups of I/O terminals can be consolidated to further potential groups via potential supply terminals or separation terminals.

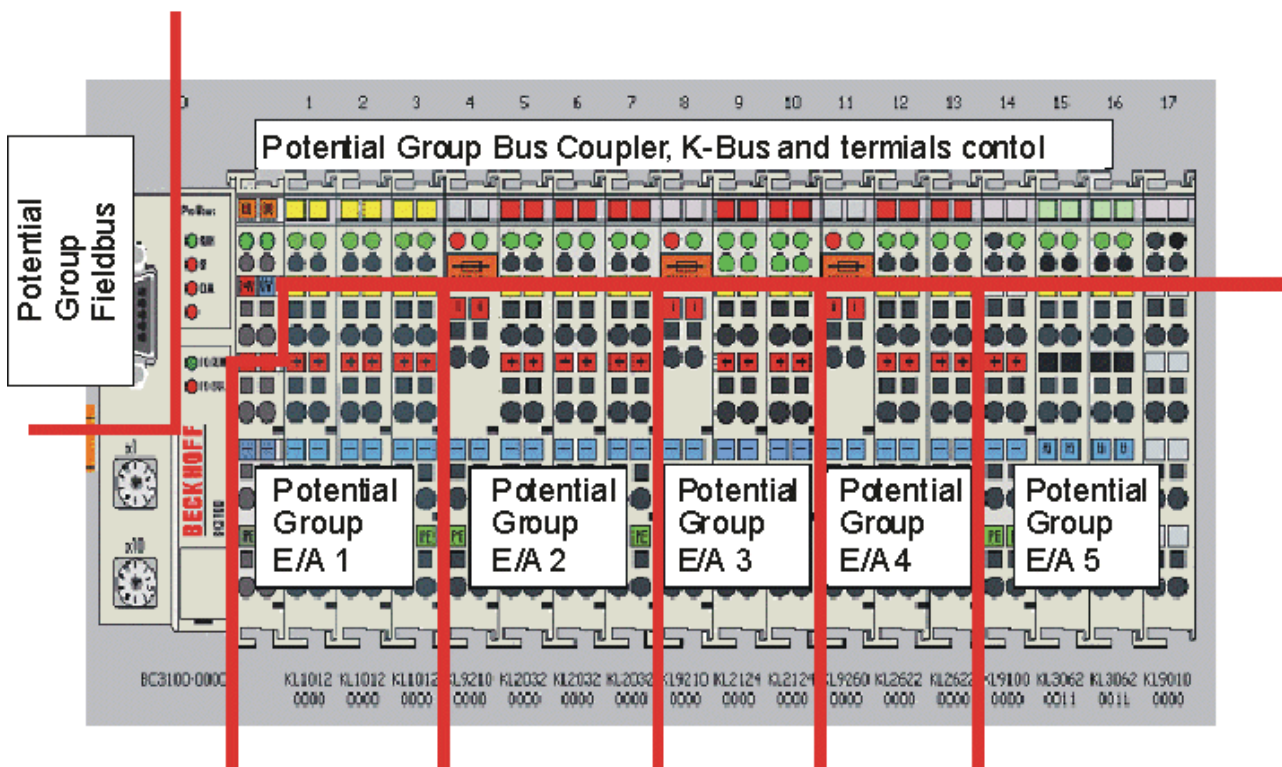


Fig. 9: Potential groups of a Bus Terminal block

### Insulation testing

The connection between Bus Coupler / Bus Terminal Controller and Bus Terminals is realized automatically by latching the components. The transfer of the data and the supply voltage for the intelligent electronics in the Bus Terminals is performed by the K-bus. The supply of the field electronics is performed through the power contacts. Plugging together the power contacts creates a supply rail. Since some Bus Terminals (e.g. analog Bus Terminals or 4-channel digital Bus Terminals) are not looped through these power contacts or not completely the Bus Terminal contact assignments must be considered.

The potential feed terminals interrupt the power contacts, and represent the start of a new supply rail. The Bus Coupler / Bus Terminal Controller can also be used for supplying the power contacts.

### PE power contacts

The power contact labelled PE can be used as a protective earth. For safety reasons this contact mates first when plugging together, and can ground short-circuit currents of up to 125 A.

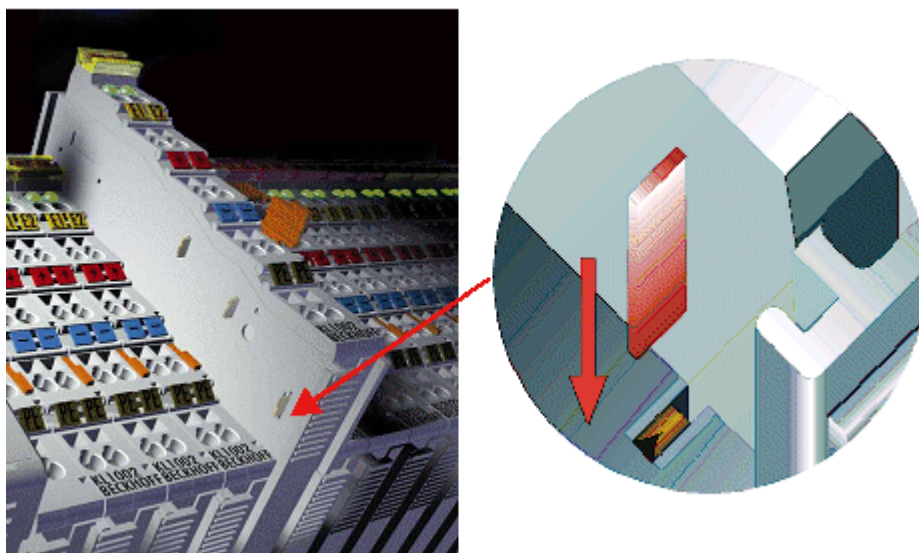





Fig. 10: Power contact on the left

It should be noted that, for reasons of electromagnetic compatibility, the PE contacts are capacitively coupled to the mounting rail. This can both lead to misleading results and to damaging the terminal during insulation testing (e.g. breakdown of the insulation from a 230 V power consuming device to the PE conductor). The PE supply line at the Bus Coupler / Bus Terminal Controller must be disconnected for an insulation test. In order to uncouple further feed locations for the purposes of testing, the feed terminals can be pulled at least 10 mm out from the connected group of other terminals. In that case, the PE conductors do not have to be disconnected.

The power contact with the label PE must not be used for other potentials.

### 3.2.2 Power supply

 <b>DANGER</b>	<b>Risk of injury through electric shock and damage to the device!</b> De-energize the Bus Terminal I/O system before you start installation, disassembly or wiring of the components!
 <b>DANGER</b>	<b>Note the UL requirements for the power supply.</b> These UL requirements apply to all supply voltages of the BX controller (Us and Up)! To comply with UL requirements, the BX controllers may only be connected to supply voltages (24 V <sub>DC</sub> ) that originate <ul style="list-style-type: none"> <li>• from an isolated source protected by a fuse of max. 4A (according to UL248) or</li> <li>• from a voltage supply complying with NEC class 2.</li> </ul> An NEC class 2 voltage source must not be connected in series or parallel with another NEC class 2 voltage source!
 <b>DANGER</b>	<b>No unlimited voltage sources!</b> In order to comply with UL requirements, the BX controllers must not be connected to unlimited voltage sources!

#### Supply of Bus Terminal Controller and Bus Terminals (Us)

The Bus Terminal Controller requires a supply voltage of 24 V<sub>DC</sub>.

The BX controller is connected via the upper terminal points labelled 24 V and 0 V. This voltage supplies the Bus Terminal Controller electronics, and the Bus Terminal electronics via the K-bus. It is galvanically separated from the field level voltage.

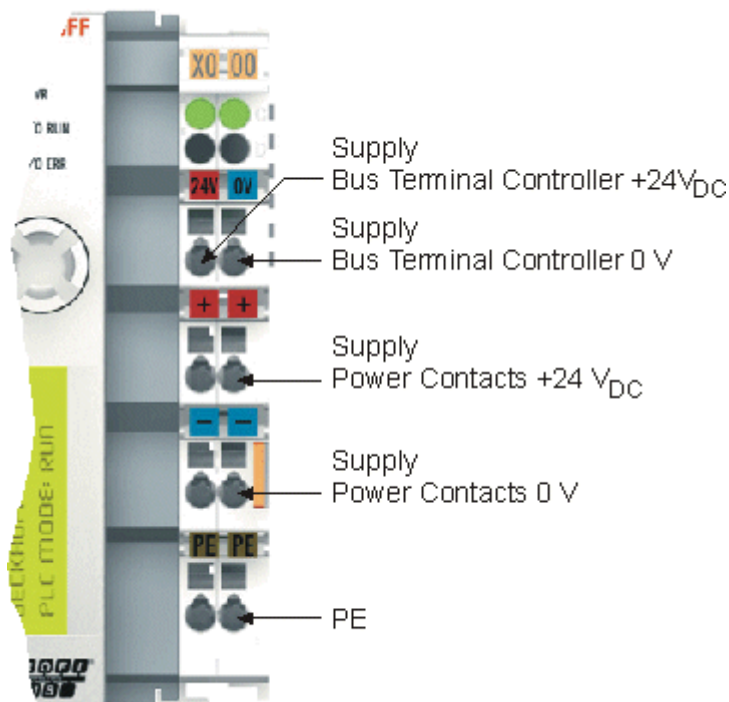


Fig. 11: Terminal points for the Bus Terminal Controller supply

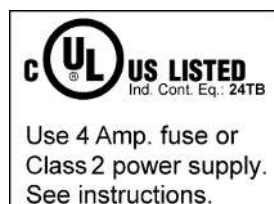


Fig. 12: UL identification

### Power contacts supply (Up)

The bottom six connections with spring-loaded terminals can be used to feed the supply for the peripherals. The spring-loaded terminals are joined in pairs to a power contact. The feed for the power contacts has no connection to the voltage supply for the BX electronics. The design of the feed permits voltages of up to 24 V.

The spring-loaded terminals are designed for wires with cross-sections from 0.08 mm<sup>2</sup> to 2.5 mm<sup>2</sup>.

The assignment in pairs and the electrical connection between feed terminal contacts allows the connection wires to be looped through to various terminal points. The current load from the power contact must not exceed 10 A for long periods. The current carrying capacity between two spring-loaded terminals is identical to that of the connecting wires.

### Power contacts

On the right hand face of the Bus Terminal Controller there are three spring contacts for the power contact connections. The spring contacts are hidden in slots so that they cannot be accidentally touched. By attaching a Bus Terminal the blade contacts on the left hand side of the Bus Terminal are connected to the spring contacts. The tongue and groove guides on the top and bottom of the Bus Terminal Controllers and of the Bus Terminals guarantees that the power contacts mate securely.

## 3.2.3 Programming cable for COM1

You can use a 1:1 cable for programming the BX Controllers (socket/plug, and only connect the pins listed below). On the BX side you need a nine-pin connector, and on the PC side usually a nine-pin socket. The wiring is 1:1, and the necessary pins can be found in the table below. The length of the cable should not exceed 5 meters!

Description	BX COM Port 1	PC COM port RS 232 serial interface
Cable	Plug connector, pin	Socket, pin
RS 232 RxD/TxD	2	2
RS 232 RxD/TxD	3	3
GND	5	5



#### Attention

#### All pins that are not listed in the table are reserved

Please note that pins that are not listed are not freely available at this [COM port \[► 25\]](#), but are reserved for other signals.

### ZK1000-0030

The programming cable can be used to program the BX controller via the COM 1 interface and connect another serial device at the COM 2 interface. Once installed, make sure the maximum overall height of the plug connector is not exceeded.



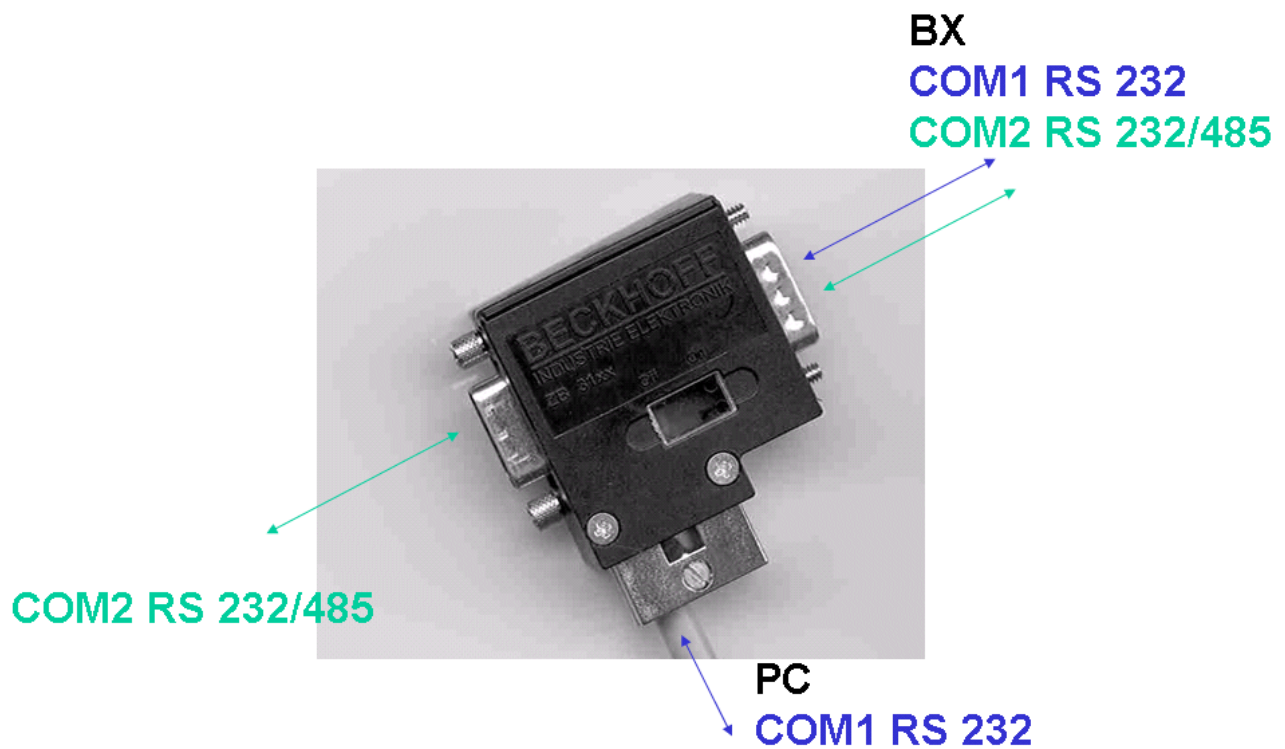


Fig. 13: Programming cable ZK1000-0030 - COM 1 and COM 2

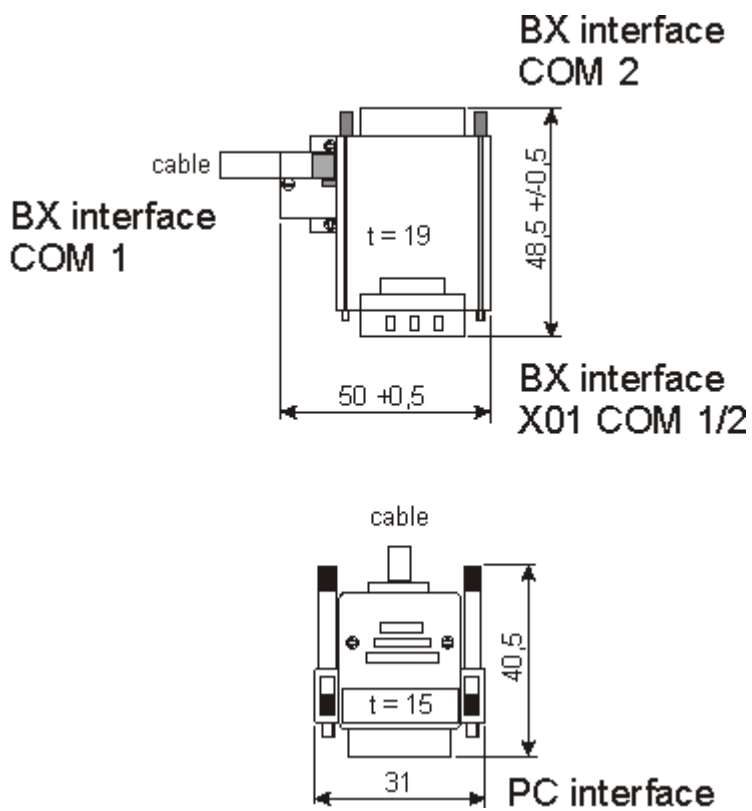


Fig. 14: Programming cable ZK1000-0030 - plug connector dimensions



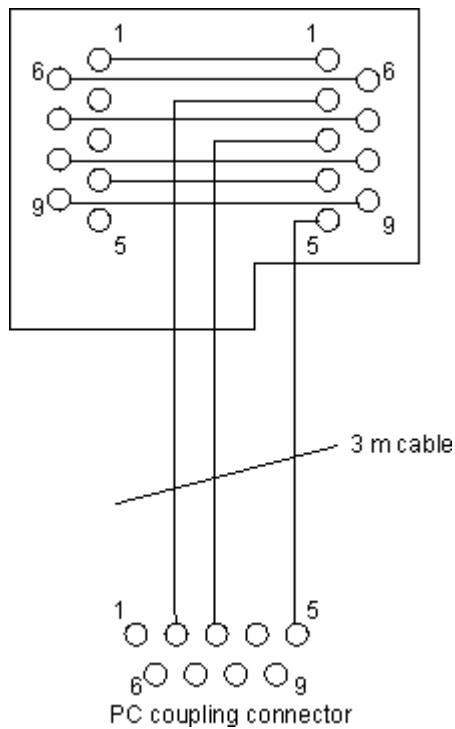


Fig. 15: Programming cable ZK1000-0030 - Pinning

### 3.2.4 SSB and COM interface

The basic BX controller module the COM1, COM2 and SSB (Smart System Bus) interfaces. A D-sub socket is used for COM1 and COM2. A special programming cable (ZK1000-0030) for connecting the two interfaces is available from Beckhoff. The COM2 interface is intended for the connection of serial devices. For the COM2 interface, you can choose between RS232 or RS485.

[Libraries \[► 114\]](#) are available for the serial COM2 interface.

#### SSB interface

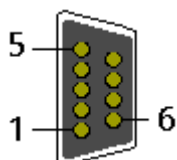


Fig. 16: SSB interface

#### SSB interface assignment (plug connector X00)

PIN	Signal
1	reserved
2	CAN low
3	GND
4	reserved
5	Shield
6	GND
7	CAN high
8	reserved
9	reserved

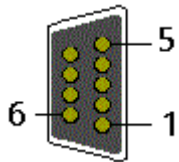
**COM1 (RS 232) and COM2 (RS 232/485) interface**

Fig. 17: COM1 (RS 232) and COM2 (RS 232/485) interface

**COM interface assignment (socket X01)**

PIN	Interface	Signal
1	COM2	RS485 D+
2	COM1	RS232 TxD
3	COM1	RS232 RxD
4	VCC +5 V	VCC
5	GND	GND
6	COM2	RS485 D-
7	COM2	RS232 RxD
8	COM2	RS232 TxD
9	GND	GND

**3.2.5 Ethernet connection**

The connection to the Ethernet bus is made via an RJ45 connector (a Western plug).



Fig. 18: RJ45 connector

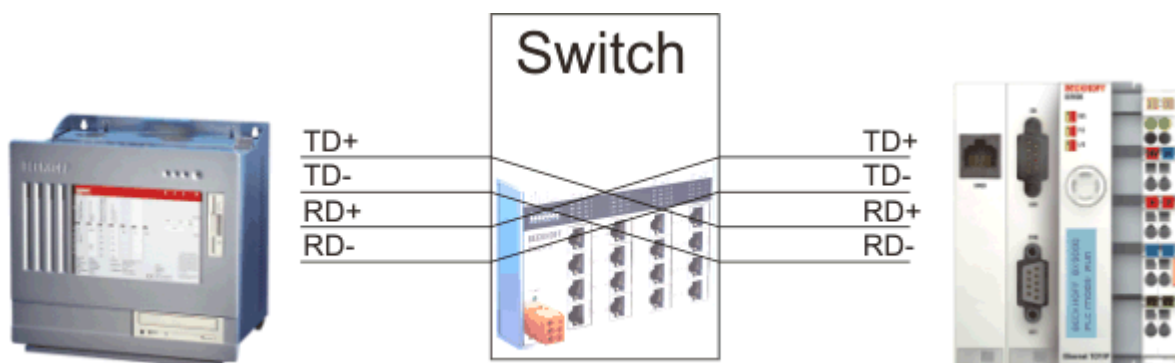
**Cabling****Ethernet connection via hub or switch**

Fig. 19: Ethernet connection between PC and BX9000 via hub or switch

Connect the PC's network card to the hub/switch using a standard Ethernet cable, and connect the hub/switch, again using a standard Ethernet cable, to the Bus Terminal controller.

### Ethernet connection via cross-over cable

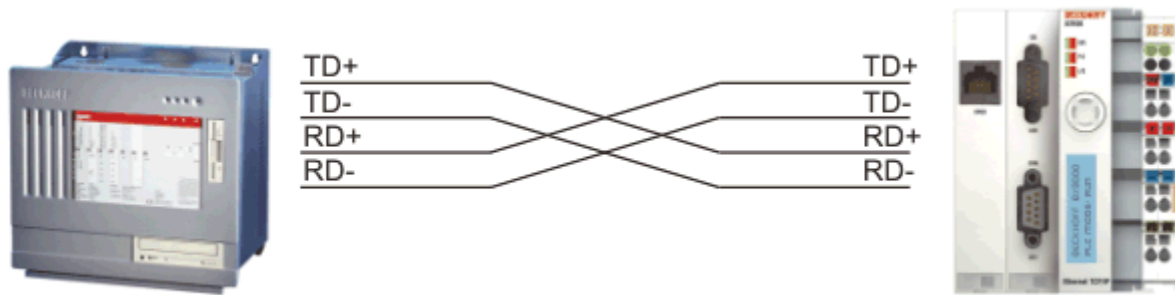


Fig. 20: Ethernet connection between PC and BX9000 via cross-over cable

Use a crossover Ethernet cable to connect the PC directly with the Bus Terminal Controller.

### Pin assignment of the RJ45 plug

PIN	Signal	Description
1	TD +	Transmit +
2	TD -	Transmit -
3	RD +	Receive +
4	-	reserved
5	-	reserved
6	RD -	Receive -
7	-	reserved
8	-	reserved

## 4 Parameterization and commissioning

### 4.1 Start-up behavior of the Bus Terminal Controller

When the Bus Terminal Controller is switched on it checks its state, configures the K-bus, creates a configuration list based on the connected Bus Terminals and starts its local PLC.

The I/O LEDs flash when the Bus Terminal Controller starts up. If the system is in an error-free state, the I/O LEDs should stop flashing after approx. 2-3 seconds. In the event of a fault the error type determines which LED flashes (see chapter *Diagnostic LEDs*).

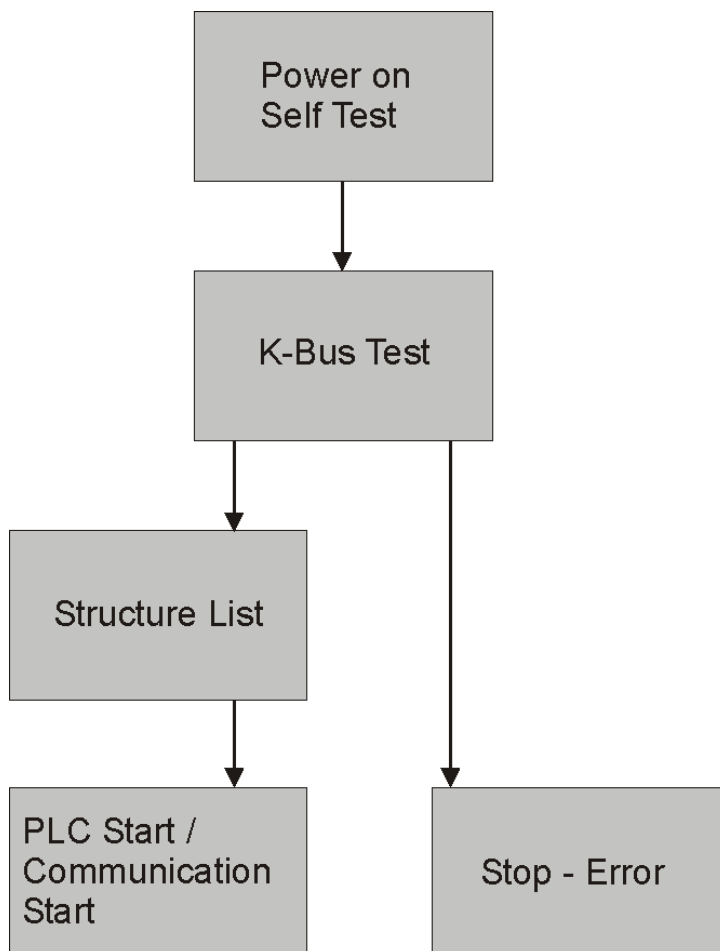


Fig. 21: Start-up behavior of the Bus Terminal Controller

## 4.2 Setting the IP address

The IP address can be set using four different procedures, and these will be described in more detail below.

BX9000: The addressing method is specified via a navigation switch and the display or via the TwinCAT configuration.

Procedure	Explanation	Necessary components
Manual	Addressing via the navigation switch [ <a href="#">► 193</a> ] (BX9000 only)	none
TwinCAT	Addressing via TwinCAT system manager [ <a href="#">► 29</a> ]	PC with network and TwinCAT
BootP*	Addressing via BootP server [ <a href="#">► 30</a> ]	BootP server
DHCP*	Addressing via DHCP server [ <a href="#">► 31</a> ]	DHCP server
Local IP address*	Local IP address	If no BootP or DHCP server responds or is available

\*) BX9000 from firmware version 1.20

### 4.2.1 Address Configuration via TwinCAT System Manager

#### Requirements

- BX firmware 1.08 or above (see the indication on the BX9000 display after the operating voltage has been switched on).
- all BC9050, BC9020 and BC9120
- from TwinCAT 2.10, build 1322

An operable ADS connection is necessary in order to set the IP address through the System Manager. This can be done via a serial connection or via Ethernet (see chapter [Finding the Bus Terminal Controller with the TwinCAT System Manager \[\[► 42\]\(#\)\]](#)).

For the BC9x20, DIP switches 9 and 10 must be ON, for the BC9050 switches 1 and 2 of the two-pin DIP switch must be ON, to enable IP addressing via the System Manager.

The settings made via the System Manager are then applied.

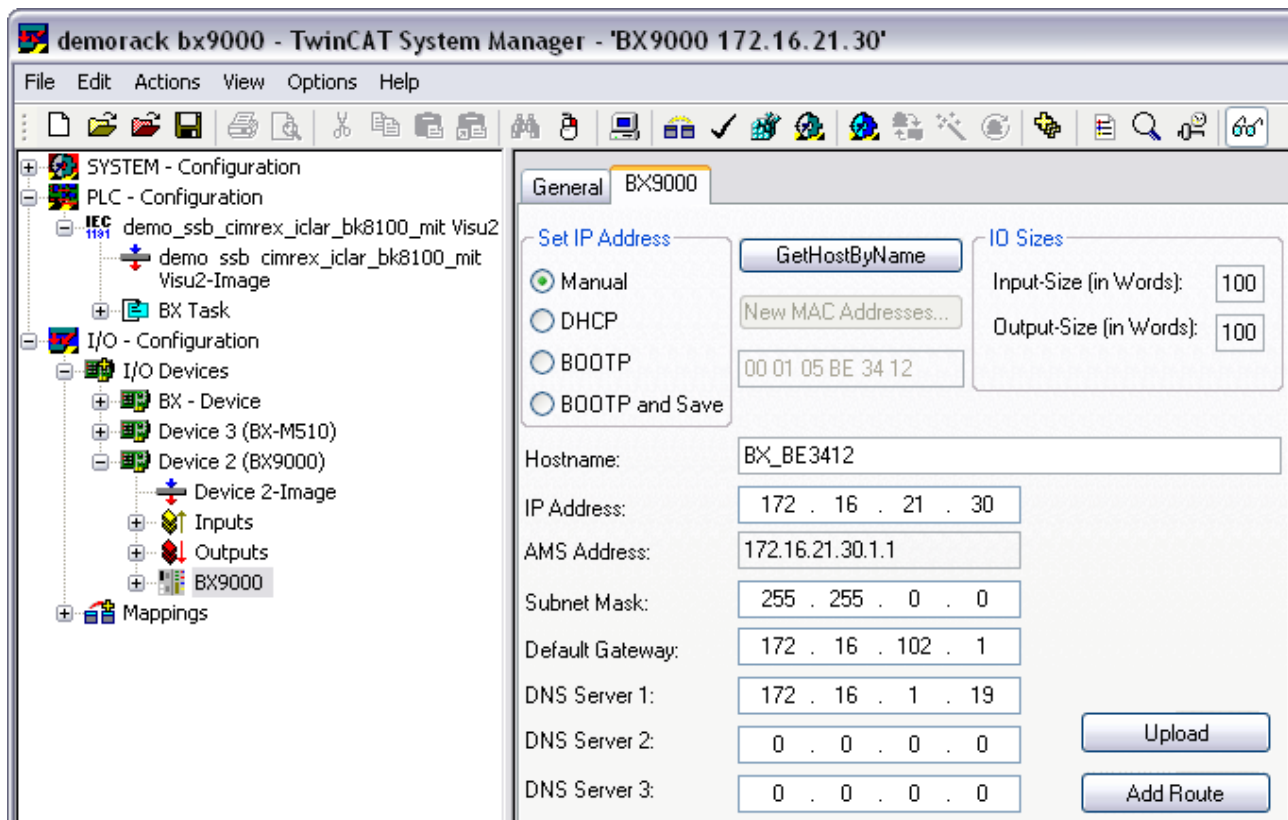



Fig. 22: Address Configuration via TwinCAT System Manager

You can now read the current settings with Upload. If the Bus Terminal Controller was configured offline, the *Add route* button can be used to establish a connection to the Bus Terminal Controller. Edit the required

settings, activate the configuration  and restart your Bus Terminal Controller with the green TwinCAT icon (shortcut [Ctrl] [F4]). If the Bus Terminal Controller has been assigned a new IP address, you have to re-enter the new route (see chapter [Finding the Bus Terminal Controller with the TwinCAT System Manager](#) [► 42]).

## 4.2.2 Setting the address via BootP server

**BX9000:** With the BX9000, assignment of the IP address via a BootP server is activated via the navigation switch of the controller (see chapter [BX menu](#) [► 86] or [First Steps](#) [► 193]).

**BC9xx0:** With the BC9xx0, the assignment of the IP address via a BootP server is activated via the DIP switch.

### IP address save modes

#### BootP & Save

With *BootP & Save*, the IP address issued by the BootP server is stored on the BX controller, and the BootP service is not queried again at the next cold start.

The address can be cleared again by reactivating the manufacturers' settings (using the KS2000 software or by DIP switch and end terminal).

#### BootP

With BootP, the IP address assigned by the BootP server is only valid until the Bus Terminal Controller is switched off. At the next restart, the BootP server has to issue a new IP address for the Bus Terminal Controller.

The address is retained during a software reset of the Bus Terminal Controller.

## Beckhoff BootP server

Beckhoff supply a BootP server for Windows 98, ME, NT4.0, NT2000 and XP. You will find the installation version in the *Unsupported Utilities* folder on the *Beckhoff Software Products* CD, or on the internet under <ftp://ftp.beckhoff.com/>.

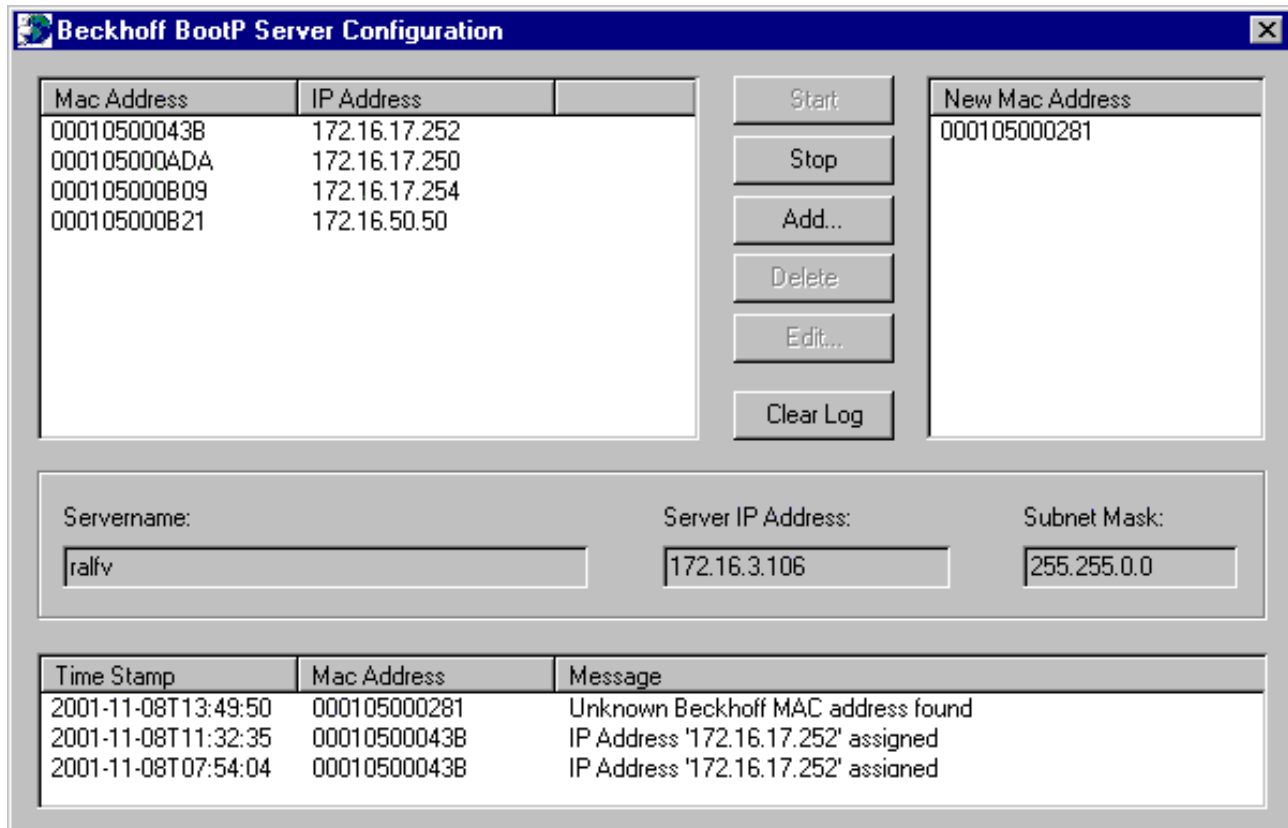


Fig. 23: Setting the address via BootP server

As soon as the BootP server has started, the *New MAC Address* window shows all the Beckhoff nodes that are working in BootP mode and still have not received an IP address. The assignment of the MAC-ID to IP address is made with the [**<<**] button. Successful assignment is displayed in the log window.

To start the BootP server automatically when your PC boots, it is only necessary to provide a shortcut in the Windows autostart folder. Include the */Start* parameter in the shortcut (.../TcBootPDlg.exe/start).

## 4.2.3 Setting the address via DHCP server

With the BX9000, the assignment of the IP address via a DHCP server is activated via the navigation switch of the Bus Terminal Controller (see chapter [BX menu](#) [► 86], or [First steps](#) [► 193]).

With the BC9xx0, the assignment of the IP address via a DHCP server is activated via the dip switch of the Bus Terminal Controller (see chapter *DIP switch*).

If DHCP is active, the Bus Terminal Controller is automatically assigned an IP number by the DHCP server. The DHCP server must know the MAC ID of the Bus Terminal Controller for this. The IP address should be set statically.

The DNS name is formed from the type and the last 3 byte of the MAC ID. The MAC ID is given on the production label of the Bus Terminal Controller.

### Example for BX9000

- MAC ID: 00-01-05-01-02-03
- DNS name: BX\_010203

**Example for BC9050**

- MAC ID: 00-01-05-03-02-01
- DNS name: BC\_030201

**4.2.4 Subnet mask**

The subnet mask is subject to the control of the network administrator, and specifies the structure of the subnet.

Small networks without a router do not require a subnet mask. The same is true if you do not use registered IP numbers. A subnet mask can be used to subdivide the network with the aid of the mask instead of using a large number of network numbers.

The subnet mask is a 32-bit number:

- Ones in the mask indicate the subnet part of an address space.
- Zeros indicate that part of the address space which is available for the host IDs.

Description	Binary representation	Decimal representation
IP address	10101100.00010000.00010001.11001000	172.16.17.200
Subnet mask	11111111.11111111.00010100.00000000	255.255.20.0
Network ID	10101100.00010000.00010000.00000000	172.16.16.0
Host ID	00000000.00000000.00000001.11001000	0.0.1.200

**Standard subnet mask**

Address class	Standard subnet mask (decimal)	Standard subnet mask (hex)
A	255.0.0.0	FF.00.00.00
B	255.255.0.0	FF.FF.00.00
C	255.255.255.0	FF.FF.FF.00

**Note****Note regarding subnets and host number**

Neither subnet 0 nor the subnet consisting only of ones may be used.

Neither host number 0 nor the host number consisting only of ones may be used!

If the IP address is set using the KS2000 configuration software, it is necessary for the subnet mask also to be changed with the KS2000 configuration software.

If ARP addressing is used, the associated standard subnet mask, based on the IP address, is entered.

Under BootP or DHCP the subnet mask is transmitted also by the server.

**4.3 Configuration****4.3.1 Overview****Configuration types**

The Bus Terminal controllers of the BCxx50, BCxx20 and BXxx00 series can be configured in two different ways: DEFAULT CONFIG or TwinCAT CONFIG.

**DEFAULT-CONFIG**

Bus Terminals are mapped in the order they are inserted, i.e. first the complex Bus Terminals followed by the digital Bus Terminals.

The complex Bus Terminals are mapped as follows:



- Word Alignment
- complex representation



**CAUTION**

**The process image depends on the connected terminals!**

The process image changes when a terminal is added or removed!

The data of the fieldbus slaves interface are referred to as PLC variables. The PLC variables have addresses from %QB1000 and %IB1000

The DEFAULT CONFIG (without PLC program) can also be used for writing and testing of the Connected Bus Terminals. To this end, the Bus Terminal Controller must be scanned in the System Manager, and FreeRun mode must be enabled (to use this function, no PLC program may be active on the Bus Terminal Controller).

## TWINCAT-CONFIG

In the TwinCAT CONFIG the Bus Terminals and PLC variables can be freely linked as required (TwinCAT System Manager file required). The configuration is transferred to the coupler via the System Manager and ADS.

The following is required for the TwinCAT configuration (TC file):

- Via the fieldbus (PROFIBUS, CANopen, Ethernet)
  - PROFIBUS: (BC3150, BX3100)
    - PC with FC310x from version 2.0 and TwinCAT 2.9 build 1000
    - BX3100 with CIF60 or CP5412
    - TwinCAT 2.9 build 946
 

(**NOTE:** with PROFIBUS cards from Hilscher only one ADS communication is permitted, i.e. either System Manager or PLC Control)
    - CANopen: (BC5150, BX5100)
    - PC with FC510x from version 1.76 TwinCAT build 1030
    - DeviceNet: (BC5250, BX5200)
    - on request
    - Ethernet: (BC9050, BC9020, BC9120, BX9000)
    - PC with TwinCAT 2.10 build 1322
- Via the serial ADS TwinCAT 2.9 build 1010
  - BX3100 version 1.00
  - BX5100 version 1.00
  - BX5200 version 1.10
  - BX8000 version 1.00
  - BC3150, BC5150, BC5250, BC9050, BC9020, BC9120 from firmware B0
  - For BC8150 from TwinCAT 2.10 build 1243

BCxx50 and Bxxx00 can be parameterized via the System Manager of the TwinCAT program.

- Variable I/O mapping
- Type-specific PROFIBUS data (BC3150 and BX3100 only)
- RTC (real-time clock) (BX series only)
- SSB (Smart System Bus) (BX series only)
- PLC settings
- K-Bus settings

The configuration can be transferred to the BCxx50 or Bxxx00 via fieldbus ADS protocol or serial ADS protocol.

The TwinCAT configuration can be used to link variables, I/Os and data. The following is possible:

- PLC - K-BUS
- PLC fieldbus (e.g. PROFIBUS slave interface to PLC)
- K-bus fieldbus (only for BX controllers)
- Support for TwinSAFE terminals (only BX controllers from firmware 1.17)

In addition, the TwinCAT configuration can be used to parameterize special behavior, for example whether data are preserved or set to "0" in the event of a fieldbus error.

The real-time clock can be set via a tab in the system manager.

### Work steps

1. Setting the fieldbus address
2. Open the System Manager and create a TC file
3. Configure fieldbus data in the TC file
4. Save the TC file
5. Opening a new system manager, creating a PC file and reading in saved TX file
6. Creating a link to a PLC task
7. Saving the configuration
8. Starting the TwinCAT system
9. Open the TC file in the System Manager, complete the configuration and transfer it to the BCxx50, BCxx20 or BXxx00
10. Transfer the program to BCxx50, BCxx20 or BXxx00
11. Creating a boot project

## 4.3.2 Creating a TwinCAT configuration

In order to configure a Bus Terminal Controller of the BCxx50, BCxx20 or BXxx00 series, create a BX file in the System Manager. To simplify matters, files for the basic units have already been prepared. Open the corresponding Bus Terminal Controller with *New from Template*.

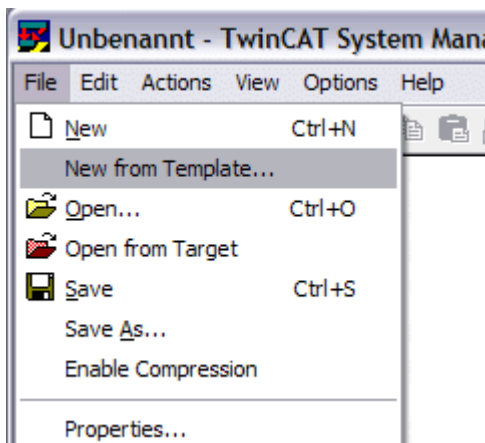


Fig. 24: Creating a TwinCAT configuration

Select the corresponding Bus Terminal Controller.

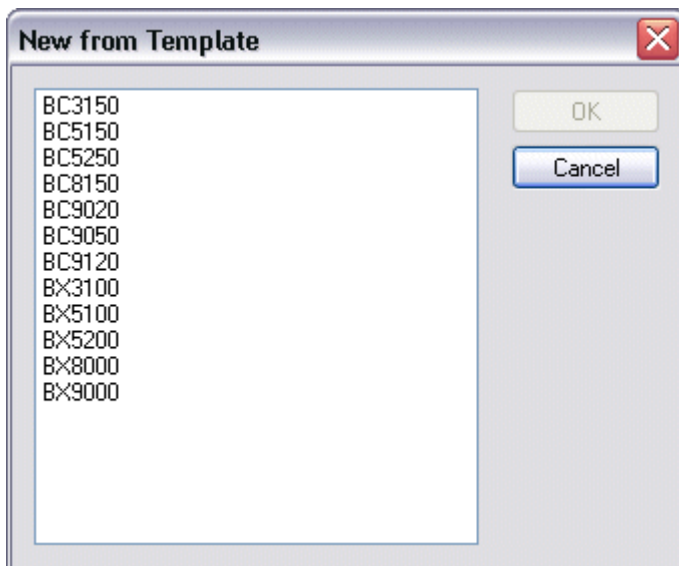


Fig. 25: Selecting the Bus Terminal Controller

All Bus Terminal Controller components are now available:

- Fieldbus interface
- K-bus interface [[► 46](#)]
- PLC Program [[► 49](#)]
- SSB [[► 52](#)] (only Bus Terminal Controllers of the BX series)

Please refer to the relevant chapter for device configuration.

### 4.3.3 Downloading a TwinCAT configuration

The TwinCAT configuration is loaded into the Bus Terminal Controller via ADS protocol.

#### Serial ADS protocol

(all Bus Terminal Controllers of the BXxx00 and BCxx50 series)

Enter the serial ADS connection, as described in the chapter [Serial ADS](#) [[► 40](#)].

#### ADS protocol via the fieldbus

(BC3150, BC5150, BC9x20, BC9050, BX3100, BX5100, BX9000 only)

A prerequisite is that TwinCAT operates as master and is engaged in data exchange, i.e. the physical and fieldbus configuration must be complete, and data exchange must take place between the master (e.g. fieldbus master card) and the Bus Terminal Controller.

#### Choose Target System

Select the Bus Terminal Controller onto which the configuration is to be loaded. Use the function key F8 to open the dialog for downloading your file to the corresponding device.

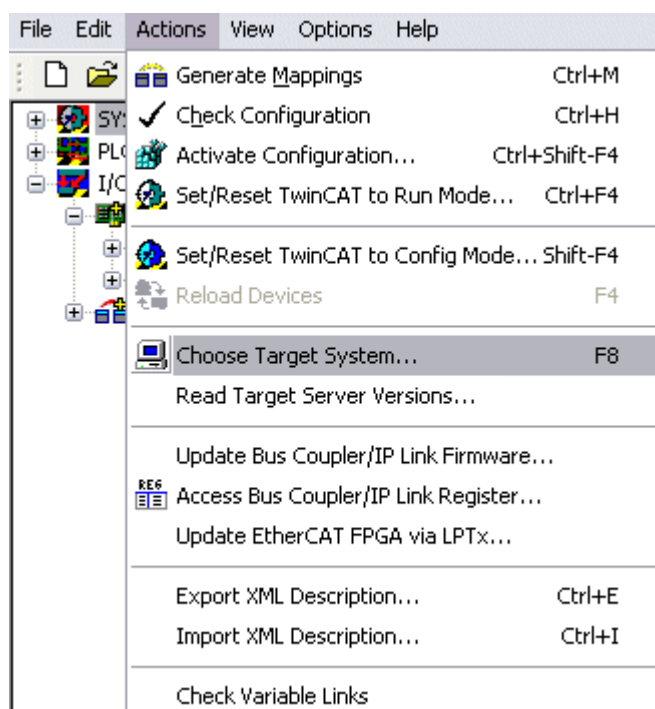


Fig. 26: Downloading a TwinCAT configuration

Select the corresponding Bus Terminal Controller.

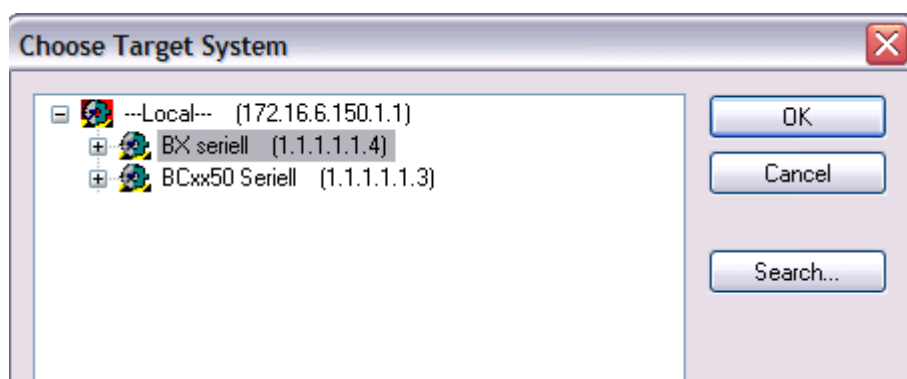


Fig. 27: Selecting the Bus Terminal Controller

The state of the Bus Terminal Controller is shown at the bottom right of the System Manager.

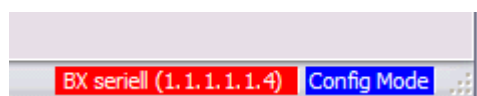


Fig. 28: State of the Bus Terminal Controller

In *Config mode / FreeRun* the configuration can now be downloaded to Bus Terminal Controller. If the Bus Terminal Controller is in *Stop mode*, ADS communication is not yet activated. In this case, it is not possible to download the configuration.

To activate the TwinCAT configuration select Ctrl+Shift+F4 or *Activate Configuration*.

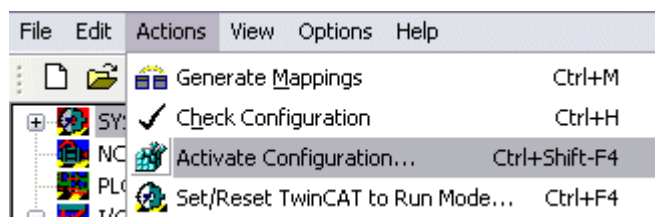


Fig. 29: Activating the TwinCAT configuration

The current configuration is loaded onto the Bus Terminal Controller. The display will show *Store Config*, and the BUS and I/O LED will flash. Once the configuration is successfully loaded onto Bus Terminal Controller, *TwinCAT Config* should appear in the display of a BXxx00. The corresponding program can now be transferred to the Bus Terminal Controller (program-download via the fieldbus) [► 169].

### 4.3.4 Uploading a TwinCAT configuration

The TwinCAT configuration is loaded into the Bus Terminal Controller via ADS protocol.

#### Serial ADS protocol

(all Bus Terminal Controllers of the BCxx50, BCxx20 and BXxx00 series)

Enter the serial ADS connection, as described in the chapter Serial ADS [► 40].

#### ADS protocol via the fieldbus

(BC3150, BC5150, BC9x20, BC9050, BX3100, BX5100, BX9000 only)

A prerequisite is that TwinCAT operates as master and is engaged in data exchange, i.e. the physical and fieldbus configuration must be complete, and data exchange must take place between the master (e.g. fieldbus card) and the Bus Terminal Controller.

#### Choose Target System

Select the Bus Terminal Controller onto which the configuration is to be loaded. Use the function key [F8] to open the dialog for downloading your file to the corresponding device.

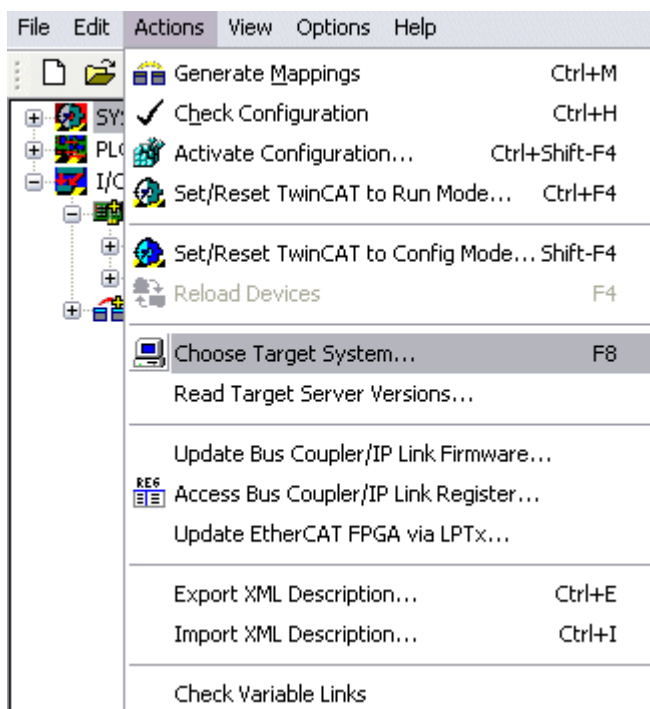


Fig. 30: Choose Target System

Select the corresponding Bus Terminal Controller.

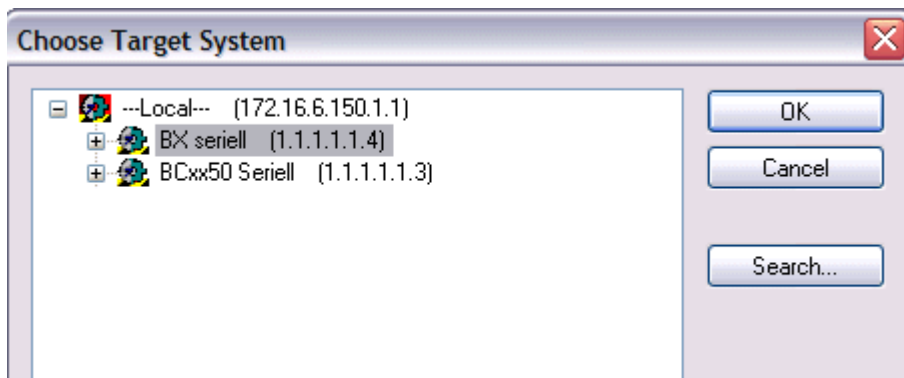


Fig. 31: Selecting the Bus Terminal Controller

The state of the Bus Terminal Controller is shown at the bottom right of the System Manager.

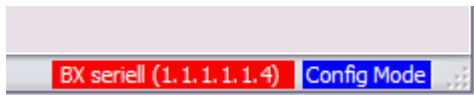


Fig. 32: State of the Bus Terminal Controller

Click on the red folder. The TwinCAT configuration will now be uploaded.

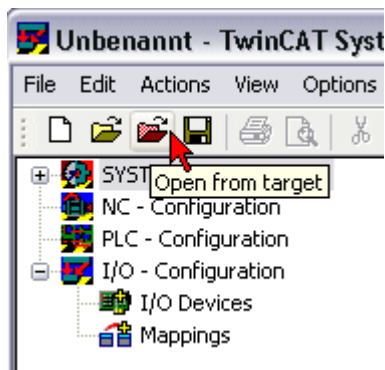


Fig. 33: Uploading the TwinCAT configuration

### 4.3.5 Resources in the Bus Terminal Controller

The memory resources assigned in the Bus Terminal Controller are shown in the System Manager in the *Resources* tab of the Bus Terminal Controller.

#### Mapping code

The mapping code is required for calculating the TwinCAT configuration (see Figure *Memory for the code mapping*). The percentages are added here. In the example from Fig. *Memory for code mapping*, 8% of the memory is allocated to the mapping calculation.

Process Image	Map In	Map Out	Code In	Code Out
! PLC1<->KBUS	0%	1%	1%	3%
! PLC1<->FBUS	0%	0%	0%	0%
! PLC1<->SSB	1%	1%	2%	2%

Used Near Heap:     Used Plc Code:   
 Used Huge Heap:     Used Plc Data:   
 Used File Area:     Used Plc Source:

Fig. 34: Memory for code mapping

### Data memory mapping

Data memory for mapping. The values are to be considered individually, i.e. each value can be up to 100%.

Process Image	Map In	Map Out	Code In	Code Out
! PLC1<->KBUS	0%	1%	1%	3%
! PLC1<->FBUS	0%	0%	0%	0%
! PLC1<->SSB	1%	1%	2%	2%

Used Near Heap:     Used Plc Code:   
 Used Huge Heap:     Used Plc Data:   
 Used File Area:     Used Plc Source:

Fig. 35: Data memory mapping

### Used code and data memory

Fig. Code and data memory (1) "Used PLC code" in %.

Fig. Code and data memory (2) "Used PLC data" in %.

Fig. Code and data memory (3) "Used PLC source" in %.

Process Image	Map In	Map Out	Code In	Code Out
! PLC1<->KBUS	0%	1%	1%	3%
! PLC1<->FBUS	0%	0%	0%	0%
! PLC1<->SSB	1%	1%	2%	2%

Used Near Heap: 17      Used Plc Code: 32  
 Used Huge Heap: 3      Used Plc Data: 13  
 Used File Area: 23      Used Plc Source: 71

Update

Fig. 36: Code and data memory

### Other memory

Fig. *Other Memory* (1) "Used Near Heap" is required for the COM interface and SSB. % values.

Fig. *Other Memory* (2) "Used Huge Heap" is required for the ADS communication. % values. This value should be less than 30 %.

Fig. *Other Memory* (3) "Used File Area" is required for the TwinCAT configuration, the TSM file and the 16 kbyte flash access. % values.

Process Image	Map In	Map Out	Code In	Code Out
! PLC1<->KBUS	0%	1%	1%	3%
! PLC1<->FBUS	0%	0%	0%	0%
! PLC1<->SSB	1%	1%	2%	2%

Used Near Heap: 17      Used Plc Code: 32  
 Used Huge Heap: 3      Used Plc Data: 13  
 Used File Area: 23      Used Plc Source: 71

Update

Fig. 37: Other memory

### 4.3.6 ADS connection via serial interface

(from firmware version 1.xx or 0.99x, Bus Terminal Controllers of the BX series and for all BCxx50)

From TwinCAT 2.9 build 1020 (TwinCAT level PLC, NC or NCI)



**Note****Use only a serial connection**

To ensure trouble-free operation of the ADS link via the serial interface, only a serial connection to the BX controller is allowed.  
After successful configuration via the System Manager, close the System Manager before starting programming.

**Note****AMS Net ID in delivery state (default)****For BX9000**

The default AMS Net ID is 172.16.21.20.1.1. If the IP address of the BX9000 is changed, the AMS Net ID of the BX9000 also changes. There is a menu option for displaying the current AMS Net ID.

Example: If you change the IP address to 10.2.3.7, the AMS Net ID changes to 10.2.3.7.1.1.

**For BC9050, BC9020, BC9120**

The default AMS Net ID is 172.16.xxx.[DIP switch].1.1. If the IP address of the BX9xxx is changed, the AMS Net ID of the BX9xxx also changes.

Example: If you change the IP address to 10.2.3.7, the AMS Net ID changes to 10.2.3.7.1.1.

BC9050: DEFAULT 172.16.21.[DIP-Switch].1.1

BC9020: DEFAULT 172.16.22.[DIP-Switch].1.1

BC9120: DEFAULT 172.16.23.[DIP-Switch].1.1

**Initializing the ADS connection**

Enter the Bus Terminal Controller in the remote connection under TwinCAT. Click on the TwinCAT icon and open the features menu. The following settings can be made under the >AMS Remote< tab.

Fig. 38: Properties of the remote connection

Remote Name: Any

AMS-Net-ID: 1.1.1.1.1.1 (Default)

Address: COM Port: Baud rate, parity, data bits, stop bits

Transport: Select "COM port"

When the Bus Terminal Controller is switched on, the default AMS Net ID is always "1.1.1.1.1.1" (except all Ethernet Controllers).

The AMS Net ID can be changed as required. Please note that the new AMS Net ID cannot be changed again in this way.

If you need to change the new AMS Net ID again, you have to restart the Bus Terminal Controller, so that the AMS Net ID is reset to the default AMS Net ID, "1.1.1.1.1.1".

You can now change the AMS Net ID again.

**Note****Strings can only be entered at the second call**

No strings can be entered under address when the dialog is first called (see above). Enter the name, AMS Net ID and transport type and close the dialog. With the second call you can enter your COM port.

The communication starts when TwinCAT is in Config mode (TwinCAT icon is blue) or RUN mode (TwinCAT icon is green). The COM interface remains open until a TwinCAT stop occurs (TwinCAT icon is red). It is then available again for other programs. No error message is issued if the COM interface is used by another program during a TwinCAT restart (e.g. by the KS2000 configuration software).

**Note****AMS Net ID after ADS connection via the fieldbus**

If you have addressed the Bus Terminal Controller with an ADS connection via the fieldbus before the serial ADS was used, the AMS Net ID was automatically changed by the System Manager. In this case a new serial ADS connection is only possible, if the AMS Net ID is adjusted.

**BX series: reading the AMS Net ID**

The current AMS Net ID can be read from the menu via the display of BX series Bus Terminal Controller.

**AMS**                      AMS Net ID  
1.1.1.1.1.1

## 4.3.7      ADS connection via Ethernet

### 4.3.7.1    Finding the Bus Terminal Controller with the TwinCAT System Manager

Prerequisite for BX9000:

- BX firmware 1.08 or above (see the indication on the BX9000 display after the operating voltage has been switched on).
- from TwinCAT 2.10 Build 1251

Prerequisite for BC9050, BC9020, BC9120:

- from Firmware B1
- from TwinCAT 2.10 Build 1322

An operable ADS connection is necessary in order to set the IP address through the System Manager. This can be done serially or via Ethernet.

A functioning Ethernet connection is necessary for the ADS connection via Ethernet. You can test the IP connection with the PING command. By default the Bus Terminal Controller is set to 172.16.21.20 with the sub-network mask 255.255.0.0. Set your PC to the same network class, e.g. 172.16.200.100 (subnet mask 255.255.0.0).

Now use PING to test whether a connection exists:

Now start the TwinCAT System Manager and look for the Bus Terminal Controller using the button highlighted in the image or press [F8]:

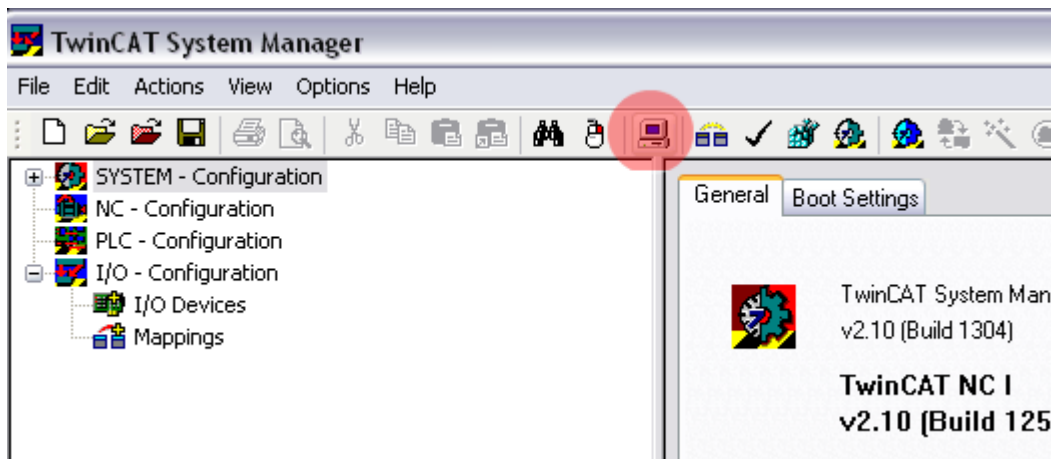


Fig. 39: Specifying the target system

Now look for the Bus Terminal Controller via Ethernet:

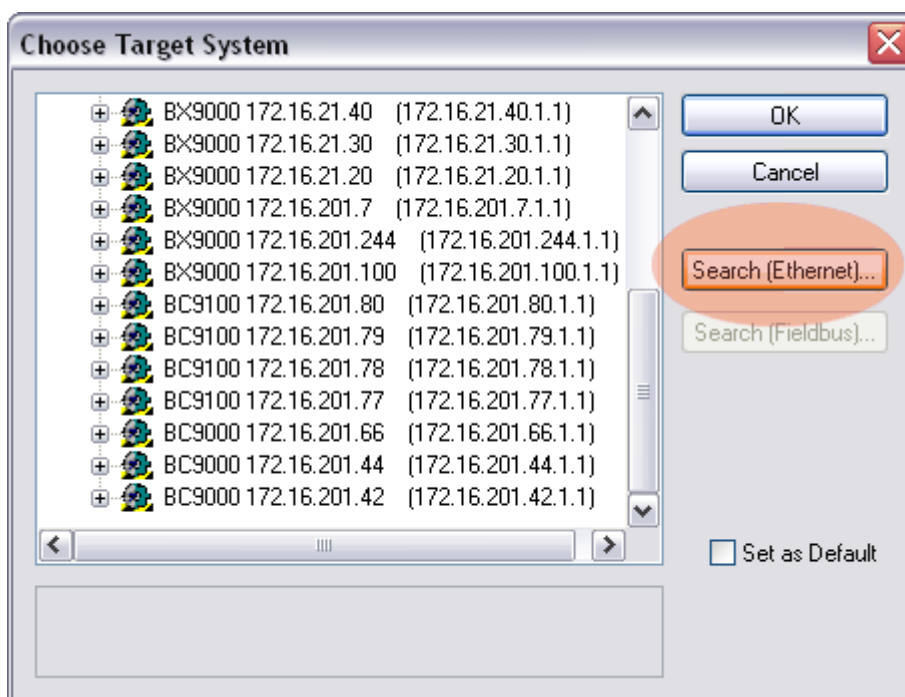


Fig. 40: Finding the Bus Terminal Controller

Now find your Bus Terminal Controller via *Broadcast Search*. If you have several Bus Terminal Controllers in your network, you can distinguish them by their names.

The name consists of the characters "BX\_" or "BC\_" and the last three bytes of the MAC ID. For the BX9000 the MAC ID is printed on the side, for the BC controllers it is printed on the underside. Example: MAC-ID: 00-01-05-00-01D-C3, then the default name is BX\_001DC3.

If the Broadcast Search fails to find a device, check the Ethernet connection and the firmware in the controller.

If the Bus Terminal Controller was addressed via DHCP, you can include the Bus Terminal Controller in your connection via the button *Add Route* in the *Host Name* dialog.

If the Bus Terminal Controller was addressed manually or via BootP, select the assigned *IP address* and establish the connection via the button *Add Route*.

Acknowledge the password query dialog without making an entry. No password is required for Bus Terminal Controllers. Bus Terminal controllers do not offer password support.

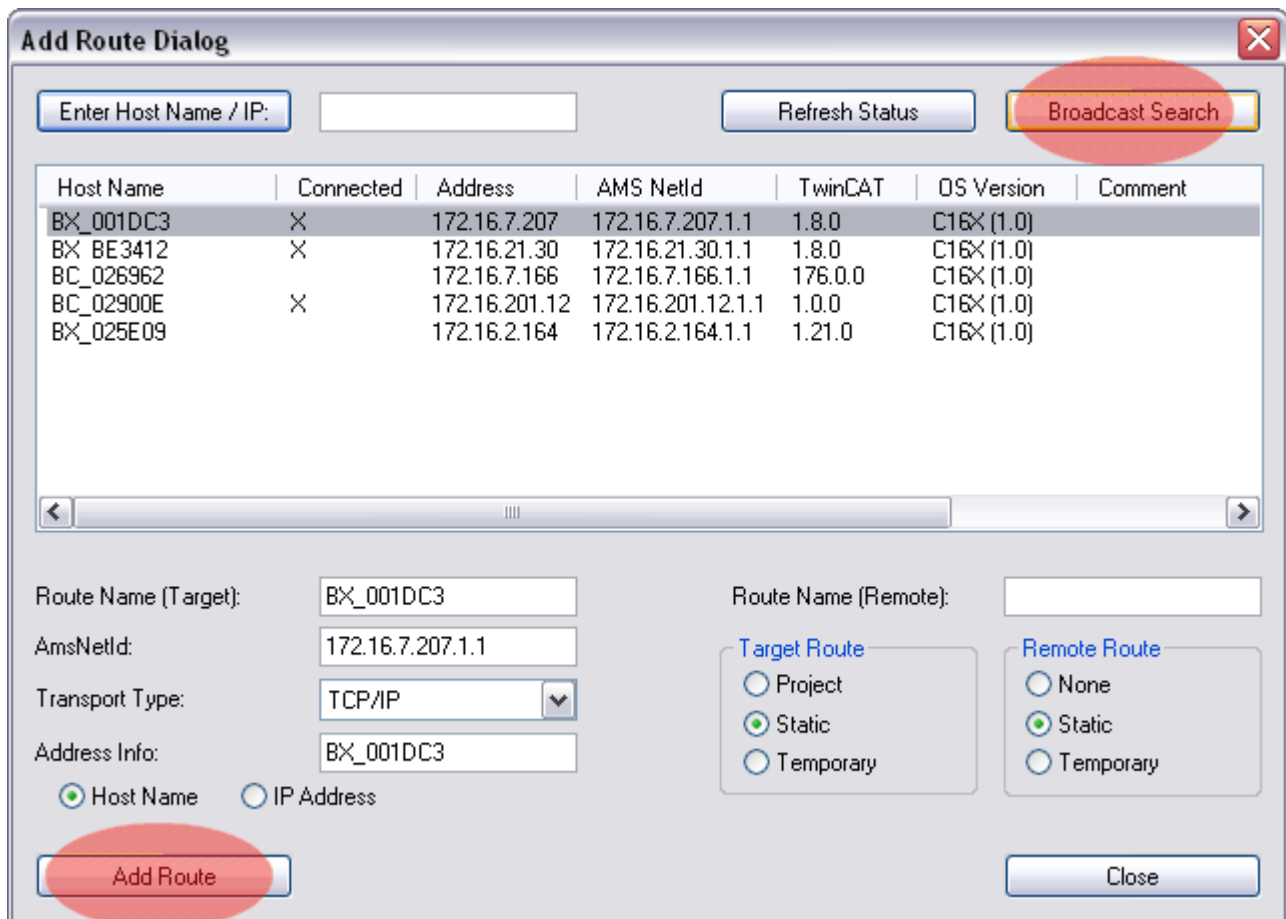


Fig. 41: Selecting the Bus Terminal Controller

Now select the Bus Terminal Controller you want to connect to and scan the devices connected to it (SSB, K-bus). The Bus Terminal Controller must be in Config mode ([Shift] [F4]).

#### 4.3.7.2 TwinCAT PLC as Master

A PC or a CX can act as master for the BC9x20/BC9050/BX9000. It then polls the PLC variables, in accordance with the set task time. In this way it is possible to transfer data between a control system and a BC9x20/BC9050/BX9000. The following communication methods are permitted:

- ADS TCP cyclic or acyclic from the PLC using the ADS READ and WRITE function blocks
- ADS UDP cyclic
- ModbusTCP (with TC Modbus client)
- Bus Terminal Controller sends or reads data from the PLC using the ADS READ and WRITE function blocks.

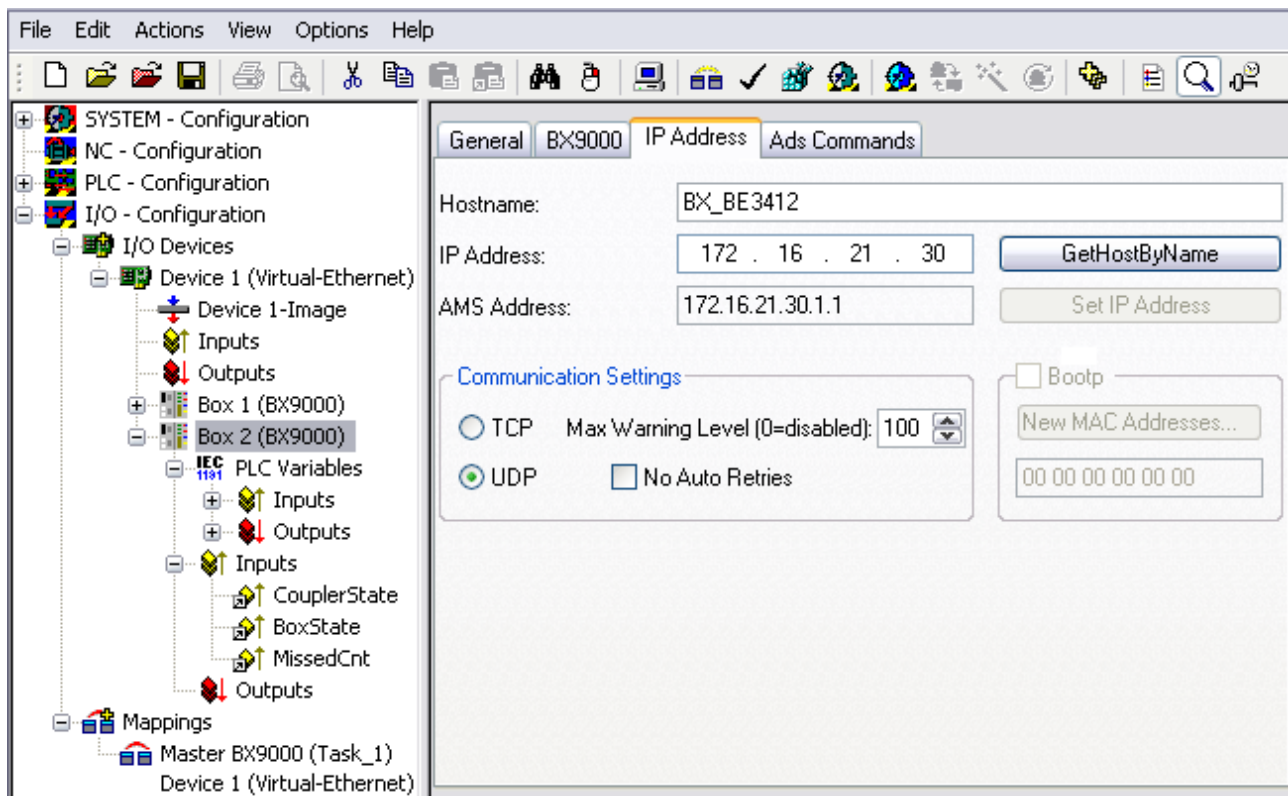


Fig. 42: IP address

**GetHostByName:** This function enables the IP address to be found based on the name (this only works if the IP address of the Bus Terminal Controller was assigned via DHCP)

**PLC variables:** Data for the cyclic data connection. This must be linked into at least one task. 256 words of input or output is the maximum. Should more data be required for the transfer, these can be read or written acyclically via the flag area of the Bus Terminal Controller.

Diagnostic Data:

**Coupler State:** Should always be zero. "1" is set if, for instance, the K-Bus signals an error.

**BoxState:** See the comment in the dialogue

**MissedCnt:** This should, if possible, not increment. Because TwinCAT operates in real time, but neither TCP nor UDP are real-time protocols, is not impossible that the counter will increase under certain circumstances. The counter is incremented every time that data that is been transmitted at the beginning of the task has not yet returned by the time the task starts again.

**The task time should be set in the following way**

#### For ADS TCP cyclic

Measure the PLC time required on the Bus Terminal Controller and set the task time on the Bus Terminal Controller by adding 20-30 %. Now take three times this task time; the result corresponds to the task time on your master controller. Example: measured PLC time is 5 ms; set a task time of 7 ms for the Bus Terminal Controller and  $3 \times 7 \text{ ms} = 21 \text{ ms}$  for the master PLC. If several Bus Terminal Controllers are used, the one with the slowest PLC time determines the task time for the master controller.

#### For ADS UDP cyclic

Measure the PLC time required on the Bus Terminal Controller and set the task time on the Bus Terminal Controller by adding 20-30 %. Now take two times this task time; the result corresponds to the task time on your master controller. Example: measured PLC time is 5 ms; set a task time of 7 ms for the Bus Terminal Controller and  $2 \times 7 \text{ ms} = 14 \text{ ms}$  for the master PLC. If several Bus Terminal Controllers are used, the one with the slowest PLC time determines the task time for the master controller.

### For ModbusTCP cyclic

Measure the PLC time required on the Bus Terminal Controller and set the task time on the Bus Terminal Controller by adding 20-30 %. Now take two times this task time; the result corresponds to the task time on your master controller. Example: measured PLC time is 5 ms; set a task time of 7 ms for the Bus Terminal Controller and  $2 \times 7 \text{ ms} = 14 \text{ ms}$  for the master PLC. If several Bus Terminal Controllers are used, the one with the slowest PLC time determines the task time for the master controller.

### Different PLC cycle times

If the Bus Terminal Controllers used in your system have different cycle times for local PLC processing, you can adjust the time, based on which the higher-level TwinCAT PLC queries each individual Bus Terminal Controller.

The screenshot shows the configuration interface for a Beckhoff device. At the top, there are tabs for 'General', 'BX9000', 'IP Address', and 'Ads Commands'. Below these are buttons for 'K-Bus Reset' and 'Firmware Update (via ADS) ...'. The 'Data exchange' section contains 'Divider' (set to 2) and 'Modulo' (set to 0). The 'VLAN Support' section has an 'Enable' checkbox (unchecked), 'Priority' (set to 0), and 'Id' (set to 0). The 'PLC' section has two rows: 'PLC Project' and 'TC Config', each with a text input field and an 'Open...' button.

Fig. 43: Different PLC cycle times

#### Divisor

Use the divisor for this purpose. The divisor used the cycle time of the higher-level TwinCAT PLC, e.g. 10 ms. If the divisor is set to 2, a telegram is sent to the Bus Terminal Controller every  $2 \times 10 \text{ ms}$ , i.e. every 20 ms.

#### Modulo

Modulo can be used to set the timing for the higher-level TwinCAT PLC.

Example:

Divisor 3, Modulo 0 means a telegram is sent after the first task cycle and then after each third task cycle. If Modulo is set to 1, a telegram is sent after the second task cycle and then after each third task cycle + 1. In systems with many Ethernet nodes this enables the number of Ethernet packets to be distributed more evenly, which results in more uniform network load and avoids network load peaks.

## 4.3.8 K-bus



#### Note

#### Bus Terminal and end terminal required

To operate a Bus Terminal Controller of the BC or BX series, at least one Bus Terminal with process image and the end terminal must be connected to the K-bus.

## BX Settings tab

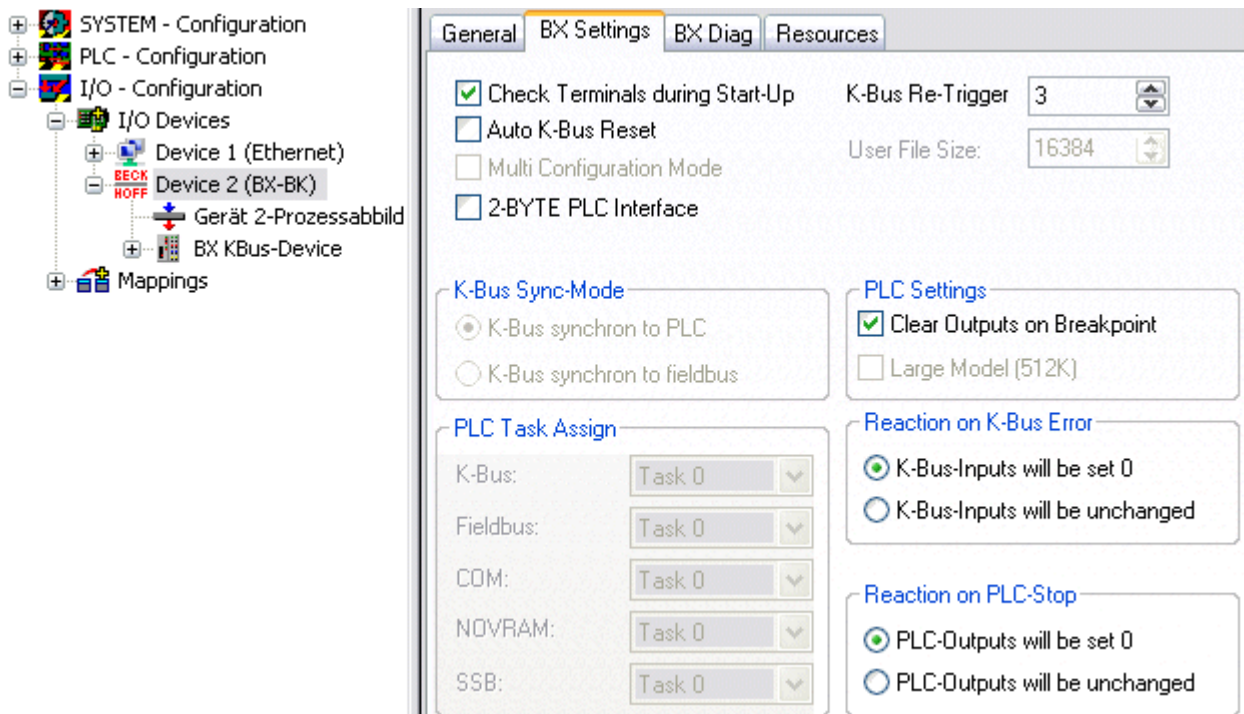


Fig. 44: BX Settings tab

### Check Terminals during Start-up

When a boot project is created, the current Bus Terminal configuration is stored. The connected Bus Terminals are checked when the Bus Terminal Controller restarts. If this option is selected, the Bus Terminal Controller does not enter into data exchange. The PLC project will not be started.

### Auto K-Bus Reset

Once a K-bus error has been rectified, the Bus Terminal Controller automatically resumes the data exchange.



**Once a K-Bus error has been rectified, the outputs become active again immediately!**

Ensure that the outputs are reactivated immediately and that analog outputs retain their programmed value, if this is not dealt with in your PLC project.

### Clear Outputs on Breakpoint

If breakpoints are set in PLC Control, the K-Bus is no longer processed, i.e. the outputs are set to a safe state (zero).

### K-Bus Sync Mode

Writing and reading of the Bus Terminals can take place synchronously with task 1 or the fieldbus.

### K-Bus Re-Trigger

If the processor is busy dealing with the PLC project or the SSB, the K-Bus cannot be processed for a certain amount of time. This leads to triggering of the Bus Terminal watchdog and dropping of the outputs. The Bus Terminal Controller is set such that the K-bus watchdog is re-triggered 3 times after 85 ms. The K-Bus watchdog would then be activated.

K-Bus Re-Trigger 0: 100 ms

K-Bus Re-Trigger 1:  $2 \times 85 \text{ ms} = 170 \text{ ms}$

K-Bus Re-Trigger 2:  $3 \times 85 \text{ ms} = 255 \text{ ms}$

K-Bus Re-Trigger 3:  $4 \times 85 \text{ ms} = 340 \text{ ms}$



## Reaction on K-Bus Error

In the event of a K-Bus error, the K-Bus inputs are set to "0" or retain their last state.

## Response on PLC-Stop

The user can set the behavior of the fieldbus output data in the event of the PLC project being stopped. The master will use these data as input data. In the event of a PLC stop, the data can be set to "0" or remain unchanged.

## BX Diag tab

Display of the cycle time for task 1, K-bus, fieldbus processing and the SSB load.

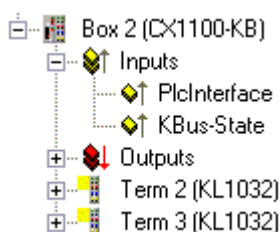
	Actual Value	Maximum Value
PLC-Task 1 (μs):	72	144
PLC-Task 2 (μs):		
PLC-Task 3 (μs):		
PLC-Task 4 (μs):		
K-Bus (μs):	246	303
Fieldbus (μs):	21	31
SSB (μs):		
SSB-Overhead (%):		
Display 1:	TWINCAT-CONFIG	
Display 2:	BC9020PROJEKT	

Fig. 45: BX Diag tab

**Factory Settings:** the Bus Terminal Controller is set to its delivery. These settings are reactivated via Restart System or by switching the system off and on again (display shows DEFAULT-CONFIG).

**Reset Maximum Values:** resets the maximum values

## K-Bus variables



**PLC interface:** Not supported (only included for moving CX or BX projects)

**K-bus state:** see [Diagnostics](#) [► 187]



## 4.3.9 PLC

### 4.3.9.1 Inserting a PLC project

For variable mapping, configuration has to be specified in the system manager. This is where the link between PLC and hardware is specified. The variables can process and link bit, byte, word or data structures. Automatic addressing via the System Manager is possible, but should be checked for offset.



#### Note

#### Word alignment, byte orientation

With data structures, ensure that the Bus Terminal Controller saves the data in word alignment and the System Manager operates byte-oriented (see [Data structures](#) [► 101])

A valid project has to be compiled and saved in PLC Control. These data are saved as a \*.tpy file. For inserting a PLC project, right-click on *PLC - Configuration*. Select your current PLC project.

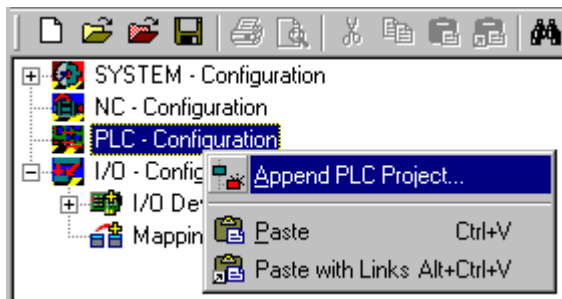


Fig. 46: Selecting the PLC project

Link the PLC variable with the hardware (e.g. digital Bus Terminal).

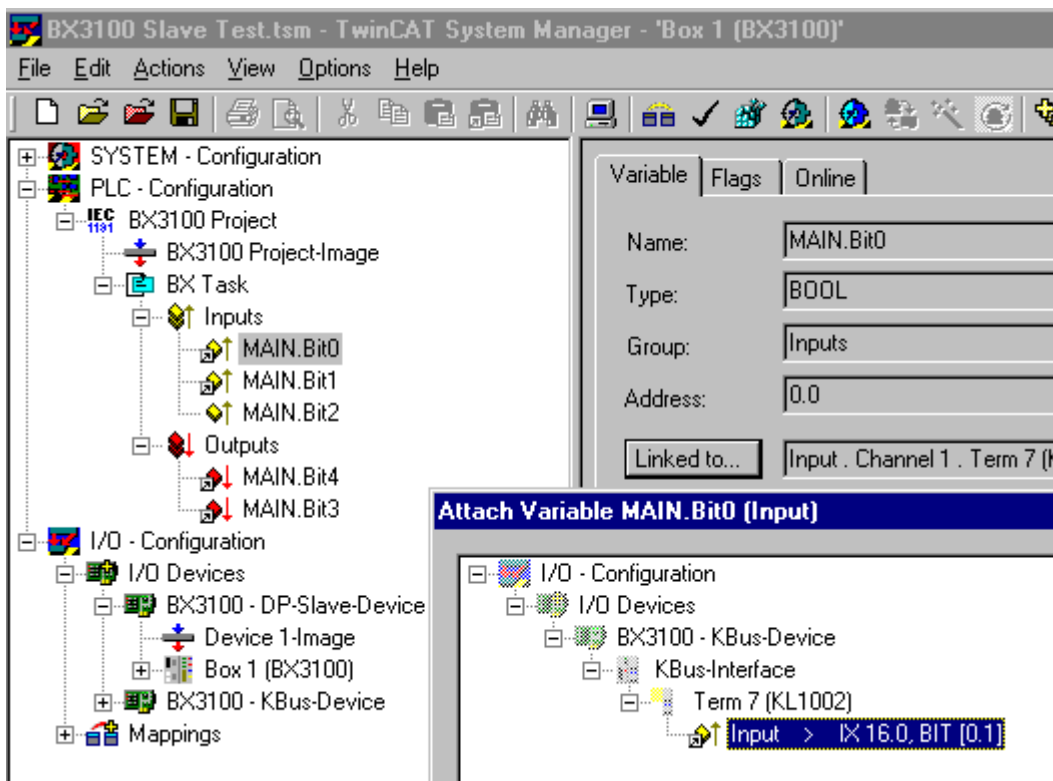


Fig. 47: Connecting PLC variable and hardware

Once all links have been created, activate the configuration *Actions/Activate Configuration* (Ctrl+Shift+F4) and start TwinCAT *Set/Reset TwinCAT to Run Mode*. Ensure that you have selected the correct target system (bottom right in the System Manager window).

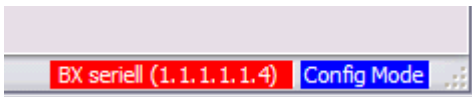


Fig. 48: Target system display

#### 4.3.9.2 Measuring the PLC cycle time

The task time is set in PLC Control. The default setting is 20 ms.

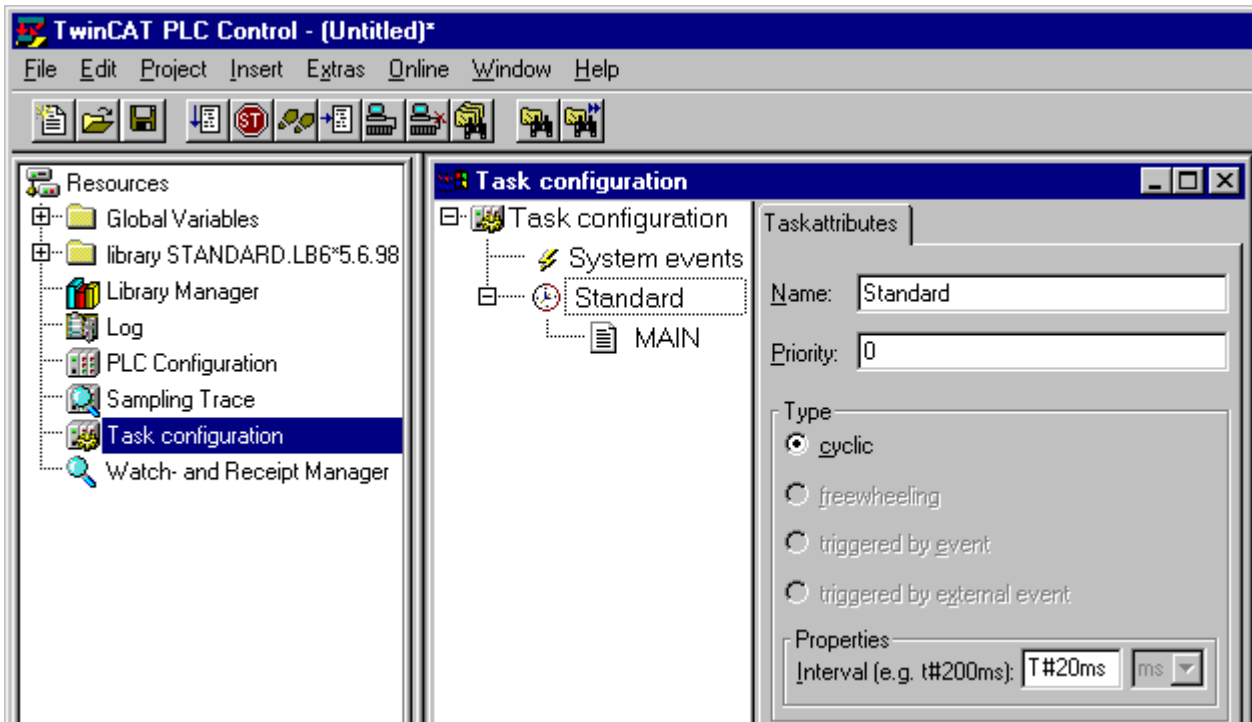


Fig. 49: Setting the task time

In the default setting, the PLC program is called every 20 ms, as long as the general cycle time is less than 20 ms. To determine the load of your system, the PLC cycle time can be measured in the System Manager. In order to ensure trouble-free operation, the set task time should be 20-30 % higher than the measured total cycle time. A precise cycle time breakdown can be found under [K-Bus tab \[► 46\]](#) description. The total cycle time is displayed with the TcBase library (see TcBase.lbx or TcBaseBCxx50.lbx).

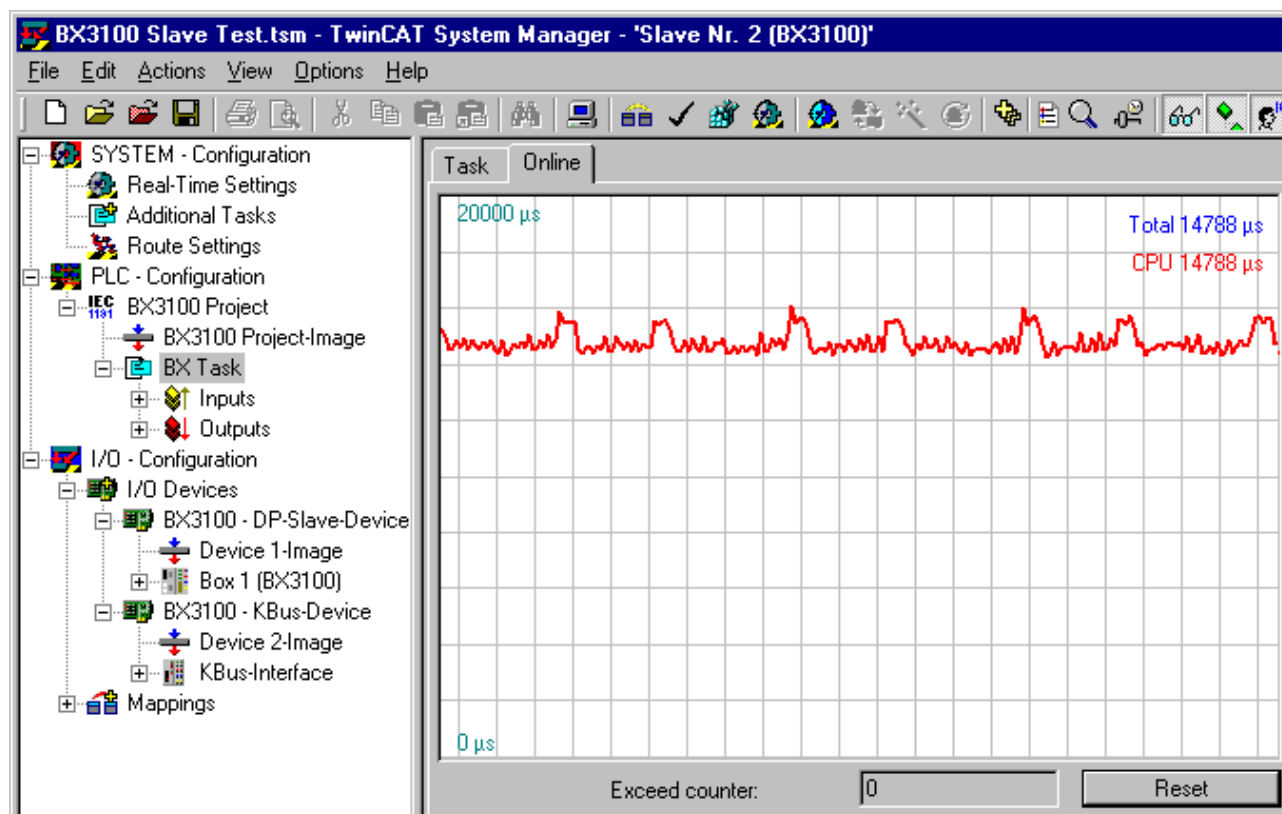


Fig. 50: Displaying the PLC cycle time

## 4.3.10 SSB

### 4.3.10.1 SSB overview

The SSB (Smart System Bus) is a sub-bus system based on CANopen. It is a CANopen master with limited functionality. CANopen slaves may be connected to this interface for reading or writing distributed I/Os. Parameterization SDOs (service data objects) can be sent to the slave via a start-up window.

#### Configuration

The SSB is configured via the TwinCAT System Manager (see TwinCAT config). The configuration is then loaded onto the BX controller via ADS.

#### Technical data

SSB	Data
Max. number of slaves	8
Max. number of PDOs	32 RxPDOs / 32 TxPDOs
Baud rate	10 kbaud to 1 Mbaud
Permitted slave addresses	1 to 64

#### Sync telegram

The sync telegram is transferred depending on the PLC task time. If a task time of 20 ms is set, the sync telegram is also sent every 20 ms (asynchronous with the PLC run time). The sync telegram is only generated when a device requires it and is configured accordingly. Sync telegrams are supported from firmware 1.12.

## Guarding

Guarding is supported and is sent after a configurable interval.

### 4.3.10.2 CANopen cabling

Notes related to checking the CAN wiring can be found in the [Trouble Shooting](#) [► 66] section.

#### 4.3.10.2.1 CAN topology

CAN is a 2-wire bus system, to which all participating devices are connected in parallel (i.e. using short drop lines). The bus must be terminated at each end with a 120 (or 121) Ohm terminating resistor to prevent reflections. This is also necessary even if the cable lengths are very short!

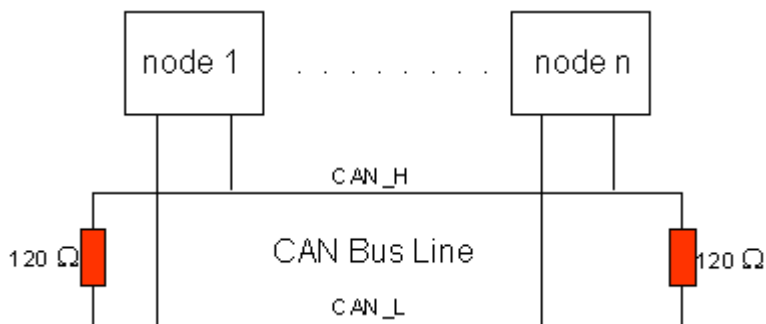


Fig. 51: Termination of the bus with a 120 Ohm termination resistor

Since the CAN signals are represented on the bus as the difference between the two levels, the CAN leads are not very sensitive to incoming interference (EMI): Both leads are affected, so the interference has very little effect on the difference.

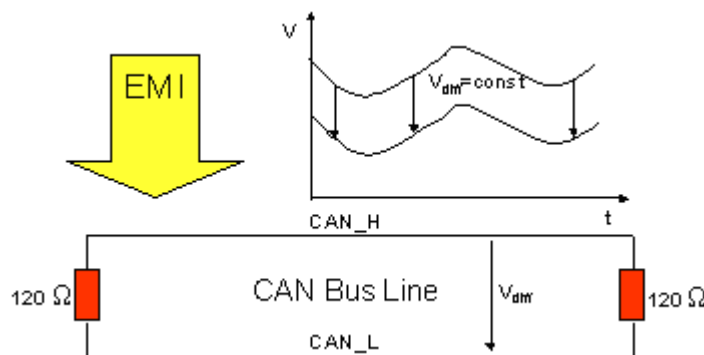


Fig. 52: Insensitivity to incoming interference

#### 4.3.10.2.2 Bus length

The maximum length of a CAN bus is primarily limited by the signal propagation delay. The multi-master bus access procedure (arbitration) requires signals to reach all the nodes at effectively the same time (before the sampling within a bit period). Since the signal propagation delays in the CAN connecting equipment (transceivers, opto-couplers, CAN controllers) are almost constant, the line length must be chosen in accordance with the baud rate:

Baud rate	Bus length
1 Mbit/s	< 20 m*
500 kbit/s	< 100 m
250 kbit/s	< 250 m
125 kbit/s	< 500 m
50 kbit/s	< 1000 m
20 kbit/s	< 2500 m
10 kbit/s	< 5000 m

\*) A figure of 40 m at 1 Mbit/s is often found in the CAN literature. This does not, however, apply to networks with optically isolated CAN controllers. The worst case calculation for opto-couplers yields a figure 5 m at 1 Mbit/s - in practice, however, 20 m can be reached without difficulty.

It may be necessary to use repeaters for bus lengths greater than 1000 m.

#### 4.3.10.2.3 Drop lines

Drop lines must always be avoided as far as possible, since they inevitably cause reflections. The reflections caused by drop lines are not however usually critical, provided they have decayed fully before the sampling time. In the case of the bit timing settings selected in the Bus Couplers it can be assumed that this is the case, provided the following drop line lengths are not exceeded:

Baud rate	Drop line length	Total length of all drop lines
1 Mbit/s	< 1 m	< 5 m
500 kbit/s	< 5 m	< 25 m
250 kbit/s	< 10 m	< 50 m
125 kbit/s	< 20 m	< 100 m
50 kbit/s	< 50 m	< 250 m

Drop lines must not have terminating resistors.

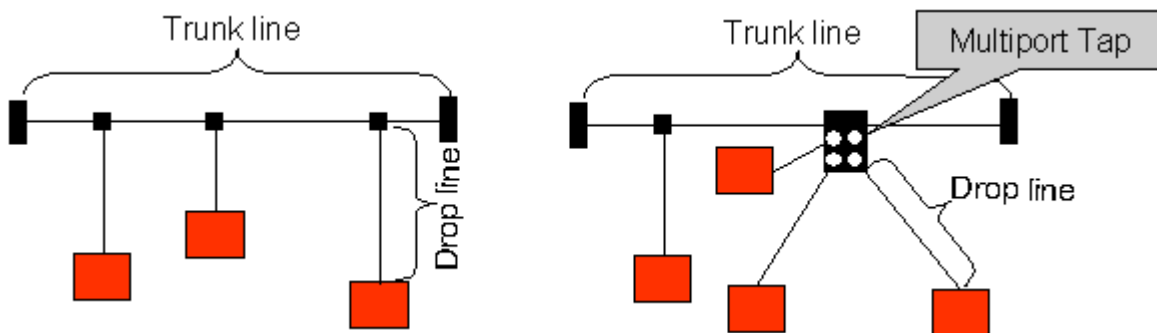


Fig. 53: Sample topology of drop lines

#### 4.3.10.2.4 Star Hub (Multiport Tap)

Shorter drop line lengths must be maintained when passive distributors ("multiport taps"), such as the Beckhoff ZS5052-4500 Distributor Box. The following table indicates the maximum drop line lengths and the maximum length of the trunk line (without the drop lines):

Baud rate	Drop line length with multiport topology	Trunk line length (without drop lines)
1 Mbit/s	< 0,3 m	< 25 m
500 kbit/s	< 1,2 m	< 66 m
250 kbit/s	< 2,4 m	< 120 m
125 kbit/s	< 4,8 m	< 310 m

#### 4.3.10.2.5 CAN cable

Screened twisted-pair cables (2x2) with a characteristic impedance of between 108 and 132 Ohm is recommended for the CAN wiring. If the CAN transceiver's reference potential (CAN ground) is not to be connected, the second pair of conductors can be omitted. (This is only recommended for networks of small physical size with a common power supply for all the participating devices).

##### ZB5100 CAN Cable

A high quality CAN cable with the following properties is included in Beckhoff's range:

- 2 x 2 x 0.25 mm<sup>2</sup> (AWG 24) twisted pairs, cable colors: red/black + white/black
- double screened
- braided screen with filler strand (can be attached directly to pin 3 of the 5-pin connection terminal)
- flexible (minimum bending radius 35 mm when bent once, 70 mm for repeated bending)
- characteristic impedance (60 kHz): 120 ohm
- conductor resistance < 80 Ohm/km
- sheath: grey PVC, outside diameter 7.3 +/- 0.4 mm
- Weight: 64 kg/km.
- printed with "Beckhoff ZB5100 CAN-BUS 2x2x0.25" and meter marking (length data every 20cm)

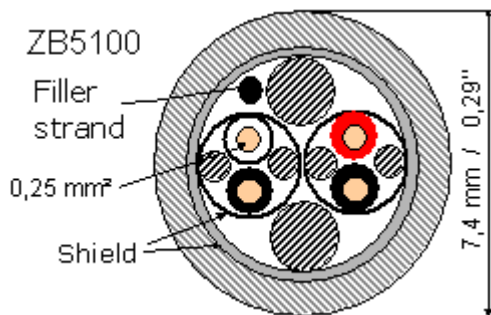


Fig. 54: Structure of CAN cable ZB5100

##### ZB5200 CAN/DeviceNet Cable

The ZB5200 cable material corresponds to the DeviceNet specification, and is also suitable for CANopen systems. The ready-made ZK1052-xxxx-xxxx bus cables for the Fieldbus Box modules are made from this cable material. It has the following specification:

- 2 x 2 x 0.34 mm<sup>2</sup> (AWG 22) twisted pairs
- double screened, braided screen with filler strand
- characteristic impedance (1 MHz): 126 ohm
- Conductor resistance 54 Ohm/km
- sheath: grey PVC, outside diameter 7.3 mm
- printed with "InterlinkBT DeviceNet Type 572" as well as UL and CSA ratings
- stranded wire colors correspond to the DeviceNet specification
- UL recognized AWM Type 2476 rating
- CSA AWM I/II A/B 80°C 300V FT1
- corresponds to the DeviceNet "Thin Cable" specification

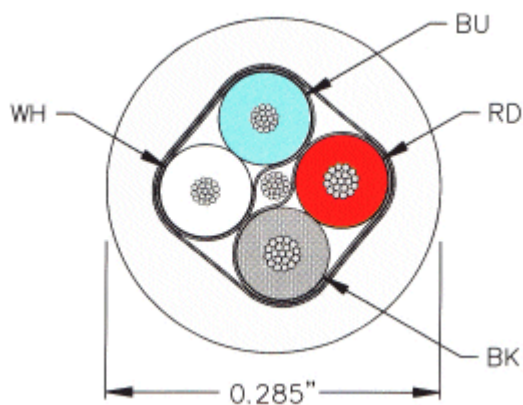


Fig. 55: Structure of CAN/DeviceNet cable ZB5200

#### 4.3.10.2.6 Shielding

The screen is to be connected over the entire length of the bus cable, and only galvanically grounded at one point, in order to avoid ground loops.

The design of the screening, in which HF interference is diverted through R/C elements to the mounting rail assumes that the rail is appropriately earthed and free from interference. If this is not the case, it is possible that HF interference will be transmitted from the mounting rail to the screen of the bus cable. In that case the screen should not be attached to the couplers - it should nevertheless still be fully connected through.

Notes related to checking the CAN wiring can be found in the [Trouble Shooting](#) [► 66] section.

#### 4.3.10.2.7 Cable colors

Suggested method of using the Beckhoff CAN cable on Bus Terminal and Fieldbus Box:

BK51x0 pin PIN BX5100 (X510)	Pin BK5151 CX8050, CX8051, CXxxxx-B510/M510	Fieldbus Box pin	Pin FC51xx	Function	ZB5100 cable color	ZB5200 ca- ble color
1	3	3	3	CAN Ground	black/ (red)	black
2	2	5	2	CAN Low	black	blue
3	5	1	5	Shield	Filler strand	Filler strand
4	7	4	7	CAN high	white	white
5	9	2	9	not used	(red)	(red)



#### 4.3.10.2.8 BK5151, FC51xx, CX with CAN interface and EL6751: D-sub, 9 pin

The CANbus cable is connected to the FC51x1, FC51x2 CANopen cards and in the case of the EL6751 CANopen master/slave terminal via 9-pin Sub-D sockets with the following pin assignment.

Pin	Assignment
2	CAN low (CAN-)
3	CAN ground (internally connected to pin 6)
6	CAN ground (internally connected to pin 3)
7	CAN high (CAN+)

The unlisted pins are not connected.

The mounting rail contact spring and the plug shield are connected together.

Note: an auxiliary voltage of up to 30 V<sub>DC</sub> may be connected to pin 9. Some CAN devices use this to supply the transceiver.

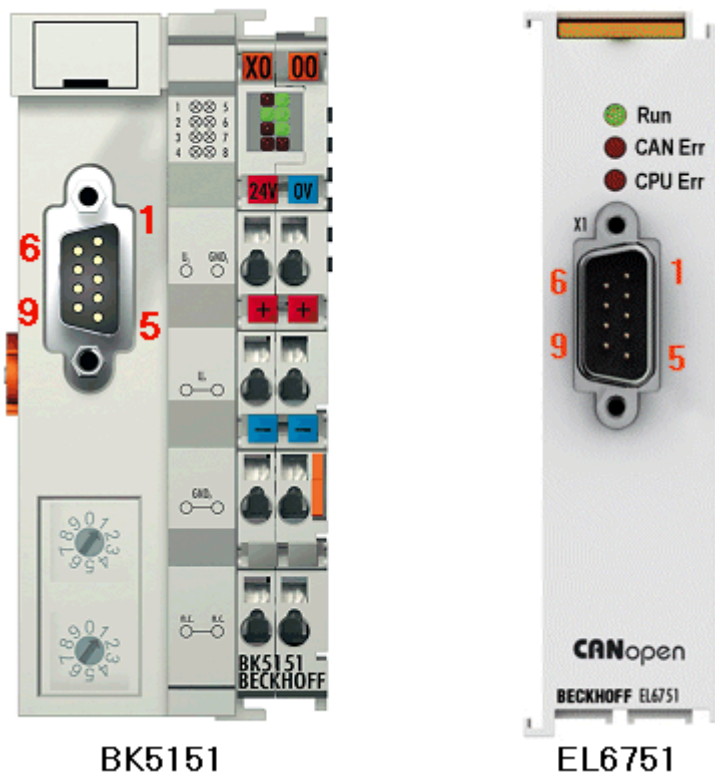


Fig. 56: BK5151, EL6751 pin assignment

#### FC51x2:

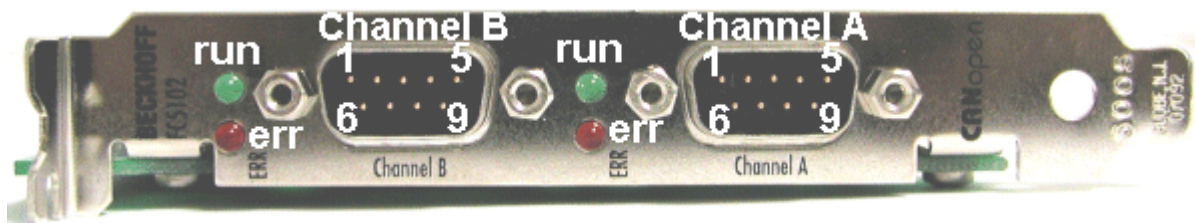


Fig. 57: FC51x2

#### 4.3.10.2.9 BK51x0/BX5100: 5-pin open style connector

The BK51x0/BX5100 (X510) Bus Couplers have a recessed front surface on the left hand side with a five pin connector.

The supplied CANopen socket can be inserted here.

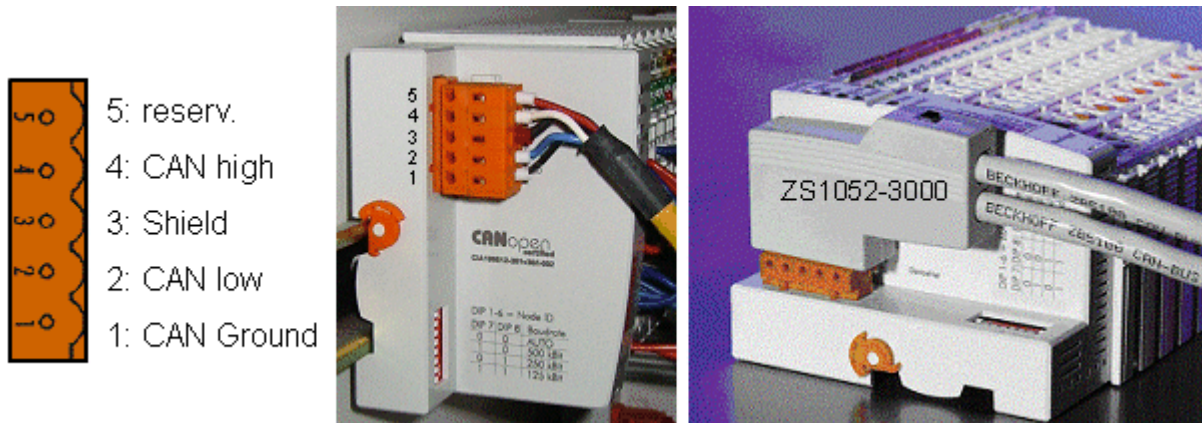


Fig. 58: BK51x0/BX5100 socket assignment

The left figure shows the socket in the BK51x0/BX5100 Bus Coupler. Pin 5 is the connection strip's top most pin. Pin 5 is not used. Pin 4 is the CAN high connection, pin 2 is the CAN low connection, and the screen is connected to pin 3 (which is connected to the mounting rail via an R/C network). CAN-GND can optionally be connected to pin 1. If all the CAN ground pins are connected, this provides a common reference potential for the CAN transceivers in the network. It is recommended that the CAN GND be connected to earth at one location, so that the common CAN reference potential is close to the supply potential. Since the CANopen BK51X0/BX5100 Bus Couplers provide full electrical isolation of the bus connection, it may in appropriate cases be possible to omit wiring up the CAN ground.

#### ZS1052-3000 Bus Interface Connector

The ZS1052-3000 CAN Interface Connector can be used as an alternative to the supplied connector. This makes the wiring significantly easier. There are separate terminals for incoming and outgoing leads and a large area of the screen is connected via the strain relief. The integrated terminating resistor can be switched externally. When it is switched on, the outgoing bus lead is electrically isolated - this allows rapid wiring fault location and guarantees that no more than two resistors are active in the network.

#### 4.3.10.2.10 LC5100: Bus connection via spring-loaded terminals

In the low cost LC5100 Coupler, the CAN wires are connected directly to the contact points 1 (CAN-H, marked with C+) and 5 (CAN-L, marked with C-). The screen can optionally be connected to contact points 4 or 8, which are connected to the mounting rail via an R/C network.

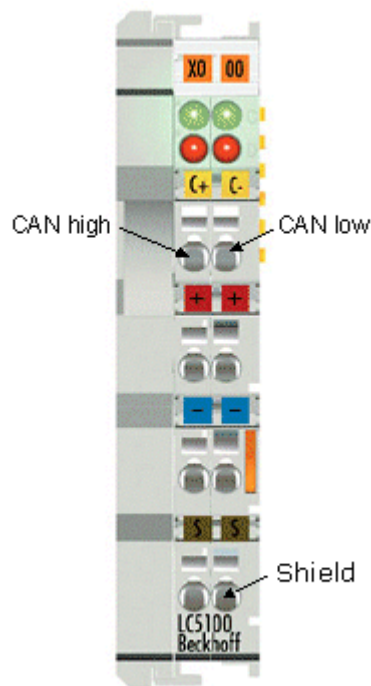


Fig. 59: LC5100



**Attention**

**Risk of device damage!**

On account of the lack of electrical isolation, the CAN driver can be destroyed or damaged due to incorrect cabling. Always carry out the cabling in the switched-off condition. First connect the power supply and then the CAN. Check the cabling and only then switch on the voltage.

**4.3.10.2.11 Fieldbus Box: M12 CAN socket**

The IPxxxx-B510, IL230x-B510 and IL230x-C510 Fieldbus Boxes are connected to the bus using 5-pin M12 plug-in connectors.

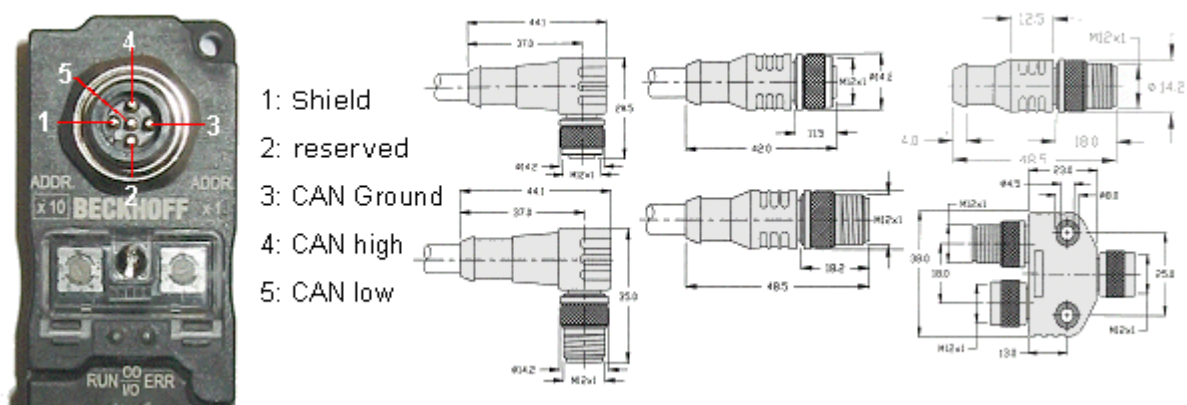


Fig. 60: Pin assignment: M12 plug, fieldbus box

Beckhoff offer plugs for field assembly, passive distributor's, terminating resistors and a wide range of pre-assembled cables for the Fieldbus Box system. Details be found in the catalogue, or under [www.beckhoff.de](http://www.beckhoff.de).

### 4.3.10.3 SSB configuration

The SSB is configured in the system manager. Open your existing configuration, in which you have already configured the PLC project, the K-bus and the higher-level fieldbus, or open a new configuration. Under I/O devices (left mouse button) a further device can now be appended.

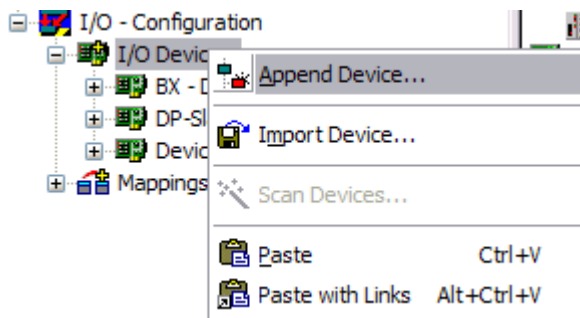


Fig. 61: Adding a further device

Select the CANopen Master SSB and confirm with OK.

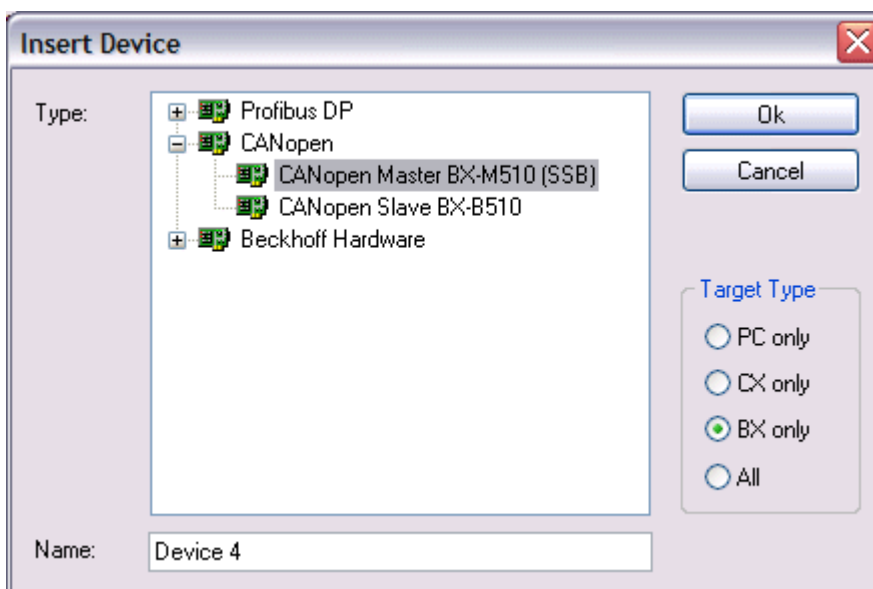


Fig. 62: Selecting the CANopen master SSB

With the left mouse button, a CANopen node can now be selected on the SSB device.

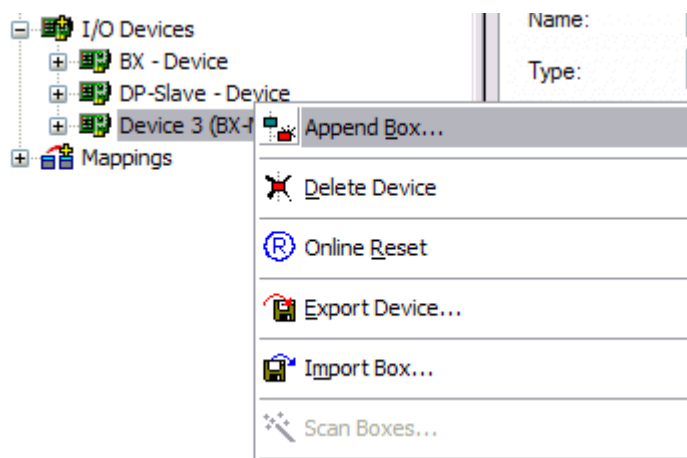


Fig. 63: Adding a CANopen device

All Beckhoff CAN nodes are available, as well as a general CANopen node for CANopen devices from other manufacturers.

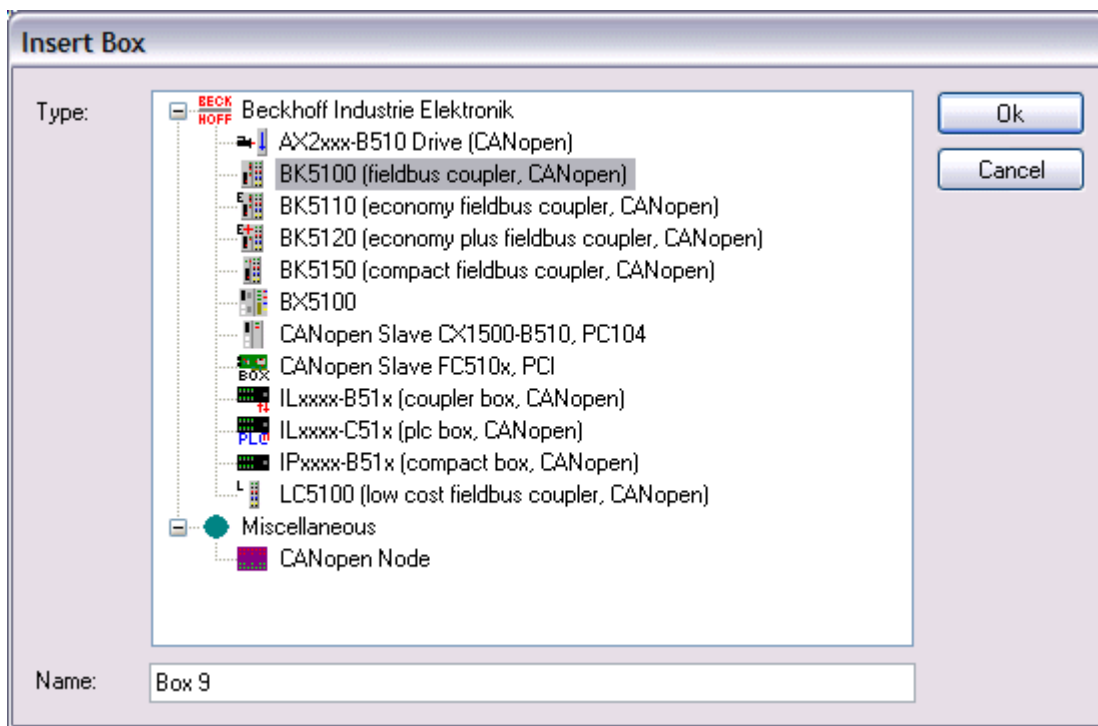


Fig. 64: Selecting a CANopen node

Now link the PLC variables with your CAN node. Once the configuration is complete, load it into the BX.

#### 4.3.10.4 SSB - SDO communication

CANopen SDO communication (Service Data Object) is used to read or write any parameters in the CANopen bus node's object directory. The SSB uses the SDO communication for configuring the communication parameters during start-up.

##### Downloading Application-Specific Parameters when Starting Up

The appropriate parameters are to be entered here in the System Manager for the corresponding node in tab "SDO". The objects that result from the configuration under CAN node appear in square brackets. Any desired number of object directory entries can then be inserted.

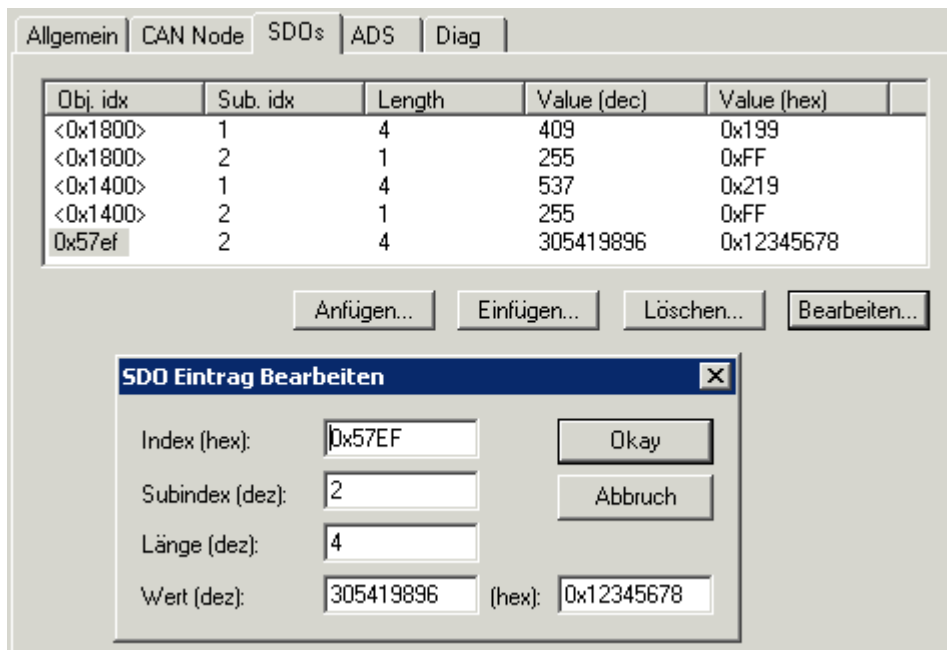


Fig. 65: Adding/editing object directory entries

The SSB expects a positive acknowledgement of the parameter download from the respective bus device. If it was not possible to write a parameter (the bus device has aborted the SDO) the card then attempts to read the corresponding value back and to compare it with the value that was to be written. This is because it could, for instance, be a read-only value, and therefore already correctly configured within the bus device. If they agree with one another, the card moves onto the next parameter entry.

#### 4.3.10.5 SDO communication from the PLC

ADS blocks are used for SDO communication from the PLC. These blocks can be used for sending SDO telegrams and receiving the response of the slave (ADSWRITE/ADSREAD).

Input parameters	Description
NETID	Local NetId of the BX or leave empty, e.g. with "
Port number	0x1000 <sub>hex</sub> + NodeId (slave number)
IDXGRP	SDO Index
IDXOFFS	SDO Subindex
LEN	Length of SDO data (1...4)



Download BX (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207257611.prx>)

#### Setting individual or all nodes to pre-operational or operational state

The ADSWRTCTL function block can be used for setting individual CANopen nodes or all slaves to pre-operational or operational state.

Input parameters	Description
NETID	Local NetId of the BX or leave empty, e.g. with "
Port number	0x1000 <sub>hex</sub> + NodeId (slave number) / 153 <sub>dec</sub> (all nodes)
ADSSTATE	ADSSTATE_RUN
DEVSTATE	1 - Pre / 0 - Operational
LEN	0
SRCADDR	0



Download BX (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207259787.prx>)



## Restarting the SSB interface

The ADSWRTCTL function block can be used to stop and restart the SSB. It should be stopped first before restarting it.

Input parameters	Description
NETID	Local NetId of the BX or leave empty, e.g. with "
Port number	153 <sub>dec</sub>
ADSSTATE	ADSSTATE_STOP, ADSSTATE_RUN
DEVSTATE	0
LEN	0
SRCADDR	0

or

Input parameters	Description (from software version 1.16 for all BX controllers)
NETID	Local NetId of the BX or leave empty, e.g. with "
Port number	300 <sub>dec</sub>
ADSSTATE	ADSSTATE_RESET
DEVSTATE	0
LEN	4
SRCADDR	ADR on a DWORD variable with the ID of the SSB device (the ID can be obtained from the System Manager file and is typically a value between 1 and 3).

### 4.3.10.6 Emergency telegrams and diagnostics

The status of the CAN slave is indicated by NodeState. The DiagFlag is set if an emergency telegram was received. The EmergencyCounter is incremented with each emergency telegram.

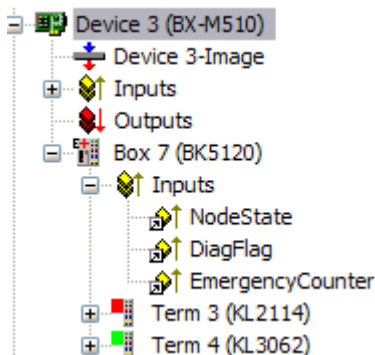


Fig. 66: NodeState, DiagFlag and EmergencyCounter

NodeState value	Description
0	No error
1	Node deactivated
2	Node not found
4	SDO syntax error at Start Up
5	SDO data mismatch at Start Up
8	Node start up in progress
11	SSB Bus off
12	Pre-Operational
13	Severe bus fault
14	Guarding: toggle error
20	TxPDO too short
22	Expected TxPDO is missing
23	Node is Operational but not all TxPDOs were received

**ADS Port 153****Reading of emergency telegrams with AdsRead**

Input parameters	Description
NETID	local NetId of BX
Port number	153
IDXGRP	16#xxxxF180 (xxxx) NodeId, the Diag flag is only reset when at least 106 bytes are read 16#xxxxF181 (xxxx) NodeId, the Diag flag is reset immediately
IDXOFFS	Byte Offset



## Description of the array

Offset	Bit	Value / description
0 - 1	Bit 0	reserved
	Bit 1	Boot up message not received or incorrect
	Bit 2	Emergency-Overflow
	Bit 3 - 15	reserved
2 - 3	Bit 0 - 14	TX-PDO (i+1) received
	Bit 15	All TX PDOs 16-n received
4 - 5	Bit 0 - 4	1: Incorrect TX PDO length
		2: Synchronous TX PDO absent
		3: Node signaling PRE-OPERATIONAL
		4: Event timer timed out for TX PDO
		5: No response and guarding is activated
		6: Toggling missed several times and guarding activated
	Bit 5 - 15	Associated COB ID
6	Bit 0 - 7	1: Incorrect value during SDO upload
		2: Incorrect length during SDO upload
		3: Abort during SDO up/download
		4: Incorrect date during a boot-up message
		5: Timeout while waiting for a boot-up message
7	Bit 0 - 7	2: Incorrect SDO command specifier
		3: SDO toggle bit has not changed
		4: SDO length too great
		5: SDO-Abort
		6: SDO-Timeout
8 - 9	Bit 0 - 7	SDO up/download index
10	Bit 0 - 7	SDO up/download subindex
11	Bit 0 - 7	reserved
12	Bit 0 - 7	errorClass des Aborts
13	Bit 0 - 7	errorCode des Aborts
14 - 15	Bit 0 - 15	Abort additionalCode
16 - 19		Read value (if offset 6 = 1)
20 - 23		Expected value (if offset 6 = 1)
24 - 25		Number of consecutive emergencies
26 - n		Emergencies (8 bytes each)

 Download BX (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207261963.prx>)

 Download sample System Manager file BX (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207264139.tsm>)

## Reading the number of PDO telegrams with AdsRead

Input parameters	Description
NETID	local NetId of BX
Port number	153
IDXGRP	16#xxxxF930 (xxxx) NodeId
IDXOFFS	0

 Download BX (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207266315.prx>)

**Note****Configuration of the node ID required**

The node ID must be configured before the ADS function blocks is called in the TwinCAT configuration.

**Sending a CAN message**

This ADSWRITE command enables any CAN message to be sent.

Input parameters	Description
NETID	local NetId of BX
Port Nummer	153
IDXGRP	16#0000F921
IDXOFFS	0
LEN	11 bytes
SRCADDR	Pointer to an 11 byte ARRAY

**Structure of the 11 byte CAN data**

Byte	Description	Example Node 7 SDO 0x607 Len 8 Download Request 0x2100 (Index) Sub Index 1 - Value "1"
1	COB-ID LowByte	0x06 (SDO Low Byte)
2	COB-ID HighByte	0x07 (SDO High Byte)
3	LEN (length)	0x08 (LEN, may be 5 in this case)
4	Data[1]	0x22 (Download Request)
5	Data[2]	0x00 (Index Low Byte)
6	Data[3]	0x21 (Index High Byte)
7	Data[4]	0x01 (Sub Index)
8	Data[5]	0x01 (Value "1")
9	Data[6]	0x00
10	Data[7]	0x00
11	Data[8]	0x00

**4.3.10.7 CANopen Trouble Shooting****Error Frames**

One sign of errors in the CAN wiring, the address assignment or the setting of the baud rate is an increased number of error frames: the diagnostic LEDs then show *Warning Limit exceeded* or *Bus-off state entered*.

**Note****Error Frames**

Warning limit exceeded, passive error or bus-off state are indicated first of all at those nodes that have detected the most errors. These nodes are not necessarily the cause for the occurrence of error frames!

If, for instance, one node contributes unusually heavily to the bus traffic (e.g. because it is the only one with analog inputs, the data for which triggers event-driven PDOs at a high rate), then the probability of its telegrams being damaged increases. Its error counter will, correspondingly, be the first to reach a critical level.

**Node ID / Setting the Baud Rate**

Care must be taken to ensure that node addresses are not assigned twice: there may only be one sender for each CAN data telegram.

### Test 1

Check node addresses. If the CAN communication functions at least some of the time, and if all the devices support the boot up message, then the address assignment can also be examined by recording the boot up messages after the devices are switched on. This will not, however, recognize node addresses that have been swapped.

### Test 2

Check that the same baud rate has been set everywhere. For special devices, if the bit timing parameters are accessible, do they agree with the CANopen definitions (sampling time, SJW, oscillator).

### Testing the CAN wiring

These tests should not be carried out if the network is active: No communication should take place during the tests. The following tests should be carried out in the stated sequence, because some of the tests assume that the previous test was successful. Not all the tests are generally necessary.

#### Network terminator and signal leads

The nodes should be switched off or the CAN cable unplugged for this test, because the results of the measurements can otherwise be distorted by the active CAN transceiver.

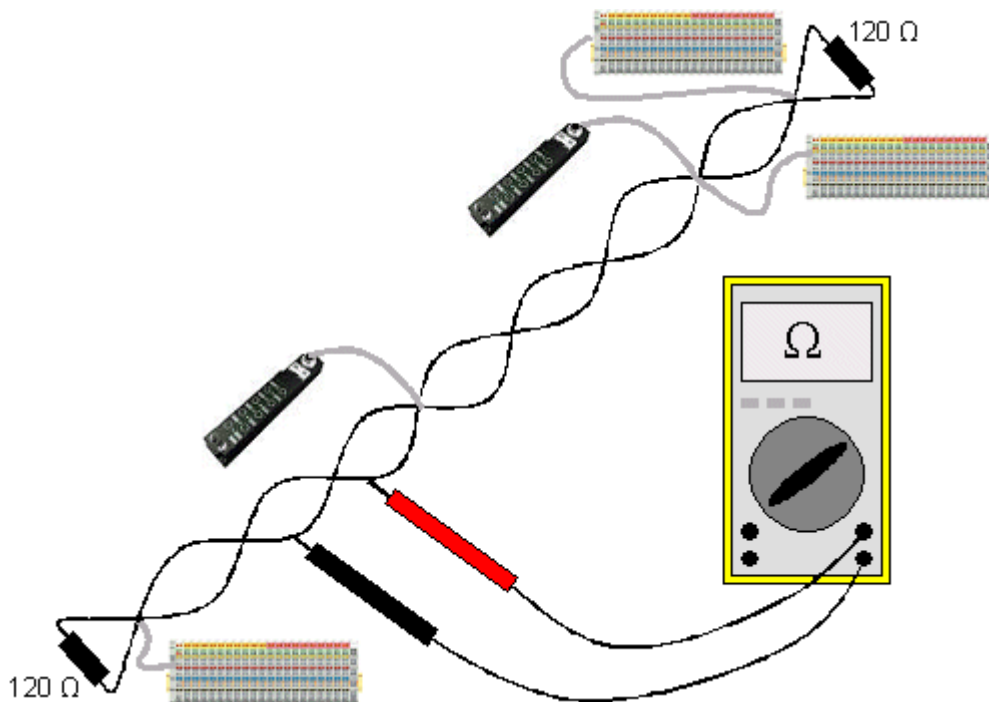


Fig. 67: Wiring diagram for test setup

### Test 3

Determine the resistance between CAN high and CAN low - at each device, if necessary.

If the measured value is greater than 65 Ohms, it indicates the absence of a terminating resistor or a break in a signal lead. If the measured value is less than 50 Ohms, look for a short circuit between the CAN lines, more than the correct number of terminating resistors, or faulty transceivers.

### Test 4

Check for a short circuit between the CAN ground and the signal leads, or between the screen and signal leads.

**Test 5**

Remove the earth connection from the CAN ground and screen. Check for a short circuit between the CAN ground and screen.

**Topology**

The possible cable length in CAN networks depends heavily on the selected baud rate. CAN will tolerate short drop lines - although this again depends on the baud rate. The maximum permitted drop line length should not be exceeded. The length of cable that has been installed is often underestimated - estimates can even be a factor of 10 less than the actual length. The following test is therefore recommended:

**Test 6**

Measure the lengths of the drop lines and the total bus lengths (do not just make rough estimates!) and compare them with the topology rules for the relevant baud rate.

**Screening and earthing**

The power supply and the screen should be carefully earthed at the power supply unit, once only and with low resistance. At all connecting points, branches and so forth the screen of the CAN cable (and possibly the CAN GND) must also be connected, as well as the signal leads. In the Beckhoff IP20 Bus Couplers, the screen is grounded for high frequencies via an R/C element.

**Test 7**

Use a DC ammeter (16 amp max.) to measure the current between the power supply ground and the shield at the end of the network most remote from the power supply unit. An equalization current should be present. If there is no current, then either the screen is not connected all the way through, or the power supply unit is not properly earthed. If the power supply unit is somewhere in the middle of the network, the measurement should be performed at both ends. When appropriate, this test can also be carried out at the ends of the drop line.

**Test 8**

Interrupt the screen at a number of locations and measure the connection current. If current is flowing, the screen is earthed at more than one place, creating a ground loop.

**Potential differences**

The screen must be connected all the way through for this test, and must not be carrying any current - this has previously been tested.

**Test 9**

Measure and record the voltage between the screen and the power supply ground at each node. The maximum potential difference between any two devices should be less than 5 volts.

**Detect and localize faults**

The "low-tech approach" usually works best: disconnect parts of the network, and observe when the fault disappears.

However, this does not work well for problems such as excessive potential differences, ground loops, EMC or signal distortion, since the reduction in the size of the network often solves the problem without the "missing" piece being the cause. The bus load also changes as the network is reduced in size, which can mean that external interference "hits" CAN telegrams less often.

Diagnosis with an oscilloscope is not usually successful: even when they are in good condition, CAN signals can look really chaotic. It may be possible to trigger on error frames using a storage oscilloscope - this type of diagnosis, however, is only possible for expert technicians.

## Protocol problems

In rare cases, protocol problems (e.g. faulty or incomplete CANopen implementation, unfavorable timing at boot up, etc.) can be the cause of faults. In this case it is necessary to trace the bus traffic for evaluation by a CANopen experts - the Beckhoff support team can help here.

A free channel on a Beckhoff FC5102 CANopen PCI card is appropriate for such a trace - Beckhoff make the necessary trace software available on the internet. Alternatively, it is of course possible to use a normal commercial CAN analysis tool.

Protocol problems can be avoided if devices that have not been conformance tested are not used. The official CANopen Conformance Test (and the appropriate certificate) can be obtained from the CAN in Automation Association (<http://www.can-cia.de>).

## 4.3.10.8 Examples

### 4.3.10.8.1 BK5120 at SSB

Required material:

- TwinCAT 2.9 build 953 or higher
- BX3100 version 0.80 or higher, BX5100 version 0.13, BX8000 version 0.04
- 1 x KL1xx4
- 1 x KL2xx4
- 1 x KL9010
- 1 x BK5120
- 1 x KL1xx4
- 1 x KL2xx4
- 1 x KL9010
- Cabling material and power supply
- TwinCAT System Manager file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207268491.tsm>)



(The system manager file has to be transferred to the BX controller via ADS).

- BX program file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207270667.prx>)



For the configuration download via ADS, either a BECKHOFF fieldbus master or a free serial port is required.

### 4.3.10.8.2 Communication between BX controllers (via SSB)

2 or more BX controllers can exchange data via the SSB. Use 2 telegrams for configuring this data exchange in the System Manager (CAN layer).

CAN telegram communication is specified via the COB ID. The BX type is irrelevant, since the SSB is present on each BX controller, and the behavior and configuration is identical.



Fig. 68: Communication between BX controllers (via SSB)

**Example configuration**

BX\_ONE:

Node Id 2

CAN\_Out AT %QB100: ARRAY[0..7] OF BYTE

COD Id 514 0x202

CAN\_In AT %IB100: ARRAY[0..7] OF BYTE

COD Id 386 0x182

BX\_TWO:

Node Id 2

CAN\_Out AT %QB100: ARRAY[0..7] OF

BYTE

COD Id 386 0x182

CAN\_In AT %IB100: ARRAY[0..7] OF BYTE

COD Id 514 0x202

Configuration and program example:

**Required material**

- TwinCAT 2.9 build 959 or higher
- 2 x BXxx00
- Cabling material and power supply
- TwinCAT System Manager file BX\_ONE (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207272843.tsm>)



- Program file BX\_ONE (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207275019.prx>)



- TwinCAT System Manager file BX\_TWO (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207277195.tsm>)



- Program file BX\_TWO (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207279371.prx>)



For the configuration download via ADS, either a BECKHOFF master (FC310x, FC510x, FC520x) or a free serial port is required.

#### 4.3.10.8.3 AX2000 at SSB



Fig. 69: AX2000

Required material:

- TwinCAT 2.9 build 953 or higher
- BX3100 version 0.80 or higher
- 1 x KL1xx4
- 1 x KL2xx4
- 1 x KL9010
- 1 x AX2000 with the following settings: Slave address 4, baud rate 500 kbyte
- Cabling material and power supply
- Example program and configuration on the BX Controller
  - TwinCAT System Manager file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207281547.tsm>)



(The System Manager file has to be transferred to the BX Controller via ADS).

- BX program file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207283723.prx>)



For the configuration download via ADS, either a BECKHOFF fieldbus master or a free serial port is required.

#### AX2000 description

The following sections are extracts from the AX2000 drive manual. Further information can be found on the internet at <http://www.beckhoff.de>.

## Hardware and interfaces

### Setting the Station Address

The station address (device address at the CAN bus) of the servo drive can be set in three ways:

- Via the front panel keyboard (see AX2000 installation guide)
- Via the "Basic settings" screen of the DRIVE.EXE commissioning software
- Via the serial interface with the following ASCII command sequence:

ADDR nn > SAVE > COLDSTART (with nn = address)

The address range can be extended from 1..63 to 1..127 with the ASCII object MDRV.

### Setting the baud rate

The CAN transfer speed (baud rate) can be set in three ways:

- Via the front panel keyboard (see AX2000 installation guide)
- Via the "Basic settings" screen of the DRIVE.EXE commissioning software
- Via the serial interface with the following ASCII command sequence:  
CBAUD bb > SAVE > COLDSTART (with bb = baud rate in kB)

Possible baud rates are 10, 20, 50, 100, 125, 250, 333, 500 (default), 666, 800 or 1000 kBaud.

### CANopen Interface (X6)

Interface for connection to the CAN bus (default 500 kBaud). The integrated profile is based on the DS301 CANopen communication profile and on the DSP402 drive profile. The following functions are available in combination with the position controller:

jogging with variable speed, reference motion, start travel command, start direct travel command, digital set value specification, data transfer functions and many others.

Detailed information can be found in the CANopen manual. The interface is electrically isolated via an optocoupler and has the same potential as the RS232 interface. The analog set value inputs can still be used. The two interfaces (RS232 and CAN) occupying the same connector (X6) can be split to two connectors via the optional 2 CAN extension card.

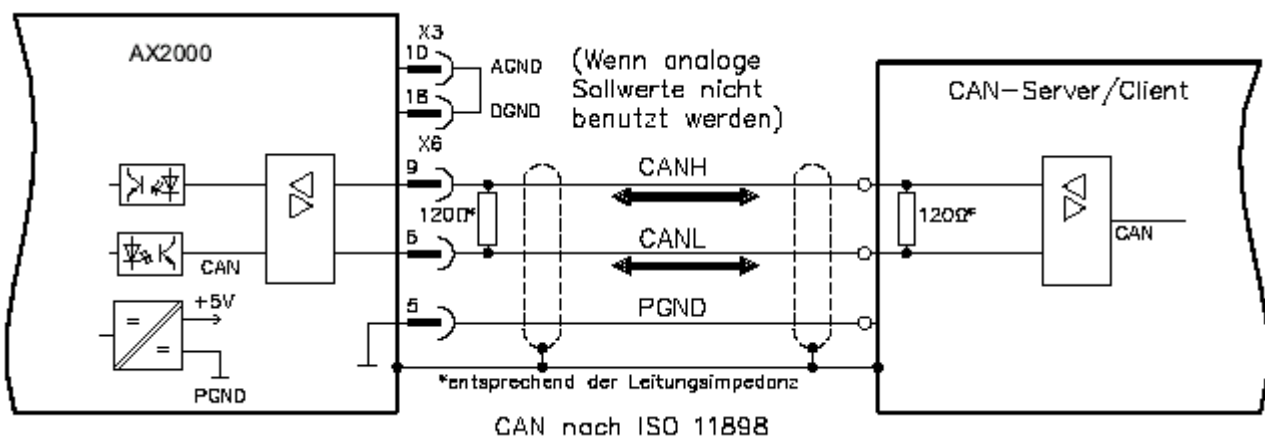


Fig. 70: CANopen Interface (X6)



#### 4.3.10.8.4 Cimrex panel at the SSB of the BX controller

The CAN interface of the BX controller can also be used for connecting an operating panel. In this example, a panel from the company Beijers is connected. Further information on the panel can be found under <http://www.beijerelectronics.de>.



Fig. 71: Cimrex panel at the SSB of the BX controller

#### Necessary components

- 1 x BX3100
- Some Bus Terminals for the K-bus (here 3 x KL2114, can be adjusted in the System Manager file)
- 1 x Cimrex 41
- 1 x CAB 15 CAN adapter
- BX sample program in ST: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207285899.prx>)



- BX example configuration: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207288075.tsm>)

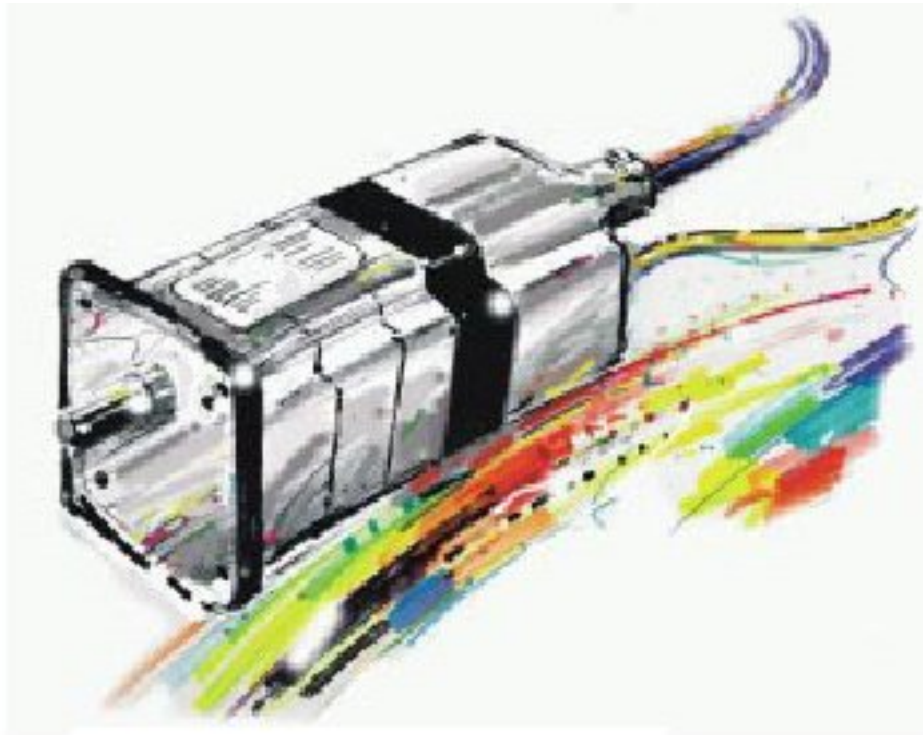


- Example for Cimrex 41: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/cpa/3207290251.cpa>)



- Baud rate 500 kbaud
- CAN slave address 10

#### 4.3.10.8.5 IclA drive at SSB



# *IclA*®

Fig. 72: IclA drive at SSB

Required material:

- TwinCAT 2.9 build 953 or higher
- BX3100 version 0.80 or higher
- 1 x KL1xx4
- 1 x KL2xx4
- 1 x KL9010
- 1 x IclA D065 with the following settings: slave address 10, baud rate 500 kbyte (Please note: These are not the default parameters of the drive)
- Cabling material and power supply

For the configuration download via ADS, either a BECKHOFF fieldbus master or a free serial port is required.

#### Reconfiguration example for TwinCAT with FC510x CANopen master card

An example for converting a drive is listed below.

- TwinCAT System Manager file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/wsm/3207292427.wsm>)



- TwinCAT PLC file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/pro/3207294603.pro>)



### Example program and configuration on the BX Controller

- TwinCAT System Manager file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207296779.tsm>)



(The System Manager file has to be transferred to the BX Controller via ADS).

- BX program file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207298955.prx>)



### IcIA D065 description

The following sections are extracts from the IcIA drive manual. They were provided by the company SIG Positec Automation GmbH for the purpose of describing the basic parameters. Further information can be found on the internet at <http://www.sig-positec.de>.

### Hardware and interfaces

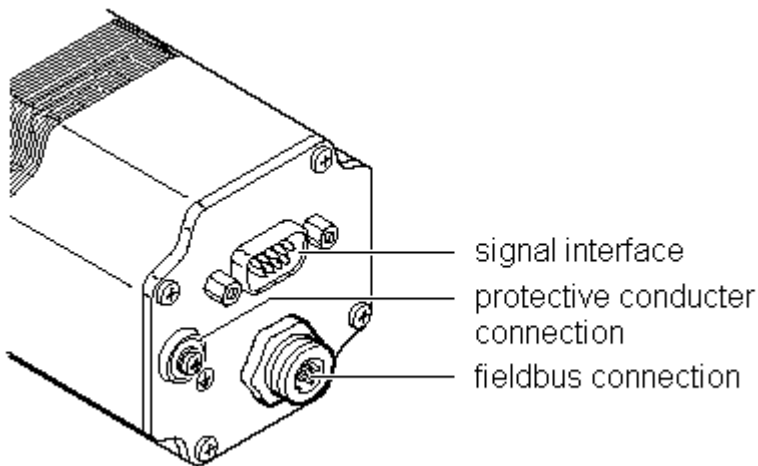


Fig. 73: IcIA drive connections

- Signal interface for
  - Supply voltage
  - Control signals for manual mode
  - Connection for emergency stop signal
- Protective conductor connection for earthing via PE bus bar
- Fieldbus connection for connecting the fieldbus cable.

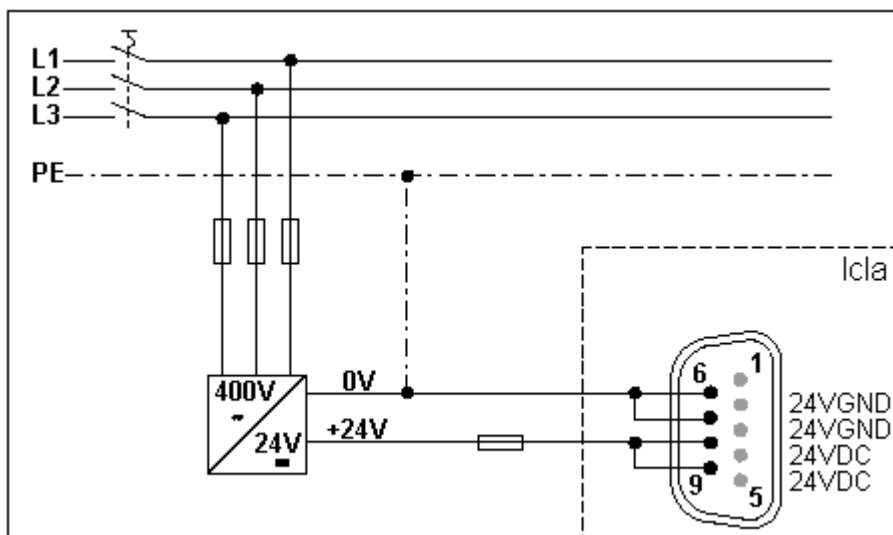


Fig. 74: Signal interface

If the emergency stop function is not required, connect pin 2 with pin 8 or 9 (24 V<sub>DC</sub>).

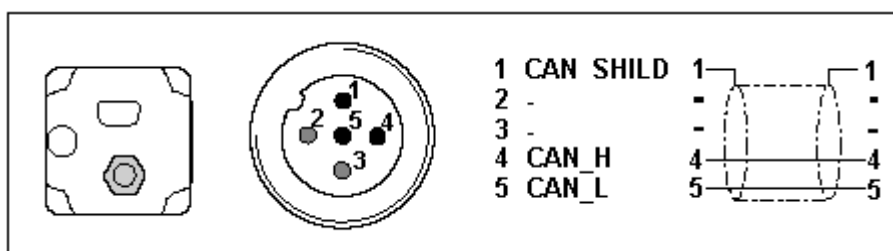


Fig. 75: Fieldbus connection

### Control word 0x6040

The object represents the control word for the device. The control word is used for several control tasks:

- Changeover between different operating states. The possible states and transitions can be found under the index keyword "machine state". Bits 0 to 3 and bit 7 are relevant for a change of state.
- Starting and stopping mode-specific functions, e.g. starting a travel command via bit 4. Bits 4 to 6 are used for mode-specific settings. Further details can be found under the keywords "Operating mode, starting", "Operating mode, monitoring" and in the description of the respective operating modes in sections "Manual mode" and "Positioning mode".
- Stopping of the positioning drive from an active travel operation. Bit 8 "Stop" is used for stopping. Further details can be found under the keywords "Operating mode, starting" and "Operating mode, monitoring".

Object description	Value description
Index	6040h
Object name	Control word
Data type	Integred16
Subindex	00h, control word
Access	read-write
PDO-Mapping	R_PDO

Bit	Name	Meaning
11..15	Manufacturer specific	not used
9, 10	-	reserved
8	Stop	Stop motor
7	Reset fault	Reset fault
4..6	-	Operating mode dependent,
3	Enable operation	Execute operating mode
2	Quick stop (low active)	Breaking with quick stop ramp
1	Disable voltage (low active)	Switch off voltage
0	Switch on	Switch into ready-to-run state

### Status word 0x6041

The object describes the current operating state of the device. The status word can be used for the following monitoring functions:

- Checking the operating state of the positioning controller. Bits 0 to 3, 5 and 6 are relevant.
- Bit 4 indicates whether the output stage is ready for processing a transport instruction.
- Bits 7 to 15 are used for monitoring the travel operation and for status monitoring of device-specific states.

Further details for monitoring travel operation can be found under the keywords "Operating mode, starting", "Operating mode, monitoring" and in the description of the respective operating modes in sections "Manual mode" and "Positioning mode". The bits for device status monitoring are described in section "Diagnostics and trouble shooting".

The control word is represented in the first two bytes of the R\_PDOs.

Object description	Value description
Index	6041h
Object name	Status word
Data type	Unsigned16
Subindex	00h, status word
Access	read-only
PDO Mapping	T_PDO

Bit	Name	Meaning
15	Out of security area	Out of security area 0->1: Limit switch position S0 or S1 exceeded
14	Out of drive area	Out of drive area 0->1: Limit switch position D0 or D1 exceeded
12..13	-	Operating mode-dependent meaning
11	Internal limit active	Out of working area
10	Target reached	Target reached 1->0: New target position transferred 0->1: Requested target position reached or motor standstill after stop request
9	Remote	0: manual mode 1: no manual mode
8	Right out of drive area	Only valid if bit 11 = 1 - 0: Limit switch position W1 exceeded - 1: Limit switch position W0 exceeded
7	Warning	Warning
6	Switch on disabled	not ready for operation
5	Quick Stop	Quick stop active
4	Voltage disabled	Voltage off
3	Fault	Fault occurred
2	Operation enabled	Operating mode enabled
1	Switched on	Ready for operation
0	Ready to switch on	Ready to switch on

### Reference ranges

A valid referencing is defined via three limit switch zones, which have to be within the possible traversing range of the drive. The limit switches protect the drive and the system from damage.

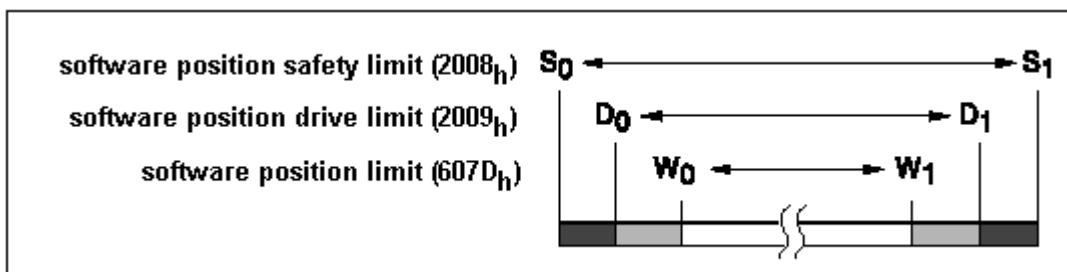


Fig. 76: Reference ranges

- Working area W0 - W1 for positioning mode.
- Drive area D0 - D1. From ranges D0 - W0 and D1 - W1, the drive can only be moved backwards towards the operating range.
- Security area S0 - S1. From areas S0 - D0 and S1 - D1, the drive can only be moved backwards manually.
- CANopen objects - three CANopen objects are used for setting up the limit switches. They contain the position values for the upper and lower range limits.
- Working area limits in software position limit (607D<sub>hex</sub>)
- Drive area limits in software position drive limit (2009<sub>hex</sub>)
- Security area limits in software position safety limit (2008<sub>hex</sub>)

### Referencing example

The following listing demonstrates the input of the referencing values. The node address of the positioning drive is set to 01<sub>hex</sub>.

COB-ID	Data	Meaning
601	2F 60 60 00 06	R_SDO: switch to homing mode
581	60 60 60 00 xx	T_SDO: OK
601	23 08 20 02 0C 7B 00 00	R_SDO: max. value safety range S <sub>1</sub> : 7B0Ch
581	60 08 20 02 xx xx xx xx	T_SDO: OK
601	23 08 20 01 00 00 00 00	R_SDO: min. value safety range S <sub>0</sub> : 0000h
581	60 08 20 01 xx xx xx xx	T_SDO: OK
601	23 09 20 02 42 72 00 00	R_SDO: max. value driving range D <sub>1</sub> : 7242h
581	60 09 20 02 xx xx xx xx	T_SDO: OK
601	23 09 20 01 CA 08 00 00	R_SDO: min. value driving range D <sub>0</sub> : 8CAh
581	60 09 20 01 xx xx xx xx	T_SDO: OK
601	23 7D 60 02 AE 60 00 00	R_SDO: max. value working range W <sub>1</sub> : 60AEh
581	60 7D 60 02 xx xx xx xx	T_SDO: OK
601	23 7D 60 01 5E 1A 00 00	R_SDO: min. value working range W <sub>0</sub> : 1A5Eh
581	60 7D 60 01 xx xx xx xx	T_SDO: OK
601	23 10 10 03 73 61 76 65	R_SDO: save application parameter: "save"
581	60 10 10 03 xx xx xx xx	T_SDO: OK
601	2F 98 60 00 FF	R_SDO: setting the reference type
581	60 98 60 00 xx	T_SDO: OK
601	23 0B 20 00 BC 34 00 00	R_SDO: dimension setting, current position to S <sub>0</sub> : 34BCh
581	60 0B 20 00 xx xx xx xx	T_SDO: OK
601	2B 40 60 00 1F 00	R_SDO: homing operation start (rising edge, bit 4)
581	60 40 60 00 xx xx	T_SDO: OK

Fig. 77: Listing of the referencing values

#### 4.3.10.8.6 Lenze frequency converter at SSB

# Lenze



Fig. 78: Frequency converter from Lenze

##### Required material

- TwinCAT 2.9 build 953 or higher
- BXxx00
- 1 x KL1xx4
- 1 x KL2xx4
- 1 x KL9010
- 1 x Lenze 8200 vector + motor
- 1 x Lenze CANopen Interface 2175
- Cabling material and power supply

For the configuration download via ADS you need a BECKHOFF fieldbus master card or a free serial port.

##### Lenze description

The following sections are extracts from the Lenze 2175 manual. They were provided by the company Lenze Drive Systems GmbH for the purpose of describing the basic parameters. Further information can be found on the internet at <http://www.Lenze.com>.

##### Initial commissioning

Set the power supply for the bus module to internal power supply.



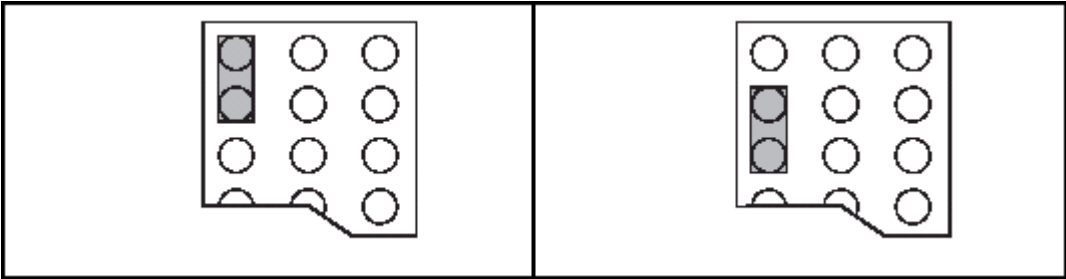


Fig. 79: External power supply - internal power supply  
(State at Delivery)

For CANopen communication set DIP switch 10 to "ON".

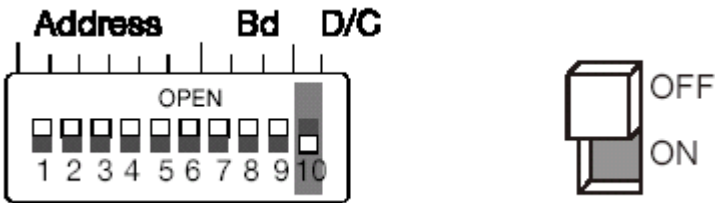


Fig. 80: DIP switch

Baud rate DIP switches 7-9

Transfer rate [kbps]	S7	S8	S9
10	ON	ON	OFF
20	ON	OFF	ON
50	OFF	ON	ON
125	OFF	ON	OFF
250	OFF	OFF	ON
500 (default)	OFF	OFF	OFF
1000	ON	OFF	OFF



Note

Priority of the DIP switches

DIP switch 6 has the smallest weighting.  
Example: Address 3 switches 5 and 6 "ON".

Enabling the communication module

Switch to operating mode 3 for enabling the communication module. This can be achieved via the SSB using the following entry:

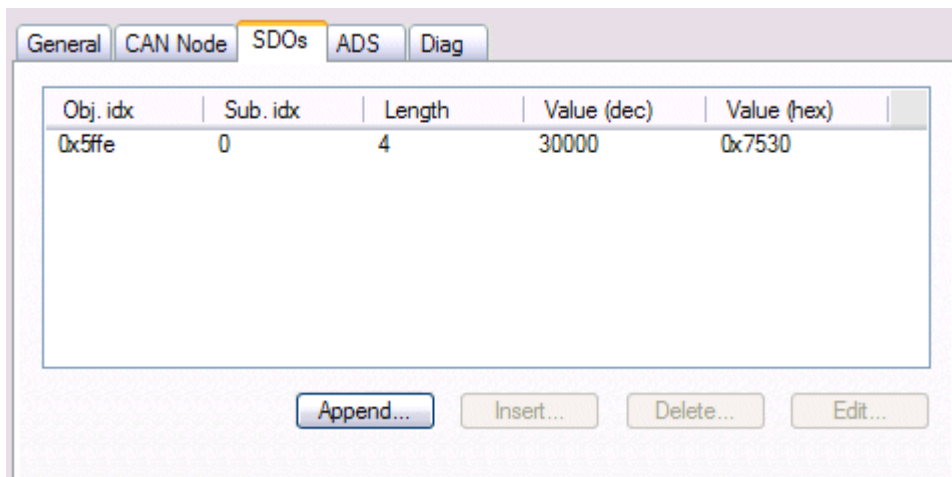
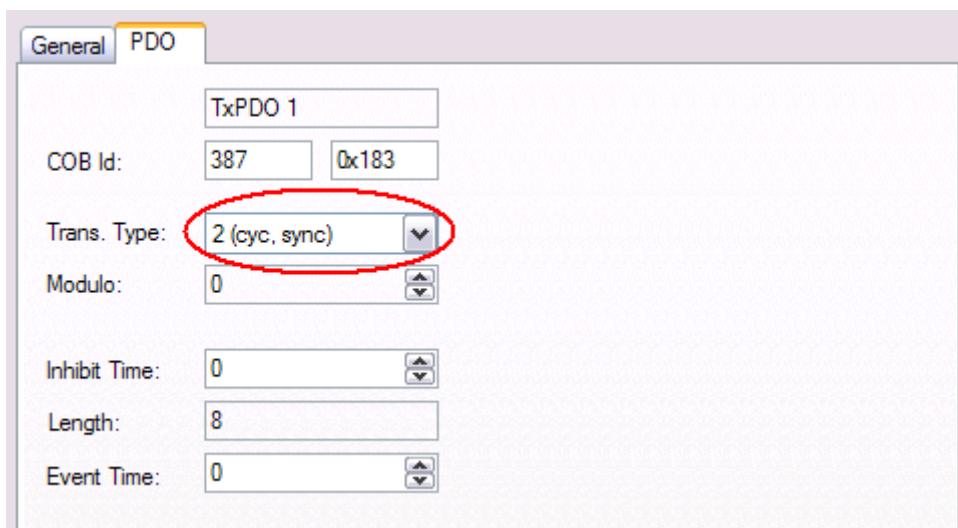


Fig. 81: Enabling the communication module

### Sync telegram

In the default setting, the Lenze drive will send its output PDOs only once it has received a sync telegram from the CAN master. If you set the trans. type to 2, for example, the Lenze drive will send an output PDO after every second sync telegram it receives.



### Sample project

- TwinCAT-System-Manager-File: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207301131.tsm>)



- TwinCAT-PLC-File: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207303307.prx>)



### 4.3.11 Real-Time Clock (RTC)

A real-time-clock (RTC) with backup battery is implemented on the BX controller. The clock has a battery.

#### Setting the real-time clock

The simplest way of setting the clock is via the System Manager. When the ADS communication is operating normally, the current time is displayed on the BX controller. To adjust the time, simply edit the time, and adjust the day, month and year with the drop down key. For setting the year, click on the year display and specify the required year. Repeat the procedure for the month. Once all parameters have been set, click on *Update RTC on BX*.

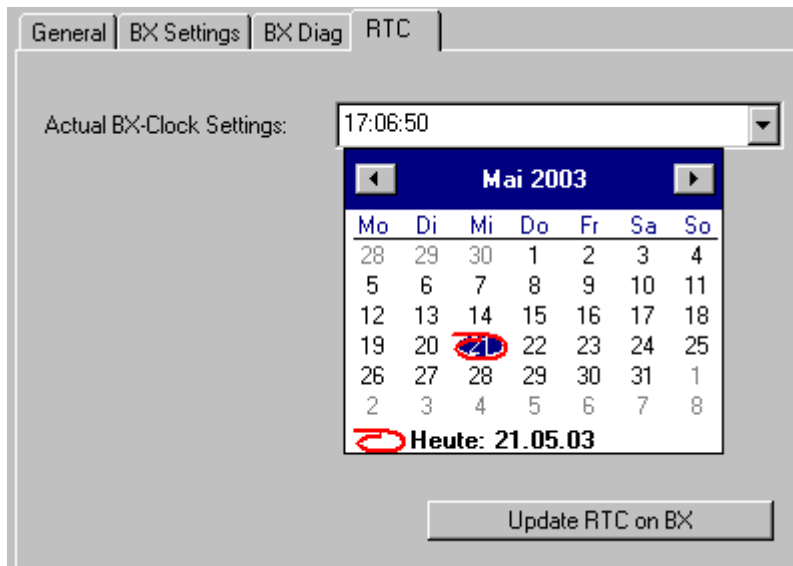


Fig. 82: Setting the real-time clock (RTC)



#### Note

#### Service life of the battery

The service life of the battery may vary depending on utilization.

#### Reading the RTC on the BX controller (see example [▶ 121] Programming\Library)

The RTC can be read via a function block. Required libraries:

- TcSystemBX.lbx
- TCBaseBX.lbx

#### Writing the RTC to the BX controller

The RTC can be set via a function block. Required libraries:

- TcSystemBX.lbx
- TCBaseBX.lbx

**Reading the RTC via ADS**

Description	Meaning	Value
NETID	Target device	see System Manager "ADS"
Port	ADS port number	150 <sub>dec</sub>
IDXGRP	IDX group	0x0000_F100 <sub>hex</sub>
IDXOFFS	IDX Offset	0x0000_0000 <sub>hex</sub>
Length	Length of the data	16 byte
Variable type	Type of variable	TIMESTRUCT

**Writing of the RTC via ADS**

Description	Meaning	Value
NETID	Target device	see System Manager "ADS"
Port	ADS port number	150 <sub>dec</sub>
IDXGRP	IDX group	0x0000_F100 <sub>hex</sub>
IDXOFFS	IDX Offset	0x0000_0000 <sub>hex</sub>
Length	Length of the data	16 byte
Variable type	Type of variable	TIMESTRUCT

**Setting via the navigation switch**

See menu. [► 86]

**Technical data for RTC**

Accuracy: approx. 1 second/day

Duration for which the time is stored: approx. 3 months with fully charged battery

Service life of the battery: approx. 10 years for 10 cycles per day (1000 cycles for complete charge/discharge cycles)

### 4.3.12 COM port

The BX Controller has two serial interfaces. For PIN assignment please refer to [Hardware description \[► 25\]](#).

#### Setting options:

Description	Selection
Baud rate [ <a href="#">► 114</a> ]	9600 baud 19200 baud 38400 baud (starting with auto baud rate detection) 57600 baud 115200 baud (COM 2 only)
Data bits	7 8 (Default)
Parity	NONE ODD EVEN (Default)
Stop bits	1 (default) 2

#### COM 1

The COM 1 interface is used for communication with the KS2000 software or with TwinCAT PLC Control (login via serial interface).

#### COM 2

The COM 2 interface (with RS 232 or RS 485) is used for the application of user protocols or protocol libraries (such as ModbusRTU, RK512, etc.) for the connection of other serial devices.

#### Library

Function blocks are available for communication with the serial interface.

- [Documentation \[► 114\]](#)
- [Example \[► 118\]](#)
- [Library \[► 114\]](#)

## 4.4 Menu

### 4.4.1 BX menu settings

To change into the menu, press the navigation switch for three seconds. The *Menu* directory appears first.

- You can change between the menu settings with the RIGHT/LEFT keys (the menu shown in row 1 is the active menu).
- Press the DOWN key for changing into a submenu.
- Press the UP key to return to the main menu.

Row 1 of the submenu shows the menu item, row 2 the current setting of this menu item.

Some settings cannot be changed (*read only*). These items are only intended to provide checks and to give the user information. To close the menu it is necessary to be in the main menu and then to hold the navigation switch down for three seconds.

Before settings can be changed, a password has to be set. The password remains stored even during a firmware update and through a reset to the factory settings. If you forget the password, the BX controller will have to be sent in.



Fig. 83: Navigation switches of the BX controller

#### Switch assignment

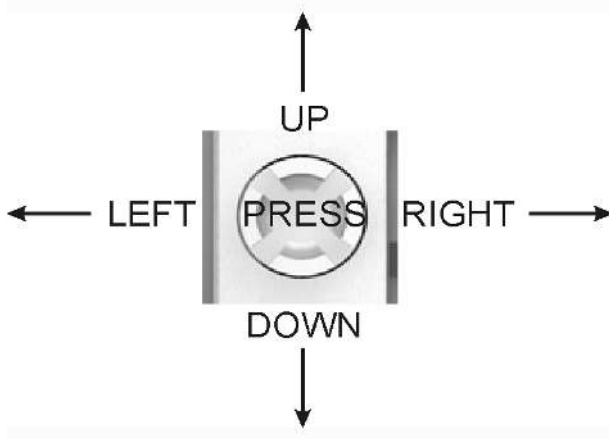


Fig. 84: Switch assignment

Main menu	Submenu 1st row	Submenu 2st row	Read/Write
MENU	Password	**** not set ???? set	See below
	Factory settings	Activate?	Pressing the key causes the factory setting to be reset and the controller to be rebooted automatically
	Reboot	Activate?	Pressing the key causes the controller to be rebooted
AMS	AMS	>AMS Net-ID<	read only
PLC	NAME	>current NAME<	read only
	Curr. Exex. Time	>current value<	[ms]
	Task time	>current value<	The cycle time can be set if the key is pressed
	Status	>Boot-Prj< >PLC Status<	Boot project exists PLC status
Config	NAME	>current NAME<	read only
	Delete config	Activate?	Pressing the right key causes the current configuration to be deleted
Real Time Clock	Date and Time	>current time<	read only
	Year	Setting	2003-2xxx
	Month	Setting	1-12
	Day	Setting	1-31
	Day of week	Setting	Mon, ... Fri
	Hour	Setting	0-23
	Minute	Setting	0-59
	Second	Setting	0-59
COM 1 <i>read only</i>	Baud rate	>current value<	9600/19200/38400/56800
COM 2 <i>read only</i>	Baud rate	>current value<	9600/19200/38400/56800/115k
SSB <i>read only</i>	Baud rate	>current value<	1MBaud, 500k, 250k, 125k, 100k, 50k
	Cycle Time	>current value< [in µs]	read only
	Utilization	>current value< [in %]	read only
K-bus <i>read only</i>	Diagnosis	>current diagnosis<	read only
	Number of Bus Terminals	>current value<	read only

### Bus-specific menu items

#### BX3100

F-bus PROFIBUS <i>read only</i>	Address*	>current value<	1-126
	Baud rate*	>current value<	read only
	Status	>current value<	read only
	Diagnostic*	>current value<	read only

#### BX5100

F-bus CANopen <i>read only</i>	Address*	>current value<	1-126
	Baud rate*	>current value<	read only
	Status	>current value<	read only
	Diagnostic*	>current value<	read only

\*) in preparation

**BX9000**

Ethernet	MAC ID	>current value<	000105-xx-xx-xx, read only
	ADDR.STATE	>current value<	read only
	ADDRESSING MODE	FIXED IP (default) DHCP BOOTP BOOTP & SAVE	read / write
	NAME	>current value<	BX_xxxxxx (xxxxxxx last 3 Bytes from the MAC ID) read / write
	DEFAULT GATEWAY	0.0.0.0	read / write
	IP MASK	255.255.0.0	read / write
	IP ADDRESS	172.16.21.20	read / write

**Code**

The default setting is "\*\*\*\*", i.e. no password is active. A password is required for setting parameters.

**Menu navigation**

Press the navigation switch for three seconds to switch to the Directory menu. Some of the menu items are described below.

**MENU****MENU**

Main Menu

DOWN (press briefly)

**PASSWORD**  
 ???

 ??? - password set  
 \*\*\*\* - no password set

PRESS (press briefly)

**PASSWORD ENTER?**  
 ???

 Do you want to enter the password  
 Yes - PRESS (press for approx. 2 seconds, then enter the password) / No - UP

PRESS (press for approx. 2 seconds, then enter the password)

**PASSWORD**  
 ????

 Enter password  
 PRESS <OK>

PRESS

**PASSWORD**  
 1234

 Enter password  
 OK <PRESS> and <PRESS> again to confirm (press for approx. 1 second until OK appears in the display)



**F bus (only BX3100)**

**F-BUS**  
**WAIT FOR SETPRM**

Fieldbus status (read only)  
WAIT FOR SETPRM - waiting for parameter data from PROFIBUS

**SSB**

**SSB**

Smart System Bus

**COM2**

**COM2**

Serial interface COM2 (read only)

DOWN

**Baudrate**  
**xxx**

Current baud rate (read only)

**COM1**

**COM1**

Serial interface COM2 (read only)

DOWN

**Baudrate**  
**xxx**

Current baud rate (read only)

**K-Bus**

**KBUS OK**  
**10 TERMINALS**

K-Bus diagnosis (read only)

DOWN

**KBUS RESET**

K-Bus reset

PRESS (short)

**KBUS RESET EXCT?**

PRESS 1 sec - the K-Bus is reset

**PLC**

**PLC**

PLC status (read only)

DOWN

**PROJECT**  
**>NAME<**

PLC project name

RIGHT

**CURR. EXEC. TIME**  
**xxx ms**

Total processing time in [ms]

RIGHT

**CYCLE TIME**  
**20 ms**

Set cycle time

## 4.4.2 Creating own menus

The display and the navigation switch can also be used for user-specific purposes, for example displaying diagnostic information or changing parameters. A simple example is provided that can be used and adapted to get you started.



Download (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207305483.prx>):

## 4.5 Configuration software KS2000

Bus Terminal controllers of the BCxx50, BXxx20 and BXxx00 series cannot be parameterized and configured with the KS2000 configuration software. These devices must be configured with the TwinCAT System Manager.

The KS2000 configuration software offers configuration and diagnostic support for the Bus Terminals attached to the Bus Terminal Controller.

It is advisable to set the baud rate in the KS2000 configuration software and the BCxx50 BCxx20 and BXxx00 to 38400 baud (8 data bits, even, 1 stop bit).

	<b>COM1 - automatic baud rate detection</b>
Note	The COM 1 interface of the BXxx00 features automatic baud rate detection between 9.6 kbaud and 56.4 kbaud.
	<b>Required KS2000 version</b>
Note	Configuration or diagnostics of Bus Terminals at BXxx00 is supported from KS2000 version 4.3.14.

In some Bus Terminals (e.g. KL25xx, KL6811, KL6201, KL6401) the following parameters must be set in order to be able to use the configuration dialogs:

- A PLC project or boot project must be deactivated.
- The BX controller must be in its default configuration. Set the manufacturer's setting or switch to Config Mode in the TwinCAT System Manager (blue TwinCAT icon).
- The BX controller must be in FreeRun mode. Activate it with the TwinCAT System Manager.

You can now log in with the KS2000 configuration software via ADS (port 100) or the serial cable and use the KS2000 dialogs in the Bus Terminals.

## 5 Programming

### 5.1 PLC features of the BX controllers

Description	Value
Data memory	256 kbyte
Program memory	256 kbyte minus task-configuration minus POU's during online change
Source code memory	256 kbyte
RETAIN	2 kbyte
INPUT	2 kbyte
OUTPUT	2 kbyte
FLAG	4 kbyte
Max. variable size	16 kbyte
Max. POU's	Limited by memory

### 5.2 TwinCAT PLC

The Beckhoff TwinCAT Software System turns any compatible PC into a real-time controller with a multi-PLC system, NC axis control, programming environment and operating station. The TwinCAT programming environment is also used for programming the BC/BX. If you have TwinCAT PLC (Windows NT4/2000/XP) installed, you can use the fieldbus connection or the serial port for downloading and debugging software.

TwinCAT I/O or TwinCAT PLC can also be used as the Ethernet Master (host), in order to exchange process data with the Bus Terminal Controller. TwinCAT provides you with the System Manager as a configuration tool, as well as the drivers and the ADS protocol.

#### Bus Terminal Controllers of the BCxx50, BCxx20 and BXxx00 series

These 2nd-generation Bus Terminal Controllers are configured with the TwinCAT System Manager and programmed with TwinCAT PLC Control. TwinCAT PLC must be installed for these couplers (Windows NT4, Windows 2000, Windows XP).

#### Programming and program transfer

- via the serial interface [► 170]
- via the fieldbus interface [► 169] (only for Bus Terminal controllers for PROFIBUS, CANopen and Ethernet)

#### Online change

The Bus Terminal Controllers of the BX series and the BCxx50 support online change. This means that the PLC program is replaced with a new program without interrupting the program. The switch-over to the new program occurs after the task is completed. This means that two versions of the PLC program have to be stored. 512 kbyte are available, which therefore have to be divided by two, leaving 256 kbyte for the actual PLC program. In addition, several kbyte are required for task configuration etc. During an online change, dynamic data are stored in memory. Should a program approach the memory limit (program size greater than 240 kbyte), the online change may no longer work, even though the program may still be written to the BX after "Rebuild all".

#### When is online change not available?

Online change is not available under certain conditions,.

- Inserting of a new library
- Changing the task setting

- "Rebuild all"
- Controller memory limit is almost reached (PLC program greater than 90%)

## 5.3 TwinCAT PLC - Error codes

Error type	Description
PLC compiler error	Maximum number of POU's (...) exceeded
PLC compiler error	Out of global data memory ...

### Error POU's

For each function block one POU (process object unit) is created. 256 function blocks are available by default.

**Error 3612: Maximum number of POU's (100) exceeded! Compile is aborted.**

Data allocation

1 Error(s), 0 Warning(s).

Fig. 85: Maximum number of POU's exceeded

If libraries are integrated this value may be insufficient. In this case, the number of POU's should be increased.

To this end, open in PLC Control under Projects/Options...

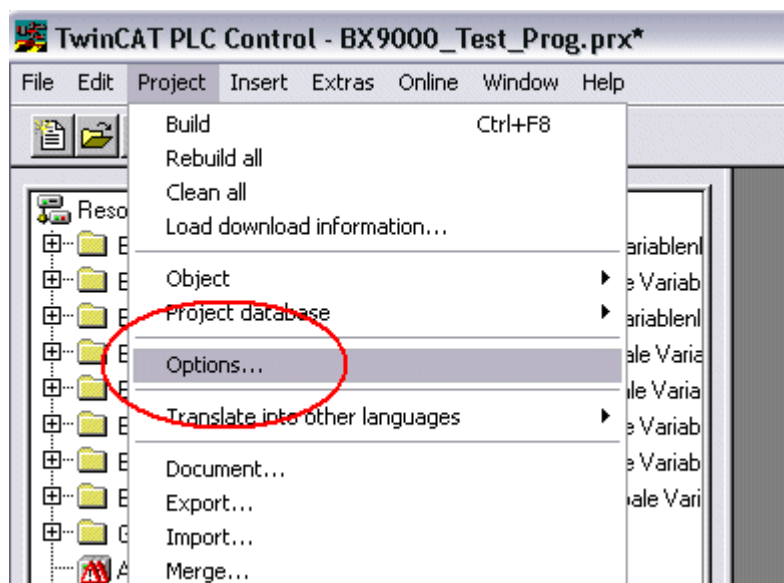


Fig. 86: Menu path Projects / Options / Controller Settings

...the controller settings.

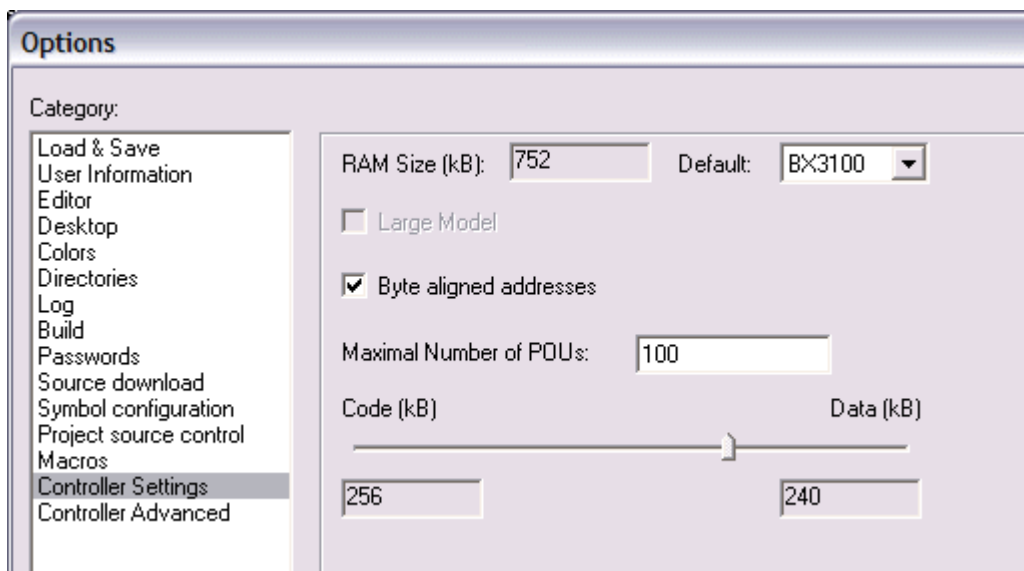


Fig. 87: Controller settings

Changing these settings will deactivate online changes.

### Global memory error

Interface of POU 'MAIN'

Data allocation

Error 3803: MAIN (7): Out of global data memory. Variable 'Test\_', 16002 bytes.

1 Error(s), 0 Warning(s).

Fig. 88: Global memory insufficient

2 x 16 kbyte of data are available by default. If large data quantities are to be used, this range should be increased. A maximum of 14 data segments are possible for the BX.

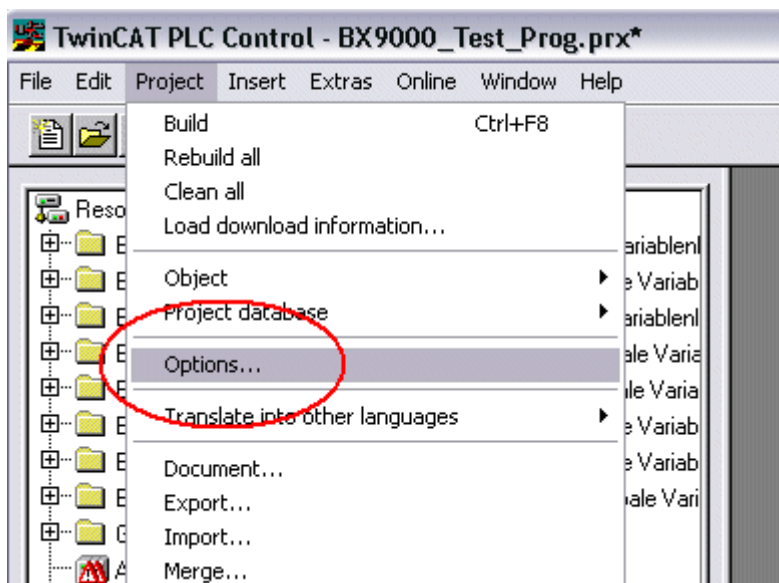


Fig. 89: Menu path Projects / Options / Build

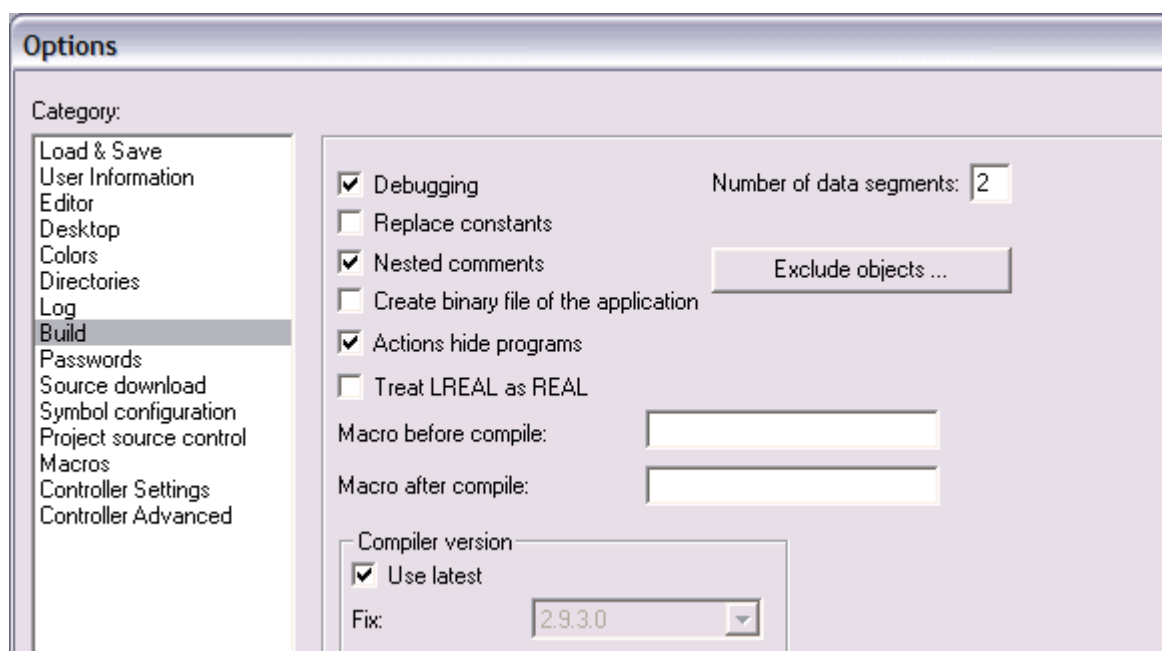


Fig. 90: Build

## 5.4 Remanent data

2000 kbyte of remanent data are available for the BX controller. These data are declared as VAR RETAIN in PLC Control:

### Example

```
VAR RETAIN
    Test    :BOOL;
    Count   :INT;
END_VAR
```

Retain data are located between VAR RETAIN and END\_VAR. These data are stored in a NOVRAM and are consistent across the whole 2 kbyte range. The RETAIN data are stored in the NOVRAM after each cycle. For 2 kbyte approx. 2 ms are required (for 1 kbyte approx. 1 ms). The variables can be configured locally or globally. Allocated variables (%MB, %QB, %IB) cannot be used as remanent data.



#### Note

### Do not use VAR\_RETAIN in function blocks

VAR\_RETAIN should not be used in function blocks. All FB data are copied into the retain memory. This leads to an unnecessary increase in cycle time, and the retain memory is filled with unnecessary data.



#### Note

### Do not use variables with address as remanent data

Variables that have been assigned an address (%MB, %QB, %IB) must not be used as remanent data.

### Example for remanent data in the function block

This should be avoided, if possible, since all the data of a function block, in which even just a single remanent bit is found, are stored by default. A program sample can be found below.

### Function block test (no program code required - in ST semicolon is sufficient)

```
FUNCTION_BLOCK Test
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
```

```

END_VAR
VAR_IN_OUT
    Counter    :INT;
END_VAR

```

### MAIN program

```

PROGRAM MAIN
VAR
    fb_Test:Test;
END_VAR
VAR_RETAIN
    iCounter1:INT;
END_VAR
fb_Test(Counter:=iCounter1);

```

## 5.5 Persistent data

The Bus Terminal Controller has 1000 bytes of persistent data available. In contrast to the retain data, these are not deleted, even with a new project, a PLC reset or a new download.

In order to use the persistent data, these must first be activated once with a function block from the PLC.

Secondly, the variables should reside in the allocated flag area. Here you can choose where the persistent data reside.

4 kbytes of allocated flags are available, of which 1000 bytes can be declared as persistent data.

### Example

```

VAR
    Test AT %MX1000 :BOOL;
    Count AT %MB1002 :INT;
END_VAR

```

The **Persistent\_Data** function block can be used to specify the start address and the length (in bytes) from which the data are to be persistent.

The input variable *WriteOffset* is used to specify the byte offset of the flag area, *WriteSize* is used for the length in bytes.

The function block can be found in the TcSystemBX.lbx library. Should this not be available, it can be downloaded from this documentation (see [Libraries](#) [► 106]).

### Example values

WriteOffset 1000  
WriteSize 10

All data in the range %MB1000 - %MB1009 are then persistent. The variable type is irrelevant.

Like the retain data, the data are copied to the NOVRAM and are therefore writeable in each cycle.



#### Note

#### Persistent data from firmware 1.17

Persistent data is supported for all BX controllers from firmware 1.17 or higher.




#### Note

#### Parameters are valid immediately

The parameters only have to be written once, after which they are valid immediately. These data are stored permanently. Activation of the factory setting deletes everything, including the persistent data.

**Sample Program**

Click on the link  (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207307659.prx>) to download a sample program from this documentation.



## 5.6 Allocated flags

4 kbyte of allocated flags are available. They can be used to assign different variable types to the same address, e.g. for converting strings to bytes. Data can also be placed here that can be read or written via ADS by the controller.



**Note**

### Allocated variables are not remanent data

For the Bus Terminal Controllers of the BX series and the BCxx50 the allocated variables are **not** saved as remanent data.

### Reading/writing of allocated flags via ADS

The flags may also be read via the controller and ADS. In PROFIBUS, the DPV-1 services are used for this purpose, in CANopen SDO communication is used.

The AmsNetID can be obtained from the System Manager, or it can be displayed via the Bus Terminal Controller menu.

The PLC port number is 800.

Index group	Meaning	Index offset (value range)
0x4020	Flag (only BXxxx0)	0..4096

### Example

#### BX program

```
VAR
    Flag_01 AT %MB0: WORD;
END_VAR
```

#### TwinCAT PC/CX master program

```
VAR
    fbADRSREAD: ADSREAD;
    Flag_M: WORD;
END_VAR

fbADRSREAD (
    NETID:='172.16.3.0.2.3' , (* AMSNetId BX *)
    PORT:=800 , (* 800 - PLC *)
    IDXGRP:=16#4020 , (* 0x4020hex falgs *)
    IDXOFFS:=0 , (* byte offset *)
    LEN:=2 , (* Lenght byte *)
    DESTADDR:=ADR(Merker) ,
    READ:=TRUE ,
    TMOUT:=t#1s );
IF NOT fbADRSREAD.BUSY THEN
    fbADRSREAD(READ:=FALSE);
END_IF
```

## 5.7 Local process image in delivery state (default config)

The process image of the Bus Terminal Controller consists of input, output and flag area. In addition, there are unallocated data without fixed address. They are created without specifying an address. For these variable types the memory allocation is as follows:

- BCxx50 48 kbyte,
- BC9x20 128 kbyte,
- BXxx00 256 kbyte.

The maximum size of a variable or structure (array) is 16 kbyte. For the allocated data 2048 bytes of input data and 2048 bytes of output data are available. The Bus Terminal Controller has 4 kbyte of memory allocated for the flag area.

In the delivery state (default configuration) of the BX/BCxx50, fixed addresses are allocated for all connected Bus Terminals. The data for Ethernet communication start from address offset 1000<sub>dec</sub>. The length of the Ethernet data depends on how much data has been configured; on the BX9000 it has a maximum length of 1000 bytes.

Inputs	Outputs
Bus Terminal %IB0 ...	Bus Terminal %QB0 ...
Ethernet DATA (PLC variables) %IB1000 ...(Modbus TCP/ADS-TCP/ADS-UDP)	Ethernet DATA (PLC variables) %QB1000 ... (Modbus TCP/ADS-TCP/ADS-UDP)
... %IB2047 maximum	... %QB2047 maximum

### Addressing of the connected Bus Terminals

The default setting is for all the connected Bus Terminals to be assigned to the local process image. Mapping within the Bus Terminal Controller is carried out according to the following rule: First all the complex Bus Terminals, in the sequence they are physically inserted, followed by the digital Bus Terminals which are filled to a byte. The default mapping of the complex Bus Terminals is:

- complete evaluation
- Intel format
- Word alignment

### Example structure

Bus Terminal Controller: 1 x BCxx50, BCxx20 or BXxx00

Position 1: 1 x KL1012

Position 2: 1 x KL1104

Position 3: 1 x KL2012

Position 4: 1 x KL2034

Position 5: 1 x KL1501

Position 6: 1 x KL3002

Position 7: 1 x KL4002

Position 8: 1 x KL6001

Position 9: 1 x KL9010

Table 1: Process image

Bus Terminal	Position	Input image	Output image	Size
KL1501	5	%IB0...%IB5	%QB0...%QB5	6 bytes
KL3002	6	%IB6...%IB13	%QB6...%QB13	8 bytes
KL4002	7	%IB14...%IB21	%QB14...%QB21	8 bytes
KL6001	8	%IB22...%IB29	%QB22...%QB29	6 bytes
KL1012	1	%IX30.0...%IX30.1	-	Bit 2
KL1104	2	%IX30.1...%IX30.5	-	Bit 4
KL2012	3	-	%QX30.0...%IX30.1	Bit 2
KL2034	4	-	%QX30.2...%IX30.5	Bit 4
KL9010	9	-	-	-

## 5.8 Mapping the Bus Terminals

The precise assignment of the byte-oriented Bus Terminals may be found in the configuration guide for the particular bus terminal. This documentation is available on the Beckhoff *Products & Solutions* CD or on the Internet under <http://www.beckhoff.de>.

Byte oriented Bus Terminals	Bit oriented Bus Terminals
KL15x1	KL10xx, KL11xx, KL12xx, KL17xx, KM1xxx
KL25xx	KL20xx, KL21xx, KL22xx, KL26xx, KL27xx, KM2xxx
KL3xxx	
KL4xxx	
KL5xxx	
KL6xxx	
KL7xxx	
KL8xxx	
	KL9110, KL9160, KL9210, KL9260

## 5.9 Local process image in the TwinCAT configuration

The TwinCAT configuration (TwinCAT CONFIG) enables free mapping between fieldbus, K-bus and PLC variables. Variables can be linked independent of their address via the System Manager.

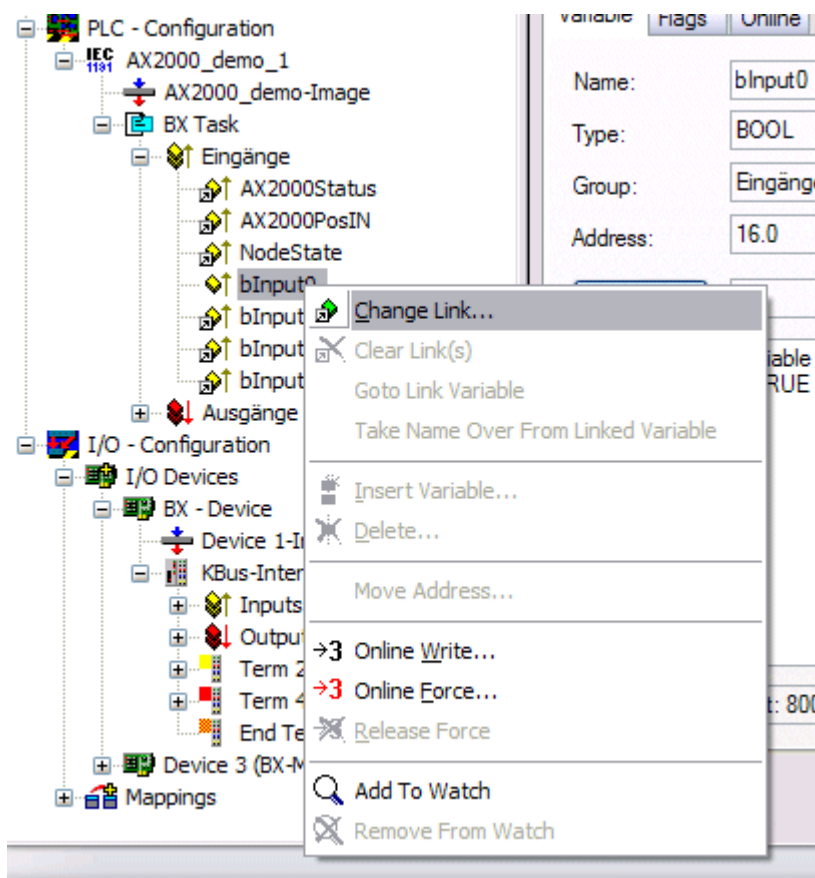


Fig. 91: Changing variable links

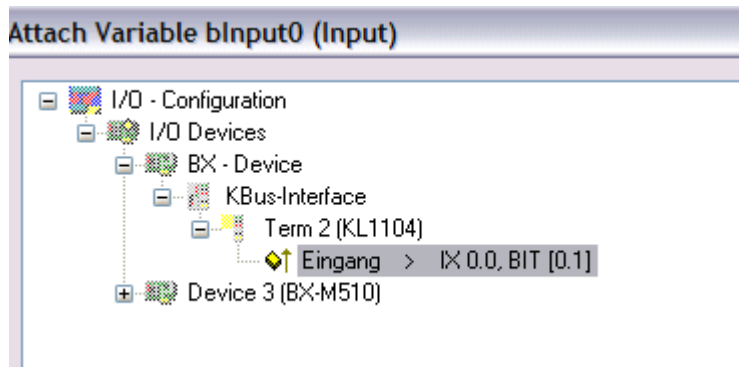


Fig. 92: Linking a variable with an input

In the default configuration all Bus Terminals are assigned fixed addresses. If a Bus Terminal is inserted, the whole address range may be shifted. The TwinCAT configuration enables allocated variables to be linked to a Bus Terminal, as required. This is parameterized in the System Manager, and the configuration is then downloaded to the Bus Terminal Controller (see [TwinCAT configuration](#) [► 32]). It is also possible to upload an existing TwinCAT configuration.

## 5.10 Creating a boot project

The following memory resources are available for generating the boot project

- approx. 250 kbyte flash on the Bus Terminal controllers of the BX series;
- approx. 48 kbyte flash on the Bus Terminal controllers of the BCxx50 series.

### PLC Control

After logging into TwinCAT PLC Control, a boot project can be created.

- Opening a PLC project
- Selecting the target system (or selection the serial interface)
- Logging into the BX/BCxx50
- Creating a boot project (Online\Create boot project)

The PLC LED lights up green once a valid boot project is available on the BX/BCxx50.

In the Bus Terminal controllers of the BX series, the PLC LED flashes orange while boot project is created. The PLC LED lights up orange if no boot project is available on the BX.

### Deleting a boot project

The boot project can be deleted from the Bus Terminal Controller. The following steps must be followed:

- Opening the project
- Logging into the Bus Terminal Controller
- Deleting the boot project (Online\Delete boot project)

The PLC LED lights up orange when the boot project is deleted.



**Note**

#### Using the current project as boot project

After an online change the old project is still shown as boot project. To use the current project (after the online change) as the boot project, the boot project has to be recreated.

### Bypassing the start of the boot project\*

With the Bus Terminal controllers of the BX series, starting of the boot project during booting can be prevented by pressing the Navi button. This does not delete the boot project. The project is reloaded when the Bus Terminal Controller is rebooted.

\* from version 0.85

## 5.11 Communication between TwinCAT and BX/BCxx50

For transferring data from TwinCAT to the Bus Terminal Controller, it makes sense to organize the data in a structure. Please note the following to account for the differences in data management on the two systems.

- If two different data types are sent in sequence (e.g. byte and INT), the following variable is set to the next even address offset
- Boolean variables should never be allocated individually within a structure, since they would invariably occupy 1 byte. Boolean expressions should always be masked in a byte or word.

### Example 1: A structure on the BX/BCxx50 and on the PC

Variable	BX/BCxx50 memory	PC memory (TwinCAT)
Byte	%..B0	%..B0
INT (1)	%..B2	%..B1
INT (2)	%..B4	%..B3

Due to the fact that another variable type (INT) follows the first byte, in the BX/BCxx50 it was assigned the next free even address. In order to achieve the same data structure on both systems, a dummy byte has to be inserted in the PC project (see example 2).

### Example 2: A structure on the BX/BCxx50 and on the PC with the same memory allocation

Variable	BX/BCxx50 memory	PC memory (TwinCAT)
Byte	%..B0	%..B0
Byte (dummy)	%..B1 (not necessarily required, since the system deals with this itself if the variable does not exist)	%..B1
INT (1)	%..B2	%..B2
INT (2)	%..B4	%..B4

### Data structure

```
Type PB_Data
STRUCT
    wVar_1:WORD;
    iValue_1:INT;
    iValue_2:INT;
    iValue_3:INT;
END_STRUCT
END_TYPE
```

### Creating a variable structure

```
VAR_Global
    strData_Out AT %QB1000:PB_Data; (*PLC Variables *)
    bInput_01 AT %IX0.0:BOOL; (* Input from a terminal *)
END_VAR
```

### Small programming example

```
strData_Out.wVar_1.0:=bInput_01;
```



#### Note

#### Do not use real values in a mixed data structure

A mixed data structure should not contain real values. If this is nevertheless the case, the high and low words must be swapped in the BX/BCxx50 or in the TwinCAT master project. It is better to use an array of Real values or to transfer the Real values individually.



#### Note

#### Larger fieldbus data blocks

You can transfer larger fieldbus data blocks, in order to have a reserve for your structure. Disadvantage: These reserves are then transferred with each fieldbus telegram, resulting in overload of the fieldbus communication.

## 5.12 Up- and downloading of programs

The Bus Terminal Controller has a memory for the source code. It can be used for storing the program, the task configuration, and the libraries. Should the memory be insufficient, the source code may be stored without task configuration and libraries. This takes up significant less memory space!

### General settings

The timing of the source code download to the target system can be specified via Edit/Options. Open the options menu.

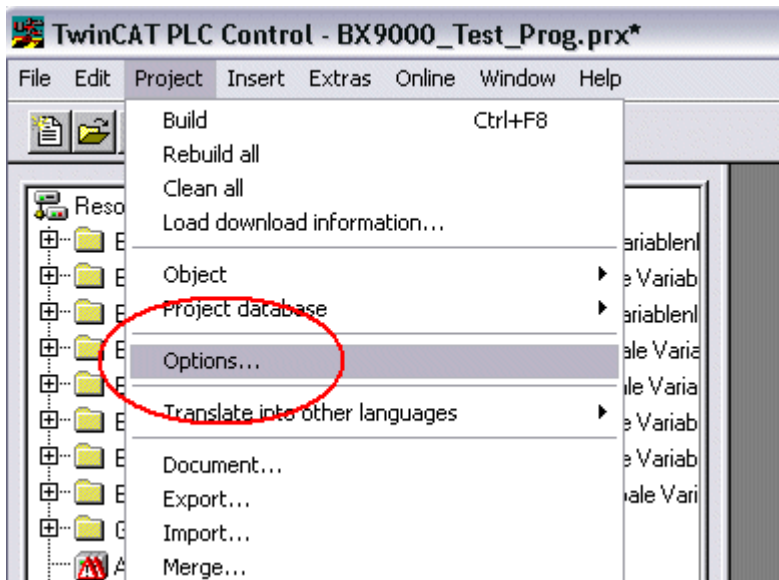


Fig. 93: Opening the options menu

Select Source Download.

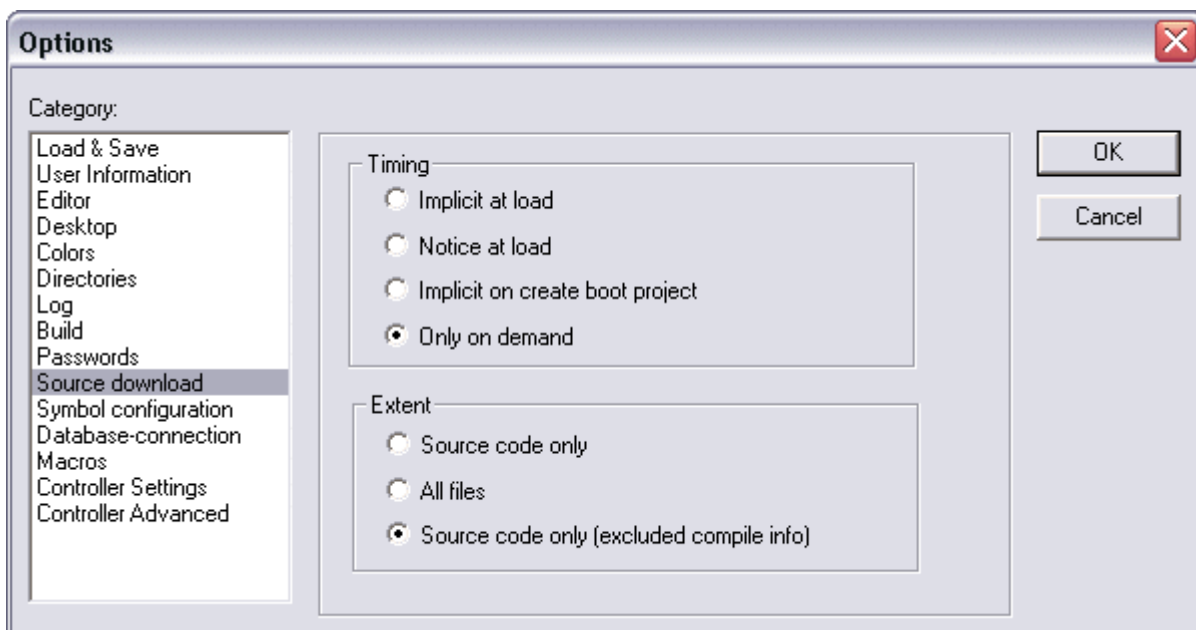


Fig. 94: Selecting Source Download

Here you can set which parts of the source code are to be downloaded to the Bus Terminal Controller, and when.

**Source code only:** the prx file with information on the online change is transferred. Login via online change is possible (the PLC does not stop).

**All files:** as *Source code only*, plus all required libraries.

**Source code only (compile info excluded):** only the prx file is transferred. Login is only possible when the PLC stops.

Which option you can use depends on the size of your projects.

### Downloading a program

The source code can be transferred to the target system on request. This requires the user to be logged in with his program. Under Online/Source code download the program code can now be transferred to the Bus Terminal Controller.

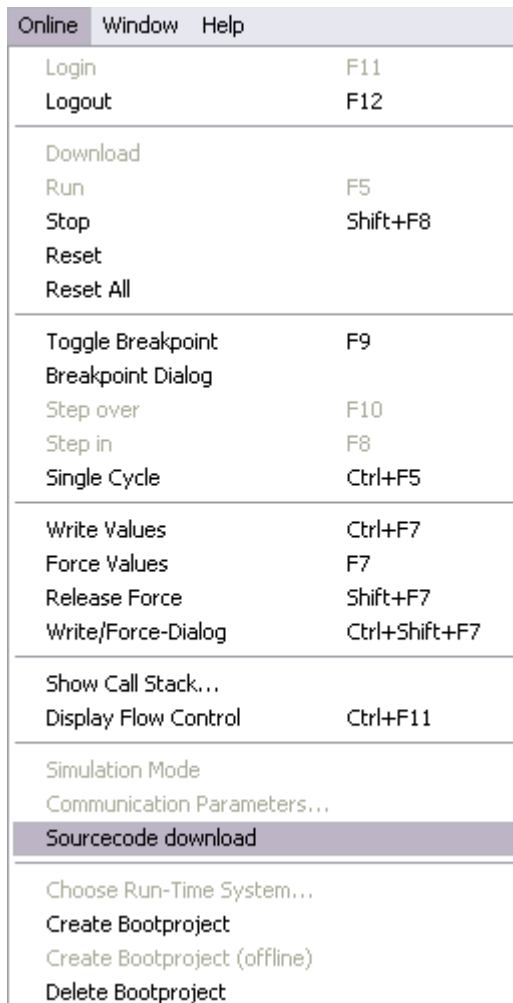


Fig. 95: Downloading the program code

After a short delay, a window will open that indicates the download progress.

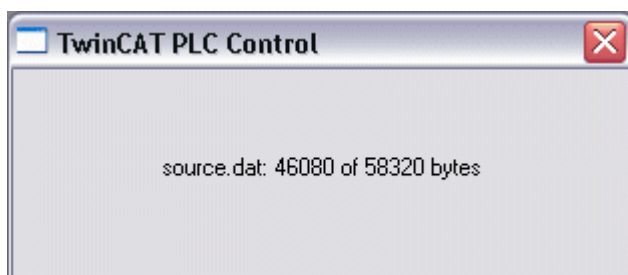


Fig. 96: Download progress



## Uploading a program

For uploading the program code again, open a new file in PLC Control. Then click on the PLC button.

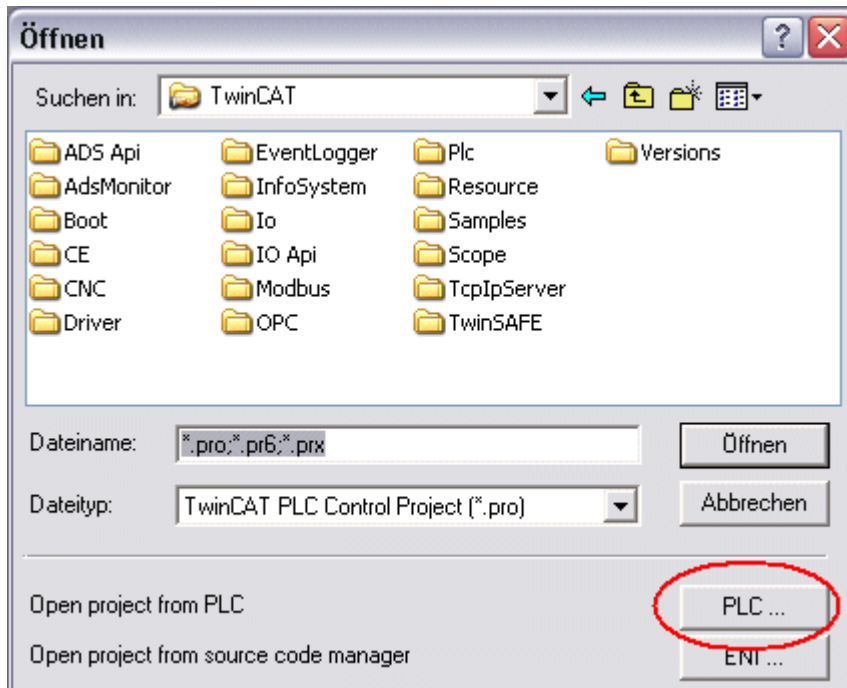


Fig. 97: Uploading a program

Select the data transfer route:

- *BCxx50 or BX via AMS*, if you are connected to the Bus Terminal Controller via the fieldbus, or
- *BCxx50 or BX via serial*, if you are connected to the Bus Terminal Controller via the serial interface.

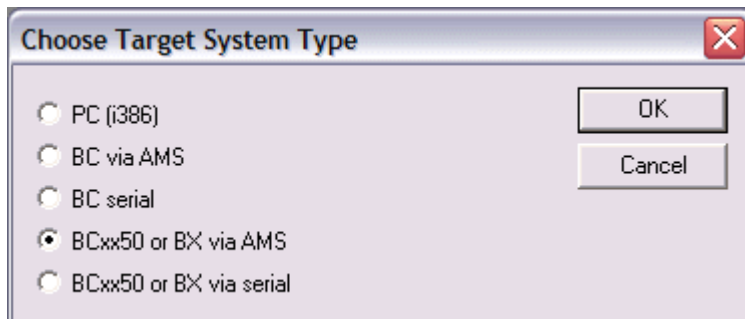


Fig. 98: Selecting the data transfer route

Then select the device and confirm with OK.

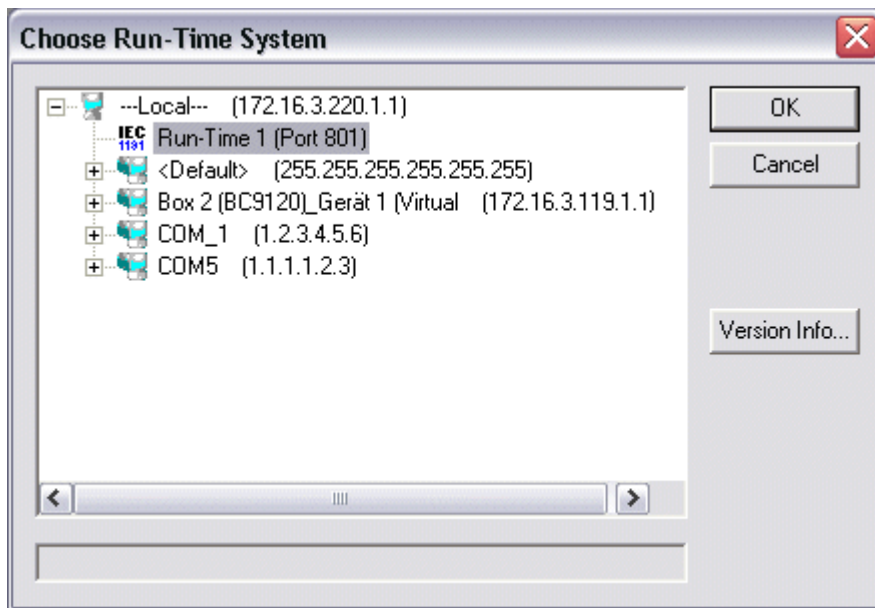


Fig. 99: Selecting the device

The source code will now be uploaded.

### Password

You can protect your project with a password (in PLC Control Project/Options/Passwords).

## 5.13 Libraries

### 5.13.1 Libraries overview

Various libraries are available for the Bus Terminal Controllers (Bus Coupler with PLC functionality: BXxxxx) (see [Beckhoff Information System](#)).

#### Download

To download the libraries click on the link. Please copy the libraries to directory TwinCAT\PLC\LIB.

- Standard (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207309835.zip>)



- TcSystemBX (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207312011.zip>)



- (the TcSystemBX requires the TcBaseBX library)

- TcBaseBX (Download)



- (TcDisplayBX, TcNaciSwitchBX and TcDebugBX are now included here)

- TcComPortBX (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207314187.zip>)



- ChrAscBX.lbx (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207316363.zip>)



**Note****Use the library that matches the firmware**

The latest firmware requires the latest library. If you update your BX Controller, please also change the libraries.

Copy these libraries to the LIB directory, remove these libraries from your project and add them again.

**TcSystemBX**

ADS	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
ADSREAD	04.03.04	0.90	0.14	1.00	0.02	1.00
ADSWRITE	04.03.04	0.90	0.14	1.00	0.02	1.00
ADSRDWRT	04.03.04	0.90	0.14	1.00	0.02	1.00
ADSWRTCTL	04.03.04	0.90	0.14	1.00	0.02	1.00
ADSRDSTATE	04.03.04	0.90	0.14	1.00	0.02	1.00
ADSRDDEVINFO	04.03.04	0.90	0.14	1.00	0.02	1.00

Bit Functions	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
CLEARBIT32	07.03.03	0.28	0.01	1.00	0.01	1.00
CSETBIT32	07.03.03	0.28	0.01	1.00	0.01	1.00
GETBIT32	07.03.03	0.28	0.01	1.00	0.01	1.00
SETBIT32	07.03.03	0.28	0.01	1.00	0.01	1.00

Display Function	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
<a href="#">FB_DispWrite</a> ▶ <a href="#">120</a>	31.03.03	0.28	0.01	1.00	0.01	1.00

Diagnosis	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
BX_Security	15.08.06	1.12	1.14	-	1.12	1.14
DeviceType	15.08.06	1.12	1.14	-	1.12	1.14
FirmwareVersion	15.08.06	1.12	1.14	-	1.12	1.14
FirmwareVersionString	15.08.06	1.12	1.14	-	1.12	1.14
DeviceType	15.08.06	1.12	1.14	-	1.12	1.14
Read_Diagnose	15.08.06	1.12	1.14	-	1.12	1.14
CRCBootproject	15.08.06	1.14	1.14	-	1.14	1.14

Read Address	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
ReadSlaveAddress	15.08.06	1.12	1.12	1.10	1.12	-

Controller	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
FB_BasicPID	04.03.04	0.64	0.01	1.00	0.01	1.00

Event Logger Functions	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
-	-	-	-	-	-	-

File Access	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
FB_ReadFromFile	03.08.04	1.04	1.04	1.00	1.04	1.00
FB_WriteToFile	03.08.04	1.04	1.04	1.00	1.04	1.00
FB_ReadWriteFile	03.08.04	1.04	1.04	1.00	1.04	1.00

Memory Functions	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
MEMCMP	07.03.03	0.41	0.01	1.00	0.01	1.00
MEMCYP	07.03.03	0.41	0.01	1.00	0.01	1.00
MEMMOVE	07.03.03	0.41	0.01	1.00	0.01	1.00
MEMSET	07.03.03	0.41	0.01	1.00	0.01	1.00

NOVRAM Functions	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
-	-	-	-	-	-	-

Serial Communication Interface	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
FB_COMPortClose	14.07.03	0.49	0.01	1.00	0.01	1.00
FB_COMPortOpen	14.07.03	0.49	0.01	1.00	0.01	1.00
F_COMPortRead	14.07.03	0.49	0.01	1.00	0.01	1.00
F_COMPortWrite	14.07.03	0.49	0.01	1.00	0.01	1.00

SFC	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
AnalyzeExpression	07.03.03	0.28	0.01	1.00	0.01	1.00
AppendErrorString	07.03.03	0.28	0.01	1.00	0.01	1.00
SFCActionControl	07.03.03	0.28	0.01	1.00	0.01	1.00

System / Time / TBus	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
DRAND	07.03.03	0.28	0.01	1.00	0.01	1.00
RTC	07.03.03	0.28	0.01	1.00	0.01	1.00
SYSTEMTIME_TO_DT	07.03.03	0.28	0.01	1.00	0.01	1.00
DT_TO_SYSTEMTIME	07.03.03	0.28	0.01	1.00	0.01	1.00
GetSysTick	14.07.03	0.49	0.01	1.00	0.01	1.00
PresetSysTick	14.07.03	0.49	0.01	1.00	0.01	1.00
Reboot	21.07.03	0.59	0.14	1.00	0.02	1.00
Persistent_Data	21.08.07	1.17	1.17	-	1.17	1.17

Debug	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
<a href="#">F_ReadDebugTimer</a> <a href="#">▶ 119</a>	08.08.03	0.59	0.14	1.00	0.02	1.00
<a href="#">F_StartDebugTimer</a> <a href="#">▶ 119</a>	08.08.03	0.59	0.14	1.00	0.02	1.00

NaviSwitch	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
All function blocks [► 119]	10.10.03	0.64	0.14	1.00	0.02	1.00

#### TcComPortBX

Com FBs	Version	Firmware				
		BX3100	BX5100	BX5200	BX8000	BX9000
All function blocks [► 125]	20.08.03	0.60	0.02	1.00	0.01	1.00

## 5.13.2 Overview of libraries for BX9000, BC9020, BC9050, BC9120

Special function blocks for the Bus Terminal Controllers BX9000, BC9050, BC9120 and BC9020.

### Download

For downloading libraries left-click on the disk icon. Then copy the libraries into the directory TwinCAT\PLC\LIB.

- TcBaseBX9000 (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207318539.zip>)



#### Note

#### Use the library that matches the firmware

The latest firmware requires the latest library. If you update your BX Controller, please also change the libraries.  
Copy the libraries into the directory TwinCAT\PLC\LIB. Then remove the libraries from your project and add them again.

#### TcBaseBX9000

Services	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_AddDnsServer [► 161]	30.05.06	1.12	B0	B0	B1
FB_GetHostByAddr [► 162]	30.05.06	1.12	B0	B0	B1
FB_GetHostByName [► 163]	30.05.06	1.12	B0	B0	B1
FB_GetNetworkConfig [► 164]	30.05.06	1.12	B0	B0	B1
FB_SetTargetName [► 165]	30.05.06	1.12	B0	B0	B1

SMTP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_SMTP [► 158]	30.05.06	1.12	B0	B0	B1

SNTP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_Sntp [► 160]	30.05.06	1.12	B0	B0	B1

IP-TCP/IP-UDP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_AddMultiRoute [► 143]	26.09.06	1.14	-	-	-
FB_DelMultiRoute [► 143]	26.09.06	1.14	-	-	-
FB_IpClose [► 143]	26.09.06	1.14	B0	B0	B1
FB_IpEndSession [► 143]	26.09.06	1.14	B0	B0	B1
FB_IpOpen [► 143]	26.09.06	1.14	B0	B0	B1
FB_IpReceive [► 143]	26.09.06	1.14	B0	B0	B1
FB_IpSend [► 143]	26.09.06	1.14	B0	B0	B1
FB_IpStartSession [► 143]	26.09.06	1.14	B0	B0	B1

ModbusTCP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_MBClose [► 155]	26.09.06	1.14	B0	B0	B1
FB_MBConnect [► 155]	26.09.06	1.14	B0	B0	B1
FB_MBGenericReq [► 155]	26.09.06	1.14	B0	B0	B1

### 5.13.3 TcBaseBX

#### 5.13.3.1 System task information

```
VAR_GLOBAL
    SystemTaskInfoArr : ARRAY[1..2] OF SYSTEMTASKINFOTYPE;
END_VAR
```

System flags are implicitly declared variables. Using the Input Assistant, a variable SystemTaskInfoArr can be found under system variables. This variable is a field with four structures of type `SYSTEMTASKINFOTYPE` [► 110]. The structure definition can be found in the system library. The index in this field is the task ID.

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.9.0	BX Controller	TcBaseBX.lbx

#### 5.13.3.2 System task information type

```
TYPE SYSTEMTASKINFOTYPE
STRUCT
    active          :   BOOL;
    taskName        :   STRING(16);
    firstCycle      :   BOOL;
    cycleTimeExceeded :   BOOL;
    cycleTime       :   UDINT;
    lastExecTime    :   UDINT;
    priority        :   BYTE;
```

```

    cycleCount      :      UDINT;
END_STRUCT
END_TYPE

```

### Legend

active: This variable indicates whether the task is active.

taskName: the task name.

firstCycle: During the first PLC task cycle, this variable has the value: TRUE.

cycleTimeExceeded: this variable indicates whether the set task cycle time was exceeded.

cycleTime: set task cycle time in multiples of 100 ns.

lastExecTime: cycle time required for the last cycle in multiples of 100 ns.

priority: set task priority.

cycleCount: cycle counter.

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.9.0	BX Controller	TcBaseBX.lbx

### 5.13.3.3 System info

```

VAR_GLOBAL
    SystemInfo      : SYSTEMINFOTYPE;
END_VAR

```

System flags are implicitly declared variables. Using the Input Assistant, a variable Systeminfo can be found under system variables. The type `SYSTEMINFOTYPE` [► 111] is declared in the system library. For accessing the variable, the system library has to be integrated in the project.

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.9.0	BX Controller	TcBaseBX.lbx

### 5.13.3.4 System information type

```

TYPE SYSTEMINFOTYPE
STRUCT
    runTimeNo      :      BYTE;
    projectName    :      STRING(32);
    numberOfTasks  :      BYTE;
    onlineChangeCount :      UINT;
    bootDataFlags  :      BYTE;
    systemStateFlags :      WORD;
END_STRUCT
END_TYPE

```

### Legend

runTimeNo: specifies the number of the runtime system (1).

projectName: project name as STRING.

numberOfTasks: number of tasks contained in the runtime system (max. 2).

onlineChangeCount: number of online changes since the last complete download.

bootDataFlags: Reserved

systemStateFlags: Reserved.

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.9.0	BX Controller	TcBaseBX.lbx

### 5.13.3.5 ADS

#### 5.13.3.5.1 Local ADS Port Numbers - Overview

Port number	Description
100 [ <a href="#">▶ 112</a> ] <sub>dec</sub>	Reading and writing of registers and tables from the coupler and the complex Bus Terminals
150 [ <a href="#">▶ 83</a> ] <sub>dec</sub>	Reading and writing of RTC (real-time clock)
153 [ <a href="#">▶ 63</a> ] <sub>dec</sub>	SSB - reading of the emergency message
800 [ <a href="#">▶ 112</a> ] <sub>dec</sub>	Local process image of the PLC, see also port 801
801 [ <a href="#">▶ 112</a> ] <sub>dec</sub>	Local process image of the PLC, see also port 800
0x1000 + node ID [ <a href="#">▶ 62</a> ]	SSB - SDO communication with CANopen node (slave number)

#### 5.13.3.5.2 ADS services

##### Local Process Image PLC Task 1 Port 800/801

Data can be read from and written to the local process image. If it is necessary for outputs to be written, it is important to ensure that they are not used by the local PLC, because the local controller will overwrite these values. The data are not associated with a watchdog, and therefore must not be used for outputs that would have to be switched off in the event of a fault.

Index group	Meaning	Index offset (value range)
0xF020	Inputs	0...2047
0xF021	Bit inputs	0...16376
0xF030	Outputs	0...2047
0xF031	Bit outputs	0...16376
0x4020	Flags	0...4095
0x4021	Flag bit	0...32760

##### ADS services

##### AdsServerAdsState

Data type (read only)	Meaning
String	Start - the local PLC is running Start - the local PLC is stopped

##### AdsServerDeviceState

Data type (read only)	Meaning
INT	0 - Start - the local PLC is running 1 - Stop - the local PLC is stopped

##### AdsServerType

Data type (read only)	Meaning
String	BX PLC Server



**ADSWriteControl**

Data type (write only)	Meaning
NetID	Net ID of the Ethernet Controller*
Port	800
ADSSTATE	5 - RUN / 6 - STOP
DEVSTATE	0
LEN	0
SRCADDR	0
WRITE	rising edge starts the function block
TMOUT	example: T#1000 ms

\* BC9050, BC9020, BC9120, BX9000

**Register access port 100**

On the Bus Terminal Controllers of the BX series, and on the BCxx50/xx20, the ADS port number for register communication is fixed at 100.

Index group	Index offset (value range)		Meaning
	Hi-Word	Low Word	
0 [READ ONLY]	0...127	0...255	Registers in the Bus Coupler Hi-Word, table number of the Bus Coupler Lo-Word, register number of the table
1...255	0...3	1...255	Register of the Bus Terminals Hi-Word, channel number Lo-Word, register number of the Bus Terminal

**Note****Minimum timeout**

For reading the registers, ensure that the timeout for the ADS function block is set to more than one second.

**Note****Setting the password**

When writing to the registers, the password has to be set (see the documentation for the particular Bus Terminal).

**5.13.3.5.3 Deactivating the LED for cycle time exceeding**

The BX controller monitors the set task cycle time. If it is exceeded, the `cycleTimeExceeded` [► 110] bit and the red *PLC* LED are set. In some applications the value may be exceeded for short periods, which is tolerable. This may be the case when many data are received in serial communication, for example. In order to avoid flickering of the red *PLC* LED, it can be switched off via `ADSWRITE`.

**Structure of the ADSWRITE command**

This `ADSWRITE` command enables the red *PLC* LED of the BX controller to be deactivated.

Input parameters	Description
NETID	local NetId of BX
Port number	800
IDXGRP	16#0000_4080
IDXOFFS	0
LEN	1 bytes
SRCADDR	Pointer on 1 bytes 0: red LED ON 1: red LED OFF

### 5.13.3.6 COM Port

#### 5.13.3.6.1 COM port - overview

The library includes function blocks that enable data exchange between the **BXxxxx** Bus Controller and a remote partner. The maximum COM buffer is 512 bytes for both directions.

#### Function blocks

Name	Description
FB_ComPortOpen [ <a href="#">► 116</a> ]	Opens a serial connection to a partner.
FB_ComPortClose [ <a href="#">► 116</a> ]	Closes a serial connection to a partner.

#### Functions

Name	Description
F_ComPortRead [ <a href="#">► 115</a> ]	Reading data from the COM buffer
F_ComPortWrite [ <a href="#">► 115</a> ]	Writing data into the COM buffer

#### Supported baud rates

Baud rate [baud]	COM 1	COM 2
300	NO	YES
600	NO	YES
1200	NO	YES
2400	NO	YES
4800	NO	YES
9600	YES	YES
19200	YES	YES
38400	YES	YES
57600	YES	YES
115200	NO	YES

Further helpful function blocks can be found in [TcComPortBX.lbx](#) [[► 124](#)].

- Function block for using ComLib, ModbusRTU etc. for the BX COM ports
- Function block for communication with the BK8x00 Bus Couplers
- Function block for emulation of a BK8x00 slave

### 5.13.3.6.2 COM port functions

#### COM Port Read

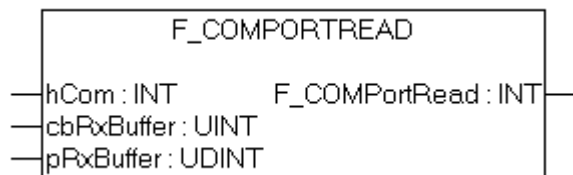


Fig. 100: Function block F\_COMPORTREAD

#### FUNCTION F\_COMPORTREAD

##### VAR\_INPUT

```

hCom      :INT;
cbRxBuffer :UINT;
pRxBuffer :UDINT;
  
```

##### Legend

hCom: is connected with the iHandle of FB\_COMPORTOPEN

cbRxBuffer: maximum length of data that can be read.

pRxBuffer: pointer to data to be written with the COM buffer content

Return value	Meaning
> 0	Number of bytes that is to be copied from the COM buffer into the PLC.
0x8000	Memory overflow

#### COM Port Write

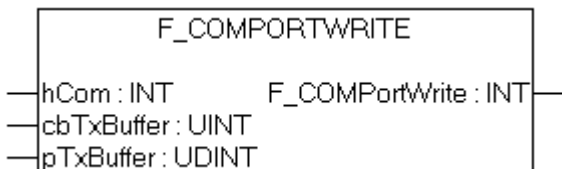


Fig. 101: Function block F\_COMPORTWRITE

#### FUNCTION F\_COMPORTWRITE

##### VAR\_INPUT

```

hCom      :INT;
cbTxBuffer :UINT;
pTxBuffer :UDINT;
  
```

##### Legend

hCom: is connected with the iHandle of FB\_COMPORTOPEN

cbTxBuffer: Number of data bytes that were copied into the COM buffer.

pTxBuffer: Pointer to the data from which the COM buffer is to be filled.

Return value	Meaning
> 0	Number of bytes that is to be copied into the COM buffer from the PLC
0x8000	Memory overflow

### 5.13.3.6.3 COM port function block

#### COM Port Open

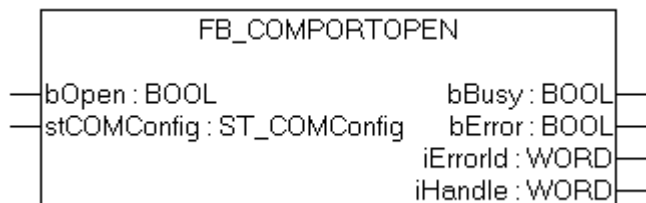


Fig. 102: Function block FB\_COMPORTOPEN

#### FUNCTION\_BLOCK FB\_COMPORTOPEN

##### VAR\_INPUT

```
bOpen      :BOOL;
stComConfig :ST_COMCONFIG;
```

##### Legend

bOpen: rising edge starts the function block  
 stComConfig [► 117]: COM interface data structure

##### VAR\_OUTPUT

```
bBusy      :BOOL;
bErr       :BOOL;
iErrId     :WORD;
iHandle    :WORD;
```

##### Legend

bBusy: The function block is active as long it is TRUE.  
 bErr: Error bit.  
 iErrId: Error number.  
 iHandle: pointer transfer.

Return parameter iErrId	Meaning
0	No error
-1, 0xFFFF	Incorrect COM port
-2, 0xFFFE	Incorrect or unsupported baud rate. Check the parameter stComConfig.BaudRate.
-3, 0xFFFD	Incorrect data format. Check the parameter stComConfig.
-4, 0xFFFC	Incorrect initialization of the COM interface
-5, 0xFFFB	Unsupported instance
-6, 0xFFFA	Incorrect size of the RX buffer
-7, 0xFFF9	Incorrect size of the TX buffer
-8, 0xFFF8	COM port is blocked

## COM Port Close

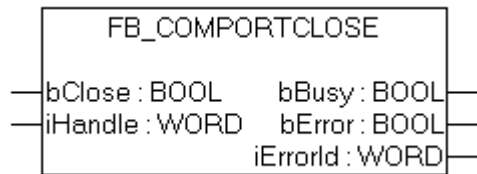


Fig. 103: Function block FB\_COMPORTCLOSE

## FUNCTION\_BLOCK FB\_COMPORTCLOSE

## VAR\_INPUT

```

bOpen      :BOOL;
iHandle     :WORD;

```

## Legend

bClose: rising edge starts the function block  
iHandle: pointer transfer of FB\_COMPORTOPEN.

## VAR\_OUTPUT

```

bBusy      :BOOL;
bErr       :BOOL;
iErrId     :WORD;

```

## Legend

bBusy: The function block is active as long it is TRUE.  
bErr: Error bit.  
iErrId: Error number.

Return parameter iErrId	Meaning
0	No error
> 0	error number (#not documented#)

## 5.13.3.6.4 ComConfig data structure

The settings for the serial interfaces of the BX are transferred with the following data structure.

```

TYPE ST_COMConfig:
STRUCT
    cbRxBufferLen    :WORD;
    cbTxBufferLen    :WORD;
    dwMode            :DWORD;
    BaudRate          :DWORD;
    eCommPort         :E_CommPort;
    eDataBits         :E_DataBits;
    eParity           :E_Parity;
    eStoppBits        :E_StoppBits;
END_STRUCT
END_TYPE

```

## Legend

cbRxBufferLen: Has no purpose (was retained for compatibility reasons)  
cbTxBufferLen: Has no purpose (was retained for compatibility reasons)  
dwMode: data mode COM 1 only "0" - COM 2 RS232 "0" and RS485 "1"  
BaudRate: Baud rate  
eCommPort: Com Port COM1/COM2  
eDataBits: number of data bits SEVEN\_DATABITS/EIGHT\_DATABITS  
eParity: EVEN/ODD/NONE  
eStoppBits: Number of stop bits ONE\_STOPPBIT/TWO\_STOPPBITS

### 5.13.3.6.5 Example

#### ST sample program



Download (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207320715.prx>)

```
PROGRAM MAIN
VAR
(* EXAMPLE - BRIDGE between PIN 7 and 8 from X01 COM 2 Port*)
fb_COMPortOpen_1          : FB_COMPortOpen;
stCOMConfig_1             : ST_COMConfig;
hCOM                      : WORD;
Result_R                  : INT;
Result_W                  : INT;
Var_M                     : ARRAY[0..9] OF BYTE:=11,22,0,33,0(6);
Var_R                     : ARRAY[0..9] OF BYTE;
Value                     : INT;
Counter_V                 : BYTE; (* It is all OK, this value counts up *)
i                          : INT;
i_k                       : INT;
fbTimer                   : TON;
END_VAR
```

```
stCOMConfig_1.cbRxBufferLen :=300;
stCOMConfig_1.cbTxBufferLen :=300;
stCOMConfig_1.dwMode :=0;
stCOMConfig_1.BaudRate :=19200;
stCOMConfig_1.eCommPort :=COM2;
stCOMConfig_1.eDataBits:=EIGHT_DATABITS;
stCOMConfig_1.eParity:=EVEN;
stCOMConfig_1.eStoppBits:=ONE_STOPPBIT;

CASE i OF
(* Open Port *)
0: fb_COMPortOpen_1(bOpen:=TRUE , stCOMConfig:=stCOMConfig_1);
   IF NOT fb_COMPortOpen_1.bBusy THEN
     IF NOTfb_COMPortOpen_1.bError THEN
       hCOM:=fb_COMPortOpen_1.iHandle ;
       i:=i+1;
     ELSE
i:=100;
       END_IF
     END_IF
(* Write data*)
1: fbTimer(IN:=FALSE);
   Result_W:=F_COMPortWrite(hCom, 4,ADR(Var_M[0]));
   IF Result_W>0 THEN
     i:=i+1;
     Var_M[2]:=Var_M[2]+1;
   ELSE
     i:=101;
   END_IF
(*Receive data*)
2: Result_R:=F_COMPortRead(hCom, 100,ADR(Var_R[Value]));
   IF Result_R<>0 THEN
     Value:=Result_R+Value;
   END_IF
   IF Value>=4 THEN
     FOR i_k:=0 TO Value DO(*Check protocol*)
       IF Var_R[i_k-4]=11 AND Var_R[i_k-3]=22 AND Var_R[i_k-1]=33 THEN
         Counter_V:=Var_R[i_k-2];
         i:=1;
         Value:=0;
       END_IF
     END_FOR
   END_IF
   fbTimer(IN:=TRUE,PT:=t#1s); (*Watchdog receive*)
   IF fbTimer.Q THEN
     fbTimer(IN:=FALSE);
     i:=102;
   END_IF
100: ; (*ERROR open port*)
```

```

101: ; (*ERROR send data*)
102: i:=1; (*WD ERROR no data receive*)
END_CASE

```

### 5.13.3.7 BX debugging function

These functions can be used for measuring command execution times in a PLC project. The unit is a tick. One tick corresponds to 5.12 µs.

#### Start Debug Timer function

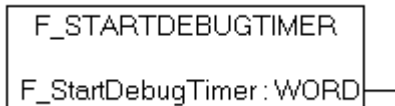


Fig. 104: Function block F\_STARTDEBUGTIMER

Calling this function starts the timer. The return value is "0".

#### Read Debug Timer function

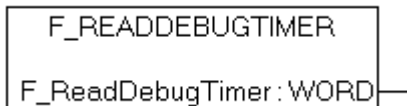


Fig. 105: Function block F\_READDEBUGTIMER

This function reads the timer value. The return value has to be multiplied with 5.12 µs.

#### Example

```

VAR
    Timer_BX      :WORD;
    i              :INT;
END_VAR

```

#### Program

```

F_STARTDEBUGTIMER();
For i:=0 to 1000 do
;
END_FOR
Timer_BX:=F_READDEBUGTIMER();

```

### 5.13.3.8 Navigation switch

#### 5.13.3.8.1 FUN GetNavSwitch

This function block enables reading of the navigation switch.

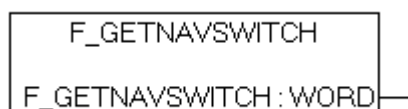


Fig. 106: Function block F\_GETNAVSWITCH

#### VAR\_Output

```

F_GETNAVSWITCH      :WORD;

```

**Legend**

F\_GETNAVSWITCH: Switch data

**WORD description**

Bit	15	14	...	5	4	3	2	1	0
Name	LOCKED	-	...	-	PRESS	RIGHT	LEFT	DOWN	UP

If bit 15 is set, you are in the BX controller menu. This bit is set for as long as the user remains in the BX3100 menu. On exit of the menu, the navigation switch is immediately released for the PLC, i.e. pressing of the Press button is still visible in the program. Please take account of this in your application. For example, the switch should only be evaluated after a short delay by starting a timer with falling edge from bit 15.



Download sample ST program <http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207322891.prx>

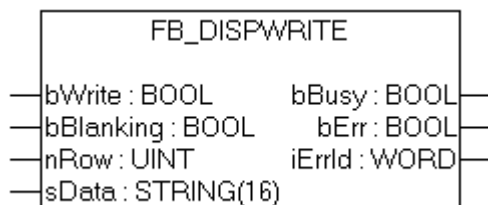
**5.13.3.9 Display****5.13.3.9.1 FB DISPWRITE**

Fig. 107: Function block FB\_DISPWRITE

**VAR\_INPUT**

```

bWrite      :BOOL;
bBlanking   :BOOL;
nRow        :UINT;
sData       :STRING(16)
  
```

**Legend**

bWrite: rising edge starts the function block

bBlanking: FALSE backlight on, TRUE backlight off, default is on (supported in all BX controllers from FW 1.15).

nRow: row in display 1 or 2.

sData: displayed character string

**VAR\_OUTPUT**

```

bBusy       :BOOL;
bErr        :BOOL;
iErrId      :WORD;
  
```

**Legend**

bBusy: The function block is active as long it is TRUE.

bErr: Error bit.

iErrId: Error number.



Return parameter	Meaning
0	No error
> 0	Error number

## ST sample program



Download <http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207325067.prx>

```

PROGRAM MAIN
VAR
    fb_DispWritel: FB_DispWrite;

i:
    udiCounter:    UDINT;
    strCounter:    STRING;
    strLine:       STRING;
    k:             INT;
END_VAR

CASE i OF
0:  strCounter:=CONCAT('Counter :',UDINT_TO_STRING(udiCounter));
    fb_DispWritel(bWrite:=TRUE , nRow:=1 , sData:=strCounter );
    IF NOT fb_DispWritel.bBusy THEN
        IF NOTfb_DispWritel.bErr THEN
            fb_DispWritel(bWrite:=FALSE);
            udiCounter:=udiCounter+1;
            i:=1;
        END_IF
    END_IF
1:  fb_DispWritel(bWrite:=TRUE , nRow:=2 , sData:=strLine);
    IF NOT fb_DispWritel.bBusy THEN
        IF NOTfb_DispWritel.bErr THEN
            fb_DispWritel(bWrite:=FALSE);
            k:=k+1;
            strLine:=REPLACE(' ','#',1,k);
            IF k=16 THEN
                k:=0;
            END_IF
            i:=0;
        END_IF
    END_IF
END_CASE

```

## Display of ASCII table

Example for the "&" sign (see row 1 column 7):  $00100110_{\text{bin}} = 38_{\text{dec}} = 26_{\text{hex}}$ . In the PLC values this corresponds to '\$26' (string.)

<http://infosys.beckhoff.com/content/1033/bx9000/Resources/pdf/3207327243.pdf>

## 5.13.4 TcSystemBX

### 5.13.4.1 Real-time clock - example

The BX Controller features a real-time clock. The current time can be read via a function block. The following example will illustrate this.

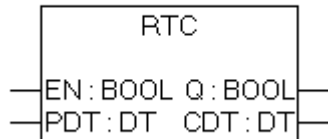


Fig. 108: Function block RTC

**FUNCTION\_BLOCK RTC****VAR\_INPUT**

```

EN      : BOOL;
PDT     : DT;

```

**Legend**

EN: Rising edge sets the time to the value available at the PDT input.

PDT: Date and time to be set.

**VAR\_OUTPUT**

```

Q       : BOOL;
CDT     : BOOL;

```

**Legend**

CDT: Current time.

Required libraries:

- TcSystemBX.lb6
- TcBaseBX.lb6



Download sample ST program (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207329419.prx>)

```

PROGRAM MAIN
VAR
    fbTimer: TON;
    fbRTC: RTC;
END_VAR

```

```
fbTimer (PT:=t#60s, IN:=NOT fbTimer.Q);
```

```

IF fbTimer.Q THEN
    fbRTC;
END_IF

```

**Note****Do not call the RTC function block in every PLC cycle**

Calling the RTC block increases the cycle time by approx. 5 ms, due to the data conversion into a TIME AND DATE variable. The function block should therefore not be called during each PLC cycle.

Alternatively, you can read the time via an [ADS function block](#). The ADS function block returns the date and the time as WORD variables.

Example 19:30 hrs - hour: 19 / minute: 30

### 5.13.4.2 Loading and storing of recipes

The function block fb\_ReadWriteFile enables data (up to 16,000 bytes) to be stored permanently in the flash memory of the BX controller. A new program or a project reset does not affect the content of this memory. This function block is not suitable for sustained and continuous use. A maximum of 10000 write cycles are permitted. There is no limit on read operations.

Application: Saving of recipes or settings that only change rarely or not at all, for example controller parameters.



#### Note

#### Note the following during writing of data

- The voltage must not be interrupted during writing. It is therefore advisable to initiate writing via an operating panel or the navigation keys or simply via a digital input, in order to ensure that the BX Controller is not switched off during writing. Automatic writing is not recommended, since uninterrupted voltage supply during writing cannot be guaranteed.
- Writing of data takes approx. two seconds, irrespective of the number of data that are written.
- The data are lost if the BX controller is switched off during the write operation.
- Only one instance of this function block is permitted.

#### Function block fb\_ReadWriteFile

Function block for reading and writing of recipes

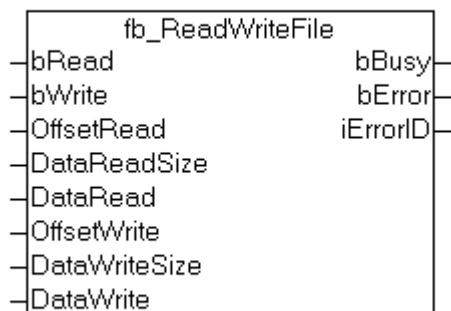


Fig. 109: Function block fb\_ReadWriteFile

#### VAR\_INPUT

```

bRead      :BOOL;
bWrite     :BOOL;
OffsetRead :WORD;
DataReadSize :WORD;
DataRead   :Pointer to Byte;
OffsetWrite :WORD;
DataWriteSize :WORD;
DataWrite  :Pointer to Byte;

```

#### Legend

*bRead*: A rising edge triggers reading of the function block (bWrite must be FALSE)

*bWrite*: A rising edge triggers writing of the function block (bRead must be FALSE)

*OffsetRead*: Offset in the memory 16,000 bytes max.

*DataReadSize*: Size of data to be read in bytes (16,000 bytes max.)

*DataRead*: The pointer should be pointed to the data via ADR

*OffsetWrite*: Offset in the memory 16,000 bytes max.

*DataWriteSize*: Size of data to be written in bytes (16,000 bytes max.)

*DataWrite*: The pointer should be pointed to the data via ADR

## VAR\_OUTPUT

```
bBusy      :BOOL;
bError     :BOOL;
bErrorId   :UDINT
```

## Legend

*bBusy*: Indicates that the function block is still active


*bError*: Function block error

*bErrorId*: Error number

Return parameter iErrorId	Meaning
0	No error
1 <sub>dec</sub>	READ: Data offset and data length more than 16,000 bytes
2 <sub>dec</sub>	WRITE: Data offset and data length more than 16,000 bytes
0x31440708	CRC error in the data memory
0x31470708	Writing of data is not yet complete

Required libraries:

- TcSystemBX.lb6
- TcBaseBX.lb6

 Download sample ST program (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207331595.prx>)

## 5.13.5 TcComPortBX

### 5.13.5.1 TcComPortBX overview

Required libraries:

- TcBaseBX
- TcSystemBX

## Overview

Name	Description
fb BX BK8x00 Master <a href="#">▶ 125</a>	BK8x00 COM port function block, communication with Bus Coupler BK8x00 or BC8x00
fb BX BK8x00 Slave <a href="#">▶ 125</a>	BK8x00 COM port function block, communication with PC. A BK8x00 is simulated.

Name	Description
FB_BX_COM_5 [► 129]	Function block for emulating a KL60x1 (if COMlib.lb6 or ModbusRTU.lb6 is used).
FB_BX_COM_64 [► 129]	Function block for emulating a PC interface (if COMlib.lib or ModbusRTU.lib is used).
FB_BX_COM_64ex [► 130]	Function block for emulating a PC interface (if COMlib.lib or ModbusRTU.lib is used). Here the COM interface can be closed during operation.

### 5.13.5.2 BK8x00 - FB COM-Port

This function block can be used to connect (via the serial interface of the BXxxxx) the BK8000 serial Bus Coupler with RS485 and BK8100 with RS232 connection. The maximum baud rate is 38400 baud.

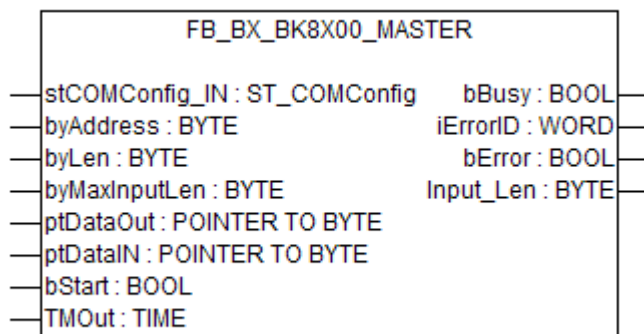


Fig. 110: Function block FB\_BX\_BK8X00\_master

#### VAR\_INPUT

```

stCOMConfig      :ST_COMConfig;
byAddress         :BYTE;
byLen             :BYTE;
byMaxInputLen     :BYTE;
ptDataOut         :POINTER TO BYTE;
ptDataIN          :POINTER TO BYTE;
bStart            :BOOL;
TmOut             :TIME;

```

#### Legend

stComConfig: Structure for selecting the COM parameters  
 byAddress: BX8x00 address 1-98 (0 and 99 are reserved)  
 byLen: data length in [BYTES] (only even numbers are permitted, i.e. 0, 2, 4, ...)  
 byMaxInputLen: is connected with SIZEOF and the variable that is linked with ptDataIN  
 ptDataOut: is connected with ADR and data out  
 ptDataIn: is connected with ADR and data in  
 bStart: rising edge starts the function block  
 TMOut: delay until process is aborted

#### VAR\_OUTPUT

```

bBusy            :BOOL;
bError           :BOOL;
iErrorId         :WORD;
Input_len        :WORD;

```

#### Legend

bBusy: The function block is active as long it is TRUE.  
 bError: error bit  
 iErrorID: Error number  
 Input\_Len: number of data that were received

Return parameter iErrorId	Meaning
0	No error
100 <sub>dec</sub>	Error during opening of the COM port
101 <sub>dec</sub>	Error during sending of data
102 <sub>dec</sub>	Watchdog error, no response from the slave within the WD time
105 <sub>dec</sub>	The input buffer is too small
200 <sub>dec</sub>	CRC error
0x80xx <sub>hex</sub>	Bus Coupler error xx status byte of the Bus Coupler (see BX8x00 documentation)

## Hardware

### RS 232 communication PIN assignment

BX COM1 RS232	BX COM2 RS232	BK8100
2	8	2
3	7	3
5	5	5

### RS485 communication PIN assignment

FB settings: When using the RS485 connection it is important that the stCOMConfig variable is set to 1 and that the COM2 interface is selected.

BX COM2 RS485	BK8000
1	3
6	8

### ST example program



Download (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207333771.prx>)

Required material:

- BX3100 + Bus Terminal
- BK8100, KL1xx8, KL2xx8, KL9010
- Serial cable, PIN assignment: see sample program

### 5.13.5.3 BK8x00 - FB Slave COM-Port

With this function block, the PC (TwinCAT or KS8000) can be connected with the BXxxxx via the serial interface. The PC acts as the serial master, and the BXxxxx emulates a BK8x00 with the aid of the function block.

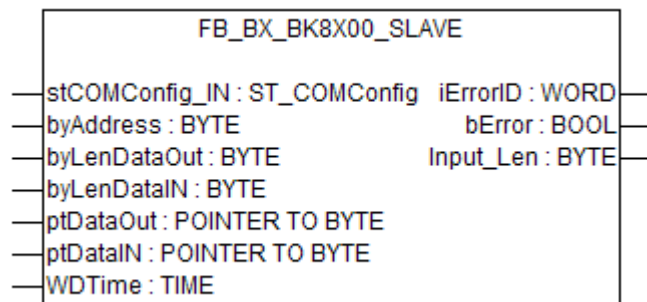


Fig. 111: Function block FB\_BX\_BK8X00\_SLAVE

**VAR\_INPUT**

```

stCOMConfig      :ST_COMConfig;
byAddress         :BYTE;
byLenDataOut      :BYTE;
byLenDataIN       :BYTE;
ptDataOut         :POINTER TO BYTE;
ptDataIN          :POINTER TO BYTE;
WdTime           :TIME;

```

**Legend**

stComConfig: Structure for selecting the COM parameters

byAddress: BX8x00 address 1-98 (0 and 99 are reserved)

byLenDataOut: data length in [BYTES] (only even numbers are permitted, i.e. 0, 2, 4, ...)

byLenDataIn: data length in [BYTES] (only even numbers are permitted, i.e. 0, 2, 4, ...)

ptDataOut: is connected with ADR and data out

ptDataIn: is connected with ADR and data in

WdTime: error message if no new data are received within the watchdog time (0 ms disable WD)

**VAR\_OUTPUT**

```

bError           :BOOL;
iErrorId         :WORD;
Input_Len        :BYTE;

```

**Legend**

**bError:** error bit

**iErrorId:** Error number

**Input\_Len:** number of data that were received

Return parameter iErrorId	Meaning
0	No error
1	Watchdog error, if greater than 0 ms (WD disable if 0 ms)
100 <sub>dec</sub>	Error during opening of the COM port
101 <sub>dec</sub>	Error during sending of data
103 <sub>dec</sub>	Internal receive buffer overflow
104 <sub>dec</sub>	Data exceed the PLC buffer capacity (more than 500 bytes)
105 <sub>dec</sub>	Data cannot be copied into the PLC buffer
200 <sub>dec</sub>	CRC error

## Hardware

### RS232 communication PIN assignment

BX COM 1 RS232	BX COM 2 RS232	PC COM interface
2	8	2
3	7	3
5	5	5

### RS485 communication PIN assignment

FB settings: When using the RS485 connection it is important that the stCOMConfig variable is set to 1 and that the COM2 interface is selected.

BX COM 2 RS485	PC COM port (e.g. RS485 card W&T #13601, 2-wire, without echo, automatic)
1	1 - 2 bridges
6	6 - 7 bridges

### ST sample program for BXxxx

 Download (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207335947.prx>)

System Manager file for TwinCAT as master. As shown in the figure, a Bus Coupler with Bus Terminals is configured. In this case, the type and number of Bus Terminals determines the data length. In principle, the type of Bus Terminal is irrelevant. Sample:

- 2 x KL3002 results in 4 words of input
- 2 x KL4002 results in 4 words of output

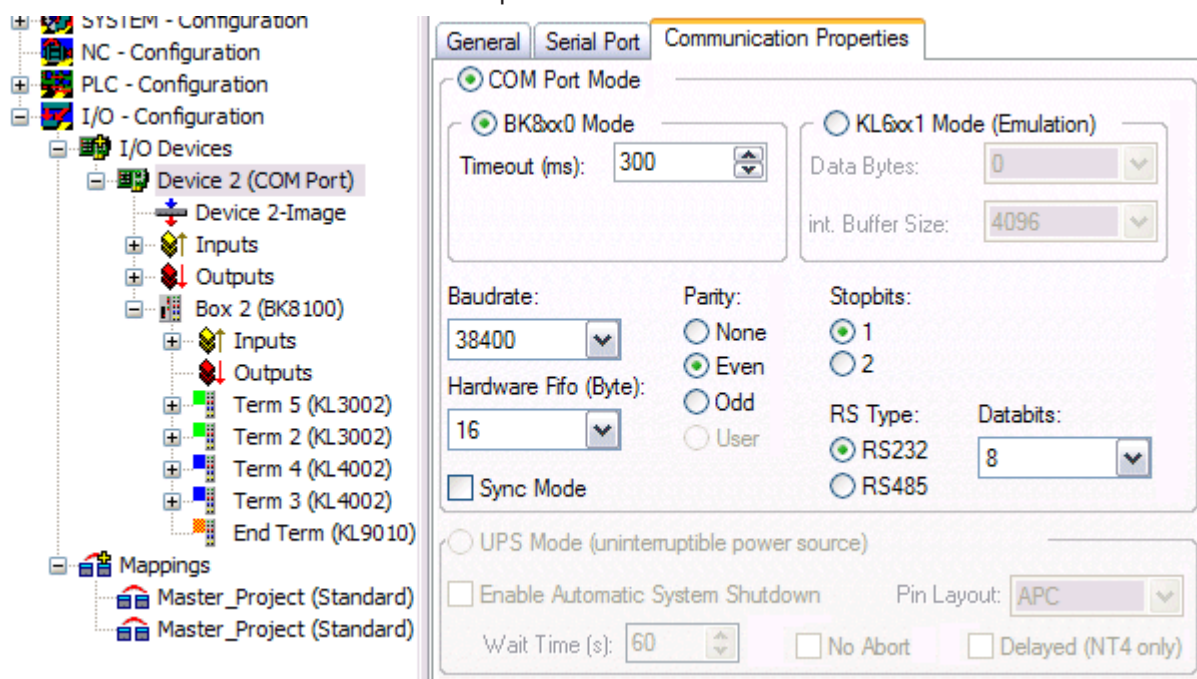


Fig. 112: Communication features

Required material:

- BX3100 + Bus Terminal
- PC with RS232 interface and TwinCAT from version 2.9, serial cable, PIN assignment: see above



#### 5.13.5.4 FB\_BX\_COM\_5

This function block connects ModbusRTU.lb6, ModbusRTU.lib or ComLib.lb6 with the serial interface of the BX Controller. It emulates a KL60x1 - data output is not via a Bus Terminal, but via one of the two serial interfaces of the BX.



Fig. 113: Function block FB\_BX\_COM\_5

##### VAR\_INPUT

```
pstrEmo_IN      :POINTER TO BYTE;
pstrEmo_OUT     :POINTER TO BYTE;
ComConfig       :ST_COMConfig;
```

##### Legend

pstrEmo\_IN: Pointer to KL6inData5B

pstrEmo\_OUT: Pointer to KL6outData5B

ComConfig [► 117]: Parameterization of the COM interface



Download sample program in ST for linking COMLib and BX: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207338123.prx>)

#### 5.13.5.5 FB\_BX\_COM\_64

This function block connects ModbusRTU.lib or ComLib.lib with the serial interface of the BX Controller. It emulates a PC interface. Data output is not via a PC interface, but via one of the two serial interfaces of the BX (COM1 or COM2).



Fig. 114: Function block FB\_BX\_COM\_64

##### VAR\_INPUT

```
pstrEmo_IN      :POINTER TO BYTE;
pstrEmo_OUT     :POINTER TO BYTE;
ComConfig       :ST_COMConfig;
```

##### Legend


pstrEmo\_IN: Pointer to ModbusPCComInData

pstrEmo\_OUT: Pointer to ModbusPCComInData

ComConfig [► 117]: Parameterization of the COM interface



Download sample program in ST for linking ModbusRTU and BX: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207340299.prx>)

 Download sample program in ST for linking ModbusRTU version 2 and BX: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207342475.prx>)

The sample requires the ModbusRTU library!

### 5.13.5.6 FB\_BX\_COM\_64ex

This function block connects ModbusRTU.lib or ComLib.lib with the serial interface of the BX Controller. A PC interface with 64 byte of user data is emulated. Data output is not via a PC interface, but via one of the two serial interfaces of the BX (COM1 or COM2).

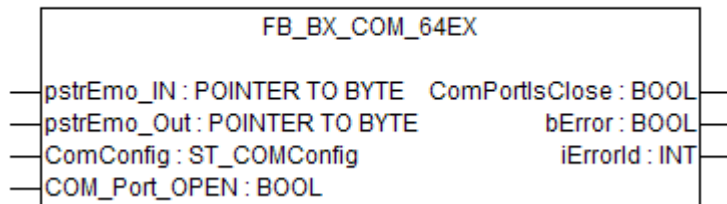


Fig. 115: Function block FB\_BX\_COM\_64EX

#### VAR\_INPUT

```
pstrEmo_IN      :POINTER TO BYTE;
pstrEmo_OUT     :POINTER TO BYTE;
ComConfig       :ST_COMConfig;
```

#### VAR\_OUTPUT

```
ComPortIsClose  :BOOL;
bError          :BOOL;
iErrorId        :INT;
```

#### Legend

##### **pstrEmo\_IN:**

Pointer to ModbusPCComInData

##### **pstrEmo\_OUT:**

Pointer to ModbusPCComOutData

##### ComConfig [► 117]:

Parameterization of the COM interface

##### **COM\_Port\_Open:**

If this bit is set, the interface is opened. If this bit is reset, the interface is closed.

##### **ComPortIsClose:**


If the interface is closed, this bit is set.

##### **bError:**

There is an error.

##### **iErrorId:**

Error code (see FB\_COMPortOpen) [► 116]

 Download sample program in ST for linking ModbusRTU and BX: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207344651.prx>)

The sample requires the ModbusRTU library!

### 5.13.5.7 Further samples

#### 5.13.5.7.1 BX COM port as ModbusRTU master

The serial interface of the BX can also be used as Modbus master.

##### Necessary components

1 x BX3100  
 Bus Terminals for the K-Bus (any, since they are not used for the example)  
 1 x BK7300  
 2 x KL2xx4  
 2 x KL1xx4  
 1 x KL9010

##### RS 485 cable\*

BX3100 COM 2 / RS 485	BK7300 / RS 485
1	3
6	8

\*) active termination resistor required for short cable lengths (< 5 m) and low baud rates (<19200 baud)

 **Download ST sample program for linking the ModbusRTU master and BX:** ( <http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207346827.prx> )

 **Download ST sample program for linking the ModbusRTU master version 2 and BX:** ( <http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207349003.prx> )

The example requires the ModbusRTU, TcComPortBC, TcBaseBX libraries.  
 Baud rate 9600, n, 8.1 default BK7300,  
 BK7300 address 11

##### Reaction times

The reaction times depend on the set task time, the number of slaves, the length of the Modbus telegrams and the response time of the slaves.  
 Beckhoff BK7300 Modbus slaves were used for determining the following table. Since this is not transferable to all slaves, the table should only be used for guidance.

Baud rate 38400 baud (one read reg. and one write reg. telegram per slave)

Number of slaves	Task time on the BX	Time for one cycle
1	5	100 ms* / 125 ms**
2	5	200 ms / 225 ms
1	10	180 ms / 220 ms
2	10	350 ms / 390 ms
1	20	350 ms / 350 ms
2	20	700 ms / 700 ms

\*) 2 words inputs and 2 words outputs

\*) 20 words inputs and 20 words outputs

#### 5.13.5.7.2 BX COM port - ComLibV2

Examples for ComLibV2 sending of strings via the internal COM interface of the BX controller. For receiving a bridge can be established from PIN 7 and 8 to X01 (COM2).


Required material

Hardware:

- BX Controller

Software:

- TwinCAT from 2.10
- COMlibV2.lib
- TcComPortBX.lbx
- Standard.lbx
- TcBase.lbx
- TcSystemBX.lbx

 Download BX sample program: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207351179.prx>)

5.13.5.7.3 BX COM port - Cimrex panel

The serial interface of the BX controller can also be used as Modbus slave. In this example, a panel from the company Beijers is connected. Further information on the panel can be found under <http://www.beijerelectronics.de>.



Fig. 116: Cimrex panel at the COM port of the BX controller

Necessary components

- 1 x BX3100
- 1 x Cimrex 12
- any Bus Terminals (any, since no Bus Terminals are used in the example)


RS232 cable


BX3100 COM 2 / RS485	Cimrex 12 RS232
7	2
8	3
9	5

**RS485\* cable**

<b>BX3100 COM 2 / RS 485</b>	<b>Cimrex 12 RS485</b>
1	2 -3
6	15 -16

\*) active termination resistor is not required for short cable lengths ( $\leq 5$  m) and low baud rates ( $\leq 19200$  baud)

 Download sample program in ST for the BX: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207340299.prx>)

 Download sample with Cimrex panel: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/cpa/3207353355.cpa>)

The example requires the ModbusRTU, TcComPortBC and TcBaseBX libraries.

- Baud rate 9600,n,8,1 D

- Cimrex 12

#### 5.13.5.7.4 BX COM port - RK512 protocol

The RK512 protocol can exchange data with a distant station via the COM1 or COM2 interface of the BX controller. Documentation for the RK512 function block can be found in the Beckhoff Information System. The serial PC interface is simulated via the 64 byte emulation of the BX controller.


#### Required material

Hardware:


- PC with RS232 interface and TwinCAT PLC from 2.9
- BX Controller
- Serial cable for the BX - PC connection

Software:

- TwinCAT from 2.9
- COMlib.lib
- COMlib3964R.lib
- COMlibRK512.lib
- TcComPortBX.lbx
- Standard.lbx
- TcBase.lbx
- TcSystemBX.lbx
- ChrAscBx.lbx

 Download sample program BX3100: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207355531.prx>)

 Download PC sample program: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/pro/3207357707.pro>)

 Download sample System Manager file PC: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/tsm/3207359883.tsm>)

#### 5.13.5.7.5 BX COM port - text message via mobile phone

The serial interface can also be used for sending a text message from the BX controller. The following example uses the SMS library with a Siemens S35 mobile phone.



Fig. 117: Mobile phone at the COM port of the BX controller



Download: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207362059.prx>)

#### Pin assignment (Siemens cable S30880-S4501 A801-2)

S35	COM 1	COM 2
2	3	7
3	2	8
5	5	9

## 5.13.6 TcTwinSAFE

### 5.13.6.1 Overview


The Bus Terminal Controllers of the BX series support the TwinSAFE Bus Terminals when the following conditions are met:

- At the Bus Terminal Controller only one logic terminal is permitted. It must be connected to the K-bus interface, not the SSB.
- At this logic terminal a maximum of seven connections are permitted.
- TwinSAFE-input and output terminals can be connected to the K-bus or the SSB, for example via BK5120 or BK515x.
- If the online change feature is to be used, the connection timeout must be set to 500 ms or greater.
- An ADS connection must exist for downloading the TwinSAFE projects. The connection can be serial or via the fieldbus.
- The firmware version of the Bus Terminal Controller must be 1.17 or higher.

#### TwinSAFE library

The TwinSAFE library includes function blocks for executing services/functions in connection with the TwinSAFE terminals KL1904, KL2904 and KL6904.

Name	Description
F_GetVersionTcTwinSAFE [► 136]	Library version number
FB_TwinSAFE_KLx904_input [► 136]	Evaluation of TwinSAFE data sent from a KL1904 or KL2904 to a KL6904
FB_TwinSAFE_KLx904_output [► 139]	Evaluation of TwinSAFE data sent from a KL6904 to a KL1904 or KL2904

 Download of the TwinSAFE library: (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/lbx/3207364235.lbx>)

### 5.13.6.2 FUNCTION F\_GetVersionTcTwinSAFE

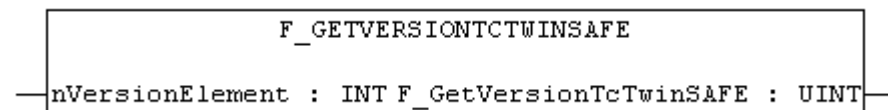


Fig. 118: Function block F\_GETVERSIONTCTWINSAFE

This function can be used to read PLC library version information.

#### FUNCTION F\_GetVersionTcTwinSAFE : UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

**nVersionElement:** Version element to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v2.10.0 Build > 914	PC (i386)	-	TcTwinSAFE.Lib (Standard.lib, TcBase.Lib and TcSystem.Lib are integrated automatically)
TwinCAT v2.10.0	BX series	-	TcTwinSAFE.LBX (Standard.LBX; TcBaseBX.LBX; TcSystemBX.LBX are integrated automatically)

### 5.13.6.3 FUNCTION\_BLOCK FB\_TwinSAFE\_KLx904\_input

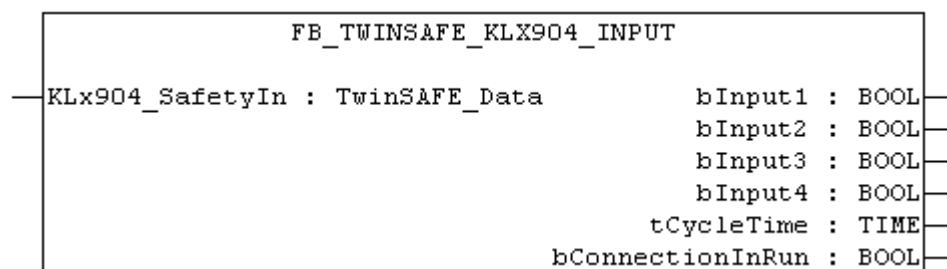


Fig. 119: Function block FB\_TWINSAFE\_KLX904\_INPUT



The function block *FB\_TwinSAFE\_KLx904\_input* can be used for evaluation of TwinSAFE data sent from a KL1904 or KL2904 to a KL6904. The input parameter is doubly linked to the SafetyIn data of a KL1904 or KL2904 in the System Manager.

## VAR\_INPUT

```
VAR_INPUT
    KLx904_SafetyIn AT%I* : TwinSAFE_Data; (* Additional link to "SafetyIn" *)
END_VAR
```

**KLx904\_SafetyIn:** TwinSAFE telegram sent from a KL1904 or KL2904 to a KL6904. This parameter is doubly linked to SafetyIn in the System Manager (input data of the KLx904).

## VAR\_OUTPUT

```
VAR_OUTPUT
    bInput1      : BOOL;
    bInput2      : BOOL;
    bInput3      : BOOL;
    bInput4      : BOOL;
    tCycleTime   : TIME;
    bConnectionInRun : BOOL;
END_VAR
```

**bInput1:** Returns input 1 of a KL1904. If this function block is used for connection to a KL2904, the value is always 0.

**bInput2:** Returns input 2 of a KL1904. If this function block is used for connection to a KL2904, the value is always 0.

**bInput3:** Returns input 3 of a KL1904. If this function block is used for connection to a KL2904, the value is always 0.

**bInput4:** Returns input 4 of a KL1904. If this function block is used for connection to a KL2904, the value is always 0.

**tCycleTime:** Returns the cycle time in ms for exchanging the TwinSAFE telegram between the devices.

**bConnectionInRun:** Returns TRUE if there is no error in the connection between the KLx904 and the KL6904.

## Example of a call in the FBD:

```
PROGRAM MAIN
VAR
    fbTwinSAFE_KLx904_input      : FB_TwinSAFE_KLx904_input;
    bInput1_KL1904_S_Address_113 : BOOL;
    bInput2_KL1904_S_Address_113 : BOOL;
    bInput3_KL1904_S_Address_113 : BOOL;
    bInput4_KL1904_S_Address_113 : BOOL;
    tCycleTime_KL1904_KL6904     : TIME;
    bConnection3_In_Run          : BOOL;
END_VAR
```

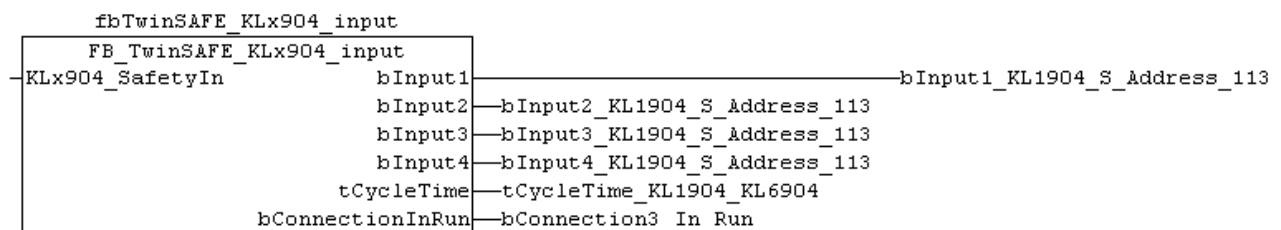


Fig. 120: Function block FB\_TWINSafe\_KLX904\_input

In the example the values of the KL1904 input data are written to the connected variables. If the output **bConnectionInRun** is FALSE, all outputs are set to FALSE.

To link the input data, select the parameter **KLx904\_SafetyIn** and select "Modify link..." from the context menu.

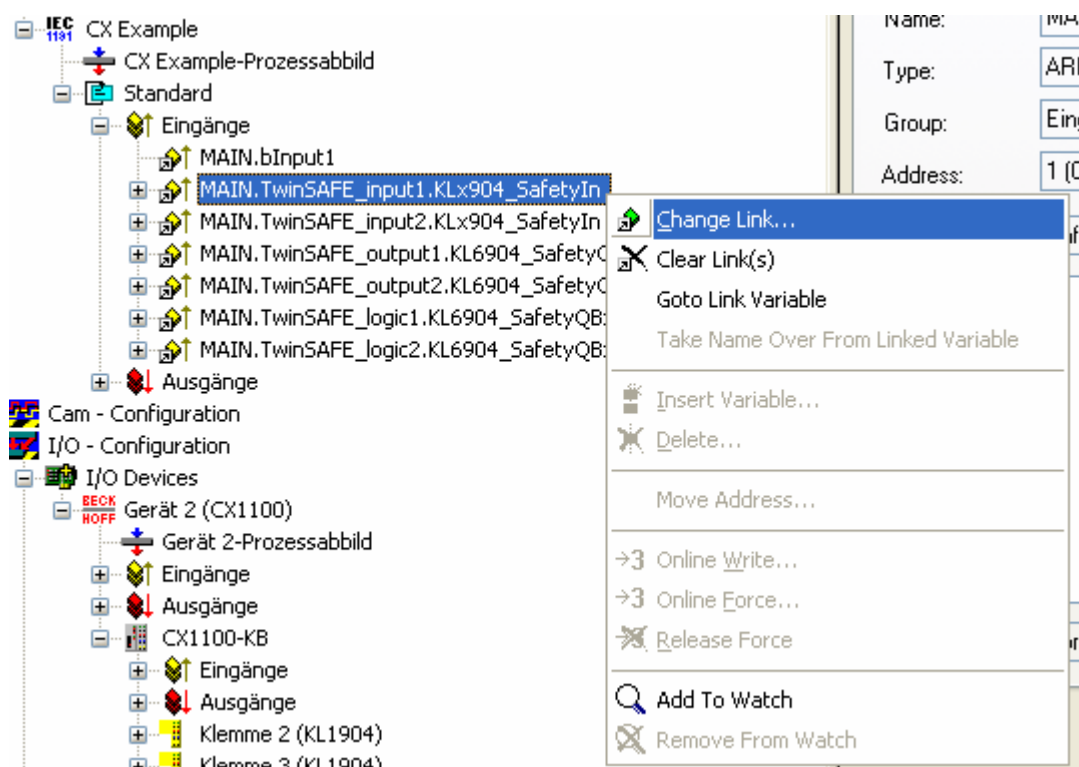


Fig. 121: Linking the input data

and select the corresponding SafetyIn variable in the dialog that follows.

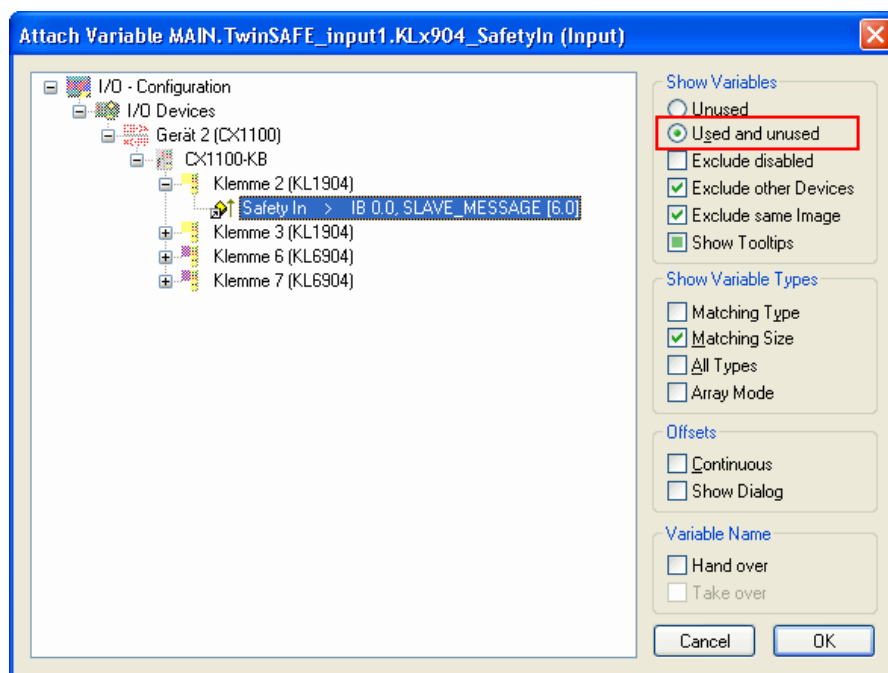


Fig. 122: Selecting the SafetyIn variable

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v2.10.0 Build > 914	PC (i386)	KLx904	TcTwinSAFE.Lib (Standard.lib, TcBase.Lib and TcSystem.Lib are integrated automatically)
TwinCAT v2.10.0 Build > 914	BX series	KLx904	TcTwinSAFE.LBX (Standard.LBX, TcBaseBX.LBX and TcSystemBX.LBX are integrated automatically)

### 5.13.6.4 FUNCTION\_BLOCK FB\_TwinSAFE\_KLx904\_output

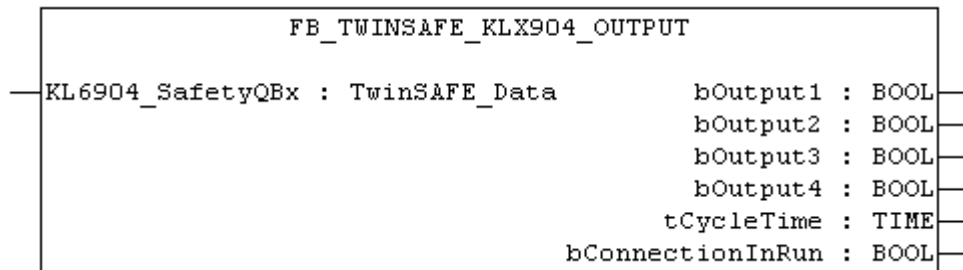


Fig. 123: Function block FB\_TWINSAFE\_KLX904\_output

The function block *FB\_TwinSAFE\_KLx904\_output* can be used for evaluation of TwinSAFE data sent from a KL6904 to a KL1904 or KL2904. The input parameter is doubly linked to the SafetyQBx data of a KL6904 in the System Manager.

#### VAR\_INPUT

```

VAR_INPUT
    KL6904_SafetyQBx AT%I* : TwinSAFE_Data; (* Additional link to "SafetyQBx" *)
END_VAR

```

**KL6904\_SafetyQBx:** TwinSAFE telegram sent from a KL6904 to a KL1904 or KL2904. This parameter is doubly linked to SafetyQBx in the System Manager (input data of the KL6904); x represents for numerals between 1 and 15, according to the TwinSAFE connection used.

#### VAR\_OUTPUT

```

VAR_OUTPUT
    bOutput1      : BOOL;
    bOutput2      : BOOL;
    bOutput3      : BOOL;
    bOutput4      : BOOL;
    tCycleTime    : TIME;
    bConnectionInRun : BOOL;
END_VAR

```

**bOutput1:** Returns output 1 of a KL2904. If the function block is used for a connection to the KL1904, this value is always 0.

**bOutput2:** Returns output 2 of a KL2904. If the function block is used for a connection to the KL1904, this value is always 0.

**bOutput3:** Returns output 3 of a KL2904. If the function block is used for a connection to the KL1904, this value is always 0.

**bOutput4:** Returns output 4 of a KL2904. If the function block is used for a connection to the KL1904, this value is always 0.

**tCycleTime:** Returns the cycle time in ms for exchanging the TwinSAFE telegram between the devices.

**bConnectionInRun:** Returns TRUE if there is no error in the connection between the KL6904 and the KLx904.

#### Example of a call in the FBD

```

PROGRAM MAIN
VAR
    fbTwinSAFE_KLx904_output      : FB_TwinSAFE_KLx904_output;
    bOutput1_KL6904_Connection_to_113 : BOOL;
    bOutput2_KL6904_Connection_to_113 : BOOL;
    bOutput3_KL6904_Connection_to_113 : BOOL;
    bOutput4_KL6904_Connection_to_113 : BOOL;
    tCycleTime_KL6904_KL1904        : TIME;
    bConnection3_In_Run_2            : BOOL;
END_VAR

```

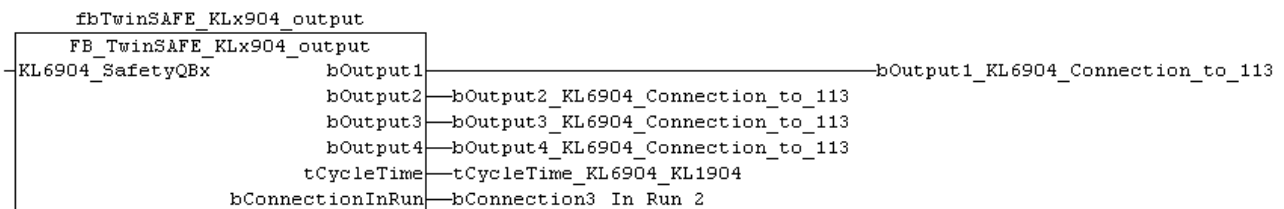


Fig. 124: Call of function block FB\_TWINSAFE\_KLX904\_OUTPUT

In the example the values of TwinSAFE terminals KL6904 and KL1904 are evaluated. Since no output signals are used in this connection, the outputs are always FALSE. Only tCycleTime and bConnectionInRun can be evaluated.

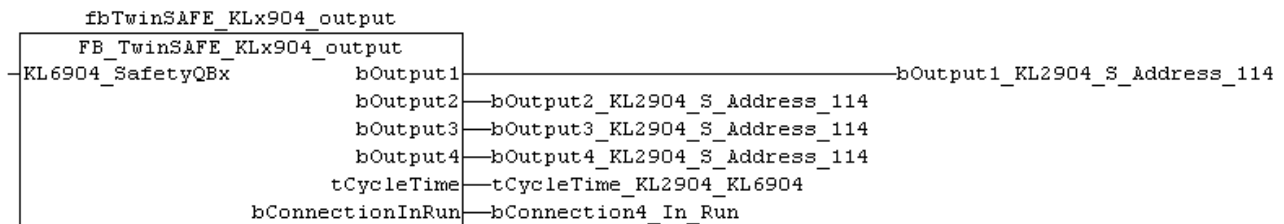


Fig. 125: Call of function block FB\_TWINSAFE\_KLX904\_OUTPUT

In the example the values of TwinSAFE terminals KL6904 and KL1904 are evaluated. In this connection the output signals are written to the KL2904 and copied from the function block to the connected variables. If the output bConnectionInRun is FALSE, all outputs are set to FALSE.

To link the input data, select the parameter KL6904\_SafetyQBx and select "Modify link..." from the context menu.

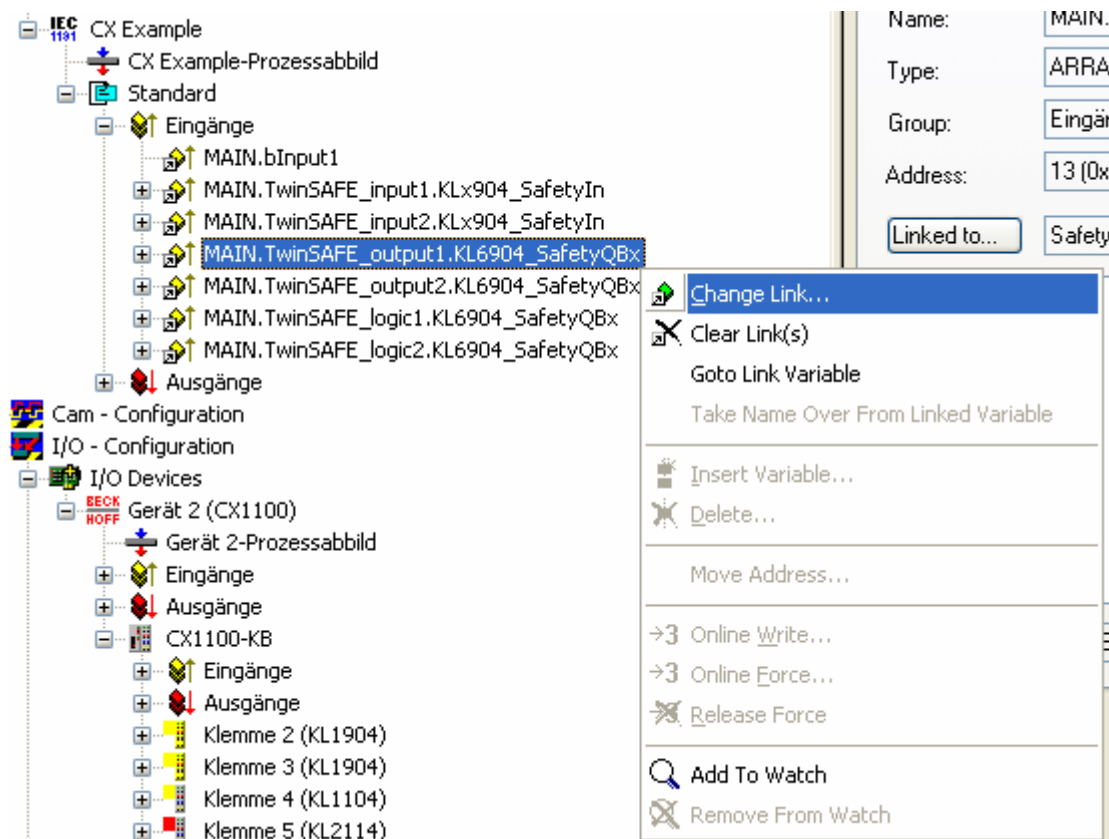


Fig. 126: Linking the input data

and select the corresponding SafetyQBx variable in the dialog that follows.

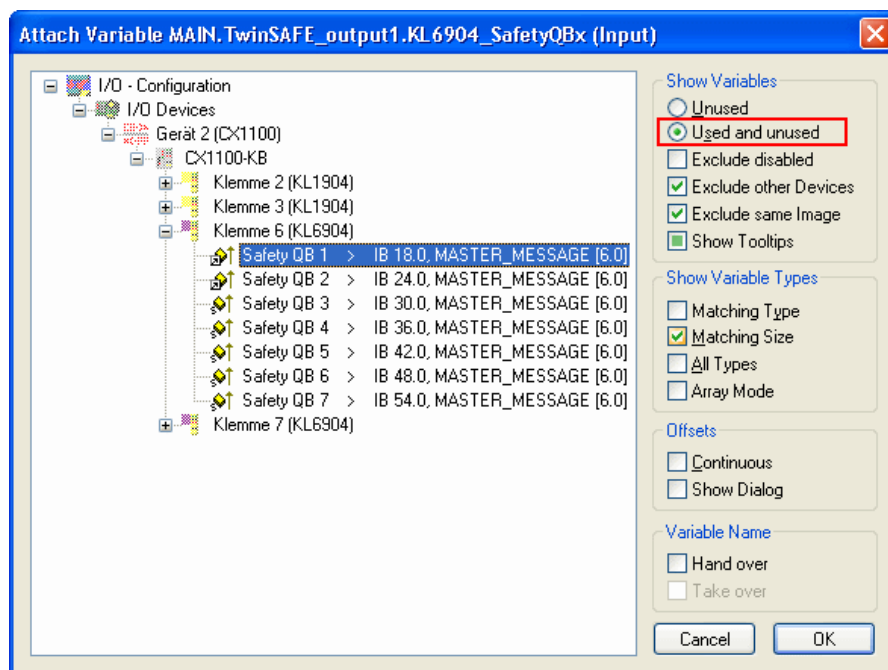


Fig. 127: Selecting the corresponding SafetyQBx variable

Development environment	Target platform	IO Hardware	PLC libraries to include
TwinCAT v2.10.0 Build > 914	PC (i386)	KLx904	TcTwinSAFE.Lib (Standard.lib, TcBase.Lib and TcSystem.Lib are integrated automatically)
TwinCAT v2.10.0 Build > 914	BX series	KLx904	TcTwinSAFE.LBX (Standard.LBX, TcBaseBX.LBX and TcSystemBX.LBX are integrated automatically)

## 5.13.7 TcBaseBX9000

### 5.13.7.1 Overview: TcBaseBX9000

#### Download

To download the libraries click on the link. Please copy the libraries to directory TwinCAT\PLC\LIB.

- Download TcBaseBX9000 (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207318539.zip>)



#### TcBaseBX9000

IP-TCP/IP-UDP	Version	Firmware
		BX9000
<a href="#">FB_IpClose [► 146]</a>	12.10.06	1.15
<a href="#">FB_IpEndSession [► 144]</a>	12.10.06	1.15
<a href="#">FB_IpOpen [► 145]</a>	12.10.06	1.15
<a href="#">FB_IpReceive [► 147]</a>	12.10.06	1.15
<a href="#">FB_IpSend [► 148]</a>	12.10.06	1.15
<a href="#">FB_IpStartSession [► 143]</a>	12.10.06	1.15

ModbusTCP	Version	Firmware
		BX9000
<a href="#">FB_MBClose [► 155]</a>	12.10.06	1.15
<a href="#">FB_MBConnect [► 155]</a>	12.10.06	1.15
<a href="#">FB_MBGenericReq [► 155]</a>	12.10.06	1.15

Services	Version	Firmware
		BX9000
<a href="#">FB_AddDnsServer [► 161]</a>	12.10.06	1.15
<a href="#">FB_GetHostByAddr [► 162]</a>	12.10.06	1.15
<a href="#">FB_GetHostByName [► 163]</a>	12.10.06	1.15
<a href="#">FB_GetNetworkConfig [► 164]</a>	12.10.06	1.15
<a href="#">FB_SetTargetName [► 165]</a>	12.10.06	1.15

SMTP	Version	Firmware
		BX9000
<a href="#">FB_Smtp [► 158]</a>	12.10.06	1.15

SNTP	Version	Firmware
		BX9000
<a href="#">FB_Sntp [► 160]</a>	12.10.06	1.15

## 5.13.7.2 Socket Interface

### 5.13.7.2.1 Overview: Socket Interface

The socket interface can be used for receiving any Ethernet telegrams or for sending telegrams from the controller. In this way any PLC layer protocols can be programmed in IEC 61131-3.

#### Supported Ethernet protocols

Type	STREAM	DGRAM	RAW
IP	n.i.	n.i.	n.i.
ICMP	n.i.	n.i.	n.i.
IGMP	n.i.	n.i.	n.i.
TCP	Implemented*	n.i.	n.i.
UDP	n.i.	Implemented*	n.i.
RAW	n.i.	n.i.	n.i.

Table 1

n.i. not implemented

#### Operating principle of a socket-connection

Before a socket can be opened resources for this type of connection must be made available to the controller. This is done by starting a session. The scope is specified here. The session can then be used for sending or receiving Ethernet telegrams. The implemented protocols are listed in Table 1.

## Client-server relationship

A client is defined as a device that intends to establish a connection and initializes the active part of a connection setup. The server is initially passive and awaits a client query. The server becomes active once a client establishes a connection. Once a connection between client and server has been established, both devices can send or receive data.

\* to Tab1

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BC9050 (165) firmware version >=B0	TcBaseBX9000.lbx
TwinCAT v2.10.0 and above	BC9020 (165) firmware version >=B0	TcBaseBX9000.lbx
TwinCAT v2.10.0 and above	BC9120 (165) firmware version >=B1	TcBaseBX9000.lbx
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.14	TcBaseBX9000.lbx

### 5.13.7.2.2 IP block overview

IP-TCP/IP-UDP	Firmware	Description
<a href="#">FB_IpStartSession [► 143]</a>	1.14	Opening a session
<a href="#">FB_IpEndSession [► 144]</a>	1.14	Closing a session
<a href="#">FB_IpOpen [► 145]</a>	1.14	Opening a TCP/IP connection (not required for UDP communication)
<a href="#">FB_IpClose [► 146]</a>	1.14	Closing a TCP/IP connection (not required for UDP communication)
<a href="#">FB_IpReceive [► 147]</a>	1.14	Receiving TCP or UDP telegrams
<a href="#">FB_IpSend [► 148]</a>	1.14	Sending TCP or UDP telegrams

Multicast function blocks	Firmware	Description
<a href="#">FB_AddMultiRoute [► 160]</a>	1.14	Creating a multicast address
<a href="#">FB_DelMultiRoute [► 160]</a>	1.14	Deleting a multicast address

#### 5.13.7.2.2.1 FB\_IpStartSession

The function block allocates resources on the controller for the Ethernet communication. The function block becomes active with *bStart*. **bBusy** is set as long as the function block is busy. **iDevice** should always be zero. **iPort** is used for the local TCP or UDP port number. **iMaxConnection** indicates the maximum number of possible connections (up to 3).

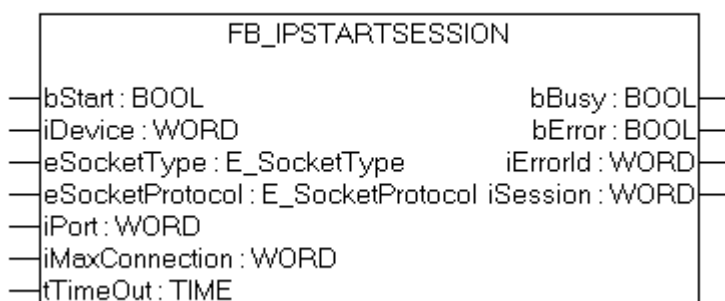


Fig. 128: Function block FB\_IPSTARTSESSION

**INPUT**

```

VAR_INPUT
    bStart      : BOOL;
    iDevice     : WORD;
    eSocketType : E_SocketType;
    eSocketProtocol : E_SocketProtocol;
    iPort       : WORD;
    iMaxConnection : WORD;
    tTimeout    : TIME;
END_VAR

```

**bStart:** A rising edge activates the function block.

**iDevice:** always "0"

**eSocketType:** For TCP/IP "SOCK\_STREAM" should be used. Please note that the data are stored as a byte stream. If possible, the length of the received data should be known or a protocol with start and end ID should be used, so that the start and end can be detected unambiguously in the data stream. For UDP/IP "SOCK\_DGRAM" should be set. A UDP frame always stored in the memory as a complete entity. 4 memories are available. If the data are not read fast enough from the memory of the PLC by the user program further UDP frames are lost.

**eSocketProtocol:** For TCP/IP "IPPROTO\_TCP" should be used, for UDP/IP "IPPROTO\_UDP".

**iPort:** Sender port number

**iMaxConnection:** Number of possible connections (max. 3)

**tTimeout:** Time after which the attempt is aborted.

**OUTPUT**

```

VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
    iSession   : WORD;
END_VAR

```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the last executed command (see table).

**iSession:** Forwards the session number to all IP function blocks for which this connection was created.

**5.13.7.2.2.2 FB\_IpEndSession**

The function block closes an open session. A positive edge of *bStart* closes the session and releases resources.

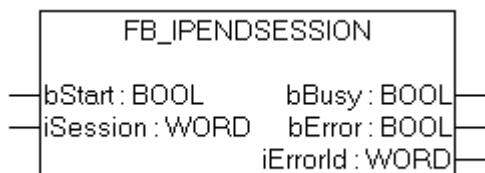


Fig. 129: Function block FB\_IPENDSESSION

**INPUT**

```

VAR_INPUT
    bStart      : BOOL;
    iSession    : WORD;
END_VAR

```



**bStart:** A rising edge activates the function block.

**iSession:** Is linked with the session number from function block FB\_IpStartSession.

## OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
END_VAR
```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

### 5.13.7.2.2.3 FB\_IpOpen

The function block is required for opening a TCP/IP connection from the PLC controller. In this case is the controller is the client that actively establishes a connection to a TCP/IP server. With a positive edge of **bStart** a connection to a server with the IP address from **sRemoteIPAddr** is established. The recipient port number was specified when the session was started (see [FB\\_IpStartSession \[► 143\]](#)). The sender port number is issued by the controller and can be read from **iPortNo**. The sender port number is required for sending (see [FB\\_IpSend \[► 148\]](#)). **bBusy** is set as long set as the function block is active. If **bBusy** is reset and **bError** is FALSE the TCP/IP connection was terminated successfully and data can be sent or received.

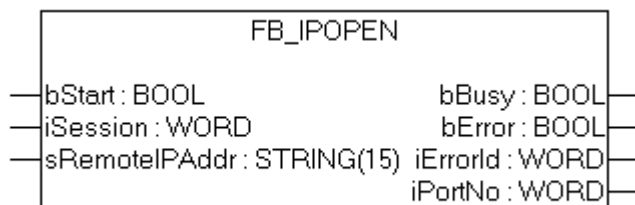


Fig. 130: Function block FB\_IPOPEN

## INPUT

```
VAR_INPUT
    bStart      : BOOL;
    iSession    : WORD;
    sRemoteIPAddr : STRING(15);
END_VAR
```

**bStart:** A rising edge activates the function block.

**iSession:** Is linked with the session number from function block FB\_IpStartSession.

## OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
    iPortNo    : WORD;
END_VAR
```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

**iPortNo:** TCP port number that was allocated when the TCP/IP connection was opened (local port number).

#### 5.13.7.2.2.4 FB\_IpClose

The function block is required for closing a TCP/IP connection from the PLC controller. With a positive edge of **bStart** a connection with the IP address from **sRemoteIpAddr** is terminated. The function block sends a FIN signal and waits for the confirmation of the disconnection. The partner device and the connection must exist and be operational. If **bResetConnection** is set the function block sends a signal to indicate that the connection was terminated, but it does not wait for confirmation from the other device. **bBusy** is set as long set as the function block is active. If **bBusy** is reset and **bError** is FALSE the TCP/IP connection was terminated successfully.

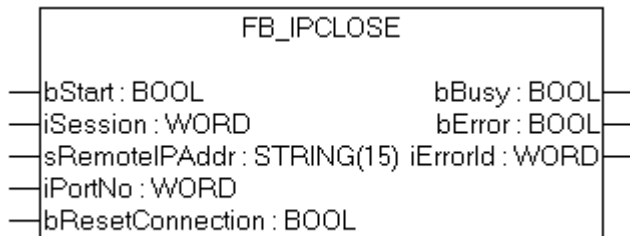


Fig. 131: Function block FB\_IPCLOSE

#### INPUT

```
VAR_INPUT
  bStart      : BOOL;
  iSession    : WORD;
  sRemoteIpAddr : STRING(15);
  iPortNo     : WORD;
  bResetConnection : BOOL;
END_VAR
```

**bStart:** A rising edge activates the function block.

**iSession:** Is linked with the session number from function block FB\_IpStartSession.

**sRemoteIpAddr:** IP address of the device with which the connection is to be terminated.

**iPortNo:** Port number of the device with which the connection is to be terminated.

**bResetConnection:** FALSE: FIN is sent; TRUE: the TCP/IP connection is closed without waiting for acknowledgement from the partner device. In both cases a new Open is required for re-establishing the connection.

#### OUTPUT

```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  iErrorId   : WORD;
END_VAR
```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

### 5.13.7.2.2.5 FB\_IpReceive

The function block *FB\_IpReceive* enables receiving of UDP or TCP telegrams. Which of the two connections is used is specified in *FB\_IpStartSession* [► 143].

Received data must be stored in a PLC variable. To this end a pointer to **pBuffAddr** is required, and the size of the variable has to be entered in *cbBuffLen*. A positive edge of **bValid** indicates that the memory contains data or the data of the associated variable are now valid. A positive edge of **bClear** indicates that the function block is ready to receive data again or, if data are still in the buffer they are copied to the variable next.

With TCP in particular you should monitor how many data were received or are still in the buffer. With TCP/IP data are stored as a "stream", i.e. there is no start or end. With UDP, on the other hand, the content of a UDP frame is always stored in its own buffer.

Four UDP telegrams can be cached; additional UDP telegrams are lost. **cbReceive** indicates the number of data in bytes copied into the variable. If the buffer contains more data than were read, the number of remaining data is in **cbBytesInStream**.

**sReceiveIPAddr** indicates the IP address of the device that has sent data to the Beckhoff controller, along with the corresponding port number **iReceivePortNo**. Both variables are cleared with **bClear**.

**sReceiveIPAddr** and *iReceivePortNo* can be used for sending data back to the device via function block *FB\_IpSend* [► 148].

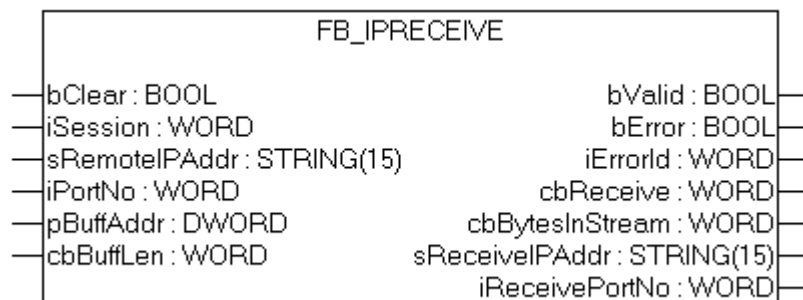


Fig. 132: Function block FB\_IPRECEIVE

#### INPUT

```
VAR_INPUT
    bClear      : BOOL;
    iSession    : WORD;
    sRemoteIPAddr : STRING(15);
    iPortNo     : WORD;
    pBuffAddr   : DWORD;
    cbBuffLen   : WORD;
END_VAR
```

**bClear**: A rising edge clears the memory. The function block is then ready to receive data again.

**iSession**: Is linked with *iSession* from function block *FB\_StartSession* [► 143].

**sRemoteIPAddr**: Can be used as filter for approving only a specific IP address.

**iPortNo**: Can be used as filter for approving only a specific port.

**pBuffAddr**: With **ADR** the pointer to the variable is transferred where the data that were received are to be copied.

**cbBuffLen**: Size of the variable, can be determined with **SIZEOF**.

## OUTPUT

```
VAR_OUTPUT
  bValid      : BOOL;
  bError      : BOOL;
  iErrorId    : WORD;
  cbReceive   : WORD;
  cbBytesInStream : WORD;
  sReceiveIPAddr : STRING(15);
  iReceivePortNo : WORD;
END_VAR
```

**bValid:** A positive edge change indicates that new data were received. These data are now available in the variable available that was linked via the pointer **pBuffAddr**.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

**cbReceive:** Indicates the number bytes that were copied.

**cbBytesInStream:** Indicates the data remaining in the memory. This should always be zero. If the value is >0, the variable linked to **pBuffAddr** is too small. A further read operation is required for obtaining the remaining data.

**sReceiveIPAddr:** Indicates the IP address of the device that has sent the data. Cleared with positive edge of **bClear**.

**iReceivePortNo:** Indicates the port number of the device that has sent the data. Cleared with positive edge of **bClear**.

### 5.13.7.2.2.6 FB\_IpSend

The function block sends data via TCP or UDP. Which of the connections is used is specified in [FB\\_IpStartSession](#) [► 143]. In **pBuffAddr** the pointer to the variable containing data for sending is specified via "ADR". **cbBuffLen** indicates the length of the data. **sRemoteIPAddr** indicates the IP address to which the data are sent. With **iPortNo** this is linked with the port number of [FB\\_IpOpen](#) [► 145] in TCP. With UDP any port number can be used. **iPortNo** is the sender port number. The function block is activated and the data are sent with a rising edge of **bStart**. **bBusy** is TRUE as long the function block is active. Once the data have been sent **bBusy** switches to FALSE. **bError** also remains FALSE. If an error occurs **bError** is set.

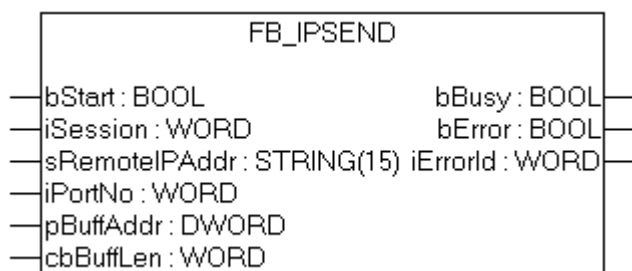


Fig. 133: Function block FB\_IPSEND

## INPUT

```
VAR_INPUT
  bStart      : BOOL;
  iSession    : WORD;
  sRemoteIPAddr : STRING(15);
  pBuffAddr   : DWORD;
  cbBuffLen   : WORD;
END_VAR
```

**bStart:** A rising edge activates the function block.

**iSession:** Is linked with the session number from function block [FB\\_IpStartSession](#) [► 143].

**sRemoteIPAddr:** IP port number of the device to which data are to be sent.

**iPortNo:** Sender port number. For TCP the port number from the [FB\\_IpOpen \[► 145\]](#) function block must be used, for UDP any port number can be used.

**pBuffAddr:** Pointer to the data to be sent (command: **ADR**).

**cbBuffLen:** Length of the data to be sent. The length should always be less or equal the variable to which the pointer of *pBuffAddr* points (command: **SIZEOF**)

## OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
END_VAR
```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

### 5.13.7.2.3 TCP/IP

#### 5.13.7.2.3.1 TCP/IP Client

TCP/IP is a connection-oriented communication type (peer-to-peer connection). In this example we illustrate how a connection is established between the Beckhoff controller and a server. The required function blocks are *FB\_IpStartSession*, *FB\_IpOpen*, *FB\_IpSend* and optionally *FB\_IpReceive*, *FB\_IpClose* and *FB\_IpEndSession*.

It is advisable to program a step sequence as illustrated in the example below.

In this case the Beckhoff controller is the client and the partner device a TCP server. The server is a VB6 program.

#### Step 1

##### FB\_IpStartSession

The function block allocates resources on the controller for the TCP/IP communication. The function block becomes active with *bStart*. **bBusy** is set as long as the function block is busy. **iDevice** should always be set to zero. **iPort** is used for the local TCP/IP port number. **iMaxConnection** indicates the maximum number of connections (up to 3).

## INPUT

**bStart:** A rising edge activates the function block.

**iDevice:** always "0"

**eSocketType:** For TCP/IP "SOCK\_STREAM" should be used. Please note that the data are stored as a byte stream.

If possible, the length of the received data should be known or a protocol with start and end ID should be used, so that the start and end can be detect unambiguously in the data stream.

**eSocketProtocol:** For TCP/IP "IPPROTO\_TCP" should be used

**iPort:** Sender port number

**iMaxConnection:** Number of possible connections (max. 3)

**tTimeout:** Time after which the attempt is aborted.

## OUTPUT

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

**iSession:** Forwards the session number to all IP function blocks for which this connection was created.

## Step 2

### FB\_IpOpen

Received data must be stored in a PLC variable. To this end a pointer to **pBuffAddr** is required, and the size of the variable has to be entered in **cbBuffLen**. A positive edge of **bValid** indicates that the memory contains data or the data of the associated variable are now valid.

## INPUT

**bClear:** A rising edge clears the memory. The function block is then ready to receive data again.

**iSession:** Is linked with *iSession* from function block **FB\_StartSession**.

**sRemoteIPAddr:** Can be used as filter for approving only a specific IP address

**eSocketProtocol:** For TCP/IP "IPPROTO\_TCP" should be used

**iPort:** Local port number

**iMaxConnection:** Number of possible connections (max. 3)

**tTimeout:** Time after which the attempt is aborted.

## OUTPUT

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

**iSession:** Forwards the session number to all IP function blocks for which this connection was created.

Data can be returned as soon as the first data have been received. This functionality is optional and is used in the example.

## Example



Download VB6 program as TCP/IP server zip file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207366411.zip>)



Download TwinCAT project as TCP/IP client prx file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207368587.prx>)

### 5.13.7.2.3.2 TCP/IP Server

TCP/IP is a connection-oriented communication type (peer-to-peer connection). In this example we illustrate how an external connection is established with the Beckhoff controller. The required function blocks are **FB\_IpStartSession**, **FB\_IpReceive** and optionally **FB\_IpSend**, **FB\_IpClose** and **FB\_IpEndSession**.

It is advisable to program a step sequence as illustrated in the example below.

The Beckhoff controller acts as server and the partner device as TCP client. The client is a VB6 program.

## Step 1

### FB\_IpStartSession

The function block allocates resources on the controller for the TCP/IP communication. The function block becomes active with *bStart*. **bBusy** is set as long as the function block is busy. **iDevice** should always be set to zero. **iPort** is used for the local TCP/IP port number. **iMaxConnection** indicates the maximum number of connections (up to 3).

#### INPUT

**bStart**: A rising edge activates the function block.

**iDevice**: always "0"

**eSocketType**: For TCP/IP "SOCK\_STREAM" should be used. Please note that the data are stored as a byte stream. The length of the data to be received should be known or a protocol with a start/end ID should be used so that the start and end can be identified from the data stream.

**eSocketProtocol**: For TCP/IP "IPPROTO\_TCP" should be used

**iPort**: Receiving port number

**iMaxConnection**: Number of possible connections (max. 3)

**tTimeout**: Time after which the attempt is aborted.

#### OUTPUT

**bBusy**: This output remains TRUE until execution of the command is complete.

**bError**: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId**: Contains the command-specific error code of the most recently executed command (see table).

**iSession**: Forwards the session number to all IP function blocks for which this connection was created.

## Step 2

### FB\_IpReceive

Received data must be stored in a PLC variable. To this end a pointer to *pBuffAddr* is required, and the size of the variable is specified in *cbBuffLen*. A positive edge of *bValid* indicates that the memory contains data or the data of the associated variable are now valid.

#### INPUT

**bClear**: A rising edge clears the memory. The function block is then ready to receive data again.

**iSession**: Is linked with *iSession* from function block *FB\_StartSession*.

**sRemoteIPAddr**: Can be used as filter for approving only a specific IP address.

**eSocketProtocol**: For TCP/IP "IPPROTO\_TCP" should be used

**iPort**: Local port number

**iMaxConnection**: Number of possible connections (max. 3)

**tTimeout**: Time after which the attempt is aborted.

## OUTPUT

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

**iSession:** Forwards the session number to all IP function blocks for which this connection was created.

## Example



Download VB6 program as TCP/IP client zip file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207370763.zip>)



Download TwinCAT project as TCP/IP server prx file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207372939.prx>)



### 5.13.7.2.4 UDP/IP

#### 5.13.7.2.4.1 UDP/IP connection

UDP is a very simple Ethernet connection. UDP data are sent without a mechanism for determining whether the telegram has arrived or not. For UDP communication to work the port number on both sides must be known.

The BX9000 sends data to a VB6 program, which returns the data again.

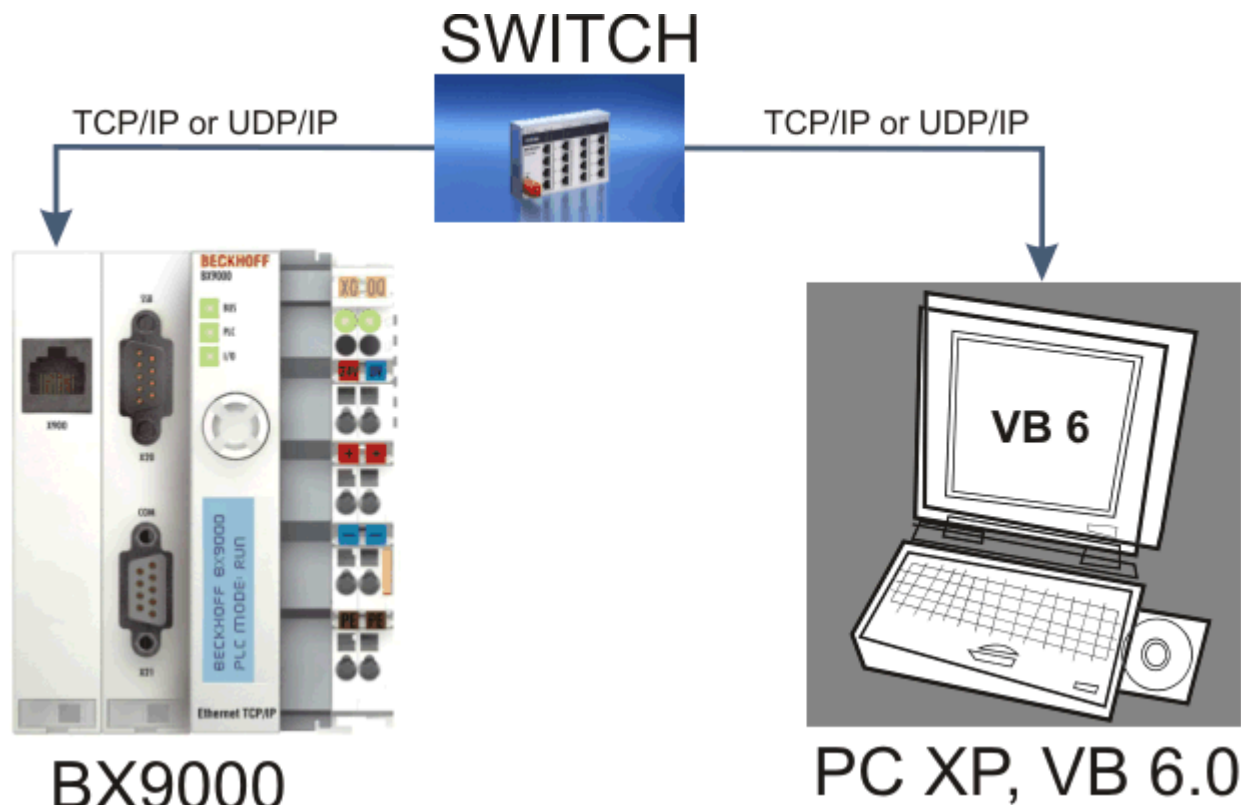


Fig. 134: UDP/IP connection

#### Step 1: Preparation of the UDP communication

##### FB\_IpStartSession

The function block allocates resources on the controller for the UDP/IP communication. The function block becomes active with *bStart*. **bBusy** is set as long as the function block is busy. **iDevice** should always be set to zero. **iPort** is used for the local UDP port number of the BX9000. **iMaxConnection** indicates the maximum number of connections (up to 3). The *eSocketType* is set to "SOCK\_DGRAM". **eSocketProtocol** must be set to "IPPROTO\_UDP" for UDP communication. *tTimeout* is not used for UDP communication. The **iSession** must be linked with the following FB\_IpSend and FB\_IpReceive function blocks.

#### Step 2: Sending of UDP frames

##### FB\_IpSend

A UDP frame is sent with a positive edge of **bStart**. The IP address is described with **sRemoteIPAddr** and the destination UDP port number with **iPortNo**. The sender UDP port number was already configured in the **FB\_IpStartSession**. If **bBusy** is reset by the function block the command was executed.

### Step 3: Receiving of UDP frames

#### FB\_IpReceive

This function block is used for receiving data. When the function block is called the block monitors the arrival of UDP frames. Up to 4 UDP frames are cached, further UDP frames are discarded. **sRemoteIPAddr** can be used to set up an IP address filter to limit data reception to a particular device. To receive all UDP frames enter an empty string or leave the variable open. If an IP address filter was configured the port number can also be filtered. Simply enter the corresponding port number in variable **iPortNo**. To receive all UDP data leave the variable open

If data are received the variable **bValid** is set to TRUE. The data are now valid. If the value of the variable **cbBytesInStream** is not zero, the variable linked with the function block is too small and data remain in the buffer.

#### Sample program

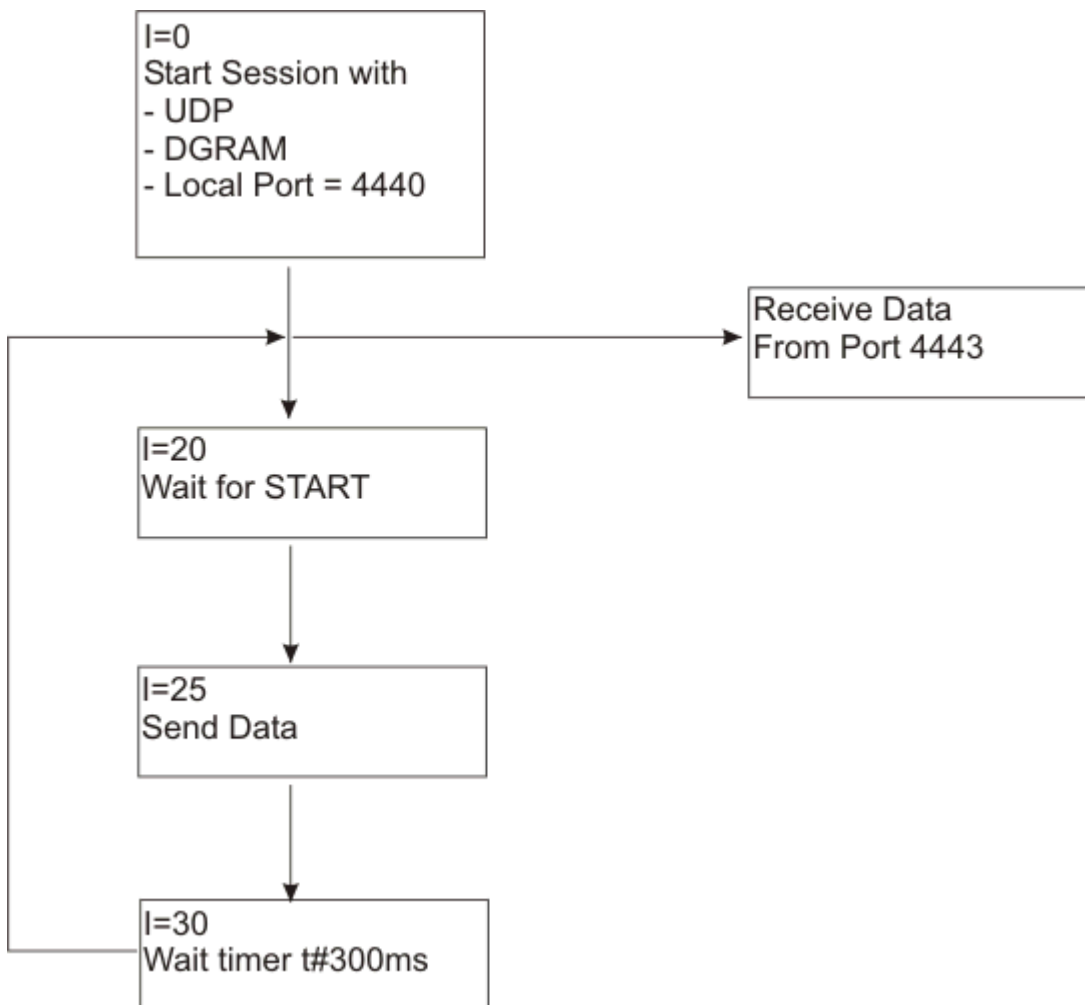




Fig. 135: Sample program

#### Example

 Download VB6 program as TCP/IP server zip file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207375115.zip>)

 Download BX9000 TwinCAT project as TCP/IP client, prx file (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207377291.prx>)

### 5.13.7.3 ModbusTCP Client

#### 5.13.7.3.1 ModbusTCP Client

Via the ModbusTCP client (Modbus master) the BX9000 can establish a connection to a ModbusTCP server (Modbus slave). 3 function blocks are available. Fb\_MBConnect for establishing a TCP/IP connection, FB\_MBGenericReq for the sending and receiving of any Modbus functions, and FB\_MBClose for closing the TCP/IP connection. Since FB\_MBGenericReq is a generic function block, the corresponding Modbus functions have to be implemented. This enables sending of protocols that are not Modbus compliant. The maximum number of simultaneous client connections is 3. The option of closing and re-opening a TCP/IP connection means that the number of ModbusTCP servers that are accessible is almost unlimited.

#### FB\_MBConnect

FB\_MBConnect connects the BX9000 with another device via ModbusTCP. A connection is established with rising edge of *bExecute*. The IP address is set via *sIPAddr*. The TCP port number to be used is set via *nTCPPort* (usually set to 502 for ModbusTCP). **bBusy** is set as long the function block is active. If the ModbusTCP connection was initialized successfully *bBusy* is set to FALSE and the *bError* flag is FALSE. If the *bError* flag is TRUE be, the connection attempt has failed and the associated error code is stored in variable *nErrId*. *nHandle* must be linked to the function blocks **FB\_MBGenericReq** and **FB\_MBClose**.

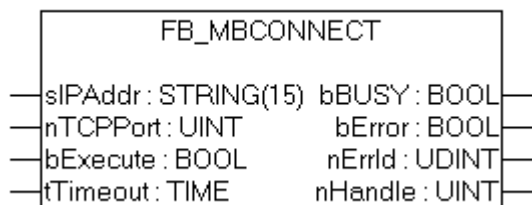


Fig. 136: Function block FB\_MBCONNECT

#### INPUT

```
VAR_INPUT
  sIPAddr      : STRING(15);
  nTCPPort     : UINT;
  bExecute     : BOOL;
  tTimeout     : TIME;
END_VAR
```

**sIPAddr:** IP address of the ModbusTCP-server (slave device) with which a Modbus connection is to be established.

**nTCPPort:** ModbusTCP port number, for ModbusTCP usually 502<sup>dec</sup>.

**bExecute:** A rising edge activates the function block.

**tTimeout:** Time after which the attempt is aborted.

#### OUTPUT

```
VAR_OUTPUT
  bBusy        : BOOL;
  bError       : BOOL;
  iErrId       : WORD;
  nHandle      : UINT;
END_VAR
```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table [► 158]).

**nHandle**: Handle that must be connected with function blocks FB\_MBGenericReq and FB\_MBClose.

## FB\_MBGENERICREQ

Function block FB\_MBGENERICREQ enables sending and receiving of any ModbusTCP functions. With a rising edge of **bExecute** the function block becomes active, **bBusy** is set and the function block sends the data contained in **pReqBuff** (pointer to the data to be sent). The function block reads the length of the data to be sent from variable **cbReqLen**. The variable **nHandle** must be linked to the variable from the function block Fb\_MBConnect **nHandle**. The response from the ModbusTCP server (slave) is stored in **pResBuff** (pointer to the receive data). The size of the variable should be adequate for receiving all data of a ModbusTCP telegram. If the ModbusClient does not receive a response within **tTimeout**, **bBusy** is set to FALSE and **bError** is set. Variable **nErrId** contains the error code. **cbResponse** contains the number of received bytes. This number can be compared with the buffer size of **cbResLen**. If **cbResponse** is greater than **dResLen** data are lost and a larger buffer should be chosen.

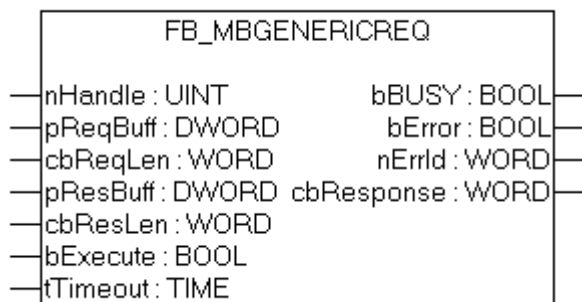


Fig. 137: Function block FB\_MBGENERICREQ

## INPUT

```
VAR_INPUT
  nHandle      : UINT;
  pReqBuff     : DWORD;
  cbReqLen     : WORD;
  pResBuff     : DWORD;
  cbResLen     : WORD;
  bExecute     : BOOL;
  tTimeout     : TIME;
END_VAR
```

**nHandle**: Handle, which will be linked to the Fb\_MBConnect function blocks.

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

**cbReqLen**: Length of the data to be sent.

**pResBuff**: Pointer to the data to be received. The size of the variable must be adequate.

**cbResLen**: Length of the data to be received. The length must be adequate.

**bExecute**: A rising edge activates the function block.

**tTimeout**: Time after which the attempt is aborted.

## OUTPUT

```
VAR_OUTPUT
  bBusy        : BOOL;
  bError       : BOOL;
  iErrId       : WORD;
  cbResponse   : WORD;
END_VAR
```

**bBusy**: This output remains TRUE until execution of the command is complete.

**bError**: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command ([see table \[► 158\]](#)).

**cbResponse:** Number of bytes that were received.

## FB\_MBCLOSE

Function block FB\_MBCLOSE enables closing of a TCP/IP connection. The function block is activated with a rising edge of **bExecute**. The bit **bBusy** is set as long as the function block is busy. **nHandle** must be linked to the function block Fb\_MBConnect **nHandle**. If an error occurs **bError** is set to TRUE. The error code can be found in **nErrId**. Once the function block has been called without error, Fb\_MBConnect can be used to open a new socket.

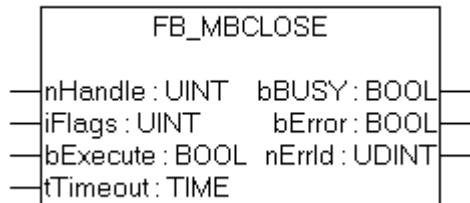


Fig. 138: Function block FB\_MBCLOSE

## INPUT

```
VAR_INPUT
    nHandle      : UINT;
    iFlags       : UINT;
    bExecute     : BOOL;
    tTimeout     : TIME;
END_VAR
```

**nHandle:** Handle, which will be linked to the Fb\_MBConnect function blocks.

**iFlags:** 0 - TCP/IP connection is closed even if the partner device cannot be reached, 1 - TCP/IP connection is terminated with "FIN"; for the function block to be able to close the TCP/IP connection without error it must exist and communication must be operational. It is recommend to use 0 because in this case the TCP/IP connection is always closed, irrespective of the state of the partner device.

**bExecute:** A rising edge activates the function block.

**tTimeout:** Time after which the attempt is aborted.

## OUTPUT

```
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    iErrId      : WORD;
END_VAR
```


**bBusy:** This output remains TRUE until execution of the command is complete.


**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command ([see table \[► 158\]](#)).

## Example

For the example you need two BX9000 or a BC9x00 instead of a BX9000.

 Download first BX9000 as ModbusTCP client (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207379467.prx>)

 Download second BX9000 as ModbusTCP server (or BC9x00) (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207381643.prx>)

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version $\geq 1.14$	TcBaseBX9000.lbx

## Error Codes

Error code	Description
0xFB00	No valid IP address
0xFC00	Response length exceeds memory
0xFD00	Request could not be sent
0xFF00	Timeout during communication

## Modbus closed

Error code	Description
0xFA00	Connection is already closed
0xFF00	Timeout during closing of the connection. RST is used, but the remote device does not exist.

## Modbus client connect

Error code	Description
0xFA00	Internal connection is refused
0xFA01	Connection timeout event
0xFA02	Connect was not accepted by remote
0xFA03	Invalid IP address
0xFA04	All Modbus connections occupied
0xFA05	No further internal socket available
0xFA06	Internal socket error
0xFA07	Error during connect call

## 5.13.7.4 SMTP

### 5.13.7.4.1 FB\_Smtp

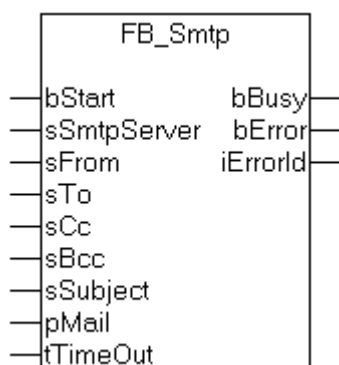


Fig. 139: Function block FB\_Smtp

This function block uses the SMTP protocol (simple mail transfer protocol) to send e-mails. The function block can, for instance, be used to send errors, diagnostic information, or warnings in the form of e-mails. The recipients' addresses are passed as strings to the *sTo*, *sCc*, *sBcc* and *sSubject* input variables. The maximum string length for recipients' addresses is limited to 80 characters in order to save resources. The string with the mail text itself may, however, be longer.

## VAR\_INPUT

```
VAR_INPUT
    bStart      : BOOL;
    sSmtpServer : STRING(15);
    sFrom       : STRING;
    sTo        : STRING;
    sCc        : STRING;
    sBcc       : STRING;
    sSubject    : STRING;
    pMail       : DWORD;
    tTimeOut    : TIME;
END_VAR
```

**bStart:** The function block is activated by a rising edge at this input.

**sSmtpServer:** IP address of the SMTP server as a string.

**sFrom:** A string containing the e-mail address of the sender. If the string supplied is empty, then the Bus Controller generates an e-mail address from the name of the Bus Controller and the MAC ID. The maximum string length is limited to 80 characters. It is possible to enter multiple recipient addresses separated by semicolons.

**sTo:** A string containing the e-mail address of the recipient. A valid e-mail address must be given. The maximum string length is limited to 80 characters. It is possible to enter multiple recipient addresses separated by semicolons.

**sCc:** A string containing the e-mail address of a further recipient (Cc = carbon copy). This string can also be empty. A copy of the e-mail is sent to this recipient. The e-mail address of this recipient is **visible** to other recipients. The maximum string length is limited to 80 characters. It is possible to enter multiple recipient addresses separated by semicolons.

**sBcc:** A string containing the e-mail address of a further recipient (Bcc = blind carbon copy). This string can also be empty. A copy of the e-mail is sent to this recipient. The e-mail address of this recipient is **not visible** to other recipients. The maximum string length is limited to 80 characters. It is possible to enter multiple recipient addresses separated by semicolons.

**sSubject:** A string containing the e-mail's subject line. This string can also be empty. The maximum string length is limited to 80 characters.

**pMail:** The address (a pointer) to a null-terminated string containing the e-mail text. This string can also be empty. The address of the string can be determined with the ADR operator.

**tTimeOut:** Maximum time allowed for the execution of the command.

## VAR\_OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
END_VAR
```

**bBusy:** This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the *tTimeOut* input.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (table).

Error code (hex)	Description
0x8000	SMTP server not found.
0x8001	Resource error.
0x8002	Socket resource error.
0x8003	Connection fault.
0x8004	Communication fault.
0x8005	Rx error. Communication time exceeded.
0x8006	Rx error. Communication fault.
0x8007	Rx error. Frame error.
0x8008	Communication error. Wrong response.
0x8009	Tx error. Communication fault.
0x800A	Communication shutdown error.
0x800B	Communication timeout.
0x8010	Invalid parameter.

### Example of a call in FBD

```

PROGRAM MAIN
VAR
    fbSMTP    : FB_Smtp;
    bSend     : BOOL;
    sMsg      : STRING(100) := 'Test';
    bBusy     : BOOL;
    bError    : BOOL;
    nErrId    : UDINT;
END_VAR

```

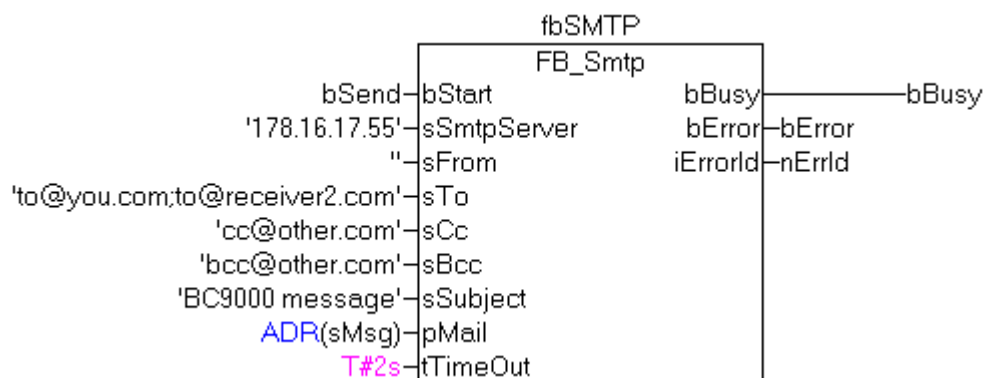


Fig. 140: Function block fbSMTP

In this example, a rising edge at the *bStart* input sends a mail to 4 recipients.

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version ≥1.12	TcBaseBX9000.lbx

## 5.13.7.5 SNTP

### 5.13.7.5.1 Time protocol (SNTP)

(BX9000 from firmware version 1.12, BC9050, BC9x20)

The Simple Network Time Protocol is used to synchronize clocks via the internet. The BX9000 can be synchronized with a time server.



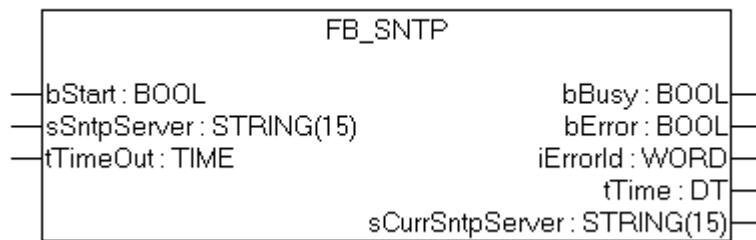


Fig. 141: Function block FB\_Sntp

### FUNCTION\_BLOCK FB\_Sntp

If an IP address is entered the Bus Terminal controller uses the Sntp protocol. If an empty string is transferred, the time protocol (UDP port 37) is used.

#### VAR\_INPUT

```
bStart          :BOOL;
sSntpServer      :STRING(15);
tTimeout         :TIME;
```

**bOpen:** rising edge starts the function block

**sSntpServer:** Sntp server entry. If an empty string is entered, the time protocol is used (UDP port 37)\*.

**tTimeout:** TMOut after which the process is to be terminated

#### VAR\_OUTPUT

```
bBusy           :BOOL;
bError          :BOOL;
iErrorId        :WORD;
tTime           :DT;
cCurrSntpServer :STRING(15);
```

**bBusy:** The function block is active as long it is TRUE.

**bError:** Error bit.

**iErrorId:** Error number.

**tTime:** Time and date.

**sCurrSntpServer:** IP address of the SMTP server.

Return parameter iErrId	Meaning
0	No error
<> 0	Error number [► 166]



Download (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/prx/3207383819.prx>)

## 5.13.7.6 Services

### 5.13.7.6.1 FB\_AddDnsServer

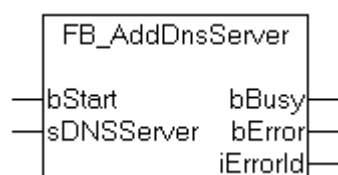


Fig. 142: Function block FB\_AddDnsServer

Up to a maximum of three DNS servers can be passed to the Bus Controller with this function block.

**INPUT**

```

VAR_INPUT
    bStart      : BOOL;
    sDNSServer   : STRING(15);
END_VAR

```

**bStart:** The function block is activated by a rising edge at this input.

**sDNSServer:** A string containing the IP address of the DNS server.

**OUTPUT**

```

VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    iErrorId    : WORD;
END_VAR

```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command ([see table \[► 166\]](#)).

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

**5.13.7.6.2 FB\_GetHostByAddr**

Fig. 143: Function block FB\_GetHostByAddr

This function block can determine the host name associated with a specified IP address.

**INPUT**

```

VAR_INPUT
    bStart      : BOOL;
    sIPAddr     : STRING(15);
    pHostName   : DWORD;
    cbMaxNameLen : WORD;
END_VAR

```

**bStart:** The function block is activated by a rising edge at this input.

**sIPAddr:** A string containing the IP address of the host.

**pHostName:** Contains the address of a string buffer into which the host name that has been found is written. It is the programmer who is responsible for dimensioning the buffer appropriately so that *cbMaxNameLen* bytes can be removed from it. The address can be determined with the ADR operator.

**cbMaxNameLen:** Contains the length, in bytes, of the buffer into which the host name that has been found is to be written.

**OUTPUT**

```

VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
END_VAR

```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table [► 166]).

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

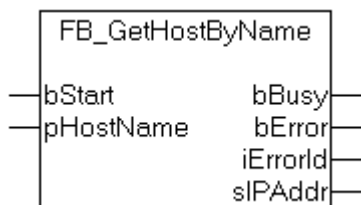
**5.13.7.6.3 FB\_GetHostByName**

Fig. 144: Function block FB\_GetHostByName

This function block can determine the IP address associated with a specified host name.

**INPUT**

```

VAR_INPUT
    bStart      : BOOL;
    pHostName   : DWORD;
END_VAR

```

**bStart:** The function block is activated by a rising edge at this input.

**pHostName:** Contains the address of a string variable with the host name. The address of a string variable can be determined with the ADR operator.

**OUTPUT**

```

VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
    sIPAddr    : STRING(15);
END_VAR

```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table [► 166]).

**sIPAddr:** When successful, this variable contains the IP address of the host.

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

#### 5.13.7.6.4 FB\_GetNetworkConfig

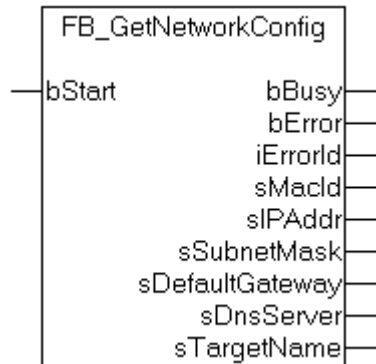


Fig. 145: Function block FB\_GetNetworkConfig

This function block can determine information on the current network configuration.

#### INPUT

```

VAR_INPUT
    bStart      : BOOL;
END_VAR
  
```

**bStart:** The function block is activated by a rising edge at this input.

#### OUTPUT

```

VAR_OUTPUT
    bBusy      : BOOL;
    bError     : BOOL;
    iErrorId   : WORD;
    sMacId     : STRING(17);
    sIPAddr    : STRING(15);
    sSubnetMask : STRING(15);
    sDefaultGateway : STRING(15);
    sDnsServer : STRING(15);
    sTargetName : STRING(20);
END_VAR
  
```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command ([see table 166](#)).

**sMacId:** Returns the MAC ID of the bus controllers, e.g. '00-01-05-13-45-63'.

**sIPAddr:** Returns the IP address of the Bus Controller.

**sSubnetMask:** Returns the SubNet mask.

**sDefaultGateway:** Returns the default gateway.

**sDnsServer:** Returns the default DNS server (assigned by the DHCP server).

**sTargetName:** Returns the Bus Controller's current target name.

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

5.13.7.6.5 FB\_SetTargetName

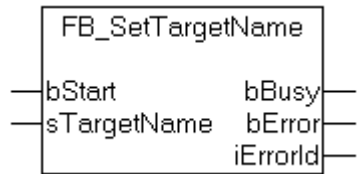


Fig. 146: Function block FB\_SetTargetName

With this function block a target name can be assigned to the Bus Controller.

INPUT

```
VAR_INPUT
    bStart      : BOOL;
    sTargetName : STRING(20);
END_VAR
```

**bStart:** The function block is activated by a rising edge at this input.

**sTargetName:** A string containing the new target name of the Bus Controller.

OUTPUT

```
VAR_OUTPUT
    bBusy      : BOOL;
    bError      : BOOL;
    iErrorId    : WORD;
END_VAR
```

**bBusy:** This output remains TRUE until execution of the command is complete.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command (see table).

Development environ- ment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

## 5.13.7.6.6 Error numbers

Hex	Description
0x8010	Incorrect parameters
0x8011	DNS server list full
0x801F	Resource error (internal)
0x8020	Invalid host length
0x8021	No host name found
0x802E	Heap error (internal)
0x802F	Resource error (internal)
0x8030	Incorrect parameters
0x8031	Invalid host name
0x8032	Host name too long
0x803F	Resource error (internal)
0x804F	Resource error (internal)
0x8050	No valid IP address
0x8052	No valid class-D IP address
0x8055	Stack error (internal)
0x805F	Resource error (internal)
0x8060	ERR_SNTP_INVALID_SERVER
0x8061	ERR_SNTP_NO_MORE_SOCKETS
0x8064	ERR_SNTP_RX_ERR
0x8065	<b>ERR_SNTP_CONN_SHUT_DOWN</b>
0x8066	ERR_SNTP_TIMEOUT
0x8100	Incorrect parameters
0x8102	No open connection
0x8104	No data
0x8106	Buffer overflow (only for UDP)
0x8200	Connection was already open

Hex	Description
0x8201	Connection error
0x8202	timeout
0x8203	Resource error (internal)
0x8300	timeout
0x8301	Remote connection terminated
0x8302	Internal fault
0x8303	Resource error (internal)
0x8304	No remote connection
0x8400	Connection not open
0x8401	timeout
0x8800	No valid network configuration (e.g. DHCP still running)
0x8F09	ADS socket not available (too many open ADS connections)
0x8F10	Session not started
0x8F11	Session not ready
0x8F12	Invalid session ID
0x8F13	Invalid state (internal)
0x8F20	No free slots. Two sessions already started
0x8F21	Error during task initialization
0x8F22	Session is already closed

## 5.13.8 TcSystemBX9000

### 5.13.8.1 Overview: TcSystemBX9000

#### Download

To download the libraries click on the link. Please copy the libraries to directory TwinCAT\PLC\LIB.

- TcSystemBX9000 (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207385995.zip>)



#### TcSystemBX9000

Ethernet diagnostics	Version	Firmware
		BX9000
FreeAdsTcpConnections	12.10.06	1.15
FreeModbusConnections	12.10.06	1.15
ModbusTCP_Diag	12.10.06	1.15
ModbusTCP_Prm	12.10.06	1.15
EthernetDiag	12.10.06	1.15

Ethernet Utilities	Version	Firmware
		BX9000
NT_GetTime	12.10.06	1.15

HTML Page	Version	Firmware
		BX9000
<a href="#">HTTP [▶ 167]</a>	12.10.06	1.15

### 5.13.8.2 HTML page on the BX9000

For diagnostic purposes you can create a simple HTML page on the BX9000. The page can be used for displaying strings from the PLC. If the page active, it can be displayed by entering the IP address in a web browser.



#### Note

#### Displaying your individual HTML page

In delivery state a general HTML interface is stored on the BX9000. This must be disabled, in order to display your individual HTML page. This is dealt with by the HTTP function block. When it is called for the first time it deactivates the default page and reports "bRe-bootNecessary:=TRUE". This means that the coupler has to be restarted. After a BX9000 restart you can call up your individual HTML page.

#### HTTP

A rising edge of **bActive** activates the web page. If the HTML page is available the bit **blsActive** is TRUE.

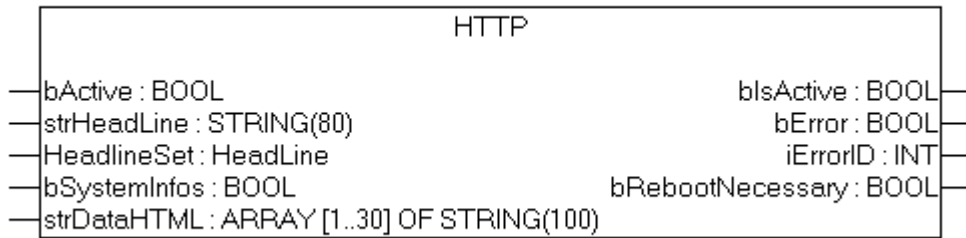


Fig. 147: Function block HTTP

**INPUT**

```

VAR_INPUT
  bActive      : BOOL;
  strHeadLine  : STRING(80);
  HeadlineSet  : HEADLINE;
  bSystemInfos : BOOL;
  strDataHTML  : ARRAY [1..30] OF STRING(100);
END_VAR

```

**bActive:** A positive edge activates the function block.

**strHeadLine:** Header of the HTML page (max. 80 characters).

**HeadLineSet:** Settings for header font size and color.

**bSystemInfos:** System information is displayed (TRUE) or not displayed (FALSE).

**strDataHTML:** max. 30 lines with max. 100 characters per line. An empty string completes the HTML page, further lines are not displayed.

**OUTPUT**

```

VAR_OUTPUT
  blsActive      : BOOL;
  bError         : BOOL;
  iErrorId       : INT;
  bRebootNecessary : BOOL;
END_VAR

```

**blsActive:** The HTML page is available in a web browser.

**bError:** This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *iErrorId*.

**iErrorId:** Contains the command-specific error code of the most recently executed command.

**bRebootNecessary:** To start the HTML page the controller must be rebooted.

**Example**

For this example a BX9000 is required.



Download BX9000 with HTML page (<http://infosys.beckhoff.com/content/1033/bx9000/Resources/zip/3207388171.zip>)

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version ≥1.15	Standard.lbx, TcBaseBX.lbx, TcBaseBX9000.lbx, TcSystemBX9000.lbx, TcSystem.lbx



## 5.14 Program transfer

### 5.14.1 Program Transfer via Ethernet

TwinCAT offers a facility for transferring the user program to the Bus Terminal Controller over the fieldbus. The BC/BX can be selected as the target system in PLC Control, after saving in the registry and restarting the TwinCAT system. The TwinCAT-level TwinCAT PLC is necessary.

Minimum requirements:

- TwinCAT 2.10 build 1251

#### Initializing the Bus Terminal Controllers

Possibility 1: If you use TwinCAT as a polling PLC.

The coupler must first be made known to the system before it can be selected in PLC Control.

Enter the Bus Terminal Controller in the System Manager, specify type, quantity and size of the fieldbus variables and link them with a task. Save your settings and activate the configuration. Then start the TwinCAT system and the cyclic task.

Possibility 2: If you only use TwinCAT for programming or configuring:

Click the TwinCAT icon, and open the features. You can enter the BX9000 under the AMS router.

Name: variable

AMS Net Id: IP address plus ".1.1"

IP address: IP address of the BX9000

Transport type: TCP/IP

Now start TwinCAT, either in the configuration mode (the blue TwinCAT icon) or in the RUN mode (the green TwinCAT icon)

#### TwinCAT System Manager

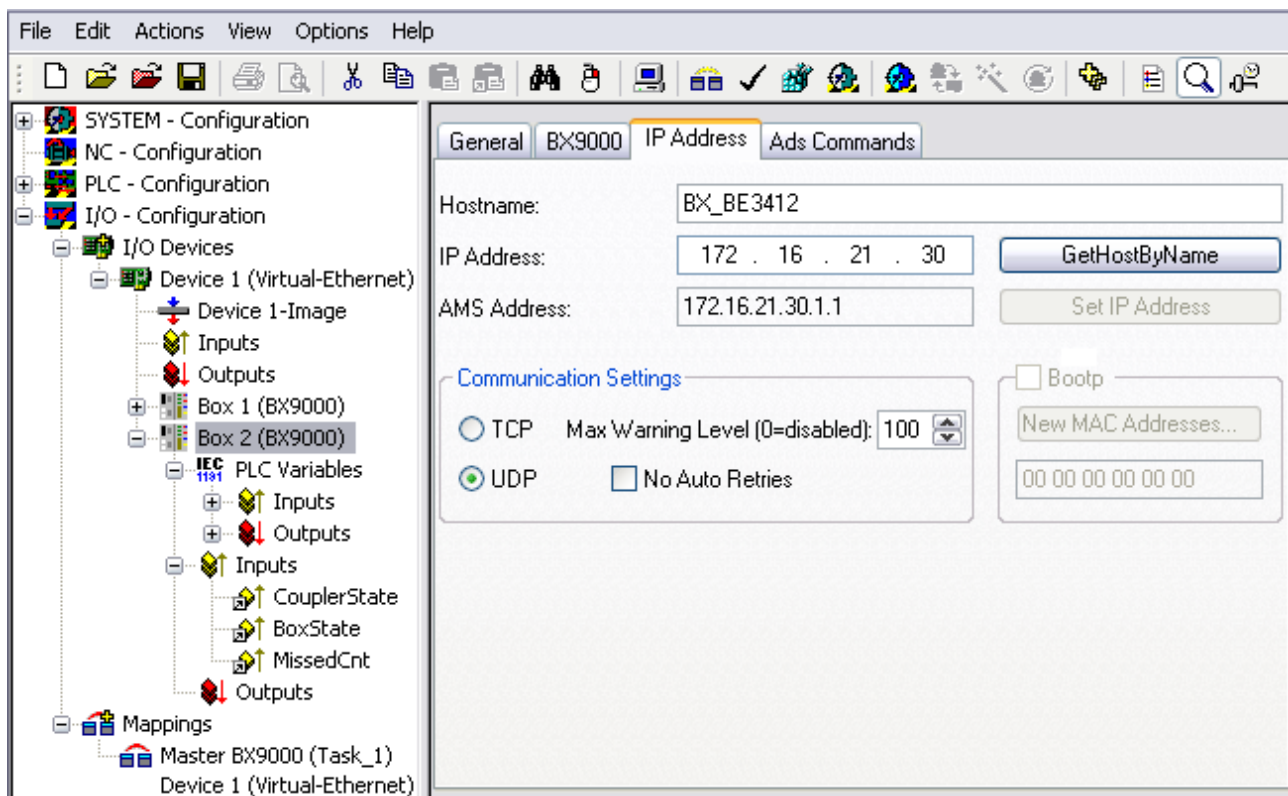


Fig. 148: IP address of the BX9000 in the TwinCAT System Manager

## PLC Control

When TwinCAT PLC Control is restarted, TwinCAT asks for the target platform, i.e. the device on which the user program is later to run. TwinCAT offers two target platforms as controller, the PC or the Bus Terminal Controller.

Two options are available to you for transmission to the Bus Terminal Controller:

- AMS for BCxx00 (Bus Terminal Controller without online change)
- AMS for BCxx50 and BX (Bus Terminal Controller with online change)
- BC serial – the serial cable for communication via the RS232 interface [► 170] of the PC and the programming interface of the Bus Terminal Controller

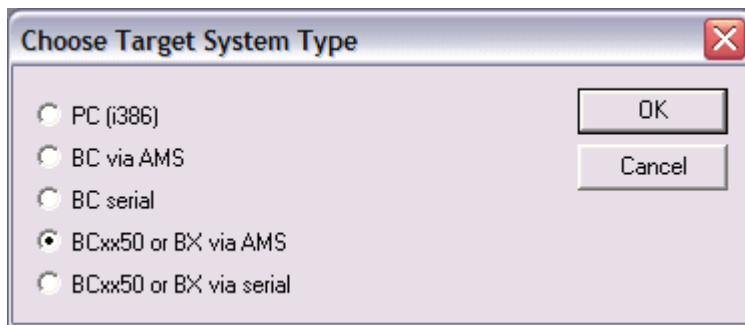


Fig. 149: Selecting the data transfer route - AMS

After your program has been created, select the target system under the *Online* toolbar. TwinCAT must be running to do this. In the sample, this is the Ethernet card with Box 1 and the Run-Time 1 of the Bus Terminal Controller.

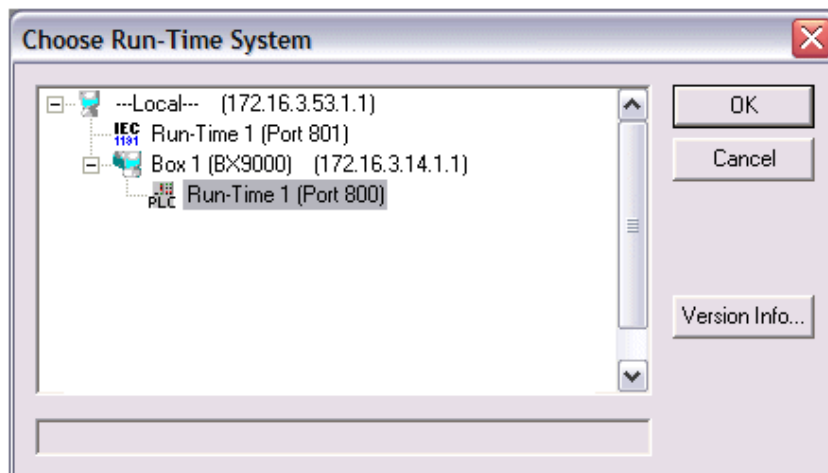


Fig. 150: Choose Target System

### 5.14.2 Program transfer via the serial interface

Every Bus Terminal Controller can be programmed via the PC's RS232 interface.

Select the serial interface in TwinCAT PLC Control.

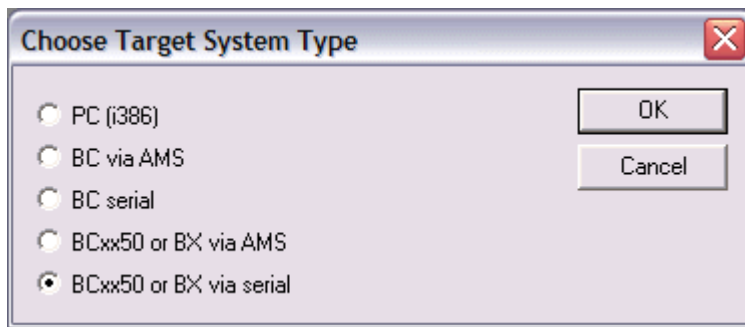


Fig. 151: Selecting the data transfer route - serial interface

The settings for the serial interface, port number, baud rate etc. are found under Online/Communication parameters in PLC Control.

The Bus Terminal Controller requires the following setting:

- Baud Rate: 9600/19200/38400/57600 baud (automatic baud rate detection)
- Stop bits: 1
- Parity: Straight line

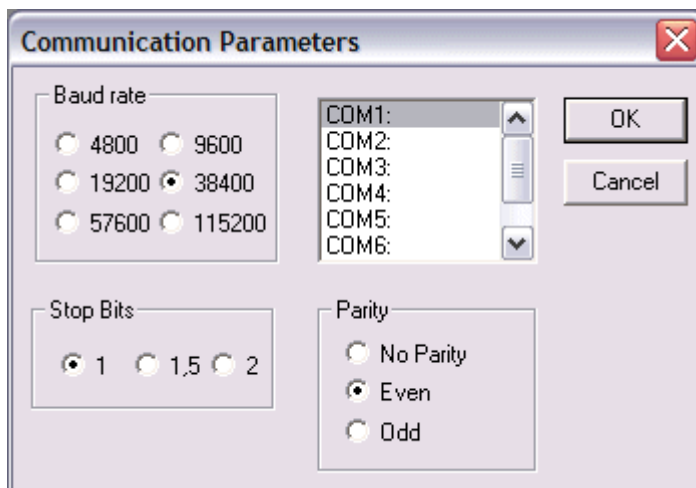


Fig. 152: Parameterization of the serial interface

### Program transfer via the serial interface and ADS

The Bus Terminal Controller can be programmed via the PC's RS232 interface. Before you can work with the Bus Terminal Controller, TwinCAT must be notified of it (see serial ADS [► 40]).

Select the ADS connection in TwinCAT PLC Control.

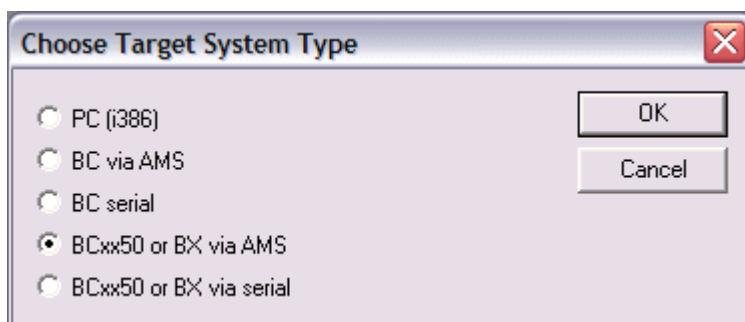


Fig. 153: Selecting the data transfer route - AMS

PLC Control can be accessed via *Online/Communication Parameters...*

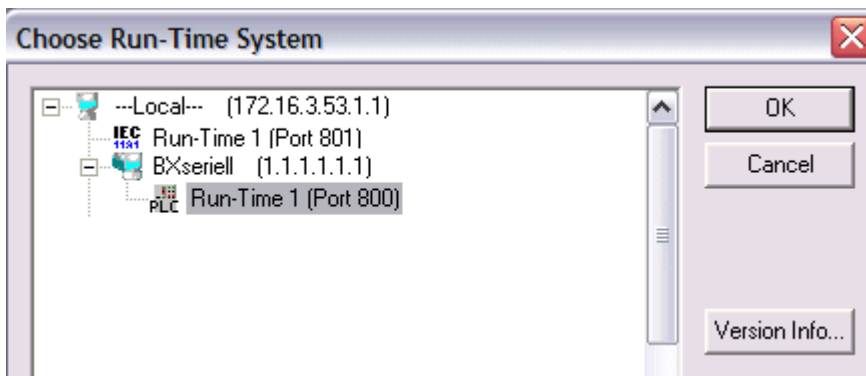


Fig. 154: Selecting the device

## 5.15 Process image

### 5.15.1 Fieldbus Process Image

There are two types of configuration on the BX Controller. You can read which of these two is active on the display of the BX after it has booted.

#### Default Config

The fieldbus data is located in the Default Config, starting from address 1000. (%IB1000... and %QB1000).

#### TwinCAT Config

In the TwinCAT Config, fieldbus data is not bound to specific addresses, but has a location that depends on how it has been linked.

The fieldbus process image is independent of which Ethernet protocol is in use.

## 6 Ethernet

### 6.1 Introduction to the system

#### 6.1.1 Ethernet

Ethernet was originally developed by DEC, Intel and XEROX (as the "DIX" standard) for passing data between office devices. The term nowadays generally refers to the *IEEE 802.3 CSMA/CD* specification, published in 1985. Because of the high acceptance around the world this technology is available everywhere and is very economical. This means that it is easy to make connections to existing networks.

There are now a number of quite different transmission media: coaxial cable (10Base5), optical fiber (10BaseF) or twisted pairs (10BaseT) with screen (STP) or without screen (UTP). Coaxial cable (10Base5), optical fiber (10BaseF) or twisted pairs (10BaseT) with screen (STP) or without screen (UTP).

Ethernet transmits Ethernet packets from a sender to one or more receivers. This transmission takes place without acknowledgement, and without the repetition of lost packets. To achieve reliable data communication, there are protocols, such as TCP/IP, that can run on top of Ethernet.

#### MAC-ID

The sender and receiver of Ethernet packets are addressed by means of the MAC-ID. The MAC-ID is a 6 byte identification code unique to every Ethernet device in the world. The MAC-ID consists of two parts. The first part (i.e. the first 3 bytes) is a manufacturer identifier. The identifier for Beckhoff is 00 01 05. The next 3 bytes are assigned by the manufacturer and implement a unique serial number. The MAC-ID can, for example, be used for the BootP protocol in order to set the TCP/IP number. This involves sending a telegram containing the information such as the name or the TCP/IP number to the corresponding node. You can read the MAC-ID with the KS2000 configuration software.

#### The Internet Protocol (IP)

The internet protocol (IP) forms the basis of this data communication. IP transports data packets from one device to another; the devices can be in the same network, or in different networks. IP here looks after the address management (finding and assigning MAC-IDs), segmentation and routing. Like the Ethernet protocol, IP does not guarantee that the data is transported - data packets can be lost, or their sequence can be changed.

TCP/IP was developed to provide standardized, reliable data exchange between any numbers of different networks. TCP/IP was developed to provide standardized, reliable data exchange between any numbers of different networks. Although the term is often used as if it were a single concept, a number of protocols are layered together: z. B. IP, TCP, UDP, ARP and ICMP.

#### Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) which runs on top of IP is a connection-oriented transport protocol. It includes error detection and handling mechanisms. Lost telegrams are repeated.

#### User Datagram Protocol (UDP)

UDP is connectionless transport protocol. It provides no control mechanism when exchanging data between sender and receiver. This results in a higher processing speed than, for example, TCP. Checking whether or not the telegram has arrived must be carried out by the higher-level protocol.

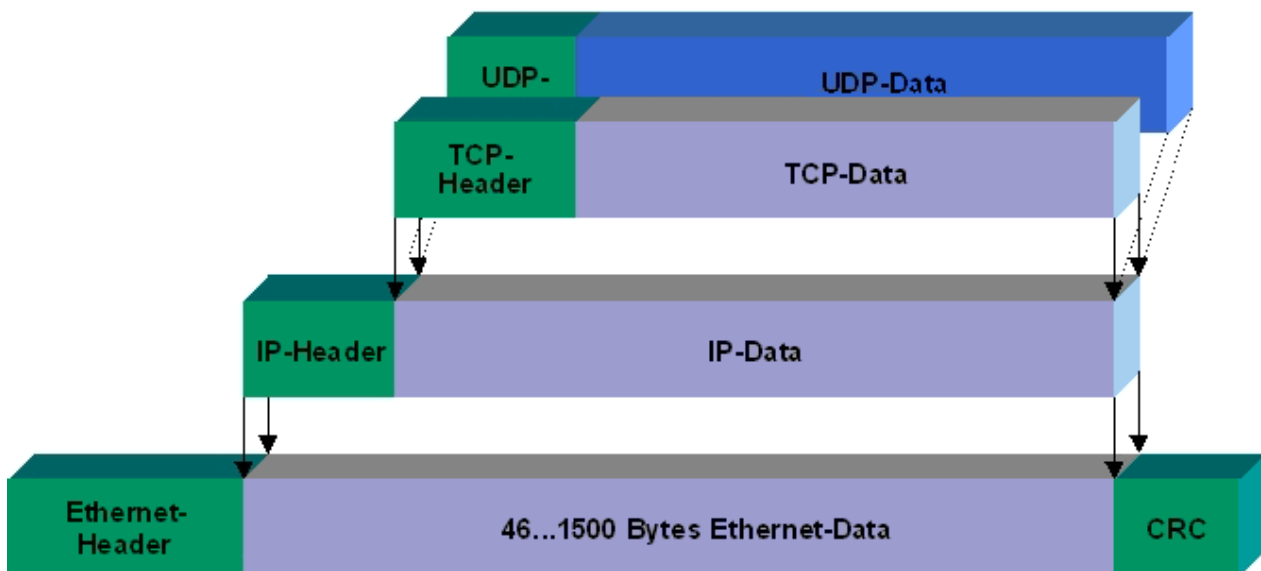


Fig. 155: User Datagram Protocol (UDP)

#### Protocols running on top of TCP/IP and UDP/IP

The following protocols can run on top of TCP/IP or UDP:

- ADS
- ModbusTCP

Both of these protocols are implemented in parallel on the Bus Coupler, so that no configuration is needed to activate the protocols.

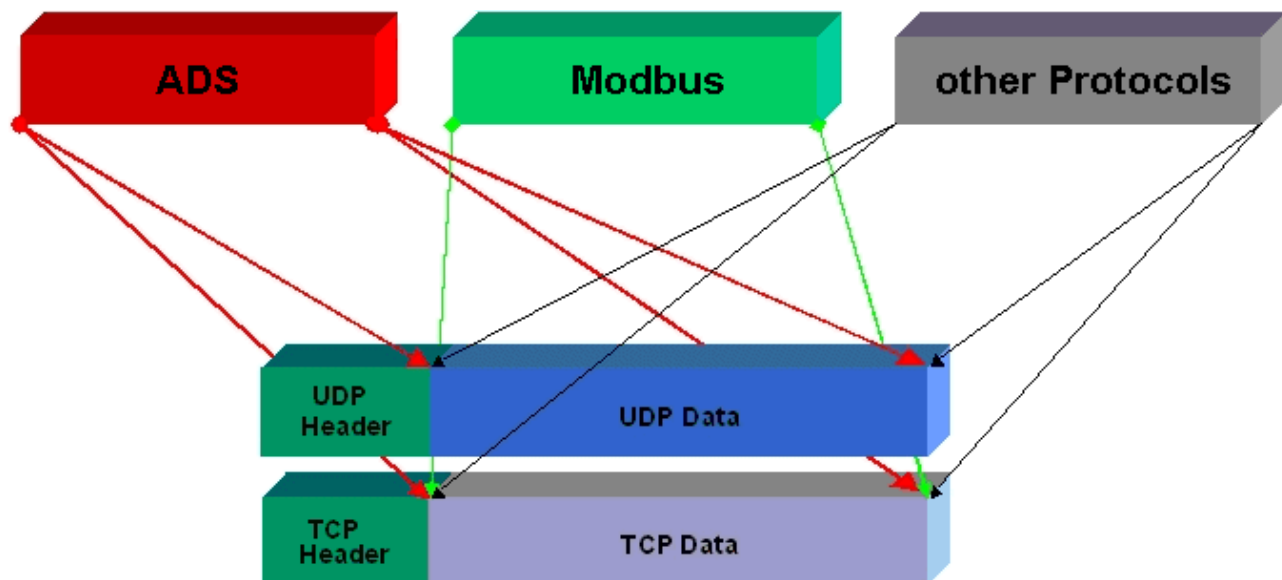


Fig. 156: Protocols running on top of TCP/IP and UDP/IP

ADS can be used on top of either TCP or UDP, but ModbusTCP is always based on TCP/IP.

## 6.1.2 Topology

In 10BaseT and 100BaseT a number of stations are star connected according to the Ethernet standard.

## Star topology

The simplest form of a star LAN consists of a single point-to-point connection. All messages pass via a central node (the hub or switch), which then passes the information to the desired device according to the destination address.

## Tree topology

A tree topology consists of a number of connected star networks. As soon as the network contains a number of hubs or switches, the topology is classified as a tree. Ideally the connections between the star couplers have a particularly wide bandwidth, since these transport the most traffic. When constructing tree topologies, the repeater rule must be observed. This is also known as the 5-4-3 repeater rule. There must be no more than two pairs of repeaters (or of hubs) in the transmission path between any two stations, unless they are separated by bridges, switches or routers. A transmission path may consist of at most five segments and four repeater sets (two repeater pairs). Up to three of these segments may be coaxial segments to which the stations are connected. The remaining segments must consist of point-to-point connections; these are also known as IRL (inter repeater link) connections.

## Cabling guidelines

*Structured cabling* provides general guidelines for constructing the cabling for a LAN. It specifies maximum permitted cable lengths for the wiring within the grounds, building or floor. Standardized in EN 50173, ISO 11801 and TIA 568-A, *structured cabling* provides the basis for an advanced, application-independent and economical network infrastructure. The wiring standards are applicable to a range defined as having a geographical extent of up to 3 km and an office area of up to one million sq meters, with between 50 and 50,000 end devices. Recommendations for the structure of a cabling system are also given. The figures can vary, depending on the topology selected, the transmission media and coupling modules used under industrial conditions, and on the use of components from various manufacturers in one network. The given figures should therefore only be considered as recommendations.

## 6.1.3 Ethernet cable

### Transmission standards

#### 10Base5

The transmission medium for 10Base5 consists of a thick coaxial cable ("yellow cable") with a max. transmission speed of 10 Mbaud arranged in a line topology with branches (drops) each of which is connected to one network device. Because all the devices are in this case connected to a common transmission medium, it is inevitable that collisions occur often in 10Base5.

#### 10Base2

10Base2 (Cheaper net) is a further development of 10Base5, and has the advantage that the coaxial cable is cheaper and, being more flexible, is easier to lay. It is possible for several devices to be connected to one 10Base2 cable. It is frequent for branches from a 10Base5 backbone to be implemented in 10Base2.

#### 10BaseT

Describes a twisted pair cable for 10 Mbaud. The network here is constructed as a star. It is no longer the case that every device is attached to the same medium. This means that a broken cable no longer results in failure of the entire network. The use of switches as star couplers enables collisions to be reduced. Using full-duplex connections they can even be entirely avoided.

#### 100BaseT

Twisted pair cable for 100 Mbaud. It is necessary to use a higher cable quality and to employ appropriate hubs or switches in order to achieve the higher data rate.

**10BaseF**

The 10BaseF standard describes several optical fiber versions.

**Short description of the 10BaseT and 100BaseT cable types**

Twisted-pair copper cable for star topologies, where the distance between two devices may not exceed 100 meters.

**UTP**

Unshielded twisted pair

This type of cable belongs to category 3, and is not recommended for use in an industrial environment.

**S/UTP**

Screened/unshielded twisted pair (screened with copper braid)

Has an overall shield of copper braid to reduce influence of external interference. This cable is recommended for use with Bus Couplers.

**FTP**

Foiled shielded twisted pair (screened with aluminium foil)

This cable has an outer screen of laminated aluminium and plastic foil.

**S/FTP**

Screened/foiled-shielded twisted pair (screened with copper braid and aluminium foil)

Has a laminated aluminium screen with a copper braid on top. Such cables can provide up to 70 dB reduction in interference power.

**STP**

Shielded twisted pair

Describes a cable with an outer screen, without defining the nature of the screen any more closely.

**S/STP**

Screened/shielded twisted pair (wires are individually screened)

This identification refers to a cable with a screen for each of the two wires as well as an outer shield.

**ITP**

Industrial Twisted-Pair

The structure is similar to that of S/STP, but, in contrast to S/STP, it has only one pair of conductors.

## **6.2 ModbusTCP**

### **6.2.1 ModbusTCP Protocol**

The Ethernet protocol is addressed by means of the MAC-ID. The user does not normally need to be concerned about this address. The IP number has a length of 4 bytes, and must be parameterized by the user on the Bus Coupler and in the application. In ModbusTCP, the TCP port is set to 502. The UNIT can be freely selected under ModbusTCP, and does not have to be configured by the user.



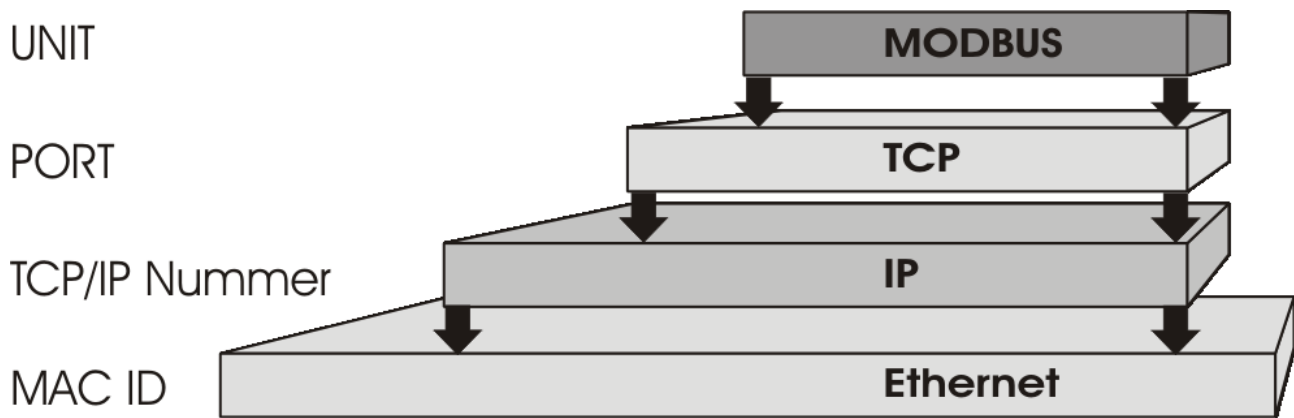


Fig. 157: ModbusTCP Protocol

**TCP port number**

The TCP port number for ModbusTCP has been standardised to 502.

**Modbus-Unit**

The unit is returned by the slave.

**ModbusTCP Protocol**

Byte	Name	Description
0	Transaction identifier	is returned by the slave
1	Transaction identifier	is returned by the slave
2	Protocol identifier	always 0
3	Protocol identifier	always 0
4	Length field	0 (if the message is less than 256 bytes in length)
5	Length field	Number of following bytes
6	UNIT identifier	returned by the slave
7	Modbus	Modbus protocol beginning with the function follows

## 6.2.2 Modbus TCP interface

Address		Description			
0x0000 0x00FF		Process data interface Inputs			
0x0800 0x08FF		Process data interface Outputs			
0x1000 0x1006	Read only	Bus Coupler identification			
0x100A		2 byte PLC interface			
0x100B		Bus terminal diagnosis			
0x100C		Bus Coupler status			
0x1010		Process image length in bits, analog outputs (without PLC variables)			
0x1011		Process image length in bits, analog inputs (without PLC variables)			
0x1012		Process image length in bits, digital outputs			
0x1013		Process image length in bits, digital inputs			
0x1020		Watchdog, current time in [ms]			
0x110A		Read / Write	2 byte PLC interface		
0x110B			Bus terminal diagnosis		
0x1120	Watchdog, pre-defined time in [ms] (Default value: 1000)				
0x1121	Watchdog reset register				
0x1122	Type of watchdog		1	Telegram watchdog (default)	
			0	Write telegram watchdog	
0x1123**	ModbusTCP mode**		1	Fast Modbus	
		0	Normal Modbus (default)		
0x4000* 0x47FF		Flags area (%MB..)*			

\* all Bus Terminal controllers BC9xx0 and BX9000

\*\* for BC9x00 from firmware B7 and BK9000 from firmware B5 and all unlisted BK9xxx and BC/BX9xxx

### Watchdog

The watchdog is active under the factory settings. After the first write telegram the watchdog timer is initiated, and is triggered each time a telegram is received from this device. Other devices have no effect on the watchdog. A second approach, which represents a more sensitive condition for the watchdog, is for the watchdog only to be re-triggered after each write telegram. To do this, write a zero into register 0x1122 (default value "1").

The watchdog can be deactivated by writing a zero to offset 0x1120. The watchdog register can only be written if the watchdog is not active. The data in this register is retained.

### Watchdog register

If the watchdog timer on your slave has elapsed it can be reset by writing twice to register 0x1121. The following must be written to the register: 0xBECF 0xAFFE. This can be done either with function 6 or with function 16.

### The Bus Coupler's status register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	FB	-	-	-	-	-	-	-	-	-	-	-	-	-	CNF	KB

## Legend

Bit	Name	Value	Description
15	FB	1 <sub>bin</sub>	Fieldbus error, watchdog time elapsed
14...2	-	-	reserved
1	CNF	1 <sub>bin</sub>	Bus Coupler configuration error
0	KB	1 <sub>bin</sub>	Bus Terminal error

## ModbusTCP mode

The fast Modbus mode should only be used in small local networks. The fast ModbusTCP is not active under the default settings. If problems are found to occur with this type of communication, the Bus Coupler should be switched to "normal" ModbusTCP communication. The mode is set in the Modbus interface, offset 0x1123. It is necessary to reset the coupler (e.g. using ModbusTCP function 8) after the change. It is not permitted to send more than one Modbus service within one Ethernet frame in fast Modbus mode.

## 2 byte PLC interface

Registers in the complex terminals and Bus Terminal Controller registers can be both read and written using the 2 byte PLC interface. The complex terminal registers are described in the associated terminal documentation. The Bus Coupler registers can be used, for example, to read terminal bus diagnostics data, the terminal composition or the cycle times, and the programmed configuration can be written. It is also possible for a manual K-bus reset to be carried out. The 2-byte PLC interface requires two bytes each of input and output data. They are handled using a special protocol. A description of the 2 byte PLC interface, the registers available in the Bus Couplers and of function blocks for various PLC systems that support the 2 byte PLC interface can be supplied on request.

## 2 byte diagnostic interface

The terminals' error messages can be sent over the 2-byte diagnostic interface. K-bus diagnostics must however be activated for this purpose. The 2-byte diagnostic interface occupies two bytes each of input and output data. A special protocol is processed via these two bytes. A description of the 2 byte-diagnostic interface can be supplied on request.

### 6.2.3 ModbusTCP slave error answer (BK9000, BX/BC9xx0, IP/ILxxxx-B/C900, EK9000)

When the user sends the slave either a request or information that the coupler does not understand, the slave responds with an error report. This answer contains the function and the error code. 0x80 is added to the value returned by the function.

Code	Name	Meaning
1	ILLEGAL FUNCTION	Modbus function not implemented
2	ILLEGAL DATA ADDRESS	Invalid address or length
3	ILLEGAL DATA VALUE	Invalid parameter - Diagnostic functions - Incorrect register
4	SLAVE DEVICE ERROR	Watchdog or K-bus error EK9000: E-bus error
6	SLAVE DEVICE BUSY	Output data is already been received from another IP device

## 6.2.4 ModbusTCP functions

### 6.2.4.1 Read holding register (Function 3)

The *Read holding register* function can be used to read the input and output words and the registers. Inputs from offset 0 - 0xFF and outputs from offset 0x800 - 0x8FF, and for controllers (BC, BX) the flag area from offset 0x4000.

In this example the first two analog outputs (or two output words) are read. The analog outputs (or output words) start at offset 0x800. The length indicates the number of channels (or words) to be read.

#### Query

Byte name	Sample
Function code	3
Start address high	8
Start address low	0
Count high	0
Count low	2

The fieldbus coupler answers with byte count 4, i.e. 4 bytes of data are returned. The query was for two analog channels, and these are distributed over two words. In the analog output process image, the first channel has the value 0x3FFF, while the second channel has the value 0x0.

#### Response

Byte name	Sample
Function code	3
Byte count	4
Data 1 high byte	63
Data 1 low byte	255
Data 2 high byte	0
Data 2 low byte	0

### 6.2.4.2 Read input register (Function 4)

The function *Read input register* reads the inputs on a word basis.

In this example the first two analog inputs (or the first two input words) are read. The analog inputs (or input words) start at an offset of 0x0000. The length indicates the number of words to be read. A KL3002, for example, has two words of input data. Therefore, the length to be entered at *Number low* is two.

#### Query

Byte name	Sample
Function code	4
Start address high	0
Start address low	0
Count high	0
Count low	2

The fieldbus coupler answers with byte count 4, i.e. four bytes of data are returned. The query was for two analog channels, and these are now distributed over 2 words. In the analog input process image, the first channel has the value 0x0038, while the second channel has the value 0x3F1B.

**Response**

Byte name	Sample
Function code	4
Byte count	4
Data 1 high byte	0
Data 1 low byte	56
Data 2 high byte	63
Data 2 low byte	11

**6.2.4.3 Preset single register (Function 6)**

The function *Preset singles register* can be used to access the output or flag process image (only for controllers) and the [Modbus TCP interface](#) [► 178].

Function 6 writes the first output word. The outputs start at an offset of 0x0800. Here again the offset always describes a word. This means offset 0x0803 refers to the fourth word in the output process image.

**Query**

Byte name	Sample
Function code	6
Start address high	8
Start address low	0
Data high	63
Data low	255

The Fieldbus Coupler replies with the same telegram and confirmation of the received value.

**Response**

Byte name	Sample
Function code	6
Start address high	8
Start address low	0
Data high	63
Data low	255

**6.2.4.4 Preset single register (Function 16)**

The *Preset multiple register* function can be used to write a number of outputs. The first two analog output words are written in this example. The outputs start at an offset of 0x0800. Here the offset always describes a word. Offset 0x0003 writes to the fourth word in the output process image. The length indicates the number of words, and the *Byte count* is formed from the combination of all the bytes that are to be written.

Sample: 4 words - correspond to a byte count of 8

The data bytes contain the values for the analog outputs. In this example, two words are to be written. The first word is to receive the value 0x7FFF, and the second word is to receive the value 0x3FFF.

**Query**

Byte name	Sample
Function code	16
Start address high	8
Start address low	0
Length high	0
Length low	2
Byte count	4
Data 1 byte 1	127
Data 1 byte 2	255
Data 2 byte 1	63
Data 2 byte 2	255

**Response**

The coupler replies with the start address and the length of the transmitted words.

Byte name	Sample
Function code	16
Start address high	8
Start address low	0
Length high	0
Length low	2

**6.2.4.5 Read / write registers (Function 23)**

A number of analog outputs can be written and a number of analog inputs read with one telegram using the *Read / write registers* function. In this example the first two analog output words are written, and the first two analog inputs are read. The analog outputs start at offset 0x0800, while the inputs start at offset 0x0000. Here the offset always describes a word. Offset 0x0003 writes to the fourth word in the output process image. The length indicates the number of words, and the *Byte count* is formed from the combination of all the bytes that are to be written. Sample: 4 words - correspond to a byte count of 8

The data bytes contain the values for the analog outputs. In this example, two words are to be written. The first word is to receive the value 0x3FFF, and the second word is to receive the value 0x7FFF.

**Query**

Byte name	Sample
Function code	23
Read start address high	0
Read start address low	0
Read length high	0
Read length low	2
Write start address high	8
Write start address low	0
Write length high	0
Write length low	2
Byte count	4
Data 1 high	63
Data 1 low	255
Data 2 high	127
Data 2 low	255

## Response

The coupler replies with the start address and the length of the bytes to be transferred in *Byte count*. The data information follows. In this example the first word contains 0x0038 while the second word contains 0x3F0B.

Byte name	Sample
Function code	23
Byte count	4
Data 1 high	0
Data 1 low	56
Data 2 high	63
Data 2 low	11

## 6.3 ADS-Communication

### 6.3.1 ADS-Communication

The ADS protocol (ADS: Automation Device Specification) is a transport layer within the TwinCAT system. It was developed for data exchange between the different software modules, for instance the communication between the NC and the PLC. This protocol enables communication with other tools from any point within the TwinCAT. If communication with other PCs or devices is required, the ADS protocol can use TCP/IP as a basis. Within a networked system it is thus possible to reach all data from any point.

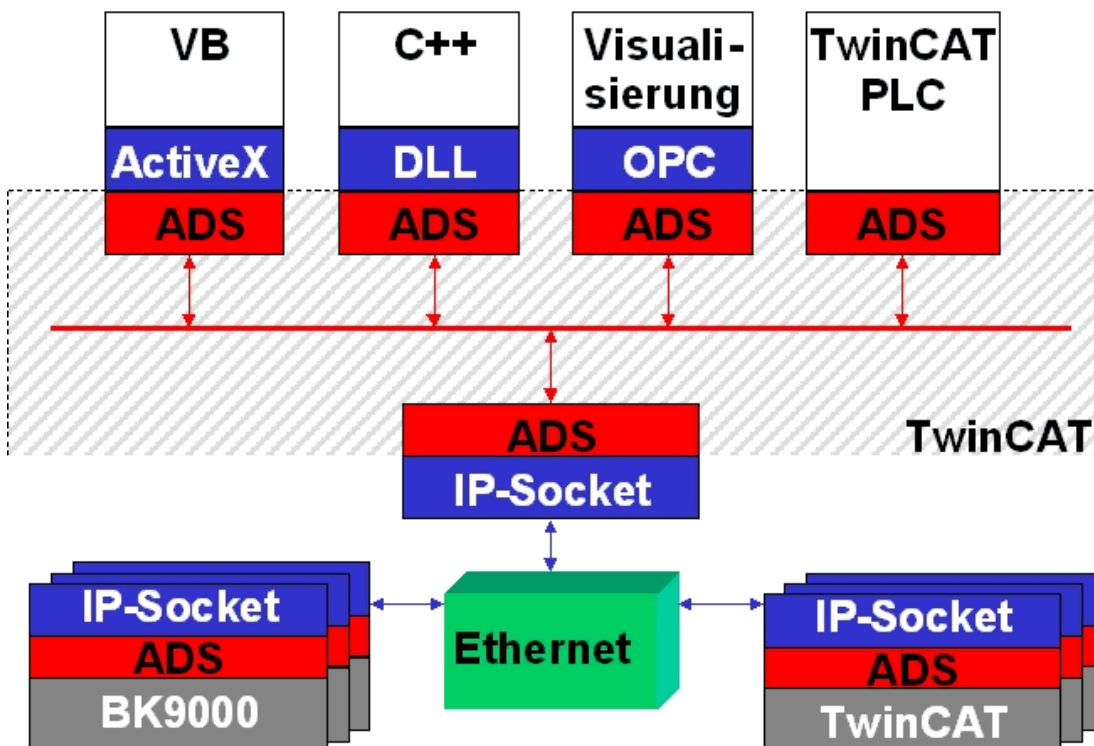


Fig. 158: The ADS protocol as a transport layer within the TwinCAT system

The ADS protocol runs on top of the TCP/IP or UDP/IP protocols. It allows the user within the Beckhoff system to use almost any connecting route to communicate with all the connected devices and to parameterize them. Outside the Beckhoff system a variety of methods are available to exchange data with other software tools.

## Software interfaces

### ADS-OCX

The ADS-OCX is an Active-X component. It offers a standard interface to, for instance, Visual Basic, Delphi, etc.

### ADS-DLL

You can link the ADS-DLL (DLL: Dynamic Link Library) into your C program.

### OPC

The OPC interface is a standardised interface for communication used in automation engineering. Beckhoff offer an OPC server for this purpose.

## 6.3.2 ADS protocol

The ADS functions provide a method for accessing the Bus Coupler information directly from the PC. ADS function blocks can be used in TwinCAT PLC Control for this. The function blocks are contained in the *TcSystem.lib* library. It is also equally possible to call the ADS functions from AdsOCX, ADSDLL or OPC. It is possible to access all the data through ADS port number 300, and to access the registers of the Bus Coupler and Bus Terminals through ADS port number 100.

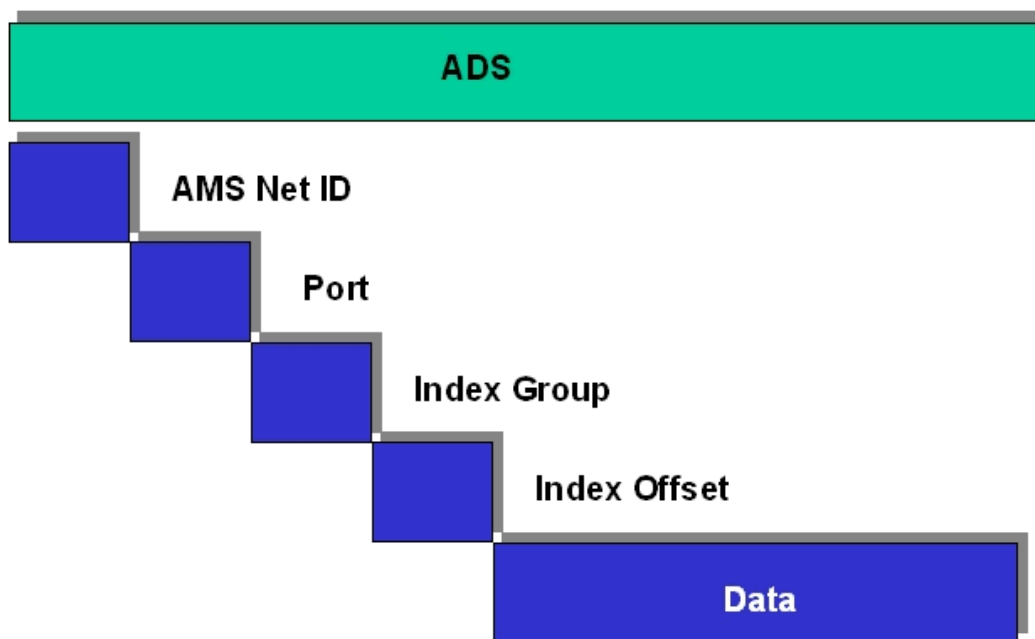


Fig. 159: Structure of the ADS protocol

### AMSNetID

The AMSNetID provides a reference to the device that is to be addressed. This is created from the set TCP/IP address and an additional 2 bytes. These additional 2 bytes consist of "1.1", and cannot be altered.

Sample:

IP address 172.16.17.128

AMSNetID 172.16.17.128.1.1

### Port number

The port number distinguishes sub-elements in the connected device.

Port 100: register access

Port 300: fieldbus process data

Port 800: local process data (BC90x0, C900 only)



**Index group**

The index group distinguishes different data within a port.

**Index offset**

Indicates the offset, from which reading or writing the byte is to start.

**Len**

Gives the length of the data, in bytes, that is to be read or written.

**TCP port number**

The TCP port number for the ADS protocol is 48898 or 0xBF02.

### 6.3.3 ADS services

**Local Process Image PLC Task 1 Port 800/801**

Data can be read from and written to the local process image. If it is necessary for outputs to be written, it is important to ensure that they are not used by the local PLC, because the local controller will overwrite these values. The data are not associated with a watchdog, and therefore must not be used for outputs that would have to be switched off in the event of a fault.

Index group	Meaning	Index offset (value range)
0xF020	Inputs	0...2047
0xF021	Bit inputs	0...16376
0xF030	Outputs	0...2047
0xF031	Bit outputs	0...16376
0x4020	Flags	0...4095
0x4021	Flag bit	0...32760

**ADS services****AdsServerAdsState**

Data type (read only)	Meaning
String	Start - the local PLC is running Start - the local PLC is stopped

**AdsServerDeviceState**

Data type (read only)	Meaning
INT	0 - Start - the local PLC is running 1 - Stop - the local PLC is stopped

**AdsServerType**

Data type (read only)	Meaning
String	BX PLC Server

**ADSWriteControl**

Data type (write only)	Meaning
NetID	Net ID of the Ethernet Controller*
Port	800
ADSSTATE	5 - RUN / 6 - STOP
DEVSTATE	0
LEN	0
SRCADDR	0
WRITE	rising edge starts the function block
TMOUT	example: t#1000ms

\* BC9050, BC9020, BC9120, BX9000

**Register Access Port 100**

On the Bus Terminal Controllers of the BX series, and on the BCxx50/xx20, the ADS port number for register communication is fixed at 100.

Index group	Index offset (value range)		Meaning
	Hi-Word	Low Word	
0 [READ ONLY]	0...127	0...255	Registers in the Bus Coupler Hi-Word, table number of the Bus Coupler Lo-Word, register number of the table
1...255	0...3	1...255	Register of the Bus Terminals Hi-Word, channel number Lo-Word, register number of the Bus Terminal

**Note****Register communication with K-bus Bus Terminals**

For reading the registers, ensure that the timeout for the ADS function block is set to more than one second.

**Note****Writing of registers to the K-bus Bus Terminals**

When writing to the registers, the password has to be set (see the documentation for the particular Bus Terminal).

## 7 Error handling and diagnosis

### 7.1 Diagnosis

#### Ethernet state (FieldbusState)

In many cases it is important to know whether the communication with the higher-level master is still OK. To this end, link the *FieldbusState* variable with your PLC program.

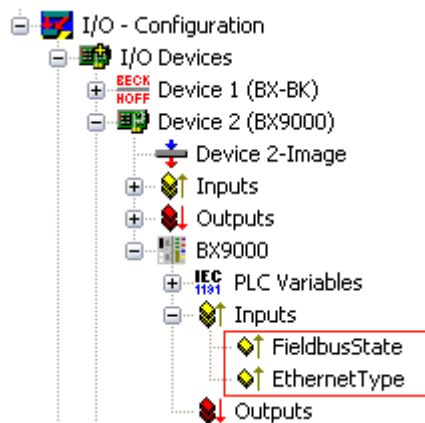


Fig. 160: Ethernet fieldbus state

#### FieldbusState

Error number	Description	Cause
0	No error	-
1	Watchdog error	Communication interrupted

#### EthernetType

Here it is possible to determine which Ethernet protocol is accessing the PLC variables, and thereby triggering the watchdog (for example, the data in the Default Config area starting from addresses %IB1000 and %QB1000).

Diagnostic number	Description	Protocol
0x0000	No protocol is accessing the PLC variables	-
0x0001	ADS TCP	Communication via ADS TCP/IP
0x0002	ADS UDP	Communication via ADS UDP/IP
0x0010	ModbusTCP	Communication via Modbus TCP/IP
0x0011	ModbusUDP	Communication via Modbus UDP

#### Reading fieldbus state by ADS

In default or TwinCAT configuration the fieldbus status can be read via ADSREAD.

Parameter ADSREAD function block	Description
NetID	local – empty string
Port	1
IndexGroup	6
IndexOffset	0x000C_A200
LEN	4

**First word (FieldbusState)**

Error number	Description	Cause
0x0000	No error	-
0x0001	Watchdog error	Communication interrupted

**Second word (EthernetType)**

Diagnostic number	Description	Protocol
0x0001	ADS TCP	Communication via ADS TCP/IP
0x0002	ADS UDP	Communication via ADS UDP/IP
0x0010	ModbusTCP	Communication via Modbus TCP/IP
0x0011	ModbusUDP	Communication via Modbus UDP

**State of the K-Bus**

An internal bus or Bus Terminal error is indicated in the K-Bus state. A more precise fault description can be obtained via a function block (in preparation). To this end, link the *K-Bus state* variable with your PLC program.

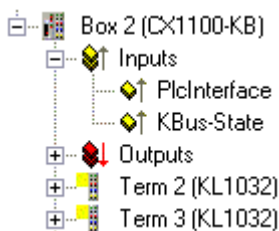


Fig. 161: K-bus status

Error bit	Description	Error type
0	No error	No ERROR.
Bit 0	K-Bus error	ERROR
Bit 2	K-Bus is re-triggered	NOTICE

**Reading K-Bus state by ADS**

In the default configuration or in the TwinCAT configuration the K-bus status can be read via ADSREAD.

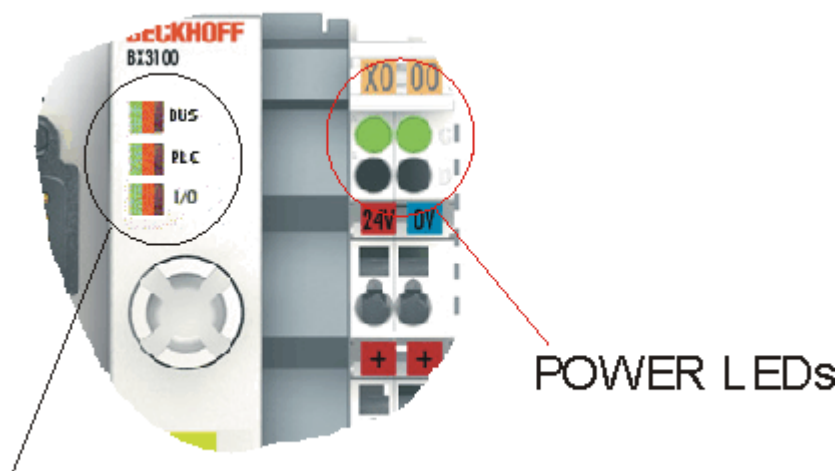
Parameter ADSREAD function block	Description
NetID	local – empty string
Port	1
IndexGroup	16#0006
IndexOffset	16#000C_9000
LEN	1

## 7.2 Diagnostic LEDs

The Bus Coupler has two groups of LEDs for the display of status. The "DIAG LEDs" on the left indicate the fieldbus, PLC and K-Bus state. Two green LEDs (Power LEDs) are located in the top right corner of the Bus Coupler. They are used for displaying the supply voltage of the BX Controller and the 24 V<sub>DC</sub> supply of the power contacts.

## LEDs for power supply diagnostics

LED (Power LEDs)	Meaning
Left LED off	Bus Coupler has no voltage 24 V <sub>DC</sub>
Right LED off	No power supply 24 V <sub>DC</sub> connected at the power contacts



## DIAG LEDs

Fig. 162: Diagnostic LEDs for the fieldbus, the PLC, the K-bus and the power supply units

The DIAG LEDs are sub-divided as follows:

- Bus: Fieldbus diagnosis
- PLC: PLC diagnosis
- I/O: K-bus diagnosis

The LEDs can be off, green, orange or red.



Fig. 163: Diagnostic LEDs for the fieldbus, the PLC and the K-bus

After switching on, the Bus Coupler immediately checks the connected configuration. The I/O LED goes out if the start-up was successful. A red I/O LED indicates a Bus Terminal error. The error type is shown in the display. This permits rapid rectification of the error.

**LED bus - fieldbus diagnosis**

LED	Meaning
LED off	no fieldbus connected, Bus Coupler searches for baud rate
LED red	error flashing - error type - display
LED orange	Bus Coupler has found baud rate, waiting for config and parameter data
LED green	Bus communication OK, BX Controller is exchanging data

**LED PLC - PLC diagnosis**

LED	Meaning
LED off	PLC stop or no program available
LED red	Flashing - the set task time is exceeded from time to time On - the set task time is always exceeded Disable the red LED
LED orange	PLC running without boot project (only during the on cycle), LED flashes orange during creating of the boot project
LED green	Boot project - PLC running (only during the on cycle)

**LED I/O - K-Bus diagnosis**

LED	Meaning
LED off	No data are exchanged via the K-Bus
LED red	error flashing - error type - display
LED orange	Register or KS2000 online access
LED green	K-Bus OK and running

Error codes for K-Bus diagnosis

Error code	Error argument	Description	Remedy
0	-	EMC problems	<ul style="list-style-type: none"> <li>Check power supply for undervoltage or overvoltage peaks</li> <li>Implement EMC measures</li> <li>If a K-Bus error is present, it can be located by a restart (switching the BX controller off and on again)</li> </ul>
1	0	EEPROM checksum error	Enter factory settings with the KS2000 configuration software
	1	Code buffer overflow	Insert fewer Bus Terminals. Too many entries in the table for the programmed configuration
	2	Unknown data type	Software update required for the BX Controller
2	-	Reserve	-
3	0	K-bus command error	<ul style="list-style-type: none"> <li>No Bus Terminal inserted</li> <li>One of the Bus Terminals is defective; halve the number of Bus Terminals attached and check whether the error is still present with the remaining Bus Terminals. Repeat until the defective Bus Terminal is located.</li> </ul>
4	0	K-Bus data error, break behind the BX Controller	Check whether the n+1 Bus Terminal is correctly connected; replace if necessary.
	n	Break behind Bus Terminal n	Check whether the Bus End Terminal KL9010 is connected.
5	n	K-bus error in register communication with Bus Terminal n	Exchange the nth bus terminal
6	0	Error at initialization	Replace the BX controller
	1	Internal data error	Hardware reset of the BX controller (switching off and on again)
	2	DIP switch changed after a software reset	Hardware reset of the BX controller (switching off and on again)
	3	IP address collision	Check whether the IP address already exists in the network.
7	0	Note: cycle time was exceeded	Warning: the set cycle time was exceeded. This indication (flashing LEDs) can only be cleared by booting the BX Controller again. Remedy: increase the cycle time
9	0	Checksum error in Flash program	Transfer the program to the BX controller again
	1	Incorrect or faulty library implemented	Remove the faulty library
10	n	Bus Terminal n is not consistent with the configuration that existed when the boot project was created	Check the nth Bus Terminal. The boot project must be deleted if the insertion of an nth Bus Terminal is intentional
14	n	nth Bus Terminal has the wrong format	Start the BX Controller again, and if the error occurs again then exchange the Bus Terminal.
15	n	Number of Bus Terminals is no longer correct	Restart the BX controller. If the error occurs again, reset the BX controller to the delivery state with the configuration software.
16	n	Length of the K-bus data is no longer correct	

## 7.3 Diagnostics display

During start-up, the display shows the current firmware version for approx. three seconds.

If an error occurs during start-up, this will be indicated via a flash sequence of the associated LED.

Configuration errors are shown in the display via TC-Config and an error number. In this case, please use the System Manager to check your hardware configuration or contact support.

Display	Meaning
TC-Config 0xE02E	A complex Bus Terminal is assigned a bit address. Check the TwinCAT configuration.
TC-Config 0xF0nn	Bus Terminal no. nn does not correspond to the configuration. Compare the bus structure of Bus Terminal no. nn with the configuration.
TC-Config 0xC0nn	Bus Terminal no. nn does not correspond to the configuration. Compare the bus structure of Bus Terminal no. nn with the configuration.

Firmware errors are shown in the display via FW-Error and an error number. Please contact support.

Display	Meaning
FW-Error 0xnxxx	Please contact support



## 8 Appendix

### 8.1 BX9000 - First steps

#### Change the IP address

Default IP address: 172.16.21.20

SubNetMask: 255.255.0.0

Default gateway: 0.0.0.0



Fig. 164: Navigation switch

To switch to the menu, press the navigation switch for three seconds. The menu directory appears.

- Use the RIGHT/LEFT keys to switch between the menu items (row 1 of the display shows the active menu level).
- Press the DOWN key to change to a submenu.
- Press the UP key to return to the main menu.

Row 1 of the submenu shows the menu item, row 2 shows the current setting of the menu item.

Some entries cannot be changed (*read only*). These entries enable verification or provide information for the user. To exit the menu display, return to the main menu and press the navigation button for three seconds.

A password must be specified before changes can be saved. The password is retained during firmware updates or a reset to the delivery state. Should you forget the password, you have to return the BX controller for resetting.

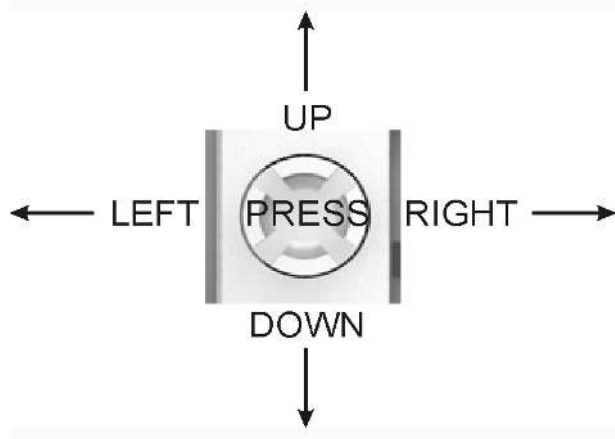
**Switch assignment**

Fig. 165: Switch assignment

**Code**

The factory setting is "\*\*\*\*", i.e. no password is set. A password is required for setting parameters.

**Menu navigation**

Open the menu directory by pressing the navigation switch for three seconds. Some of the menu items are explained below.

Table 2: MENU

<b>MENU</b>	Main menu	
	DOWN (press briefly)	
<b>PASSWORD</b>	????	???? - password set
	????	**** - no password set
	PRESS (press briefly)	
<b>PASSWORD ENTER?</b>	????	Do you want to set a password?
		Yes - (press for approx. two seconds, then enter the password) / No - press UP
	PRESS (press for approx. two seconds, then enter the password)	
<b>PASSWORD</b>	????	Enter password
		Press <OK>
	PRESS	
<b>PASSWORD</b>	1234	Enter password
		Press OK and then OK again to confirm (press for approx. one second until OK appears on the display)
	Press UP	
<b>MENU</b>	Main menu	
	Press RIGHT	
<b>AMS</b>	AMS menu	
	Press RIGHT	
<b>ETHERNET</b>	ETHERNET menu	
	Press DOWN	
	<b>MAC ID</b>	ETHERNET menu / MAC ID
	000105-xx-xx-xx	xx-xx-xx is your MAC ID
	Press LEFT	
	<b>IP ADDRESS</b>	ETHERNET menu / IP ADDRESS
	172.16.21.20	Default IP address
	Press DOWN to enter the IP address	
	<b>MODIFY</b>	ETHERNET menu / IP ADDRESS / Edit IP Address
	172.16.21.20	Press DOWN to increment down (-1), press UP to increment up (+1), press RIGHT for the next digit
	Press the button for two seconds until ACCEPT? appears	
	<b>ACCEPT ?</b>	ETHERNET menu / IP ADDRESS / Edit IP Address
	172.16.44.44	Press DOWN to increment down (-1), press UP to increment up (+1), press RIGHT for the next digit
	Press the button when ACCEPTED appears	
	<b>ACCEPTED</b>	ETHERNET menu / IP ADDRESS / Edit IP Address
	172.16.44.44	Press DOWN to increment down (-1), press UP to increment up (+1), press RIGHT for the next digit
	Press UP to exit the IP address	
	<b>ETHERNET</b>	ETHERNET menu
	Press for three seconds to exit the menu	
	Next step	
	Install TwinCAT 2.10 (can be found on the BX CD)	
	Add the BX controller in the AMS router	

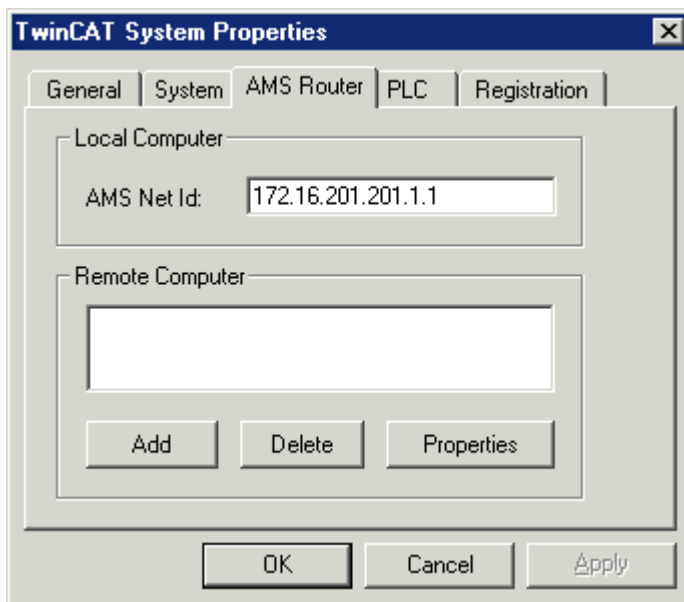


Fig. 166: Adding the BX controller in the AMS router

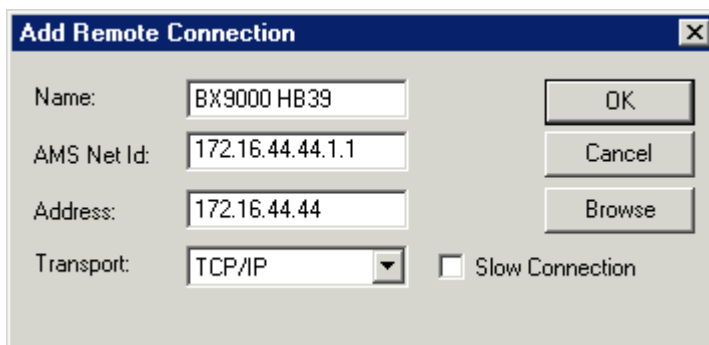


Fig. 167: Adding the remote connection

Restart TwinCAT - you can now program the BX9000.

Start the PLC Control program and create a new project.

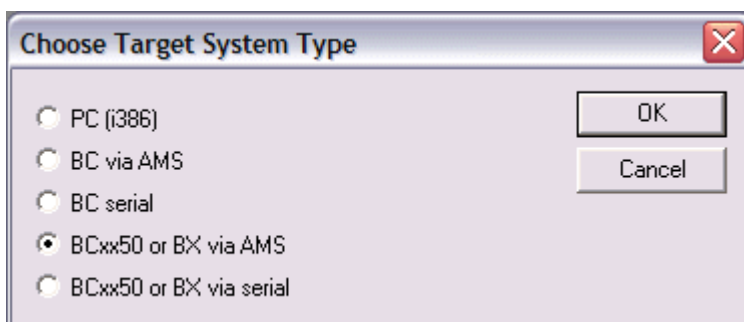


Fig. 168: Choose Target System

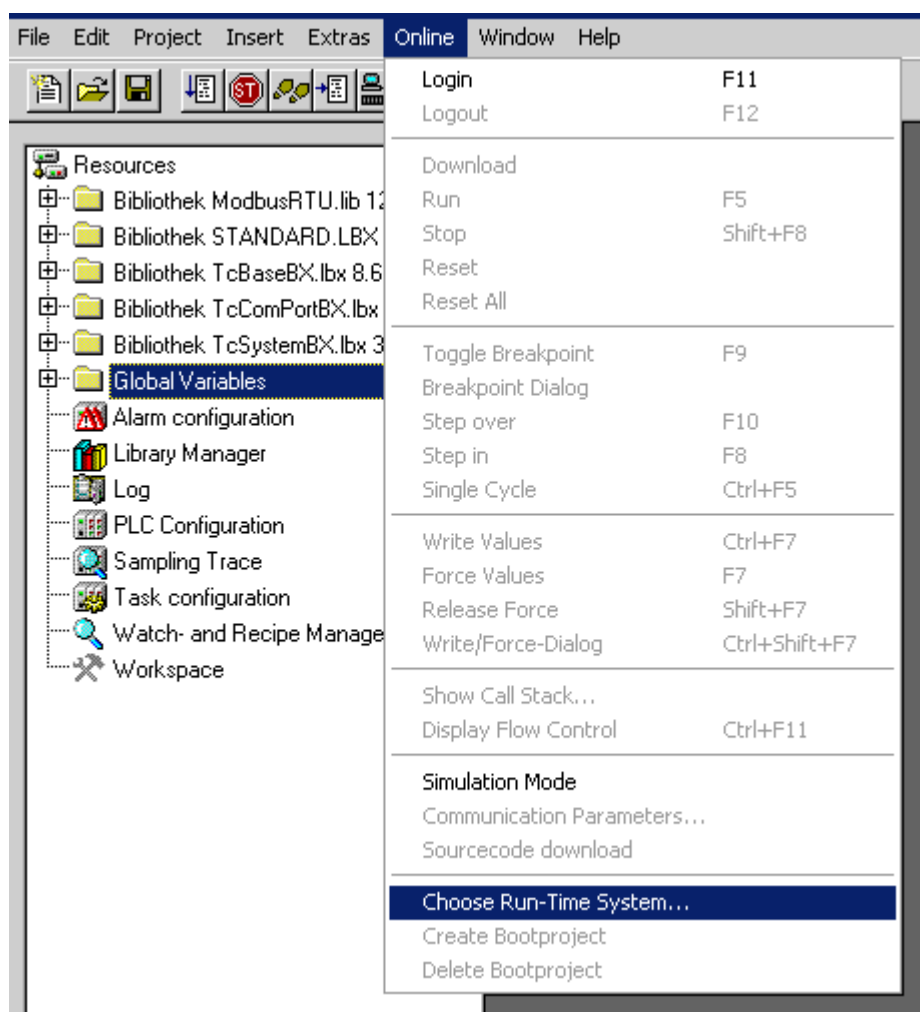


Fig. 169: Selecting the runtime system (step 1)

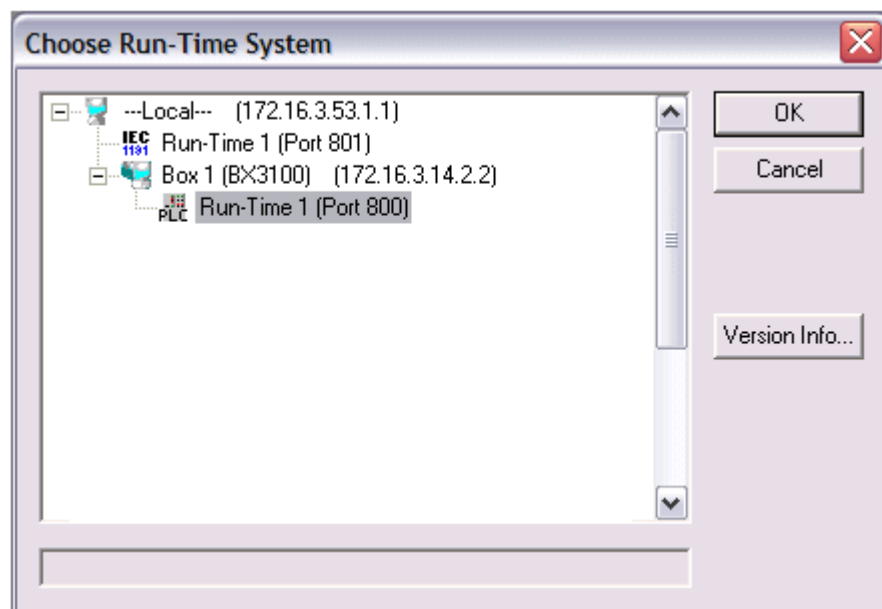


Fig. 170: Selecting the runtime system (step 2)

## 8.2 Switching between controllers

### Switching from BCxx00 to BCxx50/BCxx20

#### File names

In the Bus Terminal controllers of the BCxx50 and BCxx20 series, libraries have the extension \*.lbx, programs have the extension \*.prx.

#### Flag variables

The allocated flag variables

- of the BCxx00 are assigned %MB0...%MB511 (except BC9000/BC9100: %MB0...%B4095).
- of the BCxx20 are assigned %MB0...%MB4095
- of the BCxx50 are assigned %MB0...%MB4095

Status information such as K-bus/fieldbus status and cycle tick is not copied to the BCxx50/BCxx20. This information is available in TcSystemBCxx50.lbx as a function for the BCxx50/BCxx20.

The allocated flags do **not** act as retain variables.

#### Retain data

The retain data have to be declared as VAR RETAIN [► 94]. Up to 2 kbyte are available.

#### PLC Variables

In the Default-Config the PLC variables start from %IB1000 and %QB1000.

#### Large model

Not applicable for BCxx50 and BCxx20.

Max. memory:

- BCxx50: 48 kbyte
- BCxx20: 128 kbyte

#### Task time

The task time is specified in the PLC Control. It should be set to a realistic value (measuring of PLC cycle time and K-Bus). The background time is not used.

#### Task configuration

A maximum of one task is available. This task must be configured.

#### PLC and fieldbus terminals

For the standard Bus Terminal Controllers (BCxx00) it was possible to select whether a Bus Terminal is assigned to the fieldbus or the local PLC.

In the default configuration of the BCxx50/BCxx20 all Bus Terminals are assigned to the local PLC. An assignment to the fieldbus is not possible in this case.

### Switching from BCxx00 to BXxx00

#### File names

In the Bus Terminal controllers of the BCxx00, libraries have the extension \*.lbx, programs have the extension \*.prx.

## Flag variables

The allocated flag variables

- of the BCxx00 are assigned %MB0...%MB511 (except BC9000/BC9100: %MB0...%B4095).
- of the BXxx00 are assigned %MB0...%MB4095

Status information such as K-bus/fieldbus status and cycle tick is not copied to the BXxx20. This information is available in TcSystemBCxx00.lbx as a function for the BXxx50.

The allocated flags do **not** act as retain variables.

## Retain data

The retain data have to be declared as `VAR_RETAIN` [► 94]. Up to 2 kbyte are available.

## PLC Variables

In the Default-Config the PLC variables start from %IB1000 and %QB1000.

## Large model

Not applicable for BXxx00. Max. memory: 256 kbyte.

## Task time

The task time is specified in the PLC Control. It should be set to a realistic value (measuring of PLC cycle time and K-Bus). The background time is not used.

## Task configuration

A maximum of one task is available. This task must be configured.

## PLC and fieldbus terminals

For the standard Bus Terminal Controllers (BCxx00) it was possible to select whether a Bus Terminal is assigned to the fieldbus or the local PLC.

In the default configuration of the BXxx00 all Bus Terminals are assigned to the local PLC. An assignment to the fieldbus is not possible in this case.

## Switching from PC to BCxx50/BCxx20/BXxx00

### File names

In the Bus Terminal controllers of the BCxx50/BCxx20 and BXxx00 series, libraries have the extension \*.lbx, programs have the extension \*.prx.

### Allocated variables

For the Bus Terminal controllers of the BCxx50/BCxx20 and BXxx00 series, a limited number of allocated data are available:

- inputs 2 kbyte, %IB0...2048
- outputs 2 kbyte, %QB0...2048
- flags 4 kbyte, %MB0...4095

### Task configuration

A maximum of one task is available. A sensible task time should be selected. Adjust the task time to your application by measuring the required system time (PLC + K-Bus + fieldbus + other).

## Retain data

For the Bus Terminal controllers of the BCxx50, BCxx20 and BXxx00 series, up to 2 kbyte of retain data are available. Ensure that no (or only very few) retain data are used in function blocks (see [RETAIN data](#) [► 94]).


## 8.3 Firmware-Update BX9000

### Firmware update via Ethernet

The TwinCAT System Manager can be used to load new firmware to the BX9000 via Ethernet.  
Requirements:

TwinCAT 2.10 build 1251  
BX9000 firmware 1.07 or above  
Working Ethernet connection (from the PC to the BX9000).

Before you update the BX9000, delete the boot project and start the BX9000 without a PLC program.

Switch TwinCAT to Config mode  and add a virtual Ethernet interface as a device.

Then insert the BX9000 and set the IP address.

The next step is to click the BX9000 with the right mouse button, and select the *Firmware update* via ADS function.

Then select the appropriate firmware. The update is automatic. The BX9000 will restart automatically when the update is complete.

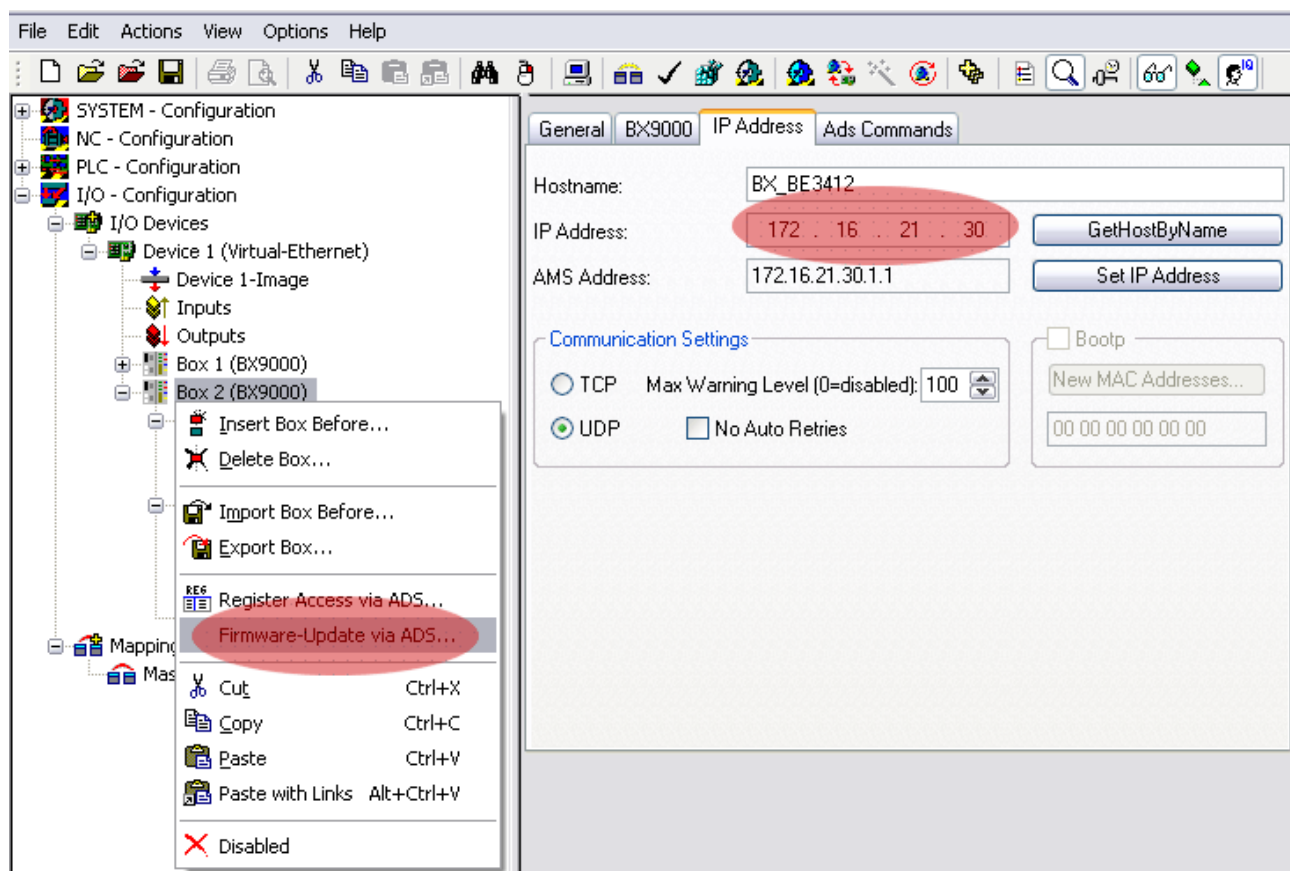


Fig. 171: Firmware update via Ethernet



## 8.4 Sample programs - overview

Denomination	Description
<a href="#">Display [► 120]</a>	Example for controlling the display
<a href="#">Navigation switch [► 119]</a>	Reading of the navigation switch from the PLC
<a href="#">Menu [► 90]</a>	Example for own menu with navigation switch and display
<a href="#">RTC [► 83]</a>	Example for reading the real-time clock (RTC) via function blocks
<a href="#">COM port - BK/BC8x00 master interface [► 125]</a>	COM1 or COM2 interface as master with the BK8x00 protocol
<a href="#">COM port - BK8x00 slave interface [► 126]</a>	COM1 or COM2 interface as slave with the BK8x00 protocol
<a href="#">COM port - Cimrex 12 [► 132]</a>	Example for controlling a Cimrex 12 display via ModbusRTU
<a href="#">COM port - ModbusRTU slave [► 129]</a>	Link of ModbusRTU Lib with the COM 1 or COM 2 interface of the BX
<a href="#">COM port - ModbusRTU master [► 131]</a>	Link of ModbusRTU Lib with the COM 1 or COM 2 interface of the BX
<a href="#">COM port - RK512 protocol [► 134]</a>	RK512 protocol via COM 1 or COM 2
<a href="#">COM port - text message via COM port [► 135]</a>	Connecting a Siemens S35 mobile phone to the COM interface for sending text messages
<a href="#">COM port - COMlibV2 [► 131]</a>	Sending and receiving strings with COMlibV2
<a href="#">SSB - Display [► 73]</a>	Cimrex panel at SSB
<a href="#">SSB - AX2000 [► 71]</a>	AX2000 at SSB
<a href="#">SSB - BK51x0 [► 69]</a>	BK5120 at SSB
<a href="#">SSB - BX / BX communication [► 69]</a>	Communication between BXs (via SSB)
<a href="#">SSB - IclA Drive [► 74]</a>	IclA drive at SSB
<a href="#">SSB - Lenze Drive [► 80]</a>	Lenze frequency converter at SSB

## 8.5 Sample programs for BX9000 - Overview

Denomination	Description
<a href="#">Sending email [► 158]</a>	Sending an SMTP to an email server
<a href="#">Provide HTML page [► 167]</a>	Displaying values on an HTML page
<a href="#">Reading the IP configuration [► 164]</a>	Reading the Ethernet settings
<a href="#">ModbusTCP client [► 155]</a>	BX9000 as ModbusTCP client (master)
<a href="#">UDP communication [► 153]</a>	Communication with an Ethernet device via UDP
<a href="#">Reading the time via SNTP [► 160]</a>	Reading the time via SNTP

## 8.6 General operating conditions

The following conditions must be met in order to ensure flawless operation of the fieldbus components.

### Environmental conditions

#### Operation

The components may not be used without additional protection in the following locations:

- in difficult environments, such as where there are corrosive vapors or gases, or high dust levels
- in the presence of high levels of ionizing radiation

Condition	Permissible range
Permissible ambient temperature during operation	See technical data
Permissible ambient temperature during operation	-25°C ... +85 °C
Installation position	variable
Vibration resistance	conforms to EN 60068-2-6
Shock resistance	conforms to EN 60068-2-27, EN 60068-2-29
EMC immunity	conforms to EN 61000-6-2
Emission	conforms to EN 61000-6-4

### Transport and storage

Condition	Permissible range
Permissible ambient temperature during storage	-25°C... +85 °C
Relative humidity	95 %, no condensation
Free fall	up to 1 m in the original packaging

### Protection classes and types

Condition	Permissible range
Protection class in accordance with IEC 536 (VDE 0106, Part 1)	A protective conductor connection to the profile rail is necessary!
Protection class conforms to IEC 529	IP20 (protection against contact with a standard test finger)
Protection against foreign objects	Less than 12 mm in diameter
Protection against water	no protection


### Component identification

Every supplied component includes an adhesive label providing information about the product's approvals.



Fig. 172: Name plate of a BX controller

The following information is printed on the label:

Printed item	Meaning for this label
Precise product identification	BX model
Supply voltage $U_s$	24 V <sub>DC</sub> (Use a 4 A fuse or a Class 2 power supply to meet UL requirements!)
Manufacturer	Beckhoff Automation GmbH
CE mark	Conformity mark
UL mark 	Mark for UL approval. UL stands for the Underwriters Laboratories Inc., the leading certification organization for North America, based in the USA. C = Canada, US = USA, UL file number: E172151
Production identification	Serial No.: Serial number HW: hardware version Date: Date of manufacture optional for BX9000 MAC-ID only

## 8.7 Test standards for device testing

### EMC

#### EMC immunity

EN 61000-6-2

#### Electromagnetic emission

EN 61000-6-4

### Vibration / shock resistance

#### Vibration resistance

EN 60068-2-6

#### Shock resistance

EN 60068-2-27, EN 60068-2-29

## 8.8 Bibliography

### TCP/IP

TCP/IP (German)

- Aufbau und Betrieb eines TCP/IP Netzes (Structure and Operation of a TCP/IP Network)
- by Kevin Washburn and Jim Evans
- Publisher: ADDISON-WESLEY Longmann Verlag

TCP/IP (English)

- Illustrated, Volume1 The Protocols
- by W. Richard Stevens
- Publisher: ADDISON-WESLEY Longmann Verlag

**Modbus/TCP**

<http://www.modicon.com/>

<http://www.modbus.org>

**TwinCAT**

Beckhoff Information System

<http://infosys.beckhoff.com/>

## **8.9 List of Abbreviations**

**ADS**

Automation Device Specification.

**IP (20)**

Bus Terminal protection class

**IPC**

Industrial PC

**I/O**

Inputs and outputs

**K-bus**

Terminal bus

**KS2000**

Configuration software for Bus Terminals, Bus Couplers, Bus Terminal Controllers, fieldbus box modules, etc.

**PE**

The PE power contact can be used as a protective earth.

**TwinCAT**

The Windows Control and Automation Technology

## 8.10 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

### Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages:

<http://www.beckhoff.com>

You will also find further documentation for Beckhoff components there.

### Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20  
33415 Verl  
Germany

Phone:	+49(0)5246/963-0
Fax:	+49(0)5246/963-198
e-mail:	info@beckhoff.com

### 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(0)5246/963-157
Fax:	+49(0)5246/963-9157
e-mail:	support@beckhoff.com

### Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49(0)5246/963-460
Fax:	+49(0)5246/963-479
e-mail:	service@beckhoff.com

# List of illustrations

Fig. 1	Bus Terminal Controllers of the BX series .....	9
Fig. 2	BX9000 - Ethernet Modbus TCP/ADS-TCP/UDP .....	10
Fig. 3	The principle of the Bus Terminal .....	14
Fig. 4	BX3100, BX5100, BX5200, BX9000 .....	17
Fig. 5	BX8000 .....	18
Fig. 6	Released BX controller .....	18
Fig. 7	Latched BX controller .....	19
Fig. 8	Disassembly .....	19
Fig. 9	Potential groups of a Bus Terminal block .....	20
Fig. 10	Power contact on the left .....	21
Fig. 11	Terminal points for the Bus Terminal Controller supply .....	22
Fig. 12	UL identification .....	23
Fig. 13	Programming cable ZK1000-0030 - COM 1 and COM 2 .....	24
Fig. 14	Programming cable ZK1000-0030 - plug connector dimensions .....	24
Fig. 15	Programming cable ZK1000-0030 - Pinning .....	25
Fig. 16	SSB interface .....	25
Fig. 17	COM1 (RS 232) and COM2 (RS 232/485) interface .....	26
Fig. 18	RJ45 connector .....	26
Fig. 19	Ethernet connection between PC and BX9000 via hub or switch .....	26
Fig. 20	Ethernet connection between PC and BX9000 via cross-over cable .....	27
Fig. 21	Start-up behavior of the Bus Terminal Controller .....	28
Fig. 22	Address Configuration via TwinCAT System Manager .....	30
Fig. 23	Setting the address via BootP server .....	31
Fig. 24	Creating a TwinCAT configuration .....	34
Fig. 25	Selecting the Bus Terminal Controller .....	35
Fig. 26	Downloading a TwinCAT configuration .....	36
Fig. 27	Selecting the Bus Terminal Controller .....	36
Fig. 28	State of the Bus Terminal Controller .....	36
Fig. 29	Activating the TwinCAT configuration .....	36
Fig. 30	Choose Target System .....	37
Fig. 31	Selecting the Bus Terminal Controller .....	38
Fig. 32	State of the Bus Terminal Controller .....	38
Fig. 33	Uploading the TwinCAT configuration .....	38
Fig. 34	Memory for code mapping .....	39
Fig. 35	Data memory mapping .....	39
Fig. 36	Code and data memory .....	40
Fig. 37	Other memory .....	40
Fig. 38	Properties of the remote connection .....	41
Fig. 39	Specifying the target system .....	43
Fig. 40	Finding the Bus Terminal Controller .....	43
Fig. 41	Selecting the Bus Terminal Controller .....	44
Fig. 42	IP address .....	45
Fig. 43	Different PLC cycle times .....	46
Fig. 44	BX Settings tab .....	47

Fig. 45	BX Diag tab.....	48
Fig. 46	Selecting the PLC project .....	49
Fig. 47	Connecting PLC variable and hardware .....	49
Fig. 48	Target system display .....	50
Fig. 49	Setting the task time .....	51
Fig. 50	Displaying the PLC cycle time .....	52
Fig. 51	Termination of the bus with a 120 Ohm termination resistor .....	53
Fig. 52	Insensitivity to incoming interference .....	53
Fig. 53	Sample topology of drop lines .....	54
Fig. 54	Structure of CAN cable ZB5100 .....	55
Fig. 55	Structure of CAN/DeviceNet cable ZB5200.....	56
Fig. 56	BK5151, EL6751 pin assignment .....	57
Fig. 57	FC51x2 .....	57
Fig. 58	BK51x0/BX5100 socket assignment.....	58
Fig. 59	LC5100 .....	59
Fig. 60	Pin assignment: M12 plug, fieldbus box .....	59
Fig. 61	Adding a further device .....	60
Fig. 62	Selecting the CANopen master SSB .....	60
Fig. 63	Adding a CANopen device.....	60
Fig. 64	Selecting a CANopen node .....	61
Fig. 65	Adding/editing object directory entries .....	62
Fig. 66	NodeState, DiagFlag and EmergencyCounter .....	63
Fig. 67	Wiring diagram for test setup .....	67
Fig. 68	Communication between BX controllers (via SSB).....	70
Fig. 69	AX2000 .....	71
Fig. 70	CANopen Interface (X6) .....	72
Fig. 71	Cimrex panel at the SSB of the BX controller .....	73
Fig. 72	IclA drive at SSB .....	74
Fig. 73	IclA drive connections.....	75
Fig. 74	Signal interface .....	76
Fig. 75	Fieldbus connection.....	76
Fig. 76	Reference ranges .....	78
Fig. 77	Listing of the referencing values .....	79
Fig. 78	Frequency converter from Lenze .....	80
Fig. 79	External power supply - internal power supply(State at Delivery) .....	81
Fig. 80	DIP switch.....	81
Fig. 81	Enabling the communication module .....	82
Fig. 82	Setting the real-time clock (RTC).....	83
Fig. 83	Navigation switches of the BX controller .....	86
Fig. 84	Switch assignment.....	86
Fig. 85	Maximum number of POU's exceeded .....	92
Fig. 86	Menu path Projects / Options / Controller Settings.....	92
Fig. 87	Controller settings.....	93
Fig. 88	Global memory insufficient .....	93
Fig. 89	Menu path Projects / Options / Build .....	93
Fig. 90	Build.....	94

Fig. 91	Changing variable links.....	99
Fig. 92	Linking a variable with an input.....	100
Fig. 93	Opening the options menu.....	103
Fig. 94	Selecting Source Download.....	103
Fig. 95	Downloading the program code.....	104
Fig. 96	Download progress .....	104
Fig. 97	Uploading a program .....	105
Fig. 98	Selecting the data transfer route.....	105
Fig. 99	Selecting the device.....	106
Fig. 100	Function block F_COMPORTREAD .....	115
Fig. 101	Function block F_COMPORTWRITE .....	115
Fig. 102	Function block FB_COMPORTOPEN .....	116
Fig. 103	Function block FB_COMPORTCLOSE .....	117
Fig. 104	Function block F_STARTDEBUGTIMER.....	119
Fig. 105	Function block F_READDEBUGTIMER .....	119
Fig. 106	Function block F_GETNAVSWITCH .....	119
Fig. 107	Function block FB_DISPWRITE .....	120
Fig. 108	Function block RTC .....	122
Fig. 109	Function block fb_ReadWriteFile .....	123
Fig. 110	Function block FB_BX_BK8X00_master .....	125
Fig. 111	Function block FB_BX_BK8X00_SLAVE .....	127
Fig. 112	Communication features.....	128
Fig. 113	Function block FB_BX_COM_5.....	129
Fig. 114	Function block FB_BX_COM_64.....	129
Fig. 115	Function block FB_BX_COM_64EX .....	130
Fig. 116	Cimrex panel at the COM port of the BX controller .....	132
Fig. 117	Mobile phone at the COM port of the BX controller .....	135
Fig. 118	Function block F_GETVERSIONTCTWINSAFE .....	136
Fig. 119	Function block FB_TWINSAFE_KLX904_INPUT .....	136
Fig. 120	Function block FB_TWINSAFE_KLX904_input.....	137
Fig. 121	Linking the input data.....	138
Fig. 122	Selecting the SafetyIn variable .....	138
Fig. 123	Function block FB_TWINSAFE_KLX904_output.....	139
Fig. 124	Call of function block FB_TWINSAFE_KLX904_OUTPUT .....	140
Fig. 125	Call of function block FB_TWINSAFE_KLX904_OUTPUT .....	140
Fig. 126	Linking the input data.....	140
Fig. 127	Selecting the corresponding SafetyQBx variable .....	141
Fig. 128	Function block FB_IPSTARTSESSION .....	143
Fig. 129	Function block FB_IPENDSESSION .....	144
Fig. 130	Function block FB_IPOPEN .....	145
Fig. 131	Function block FB_IPCLOSE .....	146
Fig. 132	Function block FB_IPRECEIVE.....	147
Fig. 133	Function block FB_IPSEND.....	148
Fig. 134	UDP/IP connection .....	153
Fig. 135	Sample program .....	154
Fig. 136	Function block FB_MBCONNECT .....	155



Fig. 137	Function block FB_MBGENERICREQ .....	156
Fig. 138	Function block FB_MBCLOSE .....	157
Fig. 139	Function block FB_Smtp .....	158
Fig. 140	Function block fbSMTP .....	160
Fig. 141	Function block FB_SNTP .....	161
Fig. 142	Function block FB_AddDnsServer .....	161
Fig. 143	Function block FB_GetHostByAddr .....	162
Fig. 144	Function block FB_GetHostByName .....	163
Fig. 145	Function block FB_GetNetworkConfig .....	164
Fig. 146	Function block FB_SetTargetName .....	165
Fig. 147	Function block HTTP .....	168
Fig. 148	IP address of the BX9000 in the TwinCAT System Manager .....	169
Fig. 149	Selecting the data transfer route - AMS .....	170
Fig. 150	Choose Target System .....	170
Fig. 151	Selecting the data transfer route - serial interface .....	171
Fig. 152	Parameterization of the serial interface .....	171
Fig. 153	Selecting the data transfer route - AMS .....	171
Fig. 154	Selecting the device .....	172
Fig. 155	User Datagram Protocol (UDP) .....	174
Fig. 156	Protocols running on top of TCP/IP and UDP/IP .....	174
Fig. 157	ModbusTCP Protocol .....	177
Fig. 158	The ADS protocol as a transport layer within the TwinCAT system .....	183
Fig. 159	Structure of the ADS protocol .....	184
Fig. 160	Ethernet fieldbus state .....	187
Fig. 161	K-bus status .....	188
Fig. 162	Diagnostic LEDs for the fieldbus, the PLC, the K-bus and the power supply units .....	189
Fig. 163	Diagnostic LEDs for the fieldbus, the PLC and the K-bus .....	189
Fig. 164	Navigation switch .....	193
Fig. 165	Switch assignment .....	194
Fig. 166	Adding the BX controller in the AMS router .....	196
Fig. 167	Adding the remote connection .....	196
Fig. 168	Choose Target System .....	196
Fig. 169	Selecting the runtime system (step 1) .....	197
Fig. 170	Selecting the runtime system (step 2) .....	197
Fig. 171	Firmware update via Ethernet .....	200
Fig. 172	Name plate of a BX controller .....	202