Documentation

BC9050, BC9020 and BC9120

Bus Terminal Controller for Ethernet

EUROTEC
TECNOLOGIE IMPIANTISTICHE - AUTOMAZIONE - TELECONTROLLO

BECKHOFF

Table of contents

1 Foreword ... 7
1.1 Notes on the documentation ... 7
1.2 Safety instructions ... 8
1.3 Documentation issue status ... 9
1.4 Beckhoff Identification Code (BIC) ... 9
2 Product overview ... 12
2.1 The Beckhoff Bus Terminal system ... 12
2.2 BC9xx0 - Overview ... 14
2.3 BC9050 Bus Terminal principle ... 15
2.4 The principle of the Bus Terminal ... 16
2.5 Technical data ... 17
2.5.1 Technical data - Ethernet ... 17
2.5.2 Technical data - Bus Terminal Controller ... 17
2.5.3 Technical Data - PLC ... 18
3 Mounting and wiring ... 19
3.1 Mounting ... 19
3.1.1 Dimensions ... 19
3.1.2 Mounting ... 21
3.2 Wiring ... 22
3.2.1 Potential groups, insulation testing and PE ... 22
3.2.2 Power supply ... 24
3.2.3 Ethernet topologies ... 26
3.2.4 Ethernet connection ... 28
3.2.5 Ethernet cable ... 31
3.2.6 ATEX - Special conditions (standard temperature range) ... 32
3.2.7 ATEX - Special conditions (extended temperature range) ... 33
4 Parameterization and Commissioning ... 34
4.1 Start-up behavior of the Bus Terminal Controller ... 34
4.2 DIP switch ... 35
4.3 Setting the IP address ... 37
4.3.1 Address Configuration via TwinCAT System Manager ... 38
4.3.2 Setting the address via BootP server ... 38
4.3.3 Setting the address via DHCP server ... 40
4.3.4 Auto IP address ... 40
4.3.5 Subnet mask ... 40
4.4 Configuration ... 41
4.4.1 Overview ... 41
4.4.2 Finding the Bus Terminal Controller with the TwinCAT System Manager ... 43
4.4.3 Creating a TwinCAT configuration ... 45
4.4.4 Downloading a TwinCAT configuration ... 46
4.4.5 Uploading a TwinCAT configuration ... 48
4.4.6 Resources in the Bus Terminal Controller ... 49
4.4.7 ADS connection via serial interface ... 52

4.4.8 Ethernet ... 53
4.4.9 K-bus ... 56
4.4.10 PLC ... 58
4.5 KS2000 ... 62
4.5.1 KS2000 Configuration Software ... 62

5 Programming ... 63
5.1 PLC features ... 63
5.2 TwinCAT PLC ... 64
5.3 TwinCAT PLC - Error codes ... 64
5.4 Remanent data ... 67
5.5 Persistent data ... 68
5.6 Allocated flags ... 68
5.7 Local process image in delivery state (default config) ... 70
5.8 Mapping the Bus Terminals ... 71
5.9 Local process image in the TwinCAT configuration ... 72
5.10 Creating a boot project ... 73
5.11 Communication between TwinCAT and BX/BCxx50 ... 73
5.12 Up- and downloading of programs ... 75
5.13 Libraries ... 78
5.13.1 Libraries overview ... 78
5.13.2 Overview of libraries for BX9000, BC9020, BC9050, BC9120 ... 79
5.13.3 TcBaseBCxx50 ... 81
5.13.4 TcBaseBX9000 ... 85
5.14 Program transfer ... 111
5.14.1 Program Transfer via Ethernet ... 111
5.14.2 Program transfer via the serial interface ... 112
5.15 Process image ... 114
5.15.1 Fieldbus Process Image ... 114

6 Ethernet ... 115
6.1 Introduction to the system ... 115
6.1.1 Ethernet ... 115
6.2 ModbusTCP ... 117
6.2.1 ModbusTCP Protocol ... 117
6.2.2 Modbus TCP interface ... 118
6.2.3 ModbusTCP slave error answer (BK9000, BX/BC9xx0, IP/ILxxxx-B/C900, EK9000) ... 119
6.2.4 ModbusTCP functions ... 120
6.3 ADS-Communication ... 123
6.3.1 ADS-Communication ... 123
6.3.2 ADS protocol ... 124
6.3.3 ADS services ... 125

7 Error handling and diagnosis ... 129
7.1 Diagnostics ... 129
7.2 Diagnostic LEDs ... 130

8 Appendix ... 134
8.1 BC9xx0 - First steps ... 134

8.2 Switching between controllers 141
8.3 General operating conditions 143
8.4 Test standards for device testing 145
8.5 Bibliography 146
8.6 List of Abbreviations 147
8.7 Support and Service 148

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®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents: EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702 with corresponding applications or registrations in various other countries.

EtherCAT® is 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 instructions

In this documentation the following instructions are used.
These instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow this safety instruction directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow this safety instruction endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow this safety instruction can lead to injuries to persons.

NOTE

Damage to environment/equipment or data loss

Failure to follow this instruction can lead to environmental damage, equipment damage or data loss.

Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Documentation issue status

Version	Comment
2.3.0	• Update chapter “Setting the address via BootP server” • Update structure
2.2.0	• Download links updated
2.1.0	• Technical data updated
2.0.0	• Migration • Technical data updated
1.0.0	• First version

BC9050 firmware

Firmware	Comment
B4	Optimizations
B0	First version

BC9020 firmware

Firmware	Comment
B4	ADS Index Group 0x4040 implemented
B3	Serial interface optimized
B2	First version
B0	Internal version

BC9120 firmware

Firmware	Comment
B4	ADS Index Group 0x4040 implemented
B3	Serial interface optimized
B2	First version
B1	Internal version

1.4 Beckhoff Identification Code (BIC)

The Beckhoff Identification Code (BIC) is increasingly being applied to Beckhoff products to uniquely identify the product. The BIC is represented as a Data Matrix Code (DMC, code scheme ECC200), the content is based on the ANSI standard MH10.8.2-2016.

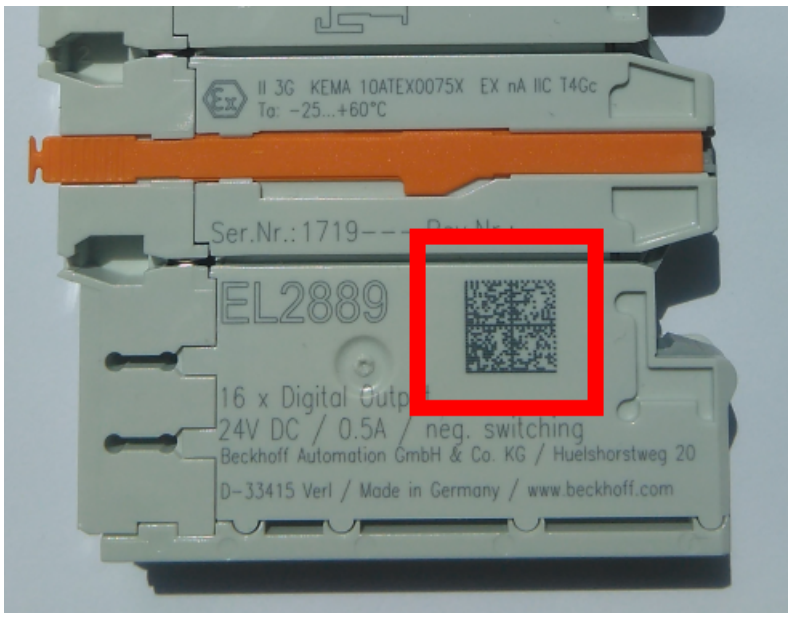


Fig. 1: BIC as data matrix code (DMC, code scheme ECC200)

The BIC will be introduced step by step across all product groups.

Depending on the product, it can be found in the following places:

- on the packaging unit
- directly on the product (if space suffices)
- on the packaging unit and the product

The BIC is machine-readable and contains information that can also be used by the customer for handling and product management.

Each piece of information can be uniquely identified using the so-called data identifier (ANSI MH10.8.2-2016). The data identifier is followed by a character string. Both together have a maximum length according to the table below. If the information is shorter, spaces are added to it. The data under positions 1 to 4 are always available.

The following information is contained:

Item no.	Type of information	Explanation	Data identifier	Number of digits incl. data identifier	Example
1	Beckhoff order number	**Beckhoff order number**	1P	8	1P072222
2	Beckhoff Traceability Number (BTN)	**Unique serial number, see note below**	S	12	SBTNk4p562d7
3	Article description	**Beckhoff article description, e.g. EL1008**	1K	32	1KEL1809
4	Quantity	**Quantity in packaging unit, e.g. 1, 10, etc.**	Q	6	Q1
5	Batch number	Optional: Year and week of production	2P	14	2P401503180016
6	ID/serial number	Optional: Present-day serial number system, e.g. with safety products	51S	12	51S678294104
7	Variant number	Optional: Product variant number on the basis of standard products	30P	32	30PF971, 2*K183
...					

Further types of information and data identifiers are used by Beckhoff and serve internal processes.

Structure of the BIC

Example of composite information from item 1 to 4 and 6. The data identifiers are marked in red for better display:

BTN

An important component of the BIC is the Beckhoff Traceability Number (BTN, item no. 2). The BTN is a unique serial number consisting of eight characters that will replace all other serial number systems at Beckhoff in the long term (e.g. batch designations on IO components, previous serial number range for safety products, etc.). The BTN will also be introduced step by step, so it may happen that the BTN is not yet coded in the BIC.

NOTE

This information has been carefully prepared. However, the procedure described is constantly being further developed. We reserve the right to revise and change procedures and documentation at any time and without prior notice. No claims for changes can be made from the information, illustrations and descriptions in this information.

2 Product overview

2.1 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.

2.2 BC9xx0 - Overview

The BC9xx0 series Bus Terminal Controllers have an Ethernet interface and offer a high degree of flexibility. They can also be operated in stand-alone mode.

The BC family Bus Terminal Controllers consist of a programmable IEC 61131-3 controller and the K-Bus interface for the connection of the Beckhoff Bus Terminals and a Ethernet interface.

The Bus Terminals can be connected directly in the usual way. 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. This small controller therefore enables a wide range of automation tasks to be solved, from garage door controllers to autonomous temperature control at injection molding machines. The BC family is also particularly suitable for a modular machine concept. Within a network, the Bus Terminal Controller can exchange data with other machine components via the fieldbus interfaces.

Bus Terminals required

At least one Bus Terminal with process image and one end terminal must be connected to the K-bus of the Bus Terminal Controller.

Fieldbus interface

The variants of the BC9xx0 series Bus Terminal Controllers have different fieldbus interfaces (with and without switch) and different maximum program size.

- BC9050: 1 x Ethernet interface, 48 kbyte program memory
- BC9020: 1 x Ethernet interface, 128 kbyte program memory
- BC9120: 2 x Ethernet interface, 128 kbyte program memory
- BX9000: 1 x Ethernet interface, 256 kbyte program memory

Programming

The BC devices are programmed according to the powerful 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 and System Manager. Data exchange is optionally via the serial port (COM1) or via Ethernet.

Configuration

The configuration is also carried out using TwinCAT. The fieldbus interface 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 BC via the serial interface. The configuration thus created can be accessed again later.

Required TwinCAT version

BC9050, BC9020, BC9120 are supported from TwinCAT 2.10 build 1322.

2.3 BC9050 Bus Terminal principle

BC9050

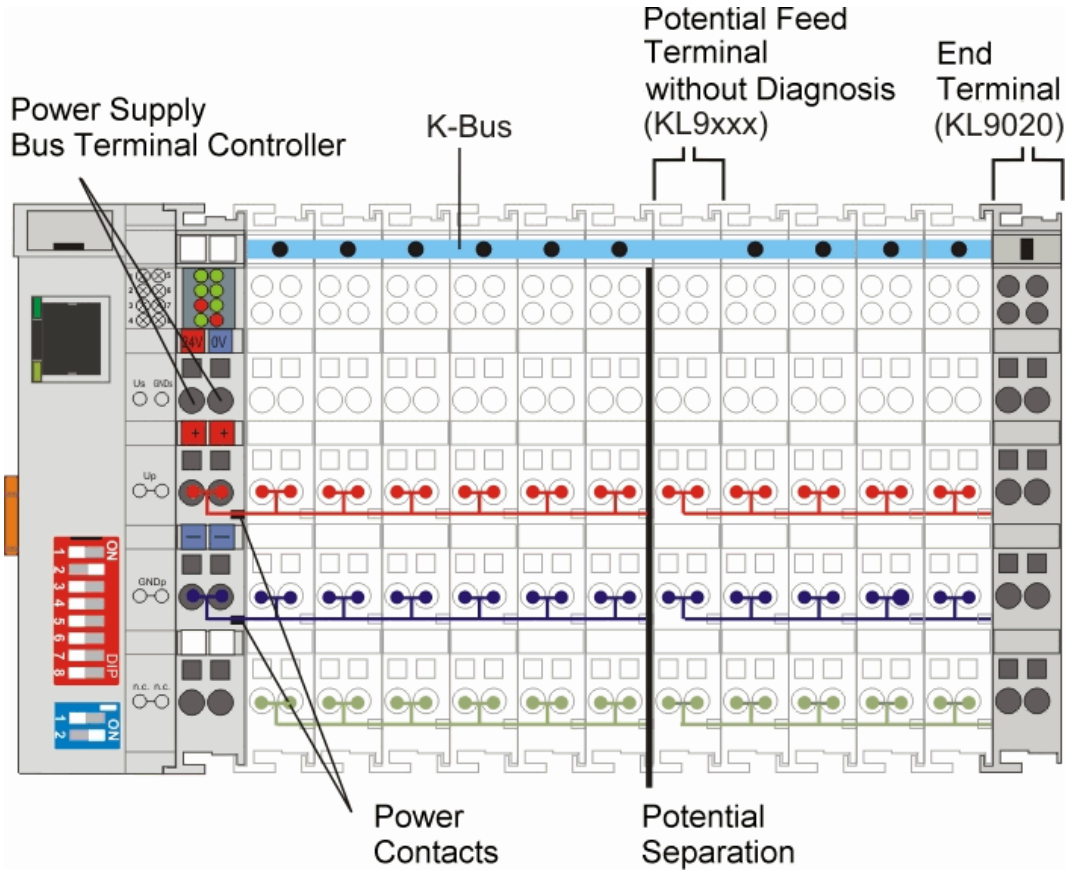


Fig. 2: BC9050 Bus Terminal principle

2.4 The principle of the Bus Terminal

BC9020, BC9120

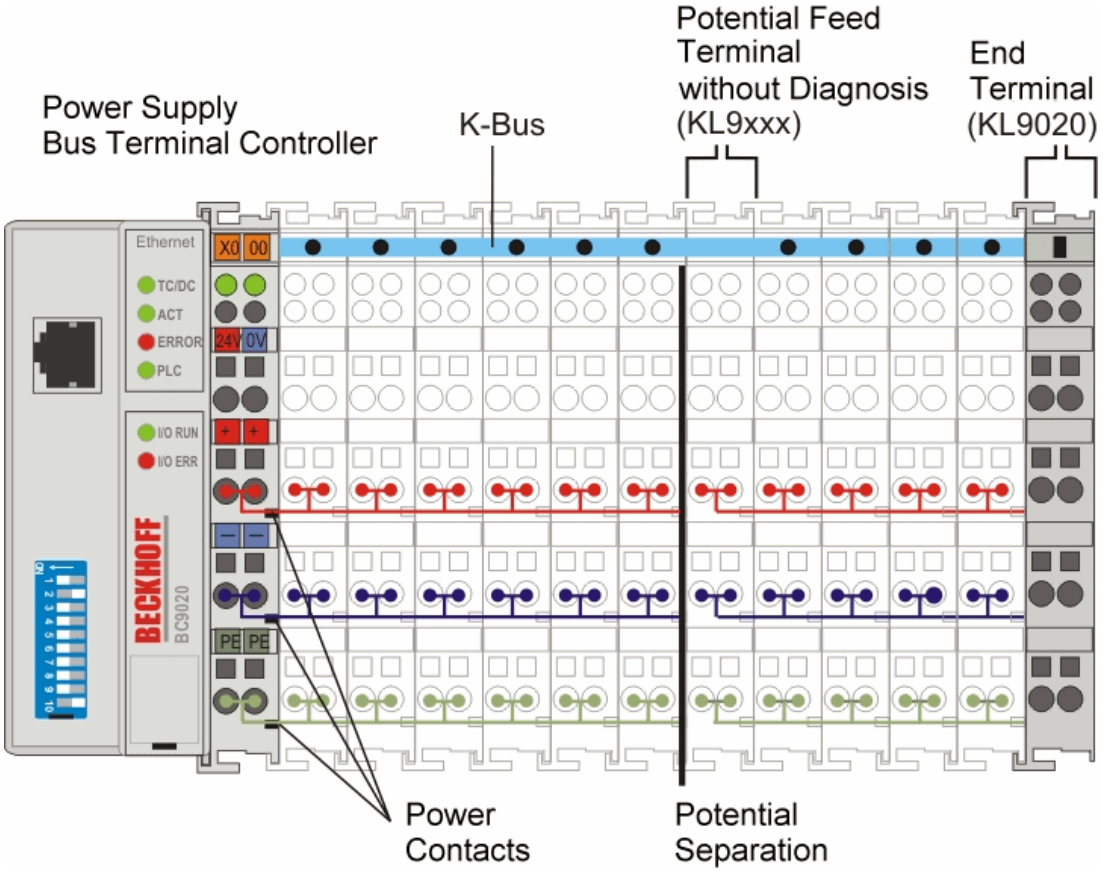


Fig. 3: BC9020, BC9120 Bus Terminal principle

2.5 Technical data

2.5.1 Technical data - Ethernet

System data	Ethernet (BC9050, BC9020, BC9120)
Number of I/O modules	depending on controller
Number of I/O points	depending on controller
Data transfer medium	4 x 2 copper cable (twisted pair); category 5 (100 Mbaud)
Cable length	100 m between switch and Bus Terminal Controller
Data transfer rate	10/100 Mbaud
Topology	Star wiring (line also possible with BC9120, max. 20 devices per line)

Connections	Ethernet (BC9050, BC9020, BC9120)
Number of ADS connections via TCP/IP	5
Number of ADS connections via UDP/IP	1
Number of ModbusTCP connections	3
SMTP (Simple Mail Transfer Protocol)	yes
SNTP (Simple Network Time Protocol)	yes
BootP/DHCP	yes/yes
UDP socket connections	3
TCP socket connections	3

Reduce the number of TCP/IP connections to a minimum

The total number of TCP/IP connections should be reduced to a minimum. The fewer you use, the more time the controller has for its actual task.
Avoid unnecessary TCP/IP connections. Try using as few TCP/IP connections as possible. If possible, use UDP/IP communication instead of TCP/IP, e.g. for ADS.
Use "reasonable" time intervals for communication via the TCP/IP or UDP/IP connection. Example: The task time on the controller is 20 ms. The minimum client cycle time should be 40 ms.

2.5.2 Technical data - Bus Terminal Controller

Technical data	BC9050	BC9020	BC9120
Processor	16 bit micro-controller		
Diagnostic LEDs	2 x power supply, 2 x K-bus, 4 x bus and status LEDs		
Configuration and programming software	TwinCAT 2.10 starting with build 1322 or higher		

Fieldbus interface	BC9050	BC9020	BC9120
Fieldbus	Ethernet		

Interfaces	BC9050	BC9020	BC9120
Terminal Bus (K-Bus)	64 (255 with K-bus extension)		

Technical data	BC9050	BC9020	BC9120
Digital peripheral signals	2040 inputs/outputs		
Analog peripheral signals	512 inputs/outputs		
Configuration possibility	via TwinCAT or the controller		
max. number of bytes, fieldbus	512 bytes of input data, 512 bytes of output data		
max. number of bytes, PLC	2048 bytes of input data, 2048 bytes of output data		
Fieldbus connection	RJ45	RJ45	2 x RJ45 switch
Power supply (Us)	24 V_{DC} (-15%/+20%)		
Input current (Us)	70 mA + (total K-bus current)/4, 4.500 mA max.	85 mA + (total K-bus current)/4	90 mA + (total K-bus current)/4
Starting current (Us)	approx. 2.5 x continuous current		
K-bus current (5 V)	maximum 1750 mA		
Power contact voltage (Up)	max. 24 V_{DC} (-15%/+20%)		
Power contact current load (Up)	maximum 10 A		
Dielectric strength	500 V (power contact/supply voltage/Ethernet/fieldbus)		
Weight	approx. 100 g	approx. 170 g	
Dimensions (W x H x D)	44 x 100 x 68 mm	44 x 100 x 70 mm	
Permissible ambient temperature range during operation	0 °C ... +55 °C	-25°C ... +60 °C	
Permissible ambient temperature range during storage	-25°C ... +85 °C	-40°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		
Mounting position / protection class	Any / IP20		
Approvals	CE cULus, ATEX [▶ 32]	CE cULus, ATEX [▶ 33], GL	

Mechanical data	BC9050	BC9020	BC9120
Mounting	on TS35 with lock		
Installation position	variable		
Connection cross-section	0.08 mm² .. 2.5 mm² AWG 28-14 8 to 9 mm strip length		

2.5.3 Technical Data - PLC

PLC data	BC9050	BC9020	BC9120
Programmability	via programming interface or fieldbus		
Program memory	48 kbyte	128 kbyte	
Source code memory	128 kbyte	256 kbyte	
Data memory	32 kbyte	128 kbyte	
Remanent data	2 kbyte		
Persistent data	1000 bytes		
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)		
Runtime	1 PLC task		
Online change	Yes		
Up/Down Load Code	Yes/Yes		

3 Mounting and wiring

3.1 Mounting

3.1.1 Dimensions

The system of the Beckhoff Bus Terminals 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 mechanical dimensions of the Bus Couplers are independent of the fieldbus system.

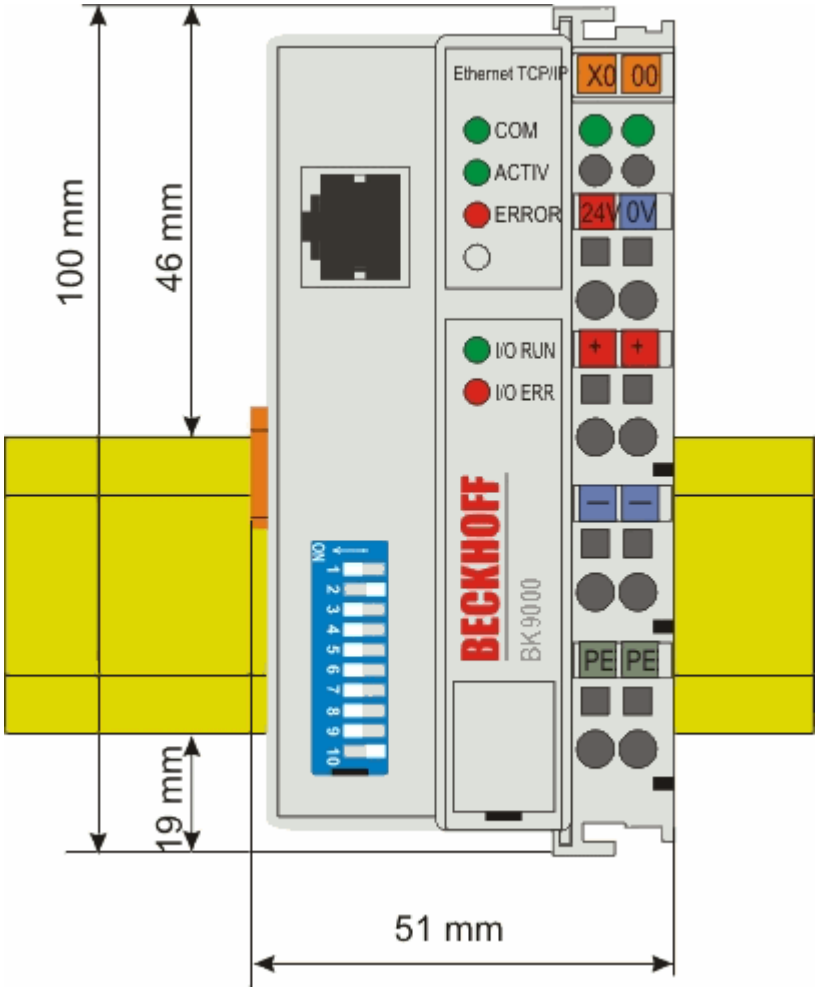


Fig. 4: BK9000, BK9100, BC9000, BC9020, BC9100, BC9120

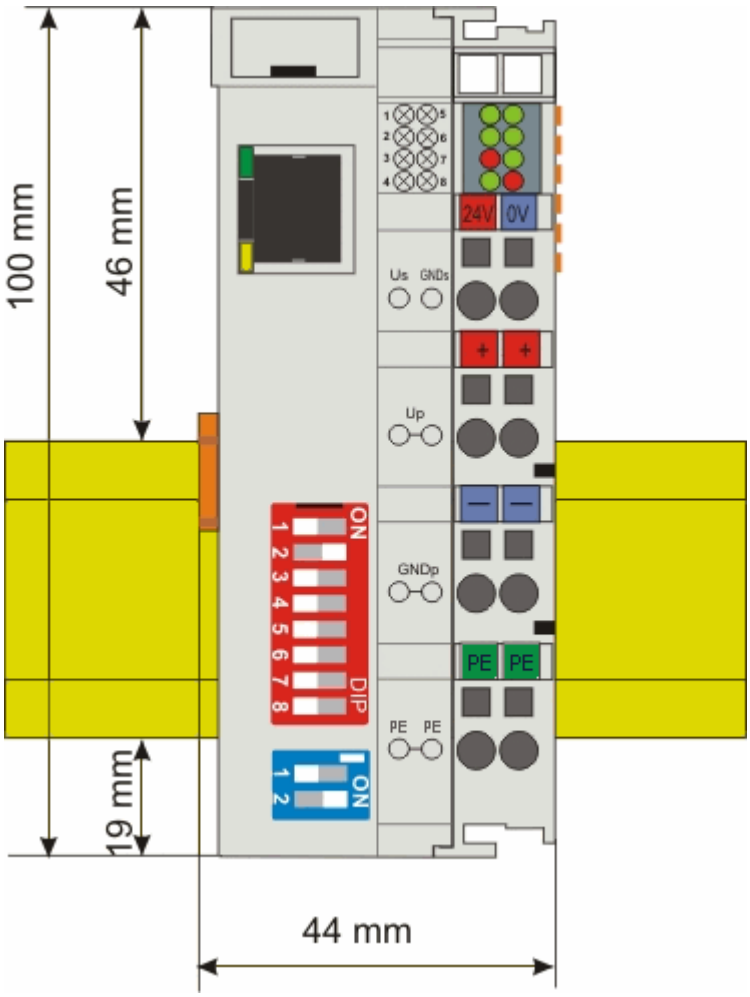


Fig. 5: BK9050, BC9050

The total width in practical cases is composed of the width of the Bus Coupler, the width of the bus terminals in use and the KL9010 Bus End Terminal. Depending on function, the Bus Terminals are 12 mm or 24 mm wide. The front wiring increases the total height of 68 mm by about 5 mm to 10 mm, depending on the wire thickness.

3.1.2 Mounting

The Bus Coupler and all the Bus Terminals can be clipped, with a light press, onto a 35 mm mounting rail. A locking mechanism prevents the individual housings from being pulled off again. For removal from the mounting rail the orange colored tension strap releases the latching mechanism, allowing the housing to be pulled off the rail without any force.

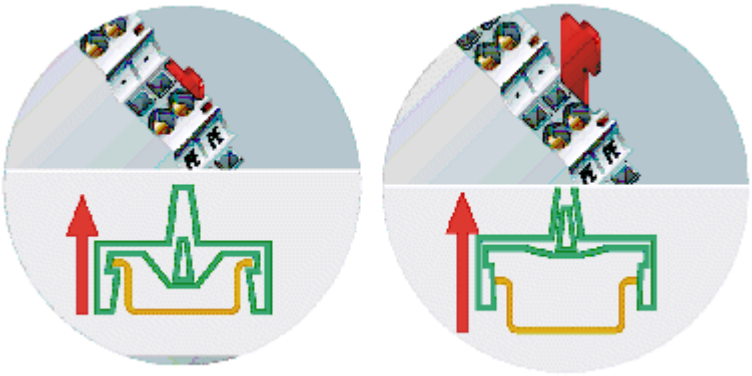


Fig. 6: Release the locking mechanism by pulling the orange tab

Up to 64 Bus Terminals can be attached to the Bus Coupler on the right hand side. When plugging the components together, be sure to assemble the housings with groove and tongue against each other. A properly working connection cannot be made by pushing the housings together on the mounting rail. When correctly assembled, no significant gap can be seen between the attached housings.

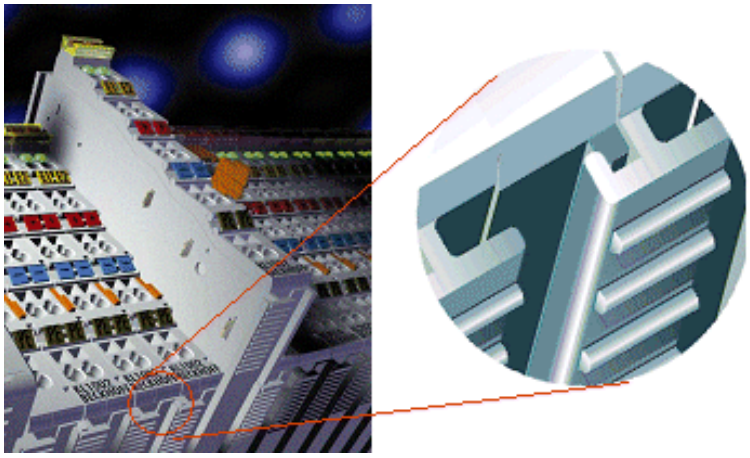


Fig. 7: Groove and tongue of the housings

NOTE

Bus Terminals should only be pulled or plugged in switched-off state.

Insertion and removal of Bus Terminals is only permitted when switched off. The electronics in the Bus Terminals and in the Bus Coupler are protected to a large measure against damage, but incorrect function and damage cannot be ruled out if they are plugged in under power.

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.

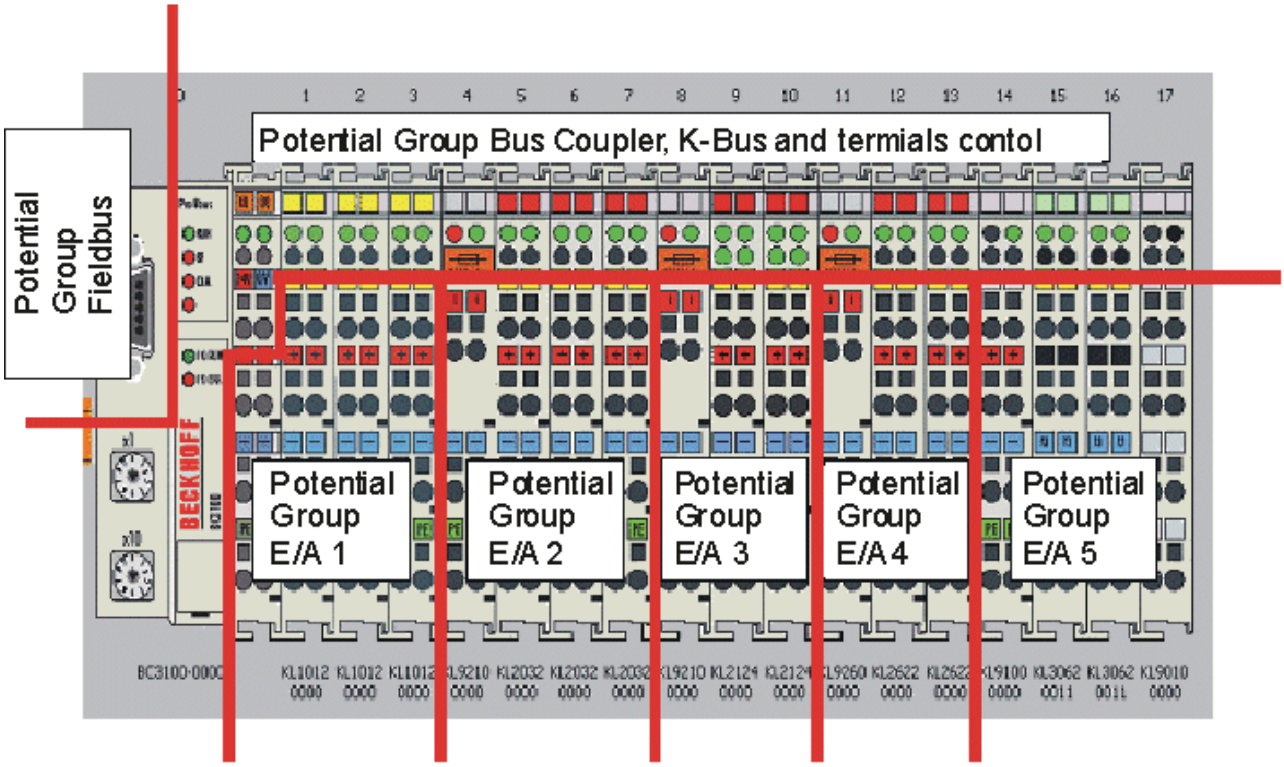


Fig. 8: Potential groups of a Bus Terminal block

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.

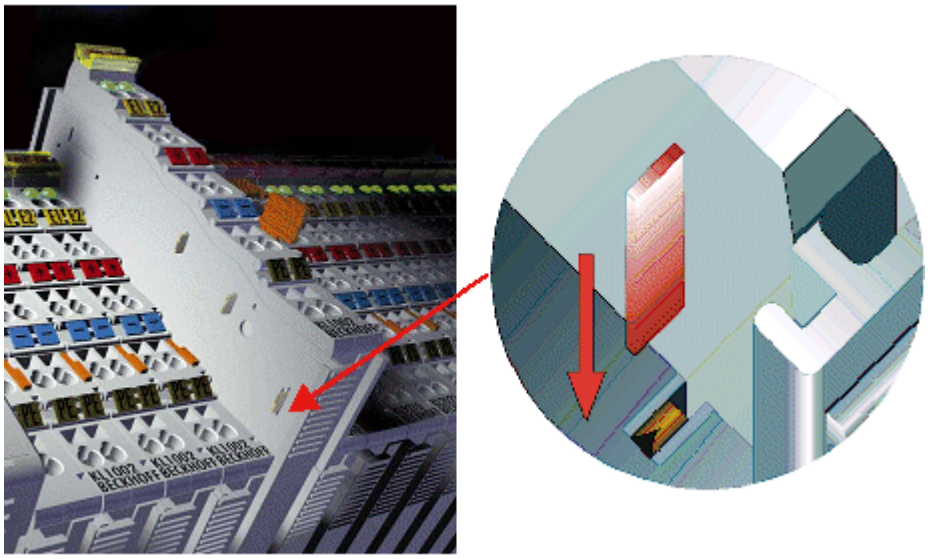


Fig. 9: Power contact on the left

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

⚠ DANGER

Risk of injury through electric shock and damage to the device!

Bring the Bus Terminals system into a safe, de-energized state
before starting mounting, disassembly or wiring of the components!

BC9050

Supply of Bus Terminal Controller and Bus Terminals (Us)

The Bus Terminal Controller requires a supply voltage of 24 V_{DC}.

The connection is made by means of the upper spring-loaded terminals labelled *24 V* and *0 V*. This supply voltage is used for the electronic components of the Bus Coupler / Bus Terminal Controllers and (via the K-bus) the electronic components of the Bus Terminals. It is galvanically separated from the field level voltage.

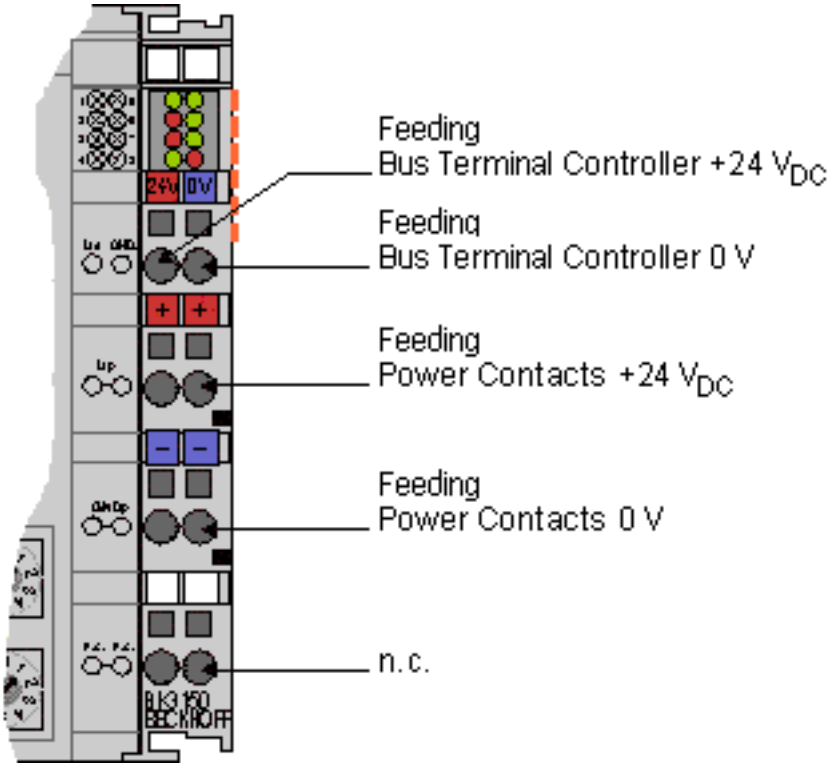


Fig. 10: Terminal points for the Bus Terminal Controller supply

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 BC electronics.

The spring-loaded terminals are designed for wires with cross-sections between 0.08 mm^2 and 2.5 mm^2.

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 Ethernet topologies

BK9000, BK9050, BC9000, BC9020, BC9050

These Bus Couplers and Bus Terminal controllers have a single Ethernet connection. This can be connected directly to an external switch. This makes it possible to construct the typical Ethernet star topology.

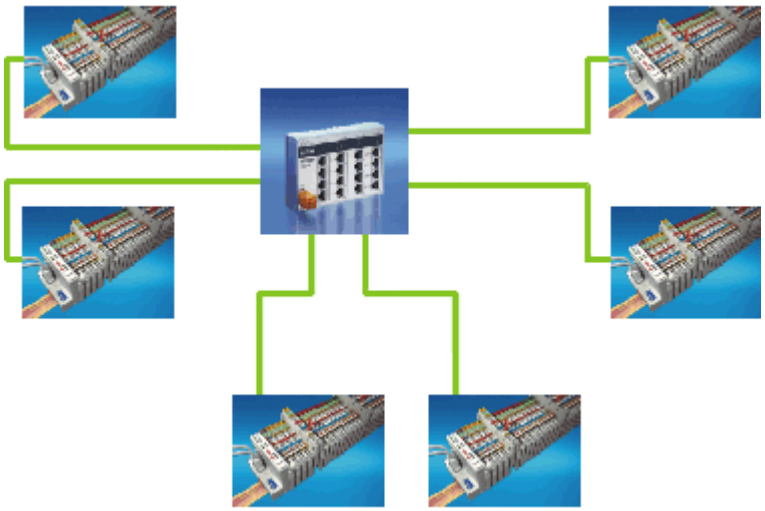


Fig. 11: Ethernet layout in star topology

BK9100, BC9100, BC9120, BC9191

These Bus Couplers and Bus Terminal controllers have an internal triple switch with one internal and two external ports. The internal switch enables the simple construction of a linear topology. A maximum of 20 BK9100/BC91x0/BC9191 can be connected in series in a physical line. However the distance between two Ethernet devices may not exceed 100 m. The maximum overall line length is therefore 2 km. No further switches may be included in this line.

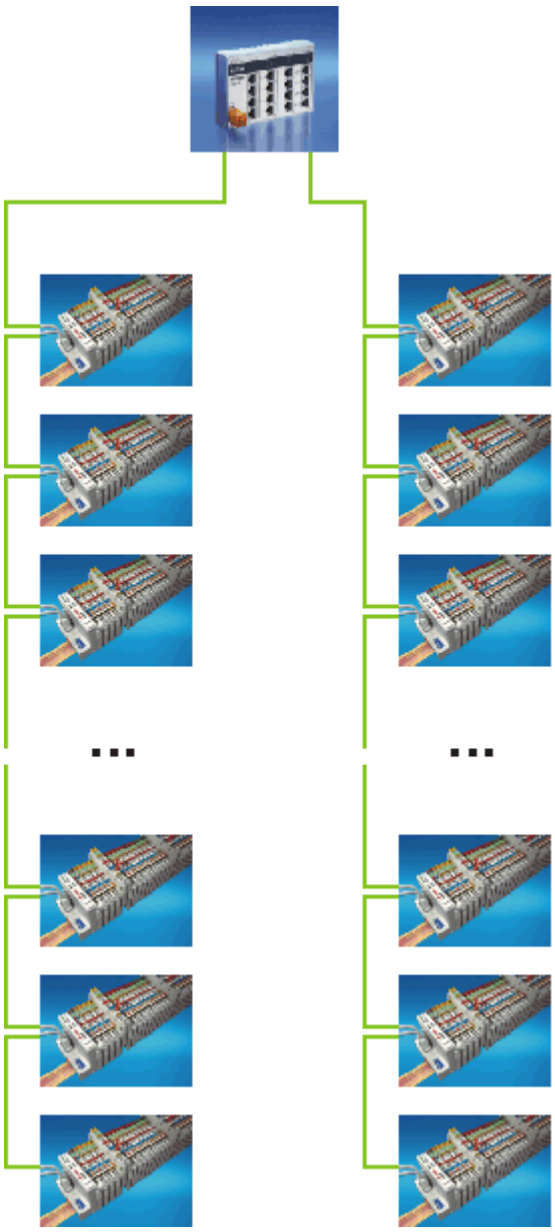


Fig. 12: Ethernet layout in linear topology

Of course, the construction of a classic star topology is also possible with these Bus Couplers and Bus Terminal controllers.

3.2.4 Ethernet connection

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

Fig. 13: RJ45 connector

Ethernet cabling BC9020 or BC9050

Connection via hub or switch

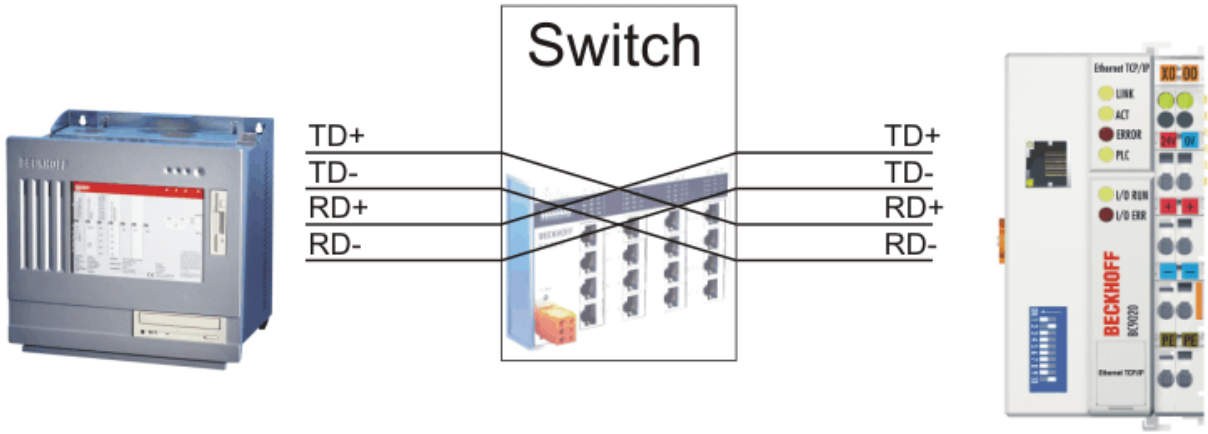


Fig. 14: Ethernet connection between PC and BC9020, BC9050 via hub or switch

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.

Direct connection between PC with Ethernet card and BX9000

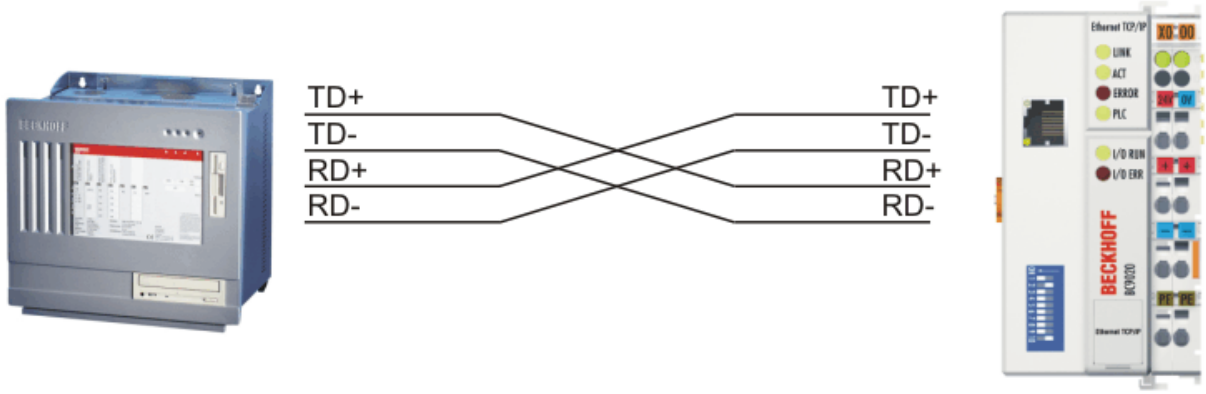


Fig. 15: Direct Ethernet connection between PC and BC9020, BC9050 via cross-over cable

To connect the PC directly with the BC9020 or BC9050, you have to use an Ethernet cable with a cross-over cable.

Ethernet cabling BC9120

Connection via hub or switch

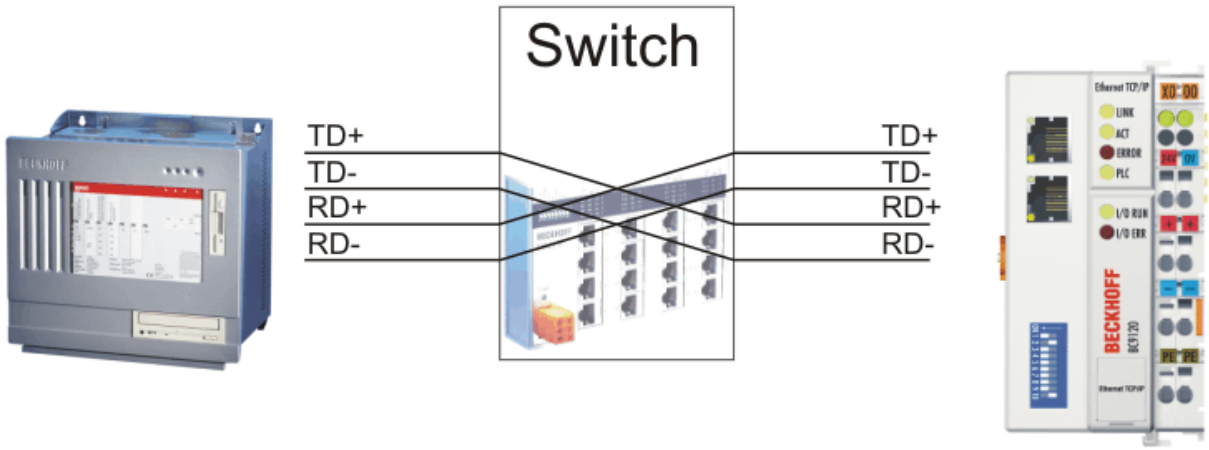
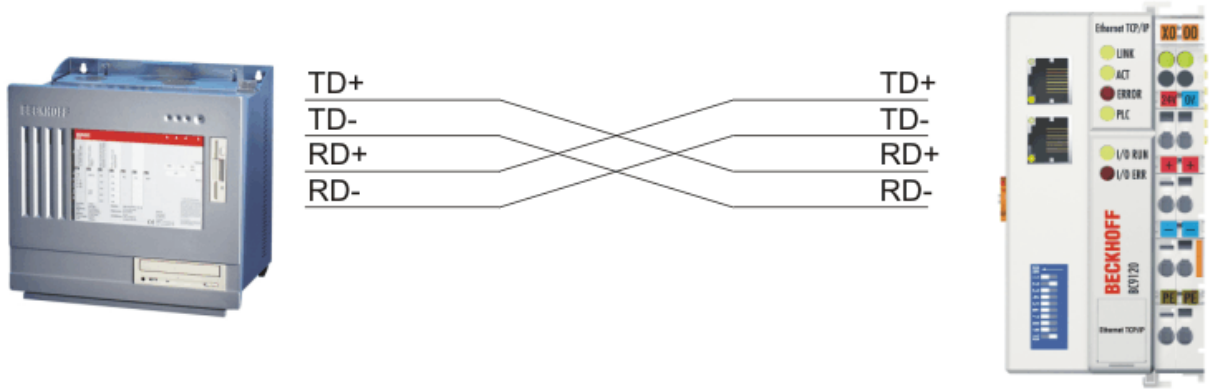


Fig. 16: Ethernet connection between PC and BC9120 via hub or switch

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.

Direct connection between PC with Ethernet card and BC9120



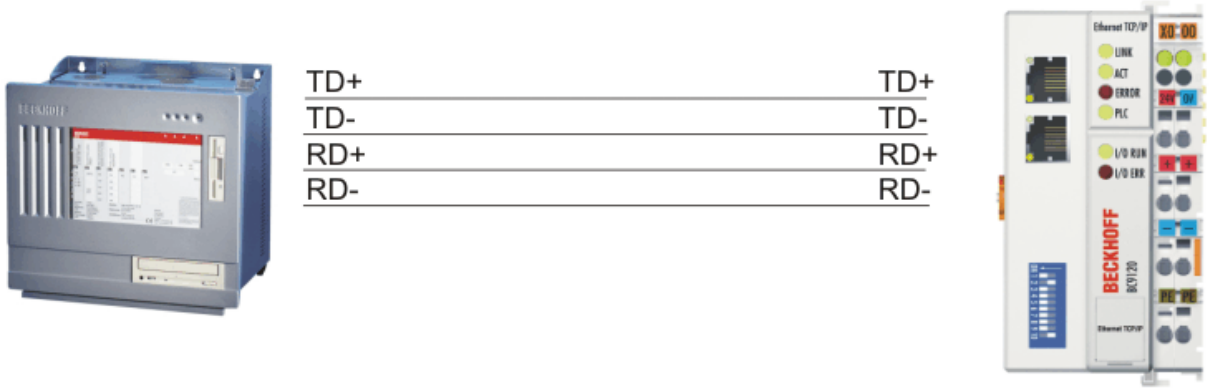


Fig. 17: Direct Ethernet connection between PC and BC9120

Any Ethernet cable can be used to connect the PC directly with the Bus Terminal Controller. The internal switch in the BC91x0, BK9100 detects this automatically.

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

3.2.5 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.

3.2.6 ATEX - Special conditions (standard temperature range)

⚠ WARNING

Observe the special conditions for the intended use of Beckhoff fieldbus components with standard temperature range in potentially explosive areas (directive 2014/34/EU)!

- The certified components are to be installed in a suitable housing that guarantees a protection class of at least IP54 in accordance with EN 60079-15! The environmental conditions during use are thereby to be taken into account!
- If the temperatures during rated operation are higher than 70°C at the feed-in points of cables, lines or pipes, or higher than 80°C at the wire branching points, then cables must be selected whose temperature data correspond to the actual measured temperature values!
- Observe the permissible ambient temperature range of 0 to 55°C for the use of Beckhoff fieldbus components standard temperature range in potentially explosive areas!
- Measures must be taken to protect against the rated operating voltage being exceeded by more than 40% due to short-term interference voltages!
- The individual terminals may only be unplugged or removed from the Bus Terminal system if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The connections of the certified components may only be connected or disconnected if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The fuses of the KL92xx/EL92xx power feed terminals may only be exchanged if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- Address selectors and ID switches may only be adjusted if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!

Standards

The fundamental health and safety requirements are fulfilled by compliance with the following standards:

- EN 60079-0:2012+A11:2013
- EN 60079-15:2010

Marking

The Beckhoff fieldbus components with standard temperature range certified according to the ATEX directive for potentially explosive areas bear one of the following markings:

II 3G KEMA 10ATEX0075 X Ex nA IIC T4 Gc Ta: 0 … +55°C

or

II 3G KEMA 10ATEX0075 X Ex nC IIC T4 Gc Ta: 0 … +55°C

3.2.7 ATEX - Special conditions (extended temperature range)

⚠ WARNING

Observe the special conditions for the intended use of Beckhoff fieldbus components with extended temperature range (ET) in potentially explosive areas (directive 2014/34/EU)!

- The certified components are to be installed in a suitable housing that guarantees a protection class of at least IP54 in accordance with EN 60079-15! The environmental conditions during use are thereby to be taken into account!
- If the temperatures during rated operation are higher than 70°C at the feed-in points of cables, lines or pipes, or higher than 80°C at the wire branching points, then cables must be selected whose temperature data correspond to the actual measured temperature values!
- Observe the permissible ambient temperature range of -25 to 60°C for the use of Beckhoff fieldbus components with extended temperature range (ET) in potentially explosive areas!
- Measures must be taken to protect against the rated operating voltage being exceeded by more than 40% due to short-term interference voltages!
- The individual terminals may only be unplugged or removed from the Bus Terminal system if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The connections of the certified components may only be connected or disconnected if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- The fuses of the KL92xx/EL92xx power feed terminals may only be exchanged if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!
- Address selectors and ID switches may only be adjusted if the supply voltage has been switched off or if a non-explosive atmosphere is ensured!

Standards

The fundamental health and safety requirements are fulfilled by compliance with the following standards:

- EN 60079-0:2012+A11:2013
- EN 60079-15:2010

Marking

The Beckhoff fieldbus components with extended temperature range (ET) certified according to the ATEX directive for potentially explosive areas bear the following marking:

II 3G KEMA 10ATEX0075 X Ex nA IIC T4 Gc Ta: -25 … +60°C

or

II 3G KEMA 10ATEX0075 X Ex nC IIC T4 Gc Ta: -25 … +60°C

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. 3-5 seconds. If the event of an error, then the LED that flashes will depend on the type of that error (see Diagnostic LEDs).

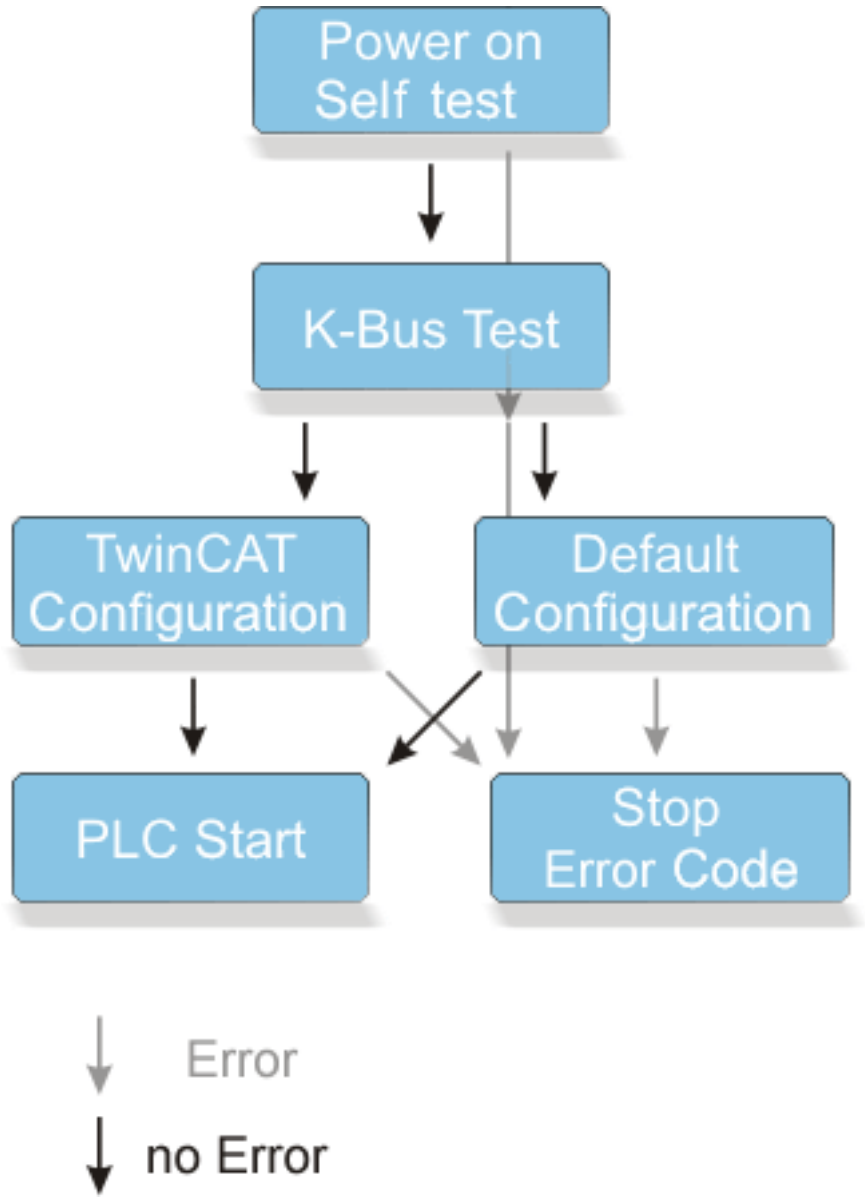


Fig. 18: Start-up behavior of the BC9xx0

The Bus Terminal Controller indicates a faulty TwinCAT configuration (LEDs/display). In the Bus Terminal controllers of the BCxx50, BCxx20 and BXxxxx series, the display can also be read via the System Manager.

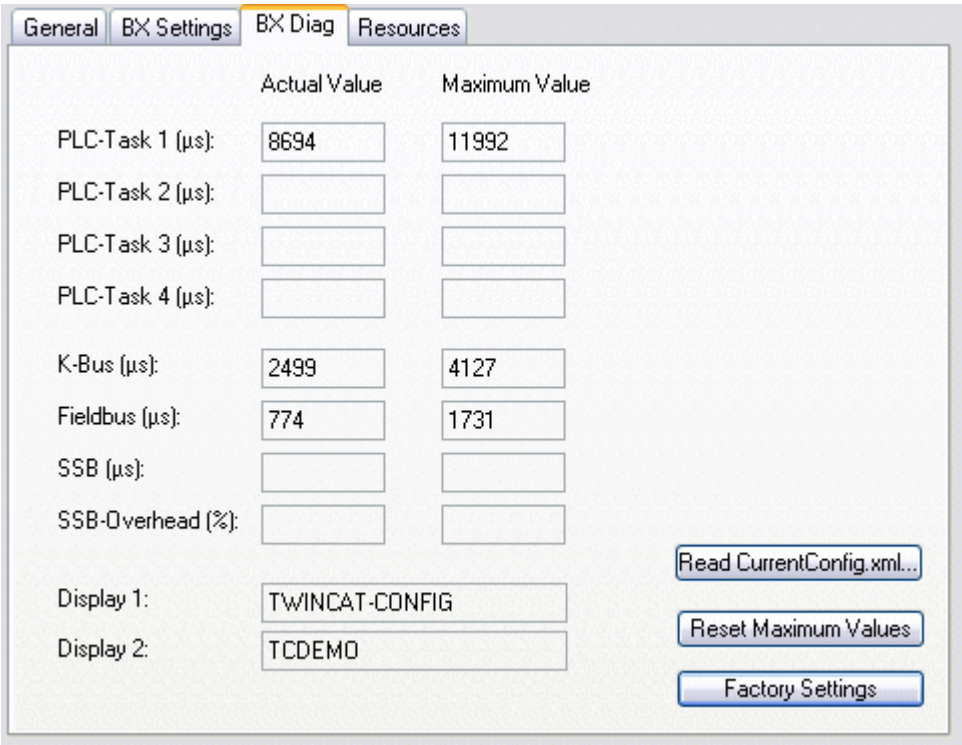


Fig. 19: Display in the System Manager

4.2 DIP switch

The BC9050 and BC9120/BC9020 are equipped with DIP switches. These DIP switches have two functions.

1. Addressing mode or setting the IP address
2. Configuration mode

Mode one is enabled if Bus Terminals are connected to the K-bus and the Bus Terminal Controller is switched on. Mode two is activated if only the end terminal is connected when the Bus Terminals Controller is switched on.

Mode 1: Addressing mode

DIP switches of the BC9050

Right switch position: ON, switch in left position: OFF.

- DIP switches 1 to 8 edit the last byte of the IP address (the image shows the red DIP switches).
- DIP switches 1 and 2 select the IP address mode (the image shows the blue DIP switches).

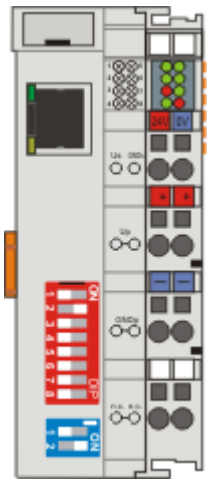


Fig. 20: DIP switches of the BC9050

DIP1	DIP2	Meaning
off	off	172.16.21.xxx (xxx corresponds to DIP switch 1-8), subnet mask 255.255.0.0, default gateway 0.0.0.0
on	off	BootP (DIP 1-8 all off), BootP & Safe (DIP 1-8 all on)
off	on	DHCP
on	on	Configuration via the TwinCAT System Manager

DIP switches of the BC9020 and BC9120

Right switch position: OFF, switch in left position: ON.

- DIP switches 1 to 8 edit the last byte of the IP address.
- DIP switches 9 to 10 select the IP address mode.

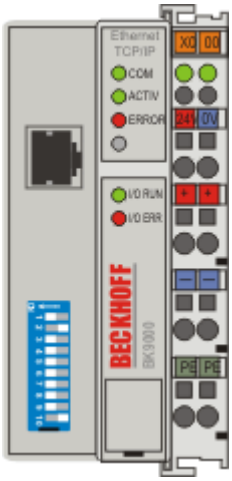


Fig. 21: DIP switches of the BC9020 and BC9120

DIP9	DIP10	Meaning
off	off	BC9020 172.16.22.xxx (xxx corresponds to DIP switches 1-8), subnet mask 255.255.0.0, default gateway 0.0.0.0 BC9120 172.16.23.xxx (xxx corresponds to DIP switches 1-8), subnet mask 255.255.0.0, default gateway 0.0.0.0
on	off	BootP (DIP 1-8 all off), BootP & Safe (DIP 1-8 all on)
off	on	DHCP
on	on	Configuration via the TwinCAT System Manager

Mode 2: Configuration mode

In the de-energized Bus Terminal Controller only connect the KL9010 end terminal. Set the DIP switch to the required function, then switch on the Bus Terminal Controller. The function is active as soon as the Controller has booted. Continue as usual. In the BC9020 and BC9120 DIP 9 and 10 must be off, in the BC9050 DIP switches 1 and 2 (2-pin DIP switch) must also be off.

Setting		Meaning
BC9050	**BC9020 BC9120**	
99	99	Manufacturer's settings
98	98	Delete boot project
97	97	Delete TwinCAT Config

99 DIP switches

DIP 1 ON	**DIP 2** ON	DIP 3 OFF	DIP 4 OFF	DIP 5 OFF	**DIP 6** ON	**DIP 7** ON	DIP 8 OFF

98 DIP switches

DIP 1 OFF	**DIP 2** ON	DIP 3 OFF	DIP 4 OFF	DIP 5 OFF	**DIP 6** ON	**DIP 7** ON	DIP 8 OFF

97 DIP switches

DIP 1 ON	DIP 2 OFF	DIP 3 OFF	DIP 4 OFF	DIP 5 OFF	**DIP 6** ON	**DIP 7** ON	DIP 8 OFF

4.3 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 (BX9000 only)	none
TwinCAT	Addressing via TwinCAT system manager [▶ 38]	PC with network and TwinCAT
BootP*	Addressing via BootP server [▶ 38]	BootP server
DHCP*	Addressing via DHCP server [▶ 40]	DHCP server
Local IP address*	Local IP address	If no BootP or DHCP server responds or is available

*) BX9000 from firmware version 1.20

4.3.1 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).

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.

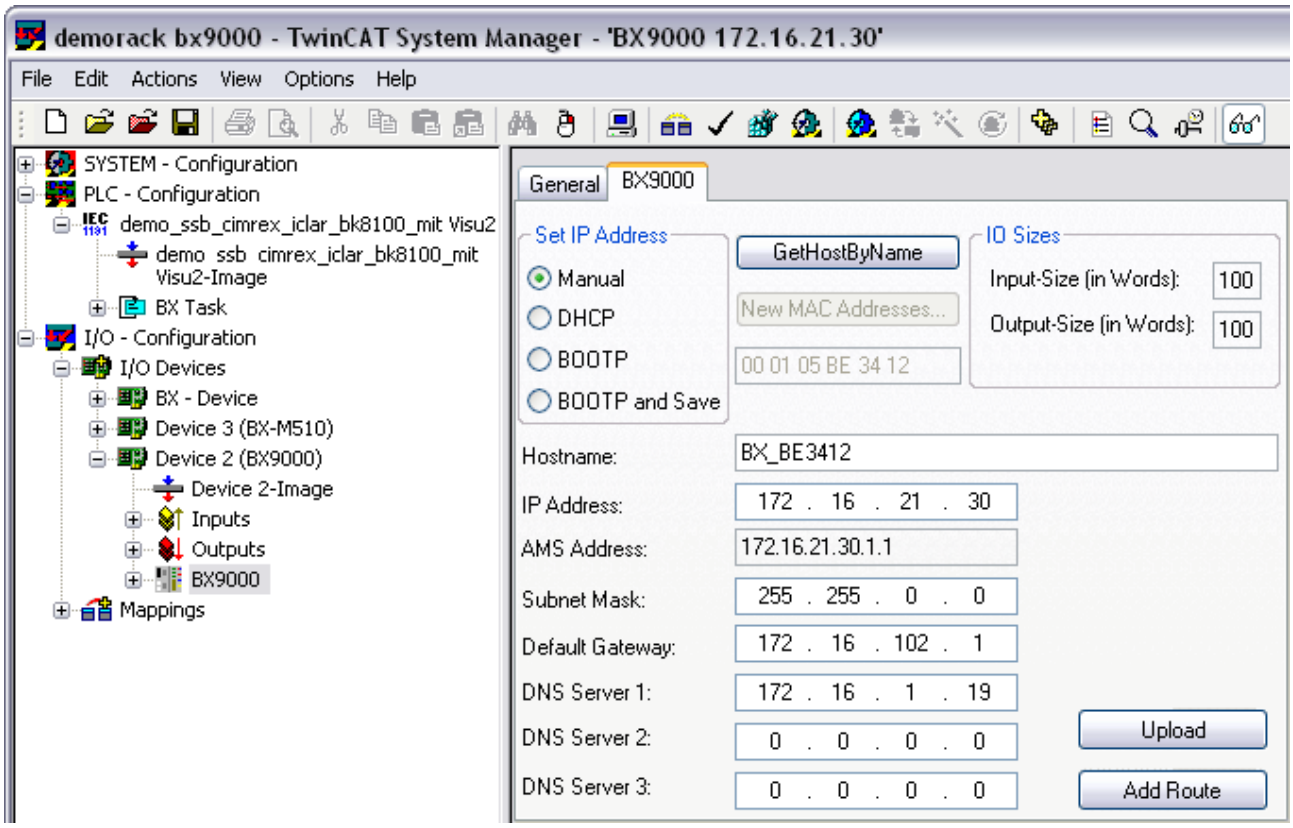


Fig. 22: Address Configuration via TwinCAT System Manager

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).

4.3.2 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 or First Steps).

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 https://download.beckhoff.com/

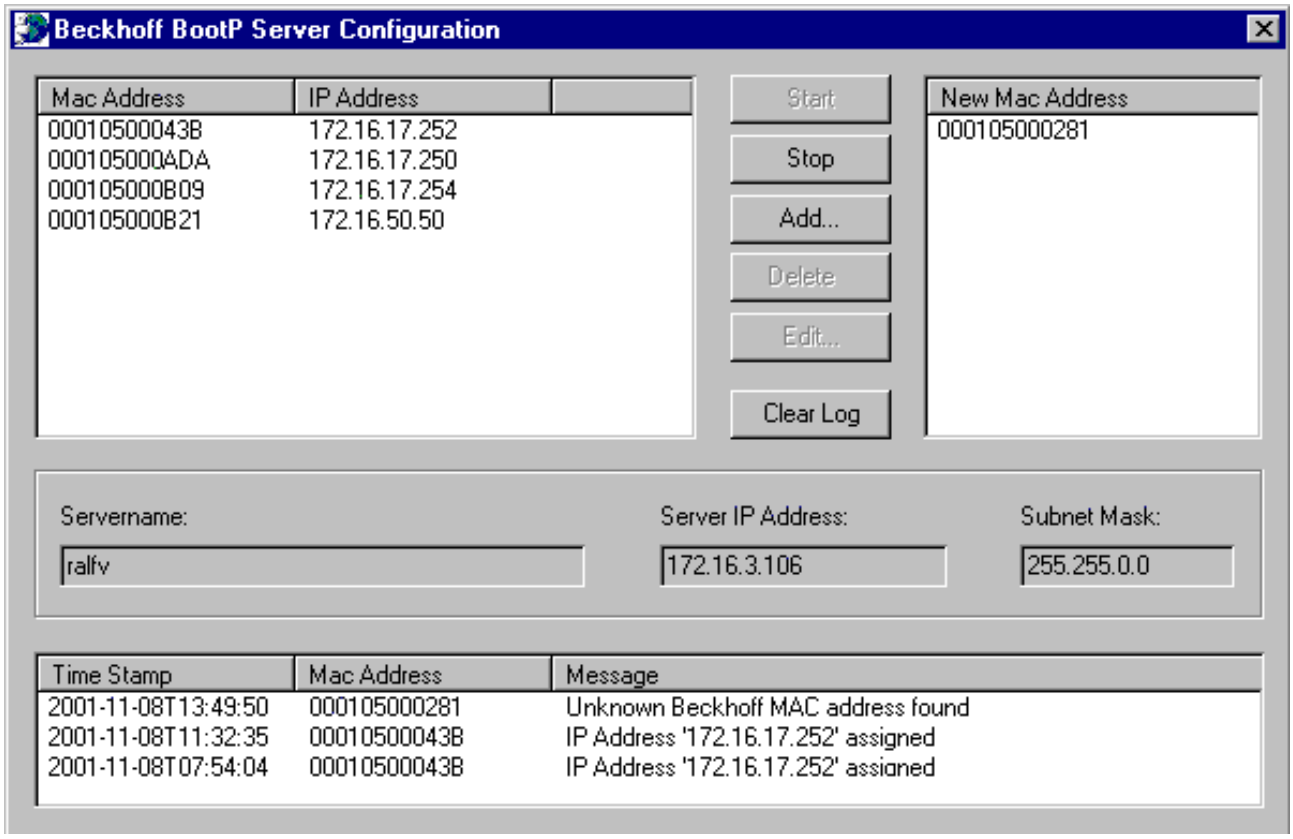


Fig. 23: Setting the address via BootP server

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).

4.3.3 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, or First steps).
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

4.3.4 Auto IP address

Auto IP address is activated if no DHCP or BootP server is found. This can take several minutes. The Auto IP address is formed as follows:

In the event of a timeout IP address 169.254.[MAC_05].[MAC06] is generated. If MAC_05 is 0 it is set to 1.

3 ARP probes are then sent. If no response is received a gratuitous ARP is sent and the IP is saved.

If the IP address already exists, the values for [MAC_05] and [MAC06] [MAC_04] are added up, and another attempt is made.

4.3.5 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 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.

4.4 Configuration

4.4.1 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

4.4.2 Finding the Bus Terminal Controller with the TwinCAT System Manager

Requirement 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, for instance 172.16.200.100 (sub-net 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:

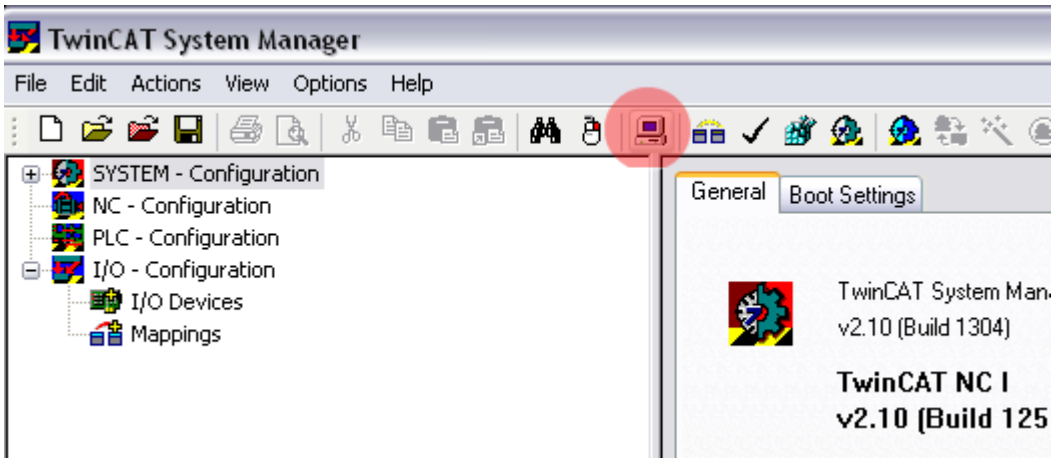


Fig. 24: Finding the Bus Terminal Controller

Now look for the Bus Terminal Controller via Ethernet:

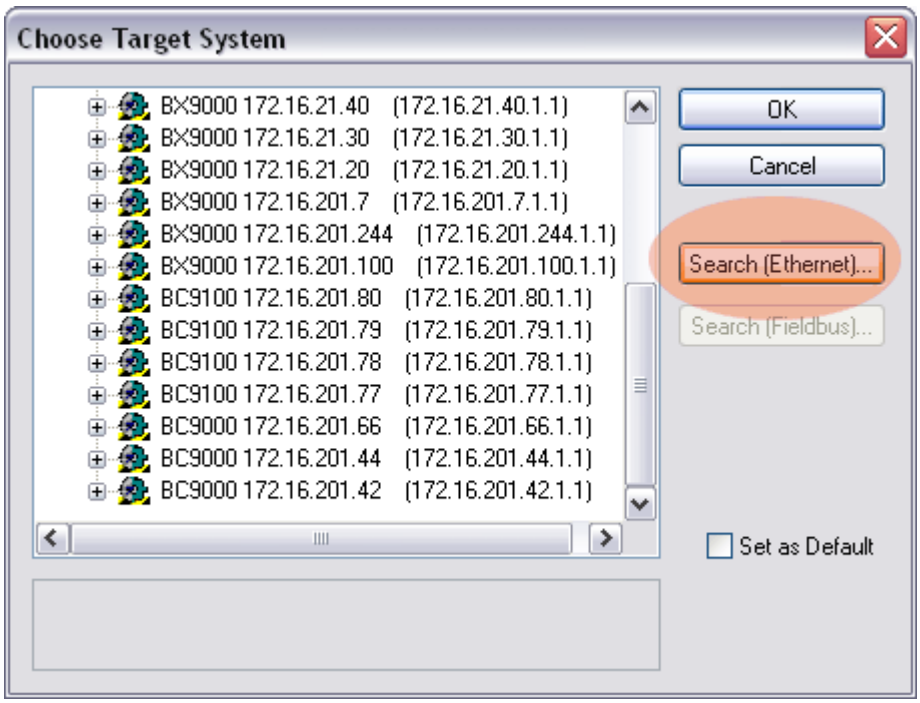


Fig. 25: Choose Target System

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 "BX_" or "BC_" and the last three bytes of the MAC ID. For the BX9000 the MAC ID can be found on the side, for the Bus Terminal controllers of the BC series on the underside of the housing. 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.

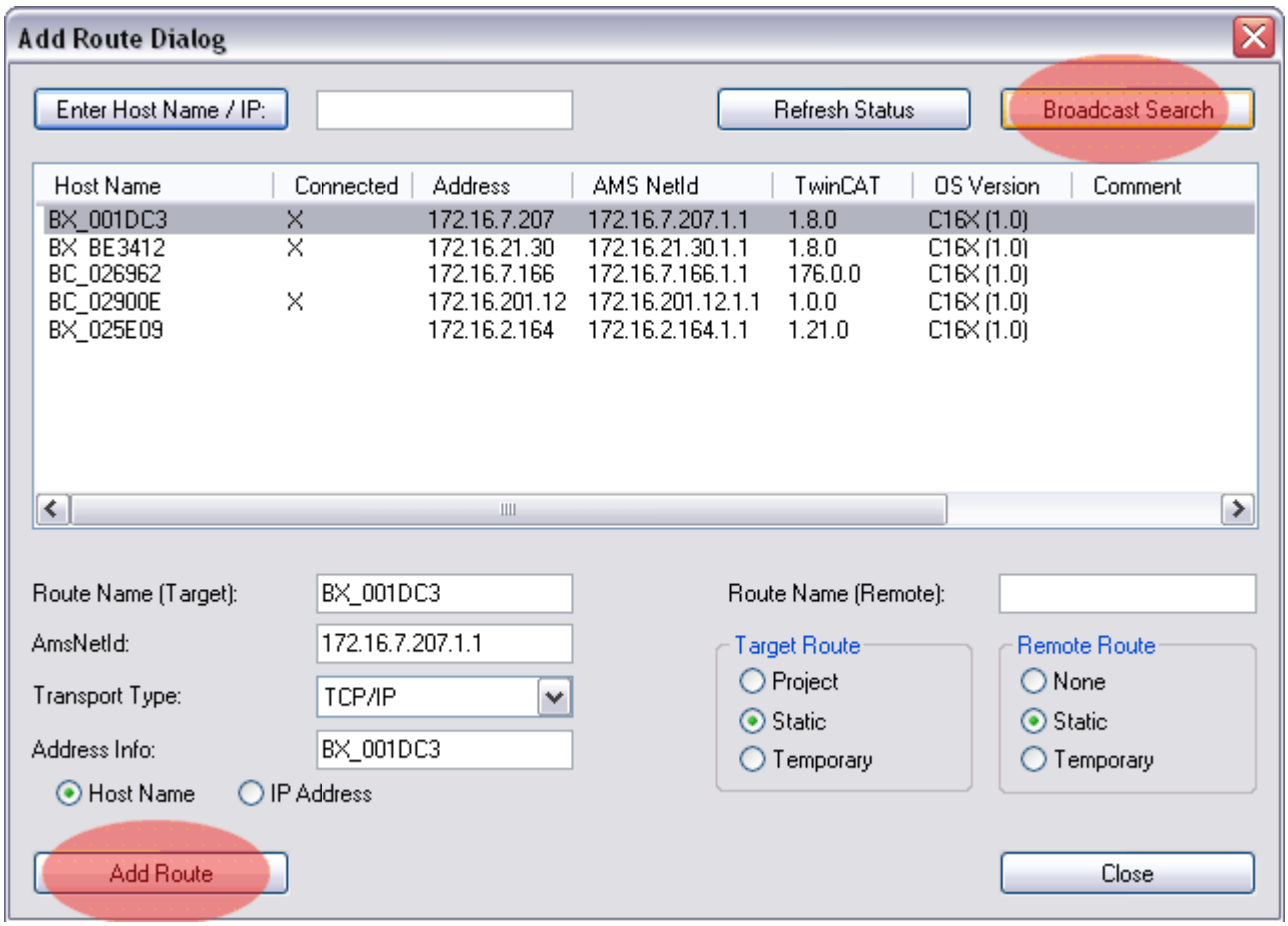


Fig. 26: Add Route Dialog

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).

4.4.3 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*.

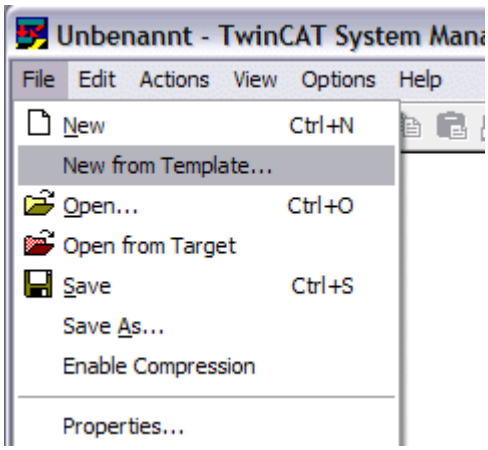


Fig. 27: Creating a TwinCAT configuration

Select the corresponding Bus Terminal Controller.

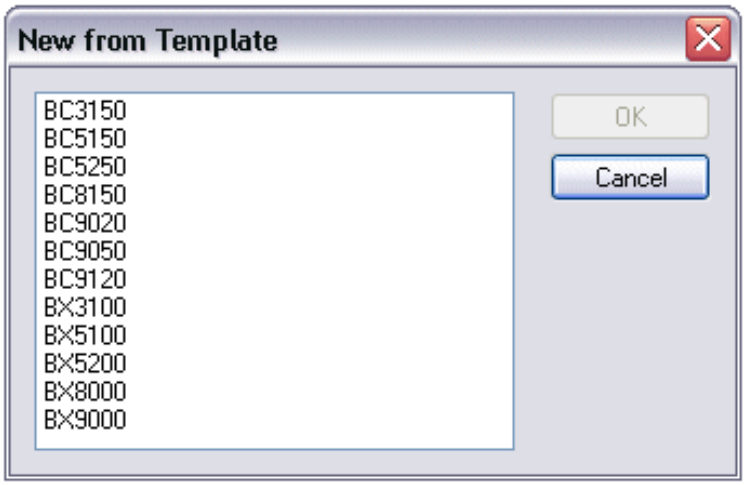


Fig. 28: Selecting the Bus Terminal Controller

All Bus Terminal Controller components are now available:

- Fieldbus interface
- K-bus interface [▶ 56]
- PLC Program [▶ 58]
- SSB (only Bus Terminal Controllers of the BX series)

Please refer to the relevant chapter for device configuration.

4.4.4 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 [▶ 52].

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.

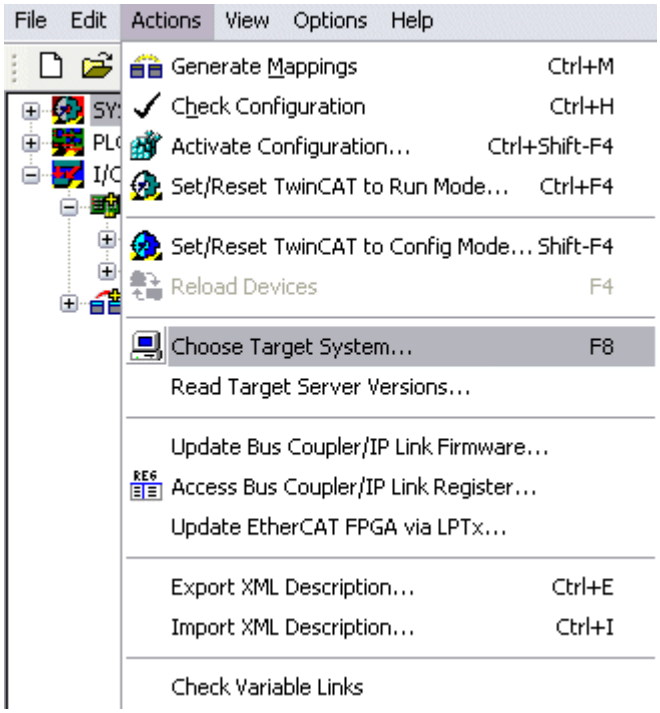


Fig. 29: Downloading a TwinCAT configuration

Select the corresponding Bus Terminal Controller.

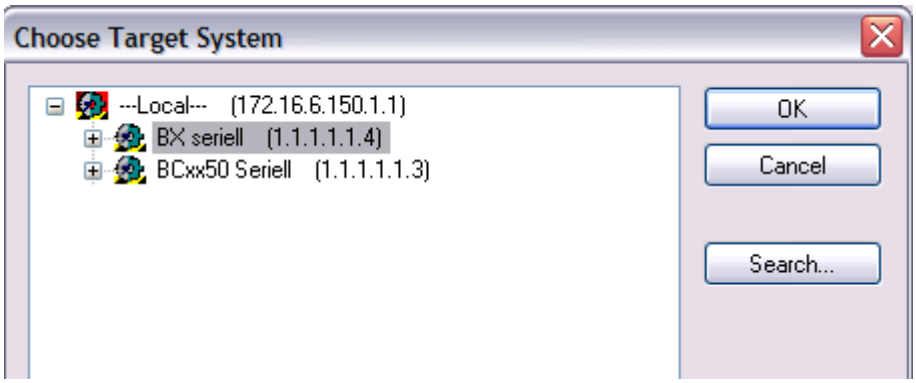


Fig. 30: Selecting the Bus Terminal Controller

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

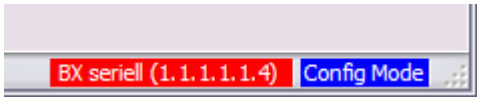


Fig. 31: State of the Bus Terminal Controller

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*.

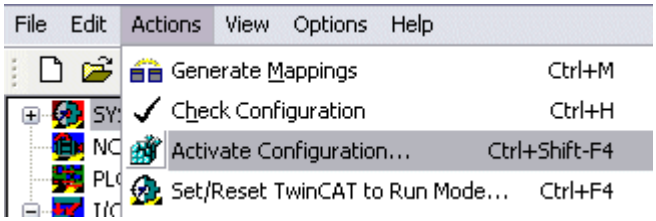


Fig. 32: Activating the TwinCAT configuration

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) [▶ 111].

4.4.5 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 [▶ 52].

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.

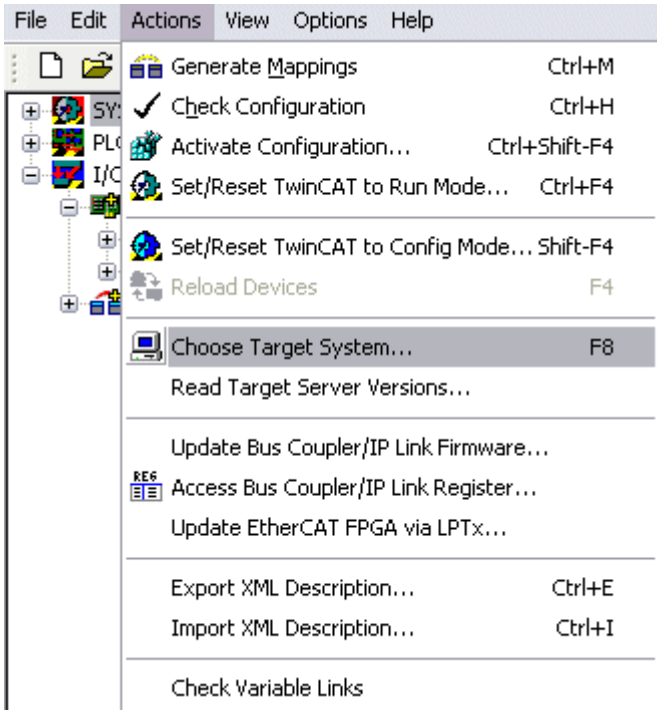


Fig. 33: Choose Target System

Select the corresponding Bus Terminal Controller.

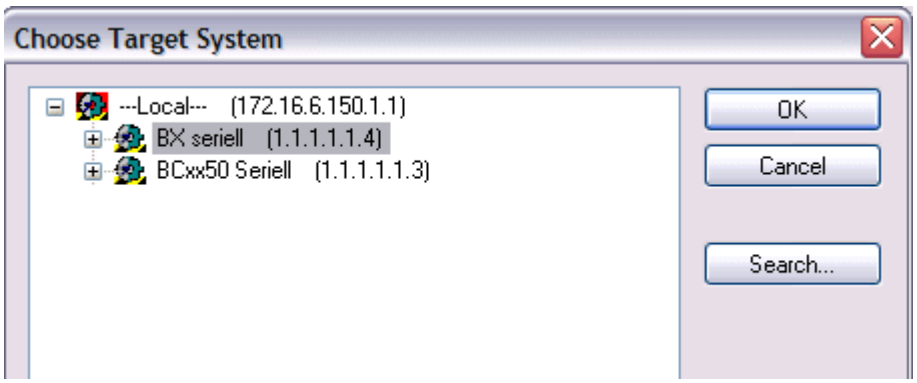


Fig. 34: Selecting the Bus Terminal Controller

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

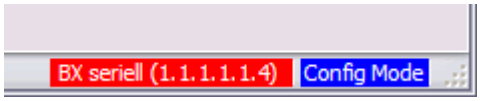


Fig. 35: State of the Bus Terminal Controller

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

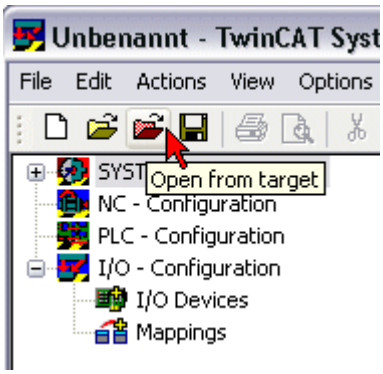


Fig. 36: Uploading the TwinCAT configuration

4.4.6 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.

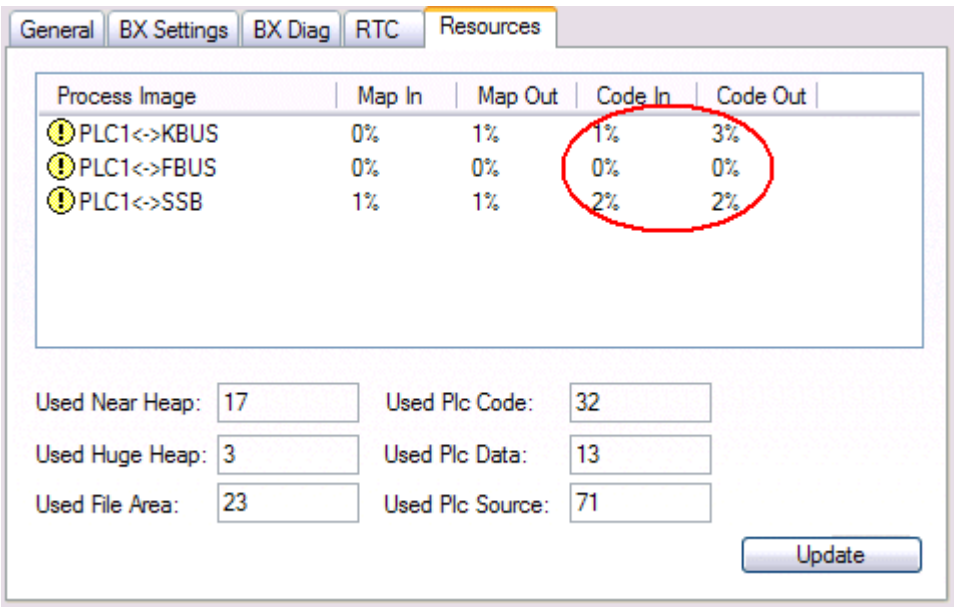


Fig. 37: Memory for code mapping

Data memory mapping

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

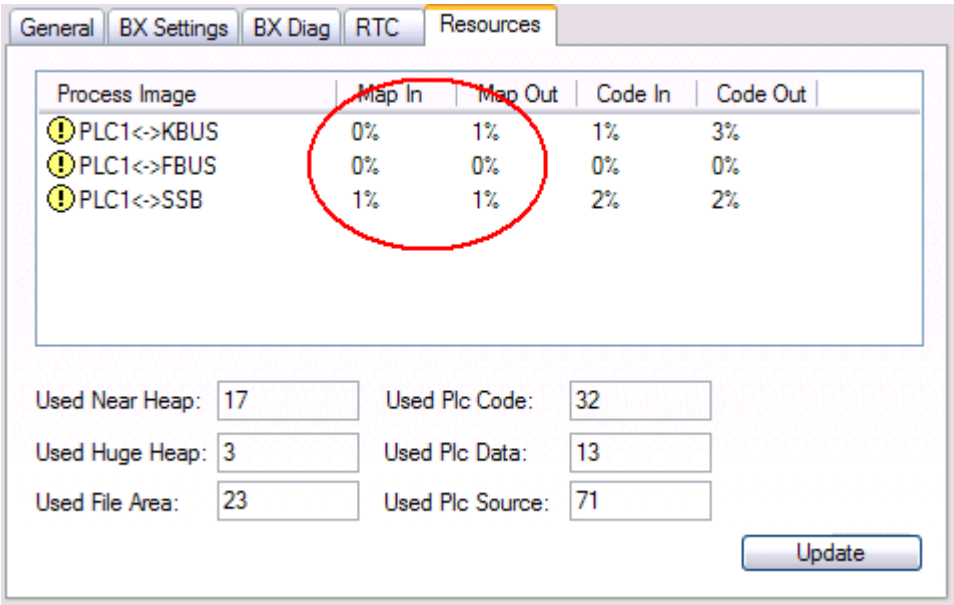


Fig. 38: Data memory mapping

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 %.

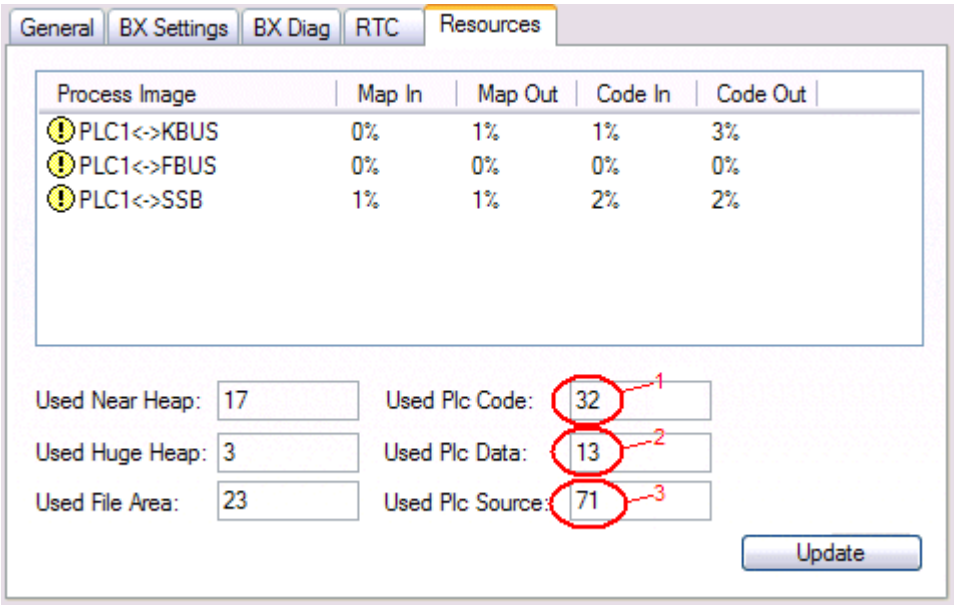


Fig. 39: Code and data memory

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.

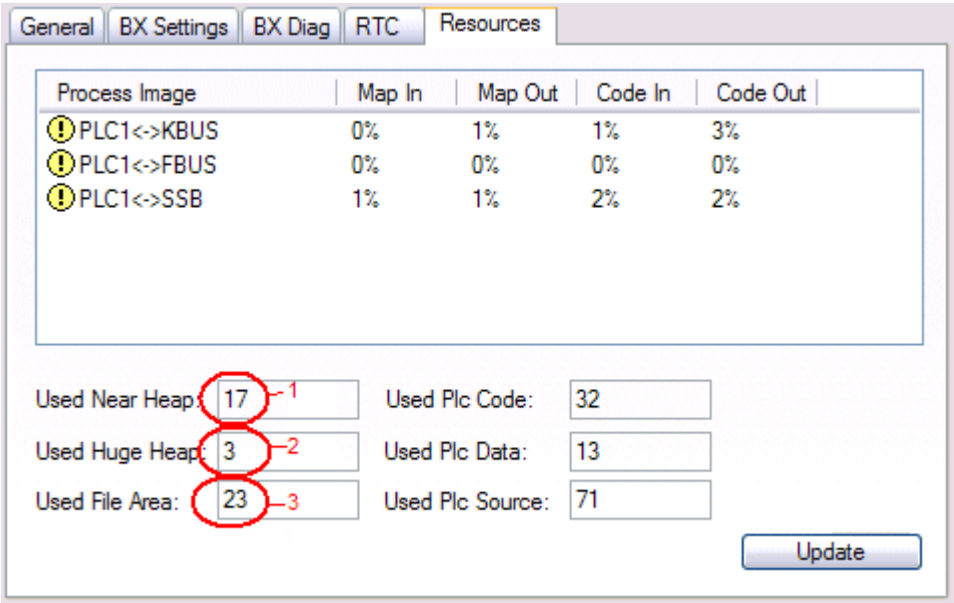


Fig. 40: Other memory

4.4.7 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 (TwnCAT level PLC, NC or NCI)

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.

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.

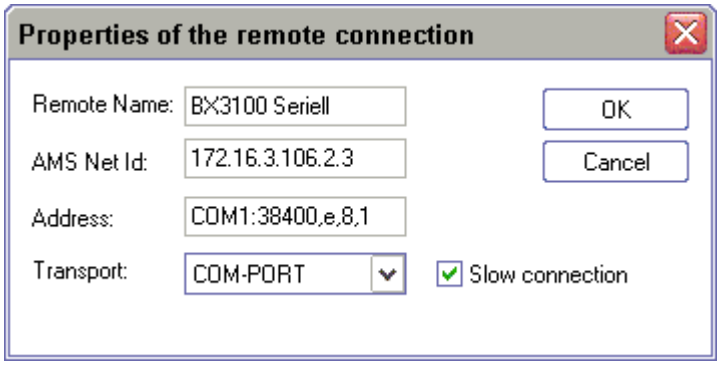


Fig. 41: Properties of the remote connection

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.

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).

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 1.1.1.1.1.1	AMS Net ID

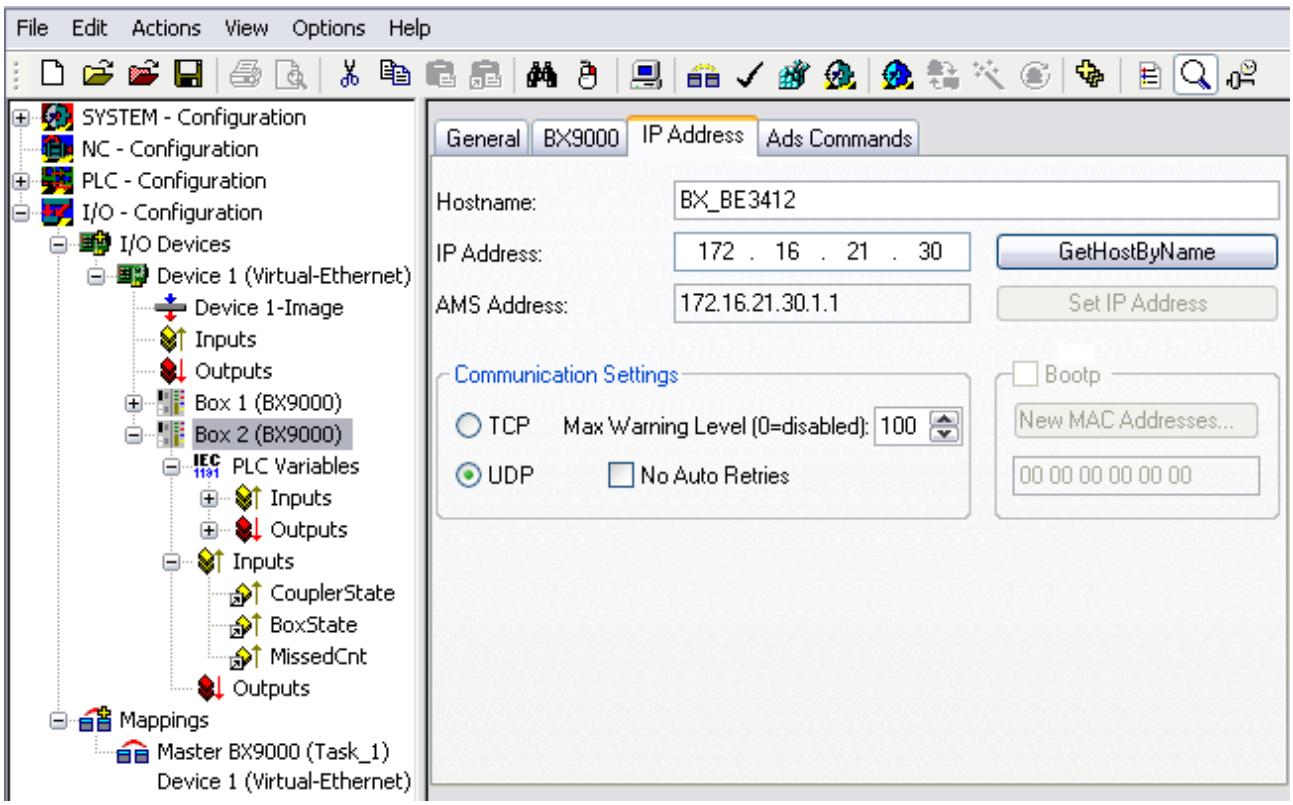
4.4.8 Ethernet

4.4.8.1 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.

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)



GetHostByName - searching for the IP address by name

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 the K-bus reports an error, for example
BoxState: see comment in the dialog
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 x 7 ms = 21 ms 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 x 7 ms = 14 ms 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 x 7 ms = 14 ms 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.

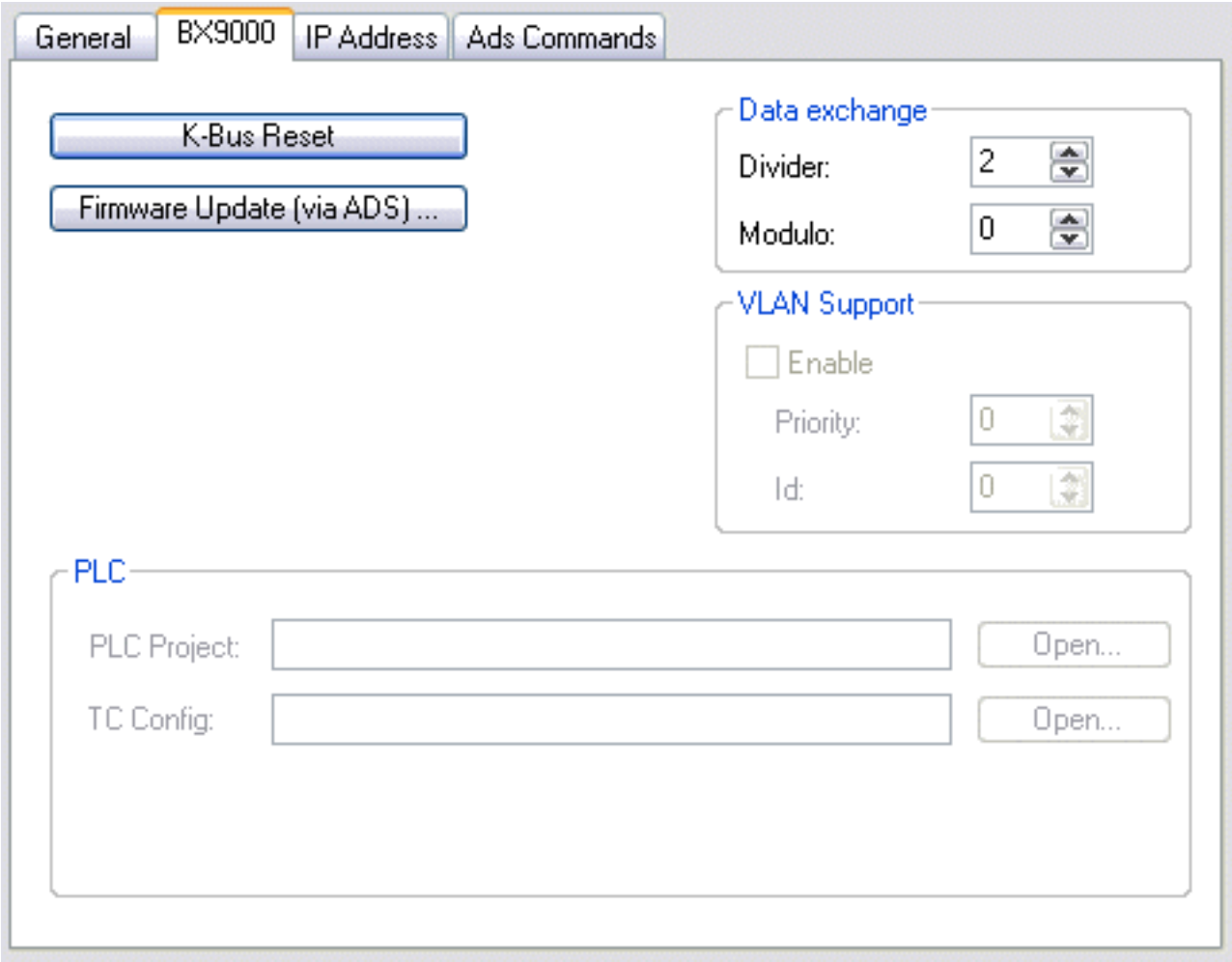


Fig. 42: Adjust the time after which the TwinCAT PLC queries a Bus Terminal Controller

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 x 10 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 1st task cycle and then after each 3rd task cycle.
If Modulo is set to 1, a telegram is sent after the 2nd task cycle and then after each 3rd 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.

4.4.9 K-bus

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

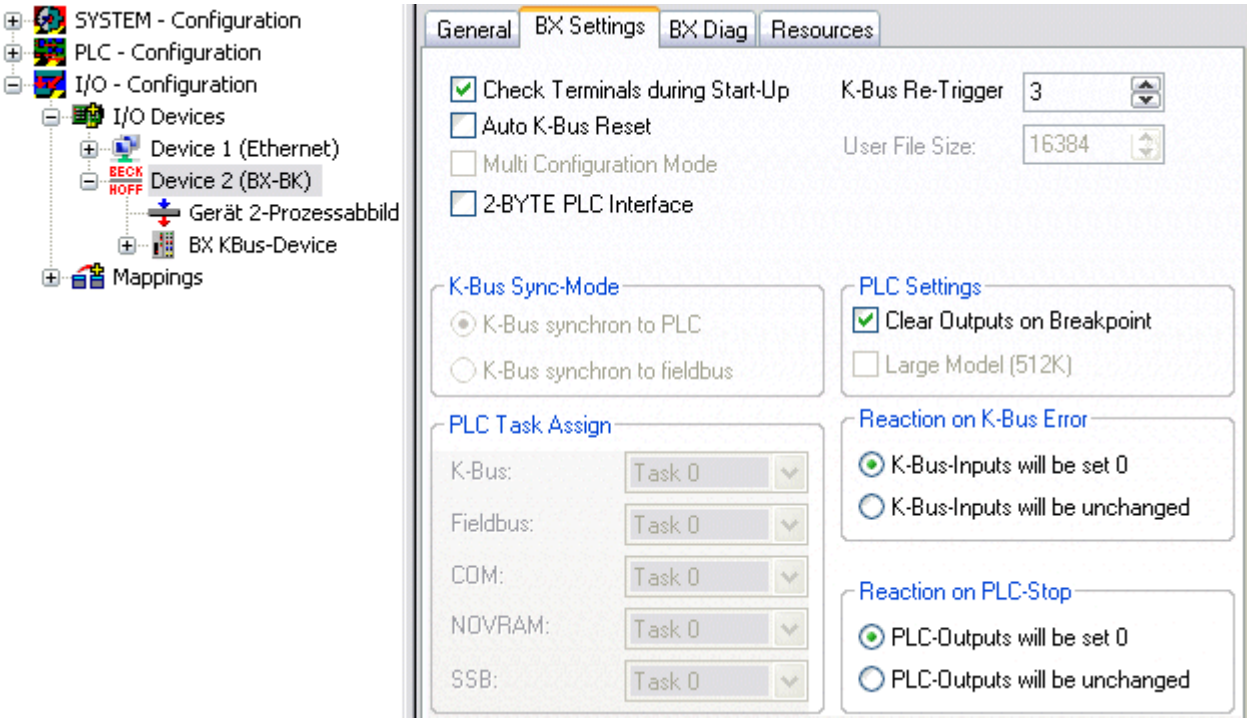


Fig. 43: BX Settings tab

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.

⚠ CAUTION

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 x 85 ms = 170 ms
K-Bus Re-Trigger 2: 3 x 85 ms = 255 ms
K-Bus Re-Trigger 3: 4 x 85 ms = 340 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.

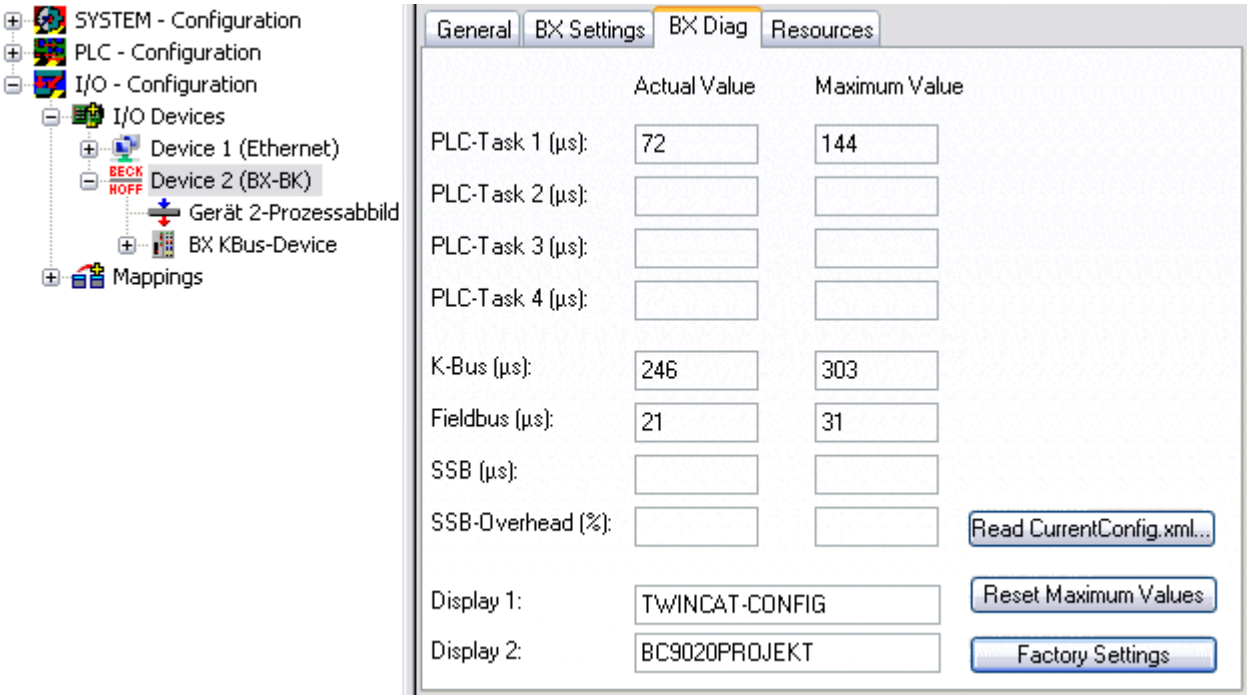
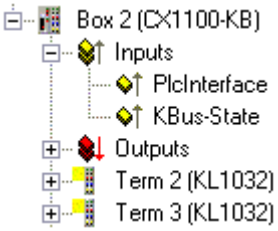


Fig. 44: BX Diag tab

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

4.4.10 PLC

4.4.10.1 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.

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 [▶ 73])

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.

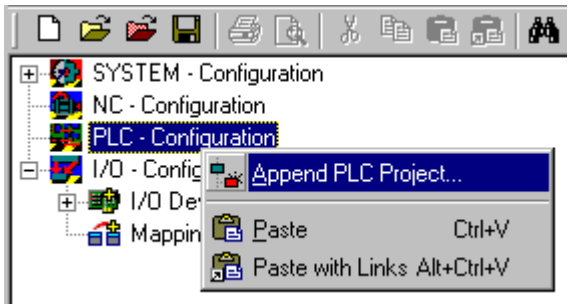


Fig. 45: Selecting the PLC project

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

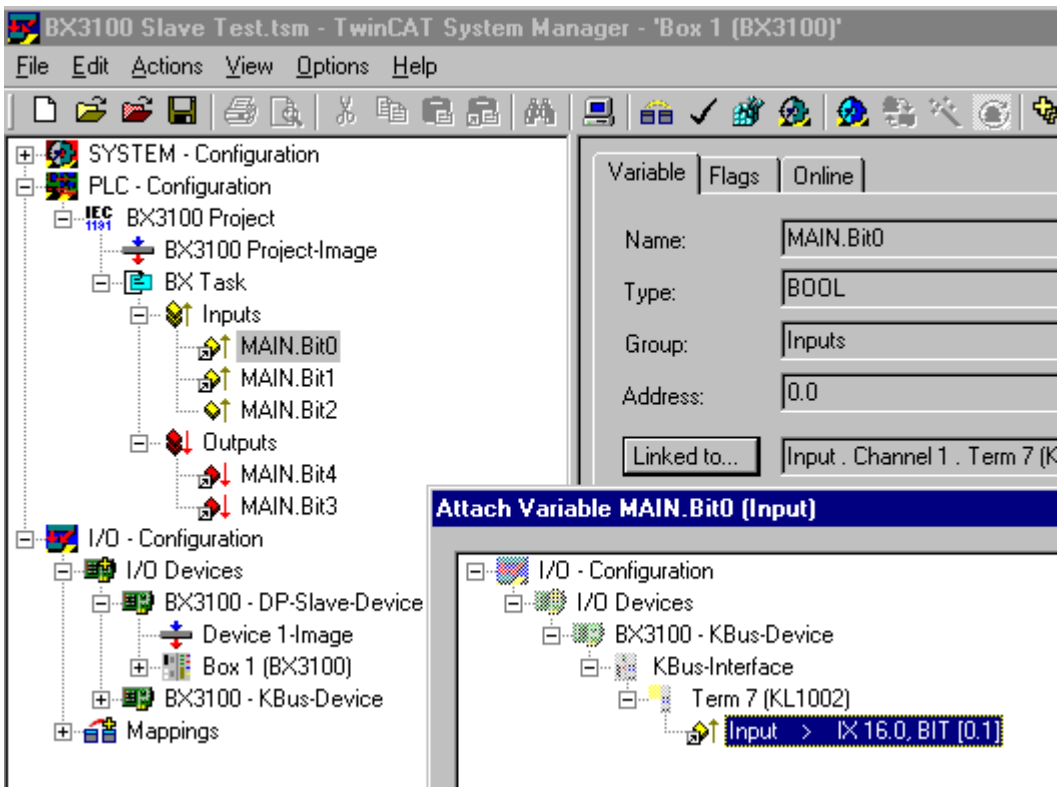


Fig. 46: Connecting PLC variable and hardware

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).

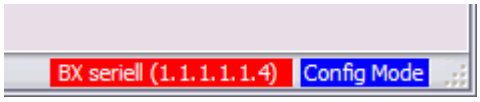


Fig. 47: Target system display

4.4.10.2 Measuring the PLC cycle time

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

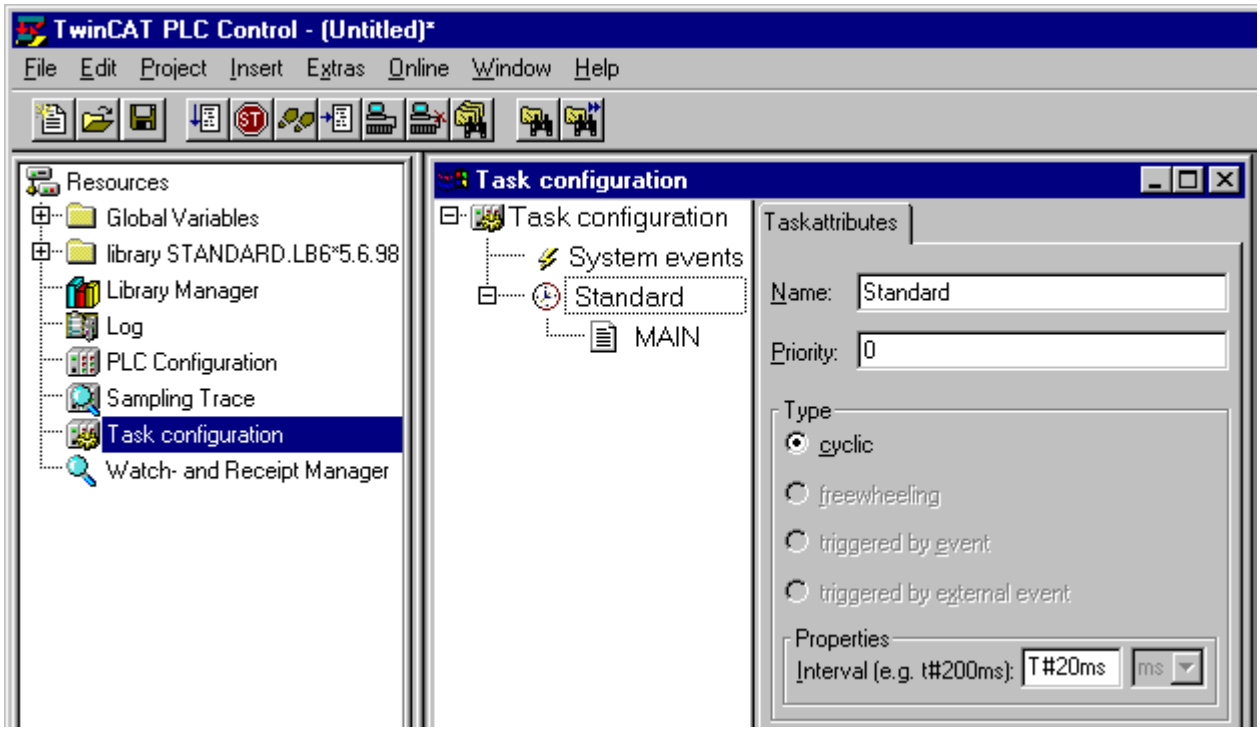


Fig. 48: Setting the task time

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 [▶ 56] description. The total cycle time is displayed with the TcBase library (see TcBase.lbx or TcBaseBCxx50.lbx).

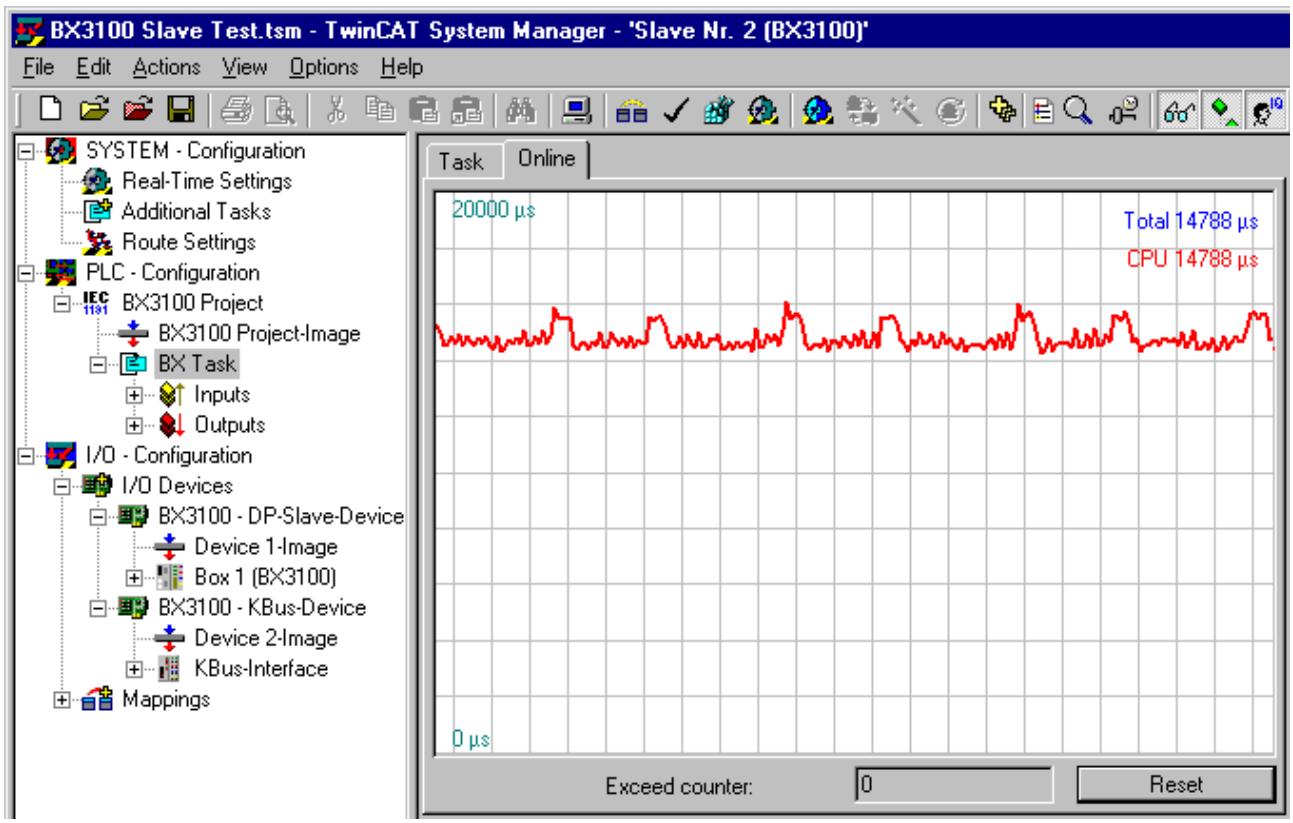


Fig. 49: Displaying the PLC cycle time

4.5 KS2000

4.5.1 KS2000 Configuration Software

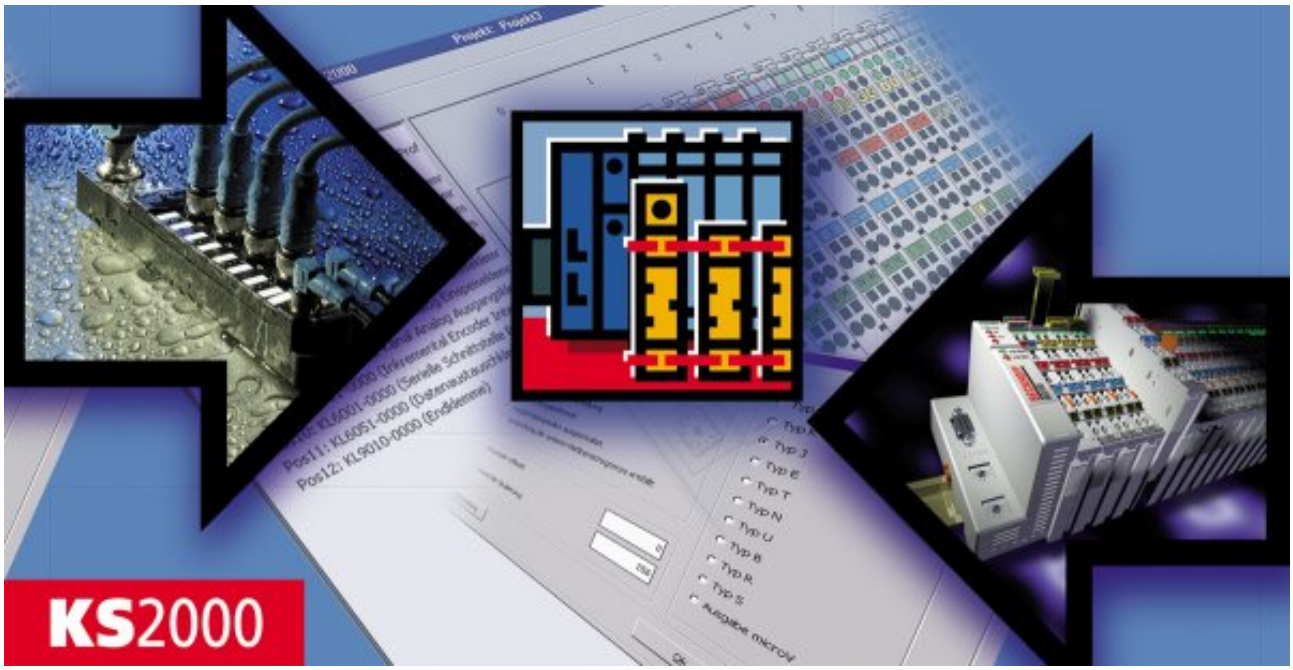


Fig. 50: KS2000 Configuration Software

The KS2000 configuration software makes parameterization and configuration of the BC9x20 and BC9050 Bus Terminal Controllers unnecessary. The configuration can be carried out with the TwinCAT System Manager.

The KS2000 configuration software offers configuration and diagnostic support for the Bus Terminals attached to the Bus Terminal Controller.
To this end it is advisable to set the baud rate in the KS2000 configuration software to 38400 baud (8 data bits, even, 1 stop bit).

COM1 - automatic baud rate detection

The COM 1 Controller interface features automatic baud rate detection between 9.6 kbaud and 38.4 kbaud.

Required KS2000 version

Bus Terminal configuration and diagnostics is supported from KS2000 version 4.3.0.65.

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 Controller must be in the default configuration. Set the manufacturer's setting or switch to Config Mode in the TwinCAT System Manager (blue TwinCAT icon).
- The 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

BC9050

Description	Value
Data memory	32 kbyte
Program memory	48 kbyte minus task-configuration minus POUs during online change
Source code memory	128 kbyte
RETAIN	2 kbyte
Persistent data	1000 bytes
INPUT	2 kbyte
OUTPUT	2 kbyte
FLAG	4 kbyte
Max. variable size	16 kbyte
Max. POUs	Limited by memory

BC9020

Description	Value
Data memory	128 kbyte
Program memory	128 kbyte minus task-configuration minus POUs during online change
Source code memory	256 kbyte
RETAIN	2 kbyte
Persistent data	1000 bytes
INPUT	2 kbyte
OUTPUT	2 kbyte
FLAG	4 kbyte
Max. variable size	16 kbyte
Max. POUs	Limited by memory

BC9120

Description	Value
Data memory	128 kbyte
Program memory	128 kbyte minus task-configuration minus POUs during online change
Source code memory	256 kbyte
RETAIN	2 kbyte
Persistent data	1000 bytes
INPUT	2 kbyte
OUTPUT	2 kbyte
FLAG	4 kbyte
Max. variable size	16 kbyte
Max. POUs	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 [▶ 112]
- via the fieldbus interface [▶ 111] (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 POUs (...) exceeded
PLC compiler error	Out of global data memory ...

Error POUs

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

```
Error 3612: Maximum number of POUs (100) exceeded! Compile is aborted.
Data allocation
1 Error(s), 0 Warning(s).
```

Fig. 51: Maximum number of POUs exceeded

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

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

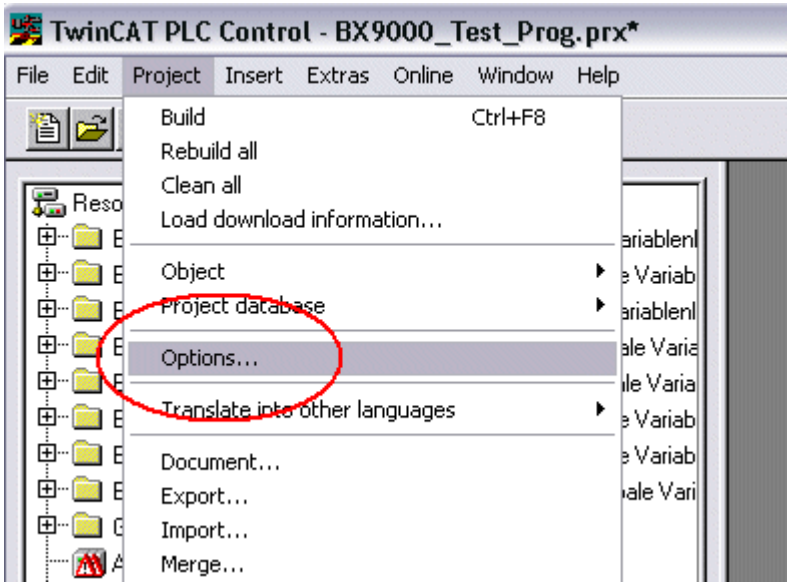


Fig. 52: Menu path Projects / Options / Controller Settings

...the controller settings.

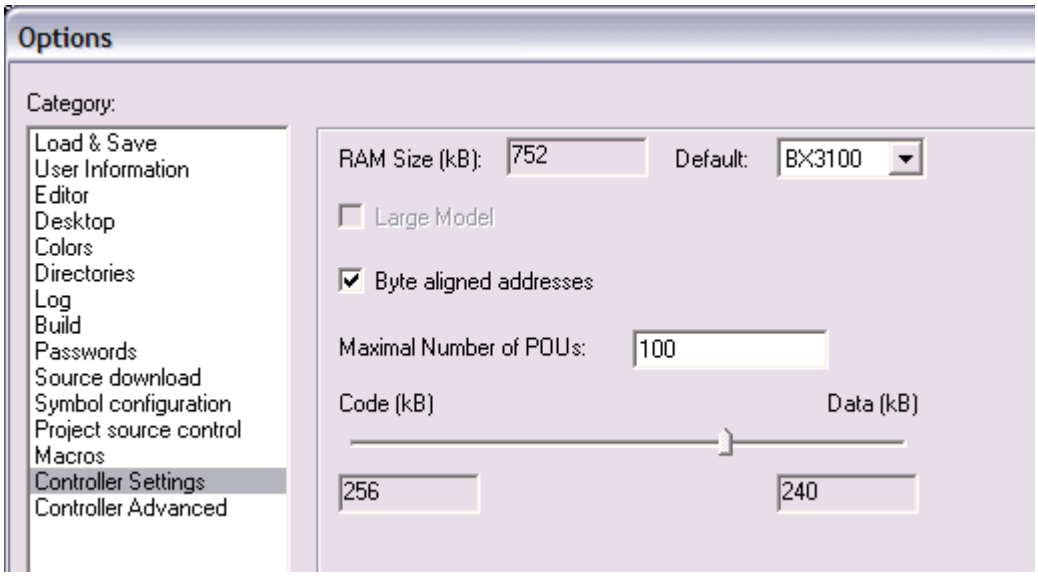


Fig. 53: Controller settings

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. 54: 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.

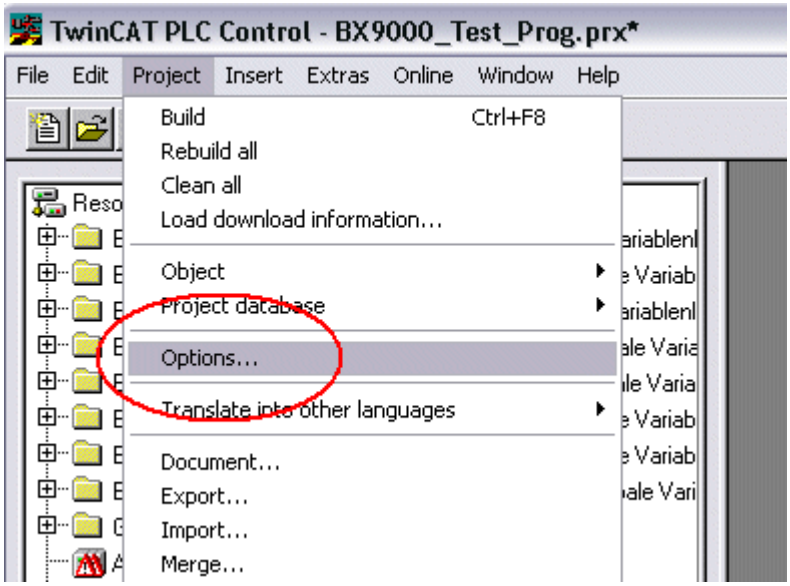


Fig. 55: Menu path Projects / Options / Build

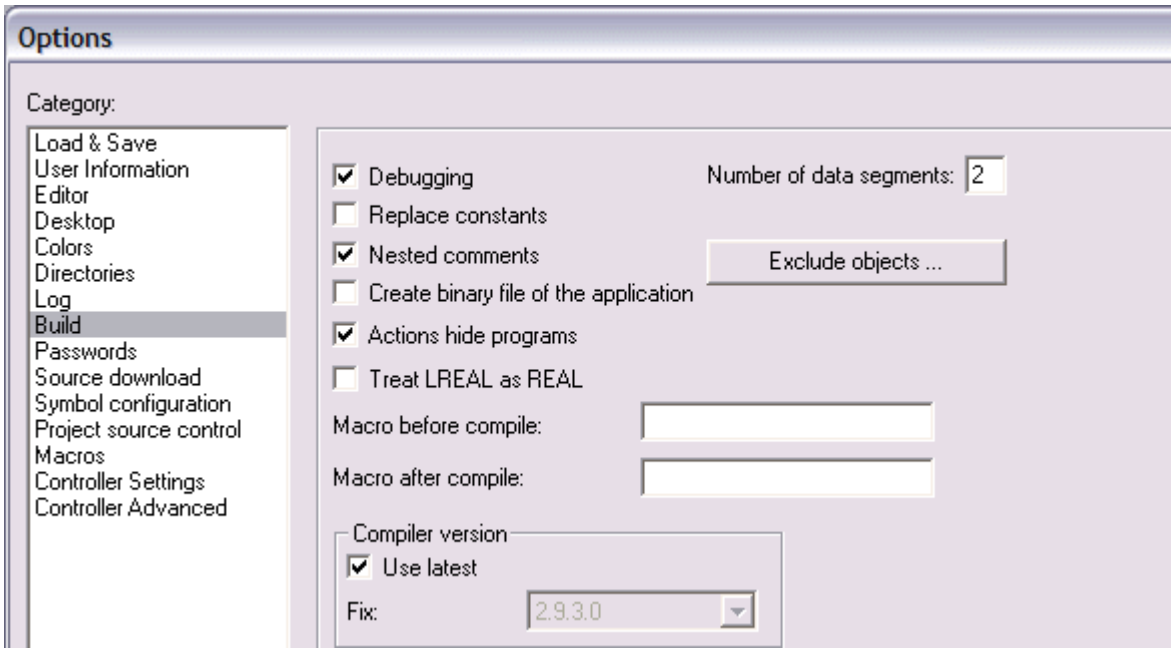


Fig. 56: Build

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.

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.

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).

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.

Persistent data from firmware 1.17

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

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 (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207307659.zip) 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.

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_{dec}. 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.

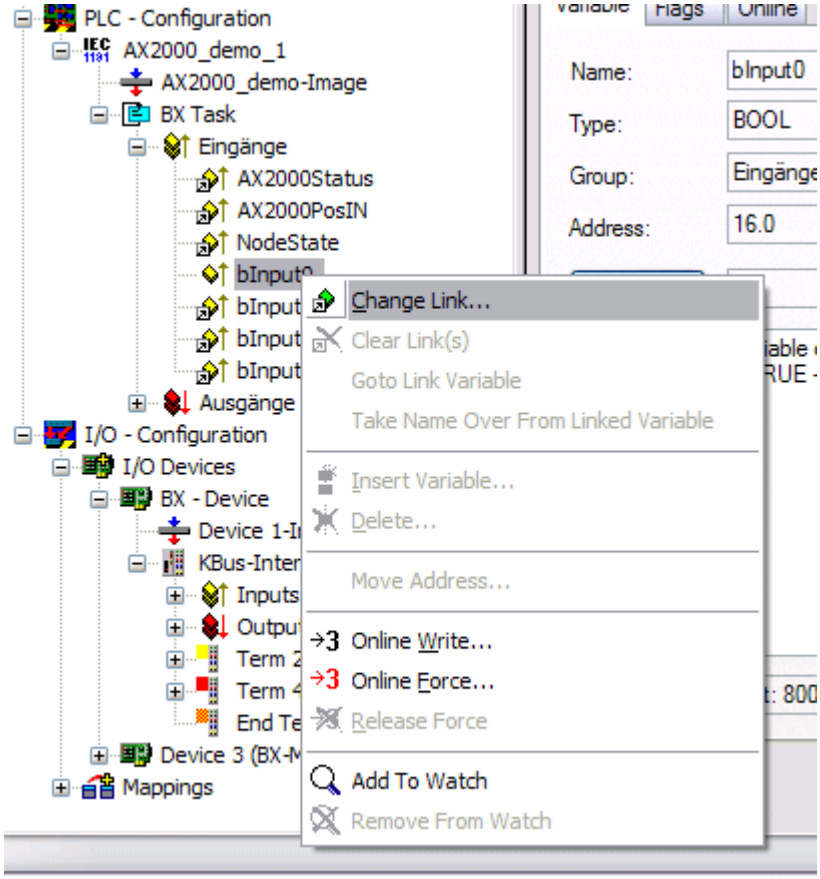


Fig. 57: Changing variable links

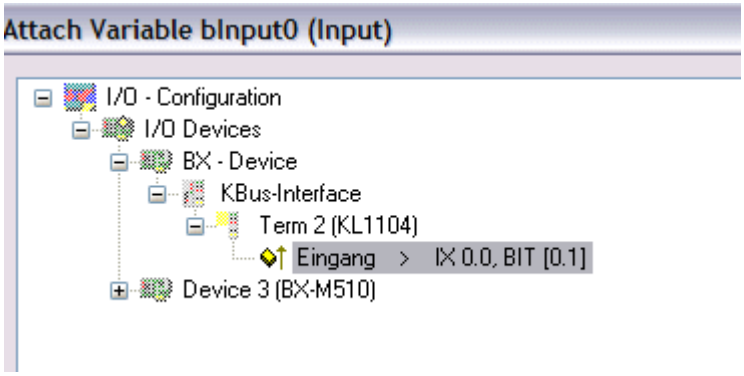


Fig. 58: Linking a variable with an input

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 [▶ 41]). 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.

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;
```

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.

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.

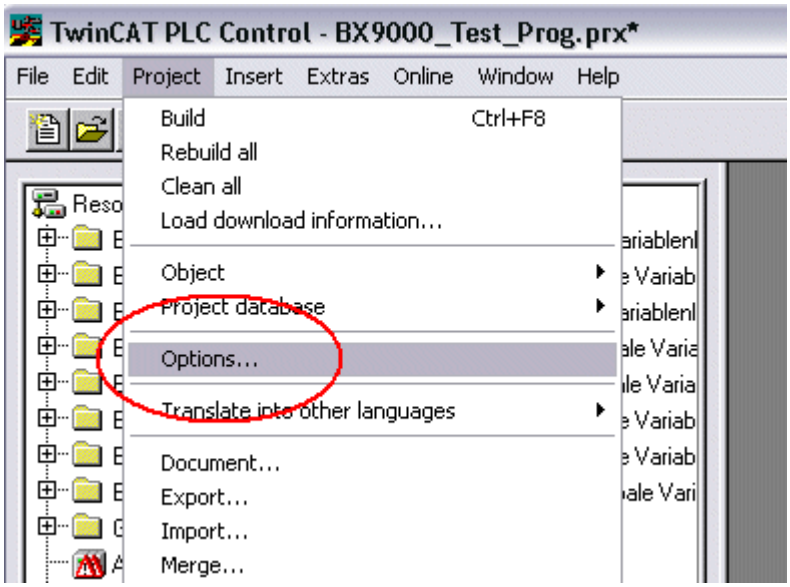


Fig. 59: Opening the options menu

Select Source Download.

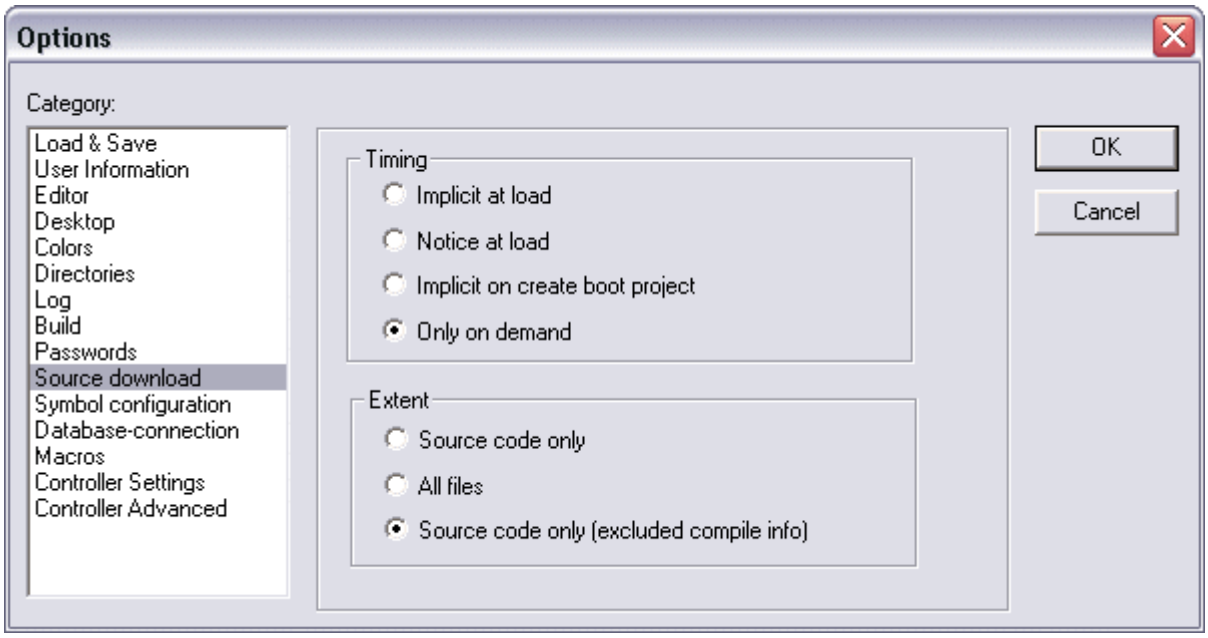


Fig. 60: Selecting Source Download

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.

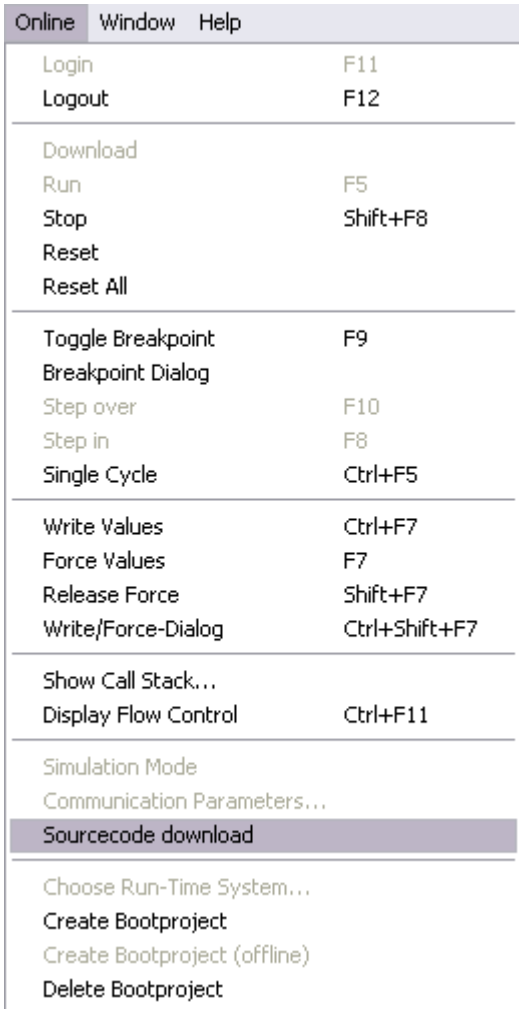


Fig. 61: Downloading the program code

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

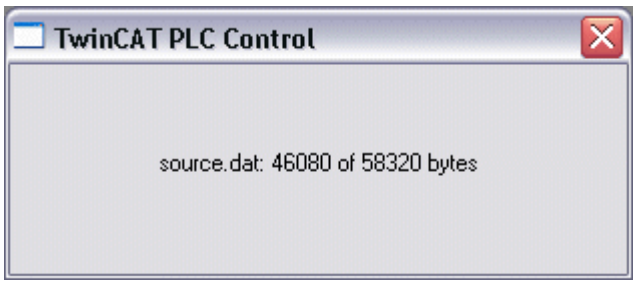


Fig. 62: Download progress

Uploading a program

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

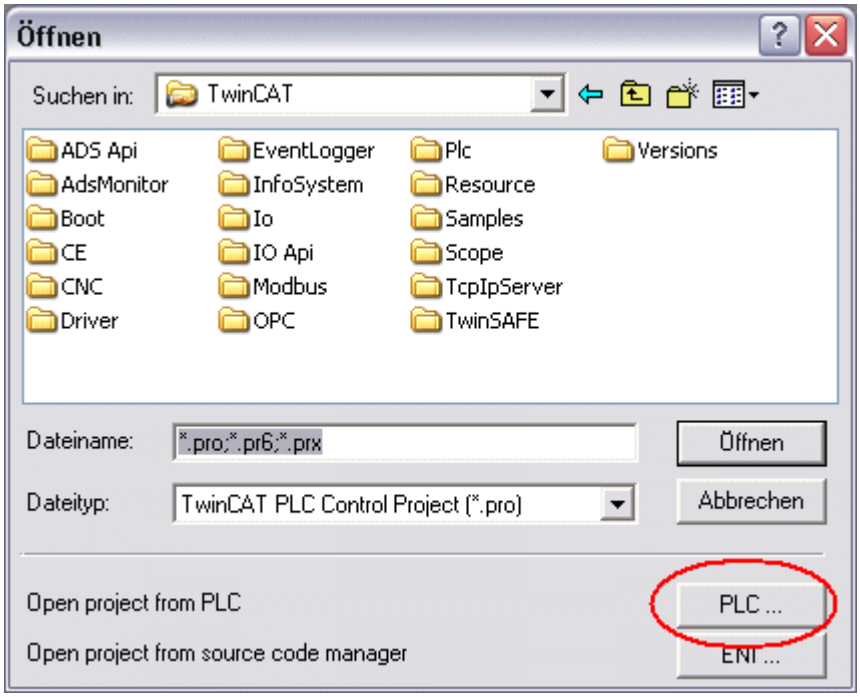


Fig. 63: Uploading a program

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.

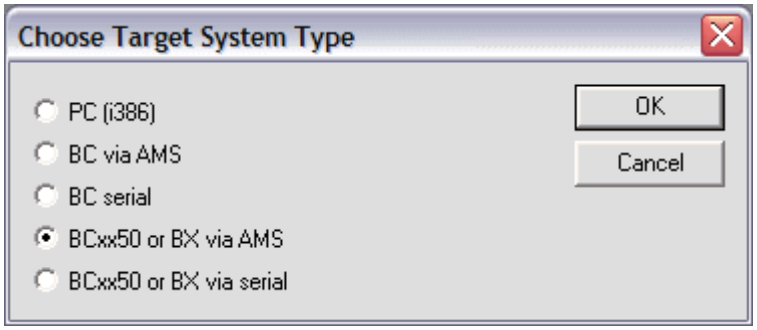


Fig. 64: Selecting the data transfer route

Then select the device and confirm with OK.

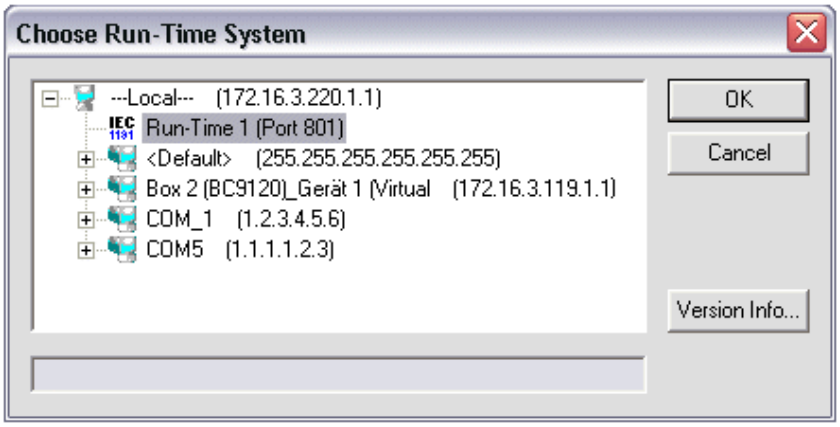


Fig. 65: Selecting the device

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

The TwinCAT automation software provides various libraries for Bus Terminal Controllers (Bus Couplers with PLC functionality) of the BC9050, BC9020 and BC9120 series (see Beckhoff Information System).

Download

The libraries are also included in this documentation. To extract the libraries, left-click on the link and copy the libraries to directory TwinCAT\PLC\LIB.

- Standard
- TcSystemBCxx50 (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3740875659.zip)

  TcSystemBCxx50 requires the TcBaseBCxx50 library.
- TcBaseBCxx50 (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3740877835.zip)
- ChrAscBX.lbx (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3740880011.zip)
- TcBaseBX9000.lbx [▶ 85]

Use the library that matches the firmware

Always use the latest libraries in conjunction with the latest BC firmware. If you update the firmware of your Bus Terminal Controller, please also update the libraries. Copy the new libraries into the LIB folder, remove them from your project and re-insert them.

TcSystemBCxx50

ADS	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
ADSREAD		B0	B0	B1	-	-
ADSWRITE		B0	B0	B1	-	-
ADSRDWRT		B0	B0	B1	-	-
ADSWRTCTL		B0	B0	B1	-	-
ADSRDSTATE		B0	B0	B1	-	-
ADSRDDEVINFO		B0	B0	B1	-	-

Bit Functions	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
CLEARBIT32		B0	B0	B1	-	-
CSETBIT32		B0	B0	B1	-	-
GETBIT32		B0	B0	B1	-	-
SETBIT32		B0	B0	B1	-	-

Controller	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
FB_BasicPID	-	B0	B0	B1	-	-
-	-	-	-	-	-	-

File Access	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
FB_ReadFromFile		-	-	-	-	-
FB_WriteToFile		-	-	-	-	-
FB_ReadWriteFile		-	-	-	-	-

Memory Functions	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
MEMCMP		B0	B0	B1	-	-
MEMCYP		B0	B0	B1	-	-
MEMMOVE		B0	B0	B1	-	-
MEMSET		B0	B0	B1	-	-

NOVRAM Functions	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
-	-	-	-	-	-	-

SFC	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
AnalyzeExpression		-	-	-	-	-
AppendErrorString		-	-	-	-	-
SFCActionControl		-	-	-	-	-

System / Time / TBus	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
DRAND		B0	B0	B1	-	-
SYSTEMTIME_TO_DT		B0	B0	B1	-	-
DT_TO_SYSTEMTIME		B0	B0	B1	-	-
GetSysTick		B0	B0	B1	-	-
PresetSysTick		B0	B0	B1	-	-
Reboot		B0	B0	B1	-	-

Debug	Version	Firmware				
		BC9050	BC9020	BC9120	-	-
F_ReadDebugTimer		B0	B0	B1	-	-
F_StartDebugTimer		B0	B0	B1	-	-

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 (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207318539.zip)

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 [▶ 105]	30.05.06	1.12	B0	B0	B1
FB_GetHostByAddr [▶ 106]	30.05.06	1.12	B0	B0	B1
FB_GetHostByName [▶ 107]	30.05.06	1.12	B0	B0	B1
FB_GetNetworkConfig [▶ 108]	30.05.06	1.12	B0	B0	B1
FB_SetTargetName [▶ 109]	30.05.06	1.12	B0	B0	B1

SMTP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_SMTP [▶ 102]	30.05.06	1.12	B0	B0	B1

SNTP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_SNTP [▶ 104]	30.05.06	1.12	B0	B0	B1

IP-TCP/IP-UDP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_AddMultiRoute [▶ 87]	26.09.06	1.14	-	-	-
FB_DelMultiRoute [▶ 87]	26.09.06	1.14	-	-	-
FB_IpClose [▶ 87]	26.09.06	1.14	B0	B0	B1
FB_IpEndSession [▶ 87]	26.09.06	1.14	B0	B0	B1
FB_IpOpen [▶ 87]	26.09.06	1.14	B0	B0	B1
FB_IpReceive [▶ 87]	26.09.06	1.14	B0	B0	B1
FB_IpSend [▶ 87]	26.09.06	1.14	B0	B0	B1
FB_IpStartSession [▶ 87]	26.09.06	1.14	B0	B0	B1

ModbusTCP	Version	Firmware			
		BX9000	BC9020	BC9050	BC9120
FB_MBClose [▶ 98]	26.09.06	1.14	B0	B0	B1
FB_MBConnect [▶ 98]	26.09.06	1.14	B0	B0	B1
FB_MBGenericReq [▶ 98]	26.09.06	1.14	B0	B0	B1

5.13.3 TcBaseBCxx50

5.13.3.1 System task information

```
VAR_GLOBAL
    SystemTaskInfo  :  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 SYTEMTASKINFOTYPE [▶ 81]. 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	BCxx50, BC9x20 Controller	TcBaseBCxx50.lbx

5.13.3.2 System Task Info 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
```

Key

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	BCxx50, BC9x20 Controller	TcBaseBCxx50.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 [▶ 82]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	BCxx50, BC9x20 Controller	TcBaseBCxx50.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
```

Key

runTimeNo: indicates the number of the runtime system (1).
projectName: project name as STRING.
numberOfTasks: number of tasks contained in the runtime system (max. 1).
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	BCxx50, BC9x20 Controller	TcBaseBCxx50.lbx

5.13.3.5 ADS

5.13.3.5.1 Local ADS Port Numbers - Overview

Port number	Description
100 [▶ 84] dec	Reading and writing of registers and tables from the coupler and the complex Bus Terminals
150 [▶ 84] dec	Reading and writing of RTC (real-time clock)
153 [▶ 84] dec	SSB - reading of the emergency message
800 [▶ 83] dec	Local process image of the PLC, see also port 801
801 [▶ 83] dec	Local process image of the PLC, see also port 800
0x1000 + Node ID [▶ 83]	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 Stop - the local PLC is in stop mode

AdsServerDeviceState

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

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	Bus Coupler register High-word table number of the Bus Coupler Low-word register number of the table
1...255	0...3	1...255	Bus Terminal registers High-word channel number Low-word register number of the Bus Terminal

Minimum timeout

When reading the register, the time out for the ADS block has to be set to a time longer than 1 second.

Setting the password

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

5.13.3.6 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

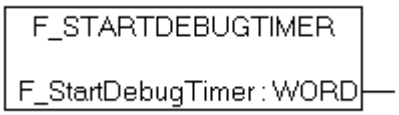


Fig. 66: Function block F_STARTDEBUGTIMER

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

Read Debug Timer function

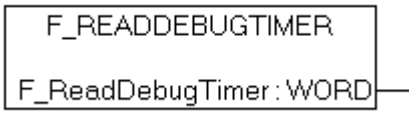


Fig. 67: Function block F_READDEBUGTIMER

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();
```

5.13.4 TcBaseBX9000

5.13.4.1 Overview of libraries for BC9020, BC9050, BC9120

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

Download

To download the libraries left-click on the link. Then copy the libraries into the directory TwinCAT\PLC\LIB.

- TcBaseBX9000 (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3740882187.zip)

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		
		BC9020	BC9050	BC9120
FB_AddDnsServer	30.05.06	B0	B0	B1
FB_GetHostByAddr	30.05.06	B0	B0	B1
FB_GetHostByName	30.05.06	B0	B0	B1
FB_GetNetworkConfig	30.05.06	B0	B0	B1
FB_SetTargetName	30.05.06	B0	B0	B1

SMTP	Version	Firmware		
		BC9020	BC9050	BC9120
FB_SMTP	30.05.06	B0	B0	B1

SNTP	Version	Firmware		
		BC9020	BC9050	BC9120
FB_SNTP	30.05.06	B0	B0	B1

IP-TCP/IP-UDP	Version	Firmware		
		BC9020	BC9050	BC9120
FB_AddMultiRoute	26.09.06	-	-	-
FB_DelMultiRoute	26.09.06	-	-	-
FB_IpClose	26.09.06	B0	B0	B1
FB_IpEndSession	26.09.06	B0	B0	B1
FB_IpOpen	26.09.06	B0	B0	B1
FB_IpReceive	26.09.06	B0	B0	B1
FB_IpSend	26.09.06	B0	B0	B1
FB_IpStartSession	26.09.06	B0	B0	B1

ModbusTCP	Version	Firmware		
		BC9020	BC9050	BC9120
FB_MBClose	26.09.06	B0	B0	B1
FB_MBConnect	26.09.06	B0	B0	B1
FB_MBGenericReq	26.09.06	B0	B0	B1

5.13.4.2 Socket Interface

5.13.4.2.1 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 sending 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

5.13.4.2.2 IP block overview

IP-TCP/IP-UDP	Firmware	Description
FB_IpStartSession [▶ 87]	1.14	Opening a session
FB_IpEndSession [▶ 88]	1.14	Closing a session
FB_IpOpen [▶ 89]	1.14	Opening a TCP/IP connection (not required for UDP communication)
FB_IpClose [▶ 90]	1.14	Closing a TCP/IP connection (not required for UDP communication)
FB_IpReceive [▶ 91]	1.14	Receiving TCP or UDP telegrams
FB_IpSend [▶ 92]	1.14	Sending TCP or UDP telegrams

Multicast function blocks	Firmware	Description
FB_AddMultiRoute [▶ 104]	1.14	Creating a muticast address
FB_DelMultiRoute [▶ 104]	1.14	Deleting a muticast address

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).

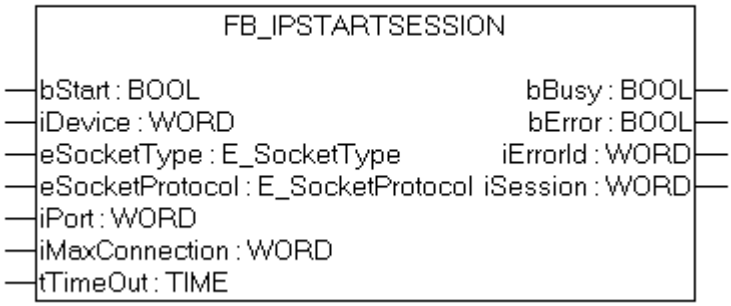


Fig. 68: Function block FB_IPSTARTSESSION

INPUT

```
VAR_INPUT
   bStart           : BOOL;
   iDevice          : WORD;
   eSocketType      : E_SocketType;
   eSocketProtocol  : E_SocketProtocol;
   iPort            : WORD;
   iMayConnection   : 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 detect 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.

FB_IpEndSession

The function block closes an open session. A positive edge of *bStart* closes the session and releases resources.

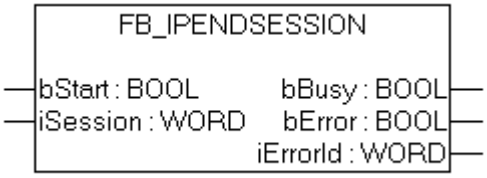


Fig. 69: Function block FB_IPENDSESSION

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).

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 [▶ 87]). 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 [▶ 92]). **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.

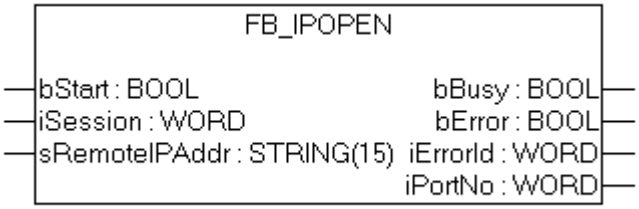


Fig. 70: Function block FB_IPOPEN

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).

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.

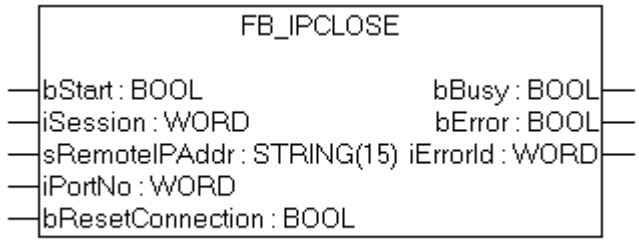


Fig. 71: Function block FB_IPCLOSE

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).

FB_IpReceive

The function block *FB_IpReceice* enables receiving of UDP or TCP telegrams. Which of the two connections is used is specified in FB_IpStartSession [▶ 87].

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 [▶ 92].

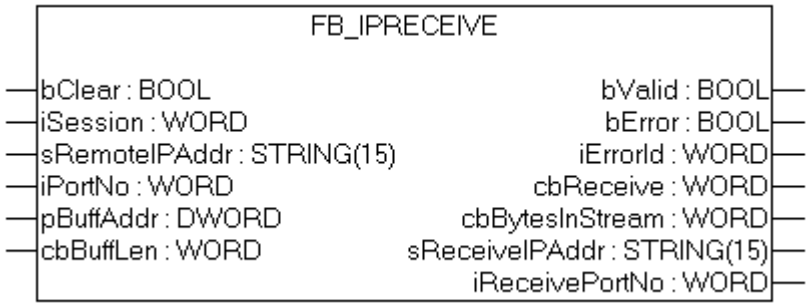


Fig. 72: Function block FB_IPRECEIVE

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 [▶ 87].

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**.

FB_IpSend

The function block sends data via TCP or UDP. Which of the connections is used is specified in FB_IpStartSession [▶ 87]. 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 [▶ 89] 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.

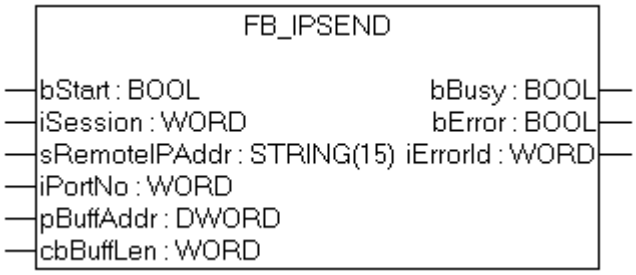


Fig. 73: Function block FB_IPSEND

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 [▶ 87].

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 [▶ 89] 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).

5.13.4.2.3 TCP/IP

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 (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207366411.zip)

Download TwinCAT project as TCP/IP client prx file (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207368587.zip)

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 (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207370763.zip)

Download TwinCAT project as TCP/IP server prx file (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207372939.zip)

5.13.4.2.4 UDP/IP

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.

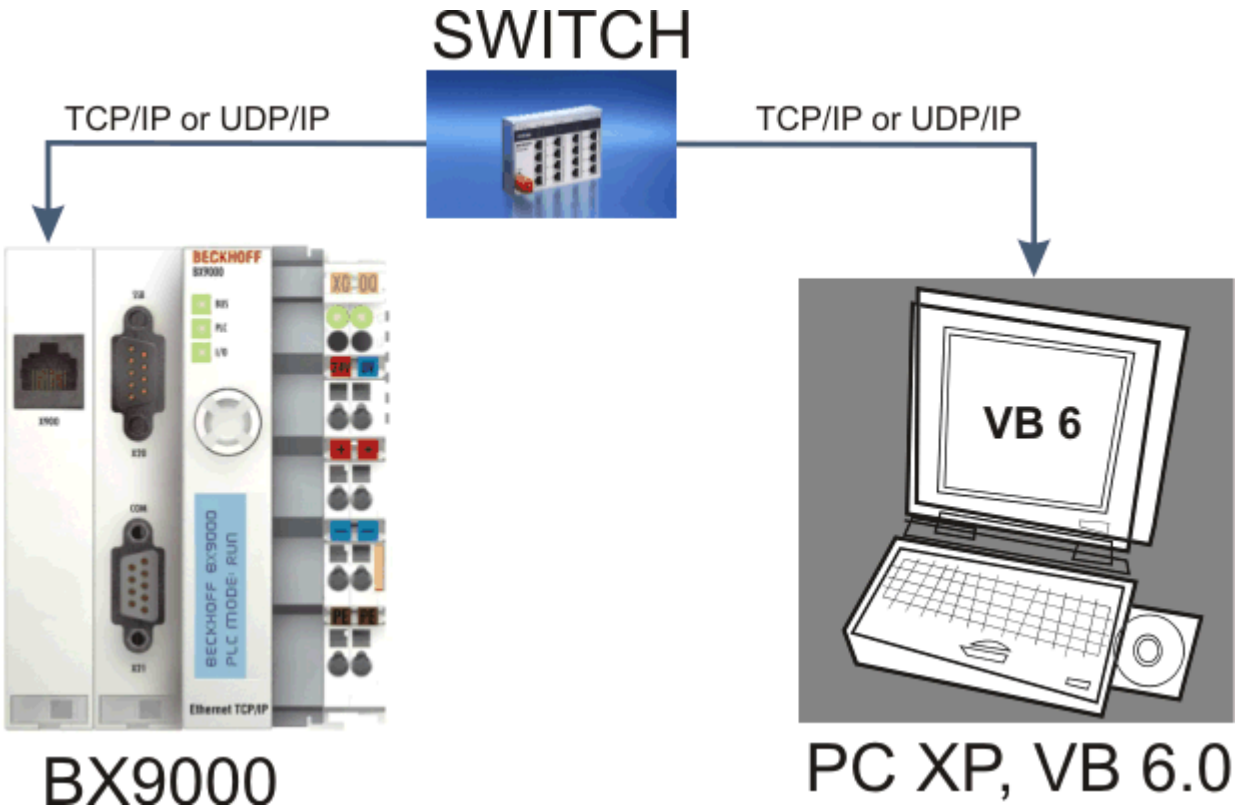


Fig. 74: UDP/IP connection

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

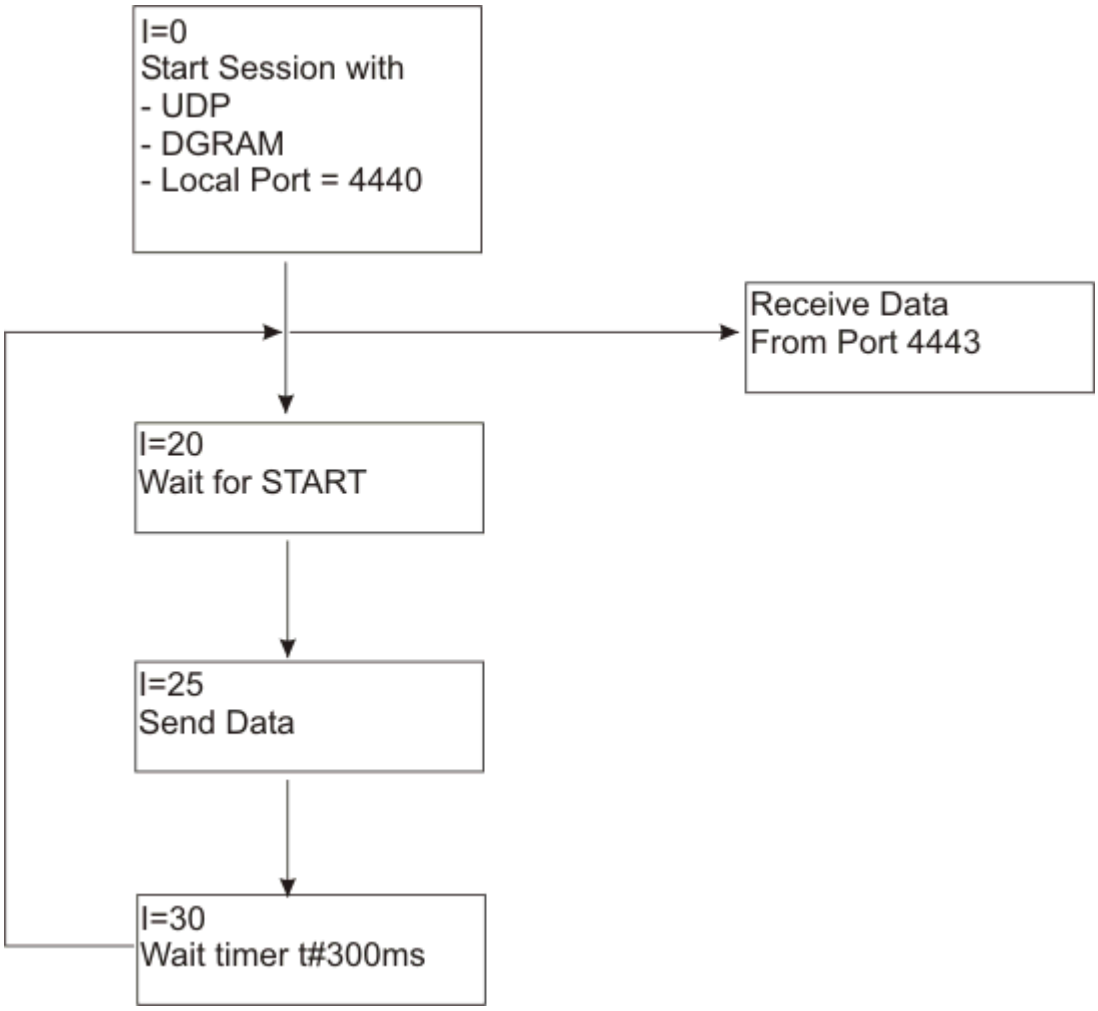


Fig. 75: Sample program

Example

Download VB6 program as TCP/IP server zip file (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207375115.zip)

Download BX9000 TwinCAT project as TCP/IP client, prx file (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207377291.zip)

5.13.4.3 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_MBGernericReq** and **FB_MBClose**.

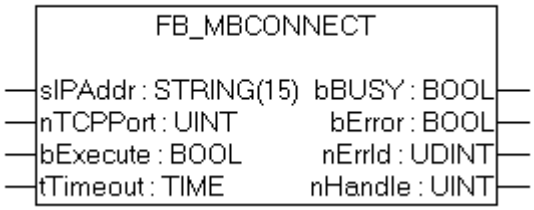


Fig. 76: Function block FB_MBCONNECT

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*dec*.

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 [▶ 102]).

nHandle: Handle that must be connected with function blocks FB_MBGernericReq 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 **cdResLen**. If **cbResponse** is greater than **dResLen** data are lost and a larger buffer should be chosen.

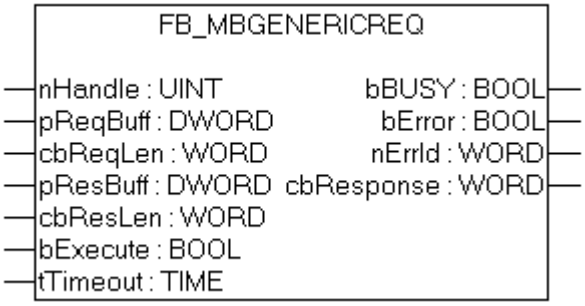


Fig. 77: Function block FB_MBGENERICREQ

INPUT

```
VAR_INPUT
   nHandle       : UINT;
   pRecBuff      : 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 [▶ 102]).

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.

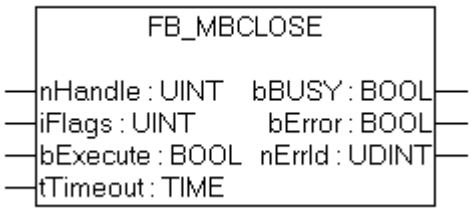


Fig. 78: Function block FB_MBCLOSE

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 [▶ 102]).

Example

For the example you need two BX9000 or a BC9x00 instead of a BX9000.

Download first BX9000 as ModbusTCP client (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207379467.zip)

Download second BX9000 as ModbusTCP server (or BC9x00) (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207381643.zip)

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=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

5.13.4.4 SMTP

5.13.4.4.1 FB_Smtp

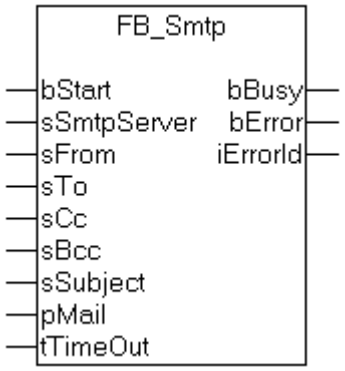


Fig. 79: Function block FB_Smtp

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
```

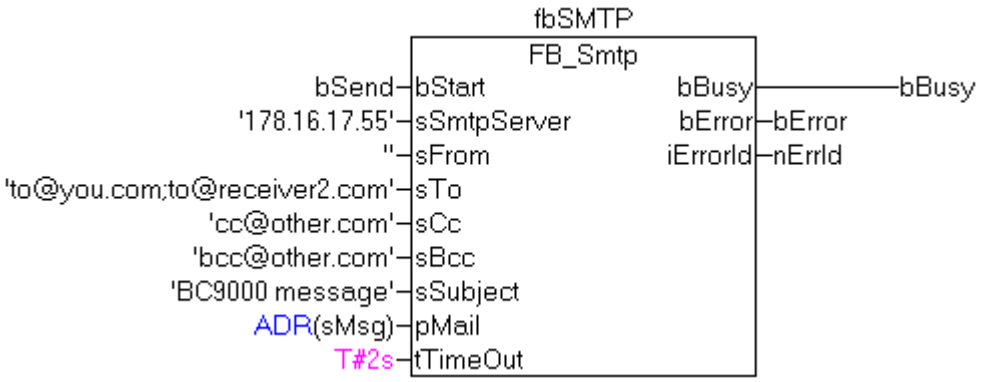


Fig. 80: Function block fbSMTP

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

5.13.4.5 SNTP

5.13.4.5.1 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.

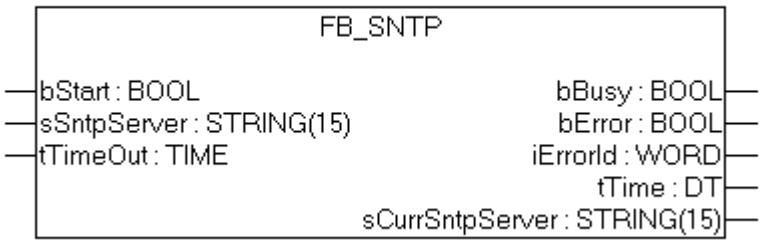


Fig. 81: Function block FB_SNTP

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.
sCurrSnptServer: IP address of the SMTP server.

Return parameter iErrId	Meaning
0	No error
<> 0	Error number [▶ 110]

Download (https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3207383819.zip)

5.13.4.6 Services

5.13.4.6.1 FB_AddDnsServer

Fig. 82: 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 [▶ 110]).

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

5.13.4.6.2 FB_GetHostByAddr

Fig. 83: 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 [▶ 110]).

Development environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

5.13.4.6.3 FB_GetHostByName

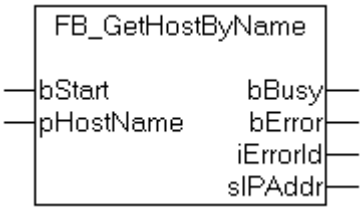


Fig. 84: Function block FB_GetHostByName

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 [▶ 110]).

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

5.13.4.6.4 FB_GetNetworkConfig

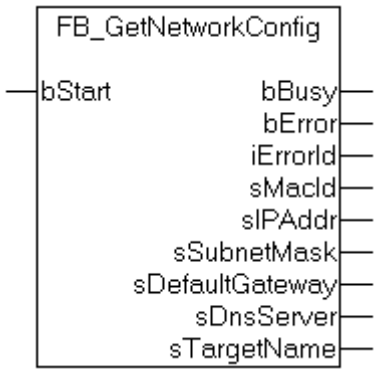


Fig. 85: Function block FB_GetNetworkConfig

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 [▶ 110]).

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

5.13.4.6.5 FB_SetTargetName

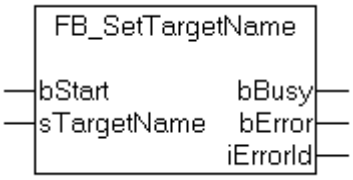


Fig. 86: Function block FB_SetTargetName

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 environment	Target platform	PLC libraries to be linked
TwinCAT v2.10.0 and above	BX9000 (165) firmware version >=1.12	TcBaseBX9000.lbx

5.13.4.6.6 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	**ERR_SNTP_CONN_SHUT_DOWN**
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

5.14 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

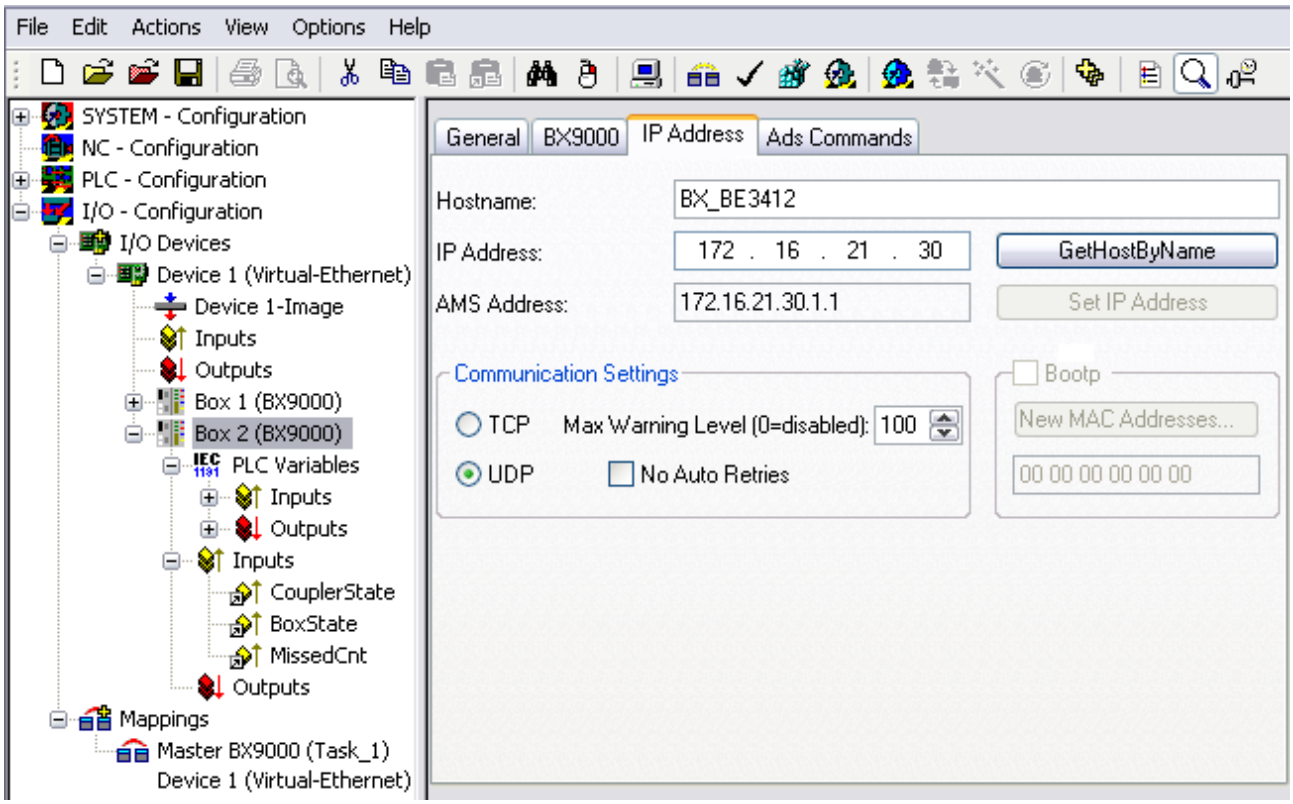


Fig. 87: IP address of the BX9000 in the TwinCAT System Manager

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 [▶ 112]of the PC and the programming interface of the Bus Terminal Controller

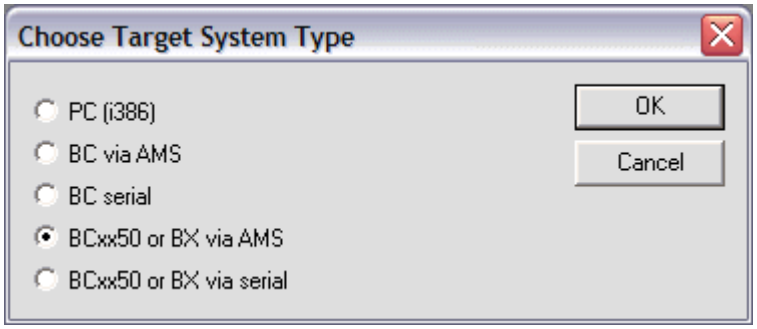


Fig. 88: Selecting the data transfer route - AMS

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.

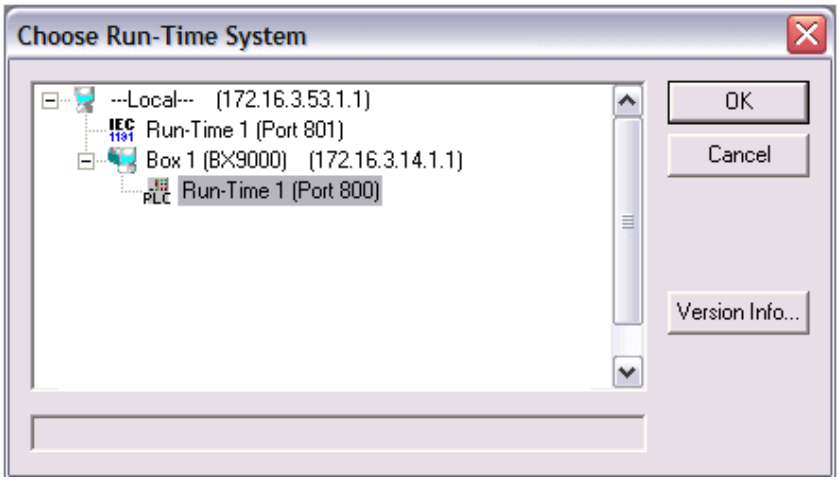


Fig. 89: Choose Target System

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.

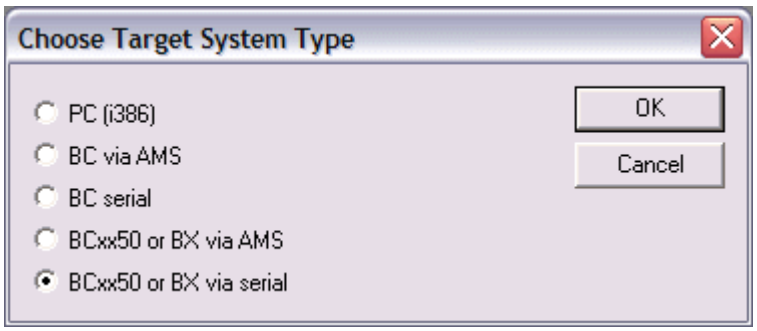


Fig. 90: *Selecting the data transfer route - serial interface*

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

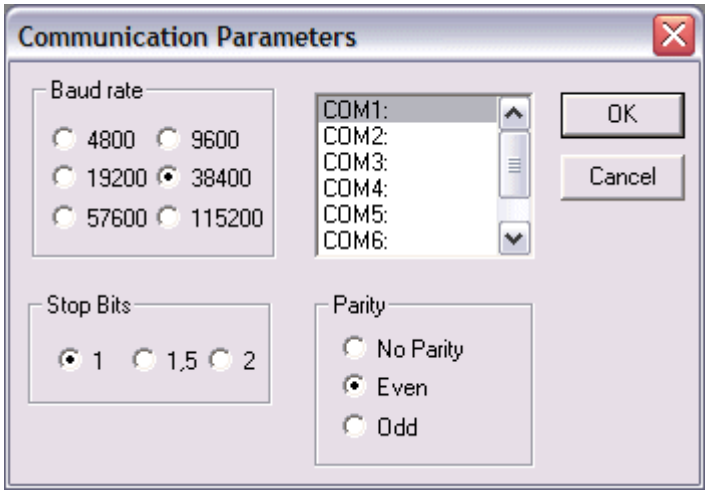


Fig. 91: Parameterization of the serial interface

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 [▶ 52]).

Select the ADS connection in TwinCAT PLC Control.

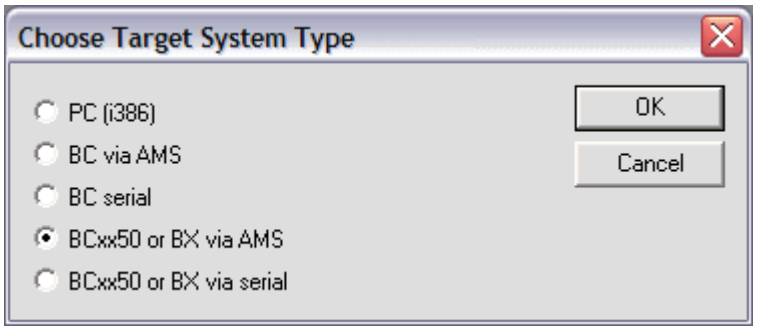


Fig. 92: *Selecting the data transfer route - AMS*

PLC Control can be accessed via *Online/Communication Parameters....*

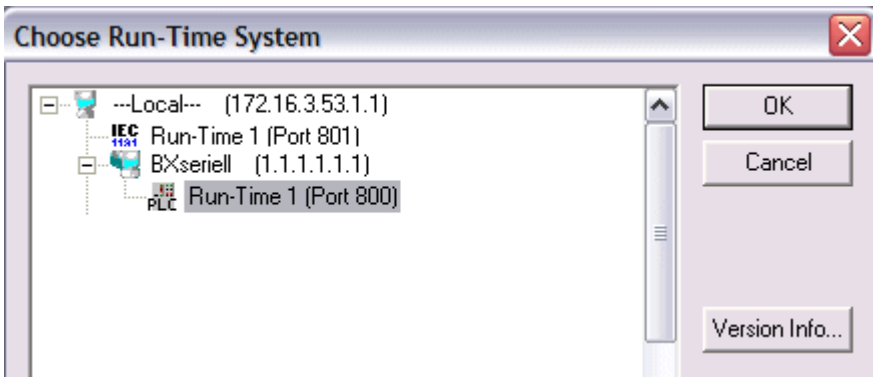


Fig. 93: Selecting the device

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.

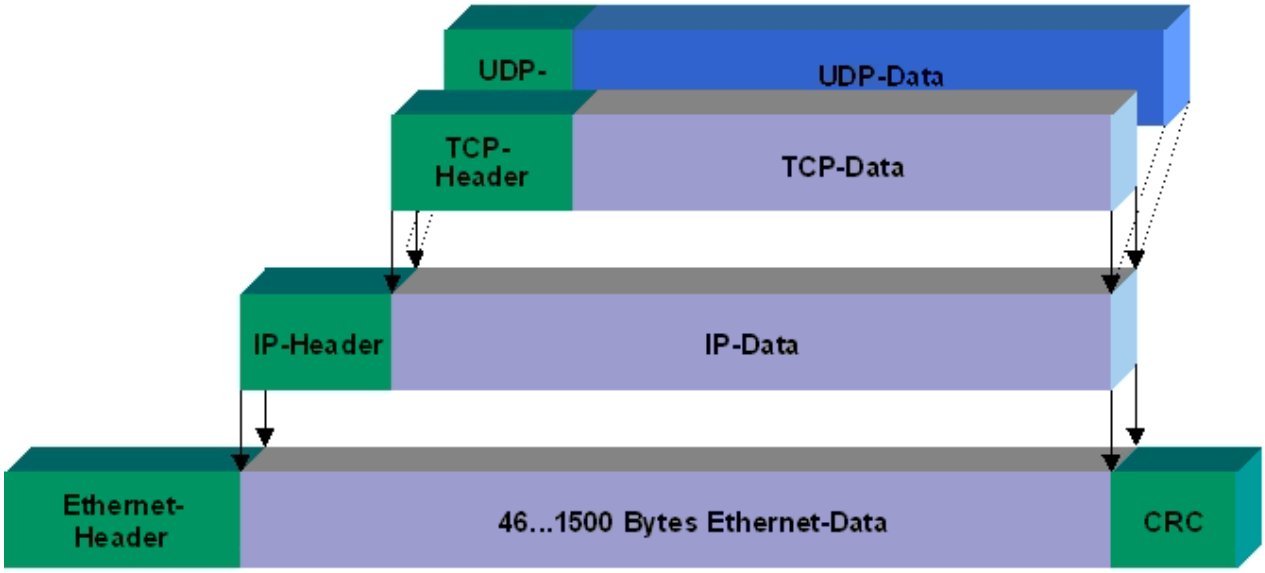


Fig. 94: User Datagram Protocol (UDP)

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.

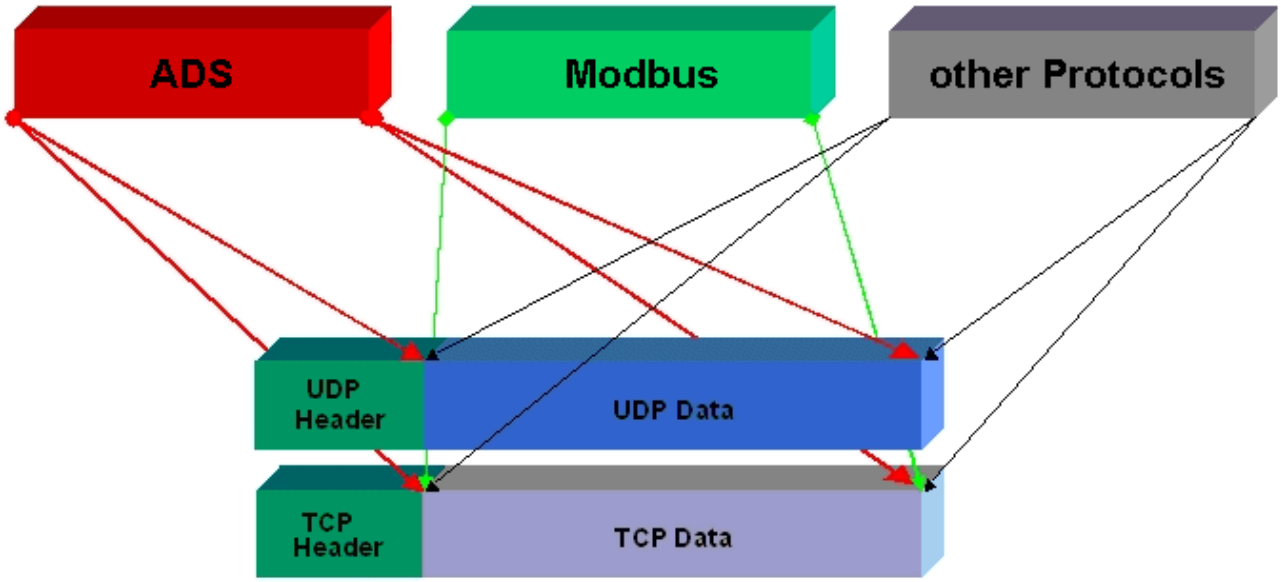


Fig. 95: Protocols running on top of TCP/IP and UDP/IP

ADS can be used on top of either TCP or UDP, but ModbusTCP is always based on TCP/IP.

6.2 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.

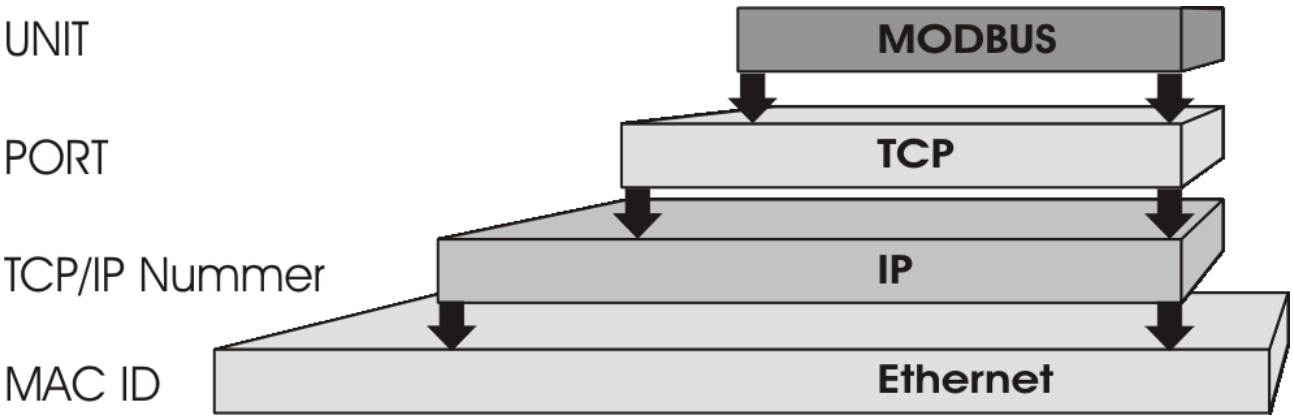


Fig. 96: ModbusTCP Protocol

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_{bin}	Fieldbus error, watchdog time elapsed
14...2	-	-	reserved
1	CNF	1_{bin}	Bus Coupler configuration error
0	KB	1_{bin}	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 [▶ 118].

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
Date 1 byte 2	255
Date 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.

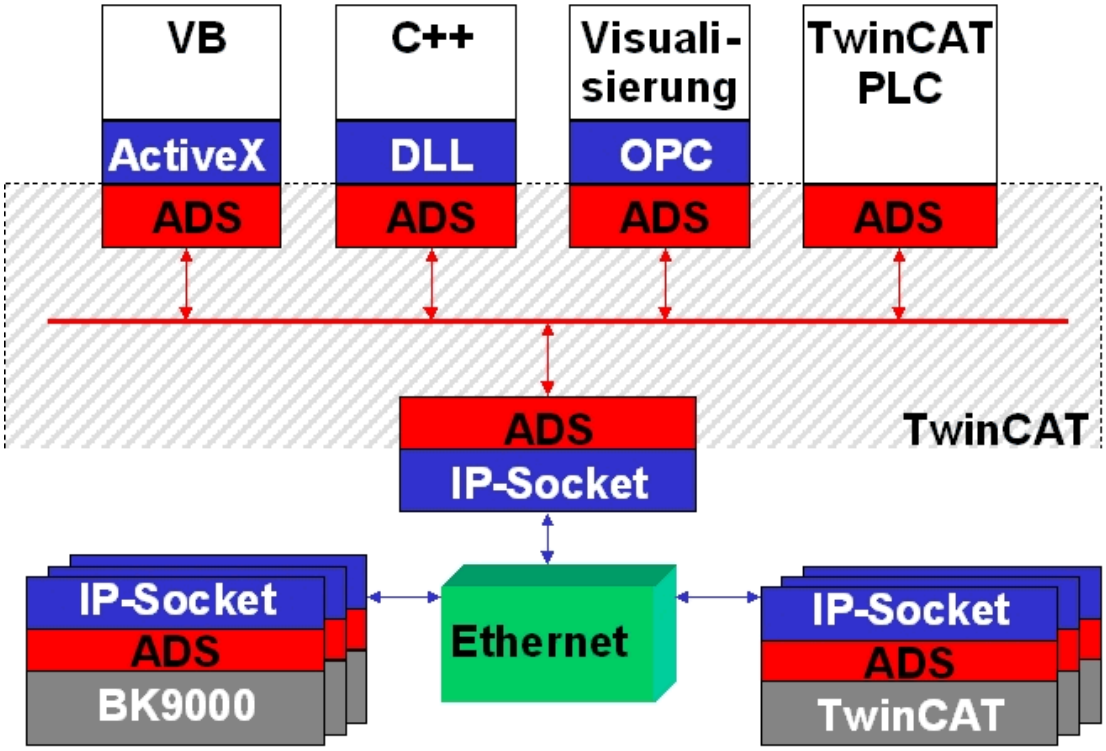


Fig. 97: The ADS protocol as a transport layer within the TwinCAT system

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.

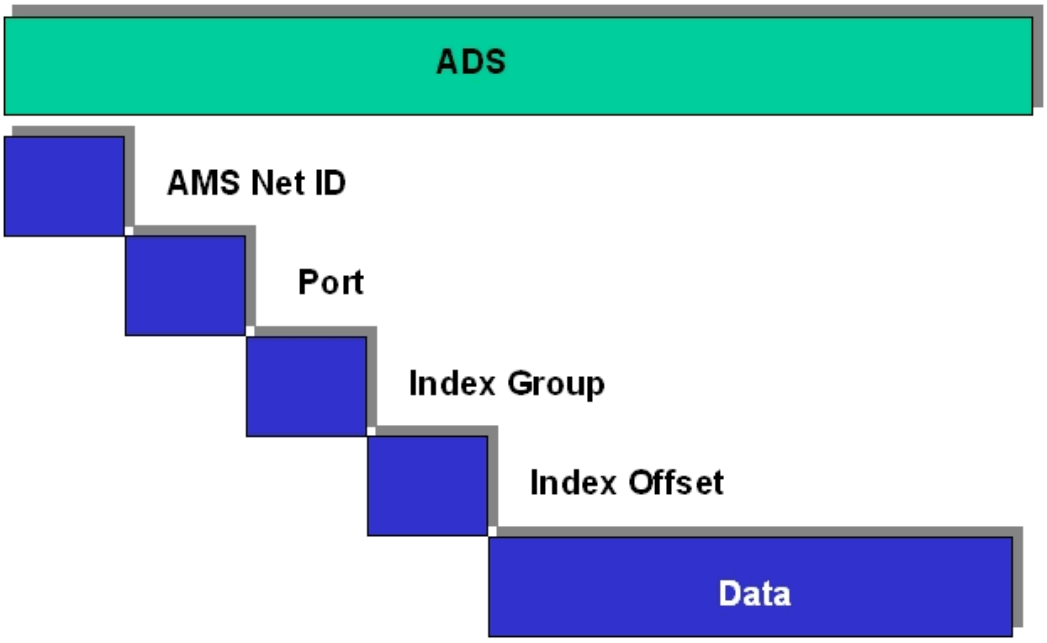


Fig. 98: Structure of the ADS protocol

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

User data of an ADS telegram

The maximum size of the user data in an ADS telegram is 1900 bytes.

Process data port 300

The fieldbus data is accessed via ADS port number 300. The data is monitored by a watchdog. If no further telegram arrives within 1000 ms the outputs will be switched to the safe state.

Index group	Meaning	Index offset (value range)
0xF020	Inputs	0...511
0xF030	Outputs	0...511

Local process image port 800 (BC9000, C900 only)

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...511
0xF030	Outputs	0...511
0x4020	Flags (BC 9000, C900 only)	0...4096

ADS services

AdsServerAdsStateB7

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

AdsServerDeviceStateB7

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

AdsServerTypeB7

Data type (read only)	Meaning
String	Coupler_PLC

ADSWriteControlBA

Data type (write only)	Meaning
NetID	Net ID of the BC9000,C900
Port	800
ADSSTATE	5 - RUN / 6 - STOP
DEVSTATE	0
LEN	0
SRCADDR	0
WRITE	rising edge starts the block
TMOUT	example: t#1000 ms

Acyclic data communication

With acyclic data communication it is important to note that about 20 to 30 ms are required in order to establish a TCP/IP connection. Following the successful establishment of a connection the ADS data are sent or read. An ADS read of 1000 bytes takes about 50 ms. If no data are transmitted for 10 seconds, the TCP/IP connection is disconnected from the BC/BK9000, B/C900.

Register port 100

The ADS port number in the BK/BC9000, B/C900 for register communication is fixed, being set at 100.

Index group	Index offset (value range)		Meaning
	Hi-Word	Lo-Word	
0	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...64	0...3	1...64	Register of the Bus Terminals Hi-Word, channel number Lo-Word, register number of the Bus Terminal

Timeout of the ADS function block

When reading the register, the time out of the ADS block has to be set to a time longer than 1 second.

Setting the password

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

Access control and IP - AMS-Net ID assignment[B6]

The *AMS Net-Id* table permits access control to the BC 9000, C900 via AMS. As soon as this table has entries, only those AMS devices that have been entered will be able to access the BC 9000.
Furthermore, an assignment of the AMS-Net ID to the IP address of the node is explicitly done here.
The *AMS Net-Id* table can be filled with ADS write commands:
a maximum of 10 entries is possible.

The structure

AMS Net ID	Size
AMS Net ID	6 byte
IP address	4 byte
Reserve	2 byte
Reserve	4 byte
Reserve	4 byte

Access takes place via port number: 10,000

Index group: 700

Index Offset (Write)	Comment	Data
0	Add an entry	Data structure, 20 bytes
1	Delete an entry	-
2	Delete all entries	-
10	Save the table in Flash memory	-

Index Offset (Read)	Comment	Data
0	Number of entries	2 byte
1..10	Entry n (1...10)	Data structure, 20 bytes

First entry

The first entry must be the device that his writing into the table, because the settings have immediate effect. Make sure that all the settings are correct. The table can also be deleted if the end terminal only is inserted, and the DIP switches 1 to 7 are set to ON.

* as from firmware B6
** as from firmware B7
)² as from firmware BA

See the example.

7 Error handling and diagnosis

7.1 Diagnostics

Ethernet State

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.

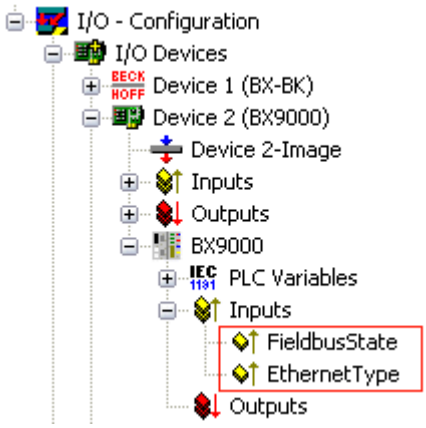


Fig. 99: Ethernet State

Fieldbus State

Error number	Description	Remedy
0	No error	-
1	Watchdog error	Communication interrupted

Ethernet Type

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).

Diagnosis number	Description	Remedy
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

Reading fieldbus state by ADS

The fieldbus status can be read through ADSREAD in the default configuration or in the TwinCAT configuration (under development).

Parameter ADSREAD function block	Description
NetID	local – empty string
Port	
IndexGroup	
IndexOffset	
LEN	

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.

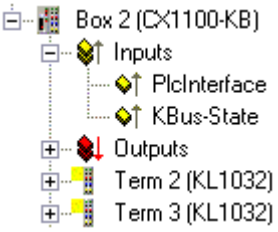


Fig. 100: State of the K-bus

Error bit	Description	Error type
0	No error	No ERROR.
Bit 0	K-bus error	ERROR
Bit 2	K-Bus is re-triggered	NOTE

Reading K-bus state by ADS

In default or TwinCAT configuration the fieldbus state 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 Terminal Controllers have status LEDs.

BC9050

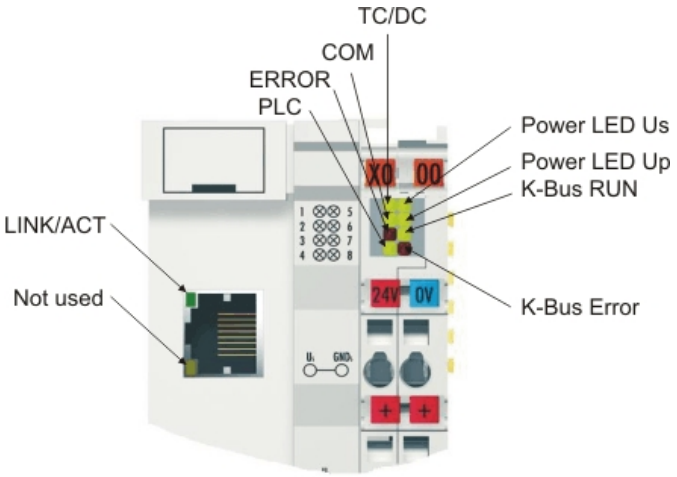


Fig. 101: BC9050 LEDs

LEDs for power supply diagnostics

LED (Power LEDs)	Meaning
Power LED Us off	Bus Coupler has no voltage 24 V_{DC}
Power LED Up off	No power supply 24 V_{DC} connected at the power contacts

LEDs for K-Bus diagnostics

LED (K-bus)	Meaning
K-bus RUN	on or flashing - K-bus running
K-bus ERR	flashing (see error code) [▶ 133]

LEDs for Ethernet diagnosis

LED (Ethernet)	Meaning
LINK/ACT	on - LINK available, flashing - LINK available and communication
ERROR	flashing - DHCP or BootP active. Waiting for an IP address
COM	Communication with controller available

LEDs for PLC diagnostics

LED (Ethernet)	Meaning
PLC	on - PLC running, flashing - cycle time is exceeded, off - cycle time exceeded permanently or PLC stopped
TC/DC	on - TwinCAT configuration active, off - Default configuration active, flashing TwinCAT configuration faulty

BC9120

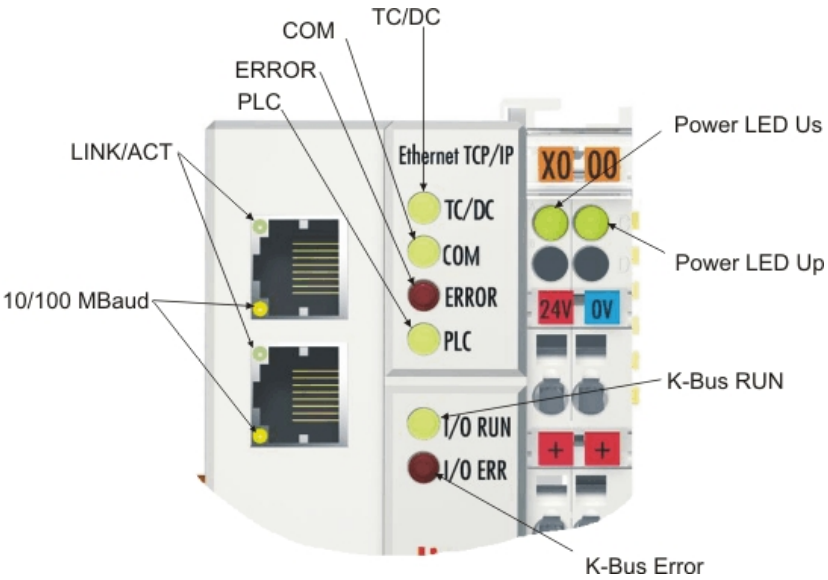


Fig. 102: BC9120 LEDs

LEDs for power supply diagnostics

LED (Power LEDs)	Meaning
Power LED Us off	Bus Coupler has no voltage 24 V_{DC}
Power LED Up off	No power supply 24 V_{DC} connected at the power contacts

LEDs for K-Bus diagnostics

LED (K-bus)	Meaning
K-bus RUN	on or flashing - K-bus running
K-bus ERR	flashing (see error code [▶ 133])

LEDs for Ethernet diagnosis

LED (Ethernet)	Meaning
LINK/ACT	on - LINK available, flashing - LINK available and communication
10/100 Mbaud	on - 100 Mbaud, off - 10 Mbaud
ERROR	flashing - DHCP or BootP active. Waiting for an IP address
COM	Communication with controller available

LEDs for PLC diagnostics

LED (Ethernet)	Meaning
PLC	on - PLC running, flashing - cycle time is exceeded, off - cycle time exceeded permanently or PLC stopped
TC/DC	on - TwinCAT configuration active, off - Default configuration active, flashing TwinCAT configuration faulty

Error codes for K-Bus diagnosis

Error code	Error argument	Description	Remedy
-	flashing continuously	EMC problems	• Check power supply for undervoltage or overvoltage peaks • Implement EMC measures • If a K-bus error is present, it can be localized by a restart of the coupler (by switching it off and then on again)
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 Bus Coupler
2	-	Reserve	-
3	0	K-bus command error	• No Bus Terminal inserted • 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.
4	0	K-bus data error, break behind the Bus Coupler	Check whether the n+1 Bus Terminal is correctly connected; replace if necessary.
	n	Break behind Bus Terminal n	Check whether the KL9010 Bus End Terminal is connected
5	n	K-bus error in register communication with Bus Terminal n	Exchange the nth Bus Terminal
6	0	Error at initialization	Exchange Bus Coupler
	1	Internal data error	Perform a hardware reset on the Bus Coupler (switch off and on again)
	2	DIP switch changed after a software reset	Perform a hardware reset on the Bus Coupler (switch off and on again)
	3	IP address already assigned	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 Bus Coupler again. Remedy: increase the cycle time
9	0	Checksum error in Flash program	Transmit program to the BC 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 Bus Coupler again, and if the error occurs again then exchange the Bus Terminal
15	n	Number of Bus Terminals is no longer correct	Start the Bus Coupler again. If the error occurs again, restore the manufacturers setting using the KS2000 configuration software
16	n	Length of the K-bus data is no longer correct	Start the Bus Coupler again. If the error occurs again, restore the manufacturers setting using the KS2000 configuration software

8 Appendix

8.1 BC9xx0 - First steps

Necessary components

- A BC9xx0
- KL1114
- KL2134
- KL9010
- A 24 V_{DC} power supply unit plus cabling material
- An Ethernet cross cable for a 1:1 connection between PC and BC9020 or BC9050 (without switch)
- A "normal" Ethernet cable between PC and BC9120

Setting the DIP switches

Set DIP switch 1 to ON, leave all other DIP switches OFF. A fixed IP address is used. Restart the BC after setting the DIP switches.
BC9050 then has the IP address 172.16.21.1
BC9020 then has the IP address 172.16.22.1
BC9120 then has the IP address 172.16.23.1

Set the IP address of your programming PC accordingly. For example: 172.16.100.100 (SubNetMask 255.255.0.0).

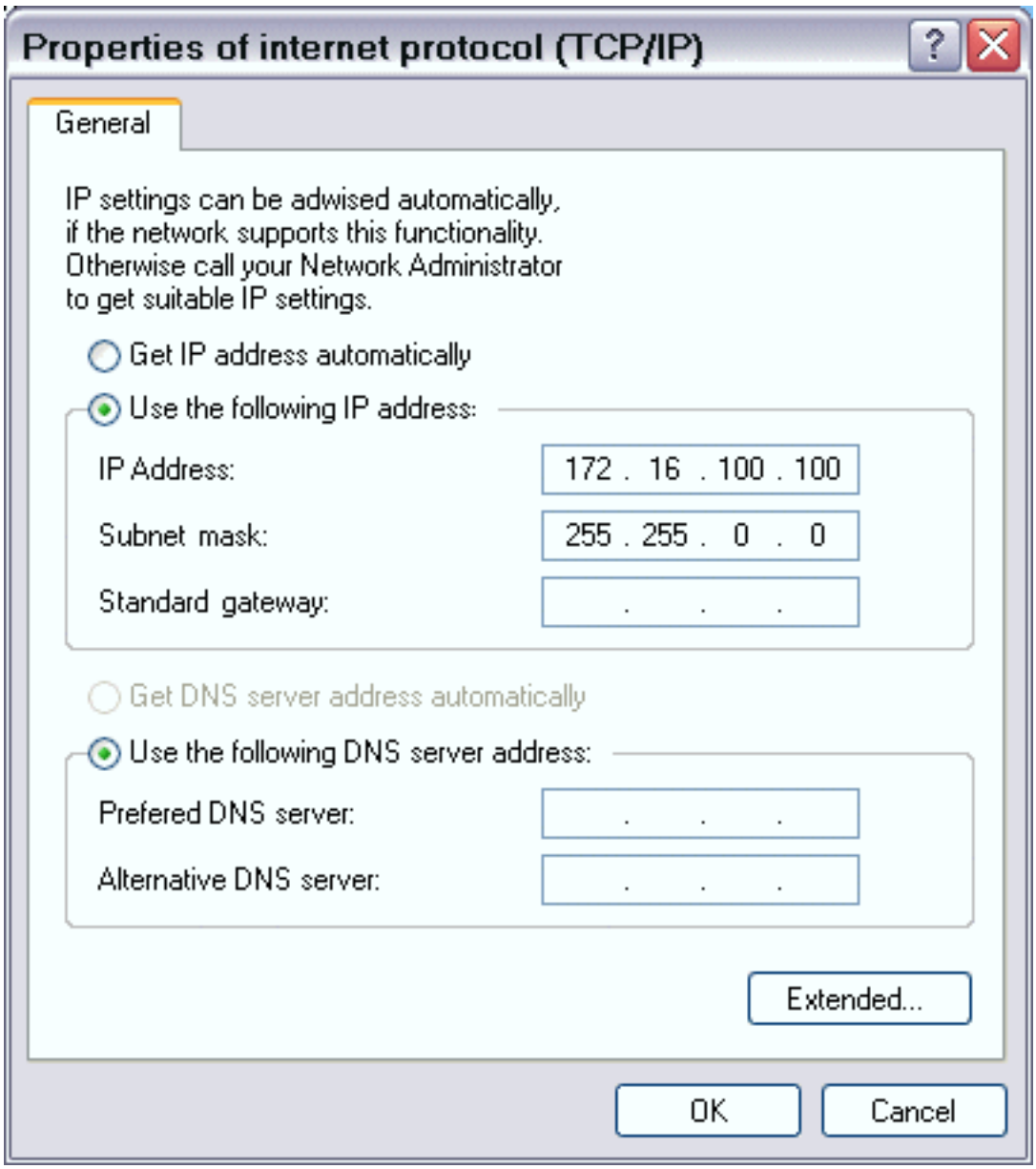


Fig. 103: Setting the IP address of the programming PC

Use the >Ping< command to check whether TCP/IP communication is available. Open the prompt (DOS box) and enter the following, e.g. if a BC9050 is used: Ping 172.16.21.1. If the ping is successful you can continue. If the ping is not answered the communication is faulty. Check the Ethernet connection - right cable, the Controller DIP switch is switched on correctly, the IP address on the PC has changed.

Search BC via the System Manager

Start the System Manager. Select a new target system.

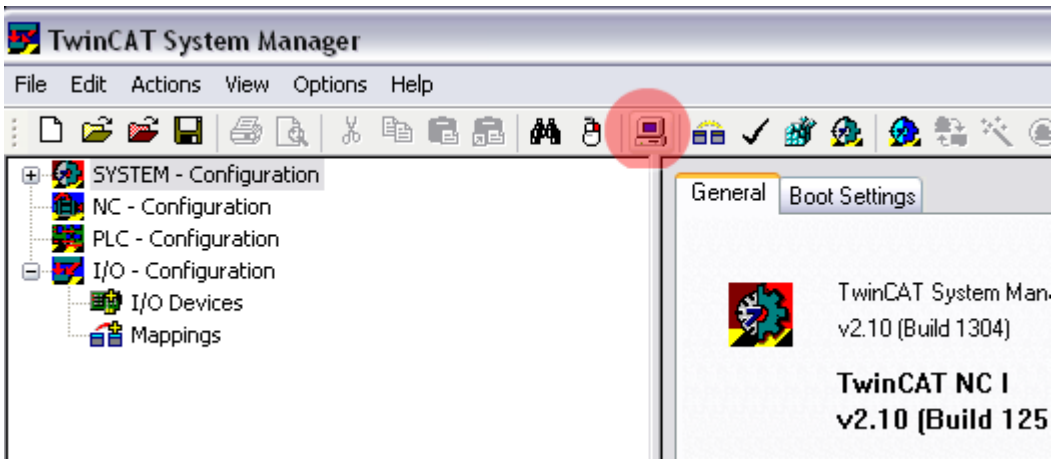


Fig. 104: Search BC via the System Manager

Click *Search (Ethernet)*...

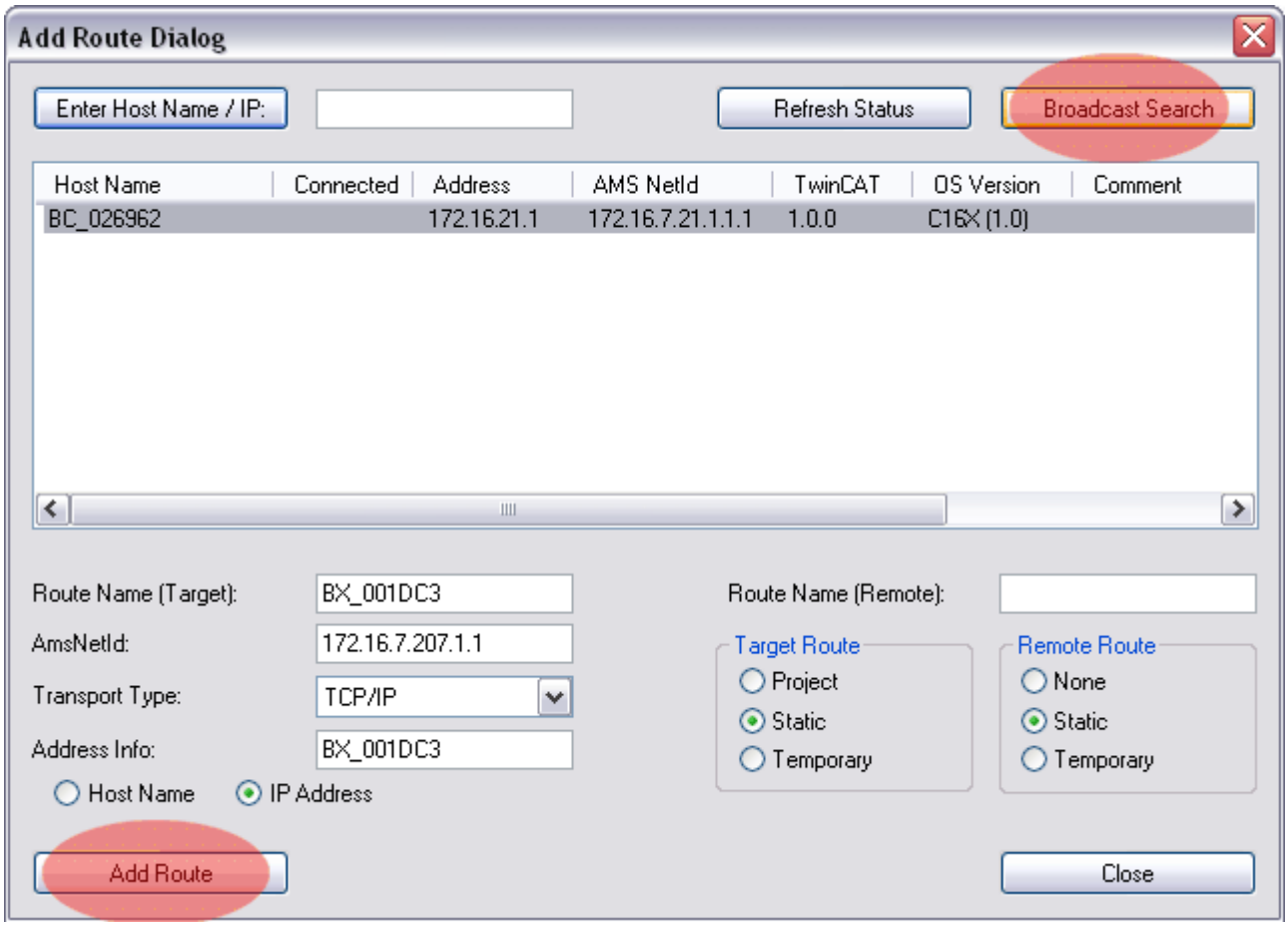


Fig. 105: System Manager - broadcast search

Then click **Broadcast Search**. The BC now appears in the window. The host name is formed based on BC plus the last 3 bytes of the MAC ID and is therefore unique in your system. The MAC ID is printed on a label below the Bus Terminal Controller. Set the options button to *IP Address* and click *Add Route*. The Connected field should show a cross or "X". If a password query appears confirm with OK (leave blank). The password function is not implemented in the Bus Terminal Controller.

You can now select the Bus Terminal Controller. Click on the BC and confirm with OK.

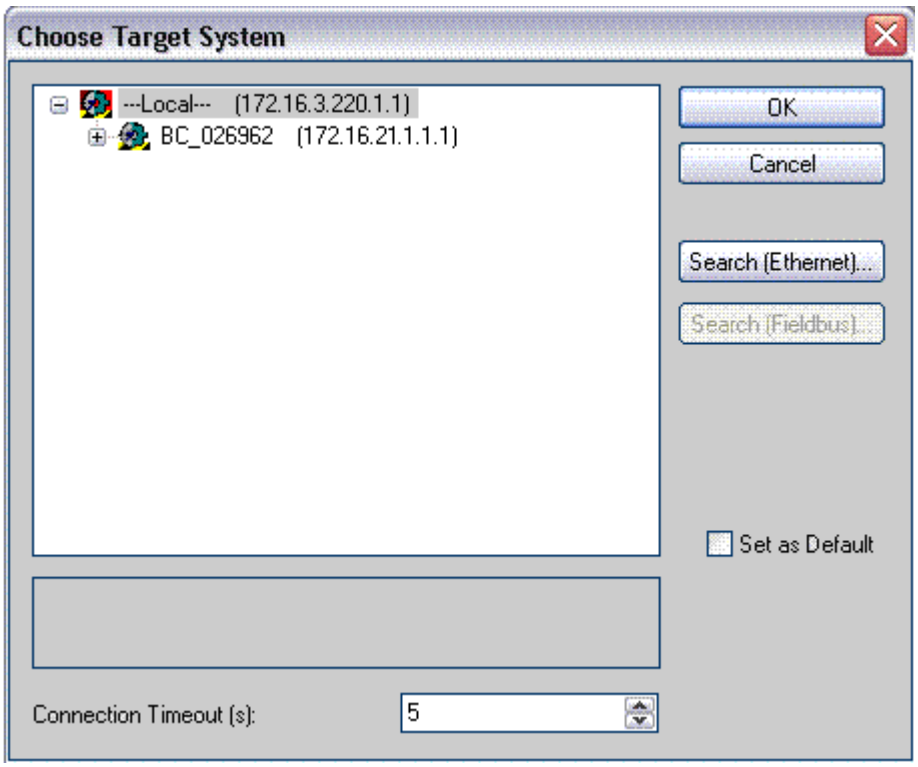


Fig. 106: Choose Target System

Your BC with name and IP address is shown in red at bottom left in the System Manager. The field next to it should be blue and show Config Mode. If this is the case you can now scan the device. Right-click on *IO Devices* then **Scan Devices**.

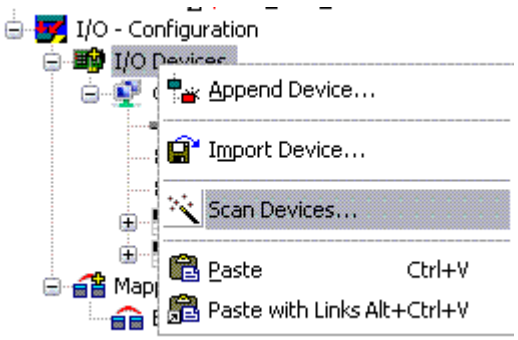


Fig. 107: System Manager - scan devices

The System Manager should find 2 devices. The Ethernet interface and the K-bus interface with the connected Bus Terminals.

Ethernet interface

Click on **Upload** to upload the current IP configuration.

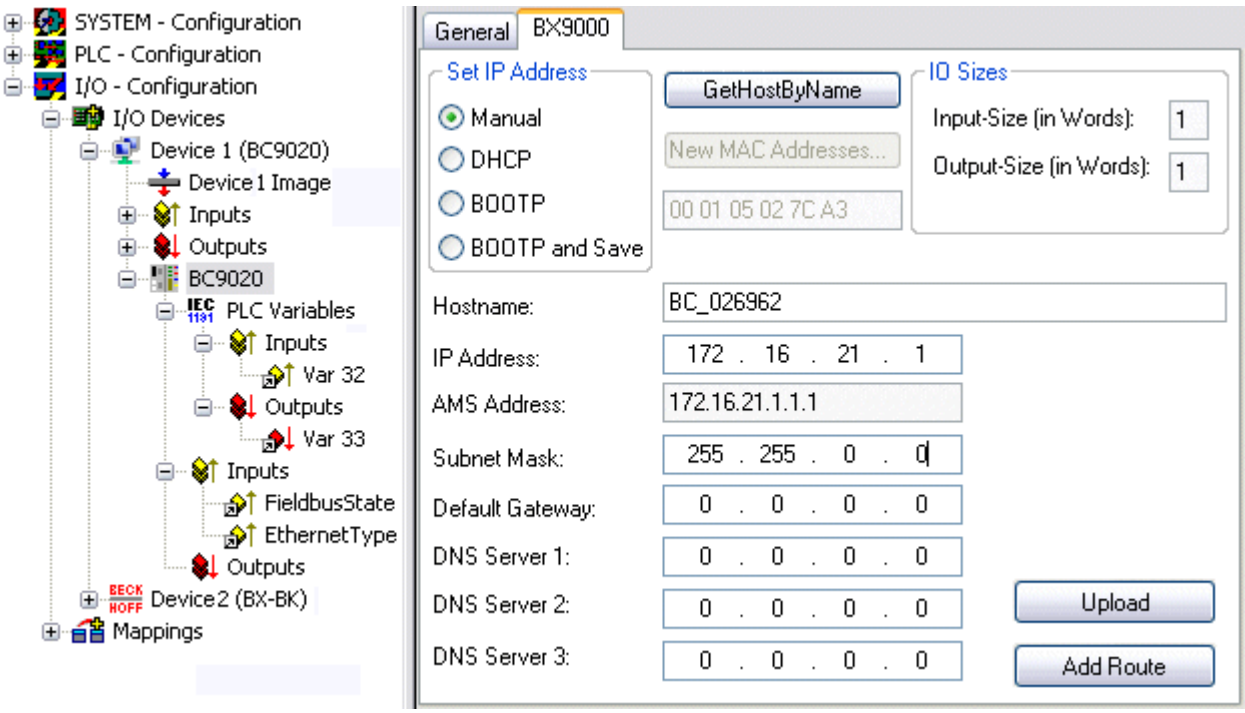


Fig. 108: Upload the current IP configuration

PLC configuration

Append your PLC program here. First open the following example:

(https://infosys.beckhoff.com/content/1033/bc9xx0/Resources/zip/3740903947.zip) Save the file locally on your computer.

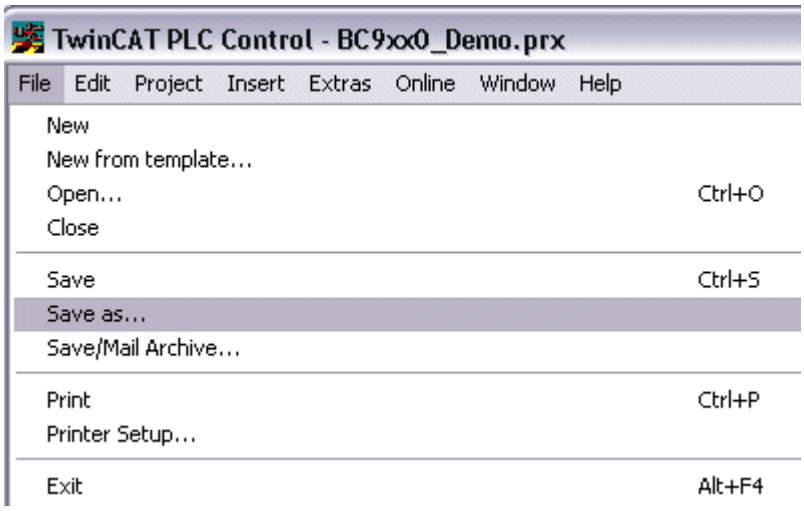


Fig. 109: Saving the file locally on your computer

Recompile the file.

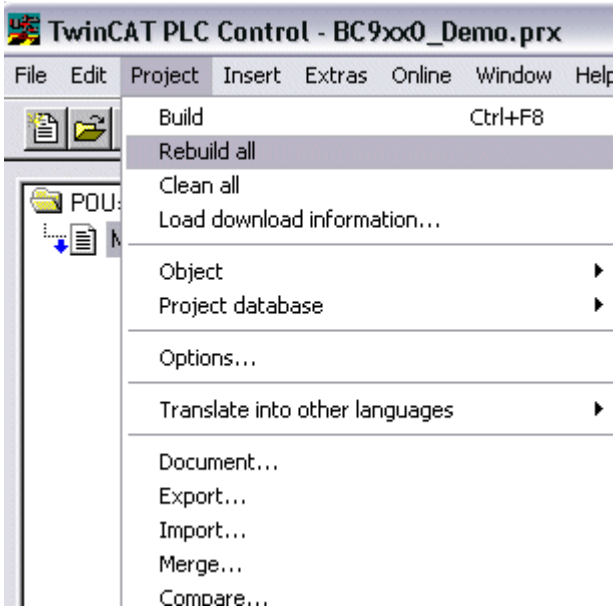


Fig. 110: Compiling the file

Re-save.

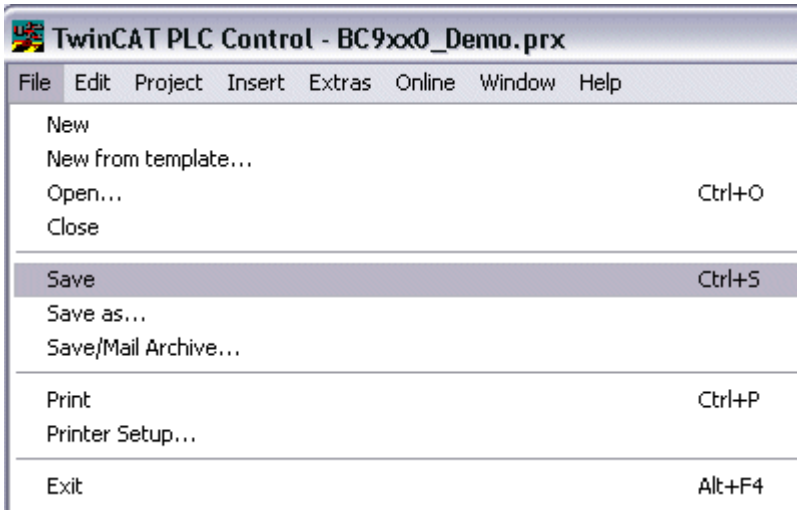


Fig. 111: Saving the file again

Change back to the System Manager and append your PLC program (right-click on PLC Configuration).

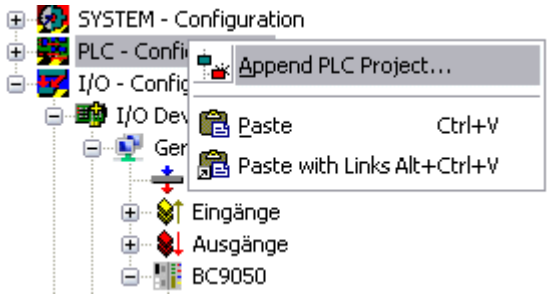


Fig. 112: Adding the PLC program

Search for your file (file type: tpy). Then open the tree. You will see all allocated data. Select the variables and link them to your Bus Terminals.
bIN_1 with KL1104 channel 1
bOUT_1 with KL2134 channel 1
bOUT_2 with KL2134 channel 2
bOUT_3 with KL2134 channel 3

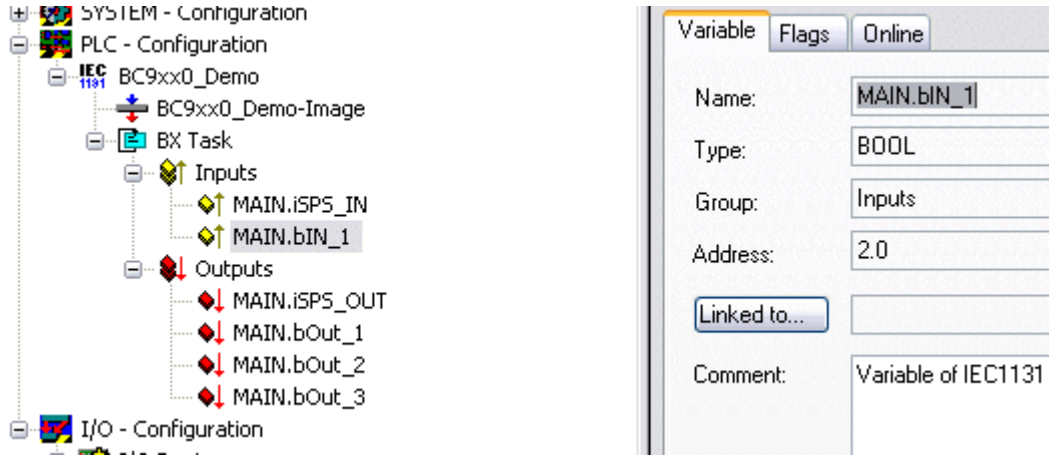


Fig. 113: Linking the allocated data to the Bus Terminals

After being linked you should now see the following.

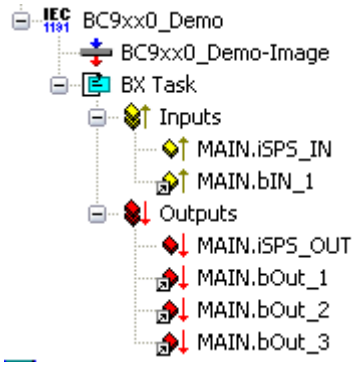


Fig. 114: Displaying the linked allocated data

Now activate the configuration in the System Manager. Click on to download the configuration to the Bus Terminal Controller. Acknowledge all messages with OK. Finally you will be asked to restart the system. The BC will then reboot. The System Manager will lose the connection with the Bus Terminal Controller. Wait until the System Manager has re-established the connection. After a few seconds the status field in the bottom left corner of the System Manager should be green

. If it is blue the BC has restarted in Config mode. Check your configuration.

Switch back to PLC Control. Go online and select the target system.

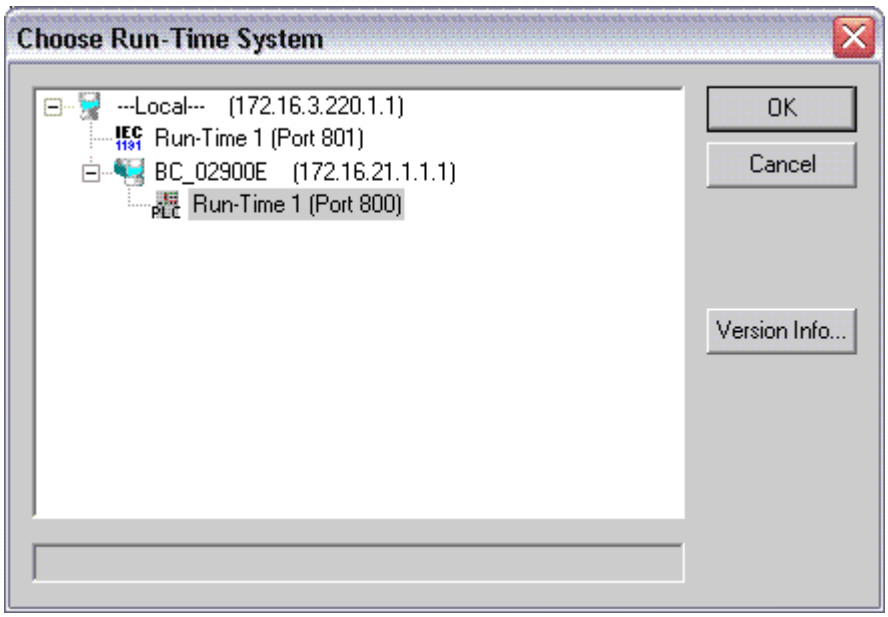


Fig. 115: Choose Target System

Log in and start the program.

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 [▶ 67]. 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 [▶ 67]. 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 [▶ 67]).

8.3 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
Installation position	variable
Vibration resistance	conforms to EN 60068-2-6
Shock resistance	conforms to EN 60068-2-27
EMC immunity	conforms to EN 61000-6-2
Emission	conforms to EN 61000-6-4

Transport and storage

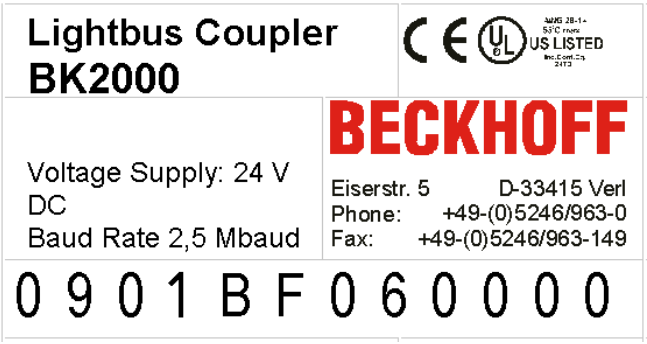
Condition	Permissible range
Permissible ambient temperature during storage	-25°C... +85°C
Relative humidity	95 %, no condensation
Free fall	up to 1 m in the original packaging

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. For sample, on the BK2000 Bus Coupler:



The following information is printed on the label:

Printed item	Meaning for this label
Precise product identification	Lightbus Coupler BK2000
Supply voltage Us	24 V_{DC} (Use a 4 A fuse or a Class 2 power supply to meet UL requirements)
Data transfer rate	2.5 Mbaud
Manufacturer	Beckhoff Automation GmbH & Co. KG
CE mark	Conformity mark
UL mark c UL US LISTED Ind. Cont. Eq.: 24TB Use 4 Amp. fuse or Class 2 power supply. See instructions.	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	From left to right, this sequence of characters indicates the week of production (2 characters), the year of production (2 characters), the software version (2 characters) and hardware version (2 characters), along with any special indications (4 characters). In this case the device is a BK2000 - produced in calendar week 9 - of 2001 - with firmware version BF - in hardware version 6 - without special designation

8.4 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

8.5 Bibliography

TCP/IP

TCP/IP (German)
Configuration and operation of a TCP/IP network
by Kevin Washburn, 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

8.6 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

8.7 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.

Castel Maggiore (BO) 40013
Via Lirone 60/i

e-mail: info@eurotecsrl.com
Tel: +39 051/6321595
Website: www.eurotecsrl.com

C.F.: 04103780377
P.I.: 01596581205