



ibaPDA-Interface-VIP-TCP/UDP

Data Interface TCP/UDP for VIP Protocol

Manual
Issue 3.0

Measurement Systems for Industry and Energy
www.iba-ag.com

Manufacturer

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Contacts

Main office +49 911 97282-0
Support +49 911 97282-14
Engineering +49 911 97282-13
E-mail iba@iba-ag.com
Web www.iba-ag.com

Unless explicitly stated to the contrary, it is not permitted to pass on or copy this document, nor to make use of its contents or disclose its contents. Infringements are liable for compensation.

© iba AG 2024, All rights reserved.

The content of this publication has been checked for compliance with the described hardware and software. Nevertheless, discrepancies cannot be ruled out, and we do not provide guarantee for complete conformity. However, the information furnished in this publication is updated regularly. Required corrections are contained in the following regulations or can be downloaded on the Internet.

The current version is available for download on our web site www.iba-ag.com.

Version	Date	Revision	Author	Version SW
3.0	01-2025	New version ibaPDA v8	nm	8.6.0

Windows® is a brand and registered trademark of Microsoft Corporation. Other product and company names mentioned in this manual can be labels or registered trademarks of the corresponding owners.

Contents

1	About this documentation	5
1.1	Target group and previous knowledge	5
1.2	Notations	6
1.3	Used symbols.....	7
2	System requirements	8
3	Data Interface TCP/UDP for VIP	9
3.1	General Information	9
3.1.1	Vendor Internet Protokoll (VIP)	9
3.1.2	TCP/IP and UDP.....	10
3.1.3	How VIP works.....	11
3.2	Communication between ibaPDA and ABB (Controller)	12
3.3	Data structures	15
3.4	Configuration Guide.....	17
3.5	ibaPDA Configuration & Engineering.....	18
3.5.1	General settings.....	18
3.5.2	General interface settings.....	19
3.5.3	Adding a module.....	20
3.5.4	General Module Settings	21
3.5.4.1	Module Type Integer	22
3.5.4.2	Module Type Real	22
3.5.4.3	Generic module type	22
3.5.5	General signal configuration.....	23
3.5.6	Module diagnostics.....	24
4	Diagnostics.....	25
4.1	License	25
4.2	Visibility of the interface.....	25
4.3	Log files.....	26
4.4	Connection diagnostics with PING.....	27
4.5	Checking the connection	28
4.6	Dagnostic modules	30

5	Appendix	34
5.1	Troubleshooting.....	34
5.1.1	TCP performance problems caused by Delayed Acknowledge	34
5.1.2	TCP data corruption resulting from the Nagle's Algorithm.....	36
6	Support and contact.....	38

1 About this documentation

This documentation describes the function and application of the software interface *ibaPDA-Interface-VIP-TCP/UDP*.

Other documentation



This documentation is a supplement to the *ibaPDA* manual. Information about all the other characteristics and functions of *ibaPDA* can be found in the *ibaPDA* manual or in the online help.

1.1 Target group and previous knowledge

This documentation is aimed at qualified professionals who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as professional if he/she is capable of assessing safety and recognizing possible consequences and risks on the basis of his/her specialist training, knowledge and experience and knowledge of the standard regulations.

This documentation in particular addresses persons, who are concerned with the configuration, test, commissioning or maintenance of Programmable Logic Controllers of the supported products. For the handling *ibaPDA-Interface-VIP-TCP/UDP* the following basic knowledge is required and/or useful:

- Windows operating system
- Basic knowledge of *ibaPDA*
- Knowledge of configuration and operation of the relevant measuring device/system

1.2 Notations

In this manual, the following notations are used:

Action	Notation
Menu command	Menu <i>Logic diagram</i>
Calling the menu command	<i>Step 1 – Step 2 – Step 3 – Step x</i> Example: Select the menu <i>Logic diagram – Add – New function block</i> .
Keys	<Key name> Example: <Alt>; <F1>
Press the keys simultaneously	<Key name> + <Key name> Example: <Alt> + <Ctrl>
Buttons	<Key name> Example: <OK>; <Cancel>
Filenames, paths	<i>Filename, Path</i> Example: <i>Test.docx</i>

1.3 Used symbols

If safety instructions or other notes are used in this manual, they mean:

Danger!



The non-observance of this safety information may result in an imminent risk of death or severe injury:

- Observe the specified measures.
-

Warning!



The non-observance of this safety information may result in a potential risk of death or severe injury!

- Observe the specified measures.
-

Caution!



The non-observance of this safety information may result in a potential risk of injury or material damage!

- Observe the specified measures
-

Note



A note specifies special requirements or actions to be observed.

Tip



Tip or example as a helpful note or insider tip to make the work a little bit easier.

Other documentation



Reference to additional documentation or further reading.

2 System requirements

The following system requirements are necessary for the use of the data interface TCP/UDP for VIP protocols:

- *ibaPDA* v8.0.0 or higher
- License for *ibaPDA-Interface-VIP-TCP/UDP*
- Network connection 10/100 Mbits
- ABB controller with TCP/IP communication interface, e.g. CI861

For more requirements on the PC hardware used and the supported operating systems, see the *ibaPDA* documentation.

Note



It is recommended carrying out the TCP/IP communication on a separate network segment to exclude a mutual influence by other network components.

System restrictions

For different ways of handling the TCP/IP-Acknowledge see ➤ *TCP performance problems caused by Delayed Acknowledge*, page 34 (all *ibaPDA* versions).

Licenses

Order no.	Product name	Description
31.001065	ibaPDA-Interface-VIP-TCP/UDP	Extension license for an <i>ibaPDA</i> system by one TCP/IP and UDP/IP interface Number of connections: 64
31.101065	one step up Interface VIP TCP/UDP	Extension license for the extension of an existing <i>ibaPDA-Interface-VIP-TCP/UDP</i> interface by other 64 TCP/UDP connections, max. 3 permitted

3 Data Interface TCP/UDP for VIP

3.1 General Information

3.1.1 Vendor Internet Protokoll (VIP)

The Vendor Internet Protocol (VIP) serves as a priority as communication between the ABB AC450RMC controller and other computers or systems which do not come from ABB but can process this protocol.

The VIP functionality depends on the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP) and the Internet Protocol (IP) for Ethernet.

Within the context of this documentation, the VIP protocol is used to measure different data from an ABB controller with the data measurement system *ibaPDA*.

The signals to be measured are selected by mapping the values in the telegram buffer whose data blocks are defined by the module types of *ibaPDA*. The telegrams are sent to the *ibaPDA* PC as standard transmitter block.

Three module types are defined in *ibaPDA Interface VIP TCP/UDP*:

- Integer: 32 analog values (Integer) and 32 binary signals
- Real: 8, 16 or 32 analog values (Real) and 32 binary signals
- Generic data structure with a maximum length of 40,961 bytes.

Every module is assigned to a connection On *ibaPDA* side up to max. 256 connections can be established. On the ABB side, the maximum number of connections depends on the CPU type.

The following ABB controller can communicate with *ibaPDA* via the VIP protocol:

- AC450 RMC
- AC800M
- AC80
- AC800 PEC

As a main advantage, this type of data acquisition does not require any special hardware if the controller already features an Ethernet connection.

3.1.2 TCP/IP and UDP

TCP/IP is a data transport protocol which can contain data of any application or participant. This means that TCP/IP is a means to transmit data via a default protocol to default interfaces (e. g. Ethernet network cards offered by a wide range of providers)

Even if a TCP/IP driver can send and receive data, the user data content has to be interpreted by the user. Only the user data are significant to the user.

TCP/IP Package	
Header	Data

The TCP/IP message header normally contains not only the control information but also the source and target address of the message. This part of data contains data with a specific structure so that an *ibaPDA* application can interpret these.

For the connection between ABB and *ibaPDA* not only the understanding of the data structure but also the transmission order and the message header is important. Thus, it is possible to write a specific TCP/IP drive which reads the data packages and enables them for the *ibaPDA* interface for transcription and analysis. *ibaPDA* works with the TCP/IP drive like a connecting server, the automation devices work like clients. This means that *ibaPDA* monitors the information sent and the connection requirements.

ibaPDA can work on the ABB TCP/IP connection with a maximum sampling rate of 5 ms. However, additional restrictions on the sender side can occur. These will be specified later.

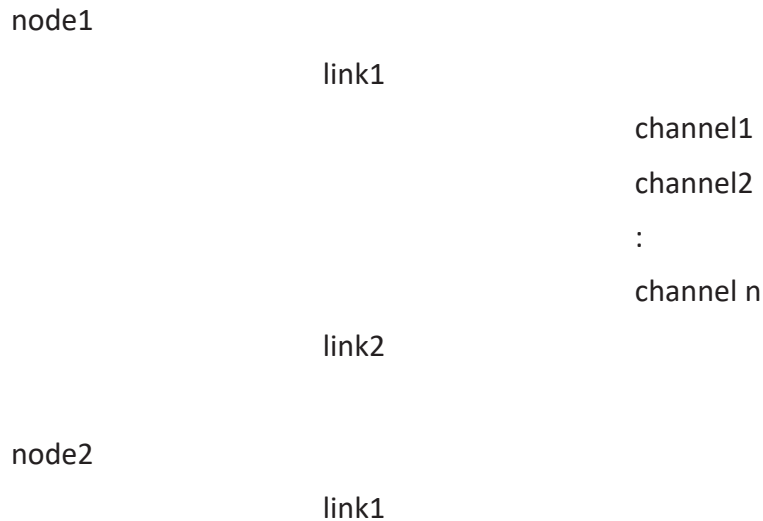
The **Transmission Control Protocol**, short TCP, is a connection-oriented protocol and shall prevent significant data losses, split data and data streams and assign data packages to applications.

The **User Datagram Protocol**, short UDP, is a connectionless transport protocol and works on the layer4, the transport layer, of the OSI layer modell. Its function is similar to that of the connection-oriented TCP. However, it works connectionless and therefore insecure. This means that the sender does not know whether the data packets it has sent have actually arrived. TCP sends confirmations upon receiving data, UDP does not. This method has the advantage that the packet header is much smaller and no acknowledgments have to be sent over the link. In principle, this enables a slightly higher data rate.

3.1.3 How VIP works

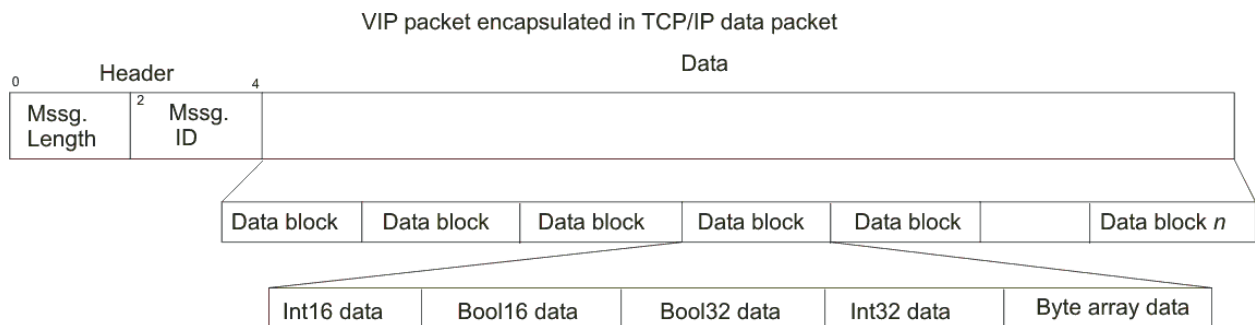
The communication topology is displayed in the following figure:

network



Each station must be assigned a unique node number (Node 1..99) and a unique IP address on the Ethernet. Every server can establish connections to a maximum of five different data sources. Every station (controller) can be configured as client or server (call VIP NETW, VIP-NODE, VIP-LINK, see also VIP manual, pages 43,45 and 39).

The following figure TCP/IP VIP Telegram Structure provides details on the VIP package structure within a TCP/IP or UDP package:



The maximum length of TCP/IP message blocks is 65,535 bytes. Usually, the message length contains the number of bytes of the complete data package, header included. Message ID can be configured and can adopt any value needed by the user. Then follow any number of data blocks which are always identically structured. The first data are always INT16, then Bool16 etc., even until the byte arrays at the end of each block. When data types are missing, no blank characters will be filled in. The overall structure of the data block has to be determined bindingly for both sides (sender and recipient).

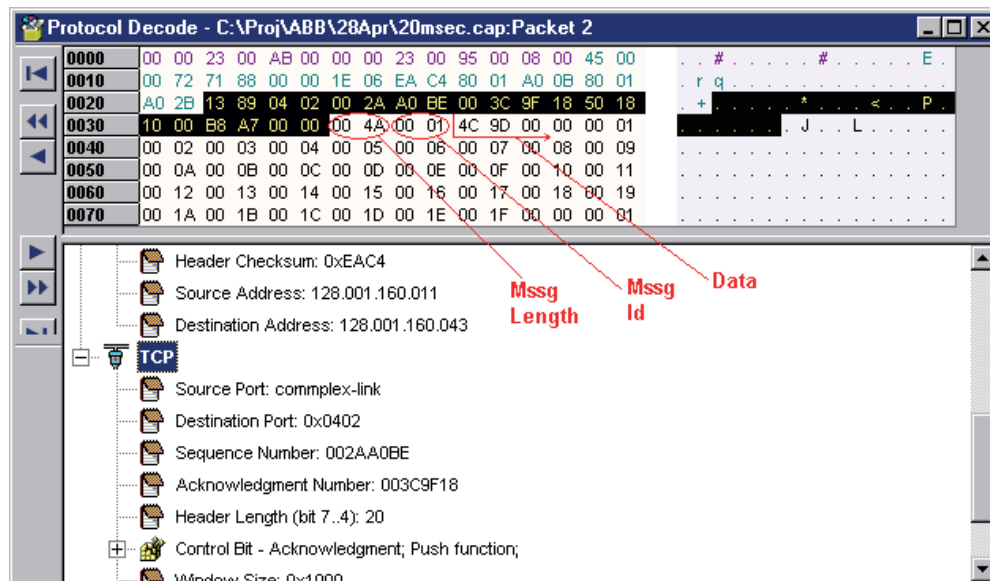
Example

If the user only sends two INT16, one bool32 and one INT32 value with the message ID 2, the data structure will be as follows:

16	2	Int16 Val1	Int16 Val2	Bool32 Val	Int32 Val
----	---	------------	------------	------------	-----------

The length in bytes 16 is therefore: 2 for message length, 2 for message ID, 2x2 for both INT16 values and 4 each for Bool32 and INT32.

Checking the data package: Please note that the notation is in hex format and that the single bytes are represented by two characters, however the bytes are separated by a small blank portion.



The complete TCP/IP package is displayed in different colours depending on their affiliation. At the end of the section marked in black in the table of TCP/IP VIP Content of the Data Package, the data section of the VIP packages begins. The first two bytes 00 4A represent the message length in bytes, consequently 74 bytes. The two following bytes 00 and 01 represent the message ID, here 1. Then follows the 2 byte INT16 value 4C 9D (19613 decimal). Thereafter, 32 INT16 values follow with the content 00 00 to 00 1F (0 to 31 decimal). In the end, there are 4 bytes with the value INT32 1. Thus, together a total length of 74 bytes.

3.2 Communication between ibaPDA and ABB (Controller)

iba AG developped a TCP/IP drive to extract data from the ABB VIP package and to display this data on *ibaPDA*.

Supported connections in *ibaPDA*:

- With *ibaPDA* from version 6.14.0 in total 64 modules of the type Integer, Real and/or “Generic” are configured for the interface TCP/IP-VIP. Every integer or real module can contain up to 32 analog and digital signals. Generic modules can contain up to 1000 analog and digital signals.

- From *ibaPDA* version 6.31.0 up to 64 modules (connections) per interface as before will be supported. However, 4 licenses in total can be used in *ibaPDA*, as a result up to 256 connections are possible.
The maximum length of the message is limited to 4096 bytes. Generic modules can contain up to 1000 analog and 1000 digital signals.
- From *ibaPDA* version 6.33.2 the UDP protocol will be supported as well. The sender can send the data via TCP/IP or UDP. Communication parameter and data structure are identical.

ibaPDA works as a server which listens to clients who have a valid connect-request and then send data (see also the illustration in [➤ Sequence counter](#), page 13). Every TCP/IP or UDP connection can be seen in the diagnostics window of *ibaPDA* or in the I/O Manager. *ibaPDA* is already preset so that the port 5001 monitors the ABB VIP information. Therefore this has to be entered as target port. Please note, that the IP address can only occur once, however more than one connection per IP address is possible. Every connection corresponds to a module on *ibaPDA*, therefore a unique message ID (complies with the *ibaPDA* module index) is needed.

The message ID has to be unique in the complete system, even if several ABB controllers are used. This means that the ABB VIP controller can establish up to 5 links and this as a fact every controller can use up to 5 *ibaPDA* modules. Every module needs a unique source port number for this which will be provided automatically by the system when the user configures a new link for every module.

Module index:

Module Index is the identifier for assigning the data record to the interface module in *ibaPDA*. The module type is also encrypted in this index: The index is created by a serial number 00....63 and an offset that corresponds to the module type and license.

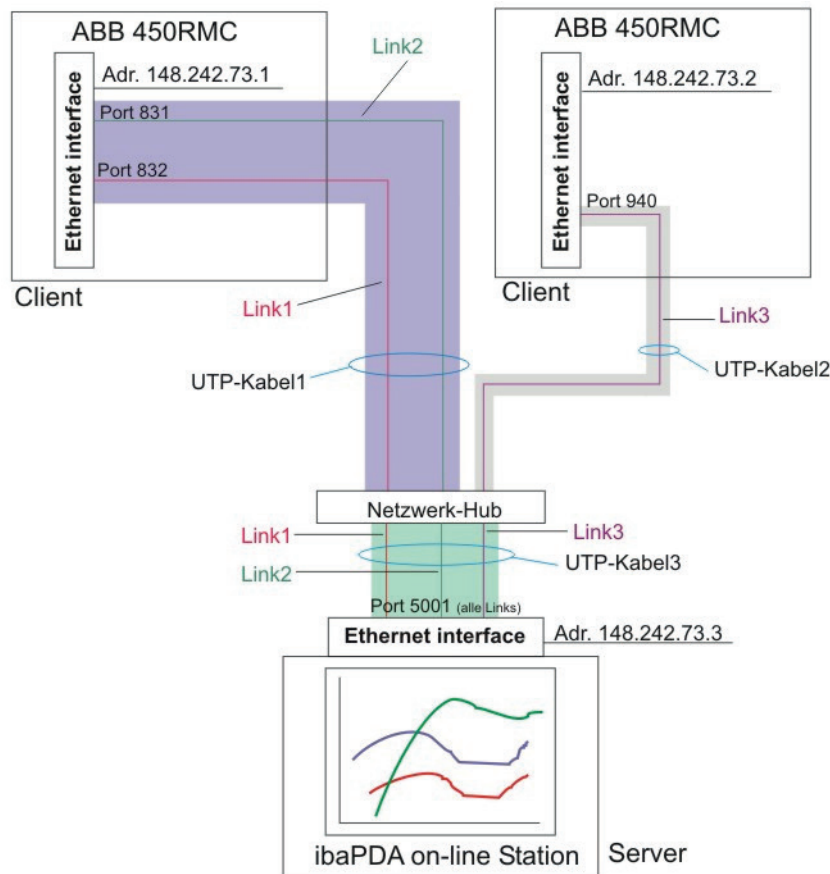
Module type	1 st license	2 nd license	3 rd license	4 th license
Integer	0-63	1000-1063	2000-2063	3000-3063
Real	100-163	1100-1163	2100-2163	3100-3163
General	200-263	1200-1263	2200-2263	3200-3263

The module index complies with the *ibaPDA* module settings. This value must not be changed during data transmission.

The message ID has to be entered in the module index in order to inform *ibaPDA* about which module should be addressed. This ID has to be entered on the ABB VIP page.

Sequence counter

ibaPDA is furthermore able to detect errors in communication. For this function *iba* has determined that the first data bytes should form an upwards counter which has to be filled by increments of 1 with every transmission cycle. This counter has to be the first INT16 value on the ABB side. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF 0x8000) or from 65535 to 0 (0xFFFF 0x0000).



Note that in the illustration two ABBAC450 controller act as clients, whereas every controller has its own Ethernet module which is connected to the network with one cable each. There is another cable which connects *ibaPDA* with the hub (the cables are displayed here in different thickness and colour). The network has three communication partner with unique IP address each.

Note: If the connections are carried out as UTP, ThinNetCoax or optical has to be determined by the respective application. Because three participants are available, a minimum of three modules within *ibaPDA* are used (indicated here with three curves in different colors). Please note that every participant has a random source port number (940, 831 and 832), however they all have the same target port number (Target Port No.) have 5001.

Note



Important for the use of a cross over cable with the ABB controller and *ibaPDA*: Because most of the ABB controller communicate with 10MBit/s in half duplex mode, we recommend to set the Ethernet interface of the *ibaPDA* PC to 10 Mbit/s, half duplex as well if you use a cross-over cable.

If you cannot set the transmission speed to half duplex, then use a network hub or switch between the controller and *ibaPDA*.

3.3 Data structures

Because *ibaPDA* has to know the exact situation and type of each value, iba has determined the following telegram structure for the different data formats:

For each message only one telegram structure can be used.

VIP_32_Integer: 32 integer values + 32 binary values

rel. #	bytes	C type	Description	Comment
00	2	short int	Message Length	set to 74 byte
02	2	short int	Message ID	Module index in <i>ibaPDA</i> , see ↗ <i>Communication between ibaPDA and ABB (Controller)</i> , page 12
04	2	unsigned short	1 st INT16-Signal	Sequence counter: Increment with every transmission cycle! When overflow 65535 0
06	64	short int	32 analog values	
70	4	long int	32 digital values	

VIP_8_Real: 8 Real values + 32 binary values

rel. #	bytes	C type	Description	Comment
00	2	short int	Message Length	Set to 42 byte.
02	2	short int	Message ID	Module index in <i>ibaPDA</i> , see ↗ <i>Communication between ibaPDA and ABB (Controller)</i> , page 12
04	2	unsigned short	1 st INT16-Signal	Sequence counter: Increment with every transmission cycle! When overflow 65535 0
06	4	long int	32 digital values	
10	32	float	8 analog values	IEEE format

VIP_16_Real: 16 Real values + 32 binary values

rel. #	bytes	C type	Description	Comment
00	2	short int	Message Length	set to 74 byte
02	2	short int	Message ID	Module index in <i>ibaPDA</i> , see ↗ <i>Communication between ibaPDA and ABB (Controller)</i> , page 12
04	2	unsigned short	1 st INT16-Signal	Sequence counter: Increment with every transmission cycle! When overflow 65535 0

rel. #	bytes	C type	Description	Comment
06	4	long int	32 digital values	
10	64	float	16 analog values	IEEE format

VIP_32_Real: 32 Real values + 32 binary values

rel. #	bytes	C type	Description	Comment
00	2	short int	Message Length	set to 138 byte
02	2	short int	Message ID	Module index in <i>ibaPDA</i> , see ↗ <i>Communication between ibaPDA and ABB (Controller)</i> , page 12
04	2	unsigned short	1 st INT16-Signal	Sequence counter: Increment with every transmission cycle! When overflow 65535 0
06	4	long int	32 digital values	
10	128	float	32 analog values	IEEE format

VIP_Generic: max. 4096 bytes

rel. #	bytes	C type	Description	Comment
00	2	short int	Message Length	maximum 4102 bytes
02	2	short int	Message ID	Module index in <i>ibaPDA</i> , see ↗ <i>Communication between ibaPDA and ABB (Controller)</i> , page 12
04	2	unsigned short	1 st INT16-Signal	Sequence counter: Increment with every transmission cycle! When overflow 65535 0
06	Max. 4096	byte, int, word, double int, dou- ble word, float, string also mixed	Analog / digital val- ues	Analog values support text signals

3.4 Configuration Guide

With ABB controllers which support the TCP/IP or UDP-VIP protocol up to 64 connections can be used to send data to *ibaPDA*, whereas in *ibaPDA* up to 64 modules can be configured.

From *ibaPDA* version 6.31.0 up to 4 TCP VIP interfaces will be supported, thus up to 256 connections are possible.

It is ensured that every message ID (module index) is unique and in the TCP/IP or UDP telegram of *ibaPDA* and therefore only one module is addressed in *ibaPDA* respectively.

ibaPDA reacts to port 5001 with regards to ABB VIP.

Note



From *ibaPDA* version 6.33.2 another port can be set up in *ibaPDA*.

ABB controller can only be used in this port. When using the technostings it has to be switched to another available port number.

With every link the first INT126 signal has to be configured as upwards counter which has to be incremented for each transmission cycle.

The length of the message (telegram length / message length) depends on the telegram chosen (32 INT, 8, 16 or 32 REAL or GENERIC). The ABB controller have to meet the layout requirements on the sender side, otherwise the message “Incomplete Datapackages” might be displayed in the diagnostics window (I/OManager) of *ibaPDA*.

Only the *ibaPDA* and the controller/s should be connected to the bus.

In order to be able to generate the time base (base clock) for *ibaPDA*, it is recommended that, in any case, an ibaFOB card has to be installed as an interrupt source.

Note



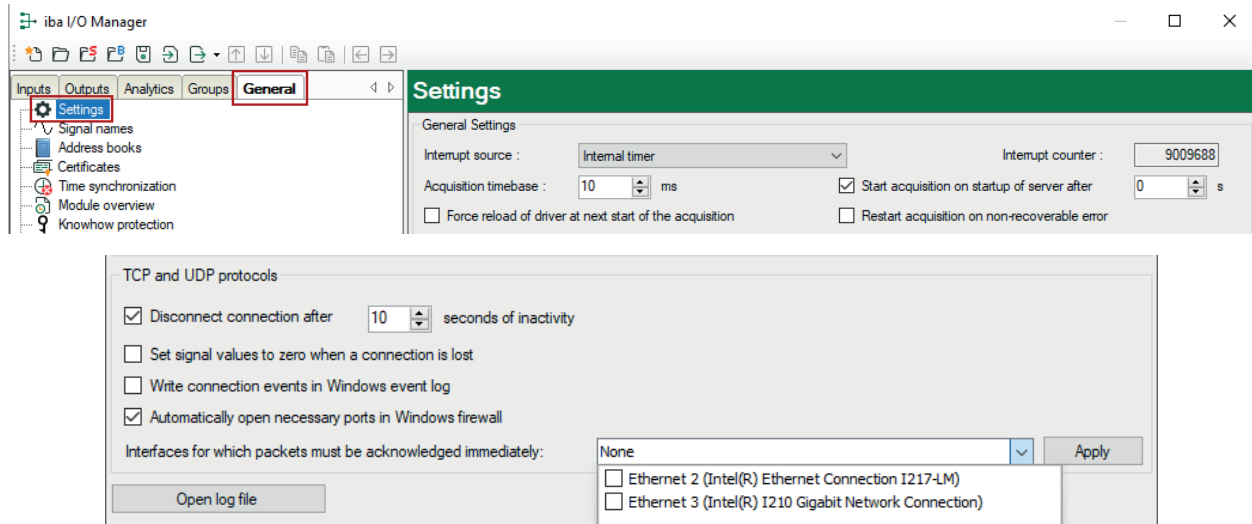
Within the VIP network only the *ibaPDA* and the controller used should be connected but no other participants in order to ensure a high dynamic of the system.

3.5 ibaPDA Configuration & Engineering

The engineering for *ibaPDA* is described in the following. If all system requirements are met (see, page), the interface *VIP TCP/UDP* is displayed in the signal tree.

3.5.1 General settings

The "Alive timeout" is configured jointly for all TCP/IP and UDP protocols supported by *ibaPDA*.



Disconnect connection after ... seconds of inactivity

Behavior and timeout duration can be specified.

Set signal values to zero when a connection is lost

If this option is disabled, the value read last will be kept.

Write connection events in Windows event log

Current events are logged in Windows.

Automatically open necessary ports in Windows firewall

If this option is enabled, all ports required for the currently licensed interfaces are automatically opened in the firewall by the *ibaPDA* server service.

If this option is disabled, the required ports can be opened manually in the I/O Manager of the licensed interfaces via <Allow port through firewall>.

Interfaces for which packets must be acknowledged immediately

Selection of required interfaces.

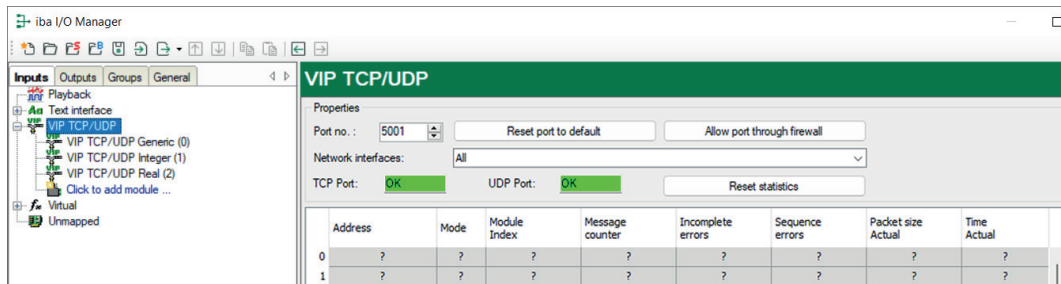
Note



In case *ibaPDA* is the active partner (Client), *ibaPDA* reestablishes the connection after only a few seconds. Thus, it gives to the passive partner the possibility to send data again.

3.5.2 General interface settings

The interface itself has the following functions and configuration options:



Port no.

Used port PC side. The port number has to be used identically in the VIP connection configuration. From *ibaPDA* version 6.33.2 the port number here can be freely chosen.

<Reset port to default>

The port number 5001 is set.

Allow ports through firewall

When installing *ibaPDA*, the default port numbers of the used protocols are automatically entered in the firewall. If you change the port number or enable the interface subsequently, you have to enable this port in the firewall with this button.

Network interfaces

Using this drop-down list, you can select which network adapters on your computer are used for this interface. The sockets will be opened for communication only on the selected network adapters. In case a network adapter has multiple IP addresses configured, a socket will be opened for all of these IP addresses. At least one network adapter should be selected to get the interface configuration validated. If you select *None*, an error message will be displayed when validating the I/O configuration. By default, the option *All* is selected.

TCP Port / UDP Port

OK is displayed here if the socket can be opened on this port. ERROR is displayed if conflicts occur, e. g. if the port is already occupied.

Connection table

see [➤ Checking the connection](#), page 28

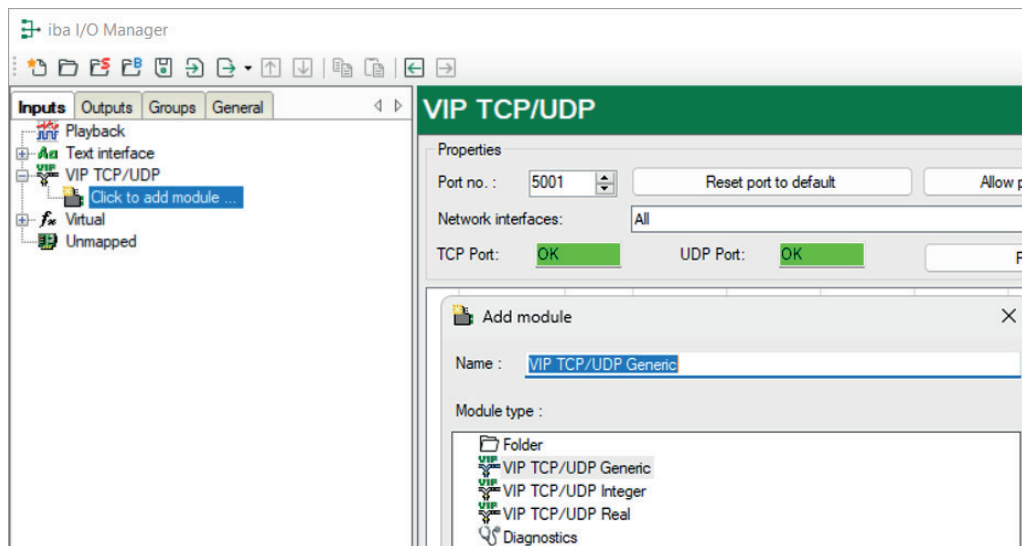
<Reset statistics>

You can use this button to set the calculated time values and the error counter in the table to 0.

3.5.3 Adding a module

Procedure

1. Click on the blue link *Click to add module* located under each data interface in the *Inputs* or *Outputs* tab.
2. Select the desired module type in the dialog box and assign a name via the input field if required.
3. Confirm the selection with <OK>.



Note



If a TCP/IP or UDP connection already exists, right-click the interface and select Auto Autodetect. Then the correct modules are automatically created for all available connections.

Module types

You can add the following module types to the interface:

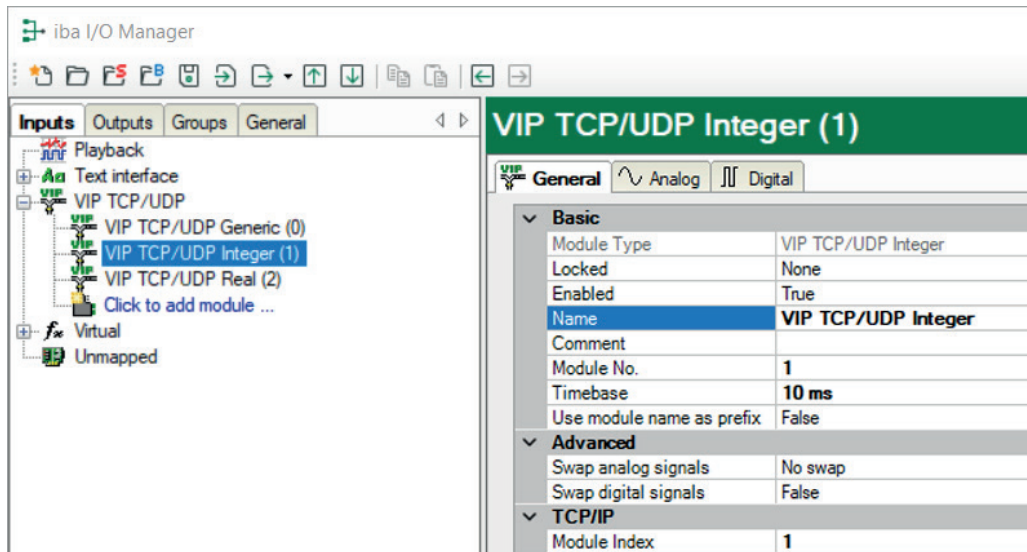
- VIP TCP/UDP Generic
- VIP TCP/UDP Integer
- VIP TCP/UDP Real

see ↗ *General Module Settings*, page 21

3.5.4 General Module Settings

To configure a module, select it in the tree structure.

All modules have the following setting options.



Basic settings

Module Type (information only)

Indicates the type of the current module.

Locked

You can lock a module to avoid unintentional or unauthorized changing of the module settings.

Enabled

Enable the module to record signals.

Name

You can enter a name for the module here.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

Module No.

This internal reference number of the module determines the order of the modules in the signal tree of *ibaPDA* client and *ibaAnalyzer*.

Timebase

All signals of the module are sampled on this timebase.

Use module name as prefix

This option puts the module name in front of the signal names.

Advanced

Swap analog signals/Swap digital signals

Option to change the order of the byte evaluation. The swap mode to be selected depends on the swap mode of the signal source.

TCP/IP

Module Index:

The module indices are created by a serial number 00....63 and an offset that corresponds to the module type and the license.

See ➤ *Communication between ibaPDA and ABB (Controller)*, page 12.

3.5.4.1 Module Type Integer

The integer module allows up to 32 analog values (integer) and 32 binary signals to be acquired. The module does not have any module-specific settings.

3.5.4.2 Module Type Real

The real module allows up to 32 analog values (real) and 32 binary signals to be acquired. The following module settings are module-specific:

Number of analog signals

The number of analog signals to be acquired is configurable in the increments 8, 16 and 32 (the number of digital signals is fixed at 32).

3.5.4.3 Generic module type

With the *Generic* module type, you can measure any data structures with a maximum length of 4096 bytes.

The following module settings are module-specific:

Text encoding

You can select the type of text encoding or the code page here for a correct interpretation and display of the received text data for inputs as well as of the text data to be sent for outputs. Available for selection are, beside system locale according to the Windows system settings (default) and UTF-8 Unicode, all other encodings.

No. analog signals/No. digital signals

Define the number of configurable analog and digital signals in the signal tables. The default value is 32 for each. The maximum value is 1000. The signal tables are adjusted accordingly.

Note



The module *VIP TCP/UDP* supports the acquisition and processing of strings as text signals. Therefore, you can select the datatype `STRING[32]` in the *Analog* tab. In order to convert a text signal oder to split it up into several text signals use the *text splitter* module under the *Virtual* interface.

3.5.5 General signal configuration

The data to be measured are selected on the ABB side by mapping the signals in data blocks, which are cyclically sent to *ibaPDA*.

Analog and Digital tab

Name	Unit	Gain	Offset	Address	DataType	Active	Actual
0		1	0	0x0	INT	<input checked="" type="checkbox"/>	0
1		1	0	0x2	INT	<input checked="" type="checkbox"/>	0
2		1	0	0x4	INT	<input checked="" type="checkbox"/>	0
3		1	0	0x6	INT	<input checked="" type="checkbox"/>	0
4		1	0	0x8	INT	<input checked="" type="checkbox"/>	0
5		1	0	0xA	INT	<input checked="" type="checkbox"/>	0

Name

Enter a meaningful plain text name for the signal.

Unit (analog signals only)

Assignment of a physical unit for the signal

You can enter a maximum of 11 characters, the field is only considered a comment field. The unit is always displayed in conjunction with a numerical display of the values.

Gain, Offset (analog signals only)

Specification of gain and offset for scaling the incoming values

The values describe a linear characteristic curve for scaling. If incoming values are specified in physical units, you can ignore this function, i.e. Gain = 1 and Offset = 0.

Address

The address space is depending on the data type. Hence, an adjustment of address entries may be necessary after change of data types.

The digital signals are addressed via the *Address* and *Bit no.* 0-31 columns.

Data type (analog signals only)

In the fields of this column you can select the data type of each signal. Just click in the corresponding field and select the data type from the list.

The following data types are available: INT, SINT, BYTE, WORD, DINT, DWORD, FLOAT, DOUBLE, STRING[32]

Active

Activation or deactivation of the respective signal

Actual

Display of the current actual value of the signal

Other documentation



For a description of the columns, please see the *ibaPDA* manual.

Tip

You can use the automatic fill function in the columns, see *ibaPDA* manual.

3.5.6 Module diagnostics

The tables *Analog* and *Digital* of the VIP-TCP/UDP modules show the telegram contents.

VIP TCP/UDP								
General		Analog		Digital				
	Name	Unit	Gain	Offset	Address	DataType	Active	Actual
0	TCP Generic Digitals 0-31		1	0	0	DWORD	✓	1128948568
1	TCP Generic Digitals 32-63		1	0	4	DWORD	✓	1153475416
2	TCP Generic Integer 0		1	0	8	INT	✓	-23720
3	TCP Generic Integer 1		1	0	10	INT	✓	-23720
4	TCP Generic Integer 2		1	0	12	INT	✓	-23720
5	TCP Generic Integer 3		1	0	14	INT	✓	-23720
6	TCP Generic Integer 4		1	0	16	INT	✓	-23720
7	TCP Generic Integer 5		1	0	18	INT	✓	-23720
8	TCP Generic Integer 6		1	0	20	INT	✓	-23720

The following errors may occur:

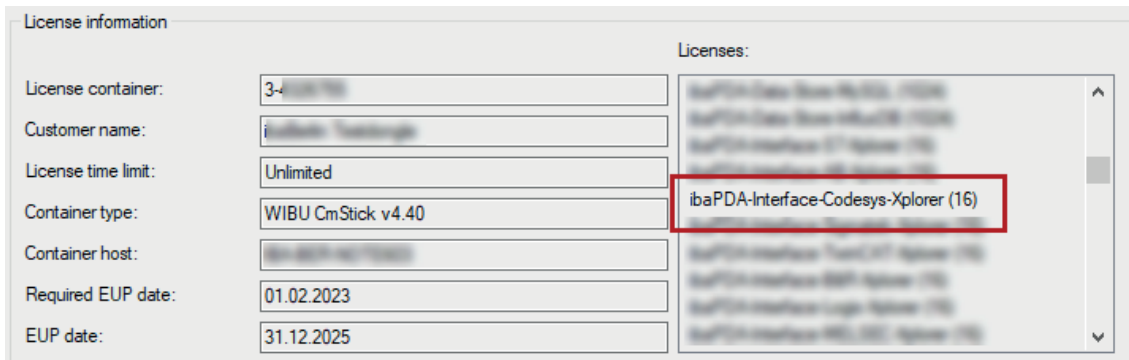
- No data are displayed:
 - The telegram buffer on the sender side is not filled correctly
 - The connectors of the transmitter block are wired incorrectly
- Incorrect values are displayed:
 - The telegram buffer on the sender side is not filled correctly (offset error)
 - The byte order is set incorrectly (see ↗ *General Module Settings*, page 21).
 - There are multiple modules with the same module index.
- The digital signals are sorted incorrectly.
 - The byte order is set incorrectly (see ↗ *General Module Settings*, page 21).
- The telegrams arrive not faster than ca. 200 ms with sequence error
 - Problem with "Delayed Acknowledge", see ↗ *TCP performance problems caused by Delayed Acknowledge*, page 34
 - Problem caused by "Nagle's Algorithm", see ↗ *TCP data corruption resulting from the Nagle's Algorithm*, page 36

4 Diagnostics

4.1 License

If the interface is not displayed in the signal tree, you can either check in *ibaPDA* in the I/O Manager under *General – Settings* or in the *ibaPDA* service status application whether your license for the interface *ibaPDA-Interface-VIP-TCP/UDP* has been properly recognized. The number of licensed connections is shown in brackets.

The figure below shows the license for the *Codesys Xplorer* interface as an example.



4.2 Visibility of the interface

If the interface is not visible despite a valid license, it may be hidden.

Check the settings in the *General* tab in the *Interfaces* node.

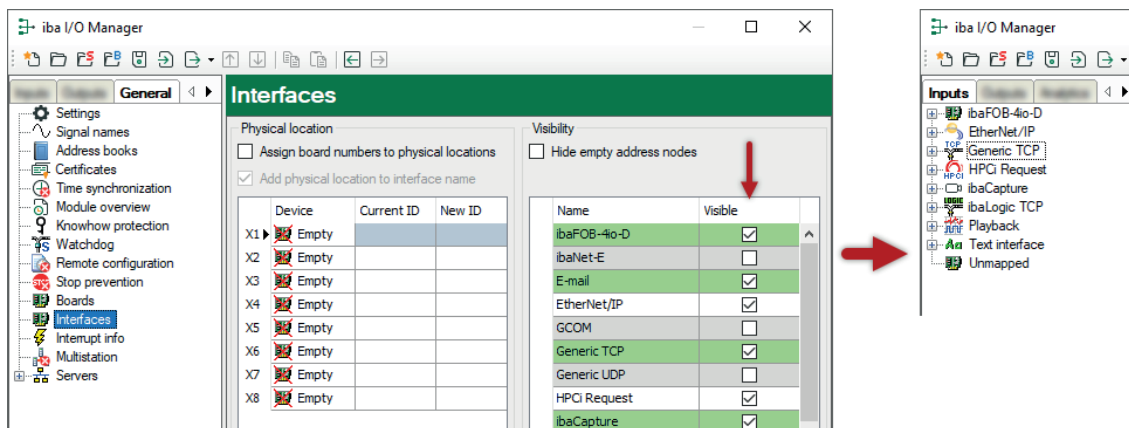
Visibility

The table *Visibility* lists all the interfaces that are available either through licenses or installed cards. These interfaces can also be viewed in the interface tree.

You can hide or display the interfaces not required in the interface tree by using the checkbox in the *Visible* column.

Interfaces with configured modules are highlighted in green and cannot be hidden.

Selected interfaces are visible, the others are hidden:



4.3 Log files

If connections to target platforms or clients have been established, all connection-specific actions are logged in a text file. You can open this (current) file and, e.g., scan it for indications of possible connection problems.

You can open the log file via the button <Open log file>. The button is available in the I/O Manager:

- for many interfaces in the respective interface overview
- for integrated servers (e.g. OPC UA server) in the *Diagnostics* tab.

In the file system on the hard drive, you can find the log files of the *ibaPDA* server (...\[ProgramData\iba\ibaPDA\Log](#)). The file names of the log files include the name or abbreviation of the interface type.

Files named [interface.txt](#) are always the current log files. Files named [Interface_yyyy_mm_dd_hh_mm_ss.txt](#) are archived log files.

Examples:

- [ethernetipLog.txt](#) (log of EtherNet/IP connections)
- [AbEthLog.txt](#) (log of Allen-Bradley Ethernet connections)
- [OpcUAServerLog.txt](#) (log of OPC UA server connections)

4.4 Connection diagnostics with PING

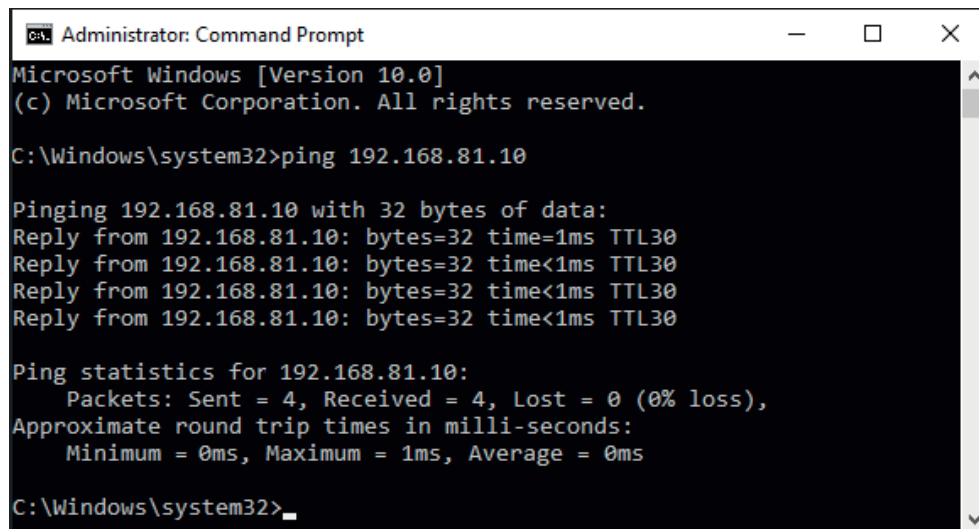
PING is a system command with which you can check if a certain communication partner can be reached in an IP network.

1. Open a Windows command prompt.



2. Enter the command "ping" followed by the IP address of the communication partner and press <ENTER>.

→ With an existing connection you receive several replies.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. All rights reserved.

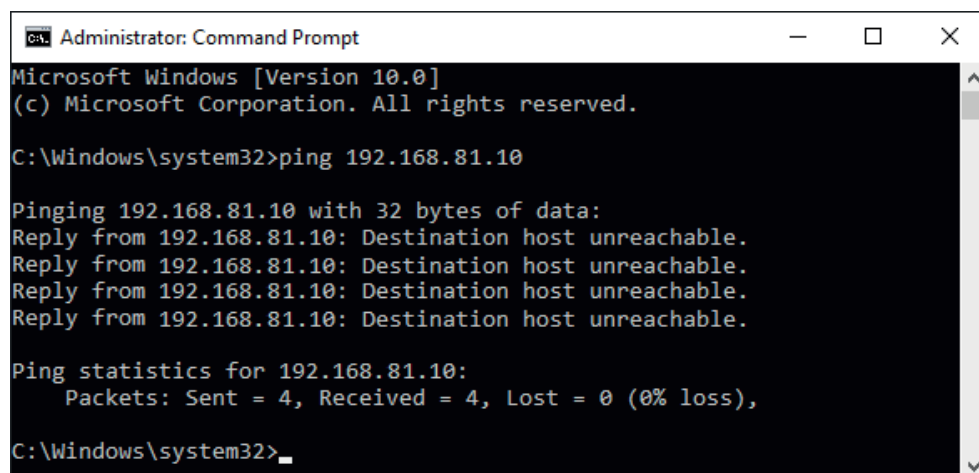
C:\Windows\system32>ping 192.168.81.10

Pinging 192.168.81.10 with 32 bytes of data:
Reply from 192.168.81.10: bytes=32 time=1ms TTL30
Reply from 192.168.81.10: bytes=32 time<1ms TTL30
Reply from 192.168.81.10: bytes=32 time<1ms TTL30
Reply from 192.168.81.10: bytes=32 time<1ms TTL30

Ping statistics for 192.168.81.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Windows\system32>
```

→ With no existing connection you receive error messages.



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>ping 192.168.81.10

Pinging 192.168.81.10 with 32 bytes of data:
Reply from 192.168.81.10: Destination host unreachable.
Reply from 192.168.81.10: Destination host unreachable.
Reply from 192.168.81.10: Destination host unreachable.
Reply from 192.168.81.10: Destination host unreachable.

Ping statistics for 192.168.81.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

C:\Windows\system32>
```

4.5 Checking the connection

After the configuration was accepted, all connections will be shown in the connections overview sorted according to their module index.

VIP TCP/UDP

Properties

Port no.:

Network interfaces:

TCP Port: UDP Port:

	Address	Mode	Module index	Message counter	Incomplete errors	Sequence errors	Packet size Actual	Time Actual
0	192.168.50.94	TCP	0	29855	0	0	74	100,6 ms
1	192.168.50.94	UDP	1	29855	0	0	74	100,9 ms
2	192.168.50.94	TCP	100	29855	0	0	138	100,5 ms
3	192.168.50.94	UDP	101	29855	0	0	138	96,9 ms
4	192.168.50.94	TCP	200	29855	0	0	206	100,7 ms
5	192.168.50.94	UDP	201	29855	0	0	206	96,1 ms

The background color of the lines has the following meaning:

Color	Meaning
Green	The connection is OK. The <i>ibaPDA</i> module timebase is equally quick or slower than the telegram cycle. The current telegram cycle is shown in the column <i>Time Actual</i> .
Orange	The connection is OK, but the telegram cycle is significantly slower than the <i>ibaPDA</i> module timebase. It is recommended to adjust the module timebase to the telegram cycle.

If the connections are not displayed or only partially, this may have the following causes:

- Sender is in Stop mode
- No Ethernet connection between *ibaPDA* PC and the ABB control
- Error in the connection configuration:
 - incorrect remote IP address
 - The *ibaPDA* port number and the connection configuration do not match
 - The port number is blocked by the firewall
- Wrong module index specified in the telegram header

Other errors

- If the telegram counters do not increment continuously, the telegrams are not called cyclically on the sender side.
- If values in the columns "Incomplete errors" and/or "Sequence errors" are incremented, this points to one of the following errors:
 - The "message_length" in the telegram header does not meet the expected value.
 - The "sequence_counter" in the telegram is not incremented correctly.
 - The "Delayed Acknowledge" problem occurs (see [TCP performance problems caused by Delayed Acknowledge](#), page 34)

4.6 Diagnostic modules

Diagnostic modules are available for most Ethernet based interfaces and Xplorer interfaces. Using a diagnostic module, information from the diagnostic displays (e.g. diagnostic tabs and connection tables of an interface) can be acquired as signals.

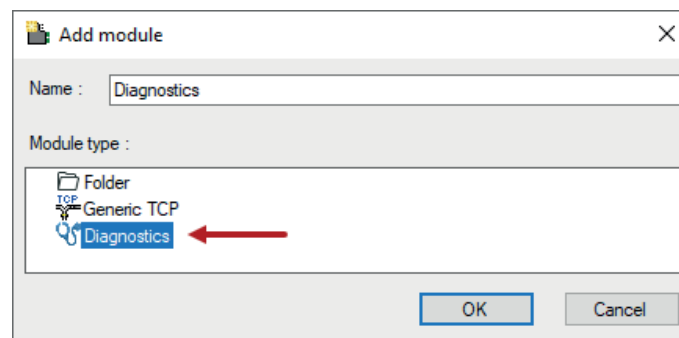
A diagnostic module is always assigned to a data acquisition module of the same interface and supplies its connection information. By using a diagnostic module, you can record and analyze the diagnostic information continuously in the *ibaPDA* system.

Diagnostic modules do not consume any license connections because they do not establish their own connection but refer to another module.

Example for the use of diagnostic modules:

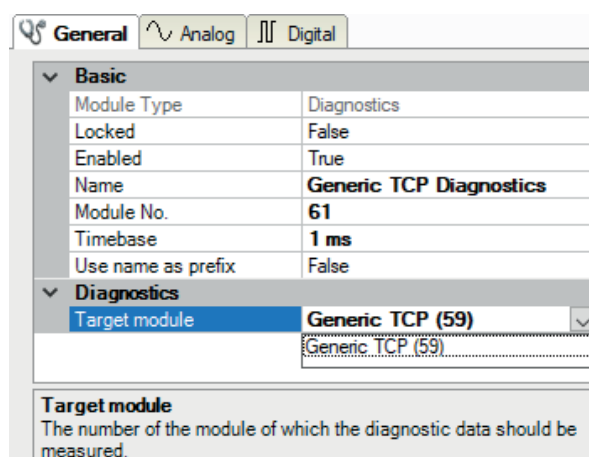
- A notification can be generated, whenever the error counter of a communication connection exceeds a certain value or the connection gets lost.
- In case of a disturbance, the current response times in the telegram traffic may be documented in an incident report.
- The connection status can be visualized in *ibaQPanel*.
- You can forward diagnostic information via the SNMP server integrated in *ibaPDA* or via OPC DA/UA server to superordinate monitoring systems like network management tools.

In case the diagnostic module is available for an interface, a "Diagnostics" module type is shown in the "Add module" dialog (example: Generic TCP).



Module settings diagnostic module

For a diagnostic module, you can make the following settings (example: Generic TCP):



General Analog Digital

Basic

Module Type	Diagnostics
Locked	False
Enabled	True
Name	Generic TCP Diagnostics
Module No.	61
Timebase	1 ms
Use name as prefix	False

Diagnostics

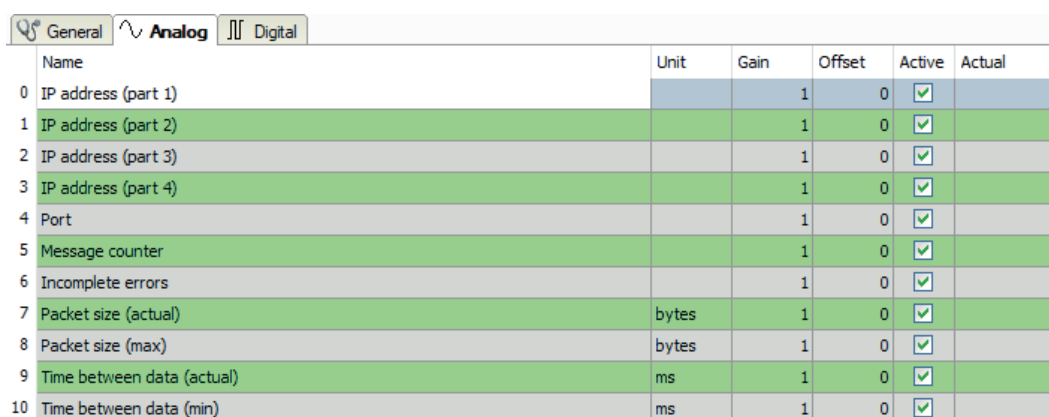
Target module	Generic TCP (59)
---------------	------------------

Target module
The number of the module of which the diagnostic data should be measured.

The basic settings of a diagnostic module equal those of other modules.

There is only one setting which is specific for the diagnostic module: the target module.

By selecting the target module, you assign the diagnostic module to the module on which you want to acquire information about the connection. You can select the supported modules of this interface in the drop-down list of the setting. You can assign exactly one data acquisition module to each diagnostic module. When having selected a module, the available diagnostic signals are immediately added to the *Analog* and *Digital* tabs. It depends on the type of interface, which signals exactly are added. The following example lists the analog values of a diagnostic module for a Generic TCP module.



	Name	Unit	Gain	Offset	Active	Actual
0	IP address (part 1)		1	0	<input checked="" type="checkbox"/>	
1	IP address (part 2)		1	0	<input checked="" type="checkbox"/>	
2	IP address (part 3)		1	0	<input checked="" type="checkbox"/>	
3	IP address (part 4)		1	0	<input checked="" type="checkbox"/>	
4	Port		1	0	<input checked="" type="checkbox"/>	
5	Message counter		1	0	<input checked="" type="checkbox"/>	
6	Incomplete errors		1	0	<input checked="" type="checkbox"/>	
7	Packet size (actual)	bytes	1	0	<input checked="" type="checkbox"/>	
8	Packet size (max)	bytes	1	0	<input checked="" type="checkbox"/>	
9	Time between data (actual)	ms	1	0	<input checked="" type="checkbox"/>	
10	Time between data (min)	ms	1	0	<input checked="" type="checkbox"/>	

For example, the IP (v4) address of a Generic TCP module (see fig. above) will always be split into 4 parts derived from the dot-decimal notation, for better reading. Also other values are being determined, as there are port number, counters for telegrams and errors, data sizes and telegram cycle times. The following example lists the digital values of a diagnostic module for a Generic TCP module.



	Name	Active	Actual
0	Active connection mode	<input checked="" type="checkbox"/>	
1	Invalid packet	<input checked="" type="checkbox"/>	
2	Connecting	<input checked="" type="checkbox"/>	
3	Connected	<input checked="" type="checkbox"/>	

Diagnostic signals

Depending on the interface type, the following signals are available:

Signal name	Description
Active	Only relevant for redundant connections. Active means that the connection is used to measure data, i.e. for redundant standby connections the value is 0. For normal/non-redundant connections, the value is always 1.
Buffer file size (actual/avg/max)	Size of the file for buffering statements
Buffer memory size (actual/avg/max)	Size of the memory used by buffered statements
Buffered statements	Number of unprocessed statements in the buffer
Buffered statements lost	Number of buffered but unprocessed and lost statements
Connected	Connection is established
Connected (in)	A valid data connection for the reception (in) is available
Connected (out)	A valid data connection for sending (out) is available
Connecting	Connection being established
Connection attempts (in)	Number of attempts to establish the receive connection (in)
Connection attempts (out)	Number of attempts to establish the send connection (out)
Connection ID O->T	ID of the connection for output data (from the target system to <i>ibaPDA</i>). Corresponds to the assembly instance number
Connection ID T->O	ID of the connection for input data (from <i>ibaPDA</i> to target system). Corresponds to the assembly instance number
Connection phase (in)	Status of the ibaNNet-E data connection for reception (in)
Connection phase (out)	Status of the ibaNNet-E data connection for sending (out)
Connections established (in)	Number of currently valid data connections for reception (in)
Connections established (out)	Number of currently valid data connections for sending (out)
Data length	Length of the data message in bytes
Data length O->T	Size of the output message in byte
Data length T->O	Size of the input message in byte
Destination IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to <i>ibaPDA</i>)
Destination IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from <i>ibaPDA</i> to target system)
Disconnects (in)	Number of currently interrupted data connections for reception (in)
Disconnects (out)	Number of currently interrupted data connections for sending (out)
Error counter	Communication error counter
Exchange ID	ID of the data exchange
Incomplete errors	Number of incomplete messages

Signal name	Description
Incorrect message type	Number of received messages with wrong message type
Input data length	Length of data messages with input signals in bytes (<i>ibaPDA</i> receives)
Invalid packet	Invalid data packet detected
IP address (part 1-4)	4 octets of the IP address of the target system
Keepalive counter	Number of KeepAlive messages received by the OPC UA Server
Lost images	Number of lost images (in) that were not received even after a retransmission
Lost Profiles	Number of incomplete/incorrect profiles
Message counter	Number of messages received
Messages per cycle	Number of messages in the cycle of the update time
Messages received since configuration	Number of received data telegrams (in) since start of acquisition
Messages received since connection start	Number of received data telegrams (in) since the start of the last connection setup. Reset with each connection loss.
Messages sent since configuration	Number of sent data telegrams (out) since start of acquisition
Messages sent since connection start	Number of sent data telegrams (out) since the start of the last connection setup. Reset with each connection loss.
Multicast join error	Number of multicast login errors
Number of request commands	Counter for request messages from <i>ibaPDA</i> to the PLC/CPU
Output data length	Length of the data messages with output signals in bytes (<i>ibaPDA</i> sends)
Packet size (actual)	Size of the currently received message
Packet size (max)	Size of the largest received message
Ping time (actual)	Response time for a ping telegram
Port	Port number for communication
Producer ID (part 1-4)	Producer ID as 4-byte unsigned integer
Profile Count	Number of completely recorded profiles
Read counter	Number of read accesses/data requests
Receive counter	Number of messages received
Response time (actual/average/max/min)	<p>Response time is the time between measured value request from <i>ibaPDA</i> and response from the PLC or reception of the data.</p> <p>Actual: current value</p> <p>Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.</p>
Retransmission requests	Number of data messages requested again if lost or delayed

Signal name	Description
Rows (last)	Number of resulting rows by the last SQL query (within the configured range of result rows)
Rows (maximum)	Maximum number of resulting rows by any SQL query since the last start of acquisition (possible maximum equals the configured number of result rows)
Send counter	Number of send messages
Sequence errors	Number of sequence errors
Source IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to <i>ibaPDA</i>)
Source IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from <i>ibaPDA</i> to target system)
Statements processed	Number of executed statements since last start of acquisition
Synchronization	Device is synchronized for isochronous acquisition
Time between data (actual/ max/min)	Time between two correctly received messages Actual: between the last two messages Max/min: statistical values since start of acquisition or reset of counters
Time offset (actual)	Measured time difference of synchronicity between <i>ibaPDA</i> and the <i>ibaNet-E</i> device
Topics Defined	Number of defined topics
Topics Updated	Number of updated topics
Unknown sensor	Number of unknown sensors
Update time (actual/average/ configured/max/min)	Specifies the update time in which the data is to be retrieved from the PLC, the CPU or from the server (configured). Default is equal to the parameter "Timebase". During the measurement the real actual update time (actual) can be higher than the set value, if the PLC needs more time to transfer the data. How fast the data is really updated, you can check in the connection table. The minimum achievable update time is influenced by the number of signals. The more signals are acquired, the greater the update time becomes. Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Write counter	Number of successful write accesses
Write lost counter	Number of failed write accesses

5 Appendix

5.1 Troubleshooting

In the following you will find help on possible errors when using *ibaPDA-Interface-VIP-TCP/UDP*. If you have any further questions, please contact the iba support.

5.1.1 TCP performance problems caused by Delayed Acknowledge

ibaPDA measurements of automation devices using TCP/IP sometimes do not work with cycle times < 200 ms.

Errors shown in ibaPDA

Incomplete telegrams and/or spikes in data values (depending on the sending controller type)

Cause

There are different variants of handling "acknowledge" in the TCP/IP protocol.

The standard WinSocket works in accordance with RFC1122 using the "delayed acknowledge" mechanism (Delayed ACK). It specifies that the "acknowledge" is delayed until other telegrams arrive in order to acknowledge them jointly. If no other telegrams arrive, the ACK telegram is sent after 200 ms at the latest (depending on the socket).

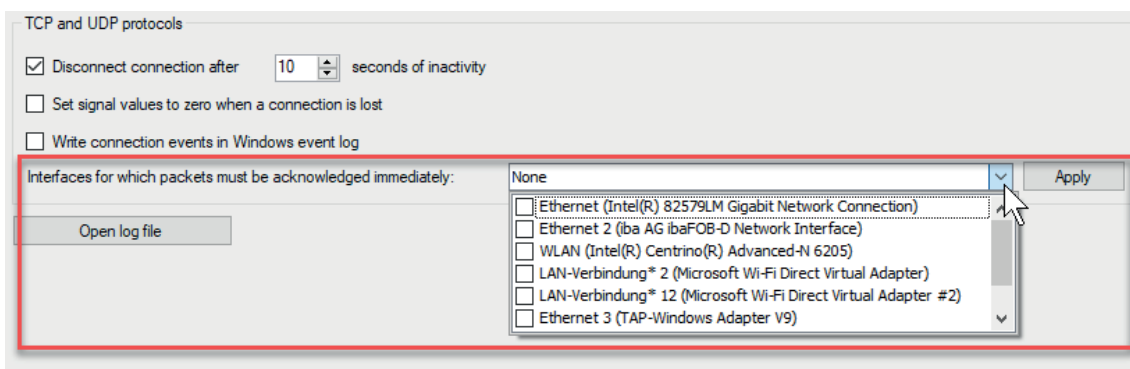
The data flow is controlled by a "sliding window" (parameter Win=nnnn). The recipient specifies how many bytes it can receive without sending an acknowledgment.

Some controllers do not accept this response, but instead, wait for an acknowledgment after each data telegram. If it does not arrive within a certain period of time (200 ms), it will repeat the telegram and include any new data to be sent, causing an error with the recipient, because the previous telegram was received correctly.

Remedy

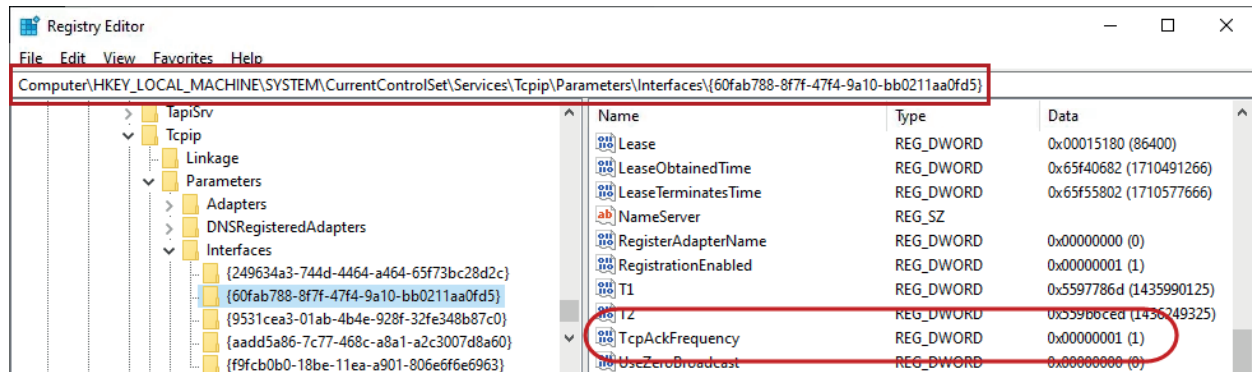
The "delayed acknowledge" can be switched off individually for each network adapter via an entry in the Windows Registry. For easy modification, *ibaPDA* offers a corresponding dialog in the I/O Manager under *General* in the tab *Settings*.

In the list of network adapters, select those for which you want to disable "delayed acknowledge" and click <Apply>.



Thus, the parameter "TcpAckFrequency" (REG_DWORD = 1) is created in the registry path of the selected network adapters:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\
{InterfaceGUID}



Note



Basically, you can avoid such TCP-specific problems by using *UDP* instead of *TCP*.

The User Datagram Protocol (UDP) is a minimal network protocol that is not connection-oriented and is unsecured against telegram loss. Among other things, reception acknowledgement of the sent data is dispensed with. In stable and high-performance networks, however, this is not of significant importance and can be neglected due to the cyclic data transmission common with *ibaPDA*.

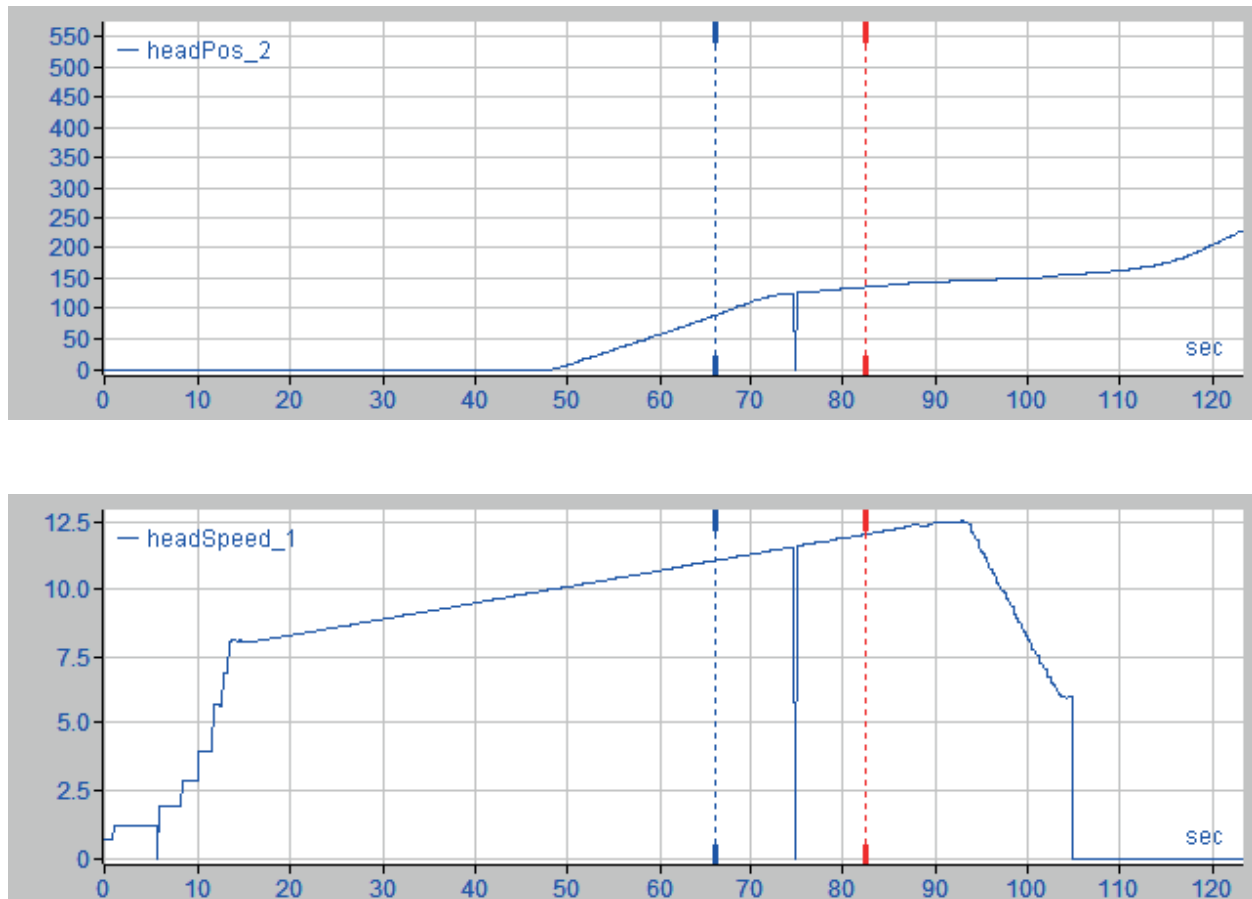
5.1.2 TCP data corruption resulting from the Nagle's Algorithm

Symptoms

ibaPDA measurements of automation devices using TCP/IP show spikes in the data.

Errors shown in ibaPDA

Incomplete telegrams and/or spikes in the data values (see examples in the following figures)



Cause

Nagle's algorithm is one mechanism for improving TCP efficiency by reducing the number of small packets sent over the network and collecting several data blocks before sending the data over the network.

Because the Generic TCP interface does not use an application level protocol, the receiver *ibaPDA* cannot handle these merged messages correctly. The Generic TCP interface expects only 1 datagram per TCP message with always the same layout and length.

But the Nagle's Algorithm and the option *Delayed ACK* (Delayed Acknowledge, see 5.1.1, page 34) do not play well together in a TCP/IP Network:

The Delayed ACK mechanism tries to send more data per segment if it can. But part of Nagle's algorithm depends on an ACK to send data. So Delayed ACKs are waiting to send the ACK while Nagle's algorithm is waiting to receive the ACK.

This creates random stalls of 200 ms to 500 ms on segments that could otherwise be sent immediately and delivered to the receive-side stack of *ibaPDA* as application.

Remedy

It is recommended to start with disabling the *Delayed ACK* mechanism, see chapter 5.1.1, page 34. In a typical real-time application, the transmitter will then send the new data to *ibaPDA* with a certain cycle time because the previous data has been acknowledged immediately. Depending on the implementation of the TCP/IP stack on the sender's side, the Nagle's algorithm can still become active and automatically aggregate a number of small buffer messages, causing the algorithm to purposely slow down the transmission.

This can also happen sporadically due to a momentary overload on the sender side that causes the stack to merge some messages.

To disable Nagle's buffering algorithm, use the *TCP_NODELAY* socket option. The *TCP_NODELAY* socket option allows the network to bypass Nagle's-induced Delays by disabling Nagle's algorithm, and sending the data as soon as it is available.

Enabling *TCP_NODELAY* forces a socket to send the data in its buffer, whatever the packet size. The *TCP_NODELAY* flag is an option that can be enabled on a per-socket basis and is applied when a TCP socket is created.

(See *Socket.NoDelay* property in .NET applications in the *System.Net.Sockets* namespace.)

Note



Basically, you can avoid such TCP-specific problems by using *UDP* instead of *TCP*.

The User Datagram Protocol (UDP) is a minimal network protocol that is not connection-oriented and is unsecured against telegram loss. Among other things, reception acknowledgement of the sent data is dispensed with. In stable and high-performance networks, however, this is not of significant importance and can be neglected due to the cyclic data transmission common with *ibaPDA*.

6 Support and contact

Support

Phone: +49 911 97282-14
Email: support@iba-ag.com

Note



If you need support for software products, please state the number of the license container. For hardware products, please have the serial number of the device ready.

Contact

Headquarters

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Phone: +49 911 97282-0
Email: iba@iba-ag.com

Mailing address

iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address

iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany

Regional and Worldwide

For contact data of your regional iba office or representative please refer to our web site:

www.iba-ag.com