ibaPDA-Interface-VIP-TCP/UDP

Data Interface TCP/UDP for VIP Protocol

Manual

Issue 2.10

Measurement Systems for Industry and Energy

www.iba-ag.com

Manufacturer

iba AG

Koenigswarterstr. 44

90762 Fuerth

Germany

Contacts

Main office	+49 911 97282-0
Fax	+49 911 97282-33
Support	+49 911 97282-14
Engineering	+49 911 97282-13
E-mail	iba@iba-ag.com
Web	www.iba-ag.com

Version	Date	Revision - Chapter / Page	Author	Version SW
2.10	03-2022	Troubleshooting, Nagle's Algorithm (chap. 5.1.2)	RM/IP	7.2.2

Content

1 About this Manual

This document describes the function and application of the software interface

ibaPDA-Interface-VIP-TCP/UDP

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 addresses qualified professionals, who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as a professional if he/she is capable of assessing the work assigned to him/her and recognizing possible risks on the basis of his/her specialist training, knowledge and experience and knowledge of 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 of *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 *Logic diagram*
Calling the menu command	*Step 1 – Step 2 – Step 3 – Step x* Example: Select the menu *Logic diagram - Add - New function block*.
Keys	<Key name> Example: <Alt>; <F1>
Press the keys simultaneously	<Key name> + <Key name> Example: <Alt> + <Ctrl>
Buttons	<Key name> Example: <OK>; <Cancel>
File names, paths	"Filename", "Path" Example: "Test.doc"

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* v7.0.0 oder höher
- 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 or UDP 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 or data corruption problems resulting from the Delayed ACK mechanism*, page 28 (all *ibaPDA* versions).

Licenses

Order no.	Product Designation	Description
31.001065	ibaPDA-Interface-VIP-TCP/UDP	Extension license for an *ibaPDA* 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 an existing interface *ibaPDA-Interface-VIP-TCP/UDP* by other 64 TCP/UDP connections, max. 3 permitted

3 Data Interface TCP/UDP for VIP

3.1 General Information

3.1.1 What is 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 manual, 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 oder 32 analog values (Real) and 32 binary values
- Generic: any data structure with a maximum length of 40961bytes

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 What is 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.

The User Datagram Protocol, short UDP, is a connectionless transport protocol and works on the layer 4, 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 transmission path In principle, this enables a slightly higher data rate.

3.1.3 How does VIP work?

The communcation topology is displayed in the following figure:

network

node1

link1

channel1

channel2

:

channel n

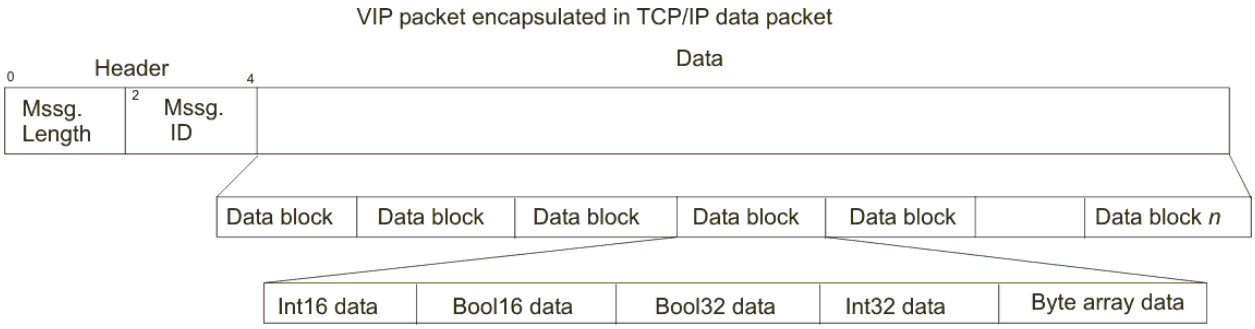
link2

node2

link1

This means that an unique node number (node 1 ... 99) as well as an unique IP address were assigned to every station 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:



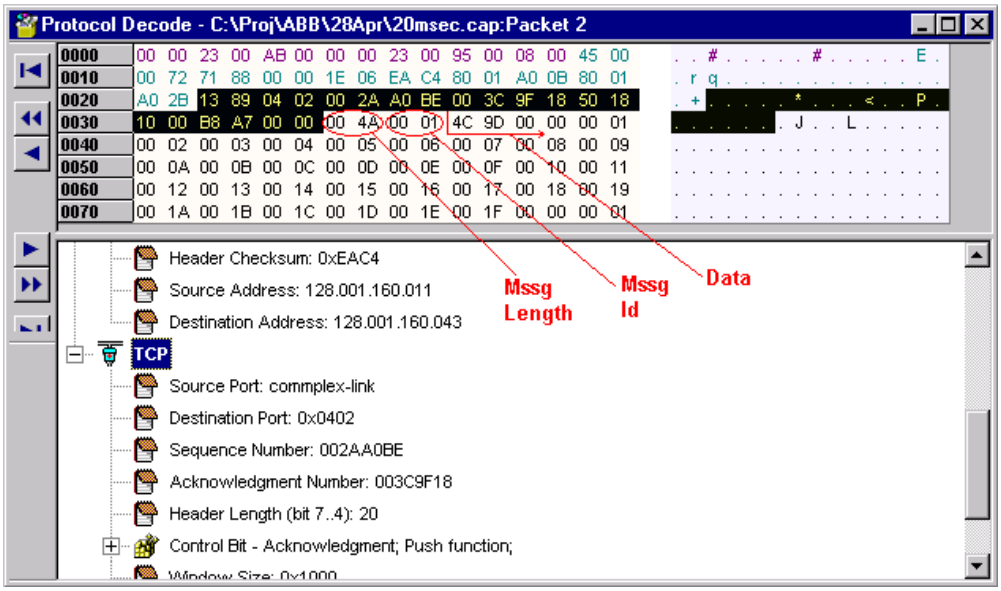
The maximum length of TCP/IP message blocks is 65535 bytes. Usually, the message length contains the number of bytes of the complete data package, header included. Message ID can be configured and can adopt any value needed by the user. Then follow any number of data blocks which are always identically 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 |

This means that the length in bytes is 16. 2 for message length, 2 for message ID, 2x2 for both INT16 values and 4 each for Bool32 and INT32.

Check 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 than send data (see also table *Communication Principle ibaPDA - ABB AC450RMC*). 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. It is important to outline 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 an 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 controller 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.

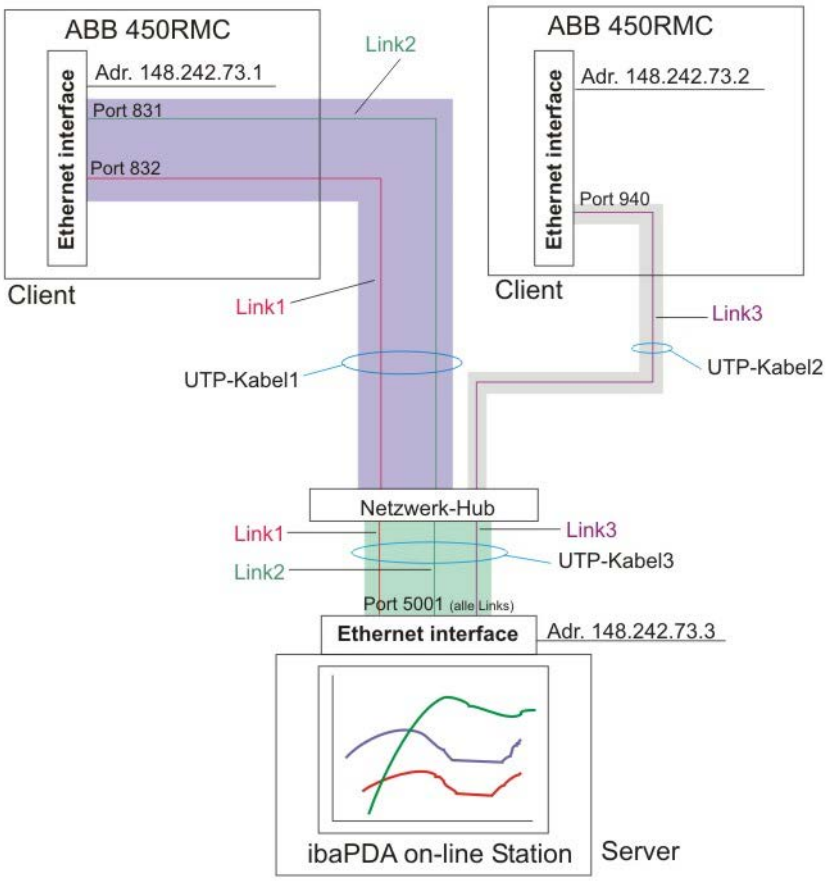
Module type	1. License	2. License	3. License	4. License
Integer	0-63	1000-1063	2000-2063	3000-3063
Real	100-163	1100-1163	2100-2163	3100-3163
Generic	200-263	1200-1263	2200-2263	3200-3263

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).



Regarding figure *Communication Principle ibaPDA - ABB AC450RMC 3* please note that two ABB AC450 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. Since 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*:

Since 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 Structure

Since *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.

3.3.1 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 *ibaPDA*, see ↗ *Communication between ibaPDA and ABB (Controller)*, page 11
04	2	unsigned short	1. 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	

3.3.2 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 *ibaPDA*, see ↗ *Communication between ibaPDA and ABB (Controller)*, page 11
04	2	unsigned short	1. 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

3.3.3 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 *ibaPDA*, see ↗ *Communication between ibaPDA and ABB (Controller)*, page 11

rel. #	bytes	C type	Description	Comment
04	2	unsigned short	1. INT16-Signal	Sequence counter: Increment with every transmission cycle! When overflow 65535 0
06	4	long int	32 digital values	
10	64	float	16 analog values	IEEE format

3.3.4 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 *ibaPDA*, see ↗ *Communication between ibaPDA and ABB (Controller)*, page 11
04	2	unsigned short	1. 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

3.3.5 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 *ibaPDA*, see ↗ *Communication between ibaPDA and ABB (Controller)*, page 11
04	2	unsigned short	1. INT16-Signal	Sequence counter: Increment with every transmission cycle! When overflow 65535 0
06	Max. 4096	byte, int, word, double int, double word, float, string also mixed	Analog / digital values	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 TCPIP 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 technostrings 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/O manager) 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

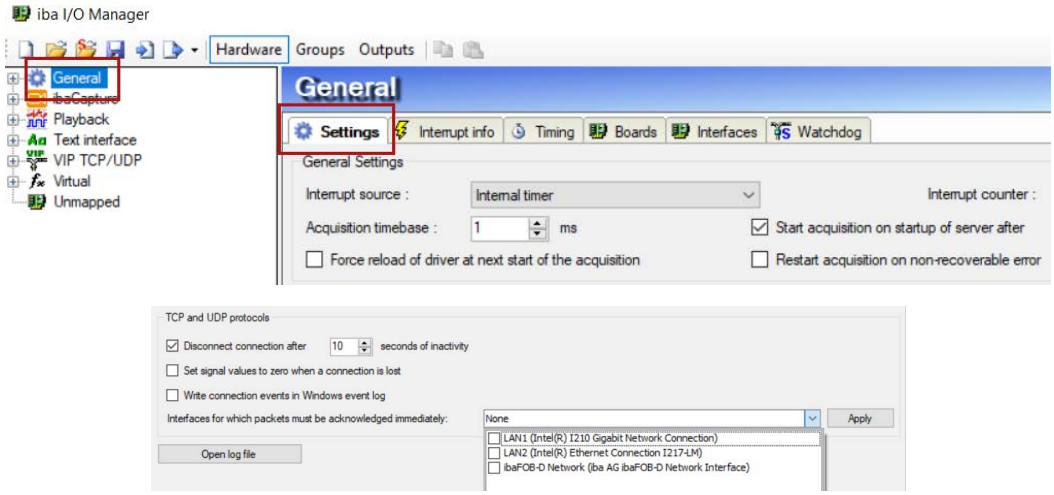
Within the VIP network only the *ibaPDA* and the controller used should be connected but no other participants in order to ensure a high dynamic of the system.

3.5 Configuration & Engineering ibaPDA

Subsequently, the engineering for *ibaPDA* is described. If all system requirements are met, 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 x 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.

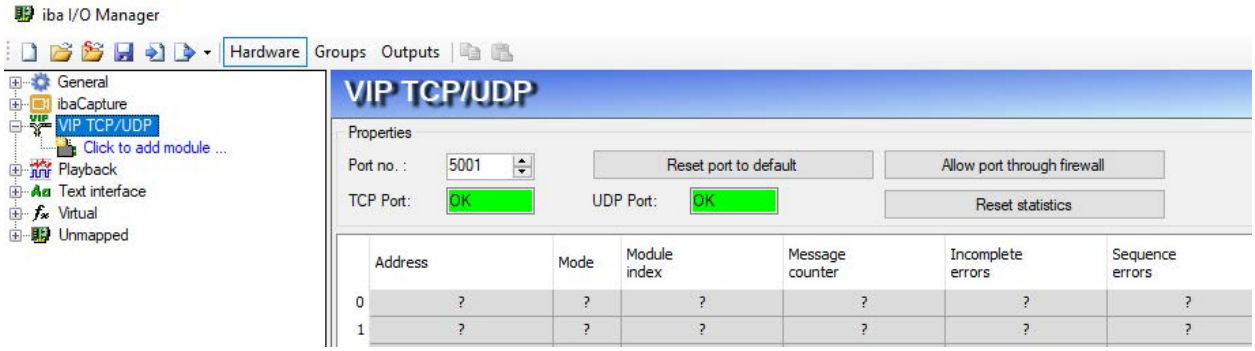
Interfaces for which packets must be acknowledged immediately
Selection of required interfaces.

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 provides the following functions and configuration options:



Port
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.

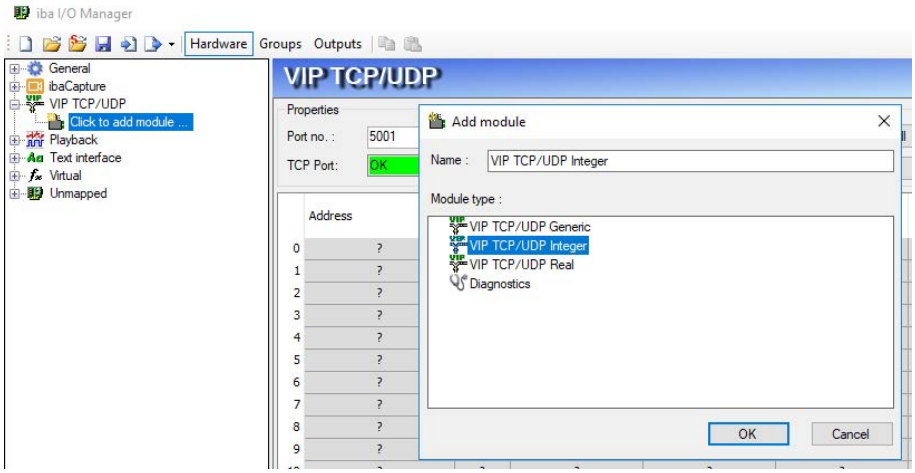
<Reset port to default>
The port number 5001 is set.

Allow port through firewall
When installing *ibaPDA*, the standard port numbers of the protocols used are automatically entered in the firewall. When the port number is changed here or when the interface was activated, subsequently it is necessary to allow this port in the firewall.

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 ↗ *Connection Check*, page 27

Adding a module
To add a module, click below the interface and select the desired module type.

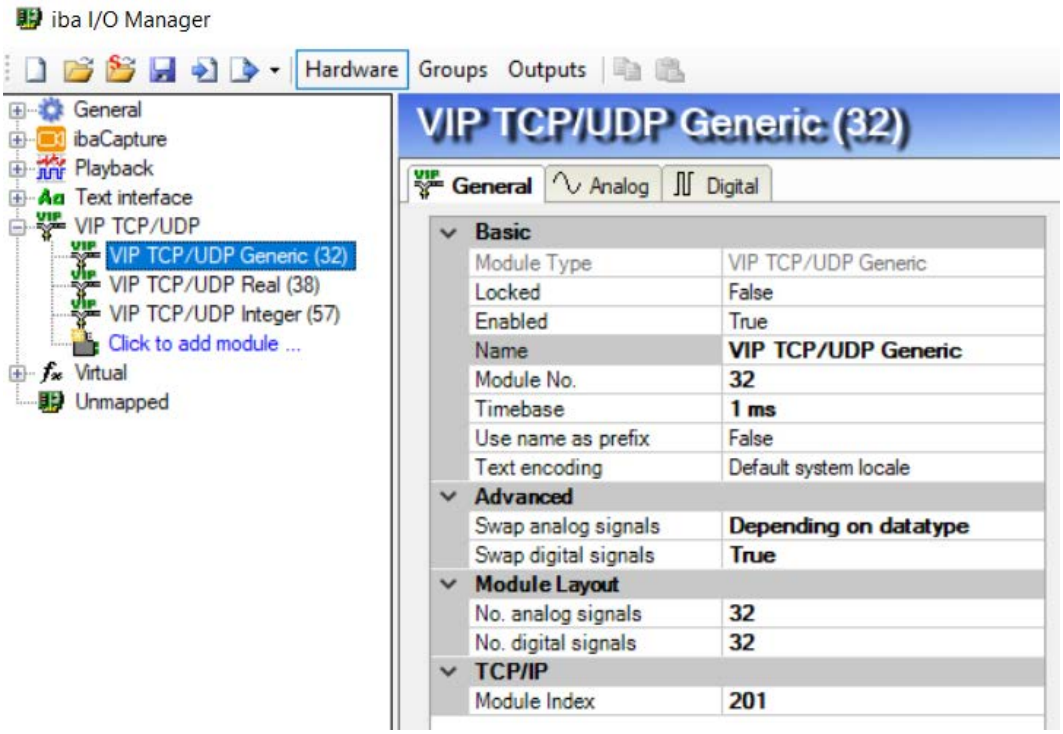


Tip

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.

3.5.3 General Module Settings

If you want to configure a module, mark the module in the tree structure and configure the following settings in the dialog:



Basic settings

Module Type (information only)
Indicates the type of the current module.

Locked
A module can be locked to avoid unintentional or unauthorized changing of the module settings.

Enabled
Disabled modules are excluded from signal acquisition.

Name
The plain text name should be entered here as the module designation.

Module No.
Internal reference number of the module. This number determines the order of the modules in the signal tree of *ibaPDA* client and *ibaAnalyzer*.

Timebase
All signals of the module will be sampled on this time base.

Use name as prefix
Puts the module name in front of the signal names.

Text encoding

Here you can specify the codepage which should be applied for interpreting the received text data.

Advanced

Swap analog signals
Possibility to change the evaluation order of the bytes

Swap digital signals
Possibility to change the evaluation order of the bytes

Module Layout

No. of analog/digital signals
You can set up the number of analog and digital signals to be measured. The maximum number of signals is 1000 (analog/digital signals).

TCP/IP

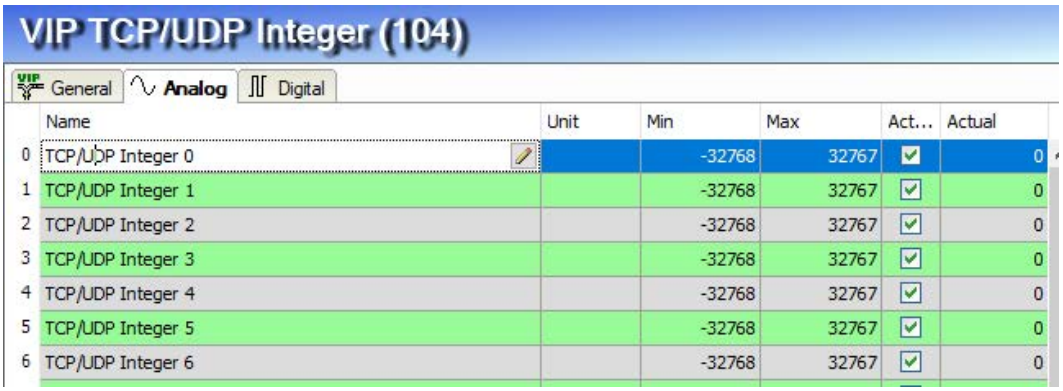
Module Index:
The module indices are created by a serial number 00....63 and an offset that corresponds to the module type and the license.
See ↗ *Communication between ibaPDA and ABB (Controller)*, page 11.

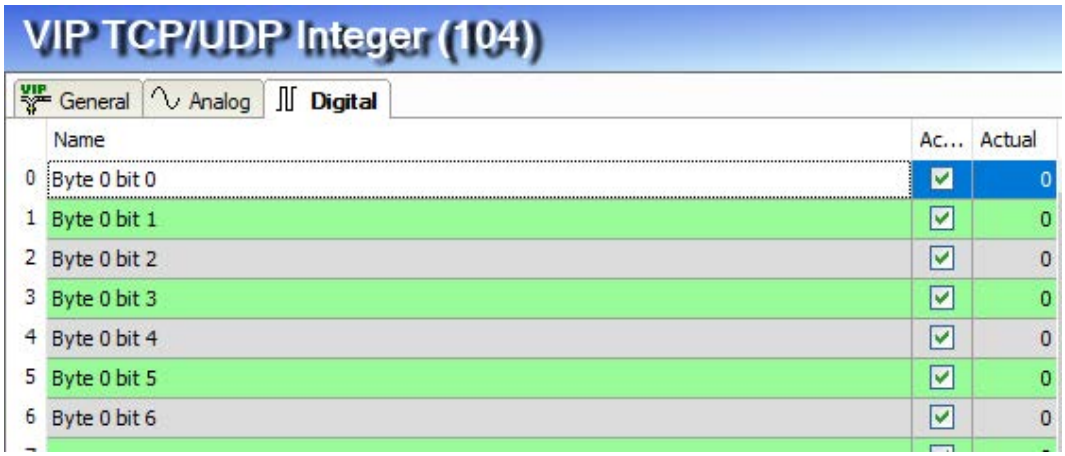
For a detailed description of the parameters, see the *ibaPDA* manual.

3.5.4 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*.

In the I/O manager, the signals can be given name and unit (only analog signals) and can be marked as active and inactive.





Tip

You can use the autofill function for the column (see *ibaPDA* manual or online help).

Other documentation

For a detailed description of additional options, see the *ibaPDA* manual.

3.5.5 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.

3.5.6 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).

3.5.7 Module Type Generic

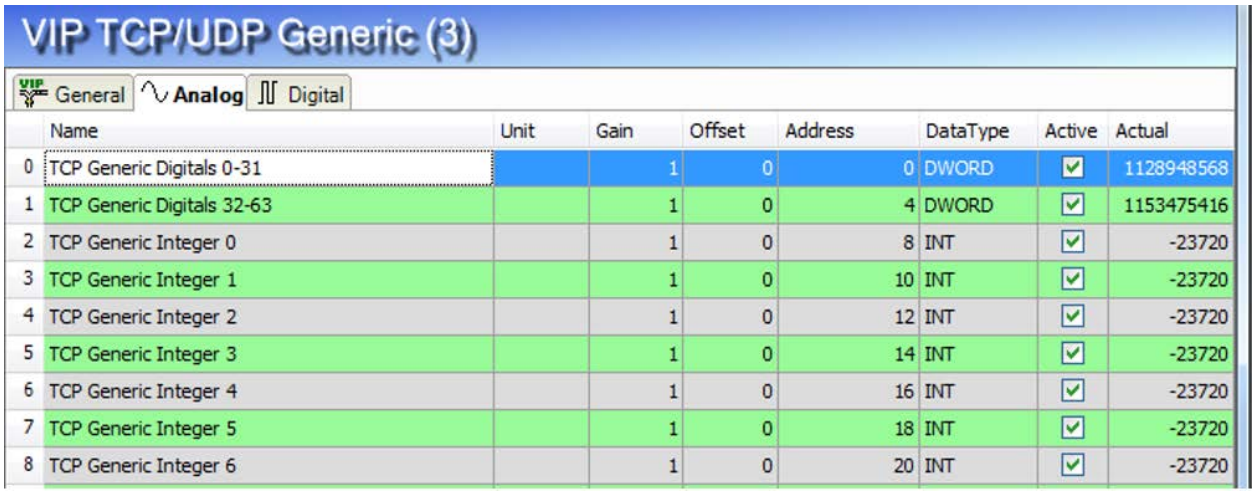
Any data block with max. length of 4096 bytes can be measured by means of the module *Generic*.

The following module settings are module-specific:

Number of analog signals
Maximum number of configurable analog signals. Various data types are supported for analog signals, incl. texts: SINT, BYTE, INT, WORD, DINT, DWORD, FLOAT, DOUBLE, STRING[32]

Number of digital signals
maximum number of configurable digital signals.

For signal configuration, enter the address, i.e. the offset in the telegram buffer and the data type for each variable. Bear in mind that counting starts from the beginning of user data without header.



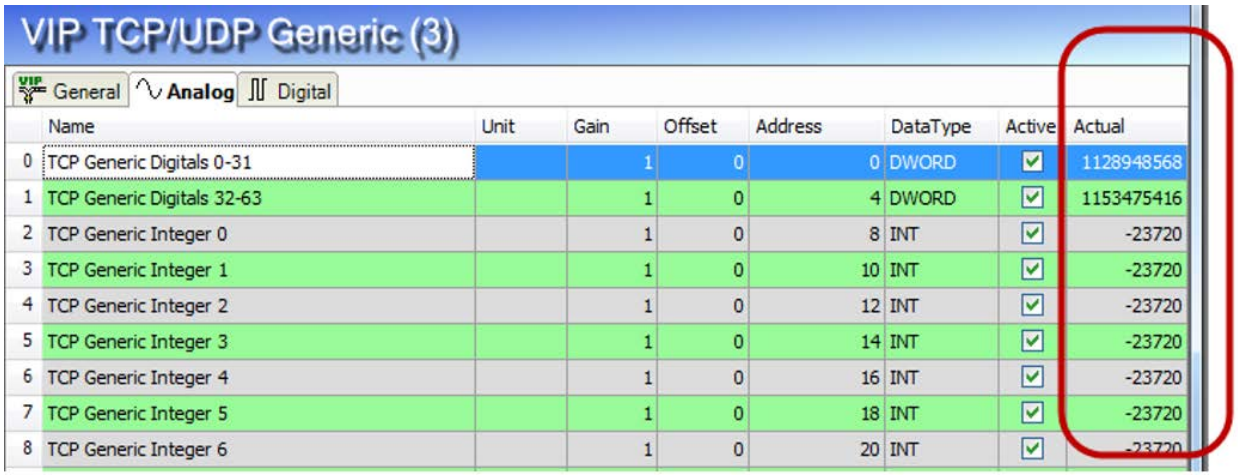
	Name	Unit	Gain	Offset	Address	DataType	Active	Actual
0	TCP Generic Digitals 0-31		1	0	0	DWORD	☑	1128948568
1	TCP Generic Digitals 32-63		1	0	4	DWORD	☑	1153475416
2	TCP Generic Integer 0		1	0	8	INT	☑	-23720
3	TCP Generic Integer 1		1	0	10	INT	☑	-23720
4	TCP Generic Integer 2		1	0	12	INT	☑	-23720
5	TCP 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

Note

The module *VIP TCP/UDP General* 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.

3.5.8 Module Diagnostics

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



	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

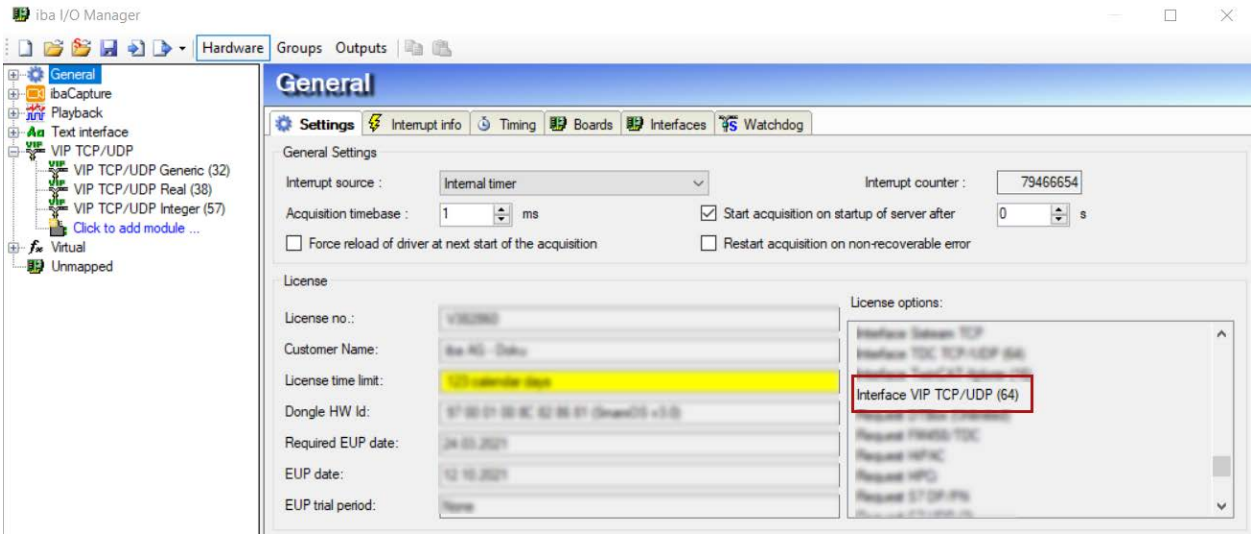
The following errors may occur:

- 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 19).
  - 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 19).
- The telegrams arrive not faster than ca. 200 ms with sequence error
  - Problem with "Delayed Acknowledge", see ↗ *TCP performance or data corruption problems resulting from the Delayed ACK mechanism*, page 28
  - Problem caused by "Nagle's Algorithm", see ↗ *TCP data corruption resulting from the Nagle's Algorithm*, page 30

4 Diagnostics

4.1 License Check

If the interface „VIP TCP/UDP" is not shown in the signal tree, you can check in the I/O manager under *General - Settings - License* or in the *ibaPDA* service status application whether your license has been correctly detected. The number of licensed connections is shown in brackets.



4.2 Interface Visibility

If the interface is not visible despite a valid license, it might be hidden. Click the register *Interfaces* and enable the "Interface VIP TCP/UDP".

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.

The log file can be opened 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 will find the log files in the program path of the *ibaPDA* server (...\Programs\iba\ibaPDA\Server\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)

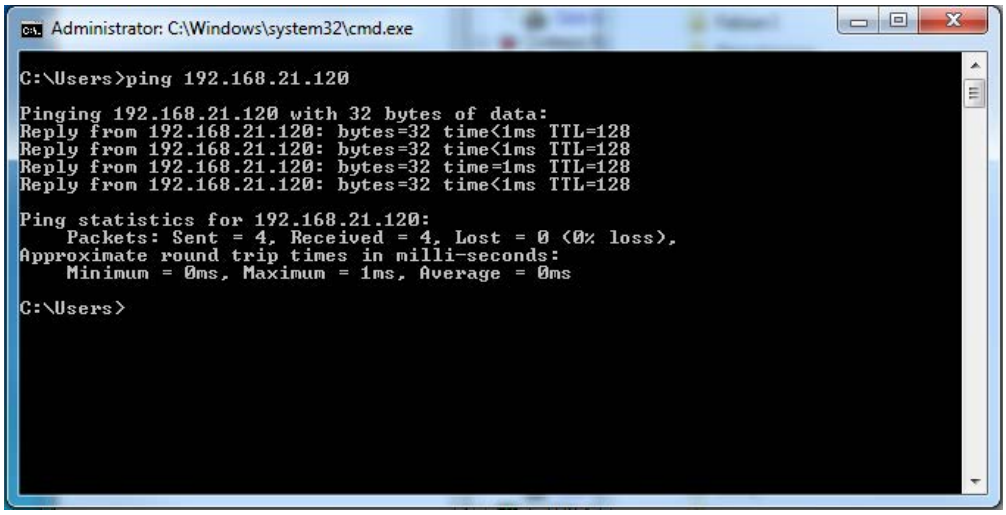
4.4 Connection diagnostics with PING

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

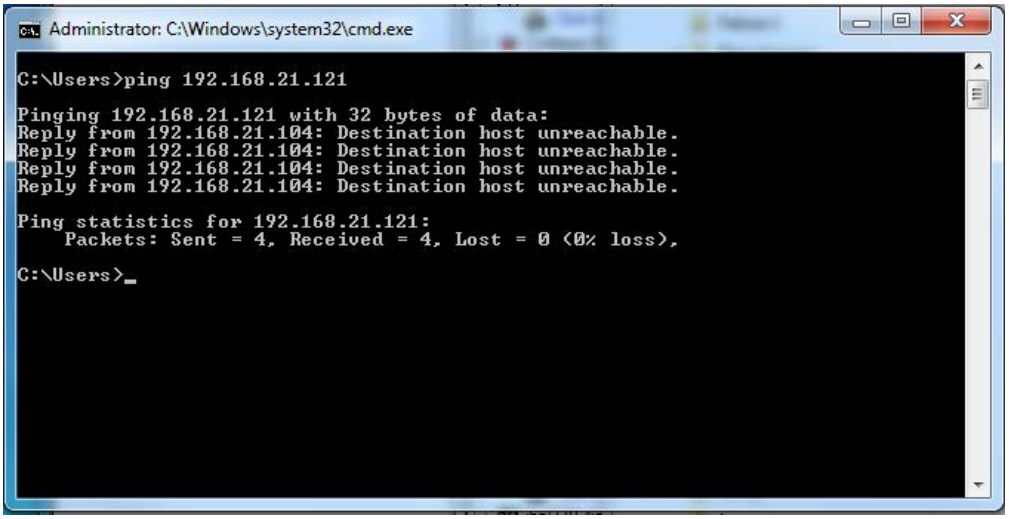
Open a Windows command prompt.

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

With an existing connection you receive several replies.

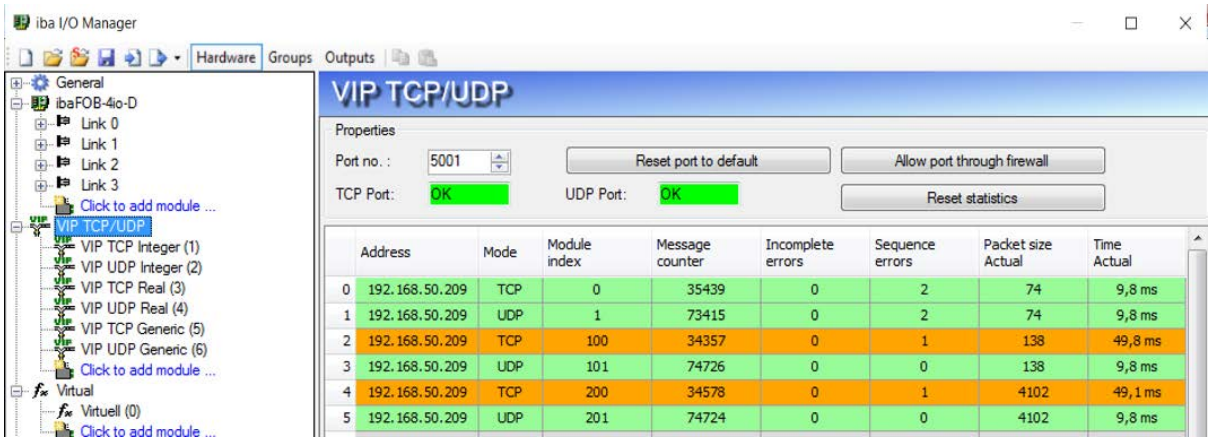


With no existing connection you receive error messages.



4.5 Connection Check

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



The background color of the lines has the following meaning:

Color	Meaning
Green:	The connection is OK. The *ibaPDA* module timebase is equally quick or slower than the telegram cycle. The current telegram cycle is shown in the column “Time Actual”.
Orange:	The connection is OK, but the telegram cycle is significantly slower than the *ibaPDA* 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 Ackowledge” problem occurs, see ↗ *TCP performance or data corruption problems resulting from the Delayed ACK mechanism*, page 28

5 Appendix

5.1 Troubleshooting

5.1.1 TCP performance or data corruption problems resulting from the Delayed ACK mechanism

Symptoms:

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).

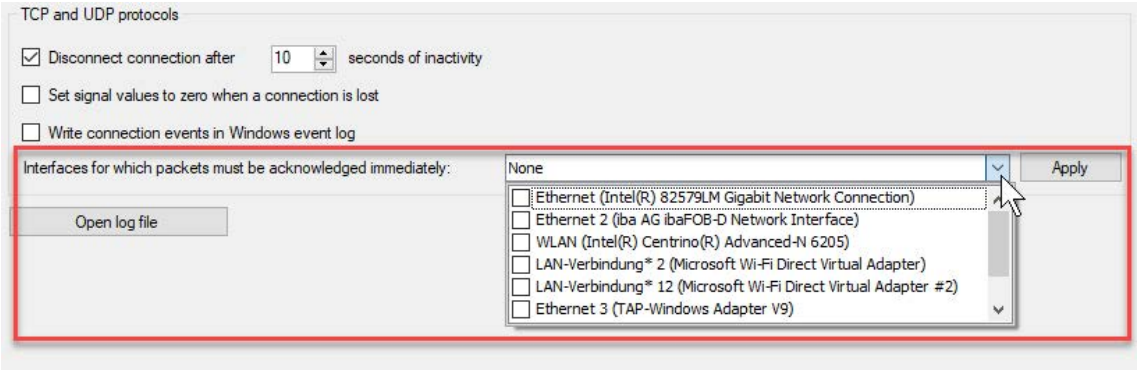
The data flow is controlled by a "sliding window" (parameter Win=nnnn). The recipient specifies how many bytes it can receive without sending an acknowledgment.

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 old one was received correctly.

Remedy:

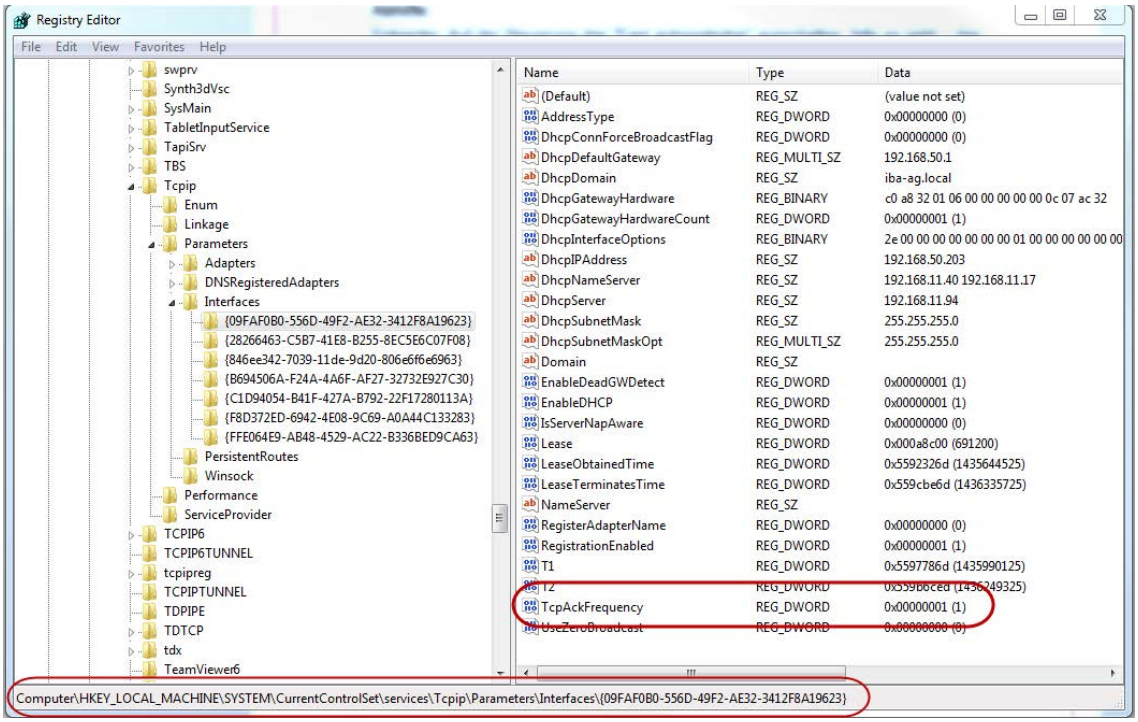
The "delayed acknowledge" can be switched off individually for each network adapter via an entry in the Windows Registry. For easy modification, *ibaPDA* offers a corresponding dialog in the I/O manager under *General* in the tab *Settings*.

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*.

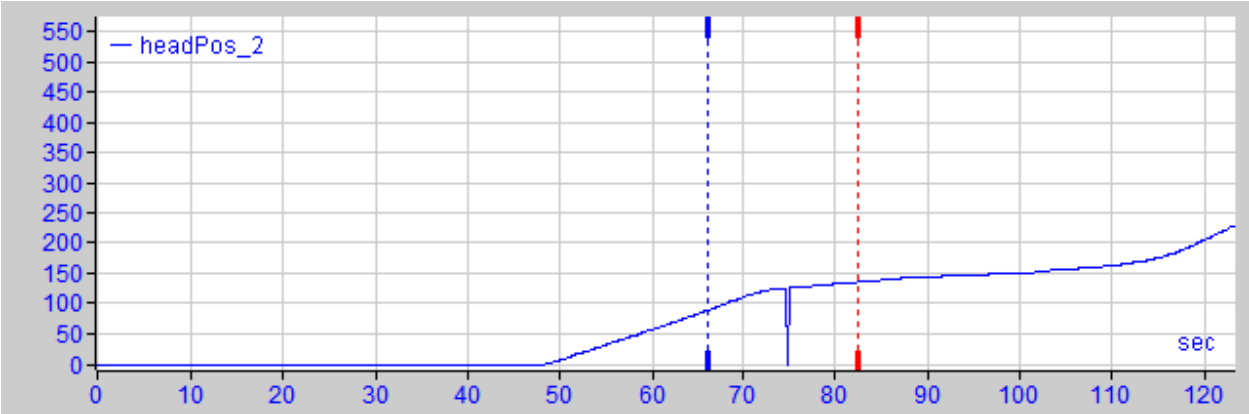
5.1.2 TCP data corruption resulting from the Nagle's Algorithm

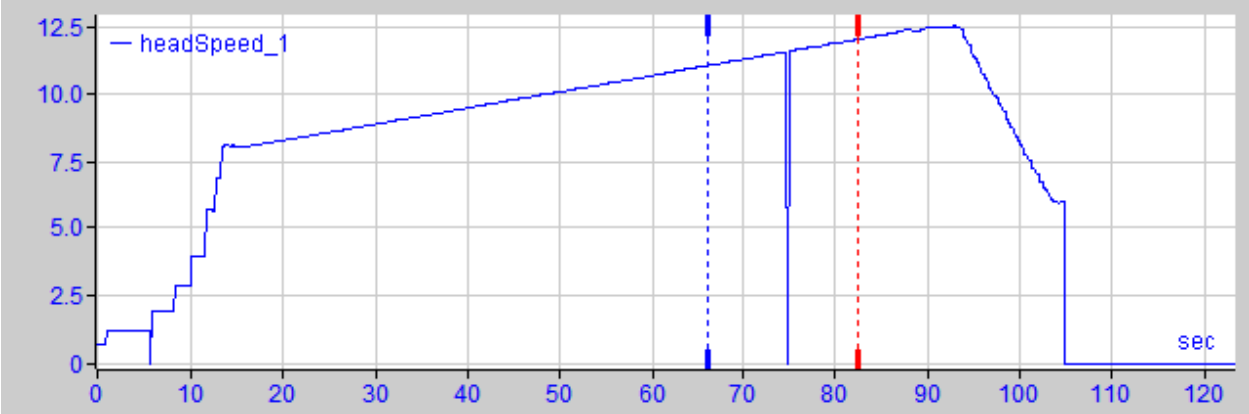
Symptoms:

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, named after its creator John Nagle, 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.

Since 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 28) 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.

Nagle's algorithm and Delayed ACKs together create a problem because Delayed ACKs are waiting around to send the ACK while "Nagle's" is waiting around to receive the ACK!

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

Remedy:

We recommend starting with disabling the *Delayed ACK* mechanism as explained in chapter 5.1.1, page 28. In a typical real-time application, the transmitter will then send the new data to *ibaPDA* with a certain cycle time, since 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

Fax: +49 911 97282-33

Email: support@iba-ag.com

Note

If you need support for software products, please state the license number or the CodeMeter container number (WIBU dongle). 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

Fax: +49 911 97282-33

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.