# ibaPDA-Interface-Generic-TCP

## Data Interface Generic TCP

Manual

Issue 2.2

Measurement Systems for Industry and Energy

www.iba-ag.com

**Manufacturer**

iba AG

Gebhardtstrasse 10-20

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 |




| Version | Date | Revision | Author | Version SW |
|---|---|---|---|---|
| 2.2 | 05-2026 | Configurable message size in modules added | rm, mm | 8.13.0 |

# Contents

# 1 About this documentation

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

**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 they are capable of assessing safety and recognizing possible consequences and risks on the basis of their 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-Generic-TCP* 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*<br>Example:<br>Select the menu *Logic diagram – Add – New function block*. |
| Keys | <Key name><br>Example: <Alt>; <F1> |
| Press the keys simultaneously | <Key name> + <Key name><br>Example: <Alt> + <Ctrl> |
| Buttons | <Key name><br>Example: <OK>; <Cancel> |
| Filenames, paths | `Filename`, `Path`<br>Example: `Test.docx` |

## 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 *Generic TCP* data interface:

- *ibaPDA* v8.7.0 or higher
- Base license for *ibaPDA* + License for *ibaPDA-Interface-Generic-TCP*
- Network connection 10/100 Mbit/s

For further requirements for the used computer hardware and the supported operating systems, refer to 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**

Depending on the device configuration and controller types, network-specific settings may cause problems when handling TCP/IP acknowledgments.
See ↗ *Troubleshooting, page 31* (all *ibaPDA* versions).

**Note**

Generic TCP vs. Generic UDP

In modern networks, data can be transferred quickly enough for many applications using the TCP/IP protocol. However, in case of highly cyclical data traffic, such as in measurement data acquisition, the TCP/IP protocol is sometimes unable to meet the requirements ideally due to its principle of operation (handshake, TCP/IP acknowledge).

Users should therefore consider whether the simpler UDP protocol is sufficient or even the better choice for the planned application. The use of the UDP protocol often also has a positive effect on the performance of the sender (e.g. PLC).

iba AG recommends using the protocols as follows:

- TCP/IP (*ibaPDA-Interface-Generic-TCP*): for noncyclical or triggered data transmission or for cyclical data transmission on timebases higher than 1 s
- UDP (*ibaPDA-Interface-Generic-UDP*): for cyclical data transmission on timebases significantly shorter than 1 s

**License information**

| Order no. | Product name | Description |
|---|---|---|
| 31.001076 | ibaPDA-Interface-Generic-TCP | Extension license for an *ibaPDA* system adding the Generic-TCP interface with 64 connections |
| 31.101076 | one-step-up-Interface-Generic-TCP | Extension license for 64 additional Generic TCP connections (max. 3 extension licenses) |

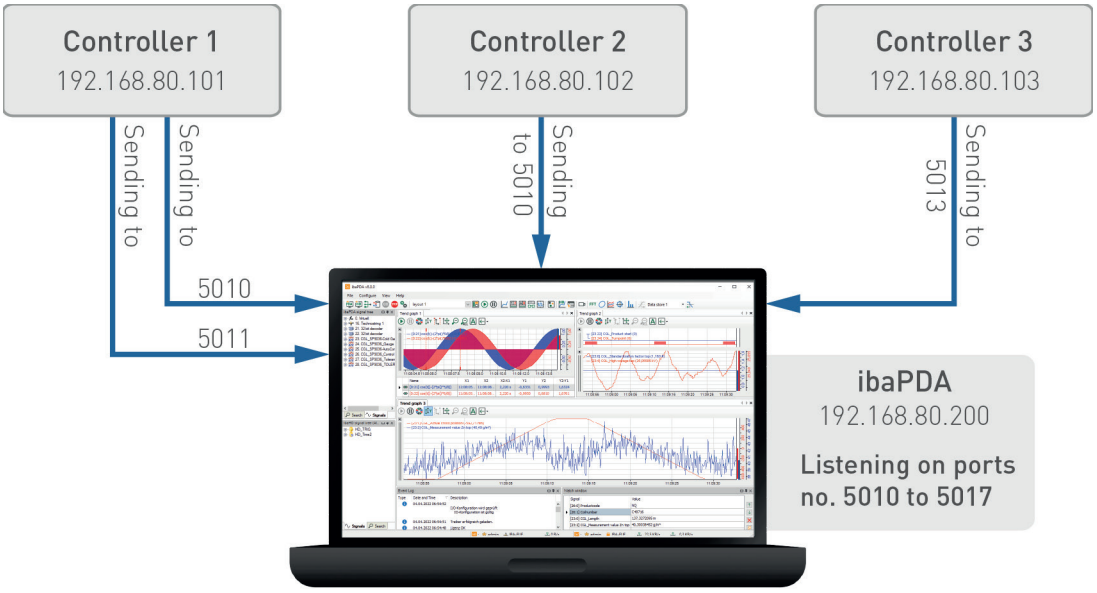
# 3 Data interface Generic TCP

*ibaPDA-Interface-Generic-TCP* can be used by any controller capable of sending TCP/IP messages.

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite. IP is responsible, on a low level, for sending messages via the Internet. TCP works on a higher level and establishes the connection between two end systems. TCP is responsible for sending a data stream in a reliable and organized way from one program in a computer to another program on a second computer.

The Generic TCP messages are IP Unicast messages that one or more controllers send to the *ibaPDA* system using a specified port number.

The following drawing gives an overview of a possible configuration where 3 controllers are sending TCP/IP messages to one *ibaPDA* system.




**Properties**

- The messages sent over each connection can have any data structure.
- You can define a port range on the Generic TCP interface in the *ibaPDA* I/O Manager. In the example above, the *ibaPDA* driver is listening on port 5010 to port 5017 for a connection.
- Each TCP connection is uniquely identified within *ibaPDA* by the destination port number and the source IP address.

Thus, *ibaPDA* can receive data from different controllers, which use the same destination port. It is also possible to send messages from one controller to *ibaPDA* over different ports.

This is shown clearly in the above example: Controllers 1 and 2 use the same port 5010 but have different IP addresses. Controller 1 sends several messages and uses different ports for sending (5010 and 5011).

**ibaPDA specific limitations**

- The number of the supported connections in *ibaPDA* depends on the Generic TCP license (64, 128, 192 or 256).

The following controllers apply:

- Any system capable of sending TCP/IP messages

**Other documentation**

Reference to additional documentation or further literature.

- *ibaPDA* manual
- TCP/IP Tutorial, RFC1180 (https://www.rfc-editor.org/rfc/rfc1180)
- Transmission Control Protocol, RFC793 (https://www.rfc-editor.org/rfc/rfc793)

## 3.1 Configuration of the controller

*ibaPDA* supports two connection modes.

- *ibaPDA* is the TCP server (Module settings, Mode = passive):

  For the controller that is intended to send data to *ibaPDA* a TCP/IP connection has to be set up. The destination address is the IP address of the *ibaPDA* server. The destination port should be declared within the defined port range in *ibaPDA*. The default destination port range defined in *ibaPDA* is 5010 – 5017 (see ↗ *General interface settings, page 12*).

  Each TCP/IP connection will be uniquely identified on the *ibaPDA* side by the "Destination port number" and the "Source IP address". This means that one controller can send several messages to the same *ibaPDA* even using a destination port already used by other controllers.

- *ibaPDA* is TCP client (Module settings, Mode = active):

  The controller that is intended to send data to *ibaPDA* has to wait on a specified port for the connection set-up by *ibaPDA*. In *ibaPDA* the user has to enter under the module settings as source address the IP address of the controller and as port number the port, on which the controller is waiting for the establishment of a connection. As soon as the connection has been established, the controller can send the data.

The maximum supported length of the TCP data is 16384 Bytes. *ibaPDA* rejects longer messages.

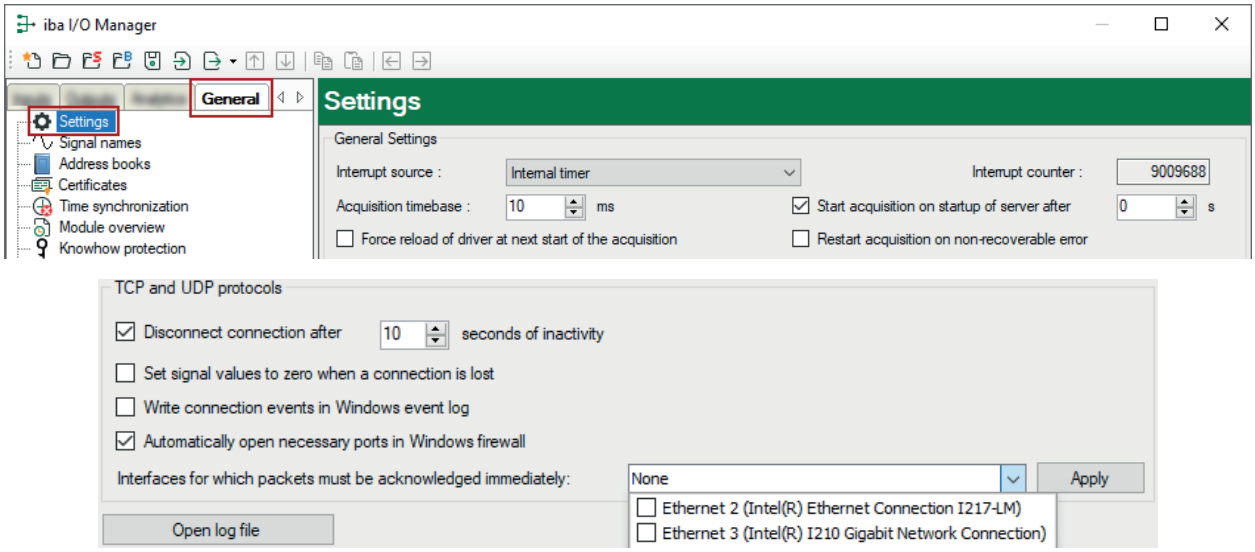
The controller should cyclically send a message with a fixed layout to the *ibaPDA* system. If more than one message with a different content is needed, another destination port within the defined range has to be used.

## 3.2 Configuration and engineering ibaPDA

The engineering for *ibaPDA* is described in the following. If all system requirements are fulfilled, *ibaPDA* displays the *Generic TCP* interface in the interface tree of the I/O Manager.

### 3.2.1 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>.

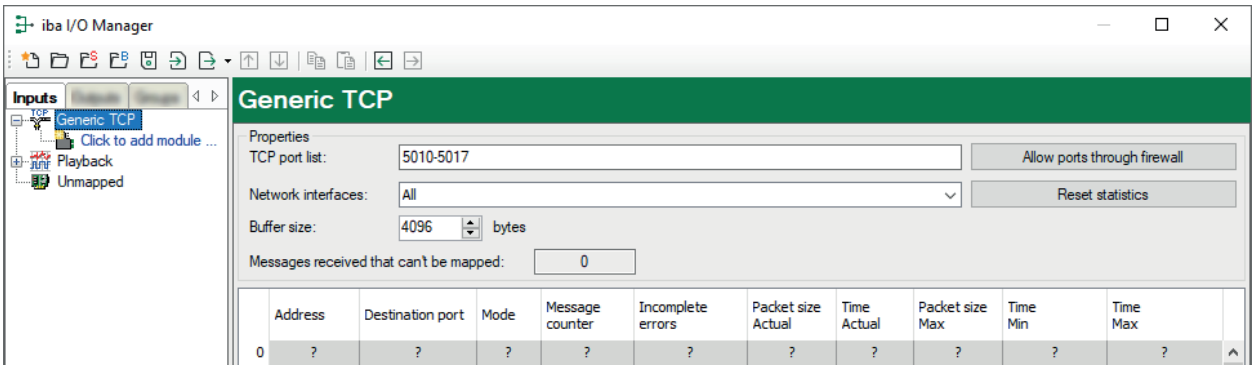
**Interfaces for which packets must be acknowledged immediately**
Selection of required interfaces.

**Note**

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

### 3.2.2 General interface settings

The interface provides the following features and configuration options.




**TCP port list**
Ports, on which *ibaPDA* waits for incoming connection set-ups.
You can enter the ports as port range, as list of individual ports or as a combination of both. Enter a range with a hyphen. Separate non-consecutive port numbers with commas. The default setting is the range 5010 – 5017.
The port number in the controller must be identical (see ↗ *Configuration of the controller, page 10*).

**Note**

The list is only valid for the "passive" mode. It contains the port numbers on which *ibaPDA* is waiting for a connection set-up by the "active" partner. For active connections, *ibaPDA* can use port numbers that cannot be found in this list. These port numbers need to be enabled in the firewall of the connection partner.

**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.

**Buffer size**
The maximum data size of a connection is configurable. It should be between 1024 and 16384 bytes. Default setting: 4096 bytes

**<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.

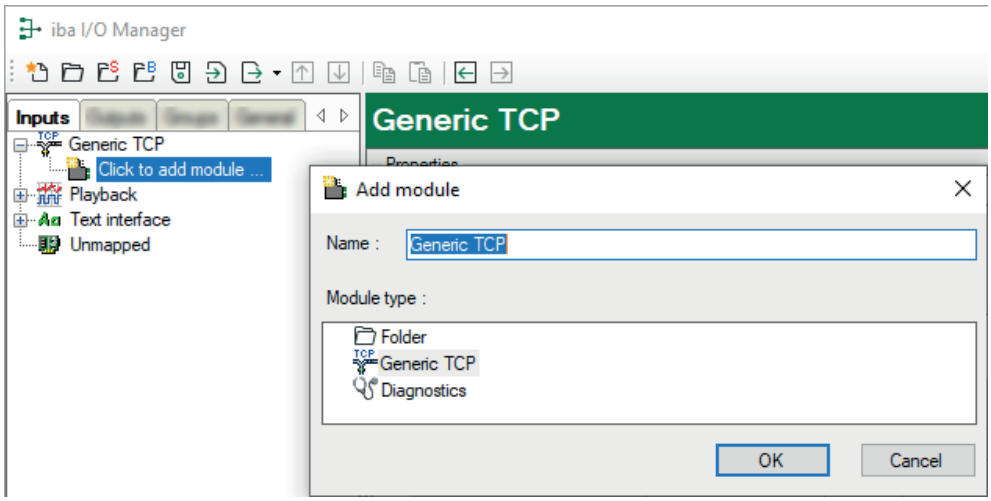
**<Reset statistics>**
Click this button to reset the calculated times and error counters in the table to 0.

**Message counter**
see ↗ *Checking the connection for received messages, page 24*.

**Connection table**
see ↗ *Checking the connection for received messages, page 24*.

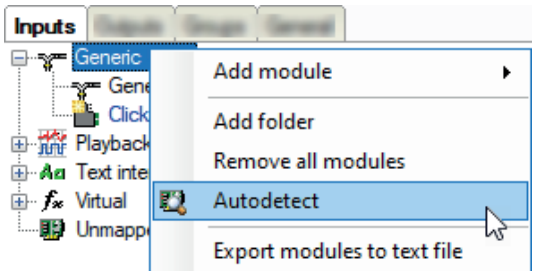
### 3.2.3 Adding a module

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>.




**Tip**

Cyclic TCP/IP messages that have their destination port in the TCP port range of the interface are displayed automatically in the connection table, even if no corresponding modules have been defined and *ibaPDA* was not started before. You can create modules for these links easily with a right click on the *Generic TCP* interface icon in the tree view and select *Autodetect* in the context menu.




By applying *Autodetect*, a Generic TCP module with 32 analog (data type FLOAT) and 32 digital signals is created for each link.

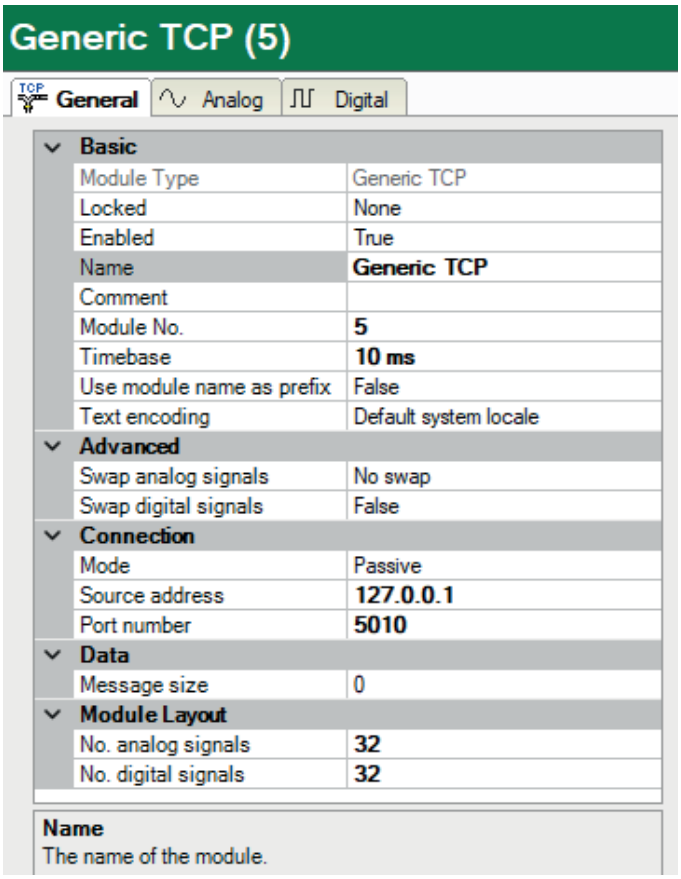
**Note**

The autodetect functionality only works for passive TCP connections.

### 3.2.4 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.

**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.

### 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.

### Connection

**Mode**
Here you select, which partner establishes the TCP/IP connection:

- *Passive* mode (standard): *ibaPDA* waits on the selected port for a connection set-up by the partner (Controller).
- *Active* mode: *ibaPDA* establishes a connection; the partner has to be configured as passive and has to wait for the connection set-up on the selected port.

"Client/Server" are other designations for the connection mode:
Client corresponds to the active mode, Server to the passive mode.

**Source address, Port number**

- In the *passive* mode, each connection to a specific controller is identified by the IP address and the port number. The port has to be within the port range defined for the interface and has to be enabled for the firewall (see ↗ *General interface settings, page 12*).
- In the *active* mode, the IP address and port number are needed for the connection set-up to the sending, but passive partner. The port can be outside the TCP port range and has to be enabled for the firewall of the partner.

### Data

**Messages size**
Defines the expected size of the TCP messages in bytes. Set the message size that the sender provides here, to ensure that all messages are reliably detected and processed. With the default setting 0, the message boundary is detected based on the PSH bit in the TCP header, as before.

### Module Layout

**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 16384. The signal tables are adjusted accordingly. Note that the total amount of data must not exceed 16384 bytes for analog and digital signals together.

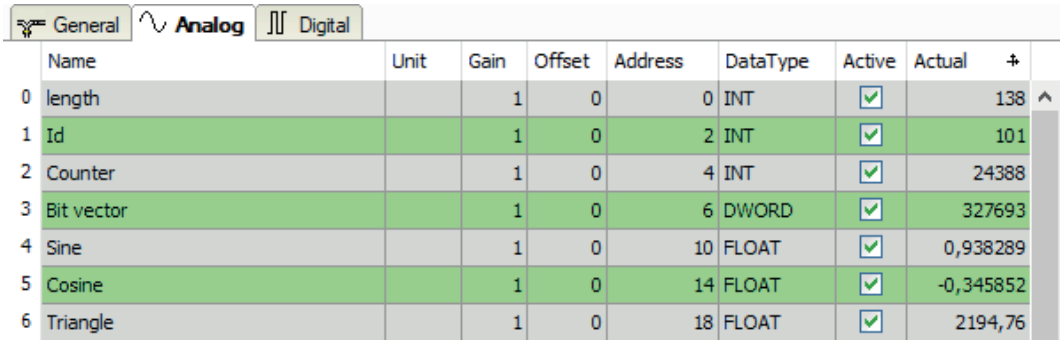
**Other documentation**

You can find further information in the *ibaPDA* manual.

### 3.2.5 Configuration of input signals

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

**Analog and Digital tab**




| | Name | Unit | Gain | Offset | Address | DataType | Active | Actual |
|---|---|---|---|---|---|---|---|---|
| 0 | length | | 1 | 0 | 0 | INT | ☑ | 138 |
| 1 | Id | | 1 | 0 | 2 | INT | ☑ | 101 |
| 2 | Counter | | 1 | 0 | 4 | INT | ☑ | 24388 |
| 3 | Bit vector | | 1 | 0 | 6 | DWORD | ☑ | 327693 |
| 4 | Sine | | 1 | 0 | 10 | FLOAT | ☑ | 0,938289 |
| 5 | Cosine | | 1 | 0 | 14 | FLOAT | ☑ | -0,345852 |
| 6 | Triangle | | 1 | 0 | 18 | FLOAT | ☑ | 2194,76 |

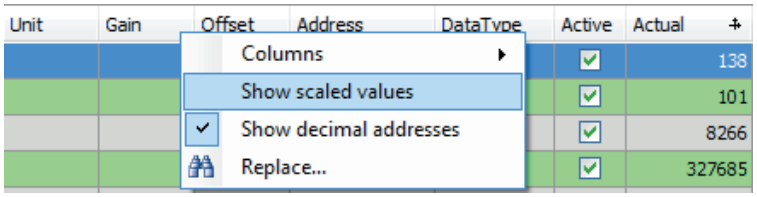
You can assign name, unit, address and data type or bit number to the analog and digital signals. Moreover, you can enable or disable the signals.

**Other documentation**

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

Specific columns for Generic TCP modules:

**Address**
The address indicates the offset of the first byte of this value within the user data telegram. You can enter the offset as hexadecimal or decimal value by selecting the desired setting in the context menu.




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

**DataType (analog signals only)**
*ibaPDA* supports the following data types: BYTE, WORD, DWORD, SINT, INT, DINT, LINT, FLOAT, DOUBLE, S5 FLOAT and STRING[32].

The address range depends on the data type. Hence, after changing the data type, you possibly have to adjust the address entries.

**Note**

The module *Generic TCP* 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 or to split it up into several text signals, use the *Text splitter* module on the *Analytics* tab.

**Tip**

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

### 3.2.6 Module diagnostics

The tables *Analog* and *Digital* of the Generic TCP modules show the telegram contents (actual values).

General | Analog | Digital

| | Name | Unit | Gain | Offset | Address | DataType | Active | Actual |
|---|---|---|---|---|---|---|---|---|
| 0 | length | | 1 | 0 | 0 | INT | ☑ | 138 |
| 1 | Id | | 1 | 0 | 2 | INT | ☑ | 101 |
| 2 | Counter | | 1 | 0 | 4 | INT | ☑ | 24388 |
| 3 | Bit vector | | 1 | 0 | 6 | DWORD | ☑ | 327693 |
| 4 | Sine | | 1 | 0 | 10 | FLOAT | ☑ | 0,938289 |
| 5 | Cosine | | 1 | 0 | 14 | FLOAT | ☑ | -0,345852 |
| 6 | Triangle | | 1 | 0 | 18 | FLOAT | ☑ | 2194,76 |

The following errors may occur:

- No data are displayed:
  - The telegram buffer on the sender side is not filled correctly.
  - The connectors of the send block are connected incorrectly.
- Incorrect values are displayed:
  - The telegram buffer on the controller side is not filled correctly (offset error).
  - The byte order is set incorrectly, see ↗ *General module settings, page 14*.
- The digital signals are sorted incorrectly:
  - The byte order is set incorrectly, see ↗ *General module settings, page 14*.
- 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 31*.
  - Problem caused by "Nagle's Algorithm", see ↗ *TCP data corruption resulting from the Nagle’s Algorithm, page 33*.

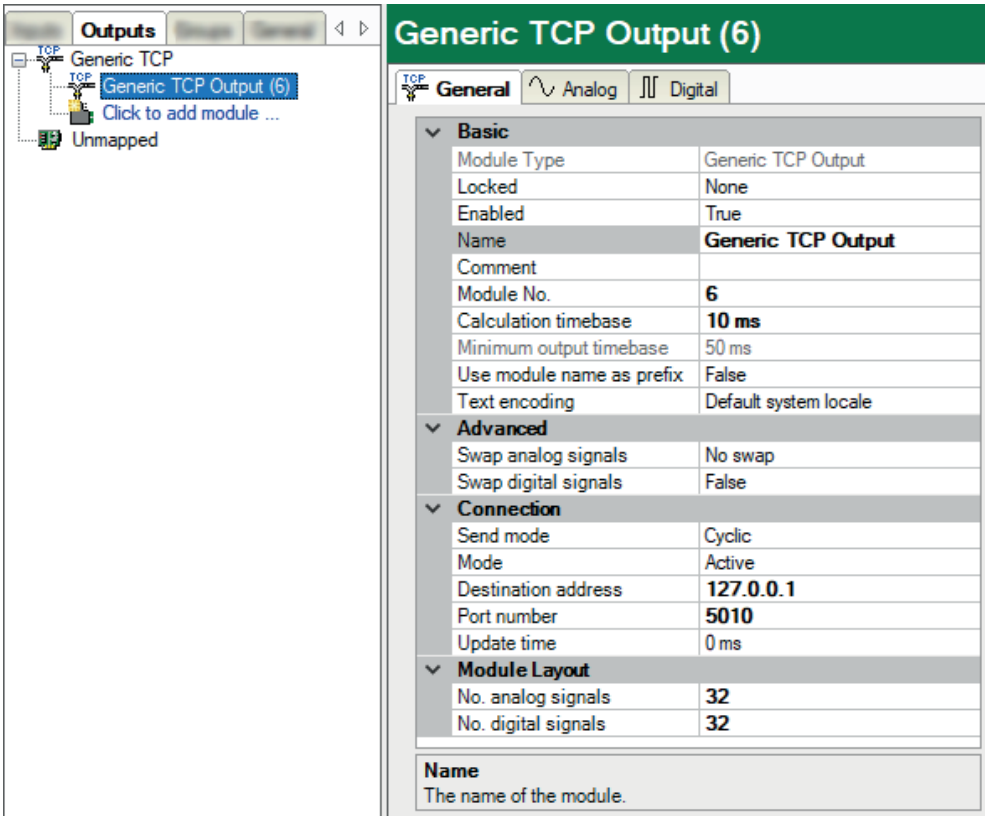
## 3.3 Configuration of the ibaPDA output modules

If all system requirements are met, *ibaPDA* offers the *Generic TCP* interface in the interface tree of the *Outputs* tab. There is no need to add the interface manually.

Add the output modules in the same way as input modules.

### 3.3.1 General module settings ibaPDA output modules

If you want to configure an output module, mark the module in the tree structure of the *Outputs* tab.




The parameters are almost identical to those of the input module ↗ *General module settings, page 14*.

Consider the following differences to the settings of the input modules:

**Calculation timebase**
Timebase (in ms) used for the calculation of the output values.

The calculation timebase is not the same as the output timebase with which the values are output!

**Minimum output timebase**
Timebase with which the outputs can be updated as quickly as possible.

The value is acquired automatically by the system based on the current I/O configuration and is only displayed here. The output timebase results from the smallest common multiple of all module timebases or is at least 50 ms.

**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.

The length of the telegram is determined by the number of the signals and the assigned data types. For disabled signals, *ibaPDA* writes 0 in the telegram buffer.

**Mode**

- *Active* mode (Standard): *ibaPDA* establishes the connection to the receiver. The port number has to be enabled in the firewall on side of the receiver.
- *Passive* mode: The receiver establishes the connection. The port number must not be contained in the port range for the input modules (see ↗ *General interface settings, page 12*). There is no field for the IP address of the receiver.

The following parameters determine the telegram send cycle:

**Update time**
If you enter 0 or a value smaller than 50, the send cycle corresponds to the *ibaPDA* task cycle or the calculation timebase respectively (at least 50 ms).

**Send mode**
Defines when *ibaPDA* sends new messages. The update time always determines the fastest send rate.

- Cyclic: *ibaPDA* sends a message every update time.
- On change: *ibaPDA* sends a message each time the signal data changes.
- On trigger: *ibaPDA* sends a message each time a rising edge is detected on the trigger signal.

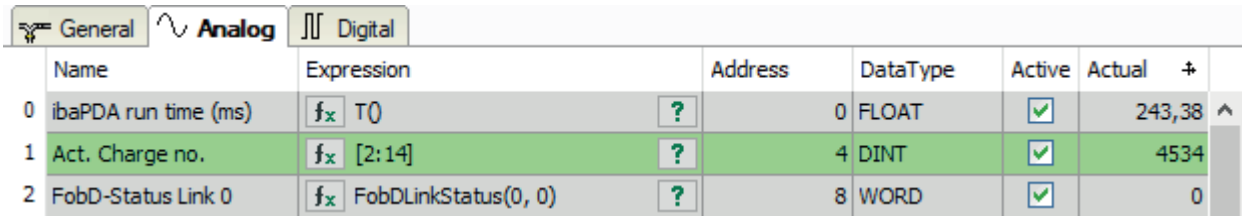
### 3.3.2 Configuration of output signals

In the *Analog* and *Digital* tab, select the signals that you want to send in a message.

**Tip**

If you define the output data in a virtual module and only enter the references to these data here, you can also include these data in the data recording as an option.

**Analog and Digital tab**




**Other documentation**

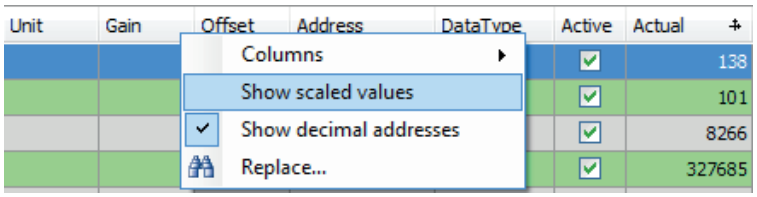
For a description of the columns, please see the *ibaPDA* manual.

**Expression**

In the *Expression* column, define the output signals in a similar way as the virtual signals. You can enter simple expressions or references to existing signals directly in the tables. You can also open the Expression editor via the <fx> button. You can analyze an incorrect expression via the <?> button.

**Address**

The address indicates the offset of the first byte of this value within the sent message. You can enter the offset as hexadecimal or decimal value by selecting the desired setting in the context menu.




The digital signals are addressed via the *Address* and *Bit no.* (0 – 15) columns.

**DataType**

*ibaPDA* supports the following data types: BYTE, WORD, DWORD, SINT, INT, DINT, FLOAT, DOUBLE, and STRING[32].

The address range depends on the data type. Hence, after changing the data type, you might have to adjust the address entries.

**Active**

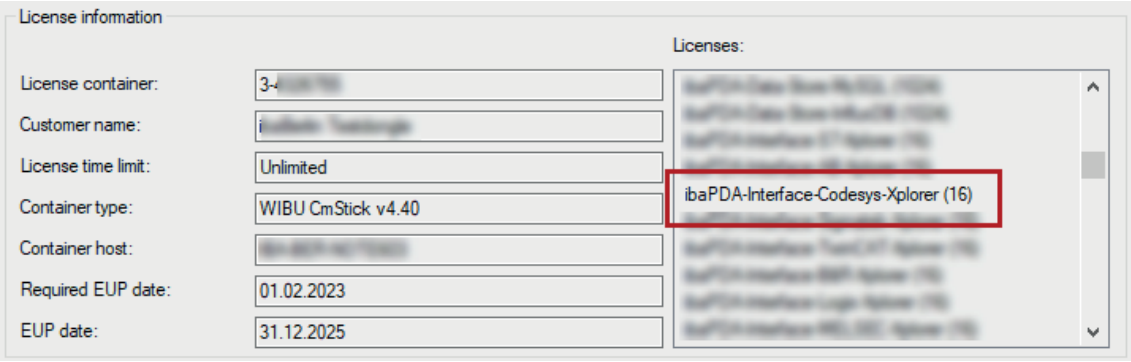
For disabled signals, *ibaPDA* writes 0 in the telegram buffer.

# 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-Generic-TCP* 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.

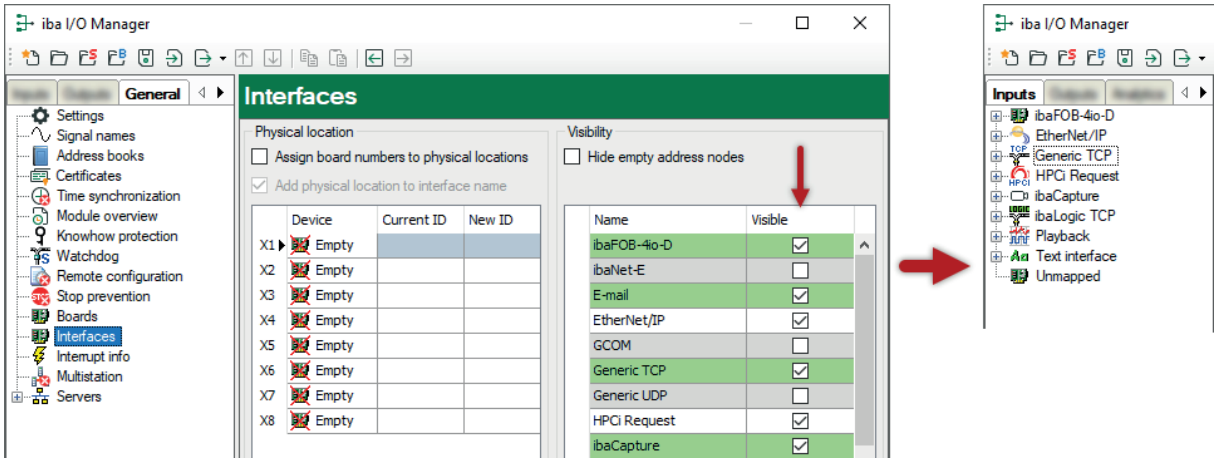
### Visibility

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 systems 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)

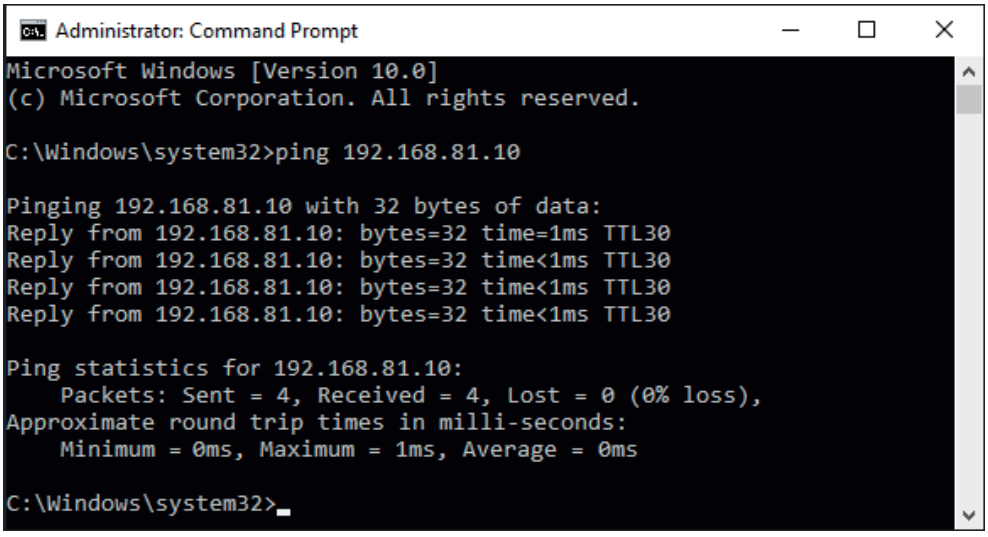
## 4.4 Connection diagnostics with PING

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

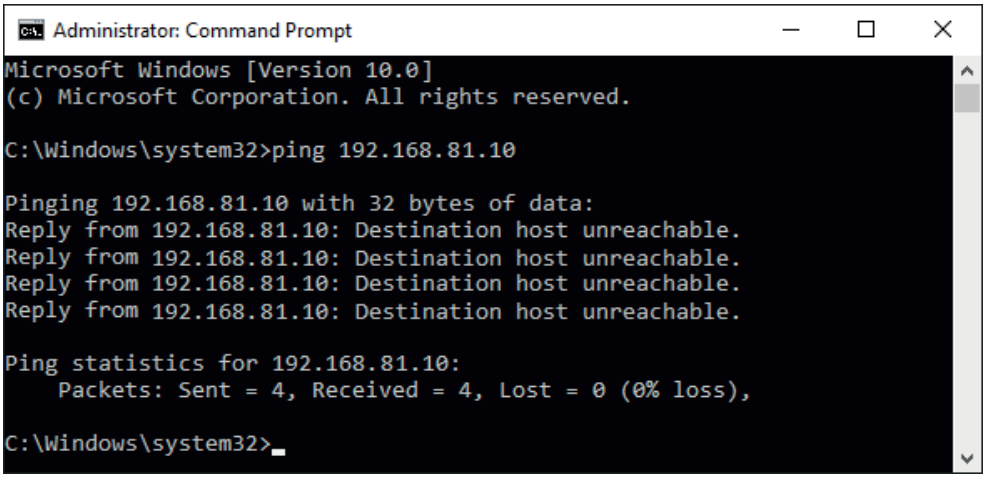
1. Open a Windows command prompt.


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.

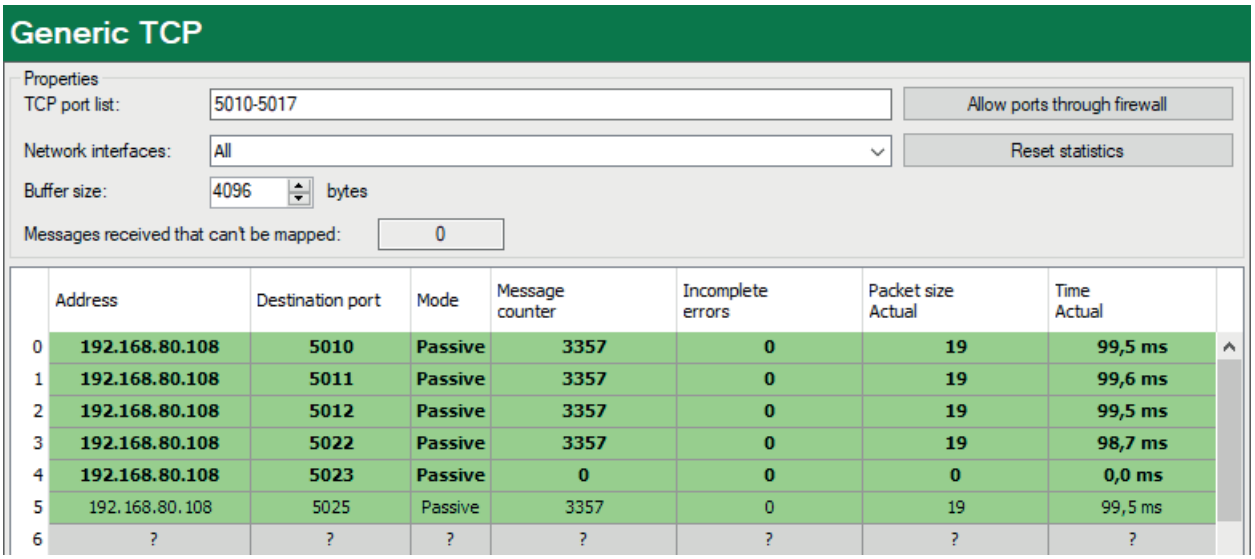



→ With no existing connection you receive error messages.



## 4.5 Checking the connection for received messages

After the configuration has been accepted, the following data are displayed:




| | Address | Destination port | Mode | Message counter | Incomplete errors | Packet size Actual | Time Actual |
|---|---|---|---|---|---|---|---|
| 0 | **192.168.80.108** | **5010** | **Passive** | **3357** | **0** | **19** | **99,5 ms** |
| 1 | **192.168.80.108** | **5011** | **Passive** | **3357** | **0** | **19** | **99,6 ms** |
| 2 | **192.168.80.108** | **5012** | **Passive** | **3357** | **0** | **19** | **99,5 ms** |
| 3 | **192.168.80.108** | **5022** | **Passive** | **3357** | **0** | **19** | **98,7 ms** |
| 4 | **192.168.80.108** | **5023** | **Passive** | **0** | **0** | **0** | **0,0 ms** |
| 5 | 192.168.80.108 | 5025 | Passive | 3357 | 0 | 19 | 99,5 ms |
| 6 | ? | ? | ? | ? | ? | ? | ? |

**Message counter**

**Messages received that cannot be mapped**
When receiving messages of a new connection, this value is incremented in case all available connection entries are already in use.
Cause: *ibaPDA* receives more connections than the license supports

**Connection table**

**Display variants**

| | |
|---|---|
| Green background/ bold text | The connection is ok and there is a module assigned to it. |
| Green background/ standard text | The connection is ok, but there is no module assigned to it. |
| Gray background/ bold text | A module is defined, but no connection has been established with these parameters. |

**Connection data**

- Source IP address
- Destination port
- Connection mode
- Receive message counter
- Data length of the received telegrams

**Possible errors**

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

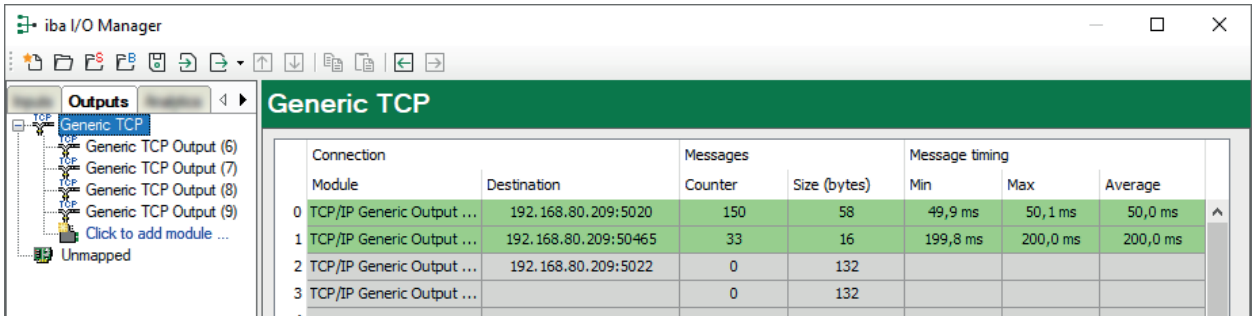
- The controller is not sending.
- No Ethernet connection between *ibaPDA* PC and the controller
- Error in the controller configuration:
  - Incorrect remote IP address
  - The port number does not match the *ibaPDA* settings.
  - The port number is blocked by the firewall.

Other errors:

- If the telegram counters do not increment continuously, the messages are not sent cyclically on the controller side.
- In case the values for the data length change, this is an indication of the following error:
  - Different messages with different layout are sent over the same connection.
  - The "delayed acknowledge" problem occurs, see ↗ *TCP performance problems caused by Delayed Acknowledge, page 31*.

## 4.6 Checking the connection for sent messages

Under the *Generic TCP* output interface, you can find the connection table.




**Display variants**

| | |
|---|---|
| Green with destination | The connection is ok and *ibaPDA* sends the data. |
| Gray with destination | An active module is defined, but there is no receiver with this IP address or port number available for the connection set-up. |
| Gray without destination | A passive module is defined, but the receiver has not established a connection yet with the specified port. This is why the partner field is empty. |

**Connection data, telegram data and statistics**

- Module name
- Destination: IP address and port number of the connection
- Sent message counter
- Size of the telegrams sent (user data)
- Telegram cycle, minimum, maximum and average value.

## 4.7 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.

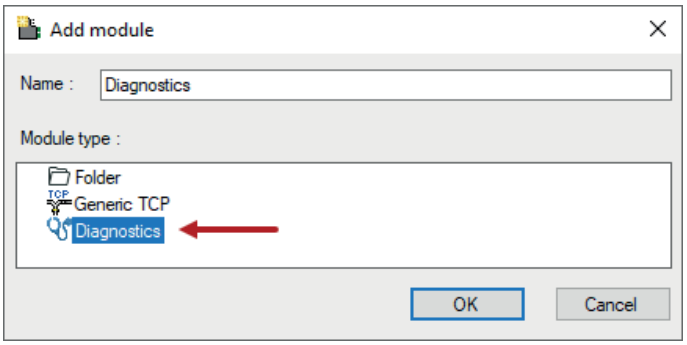
A diagnostic module is always assigned to a data acquisition module of the same interface and supplies its connection information. By using a diagnostic module, you can record and analyze the diagnostic information continuously in the *ibaPDA* system.

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:

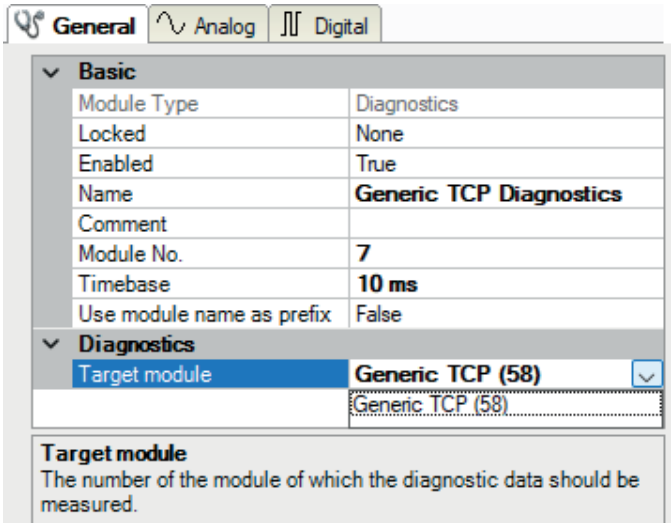
- A notification can be generated, whenever the error counter of a communication connection exceeds a certain value or the connection gets lost.
- In case of a disturbance, the current response times in the telegram traffic may be documented in an incident report.
- The connection status can be visualized 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

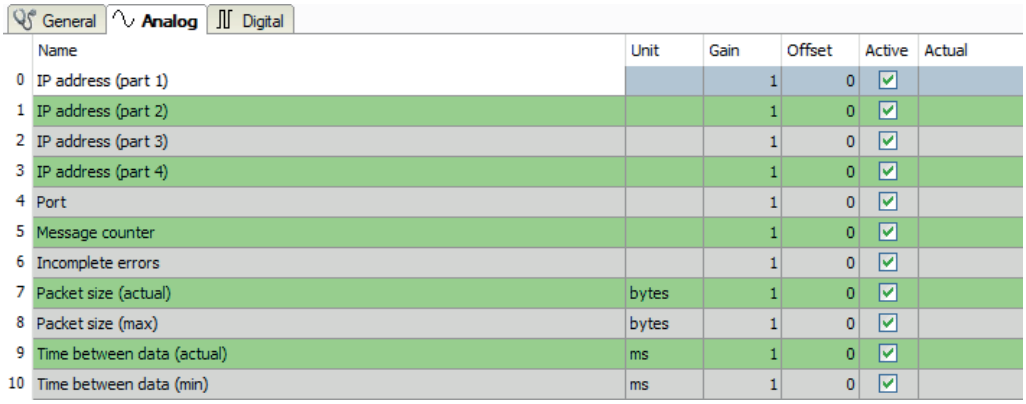
For a diagnostic module, you can make the following settings (example: Generic TCP):




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

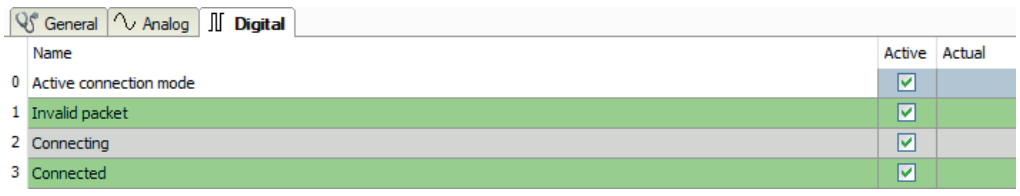
There is only one setting which is specific for the diagnostic module: the target module.

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 | ☑ | |
| 1 | IP address (part 2) | | 1 | 0 | ☑ | |
| 2 | IP address (part 3) | | 1 | 0 | ☑ | |
| 3 | IP address (part 4) | | 1 | 0 | ☑ | |
| 4 | Port | | 1 | 0 | ☑ | |
| 5 | Message counter | | 1 | 0 | ☑ | |
| 6 | Incomplete errors | | 1 | 0 | ☑ | |
| 7 | Packet size (actual) | bytes | 1 | 0 | ☑ | |
| 8 | Packet size (max) | bytes | 1 | 0 | ☑ | |
| 9 | Time between data (actual) | ms | 1 | 0 | ☑ | |
| 10 | Time between data (min) | ms | 1 | 0 | ☑ | |

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 | ☑ | |
| 1 | Invalid packet | ☑ | |
| 2 | Connecting | ☑ | |
| 3 | Connected | ☑ | |

**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.<br>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 *ibaPDA*). Corresponds to the assembly instance number |
| Connection ID T->O | ID of the connection for input data (from *ibaPDA* to target system). Corresponds to the assembly instance number |
| Connection phase (in) | Status of the ibaNet-E data connection for reception (in) |
| Connection phase (out) | Status of the ibaNet-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 *ibaPDA*) |
| Destination IP address (part 1-4) T->O | 4 octets of the IP address of the target system Input data (from *ibaPDA* 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 (*ibaPDA* receives) |
| Invalid data points | Number of received data points with missing configuration |
| 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 *ibaPDA* to the PLC/CPU |
| Output data length | Length of the data messages with output signals in bytes (*ibaPDA* 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) | Response time is the time between measured value request from *ibaPDA* and response from the PLC or reception of the data.<br>Actual: current value<br>Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters. |
| 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 *ibaPDA*) |
| Source IP address (part 1-4) T->O | 4 octets of the IP address of the target system Input data (from *ibaPDA* 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<br>Actual: between the last two messages<br>Max/min: statistical values since start of acquisition or reset of counters |
| Time offset (actual) | Measured time difference of synchronicity between *ibaPDA* and the ibaNet-E 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 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.<br>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

### 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 or spikes in data values (depending on the sending controller type)

**Cause**

The TCP/IP protocol has different options for "acknowledge" handling.

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).

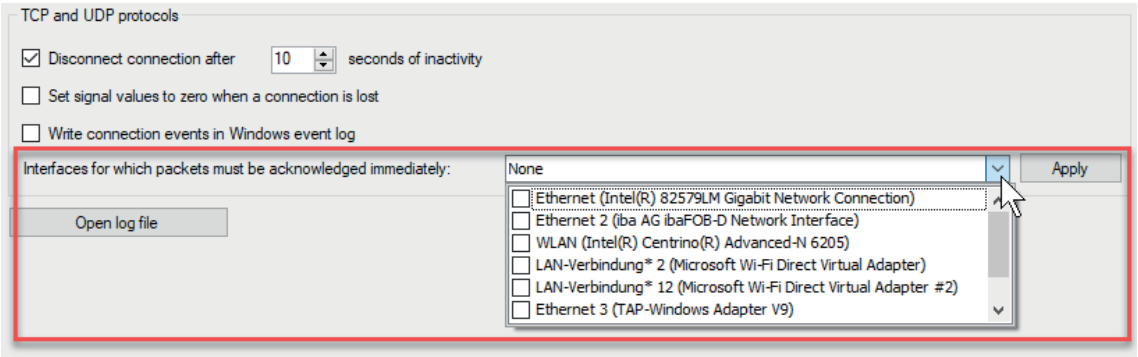
A "sliding window" (parameter Win=nnnn) controls the data flow. The recipient specifies how many bytes it can receive without sending an acknowledgment.

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**

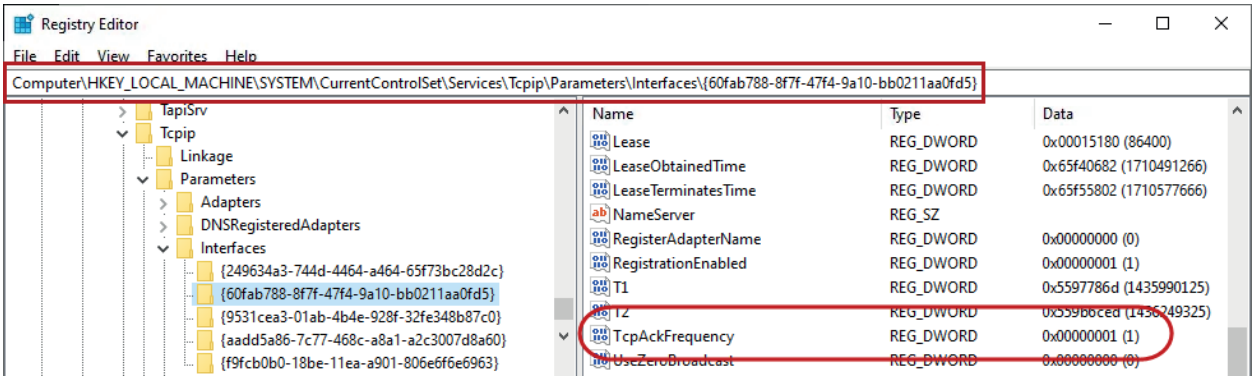
The "delayed acknowledge" can be switched off individually for each network adapter via an entry in the Windows Registry. For easy modification, *ibaPDA* offers a corresponding dialog in the I/O Manager under *General* in the tab *Settings*.

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.

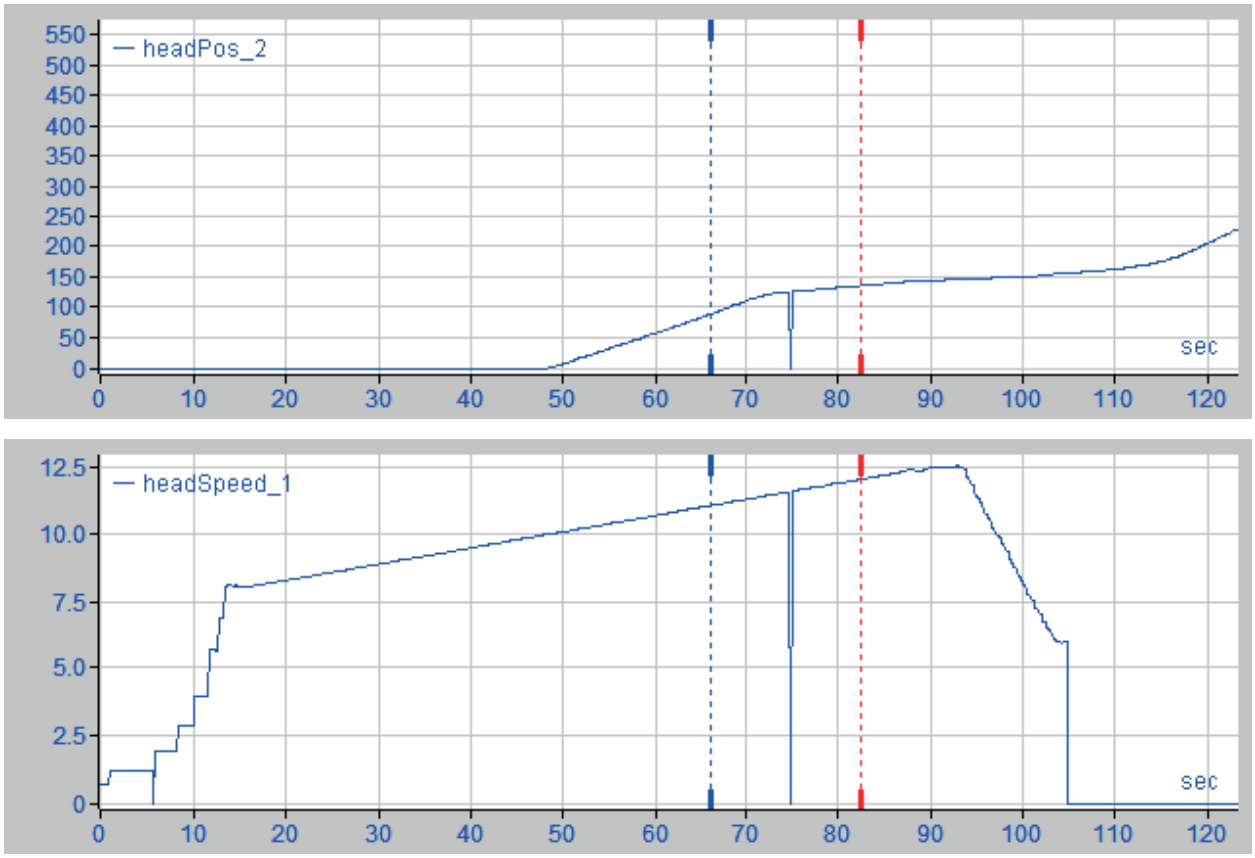
The User Datagram Protocol (UDP) is a minimal network protocol that is not connection-oriented and is unsecured against telegram loss. Notably, it does not provide acknowledgments for received data. In stable and high-performance networks, this is usually not critical and can be disregarded due to the cyclic data transmission with *ibaPDA*.

### 5.1.2 TCP data corruption resulting from the Nagle's Algorithm

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

**Errors shown in ibaPDA**

Incomplete telegrams 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.

However, the Nagle's Algorithm and the option *Delayed ACK* can negatively interact in TCP/IP networks, see also 5.1.1, page 31:

The Delayed ACK mechanism tries to send more data per segment if possible. 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 31. 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. Notably, it does not provide acknowledgments for received data. In stable and high-performance networks, this is usually not critical and can be disregarded due to the cyclic data transmission 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
Gebhardtstrasse 10-20
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**