ibaPDA-Interface-S7-TCP/UDP

Data Interface TCP/UDP to SIMATIC S7

Manual

Issue 3.1

Measurement Systems for Industry and Energy

www.iba-ag.com

Manufacturer

iba AG

Koenigswarterstrasse 44

90762 Fuerth

Germany

Contacts

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

Version	Date	Revision	Author	Version SW
3.1	03-2024	Module length in Header only with even number	RM/IP	8.4.0

Contents

1 About this documentation

This documentation describes the function and application of the software interface

ibaPDA-Interface-S7-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 is aimed at qualified professionals who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as professional if he/she is capable of assessing safety and recognizing possible consequences and risks on the basis of his/her specialist training, knowledge and experience and knowledge of the standard regulations.

This documentation in particular addresses persons, who are concerned with the configuration, test, commissioning or maintenance of Programmable Logic Controllers of the supported products. For the handling *ibaPDA-Interface-S7-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 control 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>
Filenames, paths	`Filename`, `Path` 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 to use the S7 TCP/UDP data interface:

- *ibaPDA* v7.0.0 or higher
- License for *ibaPDA-Interface S7-TCP/UDP*
- Network connection 10/100 Mbits
- STEP7 from version V4.0 or TIA Portal from V11
- S7 CPU with integrated PN port or communication processor

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

- For different ways of handling the TCP/IP acknowledge see ➚ *TCP performance problems caused by Delayed Acknowledge*, page 38 (all *ibaPDA* versions).

Licenses

Order No.	Product name	Description
31.001040	ibaPDA-Interface-S7-TCP/UDP	Extension license for an *ibaPDA* system by one TCP/IP and UDP/IP interface Number of connections: 64
31.101040	one-step-up-Interface-S7-TCP/UDP	Extension license for an existing interface ibaPDA-Interface-S7-TCP/UDP by another 64 S7-TCP/UDP connections, (max. 3 extension licenses permitted)

3 Data interface TCP/UDP to SIMATIC S7

3.1 General information

The S7-TCP/UDP interface can be used to acquire data from an S7 controller through the standard network card of the *ibaPDA* PC using the TCP/IP or UDP protocol. This requires the connection to be configured and data transmission to be programmed in the controller.

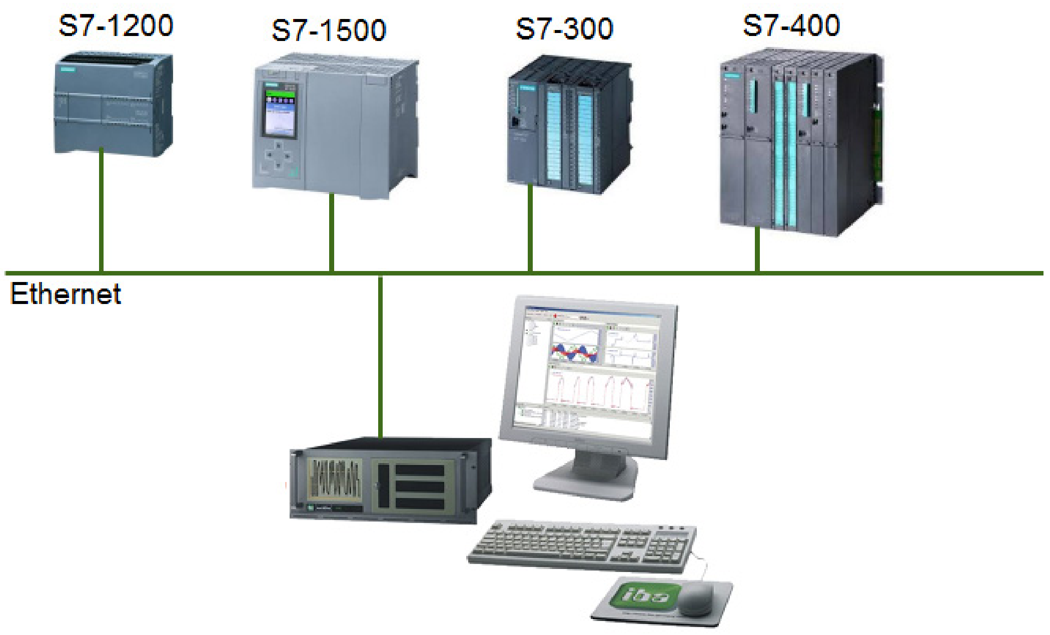
The signals to be measured are selected by arranging the values in data blocks (DB) whose data structures are defined by the module types of *ibaPDA*. Subsequently, the data blocks are sent to the *ibaPDA* PC as telegrams with S7 communication blocks.

Three module types are defined in *ibaPDA-Interface-S7-TCP/UDP*:

Integer:	32 analog values (integer) and 32 binary signals
Real:	8, 16 or 32 analog values (real) and 32 binary signals
Generic:	any data structure with a maximum length of 4096 bytes[1)]

Each module is assigned to a connection. You can create up to 256 connections on the side of *ibaPDA*. On the S7 side, the maximum number of connections depends on the CPU type.

This type of data acquisition has the main advantage of not requiring any special hardware if the controller already features an Ethernet connection.



TCP and UDP

The Transmission Control Protocol, short TCP, is a connection-oriented protocol. Its main function is to prevent data loss, divide files and data streams and assign data packets to applications.

The User Datagram Protocol, short UDP, is a connectionless transport protocol. Its function is similar to that of the connection-oriented TCP. However, it works connectionless and is thus not

1) Until ibaPDA V6.30 limited by properties of the S7 communication processors

secure, which means that the sender does not know whether the data packets it has sent have actually arrived. TCP sends confirmations upon receiving data, UDP does not. This method has the advantage that the packet header is much smaller and no acknowledgments have to be sent over the link. In principle, this enables a slightly higher data rate.

Both protocols use the IP Internet Protocol of layer 4 (transport layer) of the OSI model.

Note

The following examples use the term "connection" also for UDP. In this context, it refers only to the communication channel from sender to recipient and not to a network connection to be established and closed.

3.2 SIMATIC S7 configuration and engineering

This section describes how to establish the TCP/IP or UDP connection, the necessary data blocks and how to parameterize the send blocks. Various options are possible depending on the CPU family.

- CPUs without a local Ethernet interface
  The S7 side is configured using the STEP7 tools "HW Config" and "NetPro" included in SIMATIC Manager. In the STEP 7 program, you insert send blocks (AG_SEND, AG_LSEND) that use the configured connections.
- CPUs with a local Ethernet interface
  You do not have to configure the connections separately. Both connection establishment and sending are performed using standard blocks (TCON, TSEND, TUSEND) in STEP 7.
- The CPUs of the S7-1200 and S7-1500 series
  These CPUs generally feature local Ethernet interfaces and can only be configured using the TIA Portal. Here, there are blocks that perform both the task of connection establishment and sending data (TSEND_C).

Note

Please observe the following in all connection types described:

- The S7-CPU is the active partner in all connections.
- The partner port must match the setting in *ibaPDA* (interface S7-TCP-UDP) (default setting in *ibaPDA*: 4170).
- This port has to be enabled in the *ibaPDA* PC in the Windows firewall.
- This port must not be assigned elsewhere.
- When creating additional connections, please observe:
  - Always assign a new connection name.
  - Always assign a new local port number.
  - Always use the same partner IP address.
  - Always use the same partner port number.

3.2.1 Data blocks

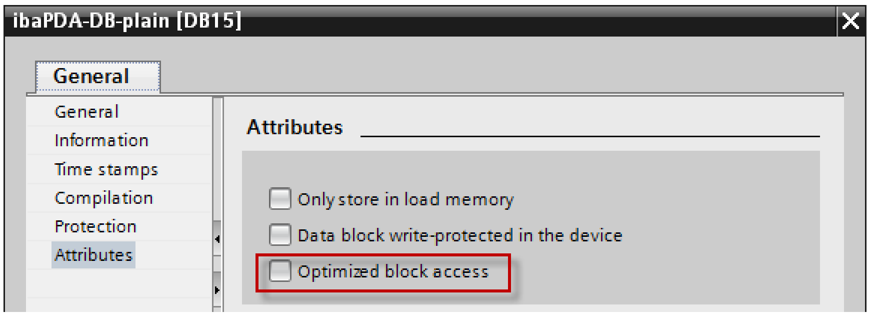
All methods described above require the data to be sent to be provided in data blocks.

According to the *ibaPDA* module structure, the data for each module are transmitted with a telegram. Each telegram is based on a data block and a connection. The data blocks have a uniform header and a data structure that corresponds to the module type.

Note

Special for the S7-1200 / S7-1500 controller:

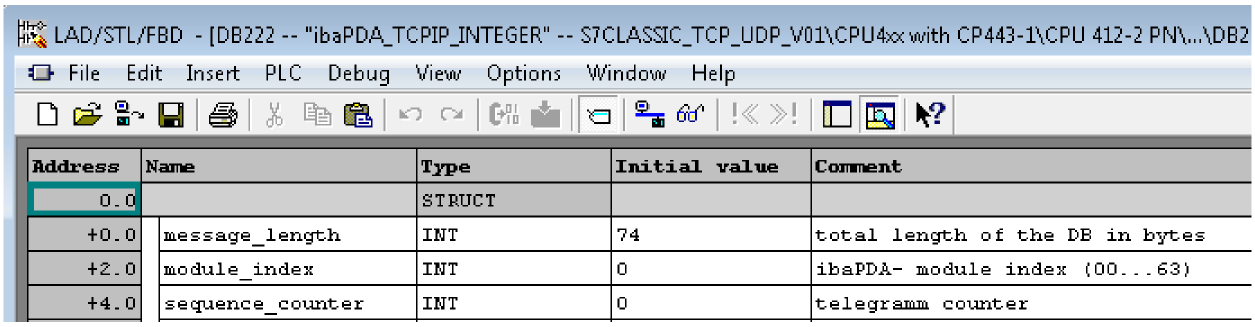
For the telegram data blocks, deactivate the block attribute *Optimized block access*. Access to optimized data blocks for S7-1200/S7-1500 controllers is not supported.



Only store in load memory
Data block write-protected in the device
Optimized block access

3.2.1.1 Header

The header consists of 3 integer values.


LAD/STL/FBD - [DB222 -- "ibaPDA_TCPIP_INTEGER" -- S7CLASSIC_TCP_UDP_V01\CPU4xx with CP443-1\CPU 412-2 PN\...\DB2
File Edit Insert PLC Debug View Options Window Help

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	message_length	INT	74	total length of the DB in bytes
+2.0	module_index	INT	0	ibaPDA- module index (00...63)
+4.0	sequence_counter	INT	0	telegramm counter

- message_length
  Total size (in bytes) of the data packet. Message_length must be an even number of Bytes. This value must not be changed during data transmission. This value also has to be specified when calling the send block.
  The length depends on the module type:

for module type Integer:	74
for module type Real:	42, 74 or 138 (for 8, 16 or 32 reals)
for module type Generic:	8...4096

- module_index
  Identifier for assigning the data record to the interface module in *ibaPDA*. The module indices are created by a serial number 00…63 and an offset that corresponds to the module type and the license.

Module type	1st License	2nd License	3rd License	4th 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 corresponds to the index in the *ibaPDA* module setting. This value must be unique and must not be changed during data transmission.

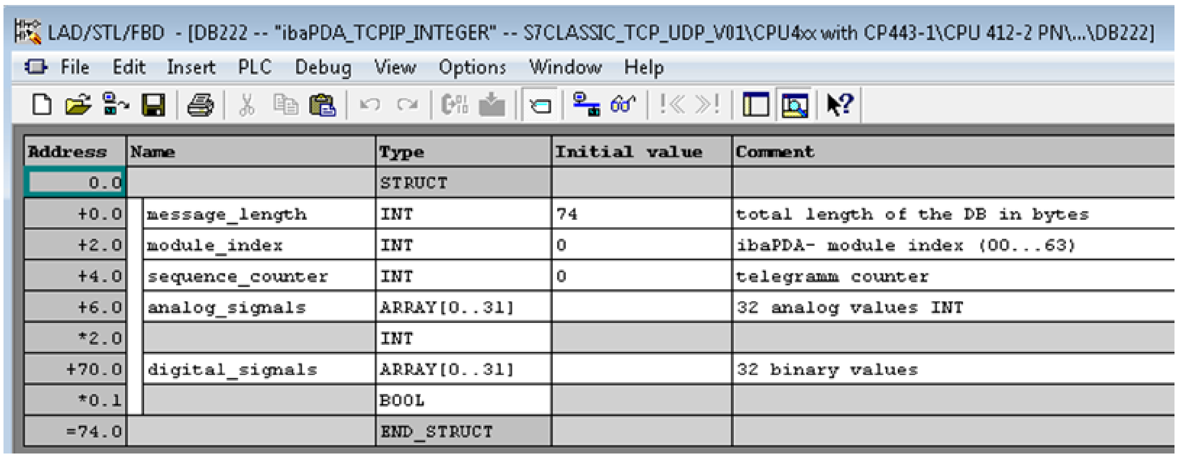
- sequence_counter
  Each successful send job increments the value by 1. This has to be programmed in the S7. If the counter value does not change by +1, *ibaPDA* displays a sequence error in the connection list.
  In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000).

3.2.1.2 Data areas

The structure of the data area depends on the module type.

Module type Integer

After the header, starting at offset 6, follow the 32 integer analog values and subsequently, starting at offset 70, the 4 bytes of binary values.


LAD/STL/FBD - [DB222 -- "ibaPDA_TCPIP_INTEGER" -- S7CLASSIC_TCP_UDP_V01\CPU4xx with CP443-1\CPU 412-2 PN\...\DB222]
File
Edit
Insert
PLC
Debug
View
Options
Window
Help

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	message_length	INT	74	total length of the DB in bytes
+2.0	module_index	INT	0	ibaPDA- module index (00...63)
+4.0	sequence_counter	INT	0	telegramm counter
+6.0	analog_signals	ARRAY[0..31]		32 analog values INT
*2.0		INT		
+70.0	digital_signals	ARRAY[0..31]		32 binary values
*0.1		BOOL		
=74.0		END_STRUCT		

Note

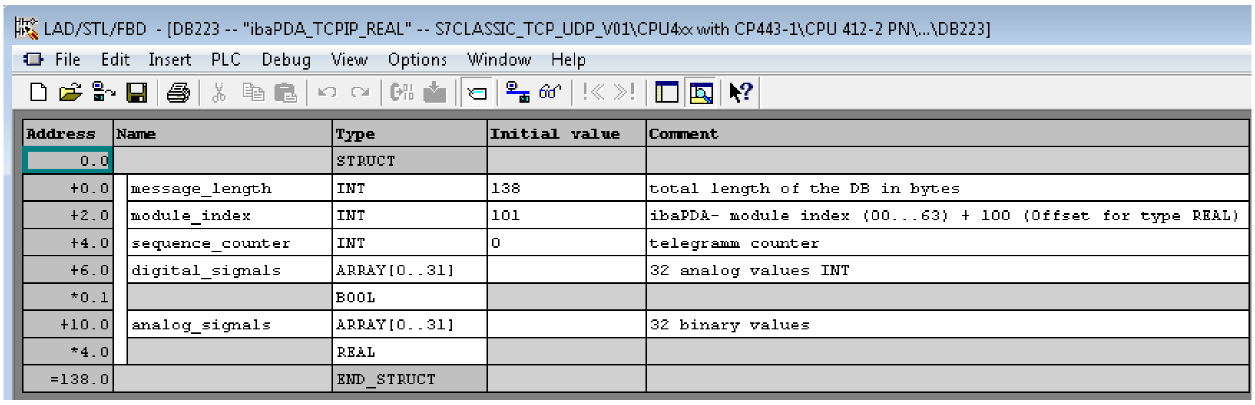
Observe the different byte order between S7 and *ibaPDA*.

Example:

If you set bit DB222.DBX70.0, it will arrive as bit 24 in *ibaPDA*. But if you write 16#00000001 to DB222.DBD70, bit 0 is set in *ibaPDA*.

Module type Real

After the header, starting at offset 6, follow the 4 bytes of binary values and subsequently, starting at offset 10, either 8, 16 or 32 analog values in the real format.


LAD/STL/FBD - [DB223 -- "ibaPDA_TCPIP_REAL" -- S7CLASSIC_TCP_UDP_V01\CPU4xx with CP443-1\CPU 412-2 PN\...\DB223]

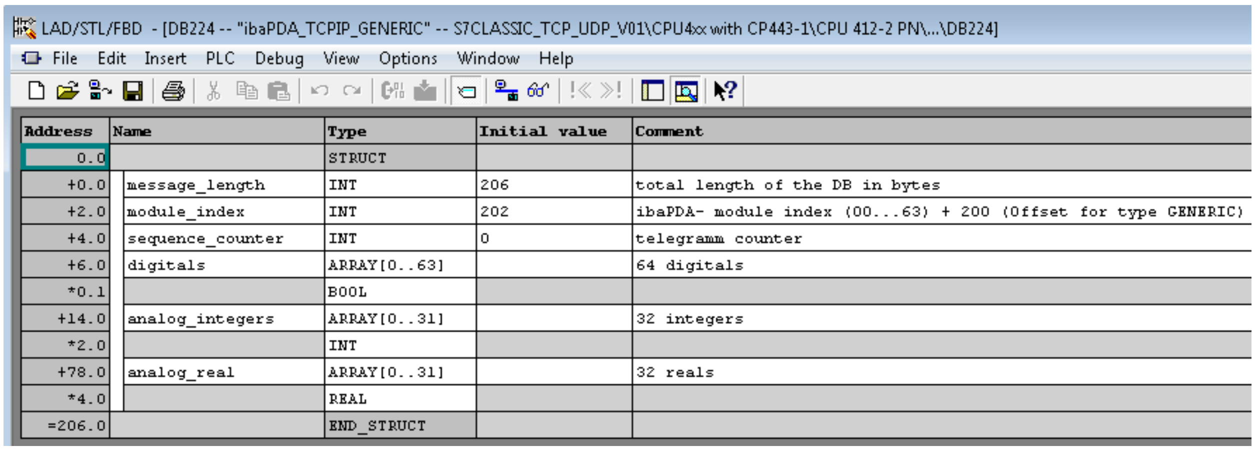
Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	message_length	INT	138	total length of the DB in bytes
+2.0	module_index	INT	101	ibaPDA- module index (00...63) + 100 (Offset for type REAL)
+4.0	sequence_counter	INT	0	telegramm counter
+6.0	digital_signals	ARRAY[0..31]		32 analog values INT
*0.1		BOOL		
+10.0	analog_signals	ARRAY[0..31]		32 binary values
*4.0		REAL		
=138.0		END_STRUCT		

Module type Generic

Any order of data with different data types can follow after the header starting at offset 6. *ibaPDA* supports the following data formats:

BYTE, WORD, DWORD, INT, DINT and FLOAT

The data structure defined here has to be copied in *ibaPDA*. The BYTE, WORD and DWORD variables may also be interpreted as 8, 16 or 32 single bits (and vice versa).


LAD/STL/FBD - [DB224 -- "ibaPDA_TCPIP_GENERIC" -- S7CLASSIC_TCP_UDP_V01\CPU4xx with CP443-1\CPU 412-2 PN\...\DB224]

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	message_length	INT	206	total length of the DB in bytes
+2.0	module_index	INT	202	ibaPDA- module index (00...63) + 200 (Offset for type GENERIC)
+4.0	sequence_counter	INT	0	telegramm counter
+6.0	digitals	ARRAY[0..63]		64 digitals
*0.1		BOOL		
+14.0	analog_integers	ARRAY[0..31]		32 integers
*2.0		INT		
+78.0	analog_real	ARRAY[0..31]		32 reals
*4.0		REAL		
=206.0		END_STRUCT		

3.2.2 S7-300/S7-400-CPUs without local Ethernet interface

The connection of a CPU without local Ethernet port is configured in the following steps:

1. Configuring the SIMATIC NET Ethernet-CP in HW Config
2. Creating the TCP or UDP connections in NetPro
3. Program code to map the signal data in the data blocks
4. Program code to increment the "sequence_counter"
5. Calling the communication blocks in the S7 program of the CPU

You have to configure a separate connection for each module configured in *ibaPDA*.

Configuring the Ethernet CP in HW Config

1. Select the PN-IO interface of the Ethernet CP (CP343-1 or CP443-1).
2. Open the *Properties* in the *General* tab.
3. Assign the IP address and the subnet mask of the S7 controller.

Creating the connection in NetPro

Select the following parameters:

Connection partners (station):	unspecified
Connection – Type:	TCP connection or UDP connection
General – Name:	assign a unique name
General – Active connection establishment	
■ for TCP connection:	enabled
■ for UDP connection:	the parameter does not exist (always passive)
Addresses – Local – Port:	Assign a unique port number.
Addresses – Partner – IP:	IP address of the *ibaPDA* PC
Addresses – Partner – Port:	Port number of the *ibaPDA* PC

Note

For assigning the port numbers, observe the notes in ➚ *SIMATIC S7 configuration and engineering*, page 9.

Mapping the signal data

Cyclically copy the desired signal data into the data blocks of the telegram modules at any position of your S7 program.

Incrementing the "sequence_counter"

Increase the "sequence_counter" in the telegram data block with each rising edge of the DONE output.

Reset the "sequence_counter" to 0 if the CPU is restarted. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000).

Calling the communication blocks

Depending on the CPU type used, different SIMATIC communication blocks are required.

Communication block	usable for		Note
	S7-300	S7-400	
AG_SEND (FC5)	X		These blocks allow sending a maximum of 240 bytes up to block version V3.0. The current block versions allow a data range of up to 8192 bytes for TCP and 2048 bytes for UDP.
		X	The data length for S7-400 is limited to 240 bytes. To transmit greater data ranges, the block AG_LSEND has to be used. [2)]
AG_LSEND (FC50)		X	The maximum data length is 8192 bytes for TCP and 2048 bytes for UDP. Please refer to the CP's product information for information on the supported data range.

The reference to the connection configured in NetPro is made through the parameter ID (first part of the local ID of the NetPro connection) and the parameter LADDR (HW address from the NetPro connection).

Note

Always use the latest version of the SIMATIC NET communication blocks in your STEP7 project. You will find it in SIMATIC Manager under *File – Open – Libraries – SIMATIC_NET_CP*.

Tip

If AG_SEND/AG_LSEND is called cyclically, the send cycle is half the size of the call cycle at the most, because the block's output parameters are updated in the call in the cycle after the send job. You can prevent this by calling the block twice in each cycle. The first time to query the status (ACT=0), the second time to send the data (ACT=1).

Other documentation

For more information on configuring the communication, see the STEP 7 online help and the following FAQ from Siemens:

Configuring a TCP connection via Ethernet (TCP native) between a SIMATIC S7 and a PC with socket interface

https://support.industry.siemens.com/cs/ww/en/view/18843927

https://support.industry.siemens.com/cs/ww/en/view/17853532

https://support.industry.siemens.com/cs/ww/en/view/18513371

2) See also SIMATIC STEP7 help subject: "FCs for the SEND/RECEIVE interface".

3.2.3 S7-300/S7-400-CPUs with local Ethernet interface

The connection of a CPU with local Ethernet port is configured in the following steps:

1. Configuring the CPU's Ethernet interface in HW Config
2. Program code to map the signal data in the data block
3. Program code to increment the "sequence_counter"
4. Creating and parameterizing the connection data (data structures TCON_PAR and if applicable TADDR_PAR)
5. Calling the communication blocks in the S7 program of the CPU

You have to configure a separate connection for each module configured in *ibaPDA*.

Configuring the CPU's Ethernet interface in HW Config

1. Select the PN-IO interface of the CPU.
2. Open the *Properties* in the *General* tab.
3. Assign the IP address and the subnet mask of the S7 controller.

Mapping the signal data

Cyclically copy the desired signal data into the data blocks of the telegram modules at any position of your S7 program.

Incrementing the "sequence_counter"

Increase the "sequence_counter" in the telegram data block with each rising edge of the DONE output.

Reset the "sequence_counter" to 0 if the CPU is restarted. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000).

Parameterizing the connection

Create a static variable or a DB with the data structure TCON_PAR (UDT65) and enter the following parameters:

id:	Connection ID, reference for the associated TCON and TSEND or TUSEND block	
connection_type:	for TCP: B#16#11	for UDP: B#16#13
active_est:	for TCP: TRUE	for UDP: FALSE
local_device_id:	2, 3 or 5 (depends on the CPU type)[3)]	
local_tsap_id:	Unique port number for each connection	
rem_staddr:	only for TCP: IP address of the *ibaPDA* PC	
rem_tsap_id:	only for TCP: Port number of the *ibaPDA* PC	

3) See online help for system functions "Parameterizing the communication connections..."

With UDP, the remote IP address and port number is not taken from the connection data, but has to be stored in a separate data range with the structure TADDR_PAR (UDT66):

rem_ip_addr	IP address of the *ibaPDA* PC
rem_port_nr	Port number of the *ibaPDA* PC

Note

For assigning the port number, observe the notes in ➚ *SIMATIC S7 configuration and engineering*, page 9.

Calling the communication blocks in the S7 program of the CPU

The following communication block is used:

- TCON (FB65): to establish the connection
- TSEND (FB63): to send the data via TCP
- TUSEND (FB67): to send the data via UDP

The connection is parameterized by means of the data range with the specified structure, which is referenced through the unique connection ID.

Note

Always use the latest version of the SIMATIC NET communication blocks in your STEP 7 project. You will find it in SIMATIC Manager under *File – Open – Libraries – Standard Library – Communication Blocks.*

Other documentation

For more information on configuring the communication, see the STEP 7 online help and the following FAQ from Siemens:

How do you program the communication blocks FB63 "TSEND", FB64 "TRCV", FB65 "TCON" and FB66 "TDISCON" in order to use the TCP protocol for data exchange by means of the integrated PROFINET interface of an S7-300/S7-400 CPU?

https://support.industry.siemens.com/cs/ww/en/view/29737950

3.2.4 S7-1200 CPUs with local Ethernet interface

Note

The following notes are applicable to firmware versions up to 3.x.

For S7-1200 from version 4.0, the settings are identical to the S7-1500 CPUs.

See also ➚ *S7-1500-CPUs with local Ethernet interface*, page 19.

The connection of a CPU with local Ethernet port is configured in the following steps:

1. Configuring the CPU's Ethernet interface in the device configuration
2. Program code to map the signal data in the data block
3. Program code to increment the "sequence_counter"
4. Creating and parameterizing the connection data (data structure TCON_Param)
5. Calling the communication blocks in the S7 program of the CPU

You have to configure a separate connection for each module configured in *ibaPDA*.

Configuring the Ethernet interface in the device configuration

1. Select the device configuration.
2. On the graphic, click the Ethernet port connected to *ibaPDA*.
3. Select the tab *General – Ethernet address*.
4. Assign the IP address and the subnet mask of the S7 controller.

Mapping the signal data

Cyclically copy the desired signal data into the data blocks of the telegram modules at any position of your S7 program.

Incrementing the "sequence_counter"

Increase the "sequence_counter" in the telegram data block with each rising edge of the DONE output.

Reset the "sequence_counter" to 0 if the CPU is restarted. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000).

Parameterizing the connection

Create a static variable or a DB with the data structure TCON_Param and enter the following parameters:

local_device_id:	ID of the local interface: 1	
id:	Unique connection ID	
connection_type:	for TCP: 17	for UDP: 19
active_est:	for TCP: TRUE	for UDP: FALSE
rem_staddr_len:	for TCP: 4	for UDP: 0
rem_tsap_id_len:	for TCP: 2	for UDP: 0
remote_staddr:	only for TCP: IP address of the *ibaPDA* PC	
remote_tsap_id:	only for TCP: Port number of the *ibaPDA* PC: 4170	
local_tsap_id_len: 2	2	
local_tsap_id:	Unique port number for each connection	

With UDP, the remote IP address and port number is not taken from the connection data, but has to be stored in a separate data range with the structure TADDR_Param:

rem_ip_addr	IP address of the *ibaPDA* PC
rem_port_nr	Port number of the *ibaPDA* PC

Note

F or assigning the port number, observe the notes in↗ *SIMATIC S7 configuration and engineering*, page 9.

Calling the communication blocks

The following communication blocks are used:

- TSEND_C: to establish the connection and send the data for TCP/IP
- TCON : to establish the connection for UDP
- TUSEND: to send the data for UDP

The connection is parameterized by means of a data range with the specified structure, which is referenced through the unique connection ID.

Other documentation

For more information on configuring the communication, see the TIA online help and the following FAQ from Siemens:

How do you program the TSEND_C and TRCV_C instructions for open user communication over the integrated PROFINET interface of the S7-1200 CPU?

https://support.industry.siemens.com/cs/ww/en/view/67196808

3.2.5 S7-1500-CPUs with local Ethernet interface

Note

The following instructions also apply to S7-1200 from version 4.0.

The connection of a CPU with local Ethernet port is configured in the following steps:

1. Configuring the CPU's Ethernet interface in the device configuration
2. Program code to map the signal data in the data block
3. Program code to increment the "sequence_counter"
4. Creating and parameterizing the connection data (data structure TCON_IP_v4)
5. Calling the communication blocks in the S7 program of the CPU

You have to configure a separate connection for each module configured in *ibaPDA*.

Configuring the Ethernet interface in the device configuration

1. Select the device configuration.
2. On the graphic, click the Ethernet port connected to *ibaPDA*.
3. Select the tab *General – Ethernet address*.
4. Assign the IP address and the subnet mask of the S7 controller.

Mapping the signal data

Cyclically copy the desired signal data into the data blocks of the telegram modules at any position of your S7 program.

Incrementing the "sequence_counter"

Increase the "sequence_counter" in the telegram data block with each rising edge of the DONE output.

Reset the "sequence_counter" to 0 if the CPU is restarted. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000).

Parameterizing the connection

Create a local variable or a DB with the data structure TCON_Param and enter the following parameters:

interface_id:	Hardware ID of the local interface: see device configuration	
id:	Connection ID	
connection_type:	for TCP: 11	for UDP: 19
active_est:	for TCP: TRUE	for UDP: FALSE
remote_address:	only for TCP: IP address of the *ibaPDA* PC	

remote_port:	only for TCP: Port number of the *ibaPDA* PC: 4170
local_port:	Unique port number for each connection

With UDP, the remote IP address and port number is not taken from the connection data, but has to be stored in a separate data range with the structure UDT 66 ("TADDR_Param"):

rem_ip_addr	IP address of the *ibaPDA* PC
rem_port_nr	Port number of the *ibaPDA* PC

Note

For assigning the port numbers, observe the notes in ➚ *SIMATIC S7 configuration and engineering*, page 9.

Calling the communication blocks

The following communication block is used:

- TSEND_C: to establish the connection and send the data

The connection is parameterized through a data range with the specified structure, which is referenced through a unique connection ID.

Note

The block TSEND_C is available in two different variants.

- For CPU S7-1200 ≤ V3.x (see ➚ *S7-1200 CPUs with local Ethernet interface*, page 17)
- For CPU S7-1500 and CPU S7-1200 ≥ V4.0

The differences refer to the different system data types for parameterizing the connection.

Other documentation

For further information on configuring the communication, please refer to the following FAQ from Siemens:

How do you program the TSEND_C and TRCV_C instructions for open user communication over the integrated PROFINET interface of the S7-1200/S7-1500 CPU?

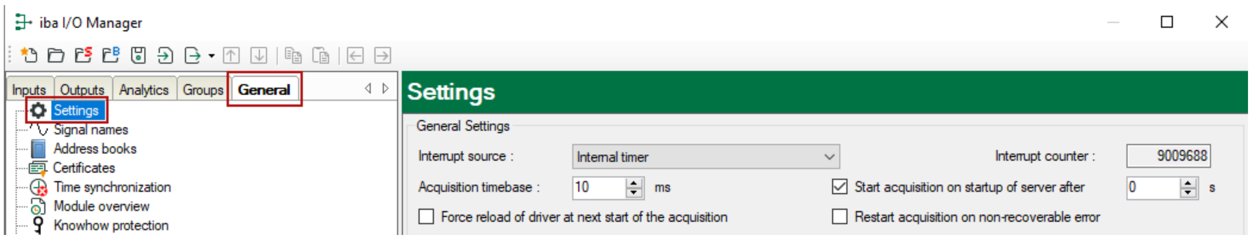
https://support.industry.siemens.com/cs/ww/en/view/67196808

3.3 Configuration and engineering ibaPDA

The engineering for *ibaPDA* is described in the following. If all system requirements are fulfilled, *ibaPDA* displays the *S7 TCP/UDP* interface in the interface tree of the I/O Manager.

3.3.1 General settings

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



Settings

General Settings

Interrupt source : Internal timer

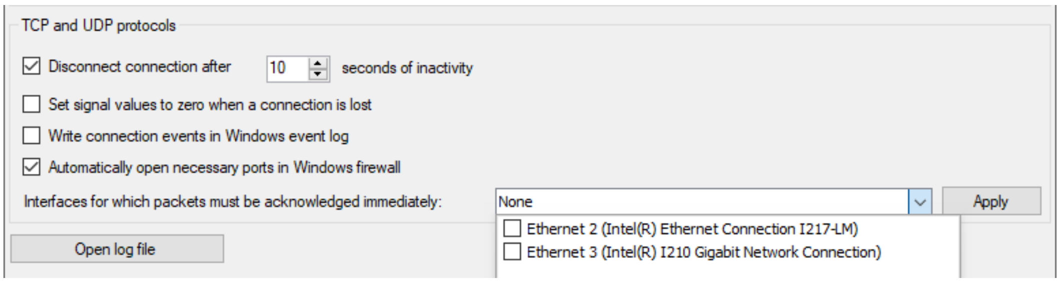
Interrupt counter : 9009688

Acquisition timebase : 10 ms

☑ Start acquisition on startup of server after 0 s

☐ Force reload of driver at next start of the acquisition

☐ Restart acquisition on non-recoverable error



TCP and UDP protocols

☑ Disconnect connection after 10 seconds of inactivity

☐ Set signal values to zero when a connection is lost

☐ Write connection events in Windows event log

☑ Automatically open necessary ports in Windows firewall

Interfaces for which packets must be acknowledged immediately: None

☐ Ethernet 2 (Intel(R) Ethernet Connection I217-LM)

☐ Ethernet 3 (Intel(R) I210 Gigabit Network Connection)

Disconnect connection after … seconds of inactivity
Behavior and timeout duration can be specified.

Set signal values to zero when a connection is lost
If this option is disabled, the value read last will be kept.

Write connection events in Windows event log
Current events are logged in Windows.

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

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

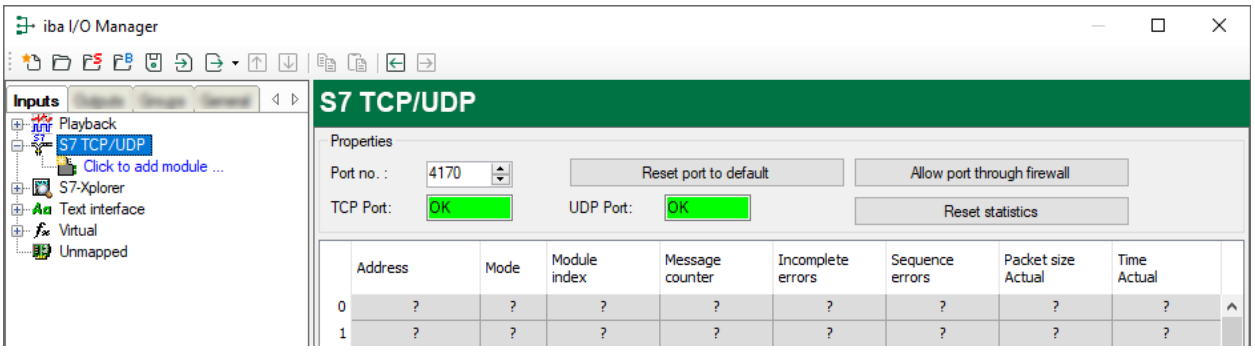
Interfaces for which packets must be acknowledged immediately
Selection of required interfaces.

Note

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

3.3.2 General interface settings

The interface provides the following functions and configuration options:



S7 TCP/UDP

Properties

Port no. : 4170

TCP Port: OK UDP Port: OK

	Address	Mode	Module index	Message counter	Incomplete errors	Sequence errors	Packet size Actual	Time Actual
0	?	?	?	?	?	?	?	?
1	?	?	?	?	?	?	?	?

Port no.
Used port on the computer. You can change the port number, but in the S7 project engineering and in *ibaPDA* you must use the same port to establish a connection.

The default port number is 4170.

<Reset port to default>
Use this button to reset the port to the default port number.

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.

TCP Port/UDP Port
Displays the port status.

- OK: You can open the socket on this port.
- ERROR: Conflicts occur, e.g. the port is already occupied.

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

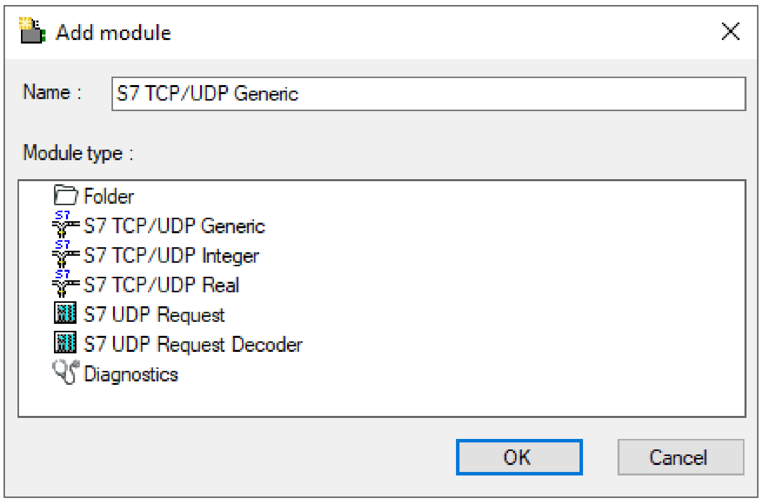
Connection table
For each connection, the table shows the connection status, the current values for the update time (current, real value, average, min. and max.) as well as the data size. In addition, there is an error counter for the individual connections during the acquisition.

See ➚ *Checking the connection*, page 32

3.3.3 Adding a module

Procedure

1. Click on the blue command *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

If a TCP/IP or UDP connection to S7 exists already, right-click the interface and select *Autodetect*. Then the correct modules are automatically created for all available connections.

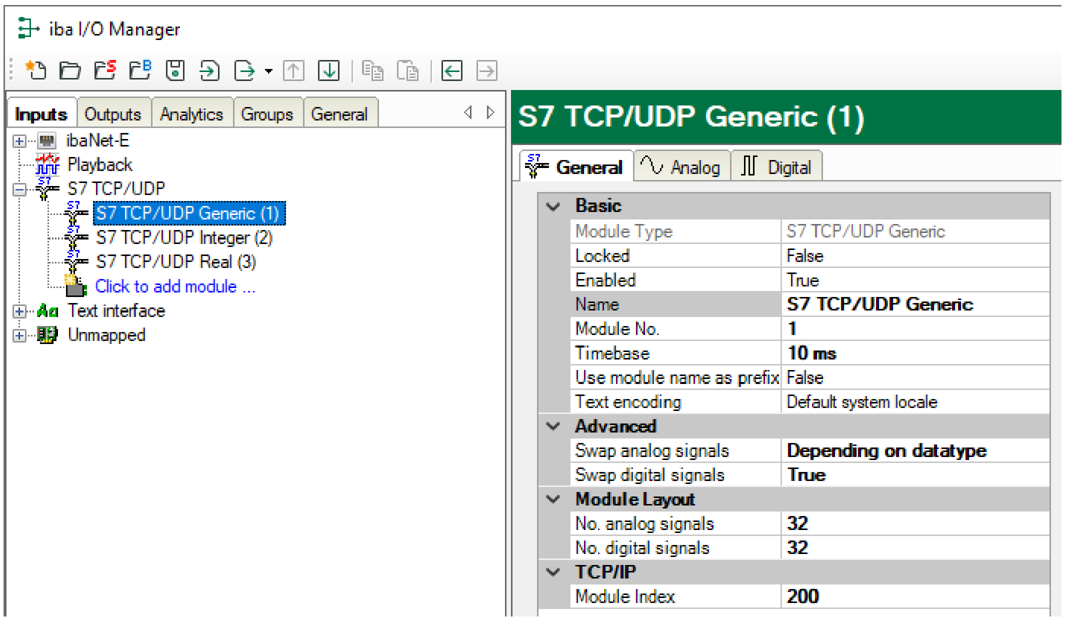
Module types

Module type	Description
S7 TCP/UDP Generic	Module for any data structure with up to 4096 bytes length
S7 TCP/UDP Integer	Module for up to 32 analog signals (integer) and 32 digital signals
S7 TCP/UDP Real	Module for up to 32 analog signals (Real) and 32 digital signals
S7 UDP Request	Request module for a maximum of 1024 analog and 1024 digital signals. See manual *ibaPDA-Request-S7-UDP*
S7 UDP Request Decoder	Request module for a maximum of 11728 digital signals transmitted in the form of a maximum of 733 words (1466 byte). See manual *ibaPDA-Request-S7-UDP*
Diagnostics	Type for creating diagnostic modules

3.3.4 General module settings

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

All modules have the following setting options.



Basic	
Module Type	S7 TCP/UDP Generic
Locked	False
Enabled	True
Name	**S7 TCP/UDP Generic**
Module No.	**1**
Timebase	**10 ms**
Use module name as prefix	False
Text encoding	Default system locale
Advanced	
Swap analog signals	**Depending on datatype**
Swap digital signals	**True**
Module Layout	
No. analog signals	**32**
No. digital signals	**32**
TCP/IP	
Module Index	**200**

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.

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

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 1000. The signal tables are adjusted accordingly.

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 also ➚ *Header*, page 10.

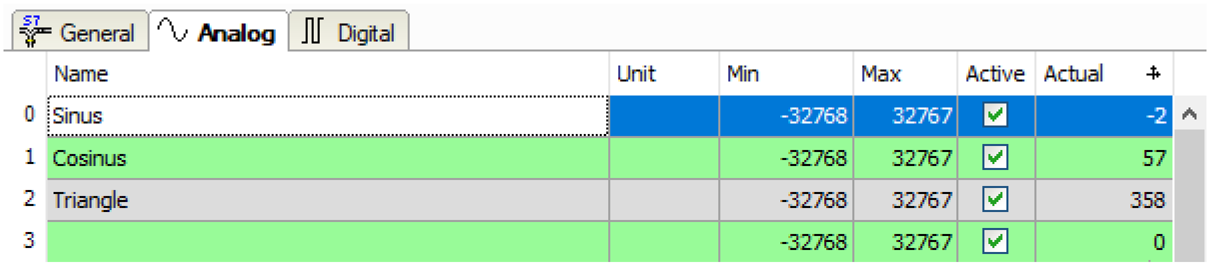
Other documentation

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

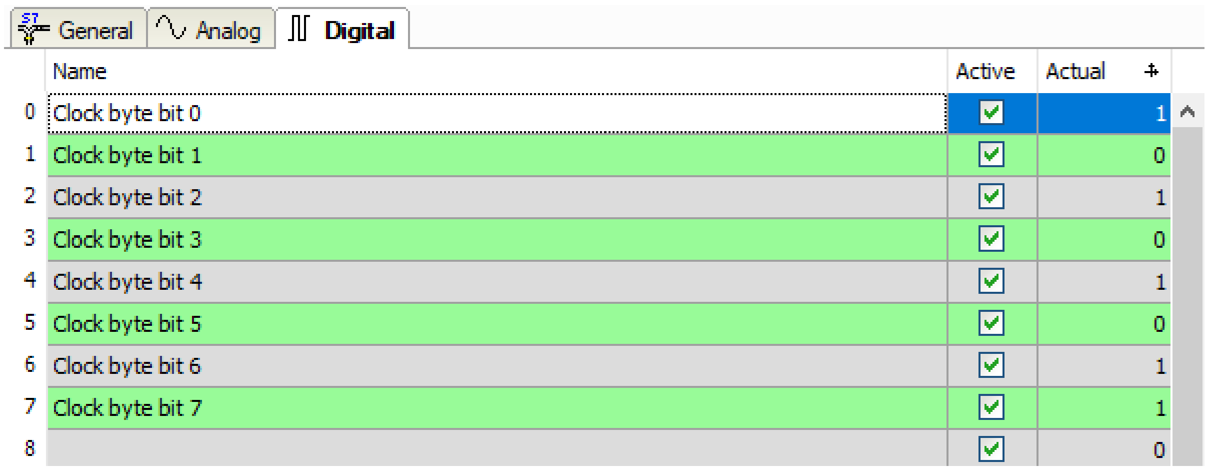
3.3.5 Signal configuration

The data to be measured are selected on the SIMATIC S7 side by mapping the signals in data blocks, which are cyclically sent to *ibaPDA*.

In the I/O Manager, you can assign name, comment and, if required, unit and scaling factor to the signals and enable them.



	Name	Unit	Min	Max	Active	Actual
0	Sinus		-32768	32767	✓	-2
1	Cosinus		-32768	32767	✓	57
2	Triangle		-32768	32767	✓	358
3			-32768	32767	✓	0



	Name	Active	Actual
0	Clock byte bit 0	✓	1
1	Clock byte bit 1	✓	0
2	Clock byte bit 2	✓	1
3	Clock byte bit 3	✓	0
4	Clock byte bit 4	✓	1
5	Clock byte bit 5	✓	0
6	Clock byte bit 6	✓	1
7	Clock byte bit 7	✓	1
8		✓	0

Tip

You can use the autofill function for the column, see the *ibaPDA* manual.

Other documentation

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

3.3.6 Module type S7 TCP/UDP Integer

With the *Integer* module, you can acquire up to 32 analog values (integer) and 32 binary signals.

The module does not have any module-specific settings.

3.3.7 Module type S7 TCP/UDP Real

With the *Real* module, you can acquire up to 32 analog values (real) and 32 binary signals.

Module-specific settings
No. analog signals

Define the number of configurable analog in the signal table steps of 8, 16 and 32. (The number of digital signals is fixed at 32.)

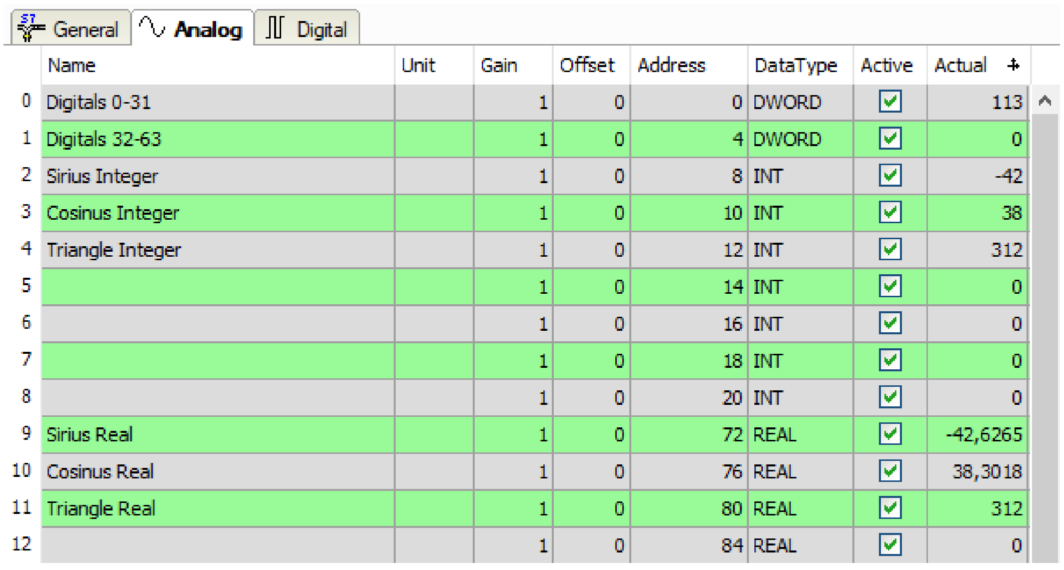
3.3.8 Module type S7 TCP/UDP Generic

With the *Generic* module, you can acquire any data structure with a maximum length of 4096 bytes (see also footer in ➚ *General information*, page 8).

Module-specific settings
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. You must therefore subtract 6 bytes from the offset address in the data block to specify the correct address for the measurement data.

Note

The module *S7 TCP/UDP Generic* supports the acquisition and processing of strings as text signals. Therefore, you can select the datatype STRING[32] or CHAR[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 under the *Virtual* interface.



	Name	Unit	Gain	Offset	Address	DataType	Active	Actual
0	Digitals 0-31		1	0	0	DWORD	✓	113
1	Digitals 32-63		1	0	4	DWORD	✓	0
2	Sirius Integer		1	0	8	INT	✓	-42
3	Cosinus Integer		1	0	10	INT	✓	38
4	Triangle Integer		1	0	12	INT	✓	312
5			1	0	14	INT	✓	0
6			1	0	16	INT	✓	0
7			1	0	18	INT	✓	0
8			1	0	20	INT	✓	0
9	Sirius Real		1	0	72	REAL	✓	-42,6265
10	Cosinus Real		1	0	76	REAL	✓	38,3018
11	Triangle Real		1	0	80	REAL	✓	312
12			1	0	84	REAL	✓	0

3.3.9 S7 UDP Request/S7 UDP Request Decoder module types

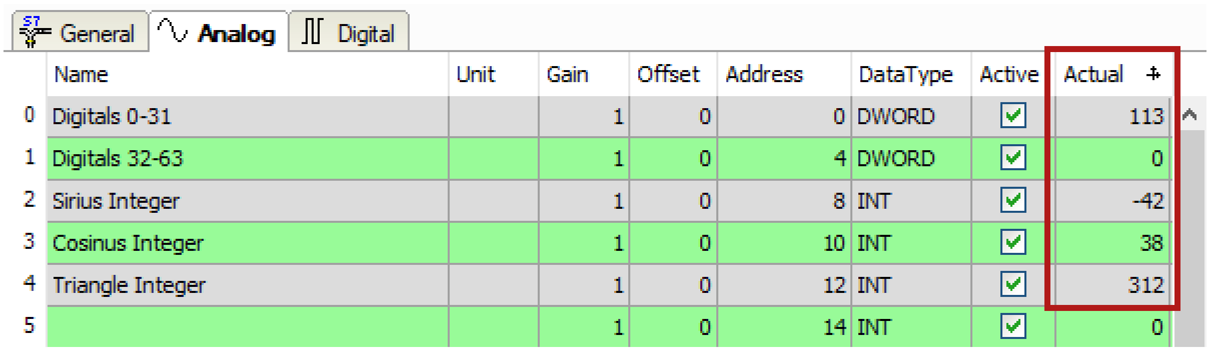
These two module types are only displayed if the *ibaPDA-Request-S7-UDP* license is available.

Other documentation

The modules and their functions are described in detail in the manual *ibaPDA-Request-S7-UDP*.

3.3.10 Module diagnostics

The tables *Analog* and *Digital* of the modules show the telegram contents.



	Name	Unit	Gain	Offset	Address	DataType	Active	Actual
0	Digitals 0-31		1	0	0	DWORD	✓	113
1	Digitals 32-63		1	0	4	DWORD	✓	0
2	Sirius Integer		1	0	8	INT	✓	-42
3	Cosinus Integer		1	0	10	INT	✓	38
4	Triangle Integer		1	0	12	INT	✓	312
5			1	0	14	INT	✓	0

The following errors may occur:

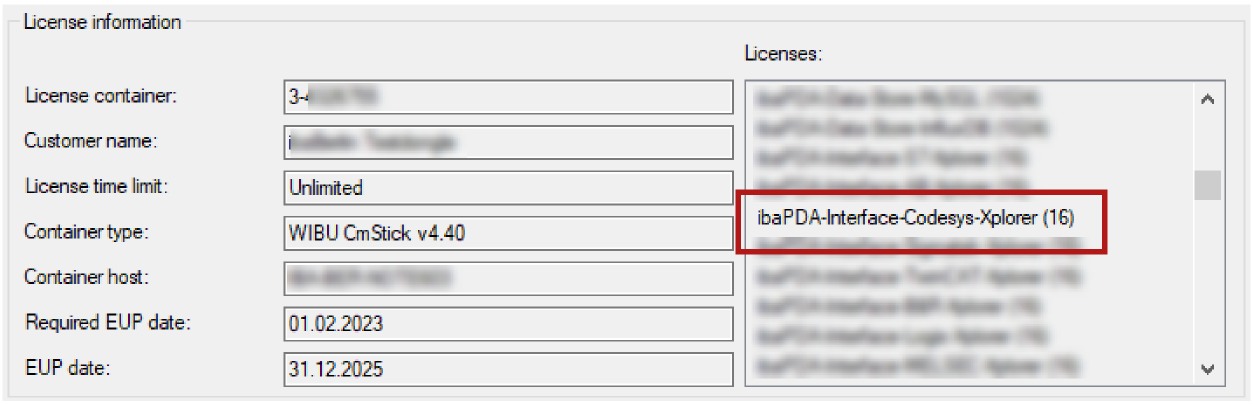
- No data are displayed:
  - The telegram DB on the S7 side is not filled correctly.
  - The connectors of the send block are connected incorrectly.
- Incorrect values are displayed:
  - The telegram DB on the S7 side is not filled correctly (offset error).
  - The byte order is set incorrectly, see ↗ *General module settings*, page 24.
  - 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 24.
- The telegrams do not arrive faster than approx. 200 ms with sequence error:
  - Problem with "delayed acknowledge", see ↗ *TCP performance problems caused by Delayed Acknowledge*, page 38
  - Problem caused by "Nagle's Algorithm", see ↗ *TCP data corruption resulting from the Nagle's Algorithm*, page 40

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 this interface 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.



License information

License container:	3-[redacted]
Customer name:	i[redacted]
License time limit:	Unlimited
Container type:	WIBU CmStick v4.40
Container host:	[redacted]
Required EUP date:	01.02.2023
EUP date:	31.12.2025

Licenses:

[redacted]
ibaPDA-Interface-Codesys-Xplorer (16)
[redacted]

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.

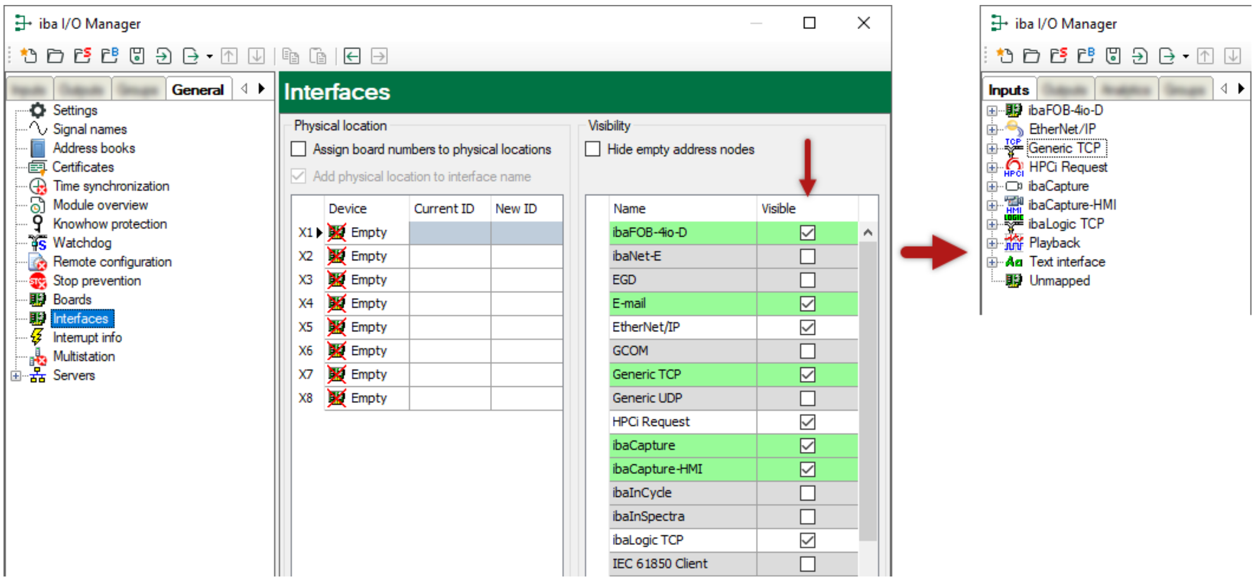
Visibility

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:



Interfaces

Physical location

☐ Assign board numbers to physical locations

✓ Add physical location to interface name

	Device	Current ID	New ID
X1	Empty		
X2	Empty		
X3	Empty		
X4	Empty		
X5	Empty		
X6	Empty		
X7	Empty		
X8	Empty		

Visibility

☐ Hide empty address nodes

Name	Visible
ibaFOB-4io-D	✓
ibaNet-E	☐
EGD	☐
E-mail	✓
EtherNet/IP	✓
GCOM	☐
Generic TCP	✓
Generic UDP	☐
HPCi Request	✓
ibaCapture	✓
ibaCapture-HMI	✓
ibaInCycle	☐
ibaInSpectra	☐
ibaLogic TCP	✓
IEC 61850 Client	☐

4.3 Log files

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

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

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

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

Files named `interface.txt` are always the current log files. Files named `Interface_yyyy_mm_dd_hh_mm_ss.txt` are archived log files.

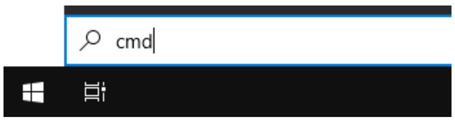
Examples:

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

4.4 Connection diagnostics with PING

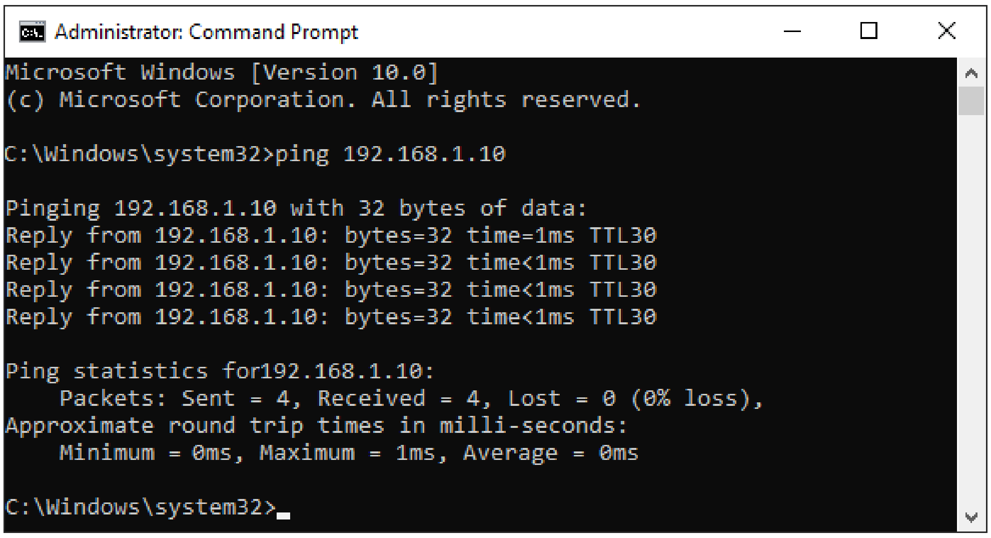
PING is a system command with which you can check if a certain communication partner can be reached in an IP network.

1. Open a Windows command prompt.



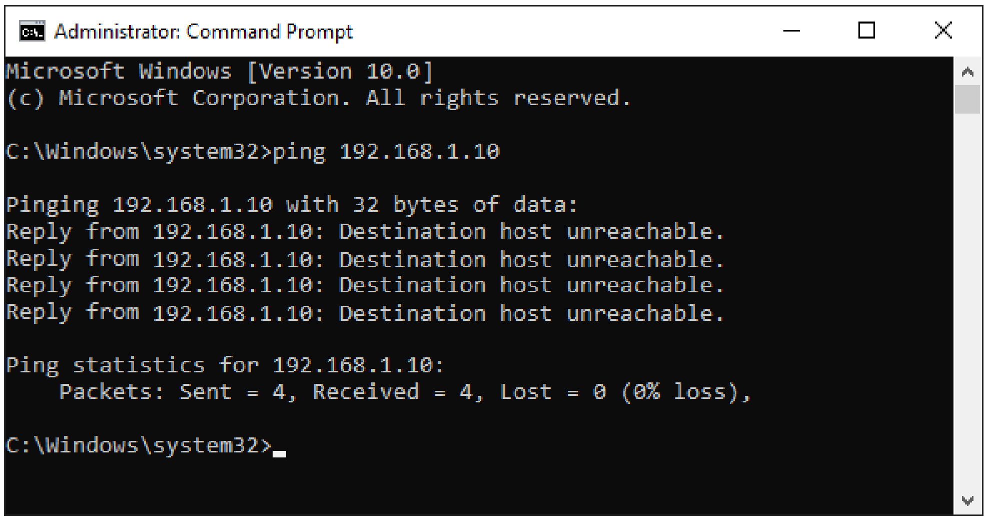
2. Enter the command "ping" followed by the IP address of the communication partner and press <ENTER>.

→ With an existing connection you receive several replies.



```
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: bytes=32 time=1ms TTL30
Reply from 192.168.1.10: bytes=32 time<1ms TTL30
Reply from 192.168.1.10: bytes=32 time<1ms TTL30
Reply from 192.168.1.10: bytes=32 time<1ms TTL30

Ping statistics for192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Windows\system32>_
```

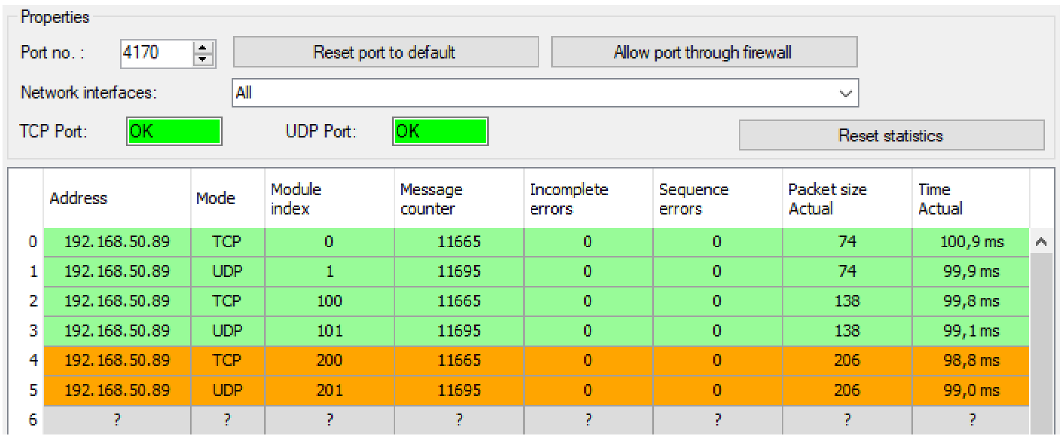
→ With no existing connection you receive error messages.



```
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>ping 192.168.1.10

Pinging 192.168.1.10 with 32 bytes of data:
Reply from 192.168.1.10: Destination host unreachable.
Reply from 192.168.1.10: Destination host unreachable.
Reply from 192.168.1.10: Destination host unreachable.
Reply from 192.168.1.10: Destination host unreachable.

Ping statistics for 192.168.1.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

C:\Windows\system32>_
```

4.5 Checking the connection

After applying the configuration, the connection overview shows all connections sorted by module index.



Port no. : 4170

Network interfaces: All

TCP Port: OK UDP Port: OK

	Address	Mode	Module index	Message counter	Incomplete errors	Sequence errors	Packet size Actual	Time Actual
0	192.168.50.89	TCP	0	11665	0	0	74	100,9 ms
1	192.168.50.89	UDP	1	11695	0	0	74	99,9 ms
2	192.168.50.89	TCP	100	11665	0	0	138	99,8 ms
3	192.168.50.89	UDP	101	11695	0	0	138	99,1 ms
4	192.168.50.89	TCP	200	11665	0	0	206	98,8 ms
5	192.168.50.89	UDP	201	11695	0	0	206	99,0 ms
6	?	?	?	?	?	?	?	?

Colors

Green	The connection is OK. The *ibaPDA* module timebase is the same or slower than the telegram cycle. The current telegram cycle is indicated in the "Time Actual" column.
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.

Connection errors

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

- S7 is in stop
- No Ethernet connection between *ibaPDA*-PC and S7 CP
- Error in the S7/NetPro configuration:
  - incorrect remote IP address
  - The port number and the S7 connection do not match.
  - The port number is blocked by the firewall.
- Wrong module index specified in the telegram header.

Other errors

- If the message counters do not increment continuously, the send blocks AG_LSEND are not called cyclically on the S7 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 DB does not have the expected value.
  - The "sequence_counter" in the DB is not incremented correctly.
  - The "delayed acknowledge" problem occurs (see ↗ *TCP performance problems caused by Delayed Acknowledge*, page 38).

4.6 Diagnostic modules

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

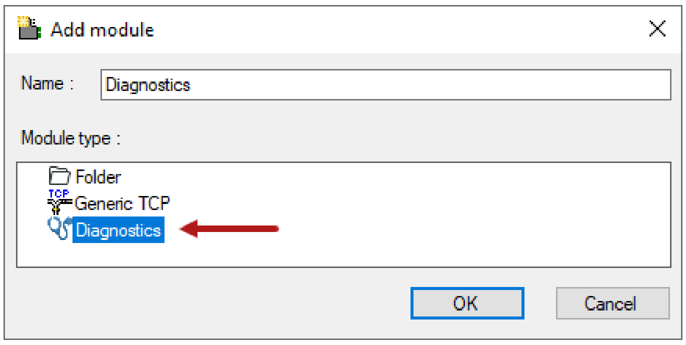
A diagnostic module is always assigned to a data acquisition module of the same interface and supplies its connection information. By using a diagnostic module you can record and analyze the diagnostic information continuously in the *ibaPDA* system.

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

Example for the use of diagnostic modules:

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

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

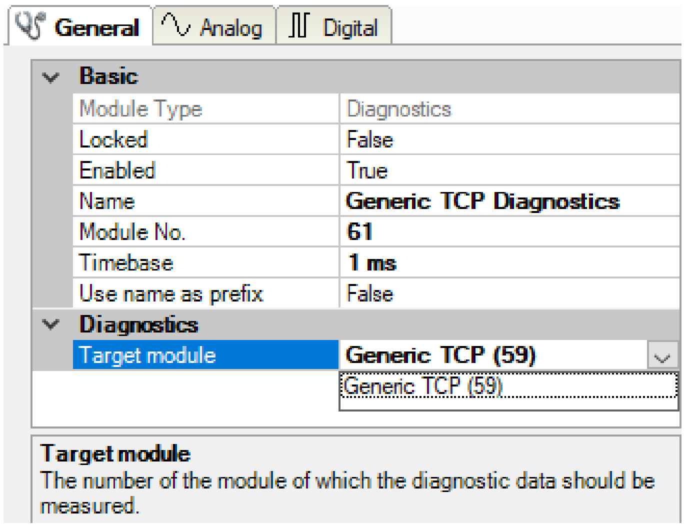

Add module

Name : Diagnostics

Module type :

Module settings diagnostic module

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


General | Analog | Digital

Basic

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

Diagnostics

Target module	Generic TCP (59)

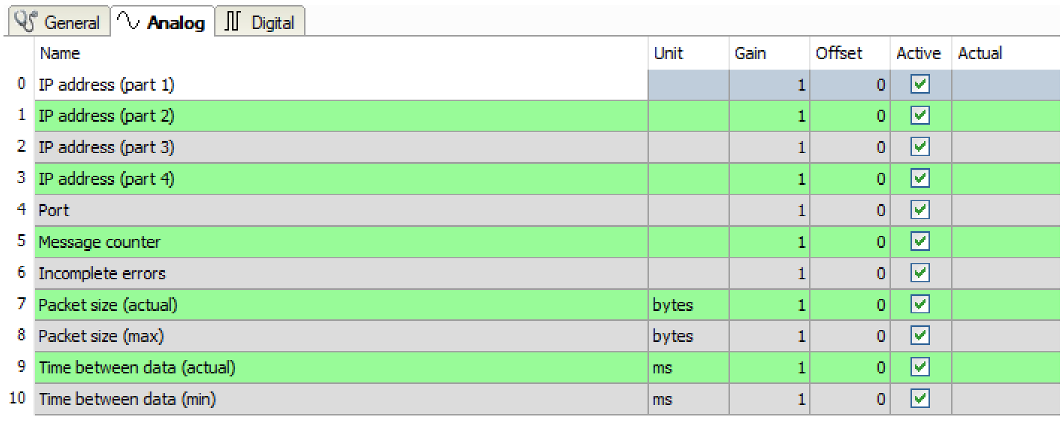
Generic TCP (59)

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

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

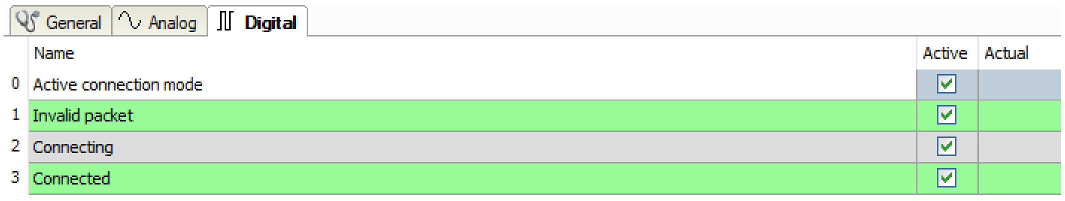
There is only one setting which is specific for the diagnostic module: the target module.

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


General | Analog | Digital

	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.


General | Analog | Digital

	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. 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
Incorrect message type	Number of received messages with wrong message type

Signal name	Description
Input data length	Length of data messages with input signals in bytes (*ibaPDA* receives)
Invalid packet	Invalid data packet detected
IP address (part 1-4)	4 octets of the IP address of the target system
Keepalive counter	Number of KeepAlive messages received by the OPC UA Server
Lost images	Number of lost images (in) that were not received even after a retransmission
Lost Profiles	Number of incomplete/incorrect profiles
Message counter	Number of messages received
Messages per cycle	Number of messages in the cycle of the update time
Messages received since configuration	Number of received data telegrams (in) since start of acquisition
Messages received since connection start	Number of received data telegrams (in) since the start of the last connection setup. Reset with each connection loss.
Messages sent since configuration	Number of sent data telegrams (out) since start of acquisition
Messages sent since connection start	Number of sent data telegrams (out) since the start of the last connection setup. Reset with each connection loss.
Multicast join error	Number of multicast login errors
Number of request commands	Counter for request messages from *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. Actual: current value 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 Actual: between the last two messages Max/min: statistical values since start of acquisition or reset of counters
Time offset (actual)	Measured time difference of synchronicity between *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 (actual) can be higher than the set value, if the PLC needs more time to transfer the data. How fast the data is really updated, you can check in the connection table. The minimum achievable update time is influenced by the number of signals. The more signals are acquired, the greater the update time becomes. Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Write counter	Number of successful write accesses
Write lost counter	Number of failed write accesses

5 Appendix

5.1 Troubleshooting

5.1.1 TCP performance problems caused by Delayed Acknowledge

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).

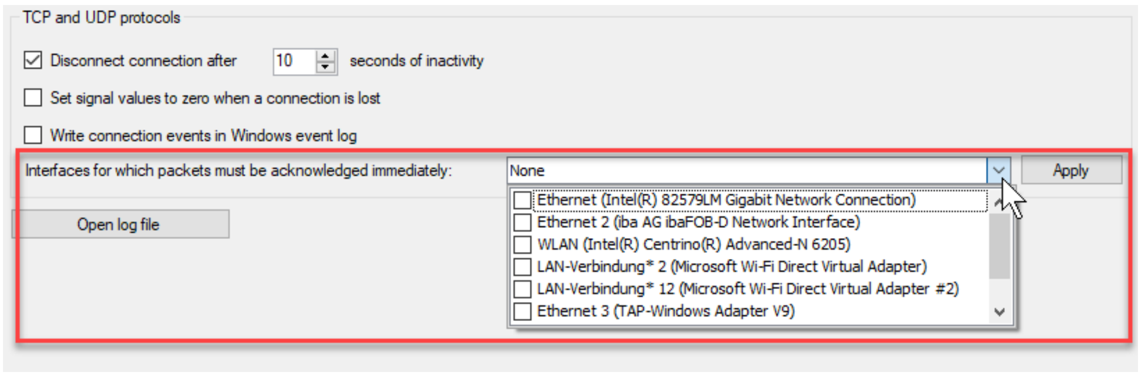
The data flow is controlled by a "sliding window" (parameter Win=nnnn). The recipient specifies how many bytes it can receive without sending an acknowledgment.

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 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>.


TCP and UDP protocols

☑ Disconnect connection after 10 seconds of inactivity

☐ Set signal values to zero when a connection is lost

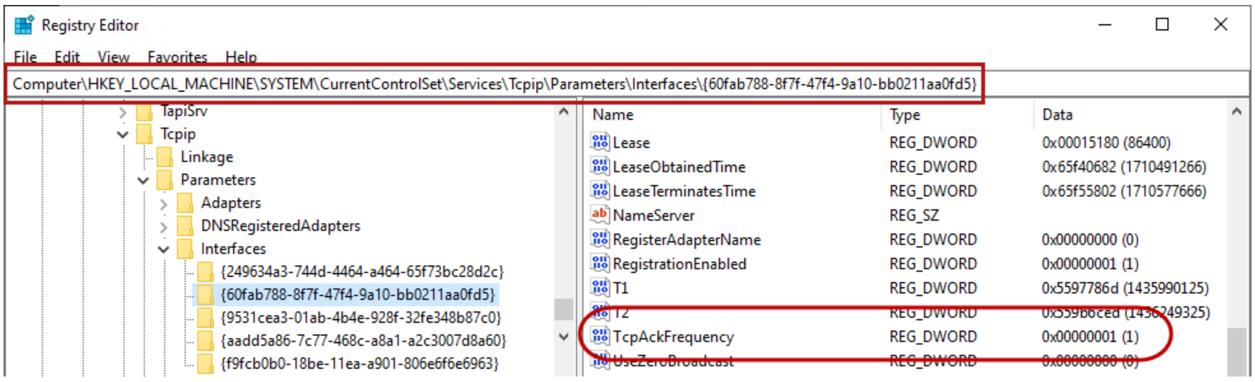
☐ Write connection events in Windows event log

Interfaces for which packets must be acknowledged immediately: None

☐ Ethernet (Intel(R) 82579LM Gigabit Network Connection)
☐ Ethernet 2 (iba AG ibaFOB-D Network Interface)
☐ WLAN (Intel(R) Centrino(R) Advanced-N 6205)
☐ LAN-Verbindung* 2 (Microsoft Wi-Fi Direct Virtual Adapter)
☐ LAN-Verbindung* 12 (Microsoft Wi-Fi Direct Virtual Adapter #2)
☐ Ethernet 3 (TAP-Windows Adapter V9)

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}
```



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{60fab788-8f7f-47f4-9a10-bb0211aa0fd5}

- TapiSrv
- Tcpip
  - Linkage
  - Parameters
    - Adapters
    - DNSRegisteredAdapters
    - Interfaces
      - {249634a3-744d-4464-a464-65f73bc28d2c}
      - {60fab788-8f7f-47f4-9a10-bb0211aa0fd5}
      - {9531cea3-01ab-4b4e-928f-32fe348b87c0}
      - {aadd5a86-7c77-468c-a8a1-a2c3007d8a60}
      - {f9fcb0b0-18be-11ea-a901-806e6f6e6963}

Name	Type	Data
Lease	REG_DWORD	0x00015180 (86400)
LeaseObtainedTime	REG_DWORD	0x65f40682 (1710491266)
LeaseTerminatesTime	REG_DWORD	0x65f55802 (1710577666)
NameServer	REG_SZ	
RegisterAdapterName	REG_DWORD	0x00000000 (0)
RegistrationEnabled	REG_DWORD	0x00000001 (1)
T1	REG_DWORD	0x5597786d (1435990125)
T2	REG_DWORD	0x559b6ced (1436249325)
TcpAckFrequency	REG_DWORD	0x00000001 (1)
UseZeroBroadcast	REG_DWORD	0x00000000 (0)

Note

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

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

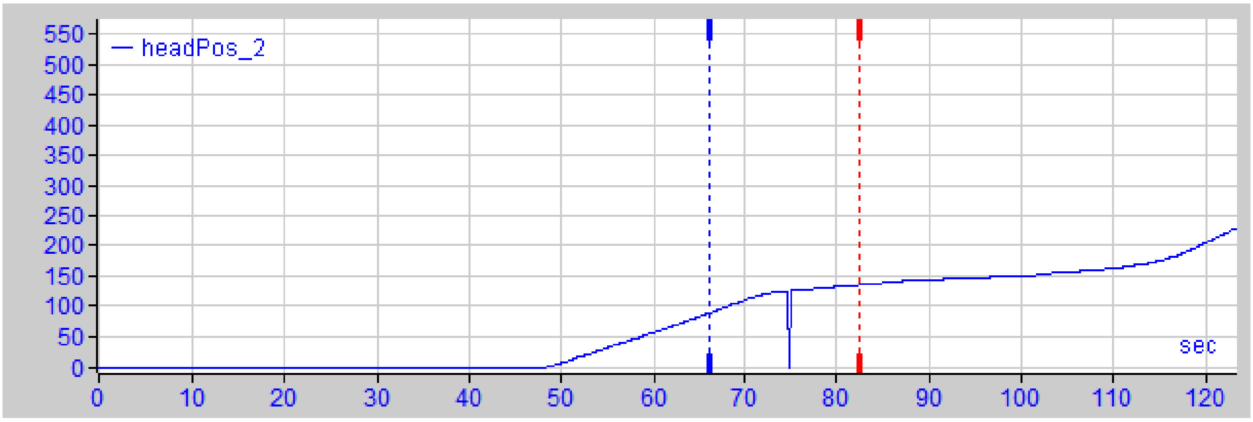
5.1.2 TCP data corruption resulting from the Nagle’s Algorithm

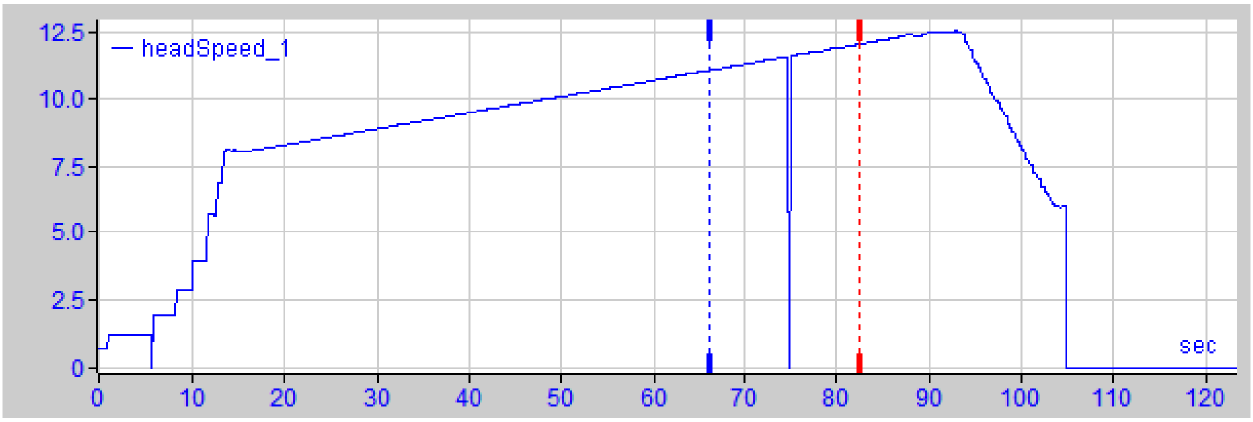
Symptoms:

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

Errors shown in ibaPDA:

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





Cause:

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

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

But the Nagle's Algorithm and the option *Delayed ACK* (Delayed Acknowledge, see 5.1.1, page 38) do not play well together in a TCP/IP Network:

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

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

Remedy:

It is recommended to start with disabling the *Delayed ACK* mechanism as explained in chapter 5.1.1, page 38. In a typical real-time application, the transmitter will then send the new data to *ibaPDA* with a certain cycle time because the previous data has been acknowledged immediately. Depending on the implementation of the TCP/IP stack on the sender's side, the Nagle's algorithm can still become active and automatically aggregate a number of small buffer messages, causing the algorithm to purposely slow down the transmission.

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

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

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

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

Note

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

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

6 Support and contact

Support

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

Note

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

Contact

Headquarters

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

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

Mailing address

iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address

iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany

Regional and Worldwide

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

www.iba-ag.com