Manufacturer
iba AG
Koenigswarterstr. 44
90762 Fuerth
Germany

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

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

© iba AG 2020, All rights reserved.

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

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

Version Date Revision - Chapter / Page Author Version SW
2.10 09-2020 3.1.2 New graphic, 4.6 Diagnostic modules RM/IP 6.34

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

1 About this Manual ................................................................. 5
  1.1 Target group and previous knowledge ....................................... 5
  1.2 Notations ........................................................................ 5
  1.3 Used symbols .................................................................. 6

2 System requirements .......................................................... 7

3 Modbus-TCP-Server data interface ....................................... 8
  3.1 General information .......................................................... 8
  3.1.1 Modbus TCP/IP .............................................................. 8
  3.1.2 Client/Server architecture ................................................. 9
  3.1.3 Modbus protocol ............................................................ 9
  3.1.4 Modbus TCP/IP - Message layout ...................................... 12
  3.1.4.1 Modbus Integer and Modbus Dig512 ............................... 12
  3.1.4.2 Modbus Real ................................................................. 13
  3.1.4.3 Modbus Generic .......................................................... 13
  3.1.4.4 Response ................................................................. 14
  3.1.5 References .................................................................. 14
  3.2 Configuration and engineering ibaPDA .................................. 15
  3.2.1 General settings ............................................................ 15
  3.2.2 General interface settings ............................................... 16
  3.2.3 General module settings ................................................ 18
  3.2.4 General signal configuration .......................................... 19
  3.2.5 Module type "Integer" ...................................................... 19
  3.2.6 Module type "Dig512" ..................................................... 20
  3.2.7 Module type "Real" ......................................................... 20
  3.2.8 Module type "Generic" .................................................... 20
  3.2.9 Module diagnostics ....................................................... 21

4 Diagnostics ............................................................................. 23
  4.1 License check ................................................................... 23
  4.2 Visibility of the interface .................................................... 23
  4.3 Log files .......................................................................... 24
  4.4 Connection diagnostics with PING ..................................... 25
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Checking the connection</td>
<td>26</td>
</tr>
<tr>
<td>4.6</td>
<td>Diagnostic modules</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td><strong>Appendix</strong></td>
<td>31</td>
</tr>
<tr>
<td>5.1</td>
<td>Restrictions</td>
<td>31</td>
</tr>
<tr>
<td>5.1.1</td>
<td>TCP/IP protocol variants</td>
<td>31</td>
</tr>
<tr>
<td>5.2</td>
<td>Engineering examples</td>
<td>33</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Engineering example Modicon Quantum</td>
<td>33</td>
</tr>
<tr>
<td>5.2.1.1</td>
<td>Configuration of the TCP/IP Interface in ProWORX NxT</td>
<td>33</td>
</tr>
<tr>
<td>5.2.1.2</td>
<td>Ladder Program for the PLC</td>
<td>35</td>
</tr>
<tr>
<td>5.2.1.3</td>
<td>ConCept Program for the PLC</td>
<td>38</td>
</tr>
<tr>
<td>5.2.1.4</td>
<td>Unity Pro XL Program for the PLC (Generic module sample)</td>
<td>40</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Engineering example in PL7 Pro</td>
<td>42</td>
</tr>
<tr>
<td>5.2.2.1</td>
<td>Network configuration</td>
<td>42</td>
</tr>
<tr>
<td>5.2.2.2</td>
<td>Message configuration (example)</td>
<td>43</td>
</tr>
<tr>
<td>6</td>
<td><strong>Support and contact</strong></td>
<td>45</td>
</tr>
</tbody>
</table>
1 About this Manual

This document describes the function and application of the software interface
ibaPDA-Interface-Modbus-TCP-Server

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

1.1 Target group and previous knowledge

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

This documentation in particular addresses persons, who are concerned with the configuration,
test, commissioning or maintenance of Programmable Logic Controllers of the supported prod-
ucts. For the handling of ibaPDA-Interface-Modbus-TCP-Server 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:

<table>
<thead>
<tr>
<th>Action</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu command</td>
<td>Menu <em>Logic diagram</em></td>
</tr>
<tr>
<td>Calling the menu command</td>
<td><em>Step 1 – Step 2 – Step 3 – Step x</em></td>
</tr>
<tr>
<td></td>
<td>Example: Select the menu <em>Logic diagram - Add - New function block</em>.</td>
</tr>
<tr>
<td>Keys</td>
<td>&lt;Key name&gt;</td>
</tr>
<tr>
<td></td>
<td>Example: &lt;Alt&gt;; &lt;F1&gt;</td>
</tr>
<tr>
<td>Press the keys simultaneously</td>
<td>&lt;Key name&gt; + &lt;Key name&gt;</td>
</tr>
<tr>
<td></td>
<td>Example: &lt;Alt&gt; + &lt;Ctrl&gt;</td>
</tr>
<tr>
<td>Buttons</td>
<td>&lt;Key name&gt;</td>
</tr>
<tr>
<td></td>
<td>Example: &lt;OK&gt;; &lt;Cancel&gt;</td>
</tr>
<tr>
<td>File names, paths</td>
<td>&quot;Filename&quot;, &quot;Path&quot;</td>
</tr>
<tr>
<td></td>
<td>Example: &quot;Test.doc&quot;</td>
</tr>
</tbody>
</table>
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 Modbus-TCP-Server data interface:

- *ibaPDA* v6.33.2 or higher
- License for *ibaPDA-Interface-Modbus-TCP-Server*
- Network connection 10/100 Mbits

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

**Note**

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

**System restrictions**

- The maximum length of a Modbus TCP/IP message is limited to 244 bytes.
- For different ways of handling the TCP/IP acknowledge, see *TCP/IP protocol variants*, page 31

**Licenses**

<table>
<thead>
<tr>
<th>Order No.</th>
<th>Product name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.001020</td>
<td><em>ibaPDA-Interface-Modbus-TCP-Server</em></td>
<td>Extension license for an <em>ibaPDA</em> system providing an additional Modbus-TCP-Server interface. Number of connections: 64</td>
</tr>
<tr>
<td>31.101020</td>
<td>one-step-up-Interface-Modbus over TCPIP-Client</td>
<td>Extension license for the extension of an existing interface by another 64 Modbus-TCP-Server connections, max. 3 permitted</td>
</tr>
</tbody>
</table>
3 Modbus-TCP-Server data interface

3.1 General information

3.1.1 Modbus TCP/IP

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite.

IP handles lower-level transmissions from computer to computer as a message makes its way across the Internet. TCP operates at a higher level (transport level), concerned with the two end systems. TCP provides a reliable data stream of bytes from a program on one computer to another program on another computer. TCP is explained in chapter 6 of RFC1180 and in RFC768, see References, page 14.

Modbus is a protocol for the client/server communication between devices connected on different types of buses or networks.

Modbus is currently implemented in the following buses or networks as shown in the following figure:

- TCP/IP over Ethernet
- Asynchronous serial transmission over a variety of media
- Modbus PLUS (a high speed communication via a token passing network)

ibaPDA has the possibility to measure signals via the Modbus protocol over serial connections (Modbus ASCII and Modbus RTU) and over TCP/IP. This manual describes the connection via TCP/IP and as variant the transmission of the Modbus RTU protocol over TCP/IP, with ibaPDA acting as client.

All systems that can receive and respond to messages with the Modbus-TCP protocol as server, can also communicate with ibaPDA.
3.1.2 Client/Server architecture

The Modbus service supports a client/server communication for devices which are connected via Ethernet TCP/IP.

The client/server model is based on 4 message types:

- Request
- Indication
- Response
- Confirmation

Read data: The Modbus-TCP-Client (ibaPDA) establishes the connection to the Modbus server, sends periodically the request and waits for the response, which contains the requested data.

Write data: The Modbus-TCP-Client (ibaPDA) establishes the connection to the Modbus server which contains the output data and waits for the response.

The port 502 is used for the Modbus TCP/IP communication by default, however you have got the possibility to enter other port numbers in ibaPDA.

With a ibaPDA-Interface-Modbus-TCP-Server license, ibaPDA can receive up to 64 connections, i.e. up to 64 Modbus servers can establish connections to ibaPDA. The number can be extended to a max. of 256 by loading the license more than once.

3.1.3 Modbus protocol

Byte sequence

Modbus uses "BIG ENDIAN", i.e. in the messages the bytes with a high significance are sent first and are thus stored in the addresses of low significance.

ibaPDA swaps all received 16- and 32-bit-values to the Intel format "LITTLE ENDIAN" ("Swapping"). You can select the Swapping method in ibaPDA for data that do not come from a Modbus controller. See ➔ General interface settings, page 16.

Modbus RTU / Modbus TCP

In the following representation, you can see the basic structure of the Modbus protocol and the differences between Modbus RTU and Modbus TCP.
For Modbus TCP, the MBAP Header is put in front of the function code. The Unit Id. corresponds to the Slave Id. of the RTU protocol. The CRC code is omitted.

### MBAP Header

The MBAP Header is a dedicated header used for the communication with TCP/IP to identify the Modbus Application Data.

The header contains the following fields:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Id.</td>
<td>2</td>
<td>Identification of Modbus request/response transaction</td>
</tr>
<tr>
<td>Protocol Id.</td>
<td>2</td>
<td>0 = Modbus protocol</td>
</tr>
<tr>
<td>Length</td>
<td>2</td>
<td>Number of following bytes</td>
</tr>
<tr>
<td>Unit Id.</td>
<td>1</td>
<td>Addressing a remote slave connected to the Modbus server</td>
</tr>
</tbody>
</table>

- **Transaction Identifier**: It is used for transaction pairing.
  The Modbus client sends it in the request; the Modbus server copies in the response message the transaction identifier of the request.
  *ibaPDA* evaluates this field as sequence count and waits for the value to be incremented by 1 in each cycle. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000) or from 65535 to 0 (0xFFFF → 0x0000).

- **Protocol Identifier**: It is used in multiplexing procedures. The Modbus protocol has the value 0.

- **Length**
  The length field is a count of the following bytes, including the Unit Identifier, Function Code and Data Fields. The maximum value is 251 (max. length of the user data bytes 244 + 7).

- **Unit Identifier (device address)**:
  This field is sent by the Modbus client in the request and must be returned with the same value in the response by the server.
  This field is not evaluated by *ibaPDA*. 
**Function code:**
One byte contains the function code that determines which function the server has to carry out depending on the request.

The *ibaPDA-Interface-Modbus-TCP-Server* driver supports only the function
- 0x10: Write Multiple Registers

**Data fields**
The user data fields contain several subfields like starting address, number of registers, number of bytes and the actual data. The content of these fields depends on the used function code. For the function code 0x10, the data fields contain the following values:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting address</td>
<td>2</td>
<td>Starting address of the used storage area</td>
</tr>
<tr>
<td>Number of objects</td>
<td>2</td>
<td>Number of used registers or coils</td>
</tr>
<tr>
<td>Number of bytes</td>
<td>1</td>
<td>Number of data</td>
</tr>
<tr>
<td>Data range</td>
<td>n</td>
<td>n data bytes</td>
</tr>
</tbody>
</table>

- **Starting address**
  The *ibaPDA-Interface-Modbus-TCP-Server* driver uses the Modbus starting address. The starting address, that is called "Module index" in *ibaPDA*, is a number in which the data are assigned to a data module.

  In *ibaPDA*, 4 module types are defined:
  - Integer: 32 analog values (integer) and 32 binary signals
  - Real: 32 analog values (real) and 32 binary signals
  - Generic: data structure with a maximum length of 244 bytes.
  - Dig512: 512 Binary signals (32 status words with 16 bits, each)

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

<table>
<thead>
<tr>
<th>Module type</th>
<th>1st License</th>
<th>2nd License</th>
<th>3rd License</th>
<th>4th License</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer and Dig512</td>
<td>0-63</td>
<td>1000-1063</td>
<td>2000-2063</td>
<td>3000-3063</td>
</tr>
<tr>
<td>Real</td>
<td>100-163</td>
<td>1100-1163</td>
<td>2100-2163</td>
<td>3100-3163</td>
</tr>
<tr>
<td>Generic</td>
<td>200-263</td>
<td>1200-1263</td>
<td>2200-2263</td>
<td>3200-3263</td>
</tr>
</tbody>
</table>

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

- **Number of objects**: This field shows the number of registers that are transmitted in one message. Modbus Integer and Modbus Dig512 send 34 registers. A Modbus Real module sends 66 registers. In the Generic module, the number of registers can be varied. However, it is limited to a max of 122. In this case, you have to enter the number of registers that are to be sent.

- **Number of bytes**: This value is always the number of registers multiplied by 2, as the registers are word-based (2 bytes). The maximum number is 244.
Data range: The field contains the actual data sent to *ibaPDA*. The data type depends on the used Modbus module. Each module type has a maximum number of values that can be sent. An exception is the Generic module. In the Generic module, all available data types can be used simultaneously. In *ibaPDA*, you only have to configure the address of the signal and the data type.

### 3.1.4 Modbus TCP/IP - Message layout

The Modbus messages have the following module layout, corresponding to the module type:

#### 3.1.4.1 Modbus Integer and Modbus Dig512

For the Integer module type, the 32 analog values are of the Integer type (16-bit) and the 32 digital values are densely packed as DWORD.

For the Dig512 module type, the 32 analog values are evaluated as 16-bit status words. The DWORD is not used.

**Request Modbus Client -> ibaPDA (Modbus Server):**

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>Remark:</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>2</td>
<td>UINT</td>
<td>Transaction Id.</td>
<td>xx xx</td>
<td>Is evaluated as sequence count by <em>ibaPDA</em>, i.e. the ID has to be incremented each cycle.</td>
</tr>
<tr>
<td>MBAP</td>
<td>02</td>
<td>UINT</td>
<td>Protocol Id.</td>
<td>00 00 0</td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>2</td>
<td>UINT</td>
<td>Cmd Length</td>
<td>4B 75</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>Bytes</td>
<td>Unit-ID</td>
<td>xx</td>
<td>not used</td>
</tr>
<tr>
<td>Fcode</td>
<td>07</td>
<td>Bytes</td>
<td>Function Code</td>
<td>10</td>
<td>&quot;Write Multiple Registers&quot;</td>
</tr>
<tr>
<td>Data</td>
<td>08</td>
<td>UINT</td>
<td>Starting Address</td>
<td>xx xx</td>
<td>Module index i000 – i063</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>UINT</td>
<td>Number of objects</td>
<td>22</td>
<td>34 number of registers</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Bytes</td>
<td>Number of bytes</td>
<td>44</td>
<td>68 Number of bytes</td>
</tr>
<tr>
<td>User</td>
<td>13</td>
<td>INT</td>
<td>Data</td>
<td>xx</td>
<td>32 analog values</td>
</tr>
<tr>
<td>data</td>
<td>77</td>
<td>DWORD</td>
<td>Data</td>
<td>xx</td>
<td>32 digital values</td>
</tr>
</tbody>
</table>
3.1.4.2 Modbus Real

The analog values are of the FLOAT (IEEE format) type and the 32 digital values are densely packed as DWORD.

**Request Modbus Client -> ibaPDA (Modbus Server):**

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>Remark:</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>2</td>
<td>UINT</td>
<td>Transaction Id.</td>
<td>xx xx</td>
<td>Is evaluated by the <em>ibaPDA</em> sequence count</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>UINT</td>
<td>Protocol Id.</td>
<td>00 00</td>
<td>0</td>
</tr>
<tr>
<td>04</td>
<td>2</td>
<td>UINT</td>
<td>Cmd Length</td>
<td>8B</td>
<td>139</td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>Bytes</td>
<td>Unit-ID</td>
<td>xx</td>
<td>not used</td>
</tr>
<tr>
<td>Fcode</td>
<td>07</td>
<td>1</td>
<td>Bytes</td>
<td>10</td>
<td>&quot;Write Multiple Registers&quot;</td>
</tr>
<tr>
<td>Data</td>
<td>08</td>
<td>2</td>
<td>Starting Address</td>
<td>xx xx</td>
<td>Module index i100 to i163</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>Number of objects</td>
<td>42</td>
<td>66 number of registers</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>1</td>
<td>Number of bytes</td>
<td>84</td>
<td>132 Number of bytes</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>128</td>
<td>FLOAT</td>
<td>xx</td>
<td>32 analog values</td>
</tr>
<tr>
<td></td>
<td>141</td>
<td>4</td>
<td>DWORD</td>
<td>xx</td>
<td>32 digital values</td>
</tr>
</tbody>
</table>

3.1.4.3 Modbus Generic

The user data area can have any desired data structure with different data formats. *ibaPDA* supports the following data formats:

BYTE, WORD, DWORD, INT, DINT and FLOAT.

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

**Request Modbus Client -> ibaPDA (Modbus Server):**

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>Remark:</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>2</td>
<td>UINT</td>
<td>Transaction Id.</td>
<td>xx xx</td>
<td>Is evaluated by the <em>ibaPDA</em> sequence count</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>UINT</td>
<td>Protocol Id.</td>
<td>00 00</td>
<td>0</td>
</tr>
<tr>
<td>04</td>
<td>2</td>
<td>UINT</td>
<td>Cmd Length</td>
<td>xx</td>
<td>n + 7</td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>Bytes</td>
<td>Unit-ID</td>
<td>xx</td>
<td>not used</td>
</tr>
<tr>
<td>Fcode</td>
<td>07</td>
<td>1</td>
<td>Bytes</td>
<td>10</td>
<td>&quot;Write Multiple Registers&quot;</td>
</tr>
</tbody>
</table>
### 3.1.4.4 Response

Each telegram is answered by the Modbus server with a response telegram by default. You can suppress this function in ibaPDA.

#### Response ibaPDA (Modbus Server) → Modbus Client:

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Content (hex)</th>
<th>ibaPDA Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>2</td>
<td>UINT</td>
<td>Transaction Id.</td>
<td>xx xx</td>
<td>Sequence count, Mirror of request</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>UINT</td>
<td>Protocol Id.</td>
<td>00 00</td>
<td>0</td>
</tr>
<tr>
<td>04</td>
<td>2</td>
<td>UINT</td>
<td>Cmd Length</td>
<td>00 06</td>
<td>6</td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>Bytes</td>
<td>Unit-ID</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>07</td>
<td>1</td>
<td>Bytes</td>
<td>Function Code</td>
<td>10</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>08</td>
<td>2</td>
<td>UINT</td>
<td>Starting Address</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>UINT</td>
<td>Number of objects</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
</tbody>
</table>

### 3.1.5 References

Other documentation

- ibaPDA manual
- Modbus Messaging Implementation Guide V1 (http://www.modbus.org)
- Modbus Application Protocol V1.1 (http://www.modbus.org)
3.2 Configuration and engineering ibaPDA

Subsequently, the engineering for *ibaPDA* is described. If all system requirements are satisfied, the interface "Modbus TCP Server" is displayed in the signal tree. There is no need to add the interface manually.

3.2.1 General settings

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

**Disconnect connection after x seconds of inactivity:**
Behavior and timeout duration can be specified.

**Set signal values to zero when a connection is lost:**
If this option is disabled, the value read last will be kept.
3.2.2 General interface settings

You find the "Modbus TCP Server" in the tree structure of the ibaPDA I/O manger, under the Hardware menu option. The interface provides the following functions and configuration options:

Use network byte order on network (= big endian, default setting)
Optional:

Use little endian byte order on network
see ➔ Modbus protocol, page 9.

Swap float values on word base
Here, you can change the byte order for data that are not provided by Modbus devices. The bytes are swapped according to the pattern ABCD → CDAB. This option concerns only the module type Real; for the module type Generic, specific setting options are defined.

Swap digital signals
Here, you can change the byte order for data that are not provided by Modbus devices. The bytes are swapped according to the pattern ABCD → BADC.

Send response to modbus master
Each telegram is acknowledged by a response telegram, see ➔ Response, page 14. You can suppress the response by de-activating this option.

Ignore sequence counter
If this option is enabled, the "Sequence errors" column is hidden in the connection overview.

<Reset statistics>

With a click on this button, you can reset the following values in the connection overview: "Message count", "Incomplete errors" and "Sequence errors".

Overview of connections:
As soon as the connection has been established, you can see the live data in the overview. See ➔ Checking the connection, page 26.

Adding a module
Select the Hardware option in the I/O manager and add a module under the "Modbus TCP Server" interface by clicking on the module type and selecting it.
Tip

If a TCP/IP connection already exists, right-click the interface and select "Autodetect". Then, the correct modules are automatically created for all available connections.
3.2.3 General module settings

If you want to configure a Modbus data module, mark the module in the tree structure, select the General tab and do the following settings in the dialog. All modules have the following common setting options:

### Basic settings

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

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

**Enabled**
Disabled modules are excluded from signal acquisition.

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

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

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

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

### 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. 

For a detailed description of the parameters, see the ibaPDA manual.
3.2.4 General signal configuration

The data to be measured are selected on the Modbus client side by mapping the signals to the respective data ranges.

In the ibaPDA I/O manager, you can assign name, unit, scaling (only analog signals) and comments to the signals. Moreover, you can enable and disable the signals.

Analog signals

![Analog signals](image)

Digital signals

![Digital signals](image)

Tip

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

Other documentation

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

3.2.5 Module type "Integer"

The "Integer" module allows up to 32 analog signals (Integer) and 32 binary signals to be acquired.

The module does not have any module specific settings.
3.2.6 Module type "Dig512"

With the "Dig512" module, you can acquire up to 512 digital values, organized as 32 status words (type Integer) with 16 bits, each.

The module does not have any module specific settings, there are only the General and Digital tabs.

3.2.7 Module type "Real"

The "Real" module allows up to 32 analog signals (Real) and 32 binary signals to be acquired.

The following module settings are module-specific:

- No. analog signals
  You can set up the number of analog signals to be measured (1 to 32).
  Please, consider the following:
  In case you want to use digital signals, the message must have the structure described in Modbus Real, page 13. If you do not use digital signals, the message can be shortened.

- In the analog table, Gain/Offset are scaled.

3.2.8 Module type "Generic"

Any data block with a max. length of 244 bytes can be measured by means of the module "Generic".

The following module settings are module-specific:

- Swap analog signals, swap digital signals
  You can change the byte evaluation order. (The interface settings are not valid here!)
No. analog signals, No. digital signals
Maximum number of analog and digital signals that can be configured.

In the analog table, Gain/Offset are scaled.

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

**Description of the columns:**

- **Name, Unit, Gain, Offset, Active:** see *ibaPDA* manual.
- **Address:**
  The address defines the byte offset of the value within the user data of the telegram.
- **Data type:**
  The following data types are supported: BYTE, WORD, DWORD, INT, DINT, FLOAT.

**Tip**

Please consider, that the address depends on the data type of the preceding data. This is why we recommend setting the data types and then the addresses using the autofill function.

### 3.2.9 Module diagnostics

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

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 18)
  - 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 18)
- The telegrams do not arrive faster than approx. 200 ms with sequence error:
  - Problem with “Delayed Acknowledge”, see TCP/IP protocol variants, page 31
4 Diagnostics

4.1 License check

If the interface "Modbus TCP Server" is not shown in the signal tree, you can check in the I/O manager under General - Settings - License info, if your license has been correctly detected. The number of licensed connections is shown in brackets.

4.2 Visibility of the interface

If the interface is not visible despite a valid license, it might be hidden. Click the Interfaces tab and enable the "Modbus TCP Server" module.
4.3 Log files

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

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

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

In the file system on the hard drive, you will find the log files in the program path of the ibaPDA server (\Programs\iba\ibaPDA\Server\Log\). The file names of the log files include the name or abbreviation of the interface type.

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

Examples:

- ethernetipLog.txt (log of EtherNet/IP connections)
- AbEthLog.txt (log of Allen-Bradley Ethernet connections)
- OpcUAServerLog.txt (log of OPC UA server connections)
4.4 Connection diagnostics with PING

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

Open a Windows command prompt.

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

With an existing connection you receive several replies.

![Fig. 1: PING successful](image1.png)

With no existing connection you receive error messages.

![Fig. 2: PING unsuccessful](image2.png)
4.5 Checking the connection

If you mark the data interface "Modbus TCP Server" in the signal tree of the I/O manager, you will see a table in the right part of the window which shows all available connections of this interface.

Buttons:
- With <Reset statistics>, you can reset the telegram counter, the "Incomplete errors" and "Sequence errors".

The list of connections shows the following values:
- IP address: Address of the Modbus server
- Module index: "Starting address" field from the telegram header.
- Incomplete errors: Is incremented each time the length of the telegram does not equal the length defined in the telegram header.
- Sequence errors: Is incremented each time, the counter in the "Transaction Id." field of the header is not incremented each cycle by 1.
- Packet size Actual: The complete telegram length
- Time Actual: Cycle in which the telegrams of the Modbus client arrive

Colors:
- Green: Connection OK. the "Time Actual" approximately corresponds to the timebase of the module
- Orange: The connection is OK, the timebase of the module is much faster than "Time Actual". The timebase of the module can be adapted for optimizing purposes.
A failed connection may have the following causes:

- Modbus client is in stop
- No Ethernet connection between ibaPDA PC and the Modbus PLC
- Error in the connection configuration:
  - incorrect remote IP address
  - The ibaPDA port number and the connection configuration do not match.

Other errors:

- If values in the columns "Incomplete errors" and/or "Sequence errors" are incremented, this points to one of the following errors:
  - Error in the message header
  - Error in the byte order
  - The "delayed acknowledge" problem occurs, see TCP/IP protocol variants, page 31
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, since 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.

Fig. 3: Add diagnostic module, example Generic TCP
Module settings diagnostic module

For a diagnostic module, you can make the following settings:

![Module settings diagnostic module, example TCP Generic](image)

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.

![Example: Analog values of a diagnostic module for a TCP Generic module](image)

For example, the IP (v4-) address of a TCP Generic 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.
Fig. 6: Example: Digital values of a diagnostic module for a TCP Generic module
5 Appendix

5.1 Restrictions

5.1.1 TCP/IP protocol variants

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

Errors shown in IbaPDA:
Sequence errors and incomplete telegrams.

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. It specifies that the "acknowledge" is delayed until other telegrams arrive in order to acknowledge them jointly. If no other telegrams arrive, the ACK telegram is sent after 200 ms at the latest (depending on the socket).
The data flow is controlled by a "sliding window" (parameter Win=nnnn). The recipient specifies how many bytes it can receive without sending an acknowledgment.
Some controllers do not accept this response, but instead, wait for an acknowledgment after each data telegram. If it does not arrive within a certain period of time (200 ms), it will repeat the telegram and include any new data to be sent, causing an error with the recipient, because the old one was received correctly.

Remedy:
The "delayed acknowledge" can be switched off individually for each network adapter via an entry in the Windows Registry. For easy modification, IbaPDA offers a corresponding dialog in the I/O manager under General in the tab Settings.
In the list of network adapters, select those for which you want to disable "delayed acknowledge” and click <Apply>.
Thus, the parameter "TcpAckFrequency" (REG_DWORD = 1) is created in the registry path of the selected network adapters:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{InterfaceGUID\}
5.2  Engineering examples

5.2.1  Engineering example Modicon Quantum

5.2.1.1  Configuration of the TCP/IP Interface in ProWORX NxT

1. Install your Ethernet Module (NOE).
2. Configure your IP address for the NOE (next figure).

3. Make a new ladder or ConCept program placing a MSTR block configured as shown in the next figure.

Fig. 7: Configuration of the NOE IP-address
**MSTR operation Code:**
- *1 Decimal*; stands for a write command

**Error Status:**
- *0 hex*; stands for possible communication errors

**# of Registers:**
- *34 Decimal*; this stands for the 34 registers to be sent to *ibaPDA* (32 analog + 32 digital values). In case you use the MODBUS_FLOAT (or MODBUS Real) module type, 66 registers have to be sent. When a Modbus Generic module type is used, a maximum of 122 registers can be sent.

**Note**
Entering a number other than 34 or 66 (with Int, Dig512 and Float) will display errors both in *ibaPDA* and the MSTR block, without any communication at all.

**Func. Dependant Info:**
- *1 Decimal*; this stands for *ibaPDA* module number 1. Use different numbers for different *ibaPDA* module numbers. Add 100 when a MODBUS_FLOAT module is used and 200 when a MODBUS_Generic module is used.
MB+Routing A1:
In the example shown here, the NOE was installed in slot number 4. The actual number entered in this field is 0x400 whereas “4” represents the slot number (Decimal = 1024). For a NOE with slot number 5, the field will be 0x500, which equates to a decimal entry of 1280. This screenshot is taken from ProWORX Nxt.

MB+Routing A2..A5:
Stands for IP address 143.1.2.48; this should be the IP address of *ibaPDA*.

### 5.2.1.2 Ladder Program for the PLC

There are two ways to program the PLC:

**If *ibaPDA* doesn’t send an acknowledge answer**
- **Advantage:** Faster communication, there is less throughput in the LAN
- **Disadvantage:** Doesn’t work with NOE series 700
  - If set too fast, sometimes it can freeze the PLC

**ibaPDA sends an acknowledge answer**
- **Advantage:** Support of NOE series 700
- **Disadvantage:** Slower communication

For option 2 please select the “Send response to modbus master” option in the general settings of Modbus TCP Server, see *General interface settings*, page 16. If option 1 is preferable simply de-select it.
5.2.1.2.1 ibaPDA doesn't send an acknowledge answer (example)

Complete the program as shown in the figure below.

![Ladder program in ProWORX NxT](image)

Fig. 9: Ladder program in ProWORX NxT

T.01 is the timer generating pulses, in this case every 10 milliseconds.

Notice that the marked switch “P 001000” in the red square is the pulse output generated by T.01.

Notice also that ibaPDA does not send a response back to the PLC so the MSTR block is sending registers every 10ms even if errors occur or there is no response. Do not expect response from the communication.

When the PLC is restarted the MSTR block should establish the communication with ibaPDA. You should be able to see the connection status in ibaPDA Diagnostics for Modbus. Look for a few minutes what happens with the connection. If the connection is held for more than approx. 2 minutes, it should be working fine.

**Note**

One (1) sequence error displayed is ok. This is shown as the first intent to communicate with ibaPDA.
Every time the MSTR block sends data, the message counter is incremented in the connection overview (I/O manager ibaPDA), for the available active connection.

It is recommended to test live data with an up-counter which should be configured to write data into the first register sent by the MSTR block.

For more than 32 analog and 32 digital values, follow the MSTR block configuration procedure as explained but change the *Func. Dependant Info (ibaPDA module number)*.

In *ibaPDA* add Modbus Server modules as required.

5.2.1.2.2 *ibaPDA* sends an acknowledge answer (example)

Complete the program as shown in the figure below.

![Fig. 10: Ladder program in ProWORX NxT](image)

Notice that this time there is no timer that enables the MSTR block to write. Instead, after each successful transaction, *ibaPDA* sends an ACK answer back which returns a SUCCESSFUL output and prepares the MSTR block to be ready for the next one.
5.2.1.3 ConCept Program for the PLC

Basically the steps of configuration in ConCept are the same as described in the preceding sections.

In ConCept, the parameter names of the MSTR block differ.

<table>
<thead>
<tr>
<th>ProWORX NxT</th>
<th>ConCept</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSTR operation Code</td>
<td>w1</td>
</tr>
<tr>
<td>Error Status</td>
<td>w2</td>
</tr>
<tr>
<td># of Registers</td>
<td>w3</td>
</tr>
<tr>
<td>Func. Dependant Info</td>
<td>w4</td>
</tr>
<tr>
<td>MB+Routing A1</td>
<td>w5</td>
</tr>
<tr>
<td>MB+Routing A2</td>
<td>w6</td>
</tr>
<tr>
<td>MB+Routing A3</td>
<td>w7</td>
</tr>
<tr>
<td>MB+Routing A4</td>
<td>w8</td>
</tr>
<tr>
<td>MB+Routing A5</td>
<td>w9</td>
</tr>
</tbody>
</table>

Example for ConCept Configuration and Program

![Fig. 11: Example ConCept configuration (list of variables)](image)

A more detailed description of the configuration and programming in ConCept is being prepared.
Fig. 12: Example ConCept Program with MSTR block

Fig. 13: Example ConCept RDE table
5.2.1.4  Unity Pro XL Program for the PLC (Generic module sample)

1. Install your Ethernet module.

2. Configure the IP addresses of the Ethernet module.
   The Ethernet module can be found in the project browser under
   Station\Communication\networks\ 

3. Create a new ladder (LAD) or structure text (ST) program. A new program can be added un-
   der Station\Program\Tasks\MAST\Sections:

![Ethernet configuration in Unity Pro XL](image-url)
To send a message via MODBUS TCP/IP, the WRITE_VAR function is used. This function needs a few parameters like the IP address, variable type, module number, amount of registers, etc.

The IP addresses are represented by the ADDR_IBA variable. This variable is the XWAY address configured in the Ethernet module. For each module a different IP address is necessary. Thus for 3 modules, 3 addresses need to be configured.

In the example different variables were created to preserve a clear overview of the program. The module numbers are set to 200, 201 and 202 because we want to send Generic modules towards ibaPDA. Each module contains floats, integers and Booleans with a length of 120 registers in total.
5.2.2 Engineering example in PL7 Pro

5.2.2.1 Network configuration

In the hardware configuration of the ETY (network) card, an XWAY address needs to be created in order to be able to send the data to *ibaPDA* via the MODBUS protocol.

As shown in the table “Connection configuration” (see picture above) the XWAY consists of:

- **XWAY Address**: This address will be used in the program as “gateway”.
- **IP address**: Here the IP address of the *ibaPDA* data acquisition system needs to be filled in.
- **Protocol**: The MODBUS protocol should be selected to send messages via MODBUS.
- **Access**: This needs to be selected
- **Mode**: Multimode is required.

After finishing these settings and the IP address configuration of the card itself, you need to validate (confirm) the configuration.
5.2.2.2  Message configuration (example)

Create a program in the "Mast" task section.

![Program in PL7 Pro](image)

The message needs to be sent cyclically towards *ibaPDA*. Therefore, a system variable “%S5” is used. This system variable is a toggling digital signal with a 100 ms period. On the rising edge of the digital signal, the message will be sent towards *ibaPDA* (network 3).

The WRITE_VAR operation (1) is used to send the data towards *ibaPDA*. This operation uses the following settings:

```
WRITE_VAR(ADR#2.115SYS,'%MW',0,34,%MW300:34,%MW296:4)
```

- **XWAY Addr.** The previous created XWAY Addr. needs to be filled in here ADR#<XWAY AddrSYS
- **Type:** Here the variable type is selected '%MW'
- **Module index:** Here the same module index as used in *ibaPDA* should be applied.
- **Number of registers:** 34 is the number of registers (words of 16 bit) which will be sent towards *ibaPDA*.
- **Start address:** The address %MW300 to %MW334 will be sent
- **Management words:** These words contain the status, send settings, etc.

The operation (2) where the word %MW298 is set towards 50 is used to set the timeout of the WRITE_VAR function.

Value MW269:X0 (3) shows if the variable has been sent or not.
The timer (%TM70) is used to prevent that the message doesn’t take more than 1 minute (normally set to for example 100 ms) to be sent towards the *ibaPDA* system.

As a result a table will be created consisting of signals which are linked to the configured addresses and which will be copied into the signal table of *ibaPDA*. This table can be found below the timer.
6 Support and contact

Support

Phone: +49 911 97282-14
Fax: +49 911 97282-33
Email: support@iba-ag.com

Note
If you require support, indicate the serial number (iba-S/N) of the product or the license number.

Contact

Headquarters
iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany
Phone: +49 911 97282-0
Fax: +49 911 97282-33
Email: iba@iba-ag.com

Mailing address
iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address
iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany

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