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.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Revision - Chapter / Page</th>
<th>Author</th>
<th>Version SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.11</td>
<td>10-2020</td>
<td>3.1.4 New graphic</td>
<td>RM/IP</td>
<td>6.34</td>
</tr>
</tbody>
</table>

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. **Data Interface Modbus-TCP-Client** .....................................................................................8
   3.1 General information ........................................................................................................8
   3.1.1 Modbus TCP/IP .........................................................................................................8
   3.1.2 Modbus data model .....................................................................................................9
   3.1.3 Addressing the Modbus ............................................................................................9
   3.1.4 Client/Server architecture ......................................................................................10
   3.1.5 Modbus Protocol ....................................................................................................11
   3.1.6 MODBUS TCP/IP - Message layout........................................................................12
   3.1.6.1 Read data.............................................................................................................13
   3.1.6.2 Write data..........................................................................................................15
   3.1.7 References ..............................................................................................................16
   3.2 Configuration and engineering ibaPDA ................................................................... 17
   3.2.1 General settings ......................................................................................................17
   3.2.2 General interface settings .......................................................................................18
   3.2.3 Configuration ibaPDA input modules ..................................................................... 19
   3.2.3.1 General module settings......................................................................................20
   3.2.3.2 Signal configuration .............................................................................................23
   3.2.4 Configuring iba output modules .............................................................................25
   3.2.4.1 General module settings......................................................................................26
   3.2.4.2 Signal configuration .............................................................................................27
   3.2.5 Module diagnostics .................................................................................................28
   3.2.6 Starting the acquisition ..........................................................................................29

4. **Diagnostics** ......................................................................................................................30
   4.1 License check .............................................................................................................30
   4.2 Visibility of the interface ..........................................................................................30
   4.3 Log files ....................................................................................................................31
4.4  Connection diagnostics with PING ................................................................. 32
4.5  Checking the connection .................................................................................. 33

5  Appendix ................................................................................................................. 35
  5.1  Restrictions ......................................................................................................... 35
  5.1.1  TCP/IP protocol variants .................................................................................. 35
  5.2  Engineering example with ABB ACS880 ............................................................. 36
  5.2.1  Configuring the ACS880 drive ......................................................................... 36
  5.2.2  Configuration ibaPDA ...................................................................................... 37
  5.3  Troubleshooting .................................................................................................. 40
  5.3.1  Message layout ............................................................................................... 40
  5.3.2  Modbus error codes ........................................................................................ 41

6  Support and contact .................................................................................................. 42
1 About this Manual

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

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-Client 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 Logic diagram</td>
</tr>
<tr>
<td>Calling the menu command</td>
<td>Step 1 – Step 2 – Step 3 – Step x</td>
</tr>
<tr>
<td></td>
<td>Example: Select the menu Logic diagram - Add - New function block.</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**

ℹ️ 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-Client data interface:

- License for *ibaPDA-Interface-Modbus-TCP-Client*
- *ibaPDA* v6.33.2 or higher
- 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 35.

**Licenses**

<table>
<thead>
<tr>
<th>Order No.</th>
<th>Product name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.001022</td>
<td><em>ibaPDA-Interface-Modbus-TCP-Client</em></td>
<td>Extension license for an <em>ibaPDA</em> system providing an additional Modbus-TCP-Client interface. Number of connections: 64</td>
</tr>
<tr>
<td>31.101022</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-Client connections, max. 3 permitted</td>
</tr>
</tbody>
</table>
3 Data Interface Modbus-TCP-Client

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 RFC1180 and in RFC793 (see References, page 16).

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 Modbus data model

Modbus bases its data model on different basic types that have distinguishing characteristics. The four basic types are:

<table>
<thead>
<tr>
<th>Basic types</th>
<th>Object type</th>
<th>Service</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Inputs (Inputs)</td>
<td>Single bits</td>
<td>Read only</td>
<td>This type of data can be provided by the I/O of a device.</td>
</tr>
<tr>
<td>Coils (Outputs)</td>
<td>Single bits</td>
<td>Read / Write</td>
<td>Bit information that can be changed by the user program.</td>
</tr>
<tr>
<td>Input Registers (Inputs)</td>
<td>16-bit words</td>
<td>Read only</td>
<td>This type of data can be provided by the I/O of a device.</td>
</tr>
<tr>
<td>Holding Registers (Outputs)</td>
<td>16-bit words</td>
<td>Read / Write</td>
<td>Bit information that can be changed by the user program.</td>
</tr>
</tbody>
</table>

The distinctions between inputs and outputs, and between bit-addressable and word-addressable data items, do not need any application program. The four basic types can overlap, provided that the target system interprets them correctly.

For each of the primary types, the protocol allows for the individual selection of 65536 data items. Reading or writing these data units can comprise several consecutive registers up to a maximum data size which depends on the function code.

32-bit values (DINT, FLOAT) are stored in two successive registers.

3.1.3 Addressing the Modbus

The 4 data types are stored in different, possibly even overlapping storage areas. The accesses to the physical storage address are mapped by means of the application on the Modbus server. Basically, Modbus distinguishes between the internal numbering (Discrete Inputs, Coils, Registers), which usually begins with 1 and the addressing of the objects, usually beginning with 0.

Example: On many Modbus servers, the basic types are mapped in the following address spaces:

<table>
<thead>
<tr>
<th>Coils</th>
<th>0x00000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>0x10000</td>
</tr>
<tr>
<td>Input Registers</td>
<td>0x30000</td>
</tr>
<tr>
<td>Holding Registers</td>
<td>0x40000</td>
</tr>
</tbody>
</table>

This means, that the Holding register 1 is stored on the address 0x40000. The Holding register 1 is accessed by the logical reference number (address) 0. The input register 1 is stored on the address 0x30000. The input register 1 is also accessed by the address 0.

The logical Modbus reference numbers that are used for Modbus functions are integer indices without sign and beginning with 0.
3.1.4 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

Request → Indication
Indication → Response
Response → Confirmation

Modbus Client = Master (active)
Modbus Server = Slave (passive)

normally ibaPDA-TCP-Modbus-Client
normally the external device where you can read measurement data and write some settings into
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, sends the request 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-Client license, ibaPDA can establish up to 64 connections, i.e., you can establish connections to up to 64 Modbus servers. The number can be extended to a max. of 256 by loading the license more than once.

3.1.5 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 module settings, page 20

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.

There is also a "Modbus RTU over TCP/IP" version. The Modbus RTU message is transmitted via TCP/IP. This version is used when using Gateways serial/Ethernet.
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 Identifier</td>
<td>2</td>
<td>Identification of a Modbus request/response transaction</td>
</tr>
<tr>
<td>Protocol Identifier</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 Identifier</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.
- **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 ID, Function Code and Data fields.
- **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. It is typically used to communicate with a Modbus slave that is connected by a serial line to the Modbus server.

**Function code**:

One byte contains the function code that determines which function the server has to carry out depending on the request.

The *ibaPDA Modbus-TCP-Client* driver supports the functions:

- 0x01: Read Coils
- 0x02: Read Discrete Inputs
- 0x03: Read Holding Registers
- 0x04: Read Input Registers
- 0x0F: Write Multiple Coils
- 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.

### 3.1.6 MODBUS TCP/IP - Message layout

The request message has the same layout for all modes of access. The requesting controller (*ibaPDA as Modbus client*) determines the Modbus slave number (Unit-ID), the mode of access (function code), the starting address (address of the first data) and the number of the data.
### 3.1.6.1 Read data

**Request ibaPDA -> Modbus Server:**

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>ibaPDA Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBAP</td>
<td>00</td>
<td>2</td>
<td>UINT Transaction Identifier</td>
<td>xx xx</td>
<td>Incremented automatically in each cycle</td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>2</td>
<td>UINT Protocol Identifier</td>
<td>00 00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>2</td>
<td>UINT Cmd Length</td>
<td>00 06</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>06</td>
<td>1</td>
<td>BYTE Unit Identifier</td>
<td>xx</td>
<td>Modbus slave number</td>
</tr>
<tr>
<td>Fcode</td>
<td>07</td>
<td>1</td>
<td>BYTE Function code</td>
<td>xx</td>
<td>01: Read Coils</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>02: Read Discrete Inputs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>03: Read Holding Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>04: Read Input Register</td>
</tr>
<tr>
<td>Data</td>
<td>08</td>
<td>2</td>
<td>UINT Starting Address</td>
<td>xx xx</td>
<td>1. Address</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>UINT Number of Data</td>
<td>xx xx</td>
<td>No. of Coils, Input bits, Holding Registers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>or Input Registers</td>
</tr>
</tbody>
</table>

**Response Modbus -> Server ibaPDA:**

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>ibaPDA Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBAP</td>
<td>00</td>
<td>2</td>
<td>UINT Transaction Identifier</td>
<td>xx xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>2</td>
<td>UINT Protocol Identifier</td>
<td>00 00</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>04</td>
<td>2</td>
<td>UINT Cmd Length</td>
<td>00 07</td>
<td>= nBytes + 3</td>
</tr>
<tr>
<td></td>
<td>06</td>
<td>1</td>
<td>BYTE Unit Identifier</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>Fcode</td>
<td>07</td>
<td>1</td>
<td>BYTE Function code</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>Data</td>
<td>08</td>
<td>1</td>
<td>BYTE Number of Bytes</td>
<td>xx</td>
<td>nBytes</td>
</tr>
<tr>
<td></td>
<td>09</td>
<td>n</td>
<td>BYTE Data</td>
<td>xx xx</td>
<td>Input values</td>
</tr>
</tbody>
</table>

**Examples:**

**Function 01: Read Coils**

<table>
<thead>
<tr>
<th>Request</th>
<th>(hex)</th>
<th>(hex)</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Identifier</td>
<td>00 01</td>
<td>00 01</td>
<td>Transaction Identifier</td>
</tr>
<tr>
<td>Protocol Identifier</td>
<td>00 00</td>
<td>00 00</td>
<td>Protocol Identifier</td>
</tr>
<tr>
<td>Cmd LEN</td>
<td>00 06</td>
<td>00 06</td>
<td>Cmd LEN</td>
</tr>
<tr>
<td>Unit Identifier</td>
<td>00</td>
<td>00</td>
<td>Unit Identifier</td>
</tr>
<tr>
<td>FCode</td>
<td>01</td>
<td>01</td>
<td>FCode</td>
</tr>
<tr>
<td>Start Adr (Hi)</td>
<td>00</td>
<td>03</td>
<td>No. byte</td>
</tr>
<tr>
<td>Start Adr (Lo)</td>
<td>13</td>
<td>xx</td>
<td>Coils 27-20(^1)</td>
</tr>
</tbody>
</table>
### Function 01: Read Coils
<table>
<thead>
<tr>
<th>No. Values (Hi)</th>
<th>00</th>
<th>xx</th>
<th>Coils 35-28</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. Values (Lo)</td>
<td>14</td>
<td>xx</td>
<td>Coils 39-36</td>
</tr>
</tbody>
</table>

### Function 02: Read Discrete Inputs
<table>
<thead>
<tr>
<th>Request</th>
<th>(hex)</th>
<th>(hex)</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Identifier</td>
<td>00 02</td>
<td>00 02</td>
<td>Transaction Identifier</td>
</tr>
<tr>
<td>Protocol Identifier</td>
<td>00 00</td>
<td>00 00</td>
<td>Protocol Identifier</td>
</tr>
<tr>
<td>Cmd LEN</td>
<td>00 06</td>
<td>00 05</td>
<td>Cmd LEN</td>
</tr>
<tr>
<td>Unit Identifier</td>
<td>00</td>
<td>00</td>
<td>Unit Identifier</td>
</tr>
<tr>
<td>FCode</td>
<td>02</td>
<td>02</td>
<td>FCode</td>
</tr>
<tr>
<td>Start Adr (Hi)</td>
<td>00</td>
<td>02</td>
<td>No. Byte</td>
</tr>
<tr>
<td>Start Adr (Lo)</td>
<td>C4</td>
<td>xx</td>
<td>Inputs 204-197</td>
</tr>
<tr>
<td>No. Values (Hi)</td>
<td>00</td>
<td>xx</td>
<td>Inputs 206-205</td>
</tr>
<tr>
<td>No. Values (Lo)</td>
<td>0A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Function 03: Read Holding Registers
<table>
<thead>
<tr>
<th>Request</th>
<th>(hex)</th>
<th>(hex)</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Identifier</td>
<td>00 03</td>
<td>00 03</td>
<td>Transaction Identifier</td>
</tr>
<tr>
<td>Protocol Identifier</td>
<td>00 00</td>
<td>00 00</td>
<td>Protocol Identifier</td>
</tr>
<tr>
<td>Cmd LEN</td>
<td>00 06</td>
<td>00 0B</td>
<td>Cmd LEN</td>
</tr>
<tr>
<td>Unit Identifier</td>
<td>00</td>
<td>00</td>
<td>Unit Identifier</td>
</tr>
<tr>
<td>FCode</td>
<td>03</td>
<td>03</td>
<td>FCode</td>
</tr>
<tr>
<td>Start Adr (Hi)</td>
<td>00</td>
<td>08</td>
<td>No. Byte</td>
</tr>
<tr>
<td>Start Adr (Lo)</td>
<td>6B</td>
<td>xx xx</td>
<td>Register 108 (Hi, Lo)</td>
</tr>
<tr>
<td>No. Values (Hi)</td>
<td>00</td>
<td>xx xx</td>
<td>Register 109 (Hi, Lo)</td>
</tr>
<tr>
<td>No. Values (Lo)</td>
<td>04</td>
<td>xx xx</td>
<td>Register 110 (Hi, Lo)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>xx xx</td>
<td>Register 111 (Hi, Lo)</td>
</tr>
</tbody>
</table>

1) Bits are always transferred and stored in the following sequence: MSB left, LSB right. In the last byte, the remaining bits (MSB, left) are filled with 0.
3.1.6.2 Write data

Request ibaPDA -> Modbus Server:

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>ibaPDA Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>2</td>
<td>UINT</td>
<td>Transaction Identifier</td>
<td>xx xx</td>
<td>Incremented automatically in each circle</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>UINT</td>
<td>Protocol Identifier</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 xx</td>
<td>nBytes + 7</td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>BYTE</td>
<td>Unit Identifier</td>
<td>xx</td>
<td>Modbus slave number</td>
</tr>
<tr>
<td>Fcode</td>
<td>07</td>
<td>BYTE</td>
<td>Function code</td>
<td>xx</td>
<td>0F: Write Multiple Coils</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10: Write Multiple Registers</td>
</tr>
<tr>
<td>Data</td>
<td>08</td>
<td>UINT</td>
<td>Starting Address</td>
<td>xx xx</td>
<td>1. Address</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>UINT</td>
<td>Number of Values</td>
<td>xx xx</td>
<td>Number of the Coils or Holding Registers</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>BYTE</td>
<td>Number of Bytes</td>
<td>xx</td>
<td>Output Values</td>
</tr>
<tr>
<td>13</td>
<td>n</td>
<td>BYTE</td>
<td>Data</td>
<td>xx xx</td>
<td></td>
</tr>
</tbody>
</table>

Response Modbus -> Server ibaPDA:

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>ibaPDA Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MBAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>2</td>
<td>UINT</td>
<td>Transaction Identifier</td>
<td>xx xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>UINT</td>
<td>Protocol Identifier</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></td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>BYTE</td>
<td>Unit Identifier</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>Fcode</td>
<td>07</td>
<td>BYTE</td>
<td>Function code</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>Data</td>
<td>08</td>
<td>UINT</td>
<td>Starting Address</td>
<td>xx xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>09</td>
<td>4</td>
<td>UINT</td>
<td>Number of values</td>
<td>xx xx</td>
<td>Mirror of request</td>
</tr>
</tbody>
</table>

Examples:

Function 0F: Write Multiple Coils

<table>
<thead>
<tr>
<th>Request</th>
<th>(hex)</th>
<th>(hex)</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Identifier</td>
<td>00 05</td>
<td>00 05</td>
<td>Transaction Identifier</td>
</tr>
<tr>
<td>Protocol Identifier</td>
<td>00 00</td>
<td>00 00</td>
<td>Protocol Identifier</td>
</tr>
<tr>
<td>Cmd LEN</td>
<td>00 09</td>
<td>00 06</td>
<td>Cmd LEN</td>
</tr>
<tr>
<td>Unit Identifier</td>
<td>01</td>
<td>01</td>
<td>Unit Identifier</td>
</tr>
<tr>
<td>FCode</td>
<td>0F</td>
<td>0F</td>
<td>FCode</td>
</tr>
</tbody>
</table>
### Function 0F: Write Multiple Coils

<table>
<thead>
<tr>
<th>Start Adr (Hi)</th>
<th>00</th>
<th>00</th>
<th>Start Adr (Hi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Adr (Lo)</td>
<td>03</td>
<td>00</td>
<td>Start Adr (Lo)</td>
</tr>
<tr>
<td>No. Values (Hi)</td>
<td>00</td>
<td>00</td>
<td>No. Values (Hi)</td>
</tr>
<tr>
<td>No. Values (Lo)</td>
<td>0A</td>
<td>03</td>
<td>No. Values (Lo)</td>
</tr>
<tr>
<td>No. Bytes</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coils 11-8</td>
<td>xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coils 13-12</td>
<td>xx</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Function 10: Write Multiple Registers

<table>
<thead>
<tr>
<th>Request</th>
<th>(hex)</th>
<th>(hex)</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Identifier</td>
<td>00 06</td>
<td>00 06</td>
<td>Transaction Identifier</td>
</tr>
<tr>
<td>Protocol Identifier</td>
<td>00 00</td>
<td>00 00</td>
<td>Protocol Identifier</td>
</tr>
<tr>
<td>Cmd LEN</td>
<td>00 13</td>
<td>00 06</td>
<td>Cmd LEN</td>
</tr>
<tr>
<td>Unit Identifier</td>
<td>02</td>
<td>02</td>
<td>Unit Identifier</td>
</tr>
<tr>
<td>FCode</td>
<td>10</td>
<td>10</td>
<td>FCode</td>
</tr>
<tr>
<td>Start Adr (Hi)</td>
<td>00</td>
<td>00</td>
<td>Start Adr (Hi)</td>
</tr>
<tr>
<td>Start Adr (Lo)</td>
<td>20</td>
<td>20</td>
<td>Start Adr (Lo)</td>
</tr>
<tr>
<td>No. Values (Hi)</td>
<td>00</td>
<td>00</td>
<td>No. Values (Hi)</td>
</tr>
<tr>
<td>No. Values (Lo)</td>
<td>06</td>
<td>06</td>
<td>No. Values (Lo)</td>
</tr>
<tr>
<td>No. Bytes</td>
<td>0C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register 33</td>
<td>xx xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register 34</td>
<td>xx xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register 35</td>
<td>xx xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register 36</td>
<td>xx xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register 37</td>
<td>xx xx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register 38</td>
<td>xx xx</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.1.7 References

**Further 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 Client" is displayed in the signal tree. There is no need to add the interface manually.

3.2.1 General settings

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

- Disconnect connection after x seconds of inactivity: Behavior and timeout duration can be specified.
- Set signal values to zero when a TCP/IP connection is lost: If this option is disabled, the value read last will be kept.
- Write connection events in Windows event log: Current events are logged in Windows.
- Interfaces for which packets must be acknowledged immediately: Selection of required interfaces
3.2.2 General interface settings

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

- **Set all values to zero when the connection to a Modbus server is lost**
  If this option is disabled, the last valid measured values will be kept.

- **Start acquisition even if a Modbus server is not accessible**
  Generally, the data acquisition cannot be started when the Modbus server is not accessible. If this option is enabled, the acquisition will be started with a warning message. In case the connection to the Modbus server is established only thereafter, the configured values will be measured without having to restart *ibaPDA*. This option is recommended for an automatic restart.

- **<Open log file>**
  Connection-specific actions are logged in a text file. The file can be opened and viewed via this button.

- **<Reset counters>**
  The calculated times and the error counter can be reset to zero.

**Overview of connections:**
As soon as the connection has been established, you can see the live data in the overview. Also see *Checking the connection*, page 33.
3.2.3 Configuration ibaPDA input modules

Adding an input module
Select the *Hardware* option in the I/O manager and add a module under the "Modbus TCP Client" interface by clicking on the module type and selecting it.
3.2.3.1 General module settings

If you want to configure a data module, mark the module in the tree structure, select the General tab and do the following settings in the dialog.

**Basic settings**

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

**Locked**
A module can be locked to avoid unintentional or unauthorized changing of the module settings.
ibapDA-Interface-Modbus-TCP-Client

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.

Advanced
Swap analog signals
Option to change the order of the byte evaluation.

Tip
For a Modicon M580 system select the setting "Swap 8 bit"!

Module Layout
No. of analog/digital input signals
You can set up the number of analog and digital signals to be measured. The maximum number of signals is 1000 (analog) and 2000 (digital).
For a detailed description of the parameters, see the ibaPDA manual.

Modbus
IP address
IP address of the Modbus server.

Port number
By default, the port number is set to the standard port 502.

Protocol
"Modbus TCP" is the default setting. Optionally, you can select "Modbus serial". Thus, the serial RTU-Modbus protocol is selected, that is transmitted via TCP/IP. Use this protocol, in case the device is connected to an Ethernet/serial gateway, e. g. the IF2E001 from IME. See Modbus Protocol, page 11

Addresses start at 1
For Modbus, the internal areas (Registers, Coils) are numbered starting at 1; the reference addresses are numbered beginning with 0.
True: The addresses on the *Analog* and *Digital* tabs start at 1.
False: The addresses on the *Analog* and *Digital* tabs start at 0.

**Tip**
For a Modicon M580 system select the setting "False"!

**Send messages in parallel**
True: All messages that are needed for reading the signals of this module are sent in parallel; then it waits for responses.
False: After each request message, the system waits for the response message, only then the next request message will be sent.

**Modbus Input**

**Analog type**
Choose the analog data type “Holding registers” or “Input registers”.

**Digital type**
Choose the digital data type "Coils", "Discrete inputs" or "Holding registers". With "Holding registers“, you can acquire single bits from holding registers.

**Maximum gap between registers**
Registers that are not consecutive, are sent in a request, if the gaps between the registers are not larger than the specified value. The default setting is 4.

**Maximum gap between coils**
Coils or Input bits that are not consecutive are sent in a request, if the gaps between the bit addresses are not larger than the specified value. The default setting is 64.

**Max. registers/coils per message**
Here the maximum number of registers or coils per message can be determined. By default the values, usually entered according to the standard, are specified. Some devices allow other registers or coils per message, which can be set here.

**Update time**
The update time determines how fast the data are being requested by the Modbus server. The standard equals the parameter "Timebase".

**Modbus Output**
see section "Modbus Input".

**Send only when changed**
False: The output message is sent in the update cycle specified above
True: The output message is only sent after a value in the message has been changed, but no later than 1 min.
3.2.3.2 Signal configuration

On the Analog and Digital tabs you can assign names and comments to the signals. Moreover, the signals can be enabled and disabled.

**Analog signals**

![Modbus-Client](image)

Description of the columns:

- **Name, Unit, Gain, Offset, Active:** see ibaPDA manual.
- **Slave:** The slave number is usually only relevant when you are working with a gateway that has multiple Modbus slaves connected. The slave is entered in the Modbus message in the Unit ID field.
- **Register:** The register number reaches from 0 to 65535 or from 1 to 65536, depending on the option "Addresses start at 1" on the General tab. Each register is 16 bit wide. You can specify the register number in hexadecimal format or in decimal format. You switch between them via the context menu.
- **Data Type:** Select one of the following data types: BYTE, WORD, DWORD, INT, DINT, FLOAT, DOUBLE.

The values of data type FLOAT, DINT and DWORD use 2 registers each.

The data type DOUBLE uses 4 registers, 2 values of the BYTE data type use one register.

**Tip**

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Slave</th>
<th>Register</th>
<th>Bit no.</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ready to switch ON</td>
<td>1</td>
<td>611</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Ready run</td>
<td>1</td>
<td>611</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Ready req</td>
<td>1</td>
<td>611</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Tripped</td>
<td>1</td>
<td>611</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Off 2 inactive</td>
<td>1</td>
<td>611</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Off 3 inactive</td>
<td>1</td>
<td>611</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Switch-on inhibited</td>
<td>1</td>
<td>611</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

**Description of the columns:**

- **Slave**: The slave number is usually only relevant when you are working with a gateway that has multiple Modbus slaves connected. In Modbus TCP mode, the slave number is ignored by most Modbus servers.

- **Register**: This column is only visible for the digital type of access "Holding registers". The bits (0-15) are acquired from one holding register, e.g. one status word.

- **Bit no.**: The bit number corresponds to the coil number, the discrete input number or the bit number within the holding register. The value range is from 0 to 65535 or 1 to 65535, depending on the parameter "Addresses start at 1".

---

**Further documentation**

For a detailed description of additional options, see the *ibaPDA* manual.
3.2.4 Configuring iba output modules

Beginning with *ibaPDA*-version 6.33.2, data can also be sent to the Modbus TCP server. The "Modbus TCP Client" interface is shown in the tree structure of the *ibaPDA* I/O manager, menu option *Outputs*, in case all system requirements are met. There is no need to add the interface manually.

**Adding an output module**

Select the *Outputs* option in the I/O manager and add a module under the "Modbus TCP Client" interface by clicking on the module type and selecting it.
3.2.4.1 General module settings

If you want to configure a data module, mark the module in the tree structure, select the General tab and do the following settings in the dialog.

The parameters are almost identical to those of the input modules, see General module settings, page 20.

Consider the differences in the settings in contrast to the input modules:

- The "Timebase" parameter does not exist.
- As analog type, only "Holding registers" is allowed.
- As digital type, only "Holding registers" and "Coils" are allowed.
- The message send cycle is determined by the following parameters:
  - Update time (in ms): If you enter 0 or a value smaller than 50, the send cycle corresponds to the ibaPDA task cycle of appr. 50 ms.
  - Send only when changed:
    - False: The output message is sent in the update cycle specified above.
    - True: The output message is only sent after a value in the message has been changed, but no later than 1·min.

### 3.2.4.2 Signal configuration

On the Analog and Digital tabs, you can define the output signals in the "Expression" column in a similar way virtual signals are defined. You can enter simple expressions or references to existing signals directly into the table. You can also click on the button \[\text{button}\] in order to call up the expression editor. See also ibaPDA manual.

**Tip**

If you define the output data in a "Virtual module" and only enter the references to these data, you can also include these data in the data recording.

---

### Analog signals

<table>
<thead>
<tr>
<th>Name</th>
<th>Expression</th>
<th>Slave</th>
<th>Register</th>
<th>Data type</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Calculated Time</td>
<td>[f] [11c]</td>
<td>7</td>
<td>201</td>
<td>DINT</td>
<td></td>
</tr>
<tr>
<td>1 Running Time</td>
<td>T0</td>
<td>7</td>
<td>203</td>
<td>FLOAT</td>
<td></td>
</tr>
<tr>
<td>2 Cons1 Min</td>
<td>1</td>
<td>7</td>
<td>205</td>
<td>INT</td>
<td></td>
</tr>
<tr>
<td>3 Cons1 Max</td>
<td>1000</td>
<td>7</td>
<td>206</td>
<td>INT</td>
<td></td>
</tr>
</tbody>
</table>

- **Name**: Enter a meaningful name for each signal in the "Name" column.
- **Expression**: Here, you can enter an expression or a reference to an existing signal. You can also call up the expression editor by clicking on the button \[\text{button}\]. With a click on the button \[\text{button}\], you can analyze an incorrect expression.
- **Slave, Register, Data type**: see description in \[\text{button}\] Signal configuration, page 23.
- **Active**: Deactivated signals will not be transmitted.
Digital signals

- **Name**: Enter a meaningful name for each signal in the "Name" column.
- **Expression**: Here, you can enter an expression or a reference to an existing signal. You can also call up the expression editor by clicking on the button [expression]. With a click on the button [expression], you can analyze an incorrect expression.
- **Slave, Register, Bit no**: see description in "Signal configuration", page 23.
- **Active**: Deactivated signals will not be transmitted.

Further documentation

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

### 3.2.5 Module diagnostics

The module diagnostics exists for the input as well as for the output direction. There are the two sub-registers *Analog* and *Digital*, that show the current values as numerical values.

The "Analog" (inputs) table always displays the unscaled raw values in the floating point format.
The following errors can be detected on the input side:

- No data are displayed:
  - No connection to the Modbus server.
  - Error message by the Modbus server, see Troubleshooting, page 40.

- Incorrect values are displayed:
  - The byte order is set incorrectly, see General module settings, page 20.

### 3.2.6 Starting the acquisition

When starting the acquisition, *ibaPDA* tries to establish a connection to the Modbus server and read the data. In case the server detects an error, a message will appear on the screen.

If you have selected the option "Start acquisition even if a Modbus server is not accessible"; (see General interface settings, page 18), you will see the warning message "Error connecting to Modbus server...". Nevertheless, the acquisition will be started.
4 Diagnostics

4.1 License check

If the interface "Modbus TCP Client" 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 Client" 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)

With no existing connection you receive error messages.

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

If you mark the data interface Modbus Client 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

- The <Reset counters> button can be used to reset the error counter and the average, min and max cycle times.
- With the <Open log file> button, you can open the log files with an ASCII Editor to see the history of errors.

The list of connections shows the following values:

- IP address: Address of the Modbus server
- Error count: incremented with the detection of sporadic transmission errors (connection failure), or in case the Modbus server sends an error code, see Troubleshooting, page 40
- Update time: The cycle in which ibaPDA requests the data from the Modbus server (maximum calculated from the configured update time and the Modbus server response time)
- Response time Actual: The time the Modbus server sends with (time between request and response)
- Response time Average, Min, Max: calculates values since the start of the acquisition or the reset of the counter with the <Reset counters> button

Colors:

- Green: The connection is OK.
- Red: The connection has failed.

A failed connection may have the following causes:

- Modbus Server 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 "Error count" column are high, this points to one of the following errors:
  - Error in message header
  - The "Delayed acknowledge" problem occurs, see TCP/IP protocol variants, page 35
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}\n
5.2 Engineering example with ABB ACS880

We use the ABB ACS880 drive with the FENA-21 Ethernet Adapter Module.

5.2.1 Configuring the ACS880 drive

On the ACS880, the following data are configured using the ABB Drive Composer pro v1.8.0.9:

- Parameter group 50 "Fieldbus adapter (FBA)"
  Using the Ethernet Adapter Module FENA-21 on Slot 1

- Parameter group 54 "FBA B settings"
  Using the Modbus-TCP (Classic) protocol,
  Setting the IP address
### 5.2.2 Configuration ibaPDA

2 modules are created in the I/O manager.
The modules have the following settings:

![Modbus iO Manager](image)

The drive parameters are addressed on the Analog and Digital tabs.

The parameters of the drives can be addressed in 2 different ways:

- **Parameter as 16 bit Integer (data type INT):** The registers are addressed according to the following formula:
  
  \[
  \text{Register} = \text{parameter group} \times 100 + \text{parameter number}
  \]

  **Examples:**
  
  Register 101 equals group 1, parameter 1
  Register 161 equals group 1, parameter 61
  Register 1211 equals group 12, parameter
The received values are raw values. You can calculate the scaling factors from the ACS880 parameter group 46 "Monitoring/Scaling settings".

- Parameter as 32 bit Integer (data type DINT): The registers are addressed according to the following formula:

  \[ \text{Register} = 20000 + \text{parameter group} \times 200 + \text{parameter number} \times 2 \]

Examples:
- Register 20202 equals group 1, parameter 1
- Register 20322 equals group 1, parameter 61
- Register 22422 equals group 12, parameter 11

The received analog values have already been scaled. For getting the physical values, for all values the factor 0.01 is entered in the "Gain" column.

The digital values (status and control bits) can be acquired directly as holding register bit number. They can also be broken down as virtual modules (16-bit decoder or 32-bit decoder) from the measured analog values.

The **Diagnostics** tab shows the raw values of the measured data:
5.3 Troubleshooting

The server does not send an acknowledgment to the client, in case there is a transmission failure or in case a non-existent (or switched off) device is being addressed. This results in a Timeout. When the communication runs over a Modbus RTU/TCP Gateway, you get an error message from this Gateway telling you that the addressed device does not answer.

The server returns the errors with a corresponding error message to the client.

Depending on the error code, ibaPDA reports the error at the start either as error or as warning:

<table>
<thead>
<tr>
<th>Offs</th>
<th>Bytes</th>
<th>Type</th>
<th>Modbus Description</th>
<th>Contents (hex)</th>
<th>ibaPDA Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>2</td>
<td>UINT</td>
<td>Transaction Identifier</td>
<td>xx xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>UINT</td>
<td>Protocol Identifier</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 03</td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>1</td>
<td>BYTE</td>
<td>Unit ID</td>
<td>xx</td>
<td>Mirror of request</td>
</tr>
<tr>
<td>07</td>
<td>1</td>
<td>BYTE</td>
<td>Function code &quot;Exception&quot;</td>
<td>0x80 + xx</td>
<td>Mirror of function code with MSB=1</td>
</tr>
<tr>
<td>08</td>
<td>1</td>
<td>BYTE</td>
<td>Error code</td>
<td>xx</td>
<td>Error code</td>
</tr>
</tbody>
</table>

The received function code is copied and the highest-order bit set (MSB).
**Example:**

<table>
<thead>
<tr>
<th>Request</th>
<th>(hex)</th>
<th>(hex)</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trans ID</td>
<td>00 03</td>
<td>00 03</td>
<td>Trans ID</td>
</tr>
<tr>
<td>Prot ID</td>
<td>00 00</td>
<td>00 00</td>
<td>Prot ID</td>
</tr>
<tr>
<td>Cmd LEN</td>
<td>00 06</td>
<td>00 03</td>
<td>Cmd LEN</td>
</tr>
<tr>
<td>Unit ID</td>
<td>01</td>
<td>01</td>
<td>Unit ID</td>
</tr>
<tr>
<td>FCode</td>
<td>03</td>
<td>83</td>
<td>FCode with error indication</td>
</tr>
<tr>
<td>Start Adr (Hi)</td>
<td>00</td>
<td>02</td>
<td>Error code</td>
</tr>
<tr>
<td>Start Adr (Lo)</td>
<td>6D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. Values (Hi)</td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No. Values (Lo)</td>
<td>04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**5.3.2 Modbus error codes**

<table>
<thead>
<tr>
<th>Error codes (hex)</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Illegal Function</td>
<td>Use of a function code that is not supported</td>
</tr>
<tr>
<td>02</td>
<td>Illegal Data Address</td>
<td>Use of an invalid storage address or attempt to write to a read-only address</td>
</tr>
<tr>
<td>03</td>
<td>Illegal Data Value</td>
<td>Use of illegal data values, e.g. a non-permitted number of registers</td>
</tr>
<tr>
<td>04</td>
<td>Slave Device Failure</td>
<td>Unrecoverable failure</td>
</tr>
<tr>
<td>05</td>
<td>Acknowledge</td>
<td>The server needs a long time for processing the request. It sends an acknowledge in order to prevent a timeout of the client.</td>
</tr>
<tr>
<td>06</td>
<td>Slave Device Busy</td>
<td>Currently, the device cannot process Modbus commands</td>
</tr>
<tr>
<td>0A</td>
<td>Gateway Path Unavailable</td>
<td>Gateway is overloaded or configured incorrectly</td>
</tr>
<tr>
<td>0B</td>
<td>Gateway Target Device</td>
<td>Error message of the gateway: No answer from the addressed device</td>
</tr>
</tbody>
</table>
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