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 TCP/UDP to SIMATIC S7 ........................................................................8
  3.1 General information .................................................................................................8
  3.2 SIMATIC S7 configuration & engineering .................................................................9
    3.2.1 Data blocks .............................................................................................................10
    3.2.1.1 Header .................................................................................................................11
    3.2.1.2 Data areas .............................................................................................................12
    3.2.2 S7-300 / S7-400-CPUs without local Ethernet interface ........................................13
    3.2.3 S7-300 / S7-400-CPUs with local Ethernet interface ...............................................15
    3.2.4 S7-1200 CPU with local Ethernet interface .............................................................17
    3.2.5 S7-1500-CPUs with local Ethernet interface ............................................................19
  3.3 ibaPDA configuration & engineering ......................................................................22
    3.3.1 General settings ......................................................................................................22
    3.3.2 General interface settings .......................................................................................22
    3.3.3 General module settings ........................................................................................25
    3.3.4 General signal configuration ...................................................................................26
    3.3.5 Module type S7 TCP/UDP Integer .........................................................................27
    3.3.6 Module type S7 TCP/UDP Real .............................................................................27
    3.3.7 Module type S7 TCP/UDP General .........................................................................27
    3.3.8 S7 UDP Request/S7 UDP Request Decoder module types ......................................28
    3.3.9 Module diagnostics ................................................................................................28

4 Diagnosis ........................................................................................................................29
  4.1 Checking the license ...................................................................................................29
  4.2 Visibility of the interface ............................................................................................29
  4.3 Log files .....................................................................................................................30
  4.4 Connection diagnostics with PING ...........................................................................31
  4.5 Checking the connection ............................................................................................32
4.6  Diagnostic modules ............................................................................................................. 34

5  Appendix .............................................................................................................................. 37
   5.1  Restrictions ....................................................................................................................... 37
   5.1.1  TCP/IP protocol variants ............................................................................................. 37
   5.2  S7 engineering examples ................................................................................................. 38
   5.2.1  S7CLASSIC_TCP_UDP ............................................................................................... 39
   5.2.1.1  CPU 317-2 with PN interface ................................................................................... 39
   5.2.1.2  CPU 317 with CP 343-1 LEAN ............................................................................... 46
   5.2.1.3  CPU 412-2 with PN interface ................................................................................... 54
   5.2.1.4  CPU 412 with CP 443-1 .......................................................................................... 54
   5.2.2  S7TIA_TCP_UDP_V13_SP1 .......................................................................................... 55
   5.2.2.1  CPU 1212C (Firmware V2.0) ................................................................................... 55
   5.2.2.2  CPU 1516 with PN interface .................................................................................... 64
   5.2.2.3  CPU 1516 with PN interface and configuration dialog ............................................. 69
   5.2.2.4  CPU 1516 with CP1543-1 ....................................................................................... 74
   5.3  ibaPDA engineering example ........................................................................................... 76
       5.3.1  Configuration of data telegrams ............................................................................... 76
       5.3.2  Configuration of Technostring ................................................................................. 77
       5.3.3  Configuration Watchdog .......................................................................................... 78
       5.3.4  Online view ............................................................................................................... 79
   6  Support and contact ............................................................................................................ 80
1 About this Manual

This document describes the function and application of the software interface 
ibiaPDA-Interface-S7-TCP/UDP

This documentation is a supplement to the ibiaPDA manual. Information about all the other 
characteristics and functions of ibiaPDA can be found in the ibiaPDA 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 the standard regulations.

This documentation in particular addresses persons, who are concerned with the configuration, 
test, commissioning or maintenance of Programmable Logic Controllers of the supported prod-
ucts. For the handling ibiaPDA-Interface-S7-TCP/UDP the following basic knowledge is required 
and/or useful:

■ Windows operating system
■ Basic knowledge of ibiaPDA
■ Knowledge of configuration and operation of the relevant control 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 to use the S7 TCP/UDP data interface:

- \textit{ibaPDA} v6.33.2 or higher
- License for \textit{ibaPDA}-Interface S7-TCP/UDP
- Network connection 10/100 Mbits
- Step7 from version V4.0 or TIA Portal from V11
- S7 CPU with integrated PN port or communication processor

For more requirements on the PC hardware used and the supported operating systems, see the \textit{ibaPDA} documentation.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|p{10cm}|}
\hline
Order No. & Product name & Description \\
\hline
31.001040 & ibaPDA-Interface-S7-TCP/UDP & Extension license for an ibaPDA system by one TCP/IP and UDP/IP interface \\
& & Number of connections: 64 \\
\hline
31.101040 & one-step-up-Interface-S7-TCP/UDP & Extension license for the extension of an existing ibaPDA-Interface-S7-TCP/UDP interface by another 64 S7-TCP/UDP connections, max. 3 permitted \\
\hline
\end{tabular}
\end{table}

\textbf{Note} \\
We recommend running the TCP/IP or UDP communication on a separate network segment to exclude any mutual influence by other network components.

System restrictions

- For different ways of handling the TCP/IP acknowledge see \textit{TCP/IP protocol variants}, page 37 (all \textit{ibaPDA} versions).

Licenses
3 Data interface TCP/UDP to SIMATIC S7

3.1 General information

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

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

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

<table>
<thead>
<tr>
<th>Module Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>32 analog values (integer) and 32 binary signals</td>
</tr>
<tr>
<td>Real</td>
<td>8, 16 or 32 analog values (real) and 32 binary signals</td>
</tr>
<tr>
<td>Generic</td>
<td>Any data structure with a maximum length of 4096 bytes¹</td>
</tr>
</tbody>
</table>

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

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

¹ Until ibaPDA V6.30 limited by properties of the S7 communication processors
TCP and UDP

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

The User Datagram Protocol, short UDP, is a connectionless transport protocol. Its function is similar to that of the connection-oriented TCP. However, it works connectionless and is thus not secure, which means that the sender does not know whether the data packets it has sent have actually arrived. TCP sends confirmations upon receiving data, UDP does not. This method has the advantage that the packet header is much smaller and no acknowledgments have to be sent over the link. In principle, this enables a slightly higher data rate.

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

---

Note

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

3.2 SIMATIC S7 configuration & engineering

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

- CPUs without a local Ethernet interface
  The S7 side is configured using the STEP7 tools "HW Config" and "NetPro" included in SIMATIC Manager. In the STEP7 program, you insert send blocks (AG_SEND, AG_LSEND) that use the configured connections.

- CPUs with a local Ethernet interface
  The connections do not need to be configured separately. Both connection establishment and sending are performed using standard blocks (TCON, TSEND, TUSEND) in STEP7.

- The CPUs of the S7-1200 and S7-1500 series
  These CPUs generally feature local Ethernet interfaces and can only be configured using the TIA Portal. Here, there are blocks that perform both the task of connection establishment and sending data (TSEND_C).

Engineering examples for different configurations, see S7 engineering examples, page 38.
Note

Please observe the following in all connection types described:

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

3.2.1 Data blocks

All methods described above require the data to be sent to be provided in data blocks. According to the ibaPDA module structure, the data for each module are transmitted with a telegram. Each telegram is based on a data block and a connection. The data blocks have a uniform header and a data structure that corresponds to the module type.

Note

Special for the S7-1200 / S7-1500 controller

The "Optimized block access" block attribute must not be set for the telegram data blocks.
3.2.1.1 Header

The header consists of 3 integer values.

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

  - for the integer module type: 74
  - for the real module type: 42, 74 or 138 (for 8, 16 or 32 reals)
  - for generic module type: 8...4096

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

<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</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 corresponds to the index in the *ibaPDA* module setting. This value must be unique and must not be changed during data transmission.

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

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

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

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th>Initial value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>message_length</td>
<td>INT</td>
<td>74</td>
<td>total length of the DB in bytes</td>
</tr>
<tr>
<td>+0.0</td>
<td>module_index</td>
<td>INT</td>
<td>0</td>
<td>ibaPDA - module index (00...63)</td>
</tr>
<tr>
<td>+4.0</td>
<td>sequence_counter</td>
<td>INT</td>
<td>0</td>
<td>telegram counter</td>
</tr>
<tr>
<td>+6.0</td>
<td>analog_signals</td>
<td>ARRAY[0..31]</td>
<td>32 analog values INT</td>
<td></td>
</tr>
<tr>
<td>+7.0</td>
<td>digital_signals</td>
<td>ARRAY[0..31]</td>
<td>32 binary values</td>
<td></td>
</tr>
</tbody>
</table>

Note
Observe the different byte order between S7 and ibaPDA.
Example:
If you set bit DB222.DBX70.0, it will arrive as bit 24 in ibaPDA. But if you write 16#00000001 to DB222.DBD70, bit 0 is set in ibaPDA.

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

Module type Generic
Any order of data with different data types can follow after the header starting at offset 6. ibaPDA supports the following data formats:
BYTE, WORD, DWORD, INT, DINT and FLOAT.
The data structure defined here has to be copied in *ibaPDA*. The BYTE, WORD and DWORD variables may also be interpreted as 8, 16 or 32 single bits (and vice versa).

### 3.2.2 S7-300 / S7-400-Cpus without local Ethernet interface

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

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

A separate connection has to be configured for each module configured in *ibaPDA*.

#### Configuration of the Ethernet CP in HW Config.
- Select the PN-IO interface of the Ethernet CP (CP343-1 or CP443-1).
- Click <Properties> in the General tab.
- Assign the IP address and the subnet mask of the S7 controller

#### Creating the connection in NetPro

Select the following parameters:

<table>
<thead>
<tr>
<th>Connection partners (station):</th>
<th>unspecified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection - Type:</td>
<td>TCP connection or UDP connection</td>
</tr>
<tr>
<td>General - Name:</td>
<td>assign a unique name</td>
</tr>
<tr>
<td>General - Active connection establishment</td>
<td>enabled</td>
</tr>
<tr>
<td>for TCP connection:</td>
<td>enabled</td>
</tr>
<tr>
<td>for UDP connection:</td>
<td>the parameter does not exist (always passive)</td>
</tr>
<tr>
<td>Addresses – Local –Port:</td>
<td>assign a unique port number</td>
</tr>
<tr>
<td>Addresses - Partner - IP:</td>
<td>IP address of the <em>ibaPDA</em> PC</td>
</tr>
<tr>
<td>Addresses - Partner - Port:</td>
<td>Port number of the <em>ibaPDA</em> PC</td>
</tr>
</tbody>
</table>
Note
For assigning the port numbers, observe the notes in SIMATIC S7 configuration & engineering, page 9

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

■ Incrementing the "sequence_counter"
Increase the "sequence_counter" in the telegram data block with each rising edge of the DONE output.
Reset the "sequence_counter" to 0 if the CPU is restarted. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000).

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

<table>
<thead>
<tr>
<th>Communication block</th>
<th>usable for</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG_SEND (FC5)</td>
<td>X</td>
<td>These blocks allow sending a maximum of 240 bytes up to block version V3.0. The current block versions allow a data range of up to 8192 bytes for TCP and 2048 bytes for UDP.</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>The data length for S7-400 is limited to 240 bytes. To transmit greater data ranges, the block AG_LSEND has to be used²)</td>
</tr>
<tr>
<td>AG_LSEND (FC50)</td>
<td>X</td>
<td>The maximum data length is 8192 bytes for TCP and 2048 bytes for UDP. Please refer to the CP’s product information for information on the supported data range.</td>
</tr>
</tbody>
</table>

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

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

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

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

Other documentation

For more information on configuring the communication, see the STEP7 online help and the following FAQ from Siemens:

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

http://support.automation.siemens.com/ww/view/de/22790099
http://support.automation.siemens.com/ww/view/de/17853532
http://support.automation.siemens.com/ww/view/de/24693800

A detailed engineering example is provided in CPU 317 with CP 343-1 LEAN, page 46.

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

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

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

A separate connection has to be configured for each module configured in ibaPDA.

- Configuration of the CPU’s Ethernet interface in HW Config
  - Select the PN-IO interface of the CPU.
  - Click <Properties> in the General tab.
  - Assign the IP address and the subnet mask of the S7 controller

- Mapping the signal data
  Cyclically copy the desired signal data into the data blocks of the telegram modules at any position of your S7 program.
**Incrementing the "sequence_counter"**

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

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

**Parameterizing the connection**

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

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

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

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

**Note**

For assigning the port number, observe the notes in SIMATIC S7 configuration & engineering, page 9

**Calling the communication blocks in the S7 program of the CPU**

The following communication block is used:

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

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

\(^3\) See online help for system functions "Parameterizing the communication connections..."
Note
Always use the latest version of the SIMATIC NET communication blocks in your STEP7 project. You will find it in SIMATIC Manager under File - Open - Libraries - Standard Library - Communication Blocks.

Other documentation
For more information on configuring the communication, see the STEP7 online help and the following FAQ from Siemens:

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

http://support.automation.siemens.com/ww/view/de/29737950

A detailed configuration example is provided in CPU 317-2 with PN interface, page 39.

3.2.4 S7-1200 CPU with local Ethernet interface

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

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

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

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

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

A separate connection has to be configured for each module configured in ibaPDA.

Configuration of the Ethernet interface in the device configuration

- Select the device configuration.
- On the graphic, click the Ethernet port connected to ibaPDA.
- Select the tab General – Ethernet address.
- Assign the IP address and the subnet mask of the S7 controller.
Data interface TCP/UDP to SIMATIC S7  

ibaPDA-Interface-S7-TCP/UDP

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

■ Incrementing the "sequence_counter"
Increase the "sequence_counter" in the telegram data block with each rising edge of the DONE output.
Reset the "sequence_counter" to 0 if the CPU is restarted. In the event of an overflow, the counter must jump from 32767 to -32768 (0x7FFF → 0x8000).

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

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

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

- rem_ip_addr: IP address of the ibaPDA PC
- rem_port_nr: Port number of the ibaPDA PC

Note
For assigning the port number, observe the notes in SIMATIC S7 configuration & engineering, page 9

■ Calling the communication blocks
The following communication block is used:

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

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

### Other documentation

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

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

http://support.automation.siemens.com/ww/view/de/67196808

A detailed engineering example is provided in CPU 1212C (Firmware V2.0), page 55.

### 3.2.5 S7-1500-CPU with local Ethernet interface

#### Note

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

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

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

A separate connection has to be configured for each module configured in ibaPDA.

#### Configuration of the Ethernet interface in the device configuration

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

#### Mapping the signal data

Cyclically copy the desired signal data into the data blocks of the telegram modules at any position of your S7 program.
- **Incrementing the "sequence_counter"**

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

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

- **Parameterizing the connection**

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

- **interface_id**: Hardware ID of the local interface: see device configuration
- **id**: Connection ID
- **connection_type**: for TCP: 11 for UDP: 19
- **active_est**: for TCP: TRUE for UDP: FALSE
- **remote_address**: only for TCP: IP address of the ibaPDA PC
- **remote_port**: only for TCP: Port number of the ibaPDA PC: 4170
- **local_port**: Unique port number for each connection

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

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

---

**Note**

For assigning the port numbers, observe the notes in "SIMATIC S7 configuration & engineering, page 9"

---

- **Calling the communication blocks**

The following communication block is used:

- **TSEND_C**: to establish the connection and send the data

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

The block TSEND_C is available in two different variants.

- For CPU S7-1200 <= V3.x (see S7-1200-CPU with local Ethernet interface)
- For CPU S7-1500 and CPU S7-1200 >= V4.0

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

**Other documentation**

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

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

http://support.automation.siemens.com/ww/view/de/67196808

A detailed engineering example is provided in "CPU 1516 with PN interface, page 64."
3.3  ibaPDA configuration & engineering

Subsequently, the engineering for ibaPDA is described. If all system requirements are satisfied (see System requirements, page 7), the interface "S7 TCP/UDP" is displayed in the signal tree. The interface does not have to be added manually.

3.3.1  General settings

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

- Disconnect TCP/IP or UDP 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.

3.3.2  General interface settings

The tree structure of the ibaPDA I/O Manager contains the data interface "S7 TCP/UDP". The interface provides the following functions and configuration options:
Port no.
Port used on PC side.
The same port number has to be used in the S7 connection configuration (see \textit{S7 engineering examples}, page 38).

Reset port to default
The port number 4170 is set.

Allow port through firewall
When installing \textit{ibaPDA}, the default port numbers of the protocols used are automatically entered in the firewall. If the port number is changed here or if the interface was subsequently enabled, this port has to be enabled in the firewall here.

TCP Port / UDP Port
OK is displayed here if the socket can be opened on this port. ERROR is displayed if conflicts occur, e.g. if the port is already occupied.

Connection table
see \textit{Checking the connection}, page 32

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

Tip
If a TCP/IP or UDP connection to S7 exists already, right-click the interface and select \textit{Autodetect}. Then the correct modules are automatically created for all available connections.
<table>
<thead>
<tr>
<th>Module type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S7 TCP/UDP General</td>
<td>Module for any data structure with up to 4096 bytes length</td>
</tr>
<tr>
<td>S7 TCP/UDP Integer</td>
<td>Module for up to 32 analog signals (integer) and 32 digital signals</td>
</tr>
<tr>
<td>S7 TCP/UDP Real</td>
<td>Module for up to 32 analog signals (Real) and 32 digital signals</td>
</tr>
<tr>
<td>S7 UDP Request</td>
<td>Request module for a maximum of 1024 analog and 1024 digital signals.</td>
</tr>
<tr>
<td></td>
<td>See manual ibaPDA-Request-S7-UDP</td>
</tr>
<tr>
<td>S7 UDP Request Decoder</td>
<td>Request module for a maximum of 11728 digital signals transmitted in the form of a maximum of 733 words (1466 byte).</td>
</tr>
<tr>
<td></td>
<td>See manual ibaPDA-Request-S7-UDP</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Type for creating diagnostic modules, see Diagnosis, page 29</td>
</tr>
</tbody>
</table>

Table 1: Module types for ibaPDA-Interface-S7-TCP-UDP interface
3.3.3 General module settings

All modules have the following common setting options.

**Basic settings**

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

- **Locked**
  A module can be locked in order to prevent a change of module settings by accident or unauthorized users.

- **Enabled**
  Deactivated modules are excluded from signal acquisition.

- **Name**
  Here, you have to enter the plain text name as module designation.

- **Module No.**
  Internal reference number of the module.

- **Timebase**
  All signals of the module will be sampled on this timebase. It is recommended to adjust the timebase to the TCP/IP or UDP telegram cycle to be expected (multiple of the general ibaPDA sampling timebase) to be expected.

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

- **Swap analog signals**
  Option to change the order of the byte evaluation

- **Swap digital signals**
  Option to change the order of the byte evaluation
■ Module index
The module indices are created by a serial number 00....63 and an offset that corresponds to the module type and the license.
See also ☞ Header, page 11.

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

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

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

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

Other documentation
For a detailed description of additional options, see the ibaPDA manual.
3.3.5 Module type S7 TCP/UDP Integer
The integer module allows up to 32 analog values (integer) and 32 binary signals to be acquired. The module does not have any module-specific settings.

3.3.6 Module type S7 TCP/UDP Real
The real module allows up to 32 analog values (real) and 32 binary signals to be acquired. The following module settings are module-specific:

- Number of analog signals
  The number of analog signals to be acquired is configurable in the increments 8, 16 and 32 (the number of digital signals is fixed at 32).

3.3.7 Module type S7 TCP/UDP General
Using the general module allows any data structure with a maximum length of 4096 bytes to be acquired (see also footer in General information, page 8).

The following module settings are module-specific:

- Number of analog signals
  Number of maximum configurable analog signals.

- Number of digital signals
  Number of maximum configurable digital signals.

For signal configuration, enter the address, i.e. the offset in the telegram buffer, and the data type for each variable. Bear in mind that counting starts from the beginning of user data without header (i.e. the address is the offset from the data block – 6).
3.3.8  S7 UDP Request/S7 UDP Request Decoder module types

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

**Other documentation**

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

3.3.9  Module diagnostics

The tables *Analog* and *Digital* of the S7-TCP/UDP modules show the telegram contents.

![Generic TCP (4)](image)

The following errors may occur:

- No data are displayed:
  - The telegram DB on the S7 side is not filled correctly.
  - The connectors of the send block are connected incorrectly.

- Incorrect values are displayed:
  - The telegram DB on the S7 side is not filled correctly (offset error).
  - The byte order is set incorrectly, see *General module settings*, page 25
  - 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 25

- The telegrams do not arrive faster than approx. 200 ms with sequence error.
  - Problem with "delayed acknowledge", see *TCP/IP protocol variants*, page 37
4 Diagnosis

4.1 Checking the license

If the "S7 TCP/UDP" interface is not displayed in the menu tree of the ibaPDA I/O Manager, you can check in the I/O Manager under General - Settings - License info whether your license is recognized properly. 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 may be hidden. Click the Interfaces and enable the "Interface S7 TCP/UDP".
4.3 Log files

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

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

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

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

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

Examples:

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

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

Open a Windows command prompt.

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

With an existing connection you receive several replies.

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

With no existing connection you receive error messages.

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

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

Colors:

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

**Orange:**
The connection is OK, but the telegram cycle is significantly slower than the *ibaPDA* module timebase. It is recommended to adjust the module timebase to the telegram cycle.

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

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

- If the telegram counters do not increment continuously, the send blocks AG_LSEND are not called cyclically on the S7 side.

- If values in the columns "Incomplete errors" and/or "Sequence errors" are incremented, this points to one of the following errors:
  - The "message_length" in the DB does not have the expected value.
  - The "sequence_counter" in the DB is not incremented correctly.
  - The "delayed acknowledge" problem occurs (see \(TCP/IP\) protocol variants, page 37)
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.

![Add module dialog with Diagnostic module type](image)

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.

![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 S7 engineering examples

You will find different engineering examples on the DVD "iba Software & Manuals" under:

```
...\04_Libraries_and_examples\50_ibaPDA-Interface-S7-TCP_UDP\FixedTelegrams
```

<table>
<thead>
<tr>
<th>Example</th>
<th>ibaPDA project</th>
<th>S7 project</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEP7 project</td>
<td>ibaPDA_S7_TCP_UDP_Vxx.zip</td>
<td>S7CLASSIC_TCP_UDP_Vxx.zip</td>
</tr>
<tr>
<td>S7-300/-400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIA-Portal project</td>
<td></td>
<td>S7TIA_TCP_UDP_Vxx.zip</td>
</tr>
<tr>
<td>S7-1200/S7-1500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the projects, the following telegrams are transmitted to *ibaPDA* (IP address 192.168.50.203, port number 4170):

<table>
<thead>
<tr>
<th>Connection ID</th>
<th>Local port in S7</th>
<th>Protocol</th>
<th>ibaPDA module type</th>
<th>Module index</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2010</td>
<td>TCP/IP</td>
<td>Integer</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2011</td>
<td>TCP/IP</td>
<td>Real</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>2012</td>
<td>TCP/IP</td>
<td>Generic</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>2013</td>
<td>TCP/IP</td>
<td>Technostring</td>
<td>-</td>
</tr>
</tbody>
</table>
## 5.2.1 S7CLASSIC_TCP_UDP

The S7 project was created using STEP7 V5.5 SP2 and it contains the demo engineering of several configurations. 3 TCP/IP connections and 3 UDP connections are configured in each project.

Additionally, 2 variants of the TCP/IP-Technostrings and the reception of an ibaPDA watchdog telegram are included.

The associated ibaPDA project can be used for all configurations, see ibaPDA engineering example, page 76.

### 5.2.1.1 CPU 317-2 with PN interface

*Demo project: "CPU3xx with PN-IF"

The connection to ibaPDA is established through the integrated PN interface of the CPU.

- Configuration of the CPU’s Ethernet interface in HW Config

The IP address and the subnet mask are set in the HW Config:

---

4) The 2nd Technostring connection was only implemented in the S7-300/-400 projects. This connection is not implemented in the S7 TIA projects.

5) This connection does not exist in the S7-3xx projects, because only 8 connections are possible in the S7-CPU 317 used.
■ Programming the send blocks

In the main program (OB1), first the test signals (sine, cosine, triangle) are generated (FC 99), then the send trigger is derived from the CPU clock memory M10.0.

Next the individual functions for sending the TCP/IP Technostring and UDP telegrams are called.

5.2.1.1 TCP/IP data telegrams

For each ibaPDA module type, a function FC100 (TCP-Integer), FC101 (TCP-Real) and FC102 (TCP-Generic) is called from OB1.

Each function has the same sequence:

1. Mapping the signal data in the TCP/IP data block
2. Default setting of the telegram headers
3. Calling the TCP/IP send blocks (FB151)
Network 1: analog demo signals (INT)

move demo signals to Transfer-DE

...  
MOVE DB222 DBW  
32 analog values INT 'ibaPDA_TCP/IP_INT00gE'.  
DB222 DBW  
IN  
EN

SIMULATED

...  
MOVE DB222 DBW  
32 analog values INT 'ibaPDA_TCP/IP_INT00gE'.  
DB222 DBW  
IN  
EN

Network 2: digital demo signals

move demo signals to Transfer-DE

...  
MOVE DB222 DBW  
DB222 DBW  
IN  
EN

clock  
marker  
byte of  
CPU  
clockbyte  

Network 3: preset PDA-Header - module 0

IbaPDA TCP/IP S7 integer module

data length: 74 byte

...  
MOVE DB222 DBW  
total length of the DH in bytes 'ibaPDA_TCP/IP_INTEGER'.  
message  
length  

MOVE DB222 DBW  
ibaPDA module  
index  
(00...63) 'ibaPDA_TCP/IP_INTEGER'.  
module  
index  

OUT  
IN  
ENO
In function block FB151 (ibaTCP-CPU), both the calls of communication blocks TCON and TSEND and the incrementing of the "sequence_counter" are encapsulated. The block is configured completely through its interface; it can be instantiated multiple times and can thus be copied easily into another S7-300 project.
Meaning of the parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IN</strong></td>
<td></td>
</tr>
<tr>
<td>SEND</td>
<td>Send trigger (Impuls)</td>
</tr>
<tr>
<td>RESET</td>
<td>Resetting and re-establishing the connection</td>
</tr>
<tr>
<td>CON_ID</td>
<td>Unique connection ID</td>
</tr>
<tr>
<td>DEST_IP_1</td>
<td>Byte 1 of the IP address of the ibaPDA PC</td>
</tr>
<tr>
<td>DEST_IP_2</td>
<td>Byte 2 of the IP address of the ibaPDA PC</td>
</tr>
<tr>
<td>DEST_IP_3</td>
<td>Byte 3 of the IP address of the ibaPDA PC</td>
</tr>
<tr>
<td>DEST_IP_4</td>
<td>Byte 4 of the IP address of the ibaPDA PC</td>
</tr>
<tr>
<td>LOCAL_PORT</td>
<td>Unique local port used</td>
</tr>
<tr>
<td>remote_port:</td>
<td>Port used of the ibaPDA PC (default: 4170)</td>
</tr>
<tr>
<td>DATA</td>
<td>ANY pointer to data range</td>
</tr>
<tr>
<td><strong>OUT</strong></td>
<td></td>
</tr>
<tr>
<td>CONNECTED</td>
<td>Status: Connection established</td>
</tr>
<tr>
<td>ERROR</td>
<td>Status: Error</td>
</tr>
<tr>
<td>ERROR_TCON</td>
<td>Status: Error TCON block</td>
</tr>
<tr>
<td>COUNT_TCON</td>
<td>Counter calls TCON block</td>
</tr>
<tr>
<td>ERROR_TSEND</td>
<td>Status: Error TSEND block</td>
</tr>
<tr>
<td>COUNT_TSEND</td>
<td>Counter calls TSEND block</td>
</tr>
</tbody>
</table>

Please note that you have to use a separate connection ID (CON_ID) and a separate local port (LOCAL_PORT) for each call, but the port at the ibaPDA PC (REMOTE_PORT) is always the same (see 🔄 General interface settings, page 22).

5.2.1.1.2 Technostring TCP/IP

The S7 provides two ways of transmitting a Technostring, either as "String" or as "Array of Char". A function exists for each alternative:

- In the example of function FC105, a real value is converted to an ASCII string and inserted into an existing Technostring, which is then transmitted with function block FB154.
- In the example of function FC106, an "Array of Char" is assigned a text by default, a real value converted to ASCII is inserted into it and then it is transmitted with function block FB154.

In both cases, FB154 is called in which the calls of the communication block TCON and TSEND are encapsulated. Block FB154 is configured completely through its interface, it can be instantiated multiple times and can thus be copied easily into another S7-300 project.
The parameters are identical to those of FB151, see TCP/IP data telegrams, page 40.
Please note also that each call requires a separate connection ID (CON_ID) and a separate local port (LOCAL_PORT).
5.2.1.1.3 UDP data telegrams

UDP data transmission uses the same principle as TCP/IP:

For each *ibaPDA* module type, a function FC110 (UDP-Integer), FC111 (UDP-Real) and FC112 (UDP-Generic) is called from OB1.

![Diagram of UDP data telegrams]

Each function has the same sequence, identical to the TCP/IP telegrams:

1. Mapping the signal data in the telegram DB
2. Default setting of the telegram headers
3. Calling the UDP send blocks (FB161)

Both the call of the communication blocks TCON and TUSEND and the incrementing of the "sequence_counter" are encapsulated in function block FB161.

Block FB161 is configured completely through its interface, it can be instantiated multiple times and can thus be copied easily into another S7-300 project.
Appendix ibaPDA-Interface-S7-TCP/UDP

5.2.1.2 CPU 317 with CP 343-1 LEAN

Demo project: "CPU3xx with CP343-1"
The connection to ibaPDA is established through the CP343-1 LEAN.

Configuration of the SIMATIC NET Ethernet-CP in HW Config

The parameters are identical to those of FB151, see TCP/IP data telegrams, page 40.

Please note also that each call requires a separate connection ID (CON_ID) and a separate local port (LOCAL_PORT). The REMOTE_PORT must correspond to the setting in the ibaPDA PC (see General interface settings, page 22).
Creating the TCP/UDP connections in NetPro
A separate connection has to be configured for each module.
It is of the type "TCP connection" or "UDP connection" and the connection partner is unspecified.

The name is freely selectable and "Active connection establishment" has to be enabled at TCP connections. The displayed block parameters are important, because they have to be parameterized as shown at the communication block that is used.
In the *Addresses* tab you can preset the local port, the partner port and the partner IP address (*ibaPDA PC*).

![Properties - TCP connection window](image)

Please note that you have to use a separate local port for each connection, but the partner port (*ibaPDA PC*) is always the same (see *General interface settings*, page 22).

**Programming the send blocks**

In the main program (OB1), first the test signals (sine, cosine, triangle) are generated (FC 99), then the send trigger is derived from the CPU clock memory M10.0.

Next the individual functions for sending the TCP/IP Technostring and UDP telegrams are called.
5.2.1.2.1 TCP/IP data telegrams

For each ibaPDA module type, a function FC100 (TCP-Integer), FC101 (TCP-Real) and FC102 (TCP-Generic) is called from OB1.

**Netzwerk 3**: ibaPDA TCP/IP S7 module index 0 (Integer)

| FC100: Fill DB222 and send it over TCP/IP |

```plaintext
FC100
"ibaPDA_TCP_Integer"

... EN ENO...
```

**Netzwerk 4**: ibaPDA TCP/IP S7 module index 100 (Real)

| FC101: Fill DB223 and send it over TCP/IP |

```plaintext
FC101
"ibaPDA_TCP_Real"

... EN ENO...
```

Each function has the same sequence:

1. Mapping the signal data in the TCP/IP telegram block
2. Default setting of the telegram headers
3. Calling the TCP/IP send blocks (FB152)

In function block FB152, both the calling of communication block AG_SEND (FC5) and incrementing of the "sequence_counter" are encapsulated.

Block FB152 is configured completely through its interface, it can be instantiated multiple times and can thus be copied easily into another S7-300 project.

Unchanged it can also be used for UDP connections.
**Meaning of the parameters:**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IN</strong></td>
<td></td>
</tr>
<tr>
<td>SEND</td>
<td>Send trigger (Impuls)</td>
</tr>
<tr>
<td>CON_ID</td>
<td>Unique connection ID</td>
</tr>
<tr>
<td>LADDR</td>
<td>Module start address (see HW Config)</td>
</tr>
<tr>
<td>DATA</td>
<td>ANY pointer to data range</td>
</tr>
<tr>
<td><strong>OUT</strong></td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>Status: Error</td>
</tr>
<tr>
<td>ERROR_AGSEND</td>
<td>Status: Error block AGSEND</td>
</tr>
<tr>
<td>COUNT-AGSEND</td>
<td>Counter calls block AGSEND</td>
</tr>
</tbody>
</table>
5.2.1.2.2 Technostring TCP/IP

The S7 provides two ways of transmitting a Technostring, either as "String" or as "Array of Char". A function exists for each alternative:

- In the example of function FC105, a real value is converted to an ASCII string and inserted into an existing Technostring, which is then transmitted with function block FB155.

- In the example of function FC106, an "Array of Char" is assigned a text by default, a real value converted to ASCII is inserted into it and then it is transmitted with function block FB155.

In FB155, the call of communication block AG_SEND is encapsulated. Block FB155 is configured completely through its interface, it can be instantiated multiple times and can thus be copied easily into another S7-300 project.

The meaning of the parameters is identical to those of FB152, see "TCP/IP data telegrams", page 50.

Please note here, too, that you need a separate connection ID (CON_ID) for each call.
5.2.1.2.3 UDP data telegrams

UDP data transmission works according to the same principle as TCP/IP:

For each ibaPDA module type, a function FC110 (UDP-Integer), FC111 (UDP-Real) and FC112 (UDP-Generic) is called from OB1.

Each function has the same sequence:

1. Mapping the signal data in the telegram DB
2. Default setting of the telegram headers
3. Calling the send blocks

Both calling of communication block AG_SEND (FC5) and incrementing of the „sequence_counter“ are encapsulated in function block FB152.

Block FB152 is configured completely through its interface, it can be instantiated multiple times and can thus be copied easily into another S7-300 project.

There is no difference to sending via a TCP/IP connection. TCP/IP and UDP are distinguished based on the connection type in the HW configuration, referenced by the connection ID.
5.2.1.3  CPU 412-2 with PN interface

Demo project: "CPU4xx with PN-IF"
The connection to *ibaPDA* is established through the integrated PN interface of the CPU.
The example is identical to the example in *CPU 317-2 with PN interface*, page 39 with one exception:
The "local_device_id", which is entered into the connection parameters within the TCP-/UDP send blocks (FB151, FB154, FB157, FB161), depends on the CPU type:
- for CPU3xx: local_device_id = 2,
- for CPU4xx: local_device_id = 5.
See also the Step7 online help.

5.2.1.4  CPU 412 with CP 443-1

Demo project: "CPU4xx with CP443-1"
The connection to *ibaPDA* is established through the CP443-1.
The example is identical to the example in *CPU 317 with CP 343-1 LEAN*, page 46, with the following exceptions:
- FB153, in which the send block AG_LSEND (FC50) is encapsulated, is called within the functions FC100, FC101, FC102, FC110, FC111, FC112.
- FB156, in which the send block AG_LSEND (FC50) is encapsulated, is called within the functions FC105, FC106.
- Reception of the *ibaPDA* watchdog telegram is accomplished in function block FB159, in which the receive block AG_LRECV (FC60) is encapsulated.
5.2.2 S7TIA_TCP_UDP_V13_SP1

The project was created with TIA Portal V13 Professional SP1 and contains the demo engineering of several configurations.

3 TCP/IP connections and 3 UDP connections are implemented in each project. Additionally, sending of a TCP/IP Technostring and reception of an ibaPDA watchdog telegram are configured.

The associated ibaPDA project can be used for all configurations, see ibaPDA engineering example, page 76.

5.2.2.1 CPU 1212C (Firmware V2.0)

Demo project: ibaDemo_S7-1212C with PN-IF (CPU 1212C AC/DC/Rly)

- Configuration of the CPU's Ethernet interface in the device configuration

The connection to ibaPDA is established through the integrated PN interface of the CPU. The IP address and the subnet mask are set in the device configuration:

- Select the device configuration.
- On the graphic, click the Ethernet port connected to ibaPDA.
- Select the tab General – Ethernet addresses.
- Enter the IP address and the subnet mask and the router address if necessary.
The HW identifier of the Ethernet interface is required to parameterize the send blocks. It can be taken from the device configuration:
Programming the send blocks

In the main program (OB1), first the test signals (sine, cosine, triangle) are generated (FC 99), then the send trigger is derived from the CPU clock memory M10.0.

Next the individual functions for sending the TCP/IP Technostring and UDP telegrams are called.

5.2.2.1 TCP/IP data telegrams

For each ibaPDA module type, the function FC100 (ibaDemo-Integer), FC101 (ibaDemo-Real) and FC102 (ibaDemo-Generic) are called from OB1 (Main).

Each function has the same sequence (here using the example of FC100 – TCP/IP-Integer):

1. Mapping the signal data into the TCP/IP data block (3 analog values "sine, cosine, triangle" and 8 digital values)
2. Default setting of the telegram headers (length, module index)
3. Incrementing the "sequence_counter"

4. Creating the data structure TCON-Param in the local data range

<table>
<thead>
<tr>
<th>ibaDemo-Integer</th>
<th>Datatype</th>
<th>Defaultwert</th>
<th>Kommentar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCON_PARAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK_LENGTH</td>
<td>UChar</td>
<td></td>
<td>byte length of SDT</td>
</tr>
<tr>
<td>ID</td>
<td>UINT</td>
<td></td>
<td>reference to the connection</td>
</tr>
<tr>
<td>CONNECTION_TYPE</td>
<td>UINT</td>
<td></td>
<td>17: TCP/IP, 18: GO on TCP, 19: UDP</td>
</tr>
<tr>
<td>ACTIVE_EST</td>
<td>BOOL</td>
<td></td>
<td>active/passive connection establishment</td>
</tr>
<tr>
<td>LOCAL_DEVICE_ID</td>
<td>UINT</td>
<td></td>
<td>1: local IF interface</td>
</tr>
<tr>
<td>LOCAL_TSAP_ID_LEN</td>
<td>UINT</td>
<td>byte length of local TSAP id/port number</td>
<td></td>
</tr>
<tr>
<td>REM_SUBNET_ID_LEN</td>
<td>UINT</td>
<td>byte length of remote subnet id</td>
<td></td>
</tr>
<tr>
<td>REM_STADDR_LEN</td>
<td>UINT</td>
<td></td>
<td>byte length of remote IP address</td>
</tr>
<tr>
<td>NEXT_STADDR_LEN</td>
<td>UINT</td>
<td></td>
<td>byte length of next station address</td>
</tr>
<tr>
<td>LOCAL_TSAP_ID</td>
<td>ARRAY[1:16] of Byte</td>
<td>TSAP id/local port number</td>
<td></td>
</tr>
<tr>
<td>REM_SUBNET_ID</td>
<td>ARRAY[1:6] of UINT</td>
<td>remote subnet id</td>
<td></td>
</tr>
<tr>
<td>REM_STADDR</td>
<td>ARRAY[1:6] of UINT</td>
<td>remote IP address</td>
<td></td>
</tr>
<tr>
<td>REM_TSAP_ID</td>
<td>ARRAY[1:16] of Byte</td>
<td>TSAP id/remote port number</td>
<td></td>
</tr>
<tr>
<td>NEXT_STADDR</td>
<td>ARRAY[1:6] of Byte</td>
<td>next station address</td>
<td></td>
</tr>
<tr>
<td>SPARE</td>
<td>UINT</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>Constant</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Return</td>
<td>ibaDemo-Integer</td>
<td>Void</td>
<td></td>
</tr>
</tbody>
</table>

5. Entering the connection parameters in the data structure "TCON_Param"
6. Calling the send block TSEND_C

Please observe the following:

- The HW identifier (interface id) has to be selected to match the physical Ethernet interface that is used. It is specified in the device configuration. Here "Local“PROFINET-Schnittstelle_1” or “64”.
- Select connection type 17 for TCP/IP.
- Use a separate connection ID and a separate local port for each call.
- The port at the ibaPDA PC (RemotePort) is always the same (see General interface settings, page 22).
5.2.2.1.2 Technostring TCP/IP

A Technostring is transmitted as "Array_of_Char" here.

The function FC103 (ibaDemo-Technostring) is called from OB1 (Main).

The following actions are performed in FC103:

1. Two integer values are converted and attached to a fixed string.
2. The string is converted into an "Array_of_Char" and stored in a DB.

```plaintext
1. //--- move demo string to ibaPDA transfer-OB
2. //--- adjust this according to your signals you want to send to ibaPDA
3. #sStatus := RC_LOC_T(#DTL);
4. #sTime := #DTL_HOUR;
5. #sMinute := #DTL_MINUTE;
6. #sSec := CONCAT(1H1, 'Hello ibaWorld! It is now ', #IN1: #UNINT_TO_STRING(#DTL_HOUR));
7. #sMin := CONCAT(1H1, 'Hello ibaWorld! It is now ', #IN2: #UNINT_TO_STRING(#DTL_MINUTE));
8. #sSec := CONCAT(1H1, 'Hello ibaWorld! It is now ', #IN3: #UNINT_TO_STRING(#DTL_MINUTE));
9. //Step_TO_Chars(Step=#Str, pChars:=0,
10.   Cnt:=#Cnt,
11.   Chars:="ibaPDA_TCP/IP_TECHNOSTRING".Technostring);

3. The data structure TCON_Param is created in the local data range and filled with the TCP/IP connection parameters.

```
4. The send block TSEND_C is called.

![Code snippet demonstrating TSEND_C](image)

Please note also that the call requires a separate connection ID and a separate local port. The remote port must match the setting in the ibaPDA Technostring.

### 5.2.2.1.3 Data telegrams UDP

**Calling the communication blocks in the S7 program of the CPU**

UDP data transmission works according to the same principle as TCP/IP:

For each ibaPDA module type, a function FC110 (ibaDemo-Integer-UDP), FC111 (ibaDemo-Real-UDP) and FC112 (ibaDemo-Generic-UDP) is called from OB1 (Main).

![Network Diagrams for UDP](image)

Each function has the same sequence, see TCP/IP data telegrams:

1. Mapping the signal data in the UDP data block (3 analog values "sine, cosine, triangle" and 8 digital values)
2. Default setting of the telegram headers (length, module index)
3. Incrementing the "sequence_counter"
4. Creating the data structures TCON-Param and TADDR_Param in the local data range

<table>
<thead>
<tr>
<th>Name</th>
<th>Datatype</th>
<th>Defaultwert</th>
<th>Kommentar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>InOut</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;Hinzufügen&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tCONPARAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLOCK_LENGTH</td>
<td>UInt</td>
<td></td>
<td>byte length of SDT</td>
</tr>
<tr>
<td>ID</td>
<td>CONN_DOC</td>
<td></td>
<td>reference to the connection</td>
</tr>
<tr>
<td>CONNECTION_TYPE</td>
<td>USint</td>
<td></td>
<td>17: TCP/IP, 18: ISO on TCP, 19: UDP</td>
</tr>
<tr>
<td>ACTIVE_EST</td>
<td>Bool</td>
<td></td>
<td>active/passive connection establishment</td>
</tr>
<tr>
<td>LOCAL_DEVICE_ID</td>
<td>USint</td>
<td></td>
<td>1: local IE interface</td>
</tr>
<tr>
<td>LOCAL_TSAP_ID</td>
<td>USint</td>
<td></td>
<td>byte length of local TSAP id/port number</td>
</tr>
<tr>
<td>REM_SUBNET_ID</td>
<td>USint</td>
<td></td>
<td>byte length of remote subnet id</td>
</tr>
<tr>
<td>REM,STADDR_LEN</td>
<td>USint</td>
<td></td>
<td>byte length of remote IP address</td>
</tr>
<tr>
<td>REM,TSAP_ID_LEN</td>
<td>USint</td>
<td></td>
<td>byte length of remote port/TSAP id</td>
</tr>
<tr>
<td>NEXT_STADDR_LEN</td>
<td>USint</td>
<td></td>
<td>byte length of next station address</td>
</tr>
<tr>
<td>LOCAL_TSAP_ID</td>
<td>Array[1..16] of Byte</td>
<td></td>
<td>TSAP id/local port number</td>
</tr>
<tr>
<td>REM,STADDR</td>
<td>Array[1..6] of USInt</td>
<td></td>
<td>remote subnet id</td>
</tr>
<tr>
<td>REM,TSAP_ID</td>
<td>Array[1..16] of Byte</td>
<td></td>
<td>TSAP id/remote port number</td>
</tr>
<tr>
<td>NEXT_STADDR</td>
<td>Array[1..6] of Byte</td>
<td></td>
<td>next station address</td>
</tr>
<tr>
<td>SPARE</td>
<td>Word</td>
<td></td>
<td>reserved</td>
</tr>
<tr>
<td>tADDR_Param</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM,IP_ADDR</td>
<td>Array[1..6] of USInt</td>
<td></td>
<td>remote station address</td>
</tr>
<tr>
<td>REM,PORT_NR</td>
<td>UInt</td>
<td></td>
<td>remote port number</td>
</tr>
<tr>
<td>RESERVED</td>
<td>Word</td>
<td></td>
<td>unused; has to be 0</td>
</tr>
</tbody>
</table>

5. Entering the connection parameters in the data structure TCON_Param

```c
41 //--- TCON parameter
42
43 //--- set default values
44 #TCONPARAM.BLOCK_LENGTH := 44;
45 #TCONPARAM.REM_SUBNET_ID_LEN := USINT#0; // not used
46 #TCONPARAM.NEXT_STADDR_LEN := USINT#0;  // not used
47 #TCONPARAM.LOCAL_DEVICE_ID := USINT#1;  // 1: local IE interface
48 #TCONPARAM.ID := 61;  // use unique ID if you have multiple connect
49 #TCONPARAM.CONNECTION_TYPE := USINT#19; // Protocol variant: 19: UDP
50 #TCONPARAM.ACTIVE_EST := false.
51 #TCONPARAM.LOCAL_PORT := 53.0
52 #TCONPARAM.LOCAL_TSAP_ID := USINT#0;  // 2015 - 0x07DF
53 #TCONPARAM.LOCAL_TSAP_ID[1] := USINT#1627; // use different local port number if y
54 #TCONPARAM.LOCAL_TSAP_ID[2] := USINT#1858; // use different local port number if y
55 #TCONPARAM.REM,STADDR := USINT#0; // remote port
56 #TCONPARAM.REM,TSAP_ID_LEN := USINT#0; // remote port
57```

---

62 Issue 2.3
6. Entering the remote address and port in the data structure TADDR_Param

```c
// --- ADDR parameter for UDP

// --- REMOTE IP ADDRESS
#define TADDR_PAR.REM_IP_ADDR1 := 192;    // adjust according to IP address of ibaPDA
#define TADDR_PAR.REM_IP_ADDR2 := 168;    // adjust according to IP address of ibaPDA PC
#define TADDR_PAR.REM_IP_ADDR3 := 10;     // adjust according to IP address of ibaPDA PC
#define TADDR_PAR.REM_IP_ADDR4 := 200;    // adjust according to IP address of ibaPDA PC

// --- REMOTE PORT
#define TADDR_PAR.REM_PORT := 4170;       // adjust remote port number according to
#define TADDR_PAR.REM_PORT := 1000;       // adjust remote port number according to
```

7. Calling the block TCON to establish the connection

8. Calling the send block TUSEND

Please observe the following:

- The HW identifier (interface id) has to be selected to match the physical Ethernet interface that is used. It is specified in the device configuration.

- Select connection type 19 for UDP.

- Use a separate connection ID and a separate local port for each call.

- The remote address and port are not entered in the TCON_Param structure, but in the TADDR_Param structure. It is specified in send block TUSEND at the ADDR connector.
5.2.2.2  CPU 1516 with PN interface

Demo project: "ibaDemo-S7-1516 with PN-IF"
The connection toibaPDA is established through the integrated PN interface of the CPU.

■ Configuration of the CPU's Ethernet interface in the device configuration
The own IP address and the subnet mask are set in the device configuration:

- Select the device configuration.
- On the graphic, click the Ethernet port connected to ibaPDA.
- Select the tab General – Ethernet addresses.
- Enter the IP address and the subnet mask and the router address if necessary.

The HW identifier of the Ethernet interface is required to parameterize the send blocks. It can be taken from the device configuration:
■ Programming the send blocks

In the main program (OB1), first the test signals (sine, cosine, triangle) are generated (FC 99),
then the send trigger is derived from the CPU clock memory M10.0.
Next the individual functions for sending the TCP/IP, Technostring and UDP telegrams are called.

5.2.2.2.1 TCP/IP data telegrams

For each ibaPDA module type, a function FC100 (ibaDemo-Integer), FC101 (ibaDemo-Real) and
FC102 (ibaDemo-Generic) is called from OB1 (Main).

Each function has the same sequence (here using the example of FC100 – TCP/IP-Integer):

1. Mapping the signal data in the TCP/IP data block (3 analog values and 8 digital values)
2. Default setting of the telegram headers (length, module index)
3. Incrementing the "sequence_counter"
4. Creating the data structure TCON_IP_v4 in the local data range

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Default value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCON_IP_v4</td>
<td></td>
<td></td>
<td>TCON IP version v4</td>
</tr>
<tr>
<td>InterfaceID</td>
<td>Integer</td>
<td></td>
<td>Identifier of the Interface submodule</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td></td>
<td>Configuration/identifier</td>
</tr>
<tr>
<td>ConnectionType</td>
<td>Byte</td>
<td></td>
<td>Type of connection: 11=TCP, 19=UDP (17=TCP/IP)</td>
</tr>
<tr>
<td>ActiveEstablished</td>
<td>Boolean</td>
<td></td>
<td>Active/passive connection establishment</td>
</tr>
<tr>
<td>RemoteAddress</td>
<td>String</td>
<td></td>
<td>Remote IP address (IPv4)</td>
</tr>
<tr>
<td>RemotePort</td>
<td>Integer</td>
<td></td>
<td>Remote UDP/TCP port number</td>
</tr>
<tr>
<td>LocalPort</td>
<td>Integer</td>
<td></td>
<td>Local UDP/TCP port number</td>
</tr>
<tr>
<td>SSEND_C</td>
<td>Struct</td>
<td></td>
<td>Struct definition</td>
</tr>
<tr>
<td>LocalPort</td>
<td>Integer</td>
<td></td>
<td>Local port number</td>
</tr>
</tbody>
</table>

5. Entering the TCP/IP connection parameters

6. Calling the send block TSEND_C

```c
//--- TCP parameter

//--- set default values
#STCOM_IP.InterfaceId := 72; // according to used interface on CPU: X1=64, X2=72
#STCOM_IP.ConnectionType := 11; // Protocol variant: TCP=11

//--- Reference to the connection
#STCOM_IP.ID := 2; // use unique IDs if you have multiple connections

//--- active connection establishment
#STCOM_IP.ActiveEstablished := true;

//--- LOCAL PORT
#STCOM_IP.LocalPort := 2010; // use different local port number if you for example have multiple interfaces

//--- REMOTE PORT
#STCOM_IP.RemotePort := 4170; // adjust remote port number according to ibaPDA

//--- REMOTE IP ADDRESS
#STCOM_IP.RemotelAddress.ADDR(1) := 192; // adjust according to IP address of ibaPDA PC
#STCOM_IP.RemotelAddress.ADDR(2) := 160; // adjust according to IP address of ibaPDA PC
#STCOM_IP.RemotelAddress.ADDR(3) := 69; // adjust according to IP address of ibaPDA PC
#STCOM_IP.RemotelAddress.ADDR(4) := 233; // adjust according to IP address of ibaPDA PC

//--- call TSEND_C send send data to ibaPDA

TDB->ibaTCP_INT("send_trigger", // send trigger
       CONT := true,
       LEN := INT_TO_UINT("ibaPDA_TCP_ITB_INTER1S.message_length"),
       DOME := #TSEND_C.DOME,
       BUSW := #TSEND_C.BUSW,
       ERRID := #TSEND_C.ERRID,
       STATUS := #TSEND_C.STATUS,
       CONNECT := #STCOM_IP,
       DATA := #DB222.dbw0.0 byte 74, // adjust telegram data range
       COM_RSB := #TSEND_C.COM_RSB);`
```
Please observe the following:

- The HW identifier (interface id) has to be selected to match the physical Ethernet interface that is used. It is specified in the device configuration. Here "Local~PROFINET-Schnittstelle_2" or "72".
- Select connection type 11 for TCP/IP.
- Use a separate connection ID and a separate local port for each call.
- The port at the ibaPDA PC (RemotePort) is always the same (see General interface settings, page 22).

5.2.2.2.2 Technostring TCP/IP

A Technostring is transmitted as "Array_of_Char" here.

The function FC103 (ibaDemo-Technostring) is called from OB1 (Main).

The function FC103 is identical to that in Technostring TCP/IP with the following difference: The TCP/IP connection parameters are defined in the data structure TCON_IP_v4 (instead of TCON_Param).

5.2.2.2.3 CPU 1516_Data telegrams UDP

UDP data transmission works according to the same principle as TCP/IP:

For each ibaPDA module type, a function FC110 (UDP-Integer), FC111 (UDP-Real) and FC112 (UDP-Generic) is called from the main program (Main).
Each function has the same sequence (here using the example of FC110):

1. Mapping the signal data in the UDP data block (3 analog values "sine, cosine, triangle" and 8 digital values), see TCP/IP data telegrams, page 65
2. Default setting of the telegram headers (length, module index), see TCP/IP data telegrams, page 65
3. Incrementing the "sequence_counter", see TCP/IP data telegrams, page 65
4. Creating the data structure TCON_IP_v4 in the local data range, see TCP/IP data telegrams, page 65
5. Entering the connection parameters in the data structure "TCON_IP"
6. Entering the remote address and port in the data structure "TADDR_PAR"
7. Calling the send block TSEND_C

---

//--- TCON parameter
//---
//--- set default values
#define TCON_ID.InterfaceId := 70; // according to used interface on CPU: X1=64, X2=72, CP1563 = 261
#define TCON_ID.ConnectionType := 19; // Protocol variant: UDP=19
//--- Reference to the connection
#define TCON_ID.ID := 6; // use unique IDs if you have multiple connections
//--- active connection establishment
#define TCON_ID.ActiveEstablished := 0;
//--- LOCAL PORT
#define TCON_ID.LocalPort := 2015; // use different local port number if you have multiple connections

---

//--- ADDR parameter for UDP
//--- REMOTE ID ADDRESS
#define TADDR_PAR.REM_IP_ADDR[1] := 192; // adjust according to IP address of ibaPDA PC
#define TADDR_PAR.REM_IP_ADDR[2] := 168; // adjust according to IP address of ibaPDA PC
#define TADDR_PAR.REM_IP_ADDR[3] := 60; // adjust according to IP address of ibaPDA PC
#define TADDR_PAR.REM_IP_ADDR[4] := 208; // adjust according to IP address of ibaPDA PC

---

//--- REMOTE PORT
#define TADDR_PAR.REM_PORT_NR := 4170; // adjust remote port number according to ibaPDA
#define TADDR_PAR.REVERS := 1490;

---

//--- call TSEND_C send data to ibaPDA
"IDB-ibaUDP-INT"(SEQ:="send_trigger", // send trigger
CUNT:=1,
LEN:=INT_TO_UINT("ibaPDA_UDP_INTEGER".message_length),
DONE:=#TSEND_C.DONE,
BUSY:=#TSEND_C.BUSY,
ERROR:=#TSEND_C.ERROR,
STATUS:=#TSEND_C.STATUS,
CONNECT=#TCON.IP,
DATA:="ibaUDP冰雪.0 byte 74", // adjust telegram data range
ADD:=#TADDR_PAR,
COM_RST=#TSEND_C.COM_RST);
Please observe the following:

- The HW identifier (interface id) has to be selected to match the physical Ethernet interface that is used. It is specified in the device configuration.
- Select connection type 19 for UDP.
- Use a separate connection ID and a separate local port for each call.
- The remote address and port are not entered in the TCON_Param structure, but in the TADR_PARAM structure. It is specified in send block TSEND_C at the ADDR connector.

5.2.2.3 CPU 1516 with PN interface and configuration dialog

Demo project: "ibaDemo_S7-1516 with PN-IF (ConfigDialog)"
The connection to ibaPDA is established through the integrated PN interface of the CPU.

- Configuration of the CPU's Ethernet interface in the device configuration
The own IP address and the subnet mask are set in the device configuration:
The HW identifier can be taken from the device configuration:
See CPU 1516 with PN interface, page 64.

- Programming the send blocks
In the main program (OB1), first the test signals (sine, cosine, triangle) are generated (FC 99), then the send trigger is derived from the CPU clock memory M10.0.

Then the function FC120 (Connections) is called in which all TCP/IP and UDP send blocks are configured in their properties dialog.

Call from "Main (OB1)"

In FC120, a send block TSEND_C is called for each connection.
When creating the send block, the instance DB is generated automatically. It has the name "TSEND_C_DB" assigned to it manually here.

The connection is parameterized in the block's properties dialog (context menu).

Properties TSEND_C – Connection parameters:
Enter the following parameters here:

- Select the local Ethernet interface, here "PROFINET-Schnittstelle_2"
- Configuration type: "Use program blocks"
- Enter a DB name for the connection data; "Send_DB_x" is entered as the name here. The DB is created automatically; afterwards the remaining parameters can be entered in the dialog:
  - Connection type: TCP
  - Connection ID: unique number
  - Active connection establishment
  - Local unique port number

Partner data:

- Endpoint: unspecified
- Address: IP address of the ibaPDA PC
- Partner port: 4170

**Note**

Due to an error in the TIA Portal software, the partner port is marked as defective here. This is probably because it is not allowed to configure multiple connections with the same partner port.

Remedy: Open the connection data DB and enter the port number 4170 there.
Properties TSEND_C – Block parameters:
Here the block connections are made. Alternatively, the connections can be established directly at the block connector.
For each connection, a send block with the associated data blocks is created, that is 7 connections in total (3 x TCP/IP data, 1 x TCP/IP Technostring, 3 x UDP data).

For each send block, an instance DB "TSEND_C_DB_x" and a data block "Send_DB_x" for the connection data is created. These data blocks are stored under Program blocks - System blocks - Program resources.

### Mapping the sending data

For each telegram type, a function (FC100, . . . ) is called.

Each function has the same sequence:

1. Mapping the signal data in the telegram data block:
   three analog values (sine, cosine, triangle) and 8 digital values
2. Default setting of the telegram headers (length, module index)
3. Incrementing the "sequence_counter"

The steps of entering the connection parameters and calling the send block are omitted here, as this is accomplished in FC120 already.
5.2.2.4 CPU 1516 with CP1543-1

Demo project: Demo project: "ibaDemo-S7-1516 with CP1543-1"
The connection to ibaPDA is established through the external CP interface 1543-1.

■ Configuration of the interface of CP1543-1 in the device configuration
The own IP address and the subnet mask are set in the device configuration:

- Select the device configuration.
- On the graphic, click the Ethernet port connected to ibaPDA.
- Select the tab General – Ethernet addresses.
- Enter the IP address and the subnet mask and the router address if necessary.

The HW identifier of the Ethernet interface is required to parameterize the send blocks. It can be taken from the device configuration:
All blocks and functions are identical to the example CPU 1516 with PN interface, page 64. The only difference: The hardware identifier (interface id) must match the CP1543-1.

For the Ethernet interface of the CPU1516C:

- "Local~PROFINET-Schnittstelle_2"

For the Ethernet interface of the CP1543-1:

- "Local~CP_1543-1_1~Ethernet-Schnittstelle_1" or 261.
5.3 **ibaPDA engineering example**

This *ibaPDA* configuration applies to all SIMATIC engineering examples mentioned above.

5.3.1 **Configuration of data telegrams**

6 modules are created in the I/O Manager. The standard port number 4170 is used for all connections.

The acquired signals are entered and activated in the *Analog* and *Digital* tabs of the individual modules.
5.3.2 Configuration of Technostring

Under the Technostring menu item, Technostrings with the following parameters are defined under interface TCP/IP2:

- Technostring 1: Terminator: 13 (CR), port number 1500 and Mode: passive.
5.3.3 Configuration Watchdog

Under Hardware - General you will find the Watchdog tab. The watchdog telegram with the following parameters is defined there:

Port number: 40001, Protocol: TCP/IP, Mode: passive, Format: Binary big endian
5.3.4 Online view

Trend curves show the currently transmitted analog and digital values.
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 and the license number.

Contact

Head office

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Phone: +49 911 97282-0
Fax: +49 911 97282-33
Email: iba@iba-ag.com
Contact: Harald Opel

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