WAGO I/O SYSTEM 750

Modular I/O-System
Linux Fieldbus Coupler
750-860

Manual

Technical description, installation and configuration

Version 1.1.1
Every conceivable measure has been taken to ensure the correctness and completeness of this documentation. However, as errors can never be fully excluded we would appreciate any information or ideas at any time.

E-Mail: documentation@wago.com

We wish to point out that the software and hardware terms as well as the trademarks of companies used and/or mentioned in the present manual are generally trademark or patent protected.

This product includes software developed by the University of California, Berkley and its contributors.
## TABLE OF CONTENTS

1 Important Comments.......................................................................................... 8
  1.1 Legal Principles .......................................................................................... 8
  1.1.1 Copyright ............................................................................................. 8
  1.1.2 Personnel Qualification ....................................................................... 8
  1.1.3 Intended Use ........................................................................................ 9
  1.2 Symbols ....................................................................................................... 9
  1.3 Font Conventions ..................................................................................... 10
  1.4 Number Notation .................................................................................... 10
  1.5 Safety Notes ............................................................................................. 11
  1.6 Scope ........................................................................................................ 8
  1.7 Abbreviation ............................................................................................. 8

2 The WAGO-I/O-SYSTEM 750 ..................................................................... 9
  2.1 System Description ................................................................................... 9
  2.2 Technical Data ........................................................................................ 10
  2.3 Manufacturing Number .......................................................................... 16
  2.4 Component Update ................................................................................. 17
  2.5 Storage, Assembly and Transport .......................................................... 17
  2.6 Mechanical Setup ................................................................................... 18
    2.6.1 Installation Position ........................................................................... 18
    2.6.2 Total Expansion ................................................................................. 18
    2.6.3 Assembly onto Carrier Rail ............................................................... 19
      2.6.3.1 Carrier Rail Properties .................................................................. 19
      2.6.3.2 WAGO DIN Rail .......................................................................... 20
    2.6.4 Spacing .............................................................................................. 20
    2.6.5 Plugging and Removal of the Components ....................................... 21
    2.6.6 Assembly Sequence........................................................................... 22
    2.6.7 Internal Bus/Data Contacts ................................................................ 23
    2.6.8 Power Contacts ................................................................................ 24
    2.6.9 Wire Connection ................................................................................ 25
  2.7 Power Supply ........................................................................................... 26
    2.7.1 Isolation ............................................................................................. 26
    2.7.2 System Supply ................................................................................... 27
      2.7.2.1 Connection .................................................................................... 27
      2.7.2.2 Alignment ..................................................................................... 28
    2.7.3 Field Supply ....................................................................................... 30
      2.7.3.1 Connection .................................................................................... 30
      2.7.3.2 Fusing ............................................................................................ 31
    2.7.4 Supplementary Power Supply Regulations ....................................... 34
    2.7.5 Supply Example ................................................................................. 35
    2.7.6 Power Supply Unit ............................................................................. 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.8 Grounding</td>
<td>37</td>
</tr>
<tr>
<td>2.8.1 Grounding the DIN Rail</td>
<td>37</td>
</tr>
<tr>
<td>2.8.1.1 Framework Assembly</td>
<td>37</td>
</tr>
<tr>
<td>2.8.1.2 Insulated Assembly</td>
<td>37</td>
</tr>
<tr>
<td>2.8.2 Grounding Function</td>
<td>38</td>
</tr>
<tr>
<td>2.8.3 Grounding Protection</td>
<td>39</td>
</tr>
<tr>
<td>2.9 Shielding (Screening)</td>
<td>40</td>
</tr>
<tr>
<td>2.9.1 General</td>
<td>40</td>
</tr>
<tr>
<td>2.9.2 Bus Conductors</td>
<td>40</td>
</tr>
<tr>
<td>2.9.3 Signal Conductors</td>
<td>40</td>
</tr>
<tr>
<td>2.9.4 WAGO Shield (Screen) Connecting System</td>
<td>41</td>
</tr>
<tr>
<td>2.10 Assembly Guidelines/Standards</td>
<td>41</td>
</tr>
<tr>
<td>3 Linux Fieldbus Coupler 750-860</td>
<td>42</td>
</tr>
<tr>
<td>3.1 View</td>
<td>43</td>
</tr>
<tr>
<td>3.2 Power Supply</td>
<td>44</td>
</tr>
<tr>
<td>3.3 Fieldbus Connection via the Ethernet Interface</td>
<td>45</td>
</tr>
<tr>
<td>3.4 Display Elements</td>
<td>46</td>
</tr>
<tr>
<td>3.5 Configuration Interface</td>
<td>47</td>
</tr>
<tr>
<td>3.6 Selector Switch</td>
<td>48</td>
</tr>
<tr>
<td>3.7 Hardware Address (MAC ID)</td>
<td>49</td>
</tr>
<tr>
<td>3.8 Memory Map</td>
<td>49</td>
</tr>
<tr>
<td>3.9 Linux System Start</td>
<td>50</td>
</tr>
<tr>
<td>3.10 Error Indication (IO LED)</td>
<td>51</td>
</tr>
<tr>
<td>3.11 Technical Data</td>
<td>53</td>
</tr>
<tr>
<td>4 Board Support Package (BSP)</td>
<td>55</td>
</tr>
<tr>
<td>4.1 Scope of delivery</td>
<td>55</td>
</tr>
<tr>
<td>4.2 File System (Root File System)</td>
<td>58</td>
</tr>
<tr>
<td>4.3 The Console of the Linux Fieldbus Coupler</td>
<td>62</td>
</tr>
<tr>
<td>4.3.1 Shells</td>
<td>63</td>
</tr>
<tr>
<td>4.3.2 Busybox</td>
<td>64</td>
</tr>
<tr>
<td>4.3.3 User Administration (Tinylogin)</td>
<td>67</td>
</tr>
<tr>
<td>4.3.4 Terminal Program (getty)</td>
<td>68</td>
</tr>
<tr>
<td>4.3.5 Setting the Hardware and System Clock</td>
<td>68</td>
</tr>
<tr>
<td>4.4 Web-Based Management (WBM)</td>
<td>71</td>
</tr>
<tr>
<td>4.5 Ethernet Interface Services</td>
<td>81</td>
</tr>
<tr>
<td>4.5.1 Telnet Server</td>
<td>81</td>
</tr>
<tr>
<td>4.5.2 FTP Server (ftpd)</td>
<td>82</td>
</tr>
<tr>
<td>4.5.3 FTP Client</td>
<td>84</td>
</tr>
<tr>
<td>4.5.4 SNTP Client (msntp)</td>
<td>85</td>
</tr>
<tr>
<td>4.5.5 HTTP Server (BOA)</td>
<td>87</td>
</tr>
<tr>
<td>4.5.6 Netflash</td>
<td>89</td>
</tr>
<tr>
<td>4.6 Ethernet Interface Services</td>
<td>91</td>
</tr>
<tr>
<td>4.6.1 Temporary Assignment of the IP Address (ifconfig)</td>
<td>91</td>
</tr>
<tr>
<td>4.6.2 Permanent Assignment of the IP Address (wagoset)</td>
<td>92</td>
</tr>
<tr>
<td>4.6.3 Dynamic IP Address Assignment during Runtime</td>
<td>93</td>
</tr>
<tr>
<td>4.6.4 Checking the Network Connection</td>
<td>95</td>
</tr>
</tbody>
</table>
5 Step-by-Step Guide to your own Linux Application.................................96
  5.1 Installation of the Development Toolchain...........................................96
  5.2 Unpack the Source Code .......................................................................97
  5.3 Linux Kernel Compilation .....................................................................98
  5.4 Setup of the Development Environment ..............................................100
    5.4.1 PC Terminal Program Configuration (serial) .................................101
    5.4.2 PC Terminal Program Configuration (Ethernet) .............................103
  5.5 Setting the IP address of the Linux Fieldbus Coupler .........................104
  5.6 Create your own User Space Application ............................................105
  5.7 Transferring the User Space Application to the Linux Coupler ..............107
  5.8 Running the new Program ....................................................................108
  5.9 Automatic start of new Programs (initab/startwago) ............................108
  5.10 File System and Linux Kernel Update .................................................110
    5.10.1 TFTP Server Setup .......................................................................111
    5.10.2 TFTP Client Setup .......................................................................112
  5.11 NFS Server Setup ................................................................................116
  5.12 Debugging of User Space Programs ....................................................117
    5.12.1 Description of the GDB ................................................................117
    5.12.2 Preparation for Using the Debugger ..............................................118
    5.12.3 Creating a Testable User Space Application ..................................119
    5.12.4 Creating an Example Application for Debugging .........................121
    5.12.5 Debugging via the GDB Console ..................................................122
    5.12.6 Debugging via the Graphical Interface DDD .................................125

6 Process Image ............................................................................................127
  6.1 Basic Structure .....................................................................................127
  6.2 Example for an Input Process Image ....................................................128
  6.3 Example for an Output Process Image ..................................................129
  6.4 Application Example for a Mixed Node: .............................................130
  6.5 Internal Bus Driver (kbus.ko) ...............................................................131
  6.6 Process Image Analysis via the /proc Directory ..................................135

7 I/O Modules ..............................................................................................136
  7.1 Overview .............................................................................................136
    7.1.1 Digital Input Modules ......................................................................136
    7.1.2 Digital Output Modules ...................................................................138
    7.1.3 Analog Input Modules .....................................................................139
    7.1.4 Analog Output Modules ..................................................................141
    7.1.5 Special Modules ............................................................................142
    7.1.6 System Modules ............................................................................144
  7.2 Process Data Architecture for MODBUS/TCP ......................................145
    7.2.1 Digital Input Modules ......................................................................145
    7.2.2 Digital Output Modules ...................................................................147
    7.2.3 Analog Input Modules .....................................................................151
    7.2.4 Analog Output Modules ..................................................................152
    7.2.5 Specialty Modules ...........................................................................153
    7.2.6 System Modules ............................................................................165
Table of Contents

8 Application Examples ................................................................. 166
  8.1 LED Indication Example ....................................................... 167
  8.2 Example kbusdemo .............................................................. 171
  8.3 Example of a Modbus TCP Server ......................................... 175
  8.4 Example Program for a CGI Application .............................. 177
  8.5 Example of Reading and Writing from the NVRAM ............... 183

9 Bootloader (U-Boot) ................................................................. 186
  9.1 Boot Loader Toolchain .......................................................... 186
  9.2 TFTP Server Setup .............................................................. 187
  9.3 Unpack the Source Code ..................................................... 188
  9.4 Compile Boot Loader ........................................................... 188
  9.5 Environment Variables ......................................................... 189

10 Fieldbus Communication ....................................................... 191
  10.1 ETHERNET ................................................................. 191
  10.1.1 General ................................................................. 191
  10.1.2 Network Architecture – Principles and Regulations .......... 192
  10.1.2.1 Transmission Media .................................................. 193
  10.1.2.2 Network Topologies ................................................... 195
  10.1.2.3 Coupler Modules ....................................................... 198
  10.1.2.4 Transmission Mode .................................................. 198
  10.1.2.5 Important Terms ....................................................... 200
  10.1.3 Network Communication ............................................... 202
  10.1.3.1 Protocol layer model .................................................. 202
  10.1.3.2 Communication Protocols .......................................... 204
  10.1.3.3 Channel access method ............................................. 205
  10.1.3.4 Administration and Diagnosis Protocols ..................... 212
  10.1.3.5 Application Protocols ............................................... 217
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>Foreword</td>
<td>218</td>
</tr>
<tr>
<td>11.2</td>
<td>Protective Measures</td>
<td>218</td>
</tr>
<tr>
<td>11.3</td>
<td>Classification Meeting CENELEC and IEC</td>
<td>218</td>
</tr>
<tr>
<td>11.3.1</td>
<td>Divisions</td>
<td>218</td>
</tr>
<tr>
<td>11.3.2</td>
<td>Explosion Protection Group</td>
<td>220</td>
</tr>
<tr>
<td>11.3.3</td>
<td>Unit Categories</td>
<td>221</td>
</tr>
<tr>
<td>11.3.4</td>
<td>Temperature Classes</td>
<td>222</td>
</tr>
<tr>
<td>11.3.5</td>
<td>Types of Ignition Protection</td>
<td>223</td>
</tr>
<tr>
<td>11.4</td>
<td>Classifications Meeting the NEC 500</td>
<td>224</td>
</tr>
<tr>
<td>11.4.1</td>
<td>Divisions</td>
<td>224</td>
</tr>
<tr>
<td>11.4.2</td>
<td>Explosion Protection Groups</td>
<td>224</td>
</tr>
<tr>
<td>11.4.3</td>
<td>Temperature Classes</td>
<td>225</td>
</tr>
<tr>
<td>11.5</td>
<td>Identification</td>
<td>226</td>
</tr>
<tr>
<td>11.5.1</td>
<td>For Europe</td>
<td>226</td>
</tr>
<tr>
<td>11.5.2</td>
<td>For America</td>
<td>227</td>
</tr>
<tr>
<td>11.6</td>
<td>Installation Regulations</td>
<td>228</td>
</tr>
<tr>
<td>12</td>
<td>Glossary</td>
<td>230</td>
</tr>
<tr>
<td>13</td>
<td>Literature List</td>
<td>241</td>
</tr>
<tr>
<td>14</td>
<td>Index</td>
<td>242</td>
</tr>
</tbody>
</table>
1 Important Comments

To ensure fast installation and start-up of the units described in this manual, we strongly recommend that the following information and explanations are carefully read and abided by.

1.1 Legal Principles

1.1.1 Copyright

This manual is copyrighted, together with all figures and illustrations contained therein. Any use of this manual which infringes the copyright provisions stipulated herein, is not permitted. Reproduction, translation and electronic and photo-technical archiving and amendments require the written consent of WAGO Kontakttechnik GmbH & Co. KG. Non-observance will entail the right of claims for damages.

WAGO Kontakttechnik GmbH & Co. KG reserves the right to perform modifications allowed by technical progress. In case of grant of a patent or legal protection of utility patents all rights are reserved by WAGO Kontakttechnik GmbH & Co. KG. Products of other manufacturers are always named without referring to patent rights. The existence of such rights can therefore not be ruled out.

1.1.2 Personnel Qualification

The product described in this manual is intended to be used only by professional electronic technicians or by persons who were instructed by them as well as by qualified persons with adequate PLC training. They all must have a good knowledge of

• the applicable standards,
• C programming and
• of the Linux operating system.

WAGO Kontakttechnik GmbH & Co. KG declines all liability resulting from improper action and damage to WAGO products and third party products due to non-observance of the information contained in this manual.
1.1.3 Intended Use

For each individual application, the components supplied are to work with a dedicated hardware and software configuration. Modifications are only permitted within the framework of the possibilities documented in the manuals. All other changes to the hardware and/or software and the non-conforming use of the components entail the exclusion of liability on part of WAGO Kontakttechnik GmbH & Co. KG.

Please direct any requirements pertaining to a modified and/or new hardware or software configuration directly to WAGO Kontakttechnik GmbH & Co. KG.

1.2 Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>Danger</td>
</tr>
<tr>
<td>✔️</td>
<td>Always abide by this information to protect persons from injury.</td>
</tr>
<tr>
<td>!</td>
<td>Warning</td>
</tr>
<tr>
<td>✔️</td>
<td>Always abide by this information to prevent damage to the device.</td>
</tr>
<tr>
<td>!</td>
<td>Attention</td>
</tr>
<tr>
<td>✔️</td>
<td>Marginal conditions must always be observed to ensure smooth operation.</td>
</tr>
<tr>
<td>🔥</td>
<td>ESD (Electrostatic Discharge)</td>
</tr>
<tr>
<td>✔️</td>
<td>Warning of damage to the components by electrostatic discharge. Observe the precautionary measure for handling components at risk.</td>
</tr>
<tr>
<td>⬅️</td>
<td>Note</td>
</tr>
<tr>
<td>✔️</td>
<td>Routines or advice for efficient use of the device and software optimization.</td>
</tr>
<tr>
<td>☑️</td>
<td>More information</td>
</tr>
<tr>
<td>✔️</td>
<td>References on additional literature, manuals, data sheets and INTERNET pages</td>
</tr>
</tbody>
</table>
1.3 Font Conventions

*Italic* 
Names of path and files are marked italic  
i.e.: *C:\programs\WAGO-IO-CHECK*

*Italic* 
Menu items are marked as bold italic  
i.e.: *Save*

\ 
A backslash between two names marks a sequence of menu items  
i.e.: *File\New*

**END** 
Press buttons are marked as bold with small capitals  
i.e.: **ENTER**

< > 
Keys are marked bold within angle brackets  
i.e.: <F5>

**Courier**  
Program code is printed with the font **Courier**.  
i.e.: **END_VAR**

1.4 Number Notation

<table>
<thead>
<tr>
<th>Number Code</th>
<th>Example</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>100</td>
<td>normal notation</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>0x64</td>
<td>C notation</td>
</tr>
<tr>
<td>Binary</td>
<td>'100'</td>
<td>Within <code>,</code></td>
</tr>
<tr>
<td></td>
<td>'0110.0100'</td>
<td>Nibble separated with dots</td>
</tr>
</tbody>
</table>
1.5 Safety Notes

Attention
Switch off the system prior to working on bus modules!

In the event of deformed contacts, the module in question is to be replaced, as its functionality can no longer be ensured on a long-term basis.

The components are not resistant against materials having seeping and insulating properties. Belonging to this group of materials is: e.g. aerosols, silicones, triglycerides (found in some hand creams).

If it cannot be ruled out that these materials appear in the component environment, then additional measures are to be taken:
- installation of the components into an appropriate enclosure
- handling of the components only with clean tools and materials.

Attention
Cleaning of soiled contacts may only be done with ethyl alcohol and leather cloths. Thereby, the ESD information is to be regarded.

Do not use any contact spray. The spray may impair the functioning of the contact area.

The WAGO-I/O-SYSTEM 750 and its components are an open system. It must only be assembled in housings, cabinets or in electrical operation rooms. Access must only be given via a key or tool to authorized qualified personnel.

The relevant valid and applicable standards and guidelines concerning the installation of switch boxes are to be observed.

ESD (Electrostatic Discharge)
The modules are equipped with electronic components that may be destroyed by electrostatic discharge. When handling the modules, ensure that the environment (persons, workplace and packing) is well grounded. Avoid touching conductive components, e.g. gold contacts.
1.6 Scope

This manual describes the Linux fieldbus coupler, item no. 750-860, WAGO-I/O-SYSTEM 750

<table>
<thead>
<tr>
<th>Item.-No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-860</td>
<td>Linux fieldbus coupler</td>
</tr>
</tbody>
</table>

1.7 Abbreviation

| AI      | Analog Input               |
| AO      | Analog Output              |
| DI      | Digital Input              |
| DO      | Digital Output             |
| I/O     | Input/Output               |
| ID      | Identifier                 |
| Linux-FBK | Programmable Linux fieldbus coupler |
2 The WAGO-I/O-SYSTEM 750

2.1 System Description

The WAGO-I/O-SYSTEM 750 is a modular, field bus independent I/O system. It is comprised of a field bus coupler/controller (1) and connected field bus modules (2) for any type of signal. Together, these make up the field bus node. The end module (3) completes the node.

Couplers/controllers for field bus systems such as PROFIBUS, INTERBUS, ETHERNET TCP/IP, CAN (CANopen, DeviceNet, CAL), MODBUS, LON and others are available.

The coupler/controller contains the field bus interface, electronics and a power supply terminal. The field bus interface forms the physical interface to the relevant field bus. The electronics process the data of the bus modules and make it available for the field bus communication. The 24 V system supply and the 24 V field supply are fed in via the integrated power supply terminal. The field bus coupler communicates via the relevant field bus. The programmable field bus controller (PFC) enables the implementation of additional PLC functions. Programming is done with the WAGO-I/O-PRO in accordance with IEC 61131-3.

Bus modules for diverse digital and analog I/O functions as well as special functions can be connected to the coupler/controller. The communication between the coupler/controller and the bus modules is carried out via an internal bus.

The WAGO-I/O-SYSTEM 750 has a clear port level with LEDs for status indication, insertable mini WSB markers and pullout group marker carriers. The 3-wire technology supplemented by a ground wire connection allows for direct sensor/actuator wiring.
## 2.2 Technical Data

### Mechanic

<table>
<thead>
<tr>
<th>Material</th>
<th>Polycarbonate, Polyamide 6.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions W x H* x L</td>
<td></td>
</tr>
<tr>
<td>* from upper edge of DIN 35 rail</td>
<td></td>
</tr>
<tr>
<td>- Coupler/Controller (Standard)</td>
<td>51 mm x 65 mm x 100 mm</td>
</tr>
<tr>
<td>- Coupler/Controller (ECO)</td>
<td>50 mm x 65 mm x 100 mm</td>
</tr>
<tr>
<td>- Coupler/Controller (FireWire)</td>
<td>62 mm x 65 mm x 100 mm</td>
</tr>
<tr>
<td>- I/O module, single</td>
<td>12 mm x 64 mm x 100 mm</td>
</tr>
<tr>
<td>- I/O module, double</td>
<td>24 mm x 64 mm x 100 mm</td>
</tr>
<tr>
<td>- I/O module, fourfold</td>
<td>48 mm x 64 mm x 100 mm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Installation</th>
<th>on DIN 35 with interlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular by</td>
<td>double featherkey-dovetail</td>
</tr>
<tr>
<td>Mounting position</td>
<td>any position</td>
</tr>
<tr>
<td>Marking</td>
<td>standard marking label type group marking label 8 x 47 mm</td>
</tr>
</tbody>
</table>

### Connection

<table>
<thead>
<tr>
<th>Connection type</th>
<th>CAGE CLAMP®</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wire range</td>
<td>0.08 mm² ... 2.5 mm², AWG 28-14</td>
</tr>
<tr>
<td>Stripped length</td>
<td>8 … 9 mm, 9 … 10 mm for components with pluggable wiring (753-xxx)</td>
</tr>
</tbody>
</table>

### Contacts

| Power jumpers contacts | blade/spring contact self-cleaning |
| Current via power contacts max | 10 A |
| Voltage drop at I max | < 1 V/64 modules |
| Data contacts | slide contact, hard gold plated 1.5 µm, self-cleaning |

### Climatic environmental conditions

<p>| Operating temperature | 0 °C ... 55 °C, -20 °C ... +60 °C for components with extended temperature range (750-xxx/025-xxx) |
| Storage temperature   | -20 °C ... +85 °C |
| Relative humidity     | 5 % ... 95 % without condensation |
| Resistance to harmful substances | acc. to IEC 60068-2-42 and IEC 60068-2-43 |
| Maximum pollutant concentration at relative humidity &lt; 75% | SO₂ ≤ 25 ppm H₂S ≤ 10 ppm |
| Special conditions | Ensure that additional measures for components are taken, which are used in an environment involving: dust, caustic vapors or gases ionization radiation |</p>
<table>
<thead>
<tr>
<th>Safe electrical isolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air and creepage distance</td>
</tr>
<tr>
<td>Degree of pollution acc. To IEC 61131-2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Degree of protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree of protection</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Electromagnetic compatibility</th>
</tr>
</thead>
</table>

### Immunity to interference for industrial areas acc. to EN 61000-6-2 (2001)

<table>
<thead>
<tr>
<th>Test specification</th>
<th>Test values</th>
<th>Strength class</th>
<th>Evaluation criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN 61000-4-2 ESD</td>
<td>4 kV/8 kV (contact/air)</td>
<td>2/3</td>
<td>B</td>
</tr>
<tr>
<td>EN 61000-4-3 electromagnetic fields</td>
<td>10 V/m 80 MHz ... 1 GHz</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>EN 61000-4-4 burst</td>
<td>1 kV/2 kV (data/supply)</td>
<td>2/3</td>
<td>B</td>
</tr>
<tr>
<td>EN 61000-4-5 surge</td>
<td>Data: +/- (line/line)</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>1 kV (line/earth)</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>DC supply: 0.5 kV (line/line)</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>0.5 kV (line/earth)</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>AC supply: 1 kV (line/line)</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>2 kV (line/earth)</td>
<td>3</td>
<td>B</td>
</tr>
<tr>
<td>EN 61000-4-6 RF disturbances</td>
<td>10 V/m 80 % AM (0.15 ... 80 MHz)</td>
<td>3</td>
<td>A</td>
</tr>
</tbody>
</table>

### Emission of interference for industrial areas acc. to EN 61000-6-4 (2001)

<table>
<thead>
<tr>
<th>Test specification</th>
<th>Limit values/[QP]*</th>
<th>Frequency range</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN 55011 (AC supply, conducted)</td>
<td>79 dB (μV)</td>
<td>150 kHz ... 500 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>73 dB (μV)</td>
<td>500 kHz ... 30 MHz</td>
<td></td>
</tr>
<tr>
<td>EN 55011 (radiated)</td>
<td>40 dB (μV/m)</td>
<td>30 MHz ... 230 MHz</td>
<td>10 m</td>
</tr>
<tr>
<td></td>
<td>47 dB (μV/m)</td>
<td>230 MHz ... 1 GHz</td>
<td>10 m</td>
</tr>
</tbody>
</table>

### Emission of interference for residential areas acc. to EN 61000-6-3 (2001)

<table>
<thead>
<tr>
<th>Test specification</th>
<th>Limit values/[QP]*</th>
<th>Frequency range</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN 55022 (AC supply, conducted)</td>
<td>66 ... 56 dB (μV)</td>
<td>150 kHz ... 500 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>56 dB (μV)</td>
<td>500 kHz ... 5 MHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60 dB (μV)</td>
<td>5 MHz ... 30 MHz</td>
<td></td>
</tr>
<tr>
<td>EN 55022 (DC supply/data, conducted)</td>
<td>40 ... 30 dB (μA)</td>
<td>150 kHz ... 500 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>30 dB (μA)</td>
<td>500 kHz ... 30 MHz</td>
<td></td>
</tr>
<tr>
<td>EN 55022 (radiated)</td>
<td>30 dB (μV/m)</td>
<td>30 MHz ... 230 MHz</td>
<td>10 m</td>
</tr>
<tr>
<td></td>
<td>37 dB (μV/m)</td>
<td>230 MHz ... 1 GHz</td>
<td>10 m</td>
</tr>
<tr>
<td>Test specification</td>
<td>Frequency range</td>
<td>Limit value</td>
<td></td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
<td>-------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>IEC 60068-2-6 vibration</td>
<td>5 Hz ≤ f &lt; 9 Hz</td>
<td>1.75 mm amplitude (permanent) 3.5 mm amplitude (short term)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9 Hz ≤ f &lt; 150 Hz</td>
<td>0.5 g (permanent) 1 g (short term)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note on vibration test:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a) Frequency change: max. 1 octave/minute</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b) Vibration direction: 3 axes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEC 60068-2-27 shock</td>
<td></td>
<td>15 g</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note on shock test:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a) Type of shock: half sine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b) Shock duration: 11 ms</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>c) Shock direction: 3x in positive and 3x in negative direction for each of the three mutually perpendicular axes of the test specimen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IEC 60068-2-32 free fall</td>
<td></td>
<td>1 m</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(module in original packing)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) QP: Quasi Peak

**Note:**
If the technical data of components differ from the values described here, the technical data shown in the manuals of the respective components shall be valid.
For Products of the WAGO-I/O-SYSTEM 750 with ship specific approvals supplementary guidelines are valid:

### Electromagnetic Compatibility

#### Immunity to interference acc. to Germanischer Lloyd (2003)

<table>
<thead>
<tr>
<th>Test specification</th>
<th>Test values</th>
<th>Strength class</th>
<th>Evaluation criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC 61000-4-2 ESD</td>
<td>6 kV/8 kV (contact/air)</td>
<td>3/3</td>
<td>B</td>
</tr>
<tr>
<td>IEC 61000-4-3 electromagnetic fields</td>
<td>10 V/m 80 MHz ... 2 GHz</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>IEC 61000-4-4 burst</td>
<td>1 kV/2 kV (data/supply)</td>
<td>2/3</td>
<td>A</td>
</tr>
<tr>
<td>IEC 61000-4-5 surge</td>
<td><strong>AC/DC Supply:</strong> 0.5 kV (line/line)</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>1 kV (line/earth)</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>IEC 61000-4-6 RF disturbances</td>
<td>10 V/m 80 % AM (0.15 ... 80 MHz)</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>Type test AF disturbances (harmonic waves)</td>
<td>3 V, 2 W</td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>Type test high voltage</td>
<td>755 V DC 1500 V AC</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### Emission of interference acc. to Germanischer Lloyd (2003)

<table>
<thead>
<tr>
<th>Test specification</th>
<th>Limit values</th>
<th>Frequency range</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type test (EMC1, conducted) allows for ship bridge control applications</td>
<td>96 ... 50 dB (μV)</td>
<td>10 kHz ... 150 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60 ... 50 dB (μV)</td>
<td>150 kHz ... 350 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50 dB (μV)</td>
<td>350 kHz ... 30 MHz</td>
<td></td>
</tr>
<tr>
<td>Type test (EMC1, radiated) allows for ship bridge control applications</td>
<td>80 ... 52 dB (μV/m)</td>
<td>150 kHz ... 300 kHz</td>
<td>3 m</td>
</tr>
<tr>
<td></td>
<td>52 ... 34 dB (μV/m)</td>
<td>300 kHz ... 30 MHz</td>
<td>3 m</td>
</tr>
<tr>
<td></td>
<td>54 dB (μV/m)</td>
<td>30 MHz ... 2 GHz</td>
<td>3 m</td>
</tr>
<tr>
<td>except:</td>
<td>24 dB (μV/m)</td>
<td>156 MHz ... 165 MHz</td>
<td>3 m</td>
</tr>
</tbody>
</table>

#### Mechanical strength acc. to Germanischer Lloyd (2003)

<table>
<thead>
<tr>
<th>Test specification</th>
<th>Frequency range</th>
<th>Limit value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC 60068-2-6 vibration (category A – D)</td>
<td>2 Hz ≤ f &lt; 25 Hz</td>
<td>± 1.6 mm amplitude (permanent)</td>
</tr>
<tr>
<td></td>
<td>25 Hz ≤ f &lt; 100 Hz</td>
<td>4 g (permanent)</td>
</tr>
</tbody>
</table>

Note on vibration test:
- a) Frequency change: max. 1 octave/minute
- b) Vibration direction: 3 axes
The WAGO-I/O-SYSTEM 750
Technical Data

### Range of application

<table>
<thead>
<tr>
<th></th>
<th>Required specification emission of interference</th>
<th>Required specification immunity to interference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial areas</td>
<td>EN 61000-6-4 (2001)</td>
<td>EN 61000-6-2 (2001)</td>
</tr>
<tr>
<td>Residential areas</td>
<td>EN 61000-6-3 (2001)*</td>
<td>EN 61000-6-1 (2001)</td>
</tr>
</tbody>
</table>

*) The system meets the requirements on emission of interference in residential areas with the field bus coupler/controller for:

- ETHERNET 750-342/-841/-842/-860
- LonWorks 750-319/-819
- CANopen 750-337/-837
- DeviceNet 750-306/-806
- MODBUS 750-312/-314/-315/-316
  750-812/-814/-815/-816

With a special permit, the system can also be implemented with other field bus couplers/controllers in residential areas (housing, commercial and business areas, small-scale enterprises). The special permit can be obtained from an authority or inspection office. In Germany, the Federal Office for Post and Telecommunications and its branch offices issues the permit.

It is possible to use other field bus couplers/controllers under certain boundary conditions. Please contact WAGO Kontakttechnik GmbH & Co. KG.

### Maximum power dissipation of the components

<table>
<thead>
<tr>
<th>Component</th>
<th>Power Dissipation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus modules</td>
<td>0.8 W / bus terminal (total power dissipation, system/field)</td>
</tr>
<tr>
<td>Field bus coupler/controller</td>
<td>2.0 W / coupler/controller</td>
</tr>
</tbody>
</table>

### Warning

The power dissipation of all installed components must not exceed the maximum conductible power of the housing (cabinet).

When dimensioning the housing, care is to be taken that even under high external temperatures, the temperature inside the housing does not exceed the permissible ambient temperature of 55 °C.
Dimensions

Side view Dimensions in mm

Fig. 2-2: Dimensions

Note:
The illustration shows a standard coupler. For detailed dimensions, please refer to the technical data of the respective coupler/controller.
2.3 Manufacturing Number

The manufacturing number indicates the delivery status directly after production. This number is part of the lateral marking on the component. In addition, starting from calendar week 43/2000 the manufacturing number is also printed on the cover of the configuration and programming interface of the field bus coupler or controller.

![Diagram of Manufacturing Number]

Fig. 2-3: Example: Manufacturing Number of a PROFIBUS field bus coupler 750-333

The manufacturing number consists of the production week and year, the software version (if available), the hardware version of the component, the firmware loader (if available) and further internal information for WAGO Kontakttechnik GmbH & Co. KG.
2.4 Component Update

For the case of an Update of one component, the lateral marking on each component contains a prepared matrix.

This matrix makes columns available for altogether three updates to the entry of the current update data, like production order number (NO; starting from calendar week 13/2004), update date (DS), software version (SW), hardware version (HW) and the firmware loader version (FWL, if available).

**Update Matrix**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Production Order Number</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Datestamp</td>
<td>DS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software index</td>
<td>SW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware index</td>
<td>HW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firmware loader index</td>
<td>FWL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If the update of a component took place, the current version data are registered into the columns of the matrix.

Additionally with the update of a field bus coupler or controller also the cover of the configuration and programming interface of the coupler or controller is printed on with the current manufacturing and production order number.

The original manufacturing data on the housing of the component remain thereby.

2.5 Storage, Assembly and Transport

Wherever possible, the components are to be stored in their original packaging. Likewise, the original packaging provides optimal protection during transport.

When assembling or repacking the components, the contacts must not be soiled or damaged. The components must be stored and transported in appropriate containers/packaging. Thereby, the ESD information is to be regarded.

Statically shielded transport bags with metal coatings are to be used for the transport of open components for which soiling with amine, amide and silicone has been ruled out, e.g. 3M 1900E.
2.6 Mechanical Setup

2.6.1 Installation Position

Along with horizontal and vertical installation, all other installation positions are allowed.

Attention
In the case of vertical assembly, an end stop has to be mounted as an additional safeguard against slipping.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>249-116</td>
<td>End stop for DIN 35 rail, 6 mm wide</td>
</tr>
<tr>
<td>249-117</td>
<td>End stop for DIN 35 rail, 10 mm wide</td>
</tr>
</tbody>
</table>

2.6.2 Total Expansion

The length of the module assembly (including one end module of 12 mm width) that can be connected to the coupler/controller is 780 mm. When assembled, the I/O modules have a maximum length of 768 mm.

Examples:

- 64 I/O modules of 12 mm width can be connected to one coupler/controller.
- 32 I/O modules of 24 mm width can be connected to one coupler/controller.

Exception:

The number of connected I/O modules also depends on which type of coupler/controller is used. For example, the maximum number of I/O modules that can be connected to a PROFIBUS coupler/controller is 63 without end module. The maximum total expansion of a node is calculated as follows:

Warning
The maximum total length of a node without coupler/controller must not exceed 780 mm. Furthermore, restrictions made on certain types of couplers/controllers must be observed (e.g. for PROFIBUS).
2.6.3 Assembly onto Carrier Rail

2.6.3.1 Carrier Rail Properties

All system components can be snapped directly onto a carrier rail in accordance with the European standard EN 50022 (DIN 35).

Warning

WAGO Kontakttechnik GmbH & Co. KG supplies standardized carrier rails that are optimal for use with the I/O system. If other carrier rails are used, then a technical inspection and approval of the rail by WAGO Kontakttechnik GmbH & Co. KG should take place.

Carrier rails have different mechanical and electrical properties. For the optimal system setup on a carrier rail, certain guidelines must be observed:

- The material must be non-corrosive.
- Most components have a contact to the carrier rail to ground electromagnetic disturbances. In order to avoid corrosion, this tin-plated carrier rail contact must not form a galvanic cell with the material of the carrier rail which generates a differential voltage above 0.5 V (saline solution of 0.3% at 20°C).
- The carrier rail must optimally support the EMC measures integrated into the system and the shielding of the bus module connections.
- A sufficiently stable carrier rail should be selected and, if necessary, several mounting points (every 20 cm) should be used in order to prevent bending and twisting (torsion).
- The geometry of the carrier rail must not be altered in order to secure the safe hold of the components. In particular, when shortening or mounting the carrier rail, it must not be crushed or bent.
- The base of the I/O components extends into the profile of the carrier rail. For carrier rails with a height of 7.5 mm, mounting points are to be riveted under the node in the carrier rail (slotted head captive screws or blind rivets).
2.6.3.2 WAGO DIN Rail

WAGO carrier rails meet the electrical and mechanical requirements.

<table>
<thead>
<tr>
<th>Item Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>210-113 /-112</td>
<td>35 x 7.5; 1 mm; steel yellow chromated; slotted/unslotted</td>
</tr>
<tr>
<td>210-114 /-197</td>
<td>35 x 15; 1.5 mm; steel yellow chromated; slotted/unslotted</td>
</tr>
<tr>
<td>210-118</td>
<td>35 x 15; 2.3 mm; steel yellow chromated; unslotted</td>
</tr>
<tr>
<td>210-198</td>
<td>35 x 15; 2.3 mm; copper; unslotted</td>
</tr>
<tr>
<td>210-196</td>
<td>35 x 7.5; 1 mm; aluminum; unslotted</td>
</tr>
</tbody>
</table>

2.6.4 Spacing

The spacing between adjacent components, cable conduits, casing and frame sides must be maintained for the complete field bus node.

The spacing creates room for heat transfer, installation or wiring. The spacing to cable conduits also prevents conducted electromagnetic interferences from influencing the operation.
2.6.5 Plugging and Removal of the Components

**Warning**
Before work is done on the components, the voltage supply must be turned off.

In order to safeguard the coupler/controller from jamming, it should be fixed onto the carrier rail with the locking disc. To do so, push on the upper groove of the locking disc using a screwdriver.

To pull out the field bus coupler/controller, release the locking disc by pressing on the bottom groove with a screwdriver and then pulling the orange colored unlocking lug.

![Fig. 2-5: Coupler/Controller and unlocking lug](image)

It is also possible to release an individual I/O module from the unit by pulling an unlocking lug.

![Fig. 2-6: removing bus terminal](image)

**Danger**
Ensure that an interruption of the PE will not result in a condition which could endanger a person or equipment!
For planning the ring feeding of the ground wire, please see chapter 2.6.3.
2.6.6 Assembly Sequence

All system components can be snapped directly on a carrier rail in accordance with the European standard EN 50022 (DIN 35).

The reliable positioning and connection is made using a tongue and groove system. Due to the automatic locking, the individual components are securely seated on the rail after installing.

Starting with the coupler/controller, the bus modules are assembled adjacent to each other according to the project planning. Errors in the planning of the node in terms of the potential groups (connection via the power contacts) are recognized, as the bus modules with power contacts (male contacts) cannot be linked to bus modules with fewer power contacts.

---

**Attention**
Always link the bus modules with the coupler/controller, and always plug from above.

---

**Warning**
Never plug bus modules from the direction of the end terminal. A ground wire power contact, which is inserted into a terminal without contacts, e.g. a 4-channel digital input module, has a decreased air and creepage distance to the neighboring contact in the example DI4.

Always terminate the field bus node with an end module (750-600).
2.6.7 Internal Bus/Data Contacts

Communication between the coupler/controller and the bus modules as well as the system supply of the bus modules is carried out via the internal bus. It is comprised of 6 data contacts, which are available as self-cleaning gold spring contacts.

![Data contacts](image)

**Warning**

Do not touch the gold spring contacts on the I/O modules in order to avoid soiling or scratching!

**ESD (Electrostatic Discharge)**

The modules are equipped with electronic components that may be destroyed by electrostatic discharge. When handling the modules, ensure that the environment (persons, workplace and packing) is well grounded. Avoid touching conductive components, e.g. data contacts.
2.6.8 Power Contacts

Self-cleaning power contacts, are situated on the side of the components which further conduct the supply voltage for the field side. These contacts come as touchproof spring contacts on the right side of the coupler/controller and the bus module. As fitting counterparts the module has male contacts on the left side.

**Danger**
The power contacts are sharp-edged. Handle the module carefully to prevent injury.

**Attention**
Please take into consideration that some bus modules have no or only a few power jumper contacts. The design of some modules does not allow them to be physically assembled in rows, as the grooves for the male contacts are closed at the top.

<table>
<thead>
<tr>
<th>Power jumper contacts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blade</td>
</tr>
<tr>
<td>Spring</td>
</tr>
</tbody>
</table>

Fig. 2-8: Example for the arrangement of power contacts

**Recommendation**
With the WAGO ProServe® Software smartDESIGNER, the structure of a field bus node can be configured. The configuration can be tested via the integrated accuracy check.
2.6.9 Wire Connection

All components have CAGE CLAMP® connections.

The WAGO CAGE CLAMP® connection is appropriate for solid, stranded and finely stranded conductors. Each clamping unit accommodates one conductor.

The operating tool is inserted into the opening above the connection. This opens the CAGE CLAMP®. Subsequently the conductor can be inserted into the opening. After removing the operating tool, the conductor is safely clamped.

More than one conductor per connection is not permissible. If several conductors have to be made at one connection point, then they should be made away from the connection point using WAGO Terminal Blocks. The terminal blocks may be jumpered together and a single wire brought back to the I/O module connection point.

**Attention**

If it is unavoidable to jointly connect 2 conductors, then a ferrule must be used to join the wires together.

Ferrule:

<table>
<thead>
<tr>
<th>Length</th>
<th>8 mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal cross section max.</td>
<td>1 mm² for 2 conductors with 0.5 mm² each</td>
</tr>
<tr>
<td>WAGO Product</td>
<td>216-103</td>
</tr>
<tr>
<td></td>
<td>or products with comparable properties</td>
</tr>
</tbody>
</table>
2.7 Power Supply

2.7.1 Isolation

Within the field bus node, there are three electrically isolated potentials.

- Operational voltage for the field bus interface.
- Electronics of the couplers/controllers and the bus modules (internal bus).
- All bus modules have an electrical isolation between the electronics (internal bus, logic) and the field electronics. Some digital and analog input modules have each channel electrically isolated, please see catalog.

**Attention**

The ground wire connection must be present in each group. In order that all protective conductor functions are maintained under all circumstances, it is recommended that a ground wire be connected at the beginning and end of a potential group. (ring format, please see chapter 2.8.3). Thus, if a bus module comes loose from a composite during servicing, then the protective conductor connection is still guaranteed for all connected field devices.

When using a joint power supply unit for the 24 V system supply and the 24 V field supply, the electrical isolation between the internal bus and the field level is eliminated for the potential group.
2.7.2 System Supply

2.7.2.1 Connection

The WAGO-I/O-SYSTEM 750 requires a 24 V direct current system supply (-15% or +20%). The power supply is provided via the coupler/controller and, if necessary, in addition via the internal system supply modules (750-613). The voltage supply is reverse voltage protected.

Attention
The use of an incorrect supply voltage or frequency can cause severe damage to the component.

The direct current supplies all internal system components, e.g. coupler/controller electronics, field bus interface and bus modules via the internal bus (5 V system voltage). The 5 V system voltage is electrically connected to the 24 V system supply.

Fig. 2-11: System Supply

Fig. 2-12: System Voltage
Attention
Resetting the system by switching on and off the system supply, must take place simultaneously for all supply modules (coupler/controller and 750-613).

2.7.2.2 Alignment

Recommendation
A stable network supply cannot be taken for granted always and everywhere. Therefore, regulated power supply units should be used in order to guarantee the quality of the supply voltage.

The supply capacity of the coupler/controller or the internal system supply module (750-613) can be taken from the technical data of the components.

<table>
<thead>
<tr>
<th>Internal current consumption *)</th>
<th>Current consumption via system voltage: 5 V for electronics of the bus modules and coupler/controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Residual current for bus terminals *)</td>
<td>Available current for the bus modules. Provided by the bus power supply unit. See coupler/controller and internal system supply module (750-613)</td>
</tr>
</tbody>
</table>

*) cf. catalogue W4 Volume 3, manuals or internet

Example

| Coupler 750-301: |
|------------------|-----------------|
| internal current consumption: 350 mA at 5V |
| residual current for bus modules: 1650 mA at 5V |
| sum I(5V) total: 2000 mA at 5V |

The internal current consumption is indicated in the technical data for each bus terminal. In order to determine the overall requirement, add together the values of all bus modules in the node.

Attention
If the sum of the internal current consumption exceeds the residual current for bus modules, then an internal system supply module (750-613) must be placed before the module where the permissible residual current was exceeded.
Example: A node with a PROFIBUS Coupler 750-333 consists of 20 relay modules (750-517) and 10 digital input modules (750-405).

Current consumption:
20 * 90 mA = 1800 mA
10 * 2 mA = 20 mA
Sum = 1820 mA

The coupler can provide 1650 mA for the bus modules. Consequently, an internal system supply module (750-613), e.g. in the middle of the node, should be added.

Recommendation
With the WAGO ProServe® Software smartDESIGNER, the assembly of a field bus node can be configured. The configuration can be tested via the integrated accuracy check.

The maximum input current of the 24 V system supply is 500 mA. The exact electrical consumption (I_{(24 V)}) can be determined with the following formulas:

**Coupler/Controller**

\[
I_{(5 V) \text{ total}} = \text{Sum of all the internal current consumption of the connected bus modules} + \text{internal current consumption coupler/controller}
\]

750-613

\[
I_{(5 V) \text{ total}} = \text{Sum of all the internal current consumption of the connected bus modules}
\]

Input current \( I_{(24 V)} = \frac{5 \, \text{V}}{24 \, \text{V}} \times \frac{I_{(5 V) \text{ total}}}{\eta} \)

\( \eta = 0.87 \) (at nominal load)

**Note**

If the electrical consumption of the power supply point for the 24 V-system supply exceeds 500 mA, then the cause may be an improperly aligned node or a defect.

During the test, all outputs, in particular those of the relay modules, must be active.
2.7.3 Field Supply

2.7.3.1 Connection

Sensors and actuators can be directly connected to the relevant channel of the bus module in 1/4 conductor connection technology. The bus module supplies power to the sensors and actuators. The input and output drivers of some bus modules require the field side supply voltage.

The coupler/controller provides field side power (DC 24V). In this case it is a passive power supply without protection equipment. Power supply modules are available for other potentials, e.g. AC 230 V. Likewise, with the aid of the power supply modules, various potentials can be set up. The connections are linked in pairs with a power contact.

![Field Supply (Sensor/Actuator)](image)

The supply voltage for the field side is automatically passed to the next module via the power jumper contacts when assembling the bus modules.

The current load of the power contacts must not exceed 10 A on a continual basis. The current load capacity between two connection terminals is identical to the load capacity of the connection wires.

By inserting an additional power supply module, the field supply via the power contacts is disrupted. From there a new power supply occurs which may also contain a new voltage potential.
Attention
Some bus modules have no or very few power contacts (depending on the I/O function). Due to this, the passing through of the relevant potential is disrupted. If a field supply is required for subsequent bus modules, then a power supply module must be used. Note the data sheets of the bus modules.

In the case of a node setup with different potentials, e.g. the alteration from DC 24 V to AC 230V, a spacer module should be used. The optical separation of the potentials acts as a warning to heed caution in the case of wiring and maintenance works. Thus, the results of wiring errors can be prevented.

2.7.3.2 Fusing

Internal fusing of the field supply is possible for various field voltages via an appropriate power supply module.

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-601</td>
<td>24 V DC, Supply/Fuse</td>
</tr>
<tr>
<td>750-609</td>
<td>230 V AC, Supply/Fuse</td>
</tr>
<tr>
<td>750-615</td>
<td>120 V AC, Supply/Fuse</td>
</tr>
<tr>
<td>750-610</td>
<td>24 V DC, Supply/Fuse/Diagnosis</td>
</tr>
<tr>
<td>750-611</td>
<td>230 V AC, Supply/Fuse/Diagnosis</td>
</tr>
</tbody>
</table>

Fig. 2-14: Supply module with fuse carrier (Example 750-610)
Warning
In the case of power supply modules with fuse holders, only fuses with a maximum dissipation of 1.6 W (IEC 127) must be used. For UL approved systems only use UL approved fuses.

In order to insert or change a fuse, or to switch off the voltage in succeeding bus modules, the fuse holder may be pulled out. In order to do this, use a screwdriver for example, to reach into one of the slits (one on both sides) and pull out the holder.

Fig. 2-15: Removing the fuse carrier

Lifting the cover to the side opens the fuse carrier.

Fig. 2-16: Opening the fuse carrier

Fig. 2-17: Change fuse

After changing the fuse, the fuse carrier is pushed back into its original position.
Alternatively, fusing can be done externally. The fuse modules of the WAGO series 281 and 282 are suitable for this purpose.

Fig. 2-18: Fuse modules for automotive fuses, series 282

Fig. 2-19: Fuse modules with pivotable fuse carrier, series 281

Fig. 2-20: Fuse modules, series 282
2.7.4 Supplementary Power Supply Regulations

The WAGO-I/O-SYSTEM 750 can also be used in shipbuilding or offshore and onshore areas of work (e.g. working platforms, loading plants). This is demonstrated by complying with the standards of influential classification companies such as Germanischer Lloyd and Lloyds Register.

Filter modules for 24-volt supply are required for the certified operation of the system.

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-626</td>
<td>Supply filter</td>
<td>Filter module for system supply and field supply (24 V, 0 V), i.e. for field bus coupler/controller and bus power supply (750-613)</td>
</tr>
<tr>
<td>750-624</td>
<td>Supply filter</td>
<td>Filter module for the 24 V-field supply (750-602, 750-601, 750-610)</td>
</tr>
</tbody>
</table>

Therefore, the following power supply concept must be absolutely complied with.

---

**Note**  
Another potential power terminal 750-601/602/610 must only be used behind the filter terminal 750-626 if the protective earth conductor is needed on the lower power contact or if a fuse protection is required.
2.7.5 Supply Example

**Attention**
The system supply and the field supply should be separated in order to ensure bus operation in the event of a short-circuit on the actuator side.

Fig. 2-22: Supply example
2.7.6 Power Supply Unit

The WAGO-I/O-SYSTEM 750 requires a 24 V direct current system supply with a maximum deviation of -15% or +20%.

Recommendation
A stable network supply cannot be taken for granted always and everywhere. Therefore, regulated power supply units should be used in order to guarantee the quality of the supply voltage.

A buffer (200 µF per 1 A current load) should be provided for brief voltage dips. The I/O system buffers for approx 1 ms.

The electrical requirement for the field supply is to be determined individually for each power supply point. Thereby all loads through the field devices and bus modules should be considered. The field supply as well influences the bus modules, as the inputs and outputs of some bus modules require the voltage of the field supply.

Note
The system supply and the field supply should be isolated from the power supplies in order to ensure bus operation in the event of short circuits on the actuator side.

<table>
<thead>
<tr>
<th>WAGO products Item No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>787-903</td>
<td>Primary switched-mode, DC 24 V, 5 A wide input voltage range AC 85-264 V PFC (Power Factor Correction)</td>
</tr>
<tr>
<td>787-904</td>
<td>Primary switched-mode, DC 24 V, 10 A wide input voltage range AC 85-264 V PFC (Power Factor Correction)</td>
</tr>
<tr>
<td>787-912</td>
<td>Primary switched-mode, DC 24 V, 2 A wide input voltage range AC 85-264 V</td>
</tr>
<tr>
<td>288-809</td>
<td>Rail-mounted modules with universal mounting carrier AC 115 V / DC 24 V; 0.5 A</td>
</tr>
<tr>
<td>288-810</td>
<td>AC 230 V / DC 24 V; 0.5 A</td>
</tr>
<tr>
<td>288-812</td>
<td>AC 230 V / DC 24 V; 2 A</td>
</tr>
<tr>
<td>288-813</td>
<td>AC 115 V / DC 24 V; 2 A</td>
</tr>
</tbody>
</table>
2.8 Grounding

2.8.1 Grounding the DIN Rail

2.8.1.1 Framework Assembly

When setting up the framework, the carrier rail must be screwed together with the electrically conducting cabinet or housing frame. The framework or the housing must be grounded. The electronic connection is established via the screw. Thus, the carrier rail is grounded.

**Attention**
Care must be taken to ensure the flawless electrical connection between the carrier rail and the frame or housing in order to guarantee sufficient grounding.

2.8.1.2 Insulated Assembly

Insulated assembly has been achieved when there is constructively no direct conduction connection between the cabinet frame or machine parts and the carrier rail. Here the earth must be set up via an electrical conductor.

The connected grounding conductor should have a cross section of at least 4 mm².

**Recommendation**
The optimal insulated setup is a metallic assembly plate with grounding connection with an electrical conductive link with the carrier rail.

The separate grounding of the carrier rail can be easily set up with the aid of the WAGO ground wire terminals.

<table>
<thead>
<tr>
<th>Item No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>283-609</td>
<td>1-conductor ground (earth) terminal block make an automatic contact to the carrier rail; conductor cross section: 0.2 - 16 mm²</td>
</tr>
</tbody>
</table>

**Note:** Also order the end and intermediate plate (283-320).
2.8.2 Grounding Function

The grounding function increases the resistance against disturbances from electro-magnetic interferences. Some components in the I/O system have a carrier rail contact that dissipates electro-magnetic disturbances to the carrier rail.

![Carrier rail contact diagram](image)

**Fig. 2-23: Carrier rail contact**

---

**Attention**

Care must be taken to ensure the direct electrical connection between the carrier rail contact and the carrier rail.

The carrier rail must be grounded.

For information on carrier rail properties, please see chapter 2.6.3.2.
2.8.3 Grounding Protection

For the field side, the ground wire is connected to the lowest connection terminals of the power supply module. The ground connection is then connected to the next module via the Power Jumper Contact (PJC). If the bus module has the lower power jumper contact, then the ground wire connection of the field devices can be directly connected to the lower connection terminals of the bus module.

---

**Attention**

Should the ground conductor connection of the power jumper contacts within the node become disrupted, e.g. due to a 4-channel bus terminal, the ground connection will need to be re-established.

The ring feeding of the grounding potential will increase the system safety. When one bus module is removed from the group, the grounding connection will remain intact.

The ring feeding method has the grounding conductor connected to the beginning and end of each potential group.

---

**Attention**

The regulations relating to the place of assembly as well as the national regulations for maintenance and inspection of the grounding protection must be observed.
2.9 Shielding (Screening)

2.9.1 General

The shielding of the data and signal conductors reduces electromagnetic interferences thereby increasing the signal quality. Measurement errors, data transmission errors and even disturbances caused by overvoltage can be avoided.

**Attention**

Constant shielding is absolutely required in order to ensure the technical specifications in terms of the measurement accuracy.

The data and signal conductors should be separated from all high-voltage cables.

The cable shield should be potential. With this, incoming disturbances can be easily diverted.

The shielding should be placed over the entrance of the cabinet or housing in order to already repel disturbances at the entrance.

2.9.2 Bus Conductors

The shielding of the bus conductor is described in the relevant assembly guidelines and standards of the bus system.

2.9.3 Signal Conductors

Bus modules for most analog signals along with many of the interface bus modules include a connection for the shield.

**Note**

For a better shield performance, the shield should have previously been placed over a large area. The WAGO shield connection system is suggested for such an application.

This suggestion is especially applicable if the equipment can have even current or high impulse formed currents running through (for example initiated by atmospheric discharge).
2.9.4 WAGO Shield (Screen) Connecting System

The WAGO Shield Connecting system includes a shield clamping saddle, a collection of rails and a variety of mounting feet. Together these allow many different possibilities. See catalog W4 volume 3 chapter 10.

Fig. 2-25: WAGO Shield (Screen) Connecting System

Fig. 2-26: Application of the WAGO Shield (Screen) Connecting System

2.10 Assembly Guidelines/Standards

- DIN 60204, Electrical equipping of machines
- DIN EN 50178 Equipping of high-voltage systems with electronic components (replacement for VDE 0160)
- EN 60439 Low voltage – switch box combinations
3 Linux Fieldbus Coupler 750-860

The Linux fieldbus coupler 750-860 is a freely programmable fieldbus coupler that comprises an ARM microcontroller and a uClinux operating system. uClinux (microcontroller Linux) is based on kernel version 2.6.

The Linux fieldbus coupler has a 10/100Base Ethernet interface, a serial interface as well as an internal bus interface for the connection of up to 64 Series 750 and 753 WAGO I/O modules.

The coupler incorporates the 32-bit processor ARM7TDMI without MMU with a RISC architecture and a frequency of 44 MHz. No MMU means no memory virtualization. This allows userspace applications to access the entire memory.

The memory consists of 16 MB RAM, 32 kB NOVRAM and 4 MB Flash. 2.5 MB of the flash memory hold the JFF2 file system. Additionally, the Linux fieldbus coupler has a real time clock.

The Linux fieldbus coupler is delivered with a base image and the fixed IP address 192.168.1.3. For many user space applications the base image is the right platform. It allows fast access via the Linux console, which is provided by both the Ethernet (telnet) and the serial interface (getty).

The base image is part of the Bord Support Package (BSP) for the Linux fieldbus coupler. You can download the BSP with the item number 759-914 from the WAGO homepage free of charge. The BSP contains the source code for the Kernel 2.6 and the boot loader (U-Boot), the ARM elf toolchain for x86 systems as well as many user space applications. Sample user space applications are an excellent starting point for the development of your own applications.

The process data of the I/O modules are provided by the internal bus driver, which is a dynamic loadable module, in separate process images for inputs and outputs. To access these process images from user space applications, you can use an internal bus API which, in the simplest case, only needs the functions kbusOpen(), kbusUpdate() and kbusClose().

The CGI interface of the Web Server (BOA) is another special feature of the base image. You can create powerful Web applications provided with total access to the process data via the Common Gateway Interface (CGI). Additionally, you have access to an FTP server, a DHCP client, a DNS client, a SNTP client, and to NFS support.
3.1 View

The Linux fieldbus coupler comprises

- Power supply for the system supply as well as power jumper contacts for the field side supply via I/O modules
- Fieldbus interface with fieldbus connection
- LEDs for status indication, i.e. operational status, fieldbus communication, operating voltages, error message and diagnostics
- Configuration Interface
- Selector switch
- Electronics for communication with the I/O modules (internal bus) and the fieldbus interface
3.2 Power Supply

The power is supplied through terminals with CAGE CLAMP® connection. The power supply provides power to both the system and the field side.

The integrated power supply provides the required power to the electronics and to the I/O modules.

An electrically isolated power supply is provided to the fieldbus interface.
3.3 Fieldbus Connection via the Ethernet Interface

An RJ45 connector is used to connect the coupler to the fieldbus. The RJ45 socket of the Linux fieldbus coupler is wired in accordance with the 100BaseTX requirements. It is mandatory to use a twisted pair cable of category 5 as a connecting cable. S-UTP (Screened Unshielded Twisted Pair) cables and STP (Shielded Twisted Pair) cables with a maximum segment length of 100 m (328 feet) can be used.

The connection point is lowered in such a way that after a connector is inserted, installation in an 80 mm high switchgear cabinet is possible.

The electrical isolation between the fieldbus system and the electronics is provided by DC/DC converters and optocouplers in the fieldbus interface.

<table>
<thead>
<tr>
<th>Contact</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TD +</td>
<td>Transmit +</td>
</tr>
<tr>
<td>2</td>
<td>TD -</td>
<td>Transmit -</td>
</tr>
<tr>
<td>3</td>
<td>RD +</td>
<td>Receive +</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Not assigned</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Not assigned</td>
</tr>
<tr>
<td>6</td>
<td>RD -</td>
<td>Receive -</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Not assigned</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>Not assigned</td>
</tr>
</tbody>
</table>

Fig. 3-3: Fieldbus connection and pin assignment, RJ45 plug

**Attention**
The bus connection is only approved for use in LAN networks, not for the connection of telecommunication lines!
3.4 Display Elements

The operational status of the Linux fieldbus coupler or the fieldbus node is indicated via LEDs. Optical fibers transmit the LED signals to the upper side of the enclosure. Some of these signals are colored red and green.

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINK</td>
<td>Green</td>
<td>Physical connection to the Ethernet network established.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>Fieldbus node has no physical connection to the Ethernet network.</td>
</tr>
<tr>
<td>STATUS</td>
<td>Green/Red</td>
<td>Function can be defined by the user.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>SERVICE</td>
<td>Green/Red</td>
<td>Function can be defined by the user.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>TxD/RxD</td>
<td>Green</td>
<td>Data is being exchanged via Ethernet.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>No data is being exchanged via Ethernet.</td>
</tr>
<tr>
<td>IO</td>
<td>Red</td>
<td>Internal bus error or fieldbus coupler error.</td>
</tr>
<tr>
<td></td>
<td>Green</td>
<td>Linux fieldbus coupler operational, no internal errors.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>Linux fieldbus coupler not operational.</td>
</tr>
<tr>
<td>USR</td>
<td>Green/Red</td>
<td>Function can be defined by the user.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Green</td>
<td>Status of operating voltage 24 V supply voltage of the Linux coupler.</td>
</tr>
<tr>
<td>B</td>
<td>Green</td>
<td>Status of operating voltage Power jumper contacts (field side supply)</td>
</tr>
</tbody>
</table>

More information
The evaluation of the LED signals is described in chapters 3.10 and 8.1.
3.5 Configuration Interface

The configuration interface is located behind the cover flap. It is used for the communication with WAGO-I/O-CHECK and for the firmware download. It can be addressed under Linux via "ttyS0" device.

Fig. 3-5: Configuration interface

The communication cable 750-920 is connected to the four pole header.

**Caution**
The communication cable 750-920 must not be connected or removed when energized, i.e. the coupler/controller must be voltage free!

<table>
<thead>
<tr>
<th>Contact</th>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TxD</td>
<td>Transmit</td>
</tr>
<tr>
<td>2</td>
<td>RxD</td>
<td>Receive</td>
</tr>
<tr>
<td>3</td>
<td>VSS</td>
<td>Voltage Source</td>
</tr>
<tr>
<td>4</td>
<td>GND</td>
<td>Ground</td>
</tr>
</tbody>
</table>

**Attention**
The level adjustment from RS232 (+15 V/-15 V) to TTL (0V/5V) is done in the plug of the communication cable (750-920). There is no hardware handshake (CTS/RTS).
3.6 Selector Switch

The selector switch is located behind the cover flap.

![Selector switch](image)

Fig. 3-6: Selector switch
g01xx10d

The selector switch is a pressure switch/slide switch with three different positions and one pressure switch function. You can control the behavior of the serial interface or the boot process via the selector switch.

The number of operations of the selector switch complies with EN61131T2.

<table>
<thead>
<tr>
<th>Selector switch</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper position</td>
<td>The Linux coupler is started with a Linux console on the serial interface (38400-8-N-1- NoFlowControl).</td>
</tr>
<tr>
<td>Middle position</td>
<td>WAGO program IO update is supported.</td>
</tr>
<tr>
<td>Lower position</td>
<td>Linux coupler starts Bootstrap Loader (BSL).</td>
</tr>
<tr>
<td>Pressed down (e.g. using a screwdriver)</td>
<td>Hardware reset</td>
</tr>
</tbody>
</table>

The boot loader analyzes the position of the selector switch, changes the Linux kernel start parameters and adds or deletes the `wago_console=yes` parameter. The serial interface driver and the boot code in the `/etc/startwago` file evaluate the start parameter. In the `/etc/startwago` file it will be decided whether or not to start the `/bin/io-check` or the `/bin/getty` program.
3.7 Hardware Address (MAC ID)

Each WAGO Linux fieldbus coupler is provided from the factory with a unique and internationally unambiguous physical address, also referred to as MAC ID (Media Access Control Identity). This is located on the rear of the coupler and on a self-adhesive tear-off label on the coupler side. The address has a fixed length of 6 bytes (48 bits) and contains the address type, the manufacturer’s ID, and the serial number.

3.8 Memory Map

The memory map in the flash memory looks as follows:

<table>
<thead>
<tr>
<th>Flash memory address</th>
<th>Description</th>
<th>Size (approx.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x003FFFFF</td>
<td>Start parameter (U-Boot environment)</td>
<td>64 kB</td>
</tr>
<tr>
<td>0x003F0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x003EFFFFF</td>
<td>Linux kernel</td>
<td>1.1 MB</td>
</tr>
<tr>
<td>0x002D0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x002CF0FF</td>
<td>File system (JFFS2)</td>
<td>2.6 MB</td>
</tr>
<tr>
<td>0x00040000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0003FFFF</td>
<td>Boot Loader (U-Boot)</td>
<td>192 kB</td>
</tr>
<tr>
<td>0x00010000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0000FFFF</td>
<td>BSL and FWL</td>
<td>64 kB</td>
</tr>
<tr>
<td>0x00000000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Attention**

Do not delete the contents of the BSL/FWL memory addresses. Otherwise the Linux fieldbus coupler will become inoperative!

Hence, 0x00290000 bytes (approx. 2.69 Mbyte) are available to the file system. After a successful start of the Linux kernel, the memory map in the RAM looks as follows:

<table>
<thead>
<tr>
<th>RAM address</th>
<th>Name</th>
<th>Size (approx.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000400000</td>
<td>Linux kernel</td>
<td>3.8 MB</td>
</tr>
<tr>
<td>0x000040000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0003FFFFF</td>
<td>Reserved</td>
<td>256 kB</td>
</tr>
<tr>
<td>0x000010000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Linux kernel manages the entire free RAM memory.
3.9 Linux System Start

The initialization follows after switching on the supply voltage, or after a hardware reset. The Linux fieldbus coupler determines the I/O module configuration and creates a process image.

First, the firmware loader (FWL) is started via the WAGO bootstrap loader (BSL) which in turn will start the boot loader (U-Boot) for Linux. The BSL cannot be deleted. The FWL may only be exchanged by WAGO after prior consultation.

The boot loader (U-Boot) for Linux is under GPL and is therefore also in the source code on the WAGO Linux distribution and can be changed or replaced. This is only necessary if you intend to change the boot behavior of the Linux kernel. To abort the boot loader boot process and to change to the U-Boot environment you have to enter the password geheim (on the serial console) when the following is output:

I2C-EEPROM detected.
I2C-RTC detected.
In: serial
Out: serial
Err: serial
Booting in 3 seconds - enter abort password to prevent this

The boot loader switches to its shell environment. If, after three seconds, no password has been entered, the boot loader (U-Boot) loads the Linux kernel from the flash memory, unpacks it into the RAM starting at address 0x0040000 and then starts it with the specified boot parameters. The kernel startup scripts will then perform the following actions:

- Set timer IRQ
- Initialize RAM memory
- Embed the JFFS2 file system
- Start serial driver
- Embed LED driver
- Create partitions in the flash memory
- Load I2C driver: embed real-time clock chip/embed EEPROM
Load network driver, IP configuration
Start init process and any other required process
Load internal bus driver

During start-up, the "I/O" LED (red) flashes fast. After approximately 20 seconds, the "I/O" LED lights up green and the Linux fieldbus coupler is operational and the internal bus driver is loaded. If an error occurs during start-up, it is indicated via the blinking "I/O" LED (red).

3.10 Error Indication (IO LED)

If errors occur during the operation of the Linux fieldbus coupler, detailed error messages are indicated via the blink code of the "I/O" LED. The "IO" LED flashes red.

A blink code consists of three blink sequences with a pause between each sequence:

- The error display starts with the first blink sequence (approx. 10 Hz).
- After a break, the second blink sequence starts (approx. 1 Hz). The number of light pulses indicates the Error Code.
- After another break, the third blink sequence starts (approx. 1 Hz). The number of light pulses indicates the Error Argument.
The blink code is repeated cyclically until the error is corrected and/or the Linux fieldbus coupler is rebooted. The following table lists the types of errors with the associated blink codes:

<table>
<thead>
<tr>
<th>Error argument</th>
<th>Error description</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault Code 1: Hardware and Configuration Fault</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Overflow of the internal buffer memory for the inline code.</td>
<td>Turn off the supply voltage of the Linux fieldbus coupler, reduce the number of modules and turn on the supply voltage again. If the error remains, replace the coupler.</td>
</tr>
<tr>
<td>4</td>
<td>Error when writing to EEPROM</td>
<td>Switch off the supply voltage of the Linux fieldbus coupler, replace the coupler and switch on again.</td>
</tr>
<tr>
<td>5</td>
<td>Fault when reading the serial EEPROM</td>
<td>Switch off the supply voltage of the Linux fieldbus coupler, replace the coupler and switch on again.</td>
</tr>
<tr>
<td>6</td>
<td>The I/O module configuration after an internal bus reset differs from the one after the last I/O-IPC start-up.</td>
<td>Restart the Linux fieldbus coupler by turning the power supply off and on again.</td>
</tr>
<tr>
<td>14</td>
<td>Maximum number of gateway modules or mailbox modules exceeded</td>
<td>Reduce the number of corresponding modules to a valid number.</td>
</tr>
<tr>
<td>Error code 4: Physical error of the internal bus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n*</td>
<td>Interruption of the internal bus behind the nth I/O module with process data.</td>
<td>Turn off the supply voltage of the Linux fieldbus coupler, replace the (n+1)th I/O module with process data and turn on the supply voltage again.</td>
</tr>
<tr>
<td>Error code 6: Node configuration error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Error during process image generation</td>
<td>Reduce the number of I/O modules on the node.</td>
</tr>
</tbody>
</table>
### 3.11 Technical Data

<table>
<thead>
<tr>
<th>System data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of I/O modules</td>
<td>Limited by ETHERNET specification</td>
</tr>
<tr>
<td>Transmission medium</td>
<td>Twisted Pair S-UTP 100 Ω CAT 5</td>
</tr>
<tr>
<td>Buscoupler connection</td>
<td>RJ45</td>
</tr>
<tr>
<td>Max. length of fieldbus segment</td>
<td>100 m between hub station and Linux_fieldbus coupler;</td>
</tr>
<tr>
<td></td>
<td>max. length of network limited by ETHERNET specification</td>
</tr>
<tr>
<td>Baud rate</td>
<td>10/100 Mbit/s</td>
</tr>
<tr>
<td>Protocols</td>
<td>MODBUS/TCP, HTTP, BootP, DHCP, DNS, SNTP, FTP, Telnet, NFS</td>
</tr>
<tr>
<td>CPU</td>
<td>32-Bit-Risc ARM7/44 MHz</td>
</tr>
<tr>
<td>RAM 1</td>
<td>16 MByte</td>
</tr>
<tr>
<td>RAM 2 (NOVRAM)</td>
<td>32 kbytes</td>
</tr>
<tr>
<td>FLASH memory</td>
<td>4 MByte (2.69 MByte for file system)</td>
</tr>
<tr>
<td>EEPROM</td>
<td>4 kbytes</td>
</tr>
</tbody>
</table>

#### Standards and guidelines (see chapter 2.2)

- EMC CE Immunity to interference: Acc. to EN 61000-6-2 (1999)
- Approvals (see chapter 2.2)
  - cULus (UL508)
  - cULus (UL1604) Class I Div2 ABCD T4A
  - DEMKO II 3 G EEx nA II T4
  - Conformity marking

#### Accessories

- Miniature WSB Quick marking system
### Technical Data

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of I/O modules</td>
<td>64</td>
</tr>
<tr>
<td>Number of I/O modules with bus extension</td>
<td>250</td>
</tr>
<tr>
<td>Per node</td>
<td></td>
</tr>
<tr>
<td>Digital signals</td>
<td>Max. 2040 (inputs and outputs)</td>
</tr>
<tr>
<td>Analog signals</td>
<td>Max. 1020 (inputs and outputs)</td>
</tr>
<tr>
<td>Configuration possibility</td>
<td>Via PC</td>
</tr>
<tr>
<td>Max. number of socket connections</td>
<td>1 HTTP, 1 FTP, 8 Telnet, 8 NFS</td>
</tr>
<tr>
<td>Powerfail RTC buffer</td>
<td>Min. 6 days</td>
</tr>
<tr>
<td>Voltage supply</td>
<td>DC 24 V (-15 %/+ 20 %)</td>
</tr>
<tr>
<td>Input current max</td>
<td>500 mA at 24 V</td>
</tr>
<tr>
<td>Efficiency of the power supply</td>
<td>87 %</td>
</tr>
<tr>
<td>Internal current consumption</td>
<td>300 mA bei 5 V</td>
</tr>
<tr>
<td>Total current for I/O modules</td>
<td>1700 mA bei 5 V</td>
</tr>
<tr>
<td>Isolation</td>
<td>500 V system/supply</td>
</tr>
<tr>
<td>Voltage via power jumper contacts</td>
<td>DC 24 V (-15 %/+ 20 %)</td>
</tr>
<tr>
<td>Current via power jumper contacts max</td>
<td>DC 10 A</td>
</tr>
<tr>
<td>Dimensions (mm) W x H x L</td>
<td>51 x 65* x 100 (*from upper edge of DIN 35 rail)</td>
</tr>
<tr>
<td>Weight</td>
<td>Approx. 180 g</td>
</tr>
</tbody>
</table>
4 Board Support Package (BSP)

The BSP for the Linux fieldbus coupler is based on the uClinux distribution and contains all source code files and tools required for programming the coupler.

The BSP (item no. 759-914) can be downloaded from the WAGO web site or ordered on CD-ROM.

The BSP contains the source code for the Kernel 2.6, the boot loader (U-Boot), the ARM toolchain for x86 systems as well as many userspace applications. The user space applications are an excellent starting point for the development of your own user space applications.

The WAGO base image for the Linux fieldbus coupler is part of the BSP. The following chapters relate to this WAGO base image.

4.1 Scope of delivery

The CD ROM of the BSP includes the following directories:

- images Hex files including the images for kernel, file system, U-Boot
- IOUpdates Update program for firmware updates via serial interface
- linux
  - sources Source codes of the entire WAGO uClinux distribution
  - toolchain Assembler, compiler, linker, ..., for uClinux distribution
- uboot
  - sources Source codes of the U-Boot boot loader
  - toolchain Assembler, compiler, linker, ..., for the U-Boot boot loader

The following gives a detailed description of the directories.
The `/images` directory

The `/images` directory contains the following files:

- **jffs2.img**
  Copying the image is started from uboot using the TFTP protocol or netflash.

- **linux.flashme**
  Image of the operating system.
  Copying the image is started from uboot using the TFTP protocol or netflash.

- **ub.bin**
  Image of the boot loader (uboot).
  Copying the image is started from uboot using the TFTP protocol or netflash.

- **image.hex**
  Complete base image consisting of the file system, operating system, boot loader and firmware loader.
  Transfer to the Linux fieldbus coupler is done using the WAGO-IO-Update windows program.

You will find further information on uboot in chapter 9 and on netflash in chapter 4.5.6

The `/IO-Update` directory

You will find here the setup to install the WAGO-IO-Update windows program.
The `/linux` directory

This directory contains the 2.6 Linux source code as well as the ARM toolchain for x86 systems

- **sources**
  The source code for the 2.6 Linux kernel.
  The source code is unpacked by the `install.sh` script into the home directory of the current user.

- **toolchain**
  ARM toolchain for x86 systems.
  The `install.sh` script requires super user privileges.

The `/uboot` directory

This directory contains the source code as well as the toolchain of the boot loader (uboot) for x86 systems.

Changes to the boot loader should only be made in some rare exeptions.

- **sources**
  The uboot source code.
  The source code is unpacked by the `install.sh` script into the home directory of the current user.

- **toolchain**
  Uboot toolchain.

  The `install.sh` script requires super user privileges.
4.2 File System (Root File System)

The directory structure of the Linux coupler file system looks as follows:

```
sbin  User programs for super user
proc   Kernel proc system
home  Free
usr   Free
var   Various
  log  Log files
  lock Lock files
www  Web pages on the web server, write rights for everybody
   cgi-bin  CGI files of the web server
   tmp  Temporary files, write rights for everybody
 run  pid files
 tmp   Link to /var/tmp
 srv   Link to /var/www
 nfs  Mount point for an NFS directory
 mnt  Free mount point
 lib  Libraries/modules
  modules … Kernel modules /iocheck or /kbus (internal bus)
 etc  Configuration files
  boa  Configuration files of the Web server
  dhcp  Configuration files of the DHCP client
 wagoconsole  Configuration files of the serial interface
 dev  Device entries
  bin  User programs for user
```

The following gives a detailed description of the directories /proc, /etc and /bin.
The `/proc` directory

The entries in the `/proc` directory provide information on the status of the Linux fieldbus coupler. You can find entries for every started process as well as from the drivers or the kernel. The entries can be read and copied like a text file. However, the entries are not in the flash memory but in the RAM. For example, the entry

`cat meminfo`

provides information on the used system memory.

The following is a list (not complete) that shows the most important entries:

- `loadavg`
  Information on processor utilization
- `meminfo`
  Information on the used RAM memory
- `modules`
  Overview of the embedded kernel modules
- `devices`
  Used device drivers
- `interrupts`
  Used interrupts with counter
- `cmdline`
  Command line when starting the Linux kernel
- `kmsg`
  Kernel messages since system start
- `driver/kbus`
  Information on the internal bus driver and its process image
- `mtd`
  Partitioning of the flash memory
- `stat`
  CPU status
- `cpuinfo`
  Information on the processor type
The /etc directory

The configuration files of the Linux system are stored in the /etc directory. The boot process is kept simple and requires fewer files than a commercially available distribution for a PC. The files are described according to the boot sequence:

- **rc**
  - Starts `startwago` once during system start
  - Sets the Host name of the Linux console
  - Links the /proc directory to the file system
  - Starts `inetd` (for Telnet, FTP)
- **inittab**
  - Starts and monitors `getty`, if process stops, it retries to start `getty` (up to 5 times)
- **startwago**
  - Starts internal bus driver
  - Starts IO-Check driver
  - Starts the Web server (BOA)
  - Changes the link to `getty`, depending on the entry in the Linux kernel command line
- **boa**
  Configuration of the Web server
- **fstab**
  Configuration of the file system
- **motd**
  Start logo of the console
- **resolv.conf**
  Contains the entries of the name servers
- **passwd**
  User password file
- **services**
  The port numbers are assigned to their corresponding services
- **group**
  Configuration of the known user groups
- **TZ**
  Time zone setting for all consoles
The /bin directory

Files in the /bin directory can be run from any directory like shell commands. This directory contains all programs that can generally be accessed. In order to save memory resources in the file system, different programs are used that are especially adapted to embedded systems.

The following programs are available (condition upon delivery):

- busybox
  Combines the common Linux console programs for system administration
- tinylogin
  Combines the common user administration programs
- boa
  Compact and safe web server
- mkfs.jffs2
  Used to create the JFFS2 file system in the flash memory
- telnetd
  Telnet server
- inetd
  Inet server for the connection with Telnet and FTP
- ftpd
  Ftp server
- msntp
  SNTP client to query an NTP time server
- leds
  Example for controlling the LEDs of the Linux fieldbus coupler (see chapter 8)
- wagoset
  Example for setting/reading the Ethernet configuration in the higher flash memory area (see chapter 4.6.2)
- kbusdemo
  Demo program for reading or writing process data to the IO modules (see chapter 8)
• gdbserver
  Server program for debugging in user space (see chapter 5.12)

• netflash
  Program for copying a complete area (block) of the flash memory via TFTP (see chapter 4.5.6)

All the other files in the /bin directory are links that refer to the busybox or tinylogin program.

### 4.3 The Console of the Linux Fieldbus Coupler

The console of the Linux fieldbus coupler provides the basic functions for administration, such as starting a program or executing scripts, for example.

You can access the console via both the serial interface and the Ethernet interface.

Before you can use the console, you need to enter a user name and a password:

![Fig. 4-1: Login of the console](image)

Home
The following table shows the different users accounts that are stored by default (conditions may change upon delivery):

<table>
<thead>
<tr>
<th>User</th>
<th>Password</th>
<th>Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>wago</td>
<td>Super user</td>
</tr>
<tr>
<td>admin</td>
<td>wago</td>
<td>Super user</td>
</tr>
<tr>
<td>user</td>
<td>user</td>
<td>User has rights for HTML files</td>
</tr>
<tr>
<td>guest</td>
<td>guest</td>
<td>Read rights</td>
</tr>
</tbody>
</table>

The users "root" and "admin" are Superusers and have all rights to change the system completely.
The user "user" has a home directory in `/var/www/` and only has rights to change or to delete files there. This directory contains the files that are used for the Web server. More users or user groups to which you can assign any rights can be created using the Busybox programs adduser and addgroup (see chapter 4.3.2).

### 4.3.1 Shells

The Linux console provides the basic functions to start a program or to execute scripts. The console of the Linux fieldbus coupler is the shell that is provided by the busybox. It is a minix shell that supports more extensive script processing functions. The shell contains builtin commands, such as `cd`. Furthermore, the Shell provides the environment variables and allows to navigate the file system as well as to run programs.
4.3.2 Busybox

Busybox combines many standard Linux distribution programs in one single program in order to minimize memory requirements of the file system. The program is only started via symbolic links. Busybox evaluates the name and can in this way implement different functions in one program.

When compiling Busybox you can choose the programs to be integrated. It is thus possible to adapt the size of the program to the required functions. The Linux fieldbus coupler implements all the required functions for a minimum system.

The following call lists the implemented programs: make menuconfig:

Customize Vendor/User Settings By changing the Busybox settings during compilation they can be changed arbitrarily (see chapter 5). The list contains the following programs:

- mount
  Linking file system to the kernel
- reboot
  Rebooting the Linux fieldbus coupler
- insmod
  Linking kernel modules
- chroot
  Sets the root directory in which programs are run
- killall
  Deletes programs (killall + program name)
- pivot_root
  Changes the root file system
- ifconfig
  View/change the current Ethernet settings. Changes are not accepted.
  Please use the wagoset program.
- hostname
  Changes the local host name
- syslogd
  Creates the log file /var/log/messages
- modprobe
  Loads kernel modules
- touch
  Updates the creation date of a file/directory
- rmmod
  Unloads kernel module
- rmdir
  Deletes directory
- reset
  Deletes screen content
- mount
  Links directory
- mknod
  Creates a device entry in the /dev directory
- mkdir
  Creates a directory
- lsmod
  Shows an overview of the linked kernel modules
- klogd
  Sets settings to log kernel messages
- clear
  Clears the screen
- chown
  Changes a file/directory owner
- chgrp
  Changes the group that owns the file/directory
- tail
  Shows the last lines of a file
- ping
  PING service (Ethernet diagnostics service)
- more
  Shows text files (page by page)
- kill
  Unpacks a file in the user space
- init
  Main process generates and manages all processes
- grep
  Searches characters/strings in text files/directories
• date  
  Showing and setting the real time clock (RTC)
• msh  
  Minix shell to execute simple scripts
• cat  
  Shows text files
• vi  
  Editor to change text files
• rm  
  Deletes file
• ps  
  Shows all loaded programs in the user space
• mv  
  Moves files
• ls  
  Lists all files of the current directory
• ln  
  Creates links
• df  
  Shows file system information (free memory)
• cp  
  Copies files
• top  
  Shows the system capacity occurring through each individual process (in %)
4.3.3 User Administration (Tinylogin)

Tinylogin is also a program that can be run using different links, just like busybox, so that less file system memory is required. It provides functions for the administration of users and groups on the Linux fieldbus coupler. Tinylogin includes the following utilities:

- **passwd**
  Changes the password

- **adduser**
  Creates a new user

- **addgroup**
  Creates a new user group

- **delgroup**
  Deletes a user group

- **login**
  Starts a login session

- **getty**
  Console support

- **deluser**
  Deletes a user

- **su**
  Switches to super user mode

The table shows the users that are already stored (condition upon delivery).

<table>
<thead>
<tr>
<th>User</th>
<th>Password</th>
<th>Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>wago</td>
<td>Super user</td>
</tr>
<tr>
<td>admin</td>
<td>wago</td>
<td>Super user</td>
</tr>
<tr>
<td>user</td>
<td>user</td>
<td>User has rights for HTML files</td>
</tr>
<tr>
<td>guest</td>
<td>guest</td>
<td>Read rights</td>
</tr>
</tbody>
</table>
4.3.4 Terminal Program (getty)

The console of the Linux fieldbus coupler is based on the terminal program getty which processes the input and output via the serial interface. The terminal program getty is started if the selector switch is in the upper position during the boot process (condition upon delivery).

4.3.5 Setting the Hardware and System Clock

`hwclock` can access the integrated hardware clock via the console drive (read and write). The `date` console command sets the time of the system clock.

The difference between system clock and hardware clock is that the time will still be stored in the hardware clock after a system reboot whereas the system clock needs to be synchronized via the hardware clock after every system reboot.

Standard time of the hardware clock is "Greenwich Main Time" (GMT or UTC), the system time corresponds to the local time.

The system variable TZ determines the system time (local time).

The system variable is stored in the file `/etc/TZ` and describes the time difference between a time zone and the "Greenwich Main Time".

The system variable TZ is defined as follows:

`TZ = MEZ-1MESZ-2,M3.5.0,M10.5.0`
Options for the hardware clock

The hardware clock time can be set using the system time or the system time can be set using the hardware clock time.

The general syntax to set the hardware clock is:
```
hwclock [-r | --show] [-s | --hctosys] [-w | --systohc] [-l | --localtime] [-u | --utc]
```

The options for the hardware clock are:

- `hwclock -r` or `hwclock --show`:
  Reads the hardware clock and displays it on the console. This time is always indicated in local time, even if the hardware clock is set to Greenwich Time (UTC).

- `hwclock -s` or `hwclock --hctosys`:
  System time is set using the hardware clock time.

- `hwclock -w` or `hwclock --systohc`:
  Hardware clock is set using the current system time

- `hwclock -l` or `--localtime`:
  The hardware clock time is the local time

- `hwclock -u` or `--utc`:
  The hardware clock time is Greenwich Time

Example:
```
hwclock -r
Tue Feb 6 14:55:28 2007 0.000000 seconds
```
Options for the system clock

The general syntax to set the system clock is:
```
date [Option]... [MMDDhhmm[[CC]YY][.ss]] [+Format]
```

The options for the system clock are:

- **date -R**
  Outputs an RFC-822 compliant data string that consists of date and system time

- **date -d STRING**
  Converts a STRING (e.g. "01241040") into an RFC compliant data string and outputs it. It consists of the delivered data and time,

- **date -s STRING**
  Sets the system time that is delivered with STRING (e.g. "01241045")

- **date -r FILE**
  Shows the last modification in the file FILE

- **date -u**
  Sets the system time to Greenwich Time and displays it

Example:

```
date -d 05131045
Sun May 13 10:45:00 MESZ 2007
```
4.4 Web-Based Management (WBM)

The implemented web pages of the Linux fieldbus coupler serve as a system overview for the Ethernet settings. A navigation bar on the left side of every page allows you to access the following pages directly: Information, TCP/IP settings, Kernel Messages, Operating State Messages, I/O terminals and Reboot. The Web server starts with the following "Status information" page:

Fig. 4-2: Screen view "Status"

The status page shows the following values:

**Coupler details**

- **Order description**
  Description of the Linux fieldbus coupler
- **Order number**
  Item number of the Linux fieldbus coupler
- **Firmware version**
  Version of the implemented firmware
• FWL version
  Version of the implemented firmware loader

• Serial number
  Wago serial number

Network details

• MAC address
  MAC address of the Linux fieldbus coupler

• IP address
  The IP address which is currently used or assigned
On the TCP/IP configuration page you have the opportunity to change relevant parameters of the Ethernet configuration:

![TCP/IP configuration](image)

Fig. 4-3: Upper screen view "TCP/IP configuration"

**DNS**

- Name server 1
  IP address of the first name server in the network
- Name server 2
  IP address of the second name server in the network
- Hostname
  Name of the Linux fieldbus coupler in the dynamic network (DHCP, BootP)
- Default Search Domain
  Name of the local domain

You require the local domain name if you want the Linux coupler to access other network participants via the DNS server using short network names. For example: `ping pc1234` would not work since the complete name is `pc1234.localdomain`. If one or several "Default Search Domains" are entered, the address `pc1234` will automatically change to `pc1234.localdomain`. 
Without Ethernet

- Disable Ethernet
  Ist die Checkbox aktiviert, dann wird die Ethernetschnittstelle beim Neustart deaktiviert.

![Fig. 4-4: Lower screen view "TCP/IP configuration"](image)

Static IP Configuration

- Static IP address
  Activate the Ethernet device to use a static IP address

- IP address
  Static IP address (if activated)

- Subnet Mask
  Subnet mask (with static IP address)

- Gateway
  Default gateway (with static IP address)
Dynamic Configuration

- BootP
  Activates configuration via BootP
- DHCP
  Activates configuration via DHCP

Save Configurations

If a user with root rights enters his name and the correct password and then clicks on the SAVE CONFIGURATION button, the corresponding CGI file will save the changed values in several different places of the Linux fieldbus coupler. The new values will only be activated after a reboot of the Linux fieldbus coupler because the Ethernet interface is only initialized during a boot process.

Note

If wrong values are entered, they cannot possibly be corrected again after a reboot of the Linux fieldbus coupler since it is no longer possible to access the Linux coupler. A correction of the values is then only possible via the serial interface using the Linux console (see chapter 4.3, 4.6.1 and 4.6.2).
On the kernel message page you can see the logged kernel messages that have been entered into the `/proc/kmsg` file since the Linux coupler system start.

Fig. 4-5: Screen view "Kernel message"

Among other things, you can find information on the hardware, on the installed Linux version, the drivers or on the IO modules that are connected to the Linux fieldbus coupler.
On the EEPROM messages page you can see the messages that are stored in the EEPROM of the Linux fieldbus coupler.

Fig. 4-6: Screen view "EEPROM messages"

Messages are, for example, error messages after an interruption of the internal bus or messages due to a Linux fieldbus coupler reset.

- **Date**
  Date of the EEPROM message

- **Time**
  Time of the EEPROM message

- **Error source**
  The cause for the EEPROM message (Hex value)

- **Error value**
  Value of the EEPROM message (Hex value)

- **Description**
  Description of the EEPROM message

The EEPROM can be erased. A user with root rights enters his name and the correct password in both text fields and clicks on the **RESET EEPROM** button.

After successful authentication the EEPROM will be erased.

If the authentication was not successful, an error message will be displayed behind the text fields.
The HTML page Read Config lists all the IO modules that are connected to the Linux fieldbus coupler and their properties.

![Screen view of the IO modules that are connected to the Linux coupler](image)

**I/O Modules**

- **Color**
  The IO module color indicates the module type. For example, yellow modules are digital input modules

- **Slot number**
  Slot number indicates the position of the IO module behind the Linux fieldbus coupler

- **Article number**
  Item number of the IO module

- **Type**
  The type of I/O module is indicated once more (see Color)

- **Number of channels**
  Number of channels of the I/O module

- **Button „R“**
  Button to open the register tables
Sub-table entries (click on the "+" button on the left side to the color entry)

- First column
  Type of I/O module + offset of the I/O module channel in the process image of the Linux fieldbus coupler

- Second column (for digital I/O modules: colored background, same color as LEDs)
  Value of the channel in the memory (Hex value)
  **Light green:** channel set
  **Dark green:** channel not set
  **Gray:** the LEDs only indicate readiness for operation, no channels set
On the reboot page you can reboot the Linux fieldbus coupler by entering the name of a user with root rights and the correct password.

![Reboot Linux Fieldbus Controller](image)

The Root password for the user "root" is stored in encoded form in the `passwd` file and is initially `wago` (condition upon delivery).

**Note**
The password is not transmitted via an encrypted connection. It can thus be seen and used by any other network participant.
4.5 Ethernet Interface Services

The Linux fieldbus coupler provides several server/client services for the Ethernet interface (condition upon delivery). The following gives a short overview:

- **Telnet server**
  Network participants can connect to the Linux coupler console via a Telnet client.

- **FTP server**
  Network participants can access the Linux coupler file system via an FTP client.

- **FTP client**
  The FTP client of the Linux fieldbus coupler can exchange data with remote FTP servers.

- **Web server**
  Network participants can call information on the Linux coupler settings via a web browser.

- **SNTP client**
  The installed MSNTP client can retrieve the exact time of a remote SNTP server via the network.

- **NFS client**
  The Linux fieldbus coupler incorporates an NFS client which can link shared network drives of NFS servers.

- **TFTP client**
  The TFTP client is only implemented in the boot loader (U-Boot) and allows update of the Linux kernel, the root file system or the boot loader (U-Boot) itself. After Linux kernel start-up, the client is no longer available.

### 4.5.1 Telnet Server

The telnet login service is started (condition upon delivery). The `inetd.conf` file has a corresponding entry. The telnet daemon is started via Ethernet if requested. A new Linux console is created.

The Linux console login and the login via the serial interface run simultaneously (see chapter 4.3).
4.5.2 FTP Server (ftpd)

FTP can be used to exchange files between a PC and the Linux fieldbus coupler. Linux need not necessarily be installed on the PC since Windows also provides FTP client functions.

Other network participants can also login via FTP (condition upon delivery). Just like for the telnet support, a corresponding entry is in the inetd.conf file. The ftpd daemon is started via Ethernet if requested.

FTP File Exchange between PC and Fieldbus Coupler via Ethernet

An FTP client is required in order to exchange FTP files between a PC and the Linux fieldbus coupler via Ethernet. The Windows Explorer also offers the possibility to do that, however, we do not recommend this approach since the Explorer has many disadvantages (very slow communication, wrong interpretation of the directory structure, view is not updated which confuses the user). To use the Explorer, the following entry is made in the data record:

ftp://username:password@hostname.

Example: ftp://user:user@192.168.1.12.

The user administration is analogous to the Linux console password and user administration and is described in chapter 4 "Board Support Package".

To use FTP, run the FTP program via the DOS console when using the Windows operating system or via the Linux console when using Linux. To do so, enter ftp <hostname/IP> in the console and answer the questions about user and password (see following figure):
After logging in, you can select the help menu via `help`, which describes the available commands (examples):

- `put Datei.html`
  Writes file in the Linux fieldbus coupler

- `get Bild.gif`
  reads file from the Linux fieldbus coupler

- `cd/`
  Switches to the root directory

The most convenient way to use FTP is to use an FTP client program such as Filezilla (http://sourceforge.net/projects/filezilla), which is distributed under the GPL.

The FTP connection is established via port 21 which needs to be registered in some FTP clients.
4.5.3 FTP Client

The Linux fieldbus coupler incorporates an FTP client. This allows to
download files from the FTP server or to write files to the server via FTP. The
FTP client is installed in the /bin directory and can thus be used from every
directory by every user. Port 21 is used for the FTP protocol.

FTP Client Operation

In order to be able to use the FTP client, an FTP server and a user for the FTP
access must be installed on the remote PC. It is started by:

    ftp <IP/hostname>
Example: ftp 192.168.1.11

The FTP server asks for user and password. You can then execute commands
on the server. What commands are supported by the server can be queried by
the command:

    help.

The server shows a list with all available FTP commands.

By

    help <command>

you get a description of the queried command from the server.

Example: help cd
4.5.4 SNTP Client (msntp)

The msnftp program provides SNTP client functionality.

SNTP (Simple Network Time Protocol) is a simplified version of NTP (Network Time Protocol). SNTP allows communication with a remote SNTP server via UDP frame and port 123. If you want to connect to an SNTP server on the Internet, routing and the firewall must be configured accordingly.

SNTP transmits time as a 64 bit value, i.e. it has a resolution of approx. 0.25 ns. The accuracy of time transmission is said to be +/-10 ms on the Internet and up to +/-200 μs in local networks (Marzullo's algorithm, by Keith Marzullo, University of San Diego).

To get the time from an SNTP server, start msnftp as follows:

```bash
msnftp <Hostname/IP-Adresse>
```

Examples:

```bash
msnftp 192.168.1.11
msnftp WagoLFBC
```

The output is, for example:

```
2005 Aug 31 06:44:41.897 – 94504 +/- 34 secs
```

First, the date and the time of the time server are shown. The following number is the deviation (in this case -94.504 seconds). The last number indicates the jitter. The more the network is loaded or the farer away the time server is, the bigger the jitter becomes.

Msnftp can also directly change the time of the local system:

```bash
msnftp -r <Hostname/IP-Adresse>
```

Example: msnftp -r 192.168.1.11

The output is, for example:

```
The correction time is 94.493 +/-0.33+0.045seconds
Do you want correct the time anyway?
```

If you enter `y`, the time in the RTC chip and in the Linux kernel is changed via the internal function `settimeofday()` using the correction value of the previous query.
In order to provide for cyclic time monitoring via SNTP, MSNTP can also be started as a daemon. Additionally, MSNTP should be registered as respawn process in the `/etc/inittab` file in order to automatically activate time monitoring when starting the Linux fieldbus coupler and to restart the process automatically if the program crashes. The entry in the file `/etc/inittab` looks as follows:

```
/etc/inittab:
::respawn:/bin/msntp -r -P no -x <minutes> <IP/Hostname>
```

Example: `::respawn:/bin/msntp -r -P no -x 10 192.168.1.11`

This would mean that the time would be checked every 10 minutes. The entry

- **r** means that the process sets the time via `settimeofday()`.
- **P no** means that any time error is corrected
- **x 10** means that the process is a daemon and that it will take at least 10 minutes before it repeats its query.

Further information on the parameters can be found on the msntp main page (e.g. under: http://pigtail.net/LRP/msntp.html).
4.5.5 HTTP Server (BOA)

Program name: boa

Installation directory: /bin/boa

Source code files:
~/.linux-dist/user/boa/*.*

Function description

Transmission of HTML pages
Transmission of different picture files
Calling CGI files
CGI file "stdout" outputs are redirected in HTML frames

The BOA web server is preloaded on the Linux fieldbus coupler. BOA is a program under GPL. You can find the official web site for this project under http://www.boa.org. The web server version is 0.94.

The BOA web server will run on startup (condition upon delivery). It provides a second graphical user interface via the web-based management system. The HTML and CGI pages that is described in chapter 4.4 are already stored.
Relevant web server directories

The root directory of the web server is in the /var/www/ directory. There you can find the Index.html file. More subdirectories are possible and are addressed accordingly.

Examples for addressing a Linux fieldbus coupler with the IP address 192.168.1.12 in a network:

- 192.168.1.12
  Launches /var/www/Index.html
- 192.168.1.12/start.html
  Launches /var/www/start.html
- 192.168.1.12/picture.gif
  Shows a GIF picture /var/www/picture.gif

CGI files are in the /var/www/cgi-bin directory. They are activated by adding the extension /cgi-bin/.

Examples for activating CGI files:

- 192.168.1.12/cgi-bin/ether
  Activates CGI file /var/www/cgi-bin/ether
- 192.168.1.12/cgi-bin/restart
  Activates CGI file /var/www/cgi-bin/restart

CGI files

The BOA web server can execute files in the /var/www/cgi-bin directory. These files will output HTML code via the standard console "stdout" which will be transmitted via the WWW by the web server. Binary files are stored for the WAGO pages (delivery status) that write on standard output. The files are written using the C programming language. See also the example program in chapter 8.

Other types of CGI files such as Perl scripts can also be used, however, it is necessary to install the appropriate interpreters / auxiliary programs. Please read more about the web server in the BOA documentation on the Internet.
4.5.6 Netflash

Netflash is a utility program that serves to load image files to your PC via a network and to write them to the flash memory. The possibility to update the system via the network offers a great advantage in terms of system maintenance and usability, especially in embedded systems.

Netflash supports to download image files with different protocols such as http, tftp and ftp. If no options are enabled (see next page), the flash memory will be updated in the following way:

- All processes are terminated
- The indicated file is loaded
- The checksum of the loaded file is verified
- If the checksum is correct, the image is written to the flash memory of the Linux fieldbus controller
- The Linux fieldbus controller is rebooted

The general syntax for netflash is:

```
netflash [-bCfFhijklmnuz?] [-c console-device] [-d delay] [-o offset] [-r flash-device] [net-server] filename
```

Options

- `-b`
  No restart after the flash memory update
- `-C`
  Checks if the image was written correctly
- `-f`
  Uses FTP as load protocol
- `-F`
  Forces all areas (blocks) of the flash memory to be overwritten
- `-h`
  Opens the help menu
- `-i`
  Ignores all version information
- `-H`
  Ignores the hardware type information
- **j**
  The image to be written contains the JFFS2 file system

- **k**
  Processes are not terminated (will be ignored if the Root file system is in the flash memory)

- **l**
  Protects the areas of the flash memory after writing data

- **n**
  Image has no checksum (image without version information)

- **p**
  Contains the blank areas of the flash memory

- **s**
  Stops deleting and writing at the end of the input data

- **t**
  Checks the image and deletes it if it is faulty

- **u**
  Enables the areas of the flash memory before data is written

- **v**
  Indicates the version number

**Example:**

After the log-in on the buscoupler via Telnet, an example application with `netflash` could look like the following:

```
netflash -k -n -j -r /dev/mtd3 192.168.1.11 jffs2.img
netflash: fetching file "jffs2.img" from 192.168.1.11
...............................................................
...............................................................
netflash: got "jffs2.img", length=1159368
netflash: flashing root filesystem, kill is forced
Connection closed by foreign host.
```
4.6 Ethernet Interface Services

The first set is to use the communication cable (item 750-920) to connect the to the serial port of the computer and the configuration interface of the Linux fieldbus coupler.

⚠️ Attention
To start the console, the selector switch must be in the upper position during start-up!

4.6.1 Temporary Assignment of the IP Address (ifconfig)

The IP address of the Linux fieldbus coupler is 192.168.1.3 (delivery status). To allow for communication between the PC and the Linux fieldbus coupler via Ethernet, they both have to be part of the same network. The subnet mask determines which part of the IP address is utilized as network address. The Linux fieldbus coupler has IP address 192.168.1.3 and subnet mask 255.255.255.0. This results in network address being 192.168.1 and the device address of 3.

If you want another address or dynamic address assignment, the configuration must be done via the serial interface of the Linux fieldbus coupler. Connect the serial interface (COM port) to the configuration interface of the Linux fieldbus coupler using the communication cable (750-920).

In order to set the configuration of the new IP address a terminal program must be used to communicate between the two devices. This can be minicom under Linux or Hyperterminal.exe under Windows. The terminal program should be configured with the parameters 38400-8-N-1-NoFlowControl (see chapter 4.3).

A volatile IP adress can then be assigned in the console, e.g.

> ifconfig eth0 192.168.1.4 (sets temporary IP address)

Please use the wagoset program for a permanent configuration of the Ethernet interface.
4.6.2 Permanent Assignment of the IP Address (wagoset)

The wagoset program serves to read and write the Ethernet configuration with the Linux console and user programs. The program only changes the Linux kernel boot parameters that are stored in the flash memory, i.e. the changes are only accepted after the next reboot, however, they are stored permanently.

The program is directly accessed via the console and has the following HELP page:

Usage: ./wagoset OPTION [VALUE] ...

WAGO ETHERNET BOOT CONFIGURATION.

OPTION:

- v, --view  View Configuration
- m, --mode  Ethernet Mode; VALUE: static, dhcp, bootp, none
- i, --ip  Static IP; VALUE: x.x.x.x
- s, --subnet  Subnet Mask; VALUE: x.x.x.x
- g, --gateway  Gateway; VALUE: x.x.x.x
- o, --hostname  Hostname; VALUE: [String Hostname]
- h, --help  Print this message.

Proprietary. Send bug reports to support@wago.com

To view the current settings use

./wagoset -v

To start the Linux fieldbus coupler with IP address 192.168.1.4 during the next boot use the following command

./wagoset -m static -i 192.168.1.4 -s 255.255.255.0 -g 0.0.0.0
Dynamic IP address assignment via DHCP during the next boot of the Linux fieldbus coupler is achieved by:

```
./wagoset -m dhcp
```

If you want to assign the IP address using a BootP server, do the following:

```
./wagoset -m bootp
```

After that, reboot the Linux fieldbus coupler.

---

**Attention**

If automatic configuration via DHCP or BootP is selected, the values for IP address and subnet mask are reset to default values, since they are not used and no longer stored. Consequently, if DHCP or BootP is changed to a static IP address, it is necessary to set the IP address and subnet mask values again.

---

### 4.6.3 Dynamic IP Address Assignment during Runtime

You can use the wagoset program (Section 4.6.2) to change the boot command of the kernel with regard to the Ethernet connection. You can select whether the kernel will get its IP address via DHCP or Bootp or whether a static IP address is set (Section 4.6.1).

Another possibility to get the IP address dynamically is to start appropriate programs (bootpc or dhcpcd) in the user space after the kernel has been started. This results in a time saving during the start-up of the Linux fieldbus controller if the DHCP server or the BootP server are not available. If the Linux fieldbus controller does not find a server, it is in a search loop for up to two minutes before it boots up without having established a connection to one of the servers.

The programs bootpc for Bootp and dhcpcd for DHCP allow to boot up the kernel quickly and search the appropriate server using an individual process. During this time, the system can be used for other tasks. The programs are already available in the file system of the 750-860.
wagoset (start without network connection)

If the Bootp or DHCP server is supposed to be searched during runtime, it is reasonable, in most cases, to start the kernel without Ethernet configuration. Start wagoset using the option `-m none`.

```
~ # wagoset -h

Usage: wagoset OPTION [VALUE] ...
WAGO ETHERNET BOOT CONFIGURATION.
OPTION:

-v, --view      View Configuration
-m, --mode      Ethernet Mode; VALUE: static, dhcp, bootp, none
-i, --ip        Static IP; VALUE: x.x.x.x
-s, --subnet    Subnet-Mask; VALUE: x.x.x.x
-g, --gateway   Gateway IP; VALUE: x.x.x.x
-o, --hostname  Hostname: VALUE [String hostname]
-h, --help      Print this message.

Proprietary. Send bug reports to support@wago.com
```

```
~ # wagoset -m none

No ethernet at bootup: Configuration value saved!
IP_Mode is: none
IP-Address configuration: 192.168.1.10
Subnet Mask configuration: 255.255.255.0
Gateway IP configuration: 192.168.1.1
Hostname configuration: WagoLFBC
```

Dynamic assignments via bootpc

In order to get an IP address and other relevant information from a Bootp server via the network, also after the start of the Linux kernel, use the bootpc program. You can find the program in the `/sbin` directory.

In order to make the use of the Bootp client easier, the `/sbin` directory also provides a Linux script, that executes the call of the bootpc. In order to run the script in the background, execute the following call:

```
bootpc-startup &
```

The Bootp client is started in the background and sends a Bootp request approx. every two seconds. With a correct reply, the Ethernet interface is parameterized automatically (IP address, netmask, gateway).

The program and the script are terminated as soon as a valid Bootp server has sent a reply. The call can also be a part of the boot script `/etc/inittab`, of course.
Dynamic assignments via dhcpcd

If a DHCP server is supposed to be requested after the start of the Linux kernel, the dhcpcd program can be used. The program is in the /sbin directory and can hence be started by any user.

The call is dhcpcd &

The DHCP client is started in the background and sends DHCP requests regularly. With a correct reply, the client automatically configures the Ethernet interface.

The program is not terminated if a valid response is received from a DHCP server.

The call can also be a part of the boot script /etc/inittab, of course.

4.6.4 Checking the Network Connection

1. In order to check communication with the Linux fieldbus coupler and the correct IP address assignment, start the DOS prompt via Start menu/Programs/Command prompt.

2. Type the command ping using the IP address you assigned and using the following syntax:
   ping [blank] XXX.XXX.XXX.XXX (IP address).
   Example: ping 10.1.254.202

   ![Ping Command Example](image)

   Fig. 4-10: Example for a fieldbus node function test

3. After pressing the return key, your PC gets a response from the Linux fieldbus coupler which is shown in the DOS prompt. Should the error message: "Request timeout", appear, please verify your input and compare it with the IP address you assigned.

   After a successful test, you can close the DOS prompt. The node is now ready to communicate.
5 Step-by-Step Guide to your own Linux Application

This chapter will guide you step by step, to create your own Linux application. First it will guide through the installation of the toolchain on the development computer. Then, after the toolchain is installed this document shows how to create and compile an application. The final step will load the application and run it on the Linux fieldbus coupler.

5.1 Installation of the Development Toolchain

To create uClinux or Linux applications you require a toolchain. Basically, you can compile a toolchain yourself. However, this requires profound knowledge about the individual GNU components, the specific characteristics of the different versions as well as their interdependencies. Since the effort is disproportionate to the results, we recommend the use of the toolchains that have already been compiled. For the Linux fieldbus coupler this is the so-called arm-elf toolchain.

Usually, your own applications will run in the user space; it is also possible to run drivers/programs in the kernel space, however, it is essential to know the hardware and driver programming (book recommendation: Linux Device Drivers by Rubini & Corbet, published by O’REILLY).

If you want to change or to compile the boot loader (U-Boot), you have to install a second toolchain.

A pre-compiled version of the arm-elf toolchain for x86 systems is availabale under http://www.uclinux.org/pub/uClinux/m68k-elf-tools/, or on the Wago distribution CD under CD:/linux/toolchain/.

The installation requires super user privileges.

> su
Change into super user mode (enter password)

> cd /media/cd-rom/linux/toolchain
Change directory (in S.U.S.E, a CD drive is mounted in /media/cd-rom if it was linked via Konqueror, for example)

> sh install.sh
Call installation script

> exit
Exit super user mode
5.2 Unpack the Source Code

The Linux kernel source codes and the source code of the example programs described in the documentation are on the CD in the following file: CD:/linux/sources/. The installation should by all means be done in the user mode, not in the super user mode:

```
> cd /media/cd-rom/linux/sources
```

(in S.U.E, a CD drive is mounted in /media/cd-rom if it was linked via Konqueror, for example)

```
> sh install.sh
```

Call installation script

The installation is done automatically. During the installation a/uclinux-dist directory is created in the home directory of the user in which are stored the files required for the compilation of the Linux kernel, for the creation of the file system and for the compilation of the application programs.
5.3 Linux Kernel Compilation

All the work on the Linux kernel and on the user space programs should be done in the user mode (not in the super user mode). You should also run the make command in the user mode. The super user mode is only required for the one-time setup of the host PC.

The make program in the ~/uclinux-dist folder is used to compile the Linux kernel, the file system or a user space application. The make function is determined by call parameters and the Makefile file. The following order is mandatory for the first compilation of the kernel.

> cd ~/uclinux-dist
> make menuconfig

Starts a graphical configuration tool to set the kernel options. (Exit the dialog with EXIT and end the program without saving changed settings.)

Note
If the graphical interface to configure the distribution cannot be opened, please install the NCURSES components from your PC Linux distribution on your host PC

> make

Note
In some versions of S.U.S.E, make aborts during the first compilation. In this case, please restart make. Make works correctly if kernel information is shown after the compilation process.

Make calls the program to execute Makefile, make is the short form of make all, all being the argument. The following attributes can be used:

- all
  Synonym for: linux uClibc user romfs image

- linux
  Kernel compilation

- romfs
  Creates the root file system in the /romfs directory
- **user**
  User program compilation

- **image**
  Creates the HEX images: linux.flashme (Linux kernel), jffs2.img
  (file system with user programs)

- **menuconfig**
  Invokes a menu-driven configuration of the Linux kernel, of the modules
  and of the application programs

- **clean**
  Deletes all files that were created by compilation

- **distclean**
  Deletes all files created by configuration and compilation
5.4 Setup of the Development Environment

For application creation, administration and diagnostics you need a PC with a current Linux distribution. We recommend S.U.S.E version 8.3 to 10.0 as Linux distribution. The performance of the developer's PC should at least meet the requirements indicated by the Linux distribution.

Communication with the Linux fieldbus coupler is possible via the serial and via the Ethernet interface. In order to avoid communication problems between PC and Linux coupler via Ethernet, we recommend that both devices operate in the same network.

A SUSE standard installation already provides you with the most important programs for the communication with the Linux coupler.

<table>
<thead>
<tr>
<th>Task</th>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial console</td>
<td>minicom</td>
<td>For communication with console of Linux fieldbus coupler via serial interface</td>
</tr>
<tr>
<td>Ethernet console</td>
<td>telnet</td>
<td>For communication with console of Linux fieldbus coupler via Ethernet</td>
</tr>
<tr>
<td>Web browser</td>
<td>konqueror</td>
<td>Tool to view HTML files or to query the web server of the Linux fieldbus coupler</td>
</tr>
<tr>
<td>File explorer</td>
<td>konqueror</td>
<td>Visual tool to navigate the file system</td>
</tr>
<tr>
<td>File transfer</td>
<td>konqueror</td>
<td>Visual tool for the FTP file transfer via &quot;Drag and Drop&quot;</td>
</tr>
<tr>
<td>Editor</td>
<td>kate, emacs</td>
<td>Any editor</td>
</tr>
</tbody>
</table>

The installation of the following programs adds to the convenience:

- ncurses
  Part of S.U.S.E installation if all development packages are installed

- FTP client/server
  For file exchange with the FTP server/client of the Linux coupler

- NFS server
  For easy file exchange with the Linux fieldbus coupler

In order to update images for Bootloader, firmware and file system on the Linux fieldbus coupler you require a TFTP server.
Please see chapter 9.2.
5.4.1 PC Terminal Program Configuration (serial)

In order to communicate with the Linux fieldbus coupler console via the serial interface of your PC, you require a terminal program. This can be minicom under Linux, or Hyperterminal under Windows.

During the Linux coupler start-up, the selector switch must be in the upper position in order to start the Linux console.

The Linux console has the following preset values (38400 8-N-1):

- Baud rate: 38400 bits per second
- Data bits: 8
- Parity: none
- Stop bits: 1
- Flow control: none

If minicom is not communicating, check if the right interface has been configured. To do so, start the program with the option "-s" and verify the settings in the menu item "Serial port setup".

![Configuration of the serial interface in minicom](image)

Press the <Strg> + <a><z> key combination to view a list with available functions.
Press the <Strg> + <a><x> kex combination to terminate minicom.

In order that the backspace key functions properly, the option "Backspace sends Del(ete)“ needs to be set. In minicom, this is done via the <Strg> + <a><t><b> key combination.

In the Windows operating systems Hyperterminal is the recommended program for the serial terminal communication. The following figure shows the required settings.

![Configuration of the serial interface in Hyperterminal](image)
5.4.2 PC Terminal Program Configuration (Ethernet)

In order to communicate with the Linux fieldbus coupler console via Ethernet, a terminal program is required. Under Linux you can simply start the telnet program via the console.

Under DOS or the MS-DOS prompt you can also start a Telnet client using the call `telnet`.

Under Windows, the Hyperterminal program can also communicate with the Linux fieldbus coupler via telnet. Hyperterminal then requires the following settings:

![Hyperterminal: Telnet settings](image)

In the "Connect to" dialog, select TCP/IP in the "Connect via" selection field. In the next window, enter the IP address (see also chapter 4.6) or the host name of the Linux fieldbus coupler and the port number 23 for Telnet.
5.5 Setting the IP address of the Linux Fieldbus Coupler

The Linux fieldbus coupler has the factory-set IP address. If you want another address or dynamic address assignment, the configuration is done via the serial interface of the Linux fieldbus coupler. Connect the serial interface (COM port) to the configuration interface of the Linux fieldbus coupler using the communication cable (750-920).

In order to set the configuration of the new IP address a terminal program must be used to communicate between the two devices. This can be minicom under Linux, or Hyperterminal.exe under Windows. The terminal program should be configured with the parameters 38400-8-N-1-NoFlowControl.

A volatile IP address can then be assigned in the console, e.g.

> ifconfig eth0 192.168.1.4 (sets temporary IP address)

Please use the wagoset program for a permanent configuration of the Ethernet interface. This program writes the settings to the flash memory of the Linux coupler. In this way the settings are also available after the next start-up.

> wagoset -i 192.168.1.8 (sets permanent IP address)

Dynamic IP address assignment via DHCP for the next start of the Linux filedbus coupler can be set as follows:

> wagoset -m dhcp
5.6 Create your own User Space Application

Please use the ~/uClinux/user directory for user space applications. Create a subfolder for your project.

> cd ~/uclinux-dist/user
Change to root directory for user space applications.

> mkdir newProg
Create project folder newProg

> cd newProg
Change to project folder newProg

A minimum project consists of a source code file and a make file. Create the source code file newProg.c, e.g. using EMACS, KATE or JOE.

> emacs newProg.c
Call EMACS editor

Insert the following code into the emacs and press CTRL-X CTRL-S to save it.

```c
int main (int argc, char **argv)
{
  printf("Hello World\n");
}
```
Create the make file, e.g. using EMACS, KATE or JOE.

> emacs Makefile
Call EMACS editor

Insert the following code into the emacs and press CTRL-X CTRL-S to save it.

```
all: newProg
newProg: newProg.o
  $(CC) $(LDFLAGS) -o $@ newProg.o $(LDLLIBS$(LDLLIBS-$(@)))
    cp newProg /targetfs/
romfs:
  $(ROMFSINST) /bin/newProg
clean:
  rm -f newProg *.elf *.gdb *.o
```

In order to connect the make file to the global make file of the cross compiler, you have to paste the new directory into the ~/.uclinux-dist/user/Makefile make file in the dir_y variable:

> emacs ../Makefile

Tippen Sie den folgenden Code in den emacs und speichern Sie ihn CTRL-X CTRL-S.

```
...
dir_y = kbusapi ... newProg
...
```

The program, along with all other user space programs, is compiled if make in the ~/.uclinux-dist directory is executed with the following option (see chapter 5.3).

> cd ~/.uclinux-dist
> make user
A binary file "newProg" is created which can be copied into the Linux fieldbus coupler via NFS or FTP, for example.

> cd ~/.uclinux-dist
> make user romfs image
This will also create a binary file newProg. At the same time, the romfs directory is updated and the new image of the file system jffs2.img is created which can be loaded via the boot loader (U-Boot) (see 5.10.2).
5.7 Transferring the User Space Application to the Linux Coupler

The user space application is copied into the /targetfs directory by the make file. The /targetfs directory can be accessed via NFS (see chapter 5.11). The /targetfs directory needs to be embedded into the Linux fieldbus coupler using the mount command. It can now be directly executed on the coupler.

The user space application can also be copied into the Linux fieldbus coupler via FTP and then be executed locally. To do so, enter the following on the host PC.

```bash
> cd ~/uclinux-dist/user/newProg
> ftp 192.168.1.8
root  (Benutzer)
wago  (Passwort)
> cd /bin
> put newProg
> quit
```

Screenshot:

```
Fig. 5-4: Linux screen view after program download
```

WAGO-I/O-SYSTEM 750
Linux Fieldbus Coupler
The Konqueror is another FTP client. Start the program and change to the /targetfs root directory. Start another instance of the program instance and change to the target directory of the Linux fieldbus coupler using ftp://<IP address>/bin. Now you can drag and drop files between the directories.

5.8 Running the new Program

The new program can be executed on the Linux coupler via the console, e.g. via the terminal (see chapter 4.3).

> chmod +x /bin/newProg
In any directory you can run the command

> newProg
HelloWorld!

5.9 Automatic start of new Programs (inittab/startwago)

During the startup of the operating system, Linux starts the processes that can be found in the inittab file during boot-up: in our system, inittab looks as follows:

::sysinit:/etc/rc
::sysinit:/etc/startwago
::respawn:/etc/getty -L ttyS0 38400

In this file, the startwago file with the attribute sysinit is run, among others. This means, the startwago script is started once during the system start-up. The getty program which also supports the serial interface is started as a respawn process. This process is thus restarted automatically if it is terminated. The attributes are:

- sysinit
  Is started once during system start-up.
- bootwait
  Is started once after all sysinit processes.
- respawn
  Is restarted automatically if the process was terminated.

Please refer to the initab manpage for more attributes.
initab calls the startwago script. startwago includes further program calls. Since this file is a shell script, it is also possible to use conditional queries and other shell commands. The startwago file on the Linux fieldbus coupler looks as follows:

```
#!/bin/msh
# change setuid/setgid-Bit for tinylogin(su)
chmod +s /bin/tinylogin
# start the KBus-Driver
insmod kbus
# start modbus tcp server
/bin/mb_tcp &
# start web server (BOA)
boa -c /etc/boa &
# if it is configured, start the IOCheck-Driver /-Application
if grep -q "wago_console=yes" /proc/cmdline
then
  rm /etc/inittab
  ln -s /etc/wago_console/inittab.nottys0 /etc/inittab
  insmod iocheck
else
  rm /etc/inittab
  ln -s /etc/wago_console/inittab.ttys0 /etc/inittab
  /bin/iocheck &
  insmod iocheck
fi
exit 0
```

During the start-up of the Linux fieldbus coupler, different programs are started or file attributes are changed. For example, the BOA web server and the Modbus program mb_tcp are started.
5.10 File System and Linux Kernel Update

If you do not want to update a single program but the entire file system or the Linux kernel, this is possible using TFTP. The make parameters `romfs` or `image` create image files which are copied into the Linux coupler via TFTP.

In order that the file system and the Linux kernel can be updated, the user needs to have access rights for the `/tftpboot` output directory in the root directory:

```
> su
Password
> mkdir /tftpboot
> chmod 777 /tftpboot
```

To create new image files, use the following call:

```
> cd ~/uclinux-dist
> make romfs image
```

To restore the Linux coupler default values, use the image files of the Bord Support Package (BSP).

The image files `jffs2.img` and `linux.flashme` are created in the `/tftpboot` directory. The image files are transferred to the Linux fieldbus coupler via TFTP. For that purpose, you have to set up and start a TFTP server, which has `/tftpboot` as a base directory, on the development computer. The boot loader (U-Boot) on the Linux coupler provides a TFTP client. It is controlled via a separate console. To open the console, enter the password `geheim` during the Linux coupler start-up. The boot process is aborted and the Linux fieldbus coupler opens the boot loader communication interface. You can use the `update-kernel` and `update-rootfs` macros for the update.
5.10.1 TFTP Server Setup

In order to download the Linux kernel, the file system or the boot loader, it is essential to have a TFTP server on the development computer.

You can find the original image files for the Wago Linux distribution under CD:/images/.

The /tftpboot/ directory is the output directory of the compiler for the file system and Linux kernel image files and has to be configured as source directory for TFTP.

In order to be able to use TFTP, it is required to install inetd and atftp (using yast2) or xinetd.

When using xinetd, the configuration of the /tftpboot/ source directory is done via the graphical desktop environment of KDE.

If you use inetd, make the following entry in the /etc/inetd.conf after the installation:

```bash
<host-ip>:tftp dgram udp wait root /usr/sbin/in.tftpd -s /tftpboot
```

inetd starts automatically after a reboot or it can be started manually with

```bash
> /etc/init.d/inetd start
```

gestartet werden.

You have to install inetd and atftp via yast2 (S.U.E) before you can install a TFTP server on the development PC. After the installation, make the following entry in the /etc/inetd.conf:

```bash
<host-ip>:tftp dgram udp wait root /usr/sbin/in.tftpd -s /tftpboot
```

inetd starts automatically after a reboot or it can be started manually with

```bash
> /etc/init.d/inetd start
```

or restarted with

```bash
> /etc/init.d/inetd stop
> /etc/init.d/inetd start
```

erneut gestartet werden.
5.10.2 TFTP Client Setup

File system changes can also be done directly in Linux, however, it is not possible to replace the entire file system at the same time. An update exclusively of the Linux kernel or exclusively of the file system is only possible via the TFTP client of the boot loader (U-Boot).

To have access to the boot loader configuration interface, turn the selector switch to the upper position, connect the communication cable to the configuration interface of the Linux coupler, start the console program, e.g. minicom, and switch on the voltage of the Linux fieldbus coupler.

When the following is output:

I2C-EEPROM detected.
I2C-RTC detected.
In: serial
Out: serial
Err: serial
Booting in 3 seconds - enter abort password to prevent this
you have to enter the password **geheim** (on the serial console) to abort the boot loader boot process and to change to the U-Boot environment. The boot loader switches to its shell environment. If after three seconds the password has not been entered, the boot loader loads the Linux kernel.

The `printenv` command outputs the environment variables:

```
LFBK # printenv

bootargs=root=/dev/mtdblock3 rw mem=16769120
bootdelay=3
baudrate=38400
ipaddr=192.168.1.3
netmask=255.255.255.0
update-rootfs=tf 0x40000 jffs2.img;protect off 01:11-51;era 01:11-51;cp.b 0x40000 0x20040000 $(filesize)
update-kernel=tf 0x40000 linux.flashme;protect off 01:52-69;era 01:52-69;cp.b 0x40000 0x202d0000 $(filesize)
update-ub=tf 0x40000 ub.bin;protect off 01:08-9;era 01:08-9;cp.b 0x40000 0x20010000 $(filesize)
bootA=setenv bootargs $(bootargs) wago_console=yes ip=$(ipmode) $(output);bootm
bootB=setenv bootargs $(bootargs) ip=$(ipmode) $(output);bootm
```
The IP addresses of the Linux fieldbus coupler and the TFTP server must be checked. If the IP addresses are not conclusive, they possibly need to be adapted using the following call:

```
setenv ipaddr <IP-Linux-FBK>; setenv serverip <IP-Host-PC>
```

See also chapter 9.5 "Environment Variables".

An update of the Linux kernel or of the entire file system can be started via the calls

```
LFBK # run update-kernel
LFBK # run update-rootfs
```

These macros execute the following U-Boot commands:

1. Load image file into the memory via TFTP.
2. Disable write protection on flash memory.
3. Delete designated area of the flash memory.
4. Copy image file into flash memory.
An example for a Linux kernel update:

LFBK # run update-kernel

Station Address: 20:00:00:7B:FF:07
HW-MAC Address: 00:20:07b:00:07ff

Intel LXT97xA Ethernet located via MII
10Mbs - half duplex link established
TFTP from server 192.168.1.2; our IP address is 192.168.1.3
Filename 'linux.flashme'.

Load address: 0x40000

Loading: #################################################
#################################################
#################################################
######################################
done

Bytes transferred = 1239096 (12e838 hex)

Un-Protect Flash Sectors 52-70 in Bank # 1
Erase Flash Sectors 52-70 in Bank # 1
Erasing sector 52 ... ok.
Erasing sector 53 ... ok.
Erasing sector 54 ... ok.
Erasing sector 55 ... ok.
Erasing sector 56 ... ok.
Erasing sector 57 ... ok.
Erasing sector 58 ... ok.
Erasing sector 59 ... ok.
Erasing sector 60 ... ok.
Erasing sector 61 ... ok.
Erasing sector 62 ... ok.
Erasing sector 63 ... ok.
Erasing sector 64 ... ok.
Erasing sector 65 ... ok.
Erasing sector 66 ... ok.
Erasing sector 67 ... ok.
Erasing sector 68 ... ok.
Erasing sector 69 ... ok.
Erasing sector 70 ... ok.
Copy to Flash... done
LFBK #

The host computer, which will provide the new image files, needs to incorporate a TFTP server in order to allow an update. In order to eliminate confusion, every file is given an unambiguous name. A JFFS2 image of the file system is called `jffs2.img` and the kernel file with bootloader header is called `linux.flashme`. The files are available on the WAGO Linux distribution CD under `CD:/images/` and must be copied into the `/tftpboot/` directory first or be generated in the uClinux directory using make.

**Note**
You can get additional information on the interface with the `help` command. Example: LFBK # help
5.11 NFS Server Setup

NFS (Network File System) allows access to files across the networks as if they were local files. This service is available under both Linux and Unix.

The local /targetfs directory serves as output directory for the application creation. It can be linked to the Linux coupler file system via NFS.

To enable NFS support, add the following line to the file /etc/exports:

```
/targetfs *(rw,sync,all_squash,anonuid=<uid>,anongid=<gid>)
```

The Linux user ID/group ID must be entered as UID and GID. One way to determine the number is

```
> id
uid=501(u04711) gid=100(users) Gruppen=100(users),14(uucp)...
```

which results in:

```
/targetfs *(rw,sync,all_squash,anonuid=501,anongid=100)
```

Linux coupler NFS client configuration

NFS is integrated into the Linux kernel and is used as a file system. In order to link a directory of a remote system it has to be assigned to the Linux directory structure using the mount call, just like a partition on the hard drive.

In order to be able to use the NFS service, a NFS server and an enabled directory must be available on the remote PC. The remote directory is linked to the Linux fieldbus coupler file system by the call

```
mount -t nfs -o nolock <IP/hostname>:<Directory> /<local directory>
```

Example: > mount -t nfs -o nolock 192.168.1.12:/targetfs /nfs

To serve this purpose, the Linux coupler provides the /nfs drive (delivery status). The drive can be accessed like a local drive. If drives are to be linked automatically during system start-up, they must be entered in the /etc/rc file.
5.12 Debugging of User Space Programs

5.12.1 Description of the GDB

The GNU Debugger (GDB) is the standard debugger for the GNU Project. It is a portable debugger that runs on many systems and works for many programming languages, including C, C++ and Fortran. GDB is free software released under the GNU General Public License (GPL).

GDB offers extensive facilities for tracing and altering the execution of programs. Using GDB, the program's internal variables can be monitored and modified and functions can be called independently of the program's normal behavior.

The GDB is to be used in connection with the GDB server to allow cross debugging between a PC and the Linux fieldbus coupler. Start both the GDB server on the Linux fieldbus coupler and the GDB debugger on the PC. Both programs are connected via the Ethernet interface providing a complete debugger including all standard functions. Using this type of configuration, all user space programs can be checked.

Debugging is done using the command lines of the GDB as no graphical user interface is available at the moment. However, there are also graphical front ends to GDB, such as DDD (Data Display Debugger). This graphical user interface allows the use of GDB in the same way as in an integrated development environment.
5.12.2 Preparation for Using the Debugger

GDB has automatically been installed on your PC when installing the toolchain. When delivered, the GDB server is already on the Linux fieldbus controller and can be called directly.

In order to use the graphical front end DDD on the PC, it must be installed over the distribution that is used. With the S.U.S.E Linux distribution, this is completed as follows:

1. Click on the program “Control center” and select “Yast2 Module” → Software → Software installation.

2. A search window opens. Enter DDD and press the Enter key.

3. If the search result shows the DDD application, install it. If the DDD is not in the distribution, it can be downloaded at: www.gnu.org/software/ddd/.
5.12.3 Creating a Testable User Space Application

In order to avoid illogical sequence in the program run during debugging with GDB, it is necessary to disable the optimization of the compiler before the user program to be tested is compiled. Enter the following calls into the console:

```
> cd ~/uclinux-dist
> make menuconfig
```

A window opens. Select the menu path **Kernel → Library → Default Settings** and press the Enter key. The following window will open:

![Fig. 5-: Window Make menuconfig, enable user settings](image)

1. Select “Customize Vendor/User Settings” and press the space bar.
2. Quit the menu and the following menus by clicking the **EXIT** button.
3. The following question appears "Do you wish to save your kernel configurations?". Click on **YES**.
4. Another menu opens. Select “Debug Builds” and press the Enter key.
5. Select “Build debuggable applications” and press the space bar.
Step-by-Step Guide to your own Linux Application
Debugging of User Space Programs

1. Quit the menu and the following menus by clicking the **Exit** button.
2. The following question appears "Do you wish to save your kernel configurations?". Click on **YES**.

After the above-described settings are done, the compiler is ready to compile user space applications without optimization. The compiled programs become bigger and will likely operate at a slower speed.

For final release versions of the programs, it is advisable to cancel the above setting ("Build debuggable applications") and completely compile the program once more. To do so use the following calls:

```
> make clean
> make (or make user).
```
5.12.4 Creating an Example Application for Debugging

The section 5.6, “Create your own user space applications” describes how to create an example program for the Linux fieldbus controller. This program example also illustrates debugging. Add the following calls to the source codes:

> cd ~/uclinux-dist/user/newProg
> emacs newProg.c

Enter the following code into the emacs and save it via CTRL-X, CTRL-S:

```c
void printfunc(void)
{
    int i = 3;
    printf("WAGO world\n");
    printf("%d..", i);
    i--;
    printf("%d..", i);
    i--;
    printf("%d..\n", i);
    return;
}

int main(int argc, char **argv)
{
    printf("Hallo\n");
    printfunc();
    return 0;
}
```

A subfunction with a counter variable is added to the code in order to be able to test the functions of the debugger better.

The makefile of section 5.6 is not modified.

Recreate the program. Enter the following calls into the console:

> cd ~/uclinux-dist
> make user

Copy the program to the Linux fieldbus controller, as described in section 5.7.
To simplify the operation of GDB, a script needs to be generated that creates defined starting conditions for the debugger. Use the following call to generate the script:

```shell
> emacs gdb.script
```

Enter the following code into the emacs and save it via CTRL-X, CTRL-S:

```plaintext
target remote 192.168.1.3:1234
b main
continue
```

If the default IP address of the Linux fieldbus controller has been changed, it is necessary to also change the IP address that is in the first line of the above-mentioned script.

In the first line of the script, a remote connection is automatically established to the GDB server of the Linux fieldbus controller with the IP address 192.168.1.3 using port number 1234. The second line sets a breakpoint to the main routine and the third line starts the program that runs until the breakpoint is reached.

### 5.12.5 Debugging via the GDB Console

In order to enable cross debugging between the PC and the Linux fieldbus controller, a connection via Ethernet is used. In the procedure described below the Linux fieldbus controller has the default IP address 192.168.1.3, the Linux PC has IP address 192.168.1.4. If your settings differ from the settings indicated above, change the IP addresses so that they match the addresses in the example.

First, start the GDB server on the Linux fieldbus controller. To do so, change the directory in which is the program that has been created in the previous section:

```shell
> cd ~/uclinux-dist/user/newProg
```

Enter the following call into the console:

```shell
> gdbserver 192.168.1.4:1234 newProg
```
Fig. 5-6: Starting the GDB server on the 750-860

The GDB server will now wait for a connection request (Listening) of the GDB client via port 1234 of IP address 192.168.1.4. Port 1234 is chosen arbitrarily, another port can also be used, of course.

To start GDB on the Linux PC, enter the following calls:

\[ \text{cd } ~/\text{uclinux-dist/user/newProg} \]

\[ \text{arm-uclinux-elf-gdb newProg.gdb --command=gdb.script} \]

Fig. 5-7: Starting GDB on the PC

You can use the following commands for debugging:

<table>
<thead>
<tr>
<th>GDB command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>step</td>
<td>The program is executed until the next program line is reached. Also steps into subroutines if it is the next code line.</td>
</tr>
<tr>
<td>stepi</td>
<td>The program is executed until the next assembler code line.</td>
</tr>
<tr>
<td>next</td>
<td>The program is executed until the next program line is reached. Skips subroutines.</td>
</tr>
<tr>
<td>nexti</td>
<td>The program is executed until the next assembler code line. Skips subroutines.</td>
</tr>
<tr>
<td>continue</td>
<td>Continues to execute the program. The next breakpoint or pressing STRG+C tops the execution.</td>
</tr>
</tbody>
</table>
### GDB command | Description
--- | ---
**break** | Sets a breakpoint.  
*break newProg.c:20* | Sets a breakpoint at line 20 in the main.c file  
*break main* | Sets a breakpoint at the main function start  
*break *0x00dd8080* | Sets a breakpoint at the indicated address in the program code
**list** | Lists ten code lines around the current position of the program counter. Another list call will list the next ten lines.  
*list newProc.c:15* lists ten lines around line 15 in the newProc.c file
**print i** | Displays the current value of the i variable.  
*print *pi* displays the content of the memory area the pointer pi points to.
**display i** | Displays the current value of the i variable. The value or the values are updated after every execution of code (step, next, continue)
**undisplay** | Undisplays the value that was displayed via the display call before
**whatis i** | Displays the data type of the i variable.
**backtrace** | Displays the calling functions and their parameter values from the stack.
**where** | Displays the content of the stack like backtrace.
**set var i = ?** | set var i = 5 sets the i variable to 5.
**help** | Displays help texts for the available commands.  
*help continue*, for example, displays a help text for continue,  
*help set* displays the help text for set
**quit** | Terminates the remote program
**exit** | Exit the debugger.
**run** | run call is not possible. The program counter is set to 0x00000000 which, with the type of CPU used (without MMU), will lead to a restart of the Linux fieldbus controller.

Debugging via the console is not a very convenient way of doing it, however, it offers a very good opportunity to control user space programs of the Linux fieldbus controller via a PC.
5.12.6 Debugging via the Graphical Interface DDD

To debug via the graphical interface, you must first install a graphical front end for command line debuggers on the PC. In this example it is DDD. See also section 5.12.2.

Like debugging via GDB, it is necessary to start the GDB server on the Linux fieldbus controller. To do so, change the directory in which the program that has been created. To start the GDB server, enter the following calls into the console:

```
> gdbserver 192.168.1.4:1234 newProg
```

The GDB server will now wait for a connection request via port 1234 of IP address 192.168.1.4.

The graphical interface DDD and the debugger GDB will be started on the Linux PC when you enter the following calls into the console:

```
> cd ~/uclinux-dist/user/newProg
> ddd newProg.gdb --command=gdb.script --debugger arm-uclinux-elf-gdb --gdb
```
Fig. -: DDD: Graphical interface of the debugger

DDD now offers all the features of an integrated development environment. The operation is not described further in this manual.
6 Process Image

6.1 Basic Structure

After being started, the Linux fieldbus coupler automatically loads the internal bus driver. It recognizes all the IO modules of the node that send or receive data (data width/bit width > 0).

The node can consist of a mixed arrangement of a maximum of 64 analog and digital I/O modules.

---

**Note**

The use of the WAGO module bus extension coupler module 750-628 and end module 750-627 makes it possible to connect up to 250 modules to the Linux fieldbus coupler.

---

**Note**

For the number of input and output bits or bytes of the individual IO modules please refer to the corresponding description of the IO modules.

The coupler generates an internal local process image from the data width and the type of I/O module as well as the position of the I/O modules in the Linux fieldbus coupler node. This is divided into an input and an output area.

The data of the IO modules is separated for the local input and output process image in the sequence of their position after the Linux fieldbus coupler in the individual process image.

First, all the byte-oriented I/O modules are filed in the process image, then the bit-oriented I/O modules. The bits of the digital modules are grouped into bytes. Once the number of digital I/O’s exceeds 8 bit, the coupler automatically starts another byte.

---

**Note**

If a node is changed, this may result in a new process image structure. In this case the process data addresses also change. In the event of adding modules, the process data of all previous modules has to be taken into account.
6.2 Example for an Input Process Image

The following figure is an example of an input process image. The configuration comprises 16 digital and 8 analog inputs. The input process image thus has a data length of 16 bytes for the analog modules and 2 bytes for the digital modules, i.e. 18 bytes in total.

Fig. 6-1: Example of an input process image
6.3 Example for an Output Process Image

The following example for the output process image comprises 2 digital and 4 analog outputs. It comprises 8 bytes for the analog outputs and 1 byte for the digital outputs, i.e. 9 bytes in total.

Fig. 6-2: Example of an output process image
6.4 Application Example for a Mixed Node

Fig. 6-3: Addressing example for a fieldbus node
6.5 Internal Bus Driver (kbus.ko)

The internal bus driver *kbus* is a char device driver. It is therefore possible to access the connected I/O modules using the standard device interface. The driver is started automatically via the `/etc/startwago` file when the Linux fieldbus controller is started.

In order not to have to create definitions manually, the header file *kbusapi.h* can be used in its own program. Functions that are additionally available in the file (e.g., KbusUpdate) are no longer required and only included for compatibility reasons.

**Note:**

The file *kbusapi.h* provides several macros that make it easier to access the process image, however, they are not essential. It may be beneficial to customize structures with meaningful names taken from the control process.

**Source code files:**

`~/uclinux-dist/user/kbusapi/kbusapi.h`

**Driver/Kernel module:** kbus.ko (internal bus support)

**Function Description**

The internal bus driver can be used with the following functions:

- `open("/dev/kbus", O_WRONLY)`
  
  Opens the kbus driver.

- `ioctl(iFD, IOCTL_KBUSUPDATE, &ulTmp)`
  
  Writes the modified data of the output process image to the modules and simultaneously reads the current data of the input modules back in.

- `ioctl(iFD, IOCTL_GETBINOUTPUTOFFSET, &ulOutputOffset)`
  
  Reads out the Offset value. The first digital output module in the output process image has the Offset value. It is therefore easier to access the digital I/O modules.

- `ioctl(iFD, IOCTL_GETBININPUTOFFSET, &ulInputOffset)`
  
  Reads out the Offset value. The first digital input module in the input process image has the Offset value. It is therefore easier to access the digital I/O modules.
• ioctl(iFD, IOCTL_CHANGE_UPDATE_SPEED, ulValue)
  [ulValue from 1-10 is valid]
  Used to set the polling interval to poll ioctl(iFD, IOCTL_KBUSUPDATE, &ulTmp). The default value is 3. This value can be reduced to 1 in order to achieve the highest possible speed of the application. However, this only makes sense with very fast I/O modules and a very small user program. The value can also be increased to 10 if only very slow analog I/O modules are involved and fast updates of the process image are not desired.

• close(iFD)
  Closes the KBus driver

Using predefined structures, it is possible to access the input or output process image byte by byte via the following lines:

pstPabIn->uc.Pab[byteNo]
pstPabOut->uc.Pab[byteNo]

or word by word via the lines:

pstPabIn->us.Pab[wordNo]
pstPabOut->us.Pab[wordNo]

Before the input process image can be analyzed, ioctl(iFD, IOCTL_KBUSUPDATE,...) must be called. After changing the output process image, it is also required to call ioctl(iFD, IOCTL_KBUSUPDATE,...) in order to send the new values to the modules.
Note
The `ioctl(iFD, IOCTL_KBUSUPDATE,...)` function writes data to the I/O modules via the internal bus and simultaneously reads in the I/O module data. The function waits and blocks the system until the new data has been read out from the I/O modules. The time for that depends on the length (number of bytes) of the I/O modules connected to the internal bus and may take up to 50ms if the maximum number of modules is involved.

Note
In order not to reduce the performance of the program, call `ioctl(iFD, IOCTL_KBUSUPDATE,...)` only once during a cyclic program run.

Note
The internal bus driver does not control if several programs change data in the process image and then call `ioctl(iFD, IOCTL_KBUSUPDATE,...)`. Therefore, if two application programs access the process image, make sure that the output data to be written is not overwritten by the other program. The programs should block each other.

Example: bit query
The following example shows how to query a bit:

Using the

```c
if (pstPabIn->uc.Pab[0] & 0x08)
{
    ...
}
```

code, allows you to check the third bit of the first element (‘0000.1000’) in the process array to determine if the bit at address 0.3 is set to TRUE.

Example: how to set a bit
The following example shows how to set a bit:

Using the

```c
pstPabOut->uc.Pab[0]|= 0x08;
```

sample code allows you to perform a boolean OR operation of the value 0x08 with the output process image which sets the bit 0.3 (‘0000.10’) to TRUE.
Example: how to reset a bit

The following example shows how to reset a bit:

Using the

\[ \text{pstPabOut->uc.Pab[0] \&= \sim 0x08;} \]

sample code allows you to perform a boolean AND operation with the inverted value of 0x08 ('1111.0111') with the output process image to set the bit 0.3 to FALSE.

Example: byte query

The following example shows how to query a byte:

Using the

\[
\text{if (pstPabIn->uc.Pab[0] == 38) }
\]
\[
\text{ ... }
\]

code allows you to check whether byte 0 has the value 38 by using the equal conditional operator.

Example: word query

The following example shows how to query a word:

Using the

\[
\text{if (pstPabIn->us.Pab[0] == 0x0008) }
\]
\[
\text{ ... }
\]

sample code allows you to check whether the 0-bit process array (us) has the value 0x0008
6.6 Process Image Analysis via the /proc Directory

In order to analyze the input and output data while the internal bus driver is running, it is possible to view and to read out the process image of the relevant modules via the /proc/driver/kbus/pab file. The start address Byteoffset.Bitoffset of the individual process image is shown.

Example: `cat /proc/driver/kbus/pab`

<table>
<thead>
<tr>
<th>Slot</th>
<th>Terminal</th>
<th>Output: Byte.Bit Offset</th>
<th>Input: Byte.Bit Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>750-553/000-000</td>
<td>0.0</td>
<td>x.x</td>
</tr>
<tr>
<td>9</td>
<td>750-559/000-000</td>
<td>6.0</td>
<td>x.x</td>
</tr>
</tbody>
</table>

---------------------------------------------------------

<table>
<thead>
<tr>
<th>Slot</th>
<th>Terminal</th>
<th>Item number</th>
<th>x.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>750-4xx</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>750-4xx</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>750-4xx</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>750-4xx</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>750-4xx</td>
<td>1.2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>750-4xx</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>750-4xx</td>
<td>1.6</td>
<td></td>
</tr>
</tbody>
</table>

---------------------------------------------------------

PAB In: 00ffe060         PAB Size: 2040 bytes
PAB Out: 00ffe858         PAB Size: 2040 bytes

Slot          Position after the Linux coupler
Terminal      Item number
x.x           No data in the input process image
PAB IN, PAB OUT Base addresses of the process images
7 I/O Modules

7.1 Overview

All listed bus modules, in the overview below, are available for modular applications with the WAGO-I/O-SYSTEM 750.
For detailed information on the I/O modules and the module variations, please refer to the manuals for the I/O modules.
You will find these manuals on CD ROM „ELECTRONICC Tools and Docs“ (Item No.: 0888-0412) or at http://www.wago.com under Documentation.

More Information

7.1.1 Digital Input Modules

Tab. 7-1: Digital input modules

<table>
<thead>
<tr>
<th>DI DC 5 V</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-414</td>
<td>4 Channel, DC 5 V, 0.2 ms, 2- to 3-conductor connection, high-side switching</td>
</tr>
<tr>
<td>DI DC 5(12) V</td>
<td></td>
</tr>
<tr>
<td>753-434</td>
<td>8 Channel, DC 5(12) V, 0.2 ms, 1-conductor connection, high-side switching</td>
</tr>
<tr>
<td>DI DC 24 V</td>
<td></td>
</tr>
<tr>
<td>750-400, 753-400</td>
<td>2 Channel, DC 24 V, 3.0 ms, 2- to 4-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-401, 753-401</td>
<td>2 Channel, DC 24 V, 0.2 ms, 2- to 4-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-410, 753-410</td>
<td>2 Channel, DC 24 V, 3.0 ms, 2- to 4-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-411, 753-411</td>
<td>2 Channel, DC 24 V, 0.2 ms, 2- to 4-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-418, 753-418</td>
<td>2 Channel, DC 24 V, 3.0 ms, 2- to 3-conductor connection; high-side switching; diagnostics and confirmation</td>
</tr>
<tr>
<td>750-419</td>
<td>2 Channel, DC 24 V, 3.0 ms, 2- to 3-conductor connection; high-side switching; diagnostics</td>
</tr>
<tr>
<td>750-421, 753-421</td>
<td>2 Channel, DC 24 V, 3.0 ms, 2- to 3-conductor connection; high-side switching; diagnostics</td>
</tr>
<tr>
<td>750-402, 753-402</td>
<td>4 Channel, DC 24 V, 3.0 ms, 2- to 3-conductor connection; high-side switching</td>
</tr>
<tr>
<td>SKU</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>750-432, 753-432</td>
<td>4 Channel, DC 24 V, 3.0 ms, 2-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-403, 753-403</td>
<td>4 Channel, DC 24 V, 0.2 ms, 2- to 3-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-433, 753-433</td>
<td>4 Channel, DC 24 V, 0.2 ms, 2-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-422, 753-422</td>
<td>4 Channel, DC 24 V, 2- to 3-conductor connection; high-side switching; 10 ms pulse extension</td>
</tr>
<tr>
<td>750-408, 753-408</td>
<td>4 Channel, DC 24 V, 3.0 ms, 2- to 3-conductor connection; low-side switching</td>
</tr>
<tr>
<td>750-409, 753-409</td>
<td>4 Channel, DC 24 V, 0.2 ms, 2- to 3-conductor connection; low-side switching</td>
</tr>
<tr>
<td>750-430, 753-430</td>
<td>8 Channel, DC 24 V, 3.0 ms, 1-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-431, 753-431</td>
<td>8 Channel, DC 24 V, 0.2 ms, 1-conductor connection; high-side switching</td>
</tr>
<tr>
<td>750-436</td>
<td>8 Channel, DC 24 V, 3.0 ms, 1-conductor connection; low-side switching</td>
</tr>
<tr>
<td>750-437</td>
<td>8 Channel, DC 24 V, 0.2 ms, 1-conductor connection; low-side switching</td>
</tr>
<tr>
<td>DI AC/DC 24 V</td>
<td></td>
</tr>
<tr>
<td>750-415, 753-415</td>
<td>4 Channel, AC/DC 24 V, 2-conductor connection</td>
</tr>
<tr>
<td>750-423, 753-423</td>
<td>4 Channel, AC/DC 24 V, 2- to 3-conductor connection; with power jumper contacts</td>
</tr>
<tr>
<td>DI AC/DC 42 V</td>
<td></td>
</tr>
<tr>
<td>750-428, 753-428</td>
<td>4 Channel, AC/DC 42 V, 2-conductor connection</td>
</tr>
<tr>
<td>DI DC 48 V</td>
<td></td>
</tr>
<tr>
<td>750-412, 753-412</td>
<td>2 Channel, DC 48 V, 3.0 ms, 2- to 4-conductor connection; high-side switching</td>
</tr>
<tr>
<td>DI DC 110 V</td>
<td></td>
</tr>
<tr>
<td>750-427, 753-427</td>
<td>2 Channel, DC 110 V, configurable high-side or low-side switching</td>
</tr>
<tr>
<td>DI AC 120 V</td>
<td></td>
</tr>
<tr>
<td>750-406, 753-406</td>
<td>2 Channel, AC 120 V, 2- to 4-conductor connection; high-side switching</td>
</tr>
<tr>
<td>DI AC 120(230) V</td>
<td></td>
</tr>
<tr>
<td>750-405, 753-405</td>
<td>2 Channel, AC 230 V, 2- to 4-conductor connection; high-side switching</td>
</tr>
</tbody>
</table>
### DI NAMUR

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-435</td>
<td>1 Channel, NAMUR EEx i, proximity switch acc. to DIN EN 50227</td>
</tr>
<tr>
<td>750-425, 753-425</td>
<td>2 Channel, NAMUR, proximity switch acc. to DIN EN 50227</td>
</tr>
<tr>
<td>750-438</td>
<td>2 Channel, NAMUR EEx i, proximity switch acc. to DIN EN 50227</td>
</tr>
</tbody>
</table>

### DI Intruder Detection

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-424, 753-424</td>
<td>2 Channel, DC 24 V, intruder detection</td>
</tr>
</tbody>
</table>

#### 7.1.2 Digital Output Modules

Tab. 7-2: Digital output modules

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DO DC 5 V</strong></td>
<td></td>
</tr>
<tr>
<td>750-519</td>
<td>4 Channel, DC 5 V, 20mA, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td><strong>DO DC 12(14) V</strong></td>
<td></td>
</tr>
<tr>
<td>753-534</td>
<td>8 Channel, DC 12(14) V, 1A, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td><strong>DO DC 24 V</strong></td>
<td></td>
</tr>
<tr>
<td>750-501, 753-501</td>
<td>2 Channel, DC 24 V, 0.5 A, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td>750-502, 753-502</td>
<td>2 Channel, DC 24 V, 2.0 A, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td>750-506, 753-506</td>
<td>2 Channel, DC 24 V, 0.5 A, short-circuit-protected; high-side switching; diagnostics</td>
</tr>
<tr>
<td>750-507, 753-507</td>
<td>2 Channel, DC 24 V, 2.0 A, short-circuit-protected; high-side switching; diagnostics; no longer available, replaced by 750-508!</td>
</tr>
<tr>
<td>750-508</td>
<td>2 Channel, DC 24 V, 2.0 A, short-circuit-protected; high-side switching; diagnostics; replacement for 750-507</td>
</tr>
<tr>
<td>750-535</td>
<td>2 Channel, DC 24 V, EEx i, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td>750-504, 753-504</td>
<td>4 Channel, DC 24 V, 0.5 A, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td>750-531, 753-531</td>
<td>4 Channel, DC 24 V, 0.5 A, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td>750-532</td>
<td>4 Channel, DC 24 V, 0.5 A, short-circuit-protected; high-side switching; diagnostics</td>
</tr>
<tr>
<td>750-516, 753-516</td>
<td>4 Channel, DC 24 V, 0.5 A, short-circuit-protected; low-side switching</td>
</tr>
<tr>
<td>750-530, 753-530</td>
<td>8 Channel, DC 24 V, 0.5 A, short-circuit-protected; high-side switching</td>
</tr>
<tr>
<td>750-537</td>
<td>8 Channel, DC 24 V, 0.5 A, short-circuit-protected; high-side switching; diagnostics</td>
</tr>
<tr>
<td>750-536</td>
<td>8 Channel, DC 24 V, 0.5 A, short-circuit-protected; low-side switching</td>
</tr>
<tr>
<td><strong>DO AC 120(230) V</strong></td>
<td></td>
</tr>
<tr>
<td>753-540</td>
<td>4 Channel, AC 120(230) V, 0.25 A, short-circuit-protected; high-side switching</td>
</tr>
</tbody>
</table>
### 7.1.3 Analog Input Modules

Tab. 7-3: Analog input modules

<table>
<thead>
<tr>
<th>AI 0 - 20 mA</th>
<th>2 Channel, 0 - 20 mA, differential input</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-452, 753-452</td>
<td></td>
</tr>
<tr>
<td>750-465, 753-465</td>
<td>2 Channel, 0 - 20 mA, single-ended</td>
</tr>
<tr>
<td>750-472, 753-472</td>
<td>2-Channel, 0 - 20 mA, 16 bit, single-ended</td>
</tr>
<tr>
<td>750-480</td>
<td>2-Channel, 0 - 20 mA, differential input</td>
</tr>
<tr>
<td>750-453, 753-453</td>
<td>4 Channel, 0 - 20 mA, single-ended</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AI 4 - 20 mA</th>
<th>2 Channel, 4 - 20 mA, differential input</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-454, 753-454</td>
<td></td>
</tr>
<tr>
<td>750-474, 753-474</td>
<td>2 Channel, 4 - 20 mA, 16 bit, single-ended</td>
</tr>
<tr>
<td>750-466, 753-466</td>
<td>2 Channel, 4 - 20 mA, single ended</td>
</tr>
<tr>
<td>750-485</td>
<td>2 Channel, 4 - 20 mA, EEx i, single-ended</td>
</tr>
<tr>
<td>750-492, 753-492</td>
<td>2 Channel, 4 - 20 mA, isolated differential input</td>
</tr>
<tr>
<td>750-455, 753-455</td>
<td>4 Channel, 4 - 20 mA, single-ended</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AI 0 - 1 A</th>
<th>2-Channel, 0 - 1 A AC/DC, differential input</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-475, 753-475</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AI 0 - 5 A</th>
<th>2-Channel, 0 - 5 A AC/DC, differential input</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-475/020-000, 753-475/020-000</td>
<td></td>
</tr>
</tbody>
</table>
### AI 0 - 10 V

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-467, 753-467</td>
<td>2 Channel, DC 0 - 10 V, single-ended</td>
</tr>
<tr>
<td>750-477, 753-477</td>
<td>2 Channel, AC/DC 0 - 10 V, differential input</td>
</tr>
<tr>
<td>750-478, 753-478</td>
<td>2 Channel, DC 0 - 10 V, single-ended</td>
</tr>
<tr>
<td>750-459, 753-459</td>
<td>4 Channel, DC 0 - 10 V, single-ended</td>
</tr>
<tr>
<td>750-468</td>
<td>4 Channel, DC 0 - 10 V, single-ended</td>
</tr>
</tbody>
</table>

### AI DC ± 10 V

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-456, 753-456</td>
<td>2 Channel, DC ± 10 V, differential input</td>
</tr>
<tr>
<td>750-479, 753-479</td>
<td>2 Channel, DC ± 10 V, differential measurement input</td>
</tr>
<tr>
<td>750-476, 753-476</td>
<td>2 Channel, DC ± 10 V, single-ended</td>
</tr>
<tr>
<td>750-457, 753-457</td>
<td>4 Channel, DC ± 10 V, single-ended</td>
</tr>
</tbody>
</table>

### AI DC 0 - 30 V

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-483, 753-483</td>
<td>2 Channel, DC 0 - 30 V, differential measurement input</td>
</tr>
</tbody>
</table>

### AI Resistance Sensors

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-461, 753-461</td>
<td>2 Channel, resistance sensors, PT100 / RTD</td>
</tr>
<tr>
<td>750-481/003-000</td>
<td>2 Channel, resistance sensors, PT100 / RTD, EEx i</td>
</tr>
<tr>
<td>750-460</td>
<td>4 Channel, resistance sensors, PT100 / RTD</td>
</tr>
</tbody>
</table>

### AI Thermocouples

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-462</td>
<td>2 Channel, thermocouples, line break detection, sensor types: J, K, B, E, N, R, S, T, U</td>
</tr>
<tr>
<td>750-469, 753-469</td>
<td>2 Channel, thermocouples, line break detection, sensor types: J, K, B, E, N, R, S, T, U, L</td>
</tr>
</tbody>
</table>

### AI Others

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-491</td>
<td>1 Channel for resistor bridges (strain gauge)</td>
</tr>
</tbody>
</table>
### 7.1.4 Analog Output Modules

Tab. 7-4: Analog output modules

<table>
<thead>
<tr>
<th>AO 0 - 20 mA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-552, 753-552</td>
<td>2 Channel, 0 - 20 mA</td>
</tr>
<tr>
<td>750-585</td>
<td>2 Channel, 0 - 20 mA, Ex i</td>
</tr>
<tr>
<td>750-553, 753-553</td>
<td>4 Channel, 0 - 20 mA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AO 4 - 20 mA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-554, 753-554</td>
<td>2 Channel, 4 - 20 mA</td>
</tr>
<tr>
<td>750-554, 753-554</td>
<td>4 Channel, 4 - 20 mA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AO DC 0 - 10 V</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-550, 753-550</td>
<td>2 Channel, DC 0 - 10 V</td>
</tr>
<tr>
<td>750-560</td>
<td>2 Channel, DC 0 - 10 V, 10 bit, 100 mW, 24 V</td>
</tr>
<tr>
<td>750-559, 753-559</td>
<td>4 Channel, DC 0 - 10 V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AO DC ± 10 V</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-556, 753-556</td>
<td>2 Channel, DC ± 10 V</td>
</tr>
<tr>
<td>750-557, 753-557</td>
<td>4 Channel, DC ± 10 V</td>
</tr>
</tbody>
</table>
## 7.1.5 Special Modules

Tab. 7-5: Special modules

<table>
<thead>
<tr>
<th>Counter Modules</th>
<th>Up / down counter, DC 24 V, 100 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-404, 753-404</td>
<td></td>
</tr>
<tr>
<td>750-638, 753-638</td>
<td>2 Channel, up / down counter, DC 24 V/ 16 bit / 500 Hz</td>
</tr>
<tr>
<td>Frequency Measuring</td>
<td>Frequency measuring</td>
</tr>
<tr>
<td>750-404/000-003, 753-400/000-003</td>
<td></td>
</tr>
<tr>
<td>Pulse Width Module</td>
<td>2-channel pulse width module, DC 24 V, short-circuit-protected, high-side switching</td>
</tr>
<tr>
<td>750-511</td>
<td></td>
</tr>
<tr>
<td>Distance and Angle Measurement Modules</td>
<td></td>
</tr>
<tr>
<td>750-630</td>
<td>SSI transmitter interface</td>
</tr>
<tr>
<td>750-631</td>
<td>Incremental encoder interface, differential inputs</td>
</tr>
<tr>
<td>750-634</td>
<td>Incremental encoder interface, DC 24 V</td>
</tr>
<tr>
<td>750-637</td>
<td>Incremental encoder interface RS 422, cam outputs</td>
</tr>
<tr>
<td>750-635, 753-635</td>
<td>Digital pulse interface, for magnetostrictive distance sensors</td>
</tr>
<tr>
<td>Serial Interfaces</td>
<td></td>
</tr>
<tr>
<td>750-650, 753</td>
<td>Serial interface RS 232 C</td>
</tr>
<tr>
<td>750-653, 753</td>
<td>Serial interface RS 485</td>
</tr>
<tr>
<td>750-651</td>
<td>TTY-Serial interface, 20 mA Current Loop</td>
</tr>
<tr>
<td>750-654</td>
<td>Data exchange module</td>
</tr>
<tr>
<td>DALI / DSI Master Module</td>
<td></td>
</tr>
<tr>
<td>750-641</td>
<td>DALI / DSI master module</td>
</tr>
<tr>
<td>AS Interface Master Module</td>
<td></td>
</tr>
<tr>
<td>750-655</td>
<td>AS interface master module</td>
</tr>
<tr>
<td>Radio Receiver Module</td>
<td></td>
</tr>
<tr>
<td>750-642</td>
<td>Radio receiver EnOcean</td>
</tr>
<tr>
<td>MP Bus Master Module</td>
<td></td>
</tr>
<tr>
<td>750-643</td>
<td>MP bus (multi point bus) master module</td>
</tr>
<tr>
<td>Vibration Monitoring</td>
<td></td>
</tr>
<tr>
<td>750-645</td>
<td>2 Channel vibration velocity / bearing condition monitoring VIB I/O</td>
</tr>
</tbody>
</table>
## I/O Modules

### Overview

<table>
<thead>
<tr>
<th>PROFIsafe Modules</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-660/000-001</td>
<td>8FDI 24V DC PROFIsafe; PROFIsafe 8 channel digital input module</td>
</tr>
<tr>
<td>750-665/000-001</td>
<td>4FDO 0.5A / 4FDI 24V DC PROFIsafe; PROFIsafe 4 channel digital input and output module</td>
</tr>
<tr>
<td>750-666/000-001</td>
<td>1FDO 10A / 2FDO 0.5A / 2FDI 24V PROFIsafe; PROFIsafe power switch module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RTC Module</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-640</td>
<td>RTC module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>KNX / EIB TP1 Module</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>750-646</td>
<td>KNX / EIB /TP1 module – device mode / router mode</td>
</tr>
</tbody>
</table>
## 7.1.6 System Modules

Tab. 7-6: System modules

<table>
<thead>
<tr>
<th>Module Bus Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-627</td>
<td>Module bus extension, end module</td>
</tr>
<tr>
<td>750-628</td>
<td>Module bus extension, coupler module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DC 24 V Power Supply Modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-602</td>
<td>DC 24 V, passive</td>
</tr>
<tr>
<td>750-601</td>
<td>DC 24 V, max. 6.3 A, without diagnostics, with fuse-holder</td>
</tr>
<tr>
<td>750-610</td>
<td>DC 24 V, max. 6.3 A, with diagnostics, with fuse-holder</td>
</tr>
<tr>
<td>750-625</td>
<td>DC 24 V, EEx i, with fuse-holder</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DC 24 V Power Supply Modules with bus power supply</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-613</td>
<td>Bus power supply, 24 V DC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AC 120 V Power Supply Modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-615</td>
<td>AC 120 V, max. 6.3 A without diagnostics, with fuse-holder</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AC 230 V Power Supply Modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-612</td>
<td>AC/DC 230 V without diagnostics, passive</td>
</tr>
<tr>
<td>750-609</td>
<td>AC 230 V, max. 6.3 A without diagnostics, with fuse-holder</td>
</tr>
<tr>
<td>750-611</td>
<td>AC 230 V, max. 6.3 A with diagnostics, with fuse-holder</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filter Modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-624</td>
<td>Filter module, field side power supply</td>
</tr>
<tr>
<td>750-626</td>
<td>Filter module, system and field side power supply</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Side Connection Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-603, 753-603</td>
<td>Field side connection module, DC 24 V</td>
</tr>
<tr>
<td>750-604, 753-604</td>
<td>Field side connection module, DC 0 V</td>
</tr>
<tr>
<td>750-614, 753-614</td>
<td>Field side connection module, AC/DC 0 ... 230 V</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Separation Modules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-616</td>
<td>Separation module</td>
</tr>
<tr>
<td>750-621</td>
<td>Separation module with power contacts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Binary Spacer Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-622</td>
<td>Binary spacer module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>End Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>750-600</td>
<td>End module, to loop the internal bus</td>
</tr>
</tbody>
</table>
7.2 Process Data Architecture for MODBUS/TCP

With some I/O modules, the structure of the process data is fieldbus specific.

In the case of a coupler/controller with MODBUS/TCP, the process image uses a word structure (with word alignment). The internal mapping method for data greater than one byte conforms to the Intel format.

The following section describes the process image for various WAGO-I/O-SYSTEM 750 and 753 I/O modules when using a coupler/controller with MODBUS/TCP.

Note
Depending on the specific position of an I/O module in the fieldbus node, the process data of all previous byte or bit-oriented modules must be taken into account to determine its location in the process data map.

For the PFC process image of the programmable fieldbus controller is the the structure of the process data mapping identical.

7.2.1 Digital Input Modules

Digital input modules supply one bit of data per channel to specify the signal state for the corresponding channel. These bits are mapped into the Input Process Image.

When analog input modules are also present in the node, the digital data is always appended after the analog data in the Input Process Image, grouped into bytes.

Some digital modules have an additional diagnostic bit per channel in the Input Process Image. The diagnostic bit is used for detecting faults that occur (e.g., wire breaks and/or short circuits).

1 Channel Digital Input Module with Diagnostics

750-435

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diagnostic bit S 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data bit DI 1</td>
</tr>
</tbody>
</table>
2 Channel Digital Input Modules

750-400, -401, -405, -406, -410, -411, -412, -427, -438, (and all variations), 753-400, -401, -405, -406, -410, -411, -412, -427

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data bit DI 2 Channel 2</td>
<td>Data bit DI 1 Channel 1</td>
<td></td>
</tr>
</tbody>
</table>

2 Channel Digital Input Modules with Diagnostics

750-419, -421, -424, -425, 753-421, -424, -425

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diagnostic bit S 2 Channel 2</td>
<td>Diagnostic bit S 1 Channel 1</td>
<td>Data bit DI 2 Channel 2</td>
<td>Data bit DI 1 Channel 1</td>
</tr>
</tbody>
</table>

2 Channel Digital Input Module with Diagnostics and Output Process Data

750-418, 753-418

The 750-418, 753-418 digital input module supplies a diagnostic and acknowledge bit for each input channel. If a fault condition occurs, the diagnostic bit is set. After the fault condition is cleared, an acknowledge bit must be set to re-activate the input. The diagnostic data and input data bit is mapped in the Input Process Image, while the acknowledge bit is in the Output Process Image.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Diagnostic bit S 2 Channel 2</td>
<td>Diagnostic bit S 1 Channel 1</td>
<td>Data bit DI 2 Channel 2</td>
<td>Data bit DI 1 Channel 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Acknowledge bit Q 2 Channel 2</td>
<td>Acknowledge bit Q 1 Channel 1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

4 Channel Digital Input Modules
750-402, -403, -408, -409, -414, -415, -422, -423, -428, -432, -433,
753-402, -403, -408, -409, -415, -422, -423, -428, -432, -433, -440

7.2.2 Digital Output Modules

Digital output modules use one bit of data per channel to control the output
of the corresponding channel. These bits are mapped into the Output
Process Image.

When analog output modules are also present in the node, the digital image
data is always appended after the analog data in the Output Process Image,
grouped into bytes.

1 Channel Digital Output Module with Input Process Data

750-523

2 Channel Digital Output Modules

750-501, -502, -509, -512, -513, -514, -517, -535, (and all variations),
753-501, -502, -509, -512, -513, -514, -517
2 Channel Digital Input Modules with Diagnostics and Input Process Data

750-507 (-508), -522, 753-507

The 750-507 (-508), -522 and 753-507 digital output modules have a diagnostic bit for each output channel. When an output fault condition occurs (i.e., overload, short circuit, or broken wire), a diagnostic bit is set. The diagnostic data is mapped into the Input Process Image, while the output control bits are in the Output Process Image.

750-506, 753-506

The 750-506, 753-506 digital output module has 2-bits of diagnostic information for each output channel. The 2-bit diagnostic information can then be decoded to determine the exact fault condition of the module (i.e., overload, a short circuit, or a broken wire). The 4-bits of diagnostic data are mapped into the Input Process Image, while the output control bits are in the Output Process Image.
### Process Data Architecture for MODBUS/TCP

#### 4 Channel Digital Output Modules

**750-504, -516, -519, -531, 753-504, -516, -531, -540**

#### 4 Channel Digital Output Modules with Diagnostics and Input Process Data

**750-532**

The 750-532 digital output modules have a diagnostic bit for each output channel. When an output fault condition occurs (i.e., overload, short circuit, or broken wire), a diagnostic bit is set. The diagnostic data is mapped into the Input Process Image, while the output control bits are in the Output Process Image.
### 8 Channel Digital Output Module

**750-530, -536, 753-530, -434**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>controls DO 8 Channel 8</td>
<td>controls DO 7 Channel 7</td>
<td>controls DO 6 Channel 6</td>
<td>controls DO 5 Channel 5</td>
<td>controls DO 4 Channel 4</td>
<td>controls DO 3 Channel 3</td>
<td>controls DO 2 Channel 2</td>
<td>controls DO 1 Channel 1</td>
</tr>
</tbody>
</table>

### 8 Channel Digital Output Modules with Diagnostics and Input Process Data

**750-537**

The 750-537 digital output modules have a diagnostic bit for each output channel. When an output fault condition occurs (i.e., overload, short circuit, or broken wire), a diagnostic bit is set. The diagnostic data is mapped into the Input Process Image, while the output control bits are in the Output Process Image.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnost ic bit S 7 Channel 8</td>
<td>Diagnost ic bit S 6 Channel 7</td>
<td>Diagnost ic bit S 5 Channel 6</td>
<td>Diagnost ic bit S 4 Channel 5</td>
<td>Diagnost ic bit S 3 Channel 4</td>
<td>Diagnost ic bit S 2 Channel 3</td>
<td>Diagnost ic bit S 1 Channel 2</td>
<td>Diagnost ic bit S 0 Channel 1</td>
</tr>
</tbody>
</table>

Diagnostic bit S = '0' no Error
Diagnostic bit S = '1' overload, short circuit, or broken wire
7.2.3 Analog Input Modules

The hardware of an analog input module has 16 bits of measured analog data per channel and 8 bits of control/status. However, the coupler/controller with MODBUS/TCP does not have access to the 8 control/status bits. Therefore, the coupler/controller with MODBUS/TCP can only access the 16 bits of analog data per channel, which are grouped as words and mapped in Intel format in the Input Process Image.

When digital input modules are also present in the node, the analog input data is always mapped into the Input Process Image in front of the digital data.

1 Channel Analog Input Module

750-491, (and all variations)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D1 D0</td>
<td>Measured Value U₀</td>
</tr>
<tr>
<td>1</td>
<td>D3 D2</td>
<td>Measured Value Uᵣ₋₀</td>
</tr>
</tbody>
</table>

2 Channel Analog Input Modules


<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D1 D0</td>
<td>Measured Value Channel 1</td>
</tr>
<tr>
<td>1</td>
<td>D3 D2</td>
<td>Measured Value Channel 2</td>
</tr>
</tbody>
</table>
4 Channel Analog Input Modules

750-453, -455, -457, -459, -460, -468, (and all variations),
753-453, -455, -457, -459

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Byte</td>
<td>Low Byte</td>
</tr>
<tr>
<td>0</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>2</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>3</td>
<td>D7</td>
<td>D6</td>
</tr>
</tbody>
</table>

7.2.4 Analog Output Modules

The hardware of an analog output module has 16 bits of measured analog data per channel and 8 bits of control/status. However, the coupler/controller with MODBUS/TCP does not have access to the 8 control/status bits. Therefore, the coupler/controller with MODBUS/TCP can only access the 16 bits of analog data per channel, which are grouped as words and mapped in Intel format in the Output Process Image.

When digital output modules are also present in the node, the analog output data is always mapped into the Output Process Image in front of the digital data.

2 Channel Analog Output Modules

750-550, -552, -554, -556, -560, -585, (and all variations),
753-550, -552, -554, -556

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Byte</td>
<td>Low Byte</td>
</tr>
<tr>
<td>0</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>1</td>
<td>D3</td>
<td>D2</td>
</tr>
</tbody>
</table>
### 4 Channel Analog Output Modules

750-553, -555, -557, -559, 753-553, -555, -557, -559

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D1 D0</td>
<td>Output Value Channel 1</td>
</tr>
<tr>
<td>1</td>
<td>D3 D2</td>
<td>Output Value Channel 2</td>
</tr>
<tr>
<td>2</td>
<td>D5 D4</td>
<td>Output Value Channel 3</td>
</tr>
<tr>
<td>3</td>
<td>D7 D6</td>
<td>Output Value Channel 4</td>
</tr>
</tbody>
</table>

### 7.2.5 Specialty Modules

WAGO has a host of Specialty I/O modules that perform various functions. With individual modules beside the data bytes also the control/status byte is mapped in the process image. The control/status byte is required for the bi-directional data exchange of the module with the higher-ranking control system. The control byte is transmitted from the control system to the module and the status byte from the module to the control system. This allows, for example, setting of a counter with the control byte or displaying of overshooting or undershooting of the range with the status byte.

**Further information**

For detailed information about the structure of a particular module’s control/status byte, please refer to that module’s manual. Manuals for each module can be found on the Internet under: http://www.wago.com.

### Counter Modules

750-404, (and all variations except of /000-005),
753-404, (and variation /000-003)

The above Counter Modules have a total of 5 bytes of user data in both the Input and Output Process Image (4 bytes of counter data and 1 byte of control/status). The counter value is supplied as 32 bits. The following tables illustrate the Input and Output Process Image, which has a total of 3 words mapped into each image. Word alignment is applied.
The above Counter Modules have a total of 5 bytes of user data in both the Input and Output Process Image (4 bytes of counter data and 1 byte of control/status). The two counter values are supplied as 16 bits. The following tables illustrate the Input and Output Process Image, which has a total of 3 words mapped into each image. Word alignment is applied.

### Input Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Status byte</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Counter Value</td>
</tr>
<tr>
<td>2</td>
<td>D3 D2</td>
<td></td>
</tr>
</tbody>
</table>

### Output Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Control byte</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Counter Setting Value</td>
</tr>
<tr>
<td>2</td>
<td>D3 D2</td>
<td></td>
</tr>
</tbody>
</table>

750-638, 753-638
The above Counter Modules have a total of 6 bytes of user data in both the Input and Output Process Image (4 bytes of counter data and 2 bytes of control/status). The two counter values are supplied as 16 bits. The following tables illustrate the Input and Output Process Image, which has a total of 4 words mapped into each image. Word alignment is applied.

### Input Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>S0</td>
</tr>
<tr>
<td>1</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>S1</td>
</tr>
<tr>
<td>3</td>
<td>D3</td>
<td>D2</td>
</tr>
</tbody>
</table>

### Output Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>C0</td>
</tr>
<tr>
<td>1</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>C1</td>
</tr>
<tr>
<td>3</td>
<td>D3</td>
<td>D2</td>
</tr>
</tbody>
</table>

Pulse Width Modules

750-511, (and all variations)

The above Pulse Width modules have a total of 6 bytes of user data in both the Input and Output Process Image (4 bytes of channel data and 2 bytes of control/status). The two channel values are supplied as 16 bits. Each channel has its own control/status byte. The following table illustrates the Input and Output Process Image, which has a total of 4 words mapped into each image. Word alignment is applied.
### Serial Interface Modules with alternative Data Format

750-650, (and the variations /000-002, -004, -006, -009, -010, -011, -012, -013)

750-651, (and the variations /000-002, -003)

750-653, (and the variations /000-002, -007)

#### Note:

With the freely parametrizable variations /003 000 of the serial interface modules, the desired operation mode can be set. Dependent on it, the process image of these modules is then the same, as from the appropriate variation.

The above Serial Interface Modules with alternative data format have a total of 4 bytes of user data in both the Input and Output Process Image (3 bytes of serial data and 1 byte of control/status). The following table illustrates the Input and Output Process Image, which have a total of 2 words mapped into each image. Word alignment is applied.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>C0/S0</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Data Value of Channel 1</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>C1/S1</td>
</tr>
<tr>
<td>3</td>
<td>D3 D2</td>
<td>Data Value of Channel 2</td>
</tr>
</tbody>
</table>

### Input and Output Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D0 C/S</td>
<td>Data byte</td>
</tr>
<tr>
<td>1</td>
<td>D2 D1</td>
<td>Data bytes</td>
</tr>
</tbody>
</table>
Serial Interface Modules with Standard Data Format

750-650/000-001, -014, -015, -016
750-651/000-001
750-653/000-001, -006

The above Serial Interface Modules with Standard Data Format have a total of 6 bytes of user data in both the Input and Output Process Image (5 bytes of serial data and 1 byte of control/status). The following table illustrates the Input and Output Process Image, which have a total of 3 words mapped into each image. Word alignment is applied.

<table>
<thead>
<tr>
<th>Offset</th>
<th>High Byte</th>
<th>Low Byte</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D0</td>
<td>C/S</td>
<td>Data byte</td>
</tr>
<tr>
<td>1</td>
<td>D2</td>
<td>D1</td>
<td>Control/Status byte</td>
</tr>
<tr>
<td>2</td>
<td>D4</td>
<td>D3</td>
<td>Data bytes</td>
</tr>
</tbody>
</table>

Data Exchange Module

750-654, (and the variation /000-001)

The Data Exchange modules have a total of 4 bytes of user data in both the Input and Output Process Image. The following tables illustrate the Input and Output Process Image, which has a total of 2 words mapped into each image. Word alignment is applied.

<table>
<thead>
<tr>
<th>Offset</th>
<th>High Byte</th>
<th>Low Byte</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D1</td>
<td>D0</td>
<td>Data bytes</td>
</tr>
<tr>
<td>1</td>
<td>D3</td>
<td>D2</td>
<td></td>
</tr>
</tbody>
</table>
SSI Transmitter Interface Modules
750-630, (and all variations)

The above SSI Transmitter Interface modules have a total of 4 bytes of user data in the Input Process Image, which has 2 words mapped into the image. Word alignment is applied.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D1 D0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>D3 D2</td>
<td></td>
</tr>
</tbody>
</table>

Data bytes

Incremental Encoder Interface Modules
750-631

The above Incremental Encoder Interface modules have 5 bytes of input data and 3 bytes of output data. The following tables illustrate the Input and Output Process Image, which have 4 words into each image. Word alignment is applied.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>- S</td>
<td>Status byte</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Counter word</td>
</tr>
<tr>
<td>2</td>
<td>- -</td>
<td>not used</td>
</tr>
<tr>
<td>3</td>
<td>D4 D3</td>
<td>Latch word</td>
</tr>
</tbody>
</table>

Input Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>- C</td>
<td>Control byte</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Counter Setting word</td>
</tr>
<tr>
<td>2</td>
<td>- -</td>
<td>not used</td>
</tr>
<tr>
<td>3</td>
<td>- -</td>
<td>not used</td>
</tr>
</tbody>
</table>

Output Process Image
The above Incremental Encoder Interface module has 5 bytes of input data (6 bytes in cycle duration measurement mode) and 3 bytes of output data. The following tables illustrate the Input and Output Process Image, which has 4 words mapped into each image. Word alignment is applied.

### Input Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>Status byte</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Counter word</td>
</tr>
<tr>
<td>2</td>
<td>(D2)*</td>
<td>(Periodic time)</td>
</tr>
<tr>
<td>3</td>
<td>D4 D3</td>
<td>Latch word</td>
</tr>
</tbody>
</table>

*) If cycle duration measurement mode is enabled in the control byte, the cycle duration is given as a 24-bit value that is stored in D2 together with D3/D4.

### Output Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>Control byte</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Counter Setting word</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>not used</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>not used</td>
</tr>
</tbody>
</table>

### Input and Output Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>Control/Status byte of Channel 1</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Data Value of Channel 1</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Control/Status byte of Channel 2</td>
</tr>
<tr>
<td>3</td>
<td>D3 D2</td>
<td>Data Value of Channel 2</td>
</tr>
</tbody>
</table>
The above Digital Pulse Interface module has a total of 4 bytes of user data in both the Input and Output Process Image (3 bytes of module data and 1 byte of control/status). The following table illustrates the Input and Output Process Image, which have 2 words mapped into each image. Word alignment is applied.

### RTC Module

750-640

The RTC Module module has a total of 6 bytes of user data in both the Input and Output Process Image (4 bytes of module data and 1 byte of control/status and 1 byte ID for command). The following table illustrates the Input and Output Process Image, which have 3 words mapped into each image. Word alignment is applied.

### DALI/DSI Master Module

750-641

The DALI/DSI Master module has a total of 6 bytes of user data in both the Input and Output Process Image (5 bytes of module data and 1 byte of control/status). The following tables illustrate the Input and Output Process Image, which have 3 words mapped into each image. Word alignment is applied.
### Input Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Byte</td>
<td>Low Byte</td>
</tr>
<tr>
<td>0</td>
<td>D0</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>3</td>
<td>D4</td>
<td>D3</td>
</tr>
</tbody>
</table>

### Output Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Byte</td>
<td>Low Byte</td>
</tr>
<tr>
<td>0</td>
<td>D0</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>3</td>
<td>D4</td>
<td>D3</td>
</tr>
</tbody>
</table>

### EnOcean Radio Receiver

750-642

The EnOcean radio receiver has a total of 4 bytes of user data in both the Input and Output Process Image (3 bytes of module data and 1 byte of control/status). The following tables illustrate the Input and Output Process Image, which have 2 words mapped into each image. Word alignment is applied.

### Input Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Byte</td>
<td>Low Byte</td>
</tr>
<tr>
<td>0</td>
<td>D0</td>
<td>S</td>
</tr>
<tr>
<td>1</td>
<td>D2</td>
<td>D1</td>
</tr>
</tbody>
</table>

### Output Process Image

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High Byte</td>
<td>Low Byte</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>C</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
# MP Bus Master Module

**750-643**

The MP Bus Master Module has a total of 8 bytes of user data in both the Input and Output Process Image (6 bytes of module data and 2 bytes of control/status). The following table illustrates the Input and Output Process Image, which have 4 words mapped into each image. Word alignment is applied.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>C1/S1 C0/S0</td>
<td>extended Control/Status byte</td>
</tr>
<tr>
<td>1</td>
<td>D1 D0</td>
<td>Control/Status byte</td>
</tr>
<tr>
<td>2</td>
<td>D3 D2</td>
<td>Data bytes</td>
</tr>
<tr>
<td>3</td>
<td>D5 D4</td>
<td></td>
</tr>
</tbody>
</table>
Vibration Velocity/Bearing Condition Monitoring VIB I/O

750-645

The Vibration Velocity/Bearing Condition Monitoring VIB I/O has a total of 12 bytes of user data in both the Input and Output Process Image (8 bytes of module data and 4 bytes of control/status). The following table illustrates the Input and Output Process Image, which have 8 words mapped into each image. Word alignment is applied.

<table>
<thead>
<tr>
<th>Offset</th>
<th>byte Destination</th>
<th>Remark</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>C0/S0</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control/Status byte (log. Channel 1, Sensor input 1)</td>
</tr>
<tr>
<td>1</td>
<td>D1</td>
<td>D0</td>
<td>Data bytes (log. Channel 1, Sensor input 1)</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>C1/S1</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control/Status byte (log. Channel 2 Sensor input 2)</td>
</tr>
<tr>
<td>3</td>
<td>D3</td>
<td>D2</td>
<td>Data bytes (log. Channel 2 Sensor input 2)</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>C2/S2</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control/Status byte (log. Channel 3 Sensor input 1)</td>
</tr>
<tr>
<td>5</td>
<td>D5</td>
<td>D4</td>
<td>Data bytes (log. Channel 3 Sensor input 1)</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>C3/S3</td>
<td>Not used</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Control/Status byte (log. Channel 4 Sensor input 2)</td>
</tr>
<tr>
<td>7</td>
<td>D7</td>
<td>D6</td>
<td>Data bytes (log. Channel 4 Sensor input 2)</td>
</tr>
</tbody>
</table>
AS-interface Master Module

750-655

The length of the process image of the AS-interface master module can be set to fixed sizes of 12, 20, 24, 32, 40 or 48 bytes. It consists of a control or status byte, a mailbox with a size of 0, 6, 10, 12 or 18 bytes and the AS-interface process data, which can range from 0 to 32 bytes.

The AS-interface master module has a total of 6 to maximally 24 words data in both the Input and Output Process Image. Word alignment is applied.

The first Input and output word, which is assigned to an AS-interface master module, contains the status / control byte and one empty byte. Subsequently the mailbox data are mapped, when the mailbox is permanently superimposed (Mode 1).

In the operating mode with suppressable mailbox (Mode 2), the mailbox and the cyclical process data are mapped next. The following words contain the remaining process data.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Byte Destination</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>C0/S0, not used</td>
</tr>
<tr>
<td>1</td>
<td>D1</td>
<td>D0, Mailbox (0, 3, 5, 6 or 9 words) / Process data (0-16 words)</td>
</tr>
<tr>
<td>2</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>3</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>max. 23</td>
<td>D45</td>
<td>D44</td>
</tr>
</tbody>
</table>
7.2.6 System Modules

System Modules with Diagnostics

750-610, -611

The 750-610 and 750-611 Supply Modules provide 2 bits of diagnostics in the Input Process Image for monitoring of the internal power supply.

<table>
<thead>
<tr>
<th>Input Process Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Binary Space Module

750-622

The Binary Space Modules 750-622 behave alternatively like 2 channel digital input modules or output modules and seize depending upon the selected settings 1, 2, 3 or 4 bits per channel. According to this, 2, 4, 6 or 8 bits are occupied then either in the process input or the process output image.

<table>
<thead>
<tr>
<th>Input or Output Process Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 7</td>
</tr>
<tr>
<td>(DI 8)</td>
</tr>
</tbody>
</table>
8 Application Examples

The example programs in this manual serve to demonstrate how the Linux fieldbus coupler hardware can be accessed, all the programs have been installed on the coupler and are ready to run. They can be called directly from the console using the appropriate call parameters.

- **leds**
  Describes how to access the LEDs of the Linux coupler

- **kbusdemo**
  Describes how to access the process image

- **mb_tcp**
  Describes a more complex program which allows to access the internal bus via a Modbus TCP protocol.

- **CGI application**
  Describes the use of the CGI interface of the BOA web server for the transmission of dynamic web content

---

**Note**

The example programs are a basis for developing your own applications. In order to keep it clear and simple, we left out error handling for the most part, however, it should be implemented to use the code.
8.1 LED Indication Example

Program name: leds

Installation directory: /bin/leds

Source code files:
~/uclinux-dist/user/wagoled/leds.h
~/uclinux-dist/user/wagoled/leds.c

Driver/Kernel module: ledman (LED support)

Function description

Using this example program, you can access the freely programmable LEDs: STATUS, SERVICE and USR.

**Fig. 8-1: Display elements of the fieldbus coupler**

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>Signification</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATUS</td>
<td>Green/Red</td>
<td>Function can be defined by the user.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>SERVICE</td>
<td>Green/Red</td>
<td>Function can be defined by the user.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td></td>
</tr>
<tr>
<td>USR</td>
<td>Green/Red</td>
<td>Function can be defined by the user.</td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td></td>
</tr>
</tbody>
</table>
The kernel module ledman serves to easily access the free LEDs from a user program. In order to illustrate the driver interface, the Linux coupler incorporates the leds example program that allows to changes to the state of the LEDs via the Linux console.

The program is called directly via the console and has the following HELP page:

Usage: ./leds OPTION [LEDNAME] [OPTION] [LEDNAME] ...

WAGO LED SERVER.

OPTION:

- n, --on   Switch LED on
- o, --off  Switch LED off
- f, --flash Switch LED to flashing Mode
- h, --help Print this message.

LEDNAME:

sr   Status LED 1 (red)
sg   Status LED 1 (green)
er   Service LED 2 (red)
eg   Service LED 2 (green)
ur   User LED (red)
ung  User LED (green)
a    All LEDs

Proprietary. Send bug reports to support@wago.com

Different OPTIONS/commands can be sent to the LEDs via the example program. The LEDs can be turned on or off or they can blink.

Each OPTION requires an LEDNAME in order to define to which LED the OPTION/function is applied.
Example:

```bash
./leds -n ur  Switches on the red user LED
./leds -o ur  Switches off the red user LED
./leds -f sg  Status LED starts blinking green
./leds -n er -f ug Switches on the red service LED and the user LED starts blinking green
./leds -o a  Switches off all three LEDs
```

Source code description: leds.h

This header file contains definitions that control the access to the ledman driver. The following definitions describe the functions:

```c
#define WAGO_FBK_LED_CMD_SET 0x01/* turn on briefly to show activity*/
#define WAGO_FBK_LED_CMD_ON 0x02 /* turn LED on permanently */
#define WAGO_FBK_LED_CMD_OFF 0x03 /* turn LED off permanently */
#define WAGO_FBK_LED_CMD_FLASH 0x04 /* flash this LED */
```

Using these definitions, you can assign a function to every LED.

Additionally, the following LED names are defined which can be used to access the appropriate LED:

```c
#define WAGO_FBK_LED_DEBUG1_RED        2
#define WAGO_FBK_LED_DEBUG1_GREEN      3
#define WAGO_FBK_LED_DEBUG2_RED        4
#define WAGO_FBK_LED_DEBUG2_GREEN      5
#define WAGO_FBK_LED_USER_RED        6
#define WAGO_FBK_LED_USER_GREEN       7
#define WAGO_FBK_LED_ALL               8
```
For every LED there is one definition for red and one for green. Each of the three LEDs can be regarded as two separate LEDs. If, for example, the user LED is switched to RED (on) and at the same time switched to GREEN (on), it lights orange-green. The combination RED (on) and GREEN-BLINKING is also possible and results in a user LED that blinks rot, orange-green.

The header file also contains the `ledman_cmd` macro which implements the access to the kernel driver. Every time the driver is called, `/dev/ledman` is opened, written to and closed again.

**Source code description: leds.c**

The `leds.c` file contains the main() function. The console entries are analyzed and the `ledman_cmd()` macro (which is defined in `leds.h`) with its parameter values is called. The help text for the option "-help" is also defined here.
8.2 Example kbusdemo

Program name: kbusdemo

Installation directory: /bin/kbusdemo

Source code files:
~/uclinux-dist/user/wagokbusdemo/kbusdemo.h
~/uclinux-dist/user/wagokbusdemo/kbusdemo.c
~/uclinux-dist/user/wagokbusdemo/kbusapi.h
~/uclinux-dist/user/wagokbusdemo/kbusapi.c

Driver/Kernel module: kbus.o (internal bus support)

Function description

This example program can read the state of the input modules. It is furthermore possible to set or to clear bits of the output modules. The data is addressed via byte addresses. Writing data multiple bytes of data is not possible in this example. However, this feature could be added in the source code with little effort.

The dynamic loadable module "kbus.o" is used for the internal bus support.

In addition, kbusapi is used which makes it easier to access the internal bus module. This API (Application Interface) is added to the program during compilation and linking. The API interfaces are also described in this chapter.

The dynamic loadable module kbus.o serves to access the connected modules from a user program environment. In order to illustrate the driver interface, the Linux coupler incorporates the kbusdemo example program. This program can read or write the state/data of the internal bus via the Linux console.
The program is called directly via the console and has the following HELP page:

```
Usage: ./kbusdemo OPTION [ADDRESS] [OPTION] [ADDRESS] ...

WAGO KBUS DEMO.

OPTION:
-r, --read  Read Inputbyte [ADDRESS=xxx]
-n, --on    Switch on Outputbit [ADDRESS=xxx.x]
-o, --off   Switch off Outputbit [ADDRESS=xxx.x]
-h, --help  Print this message.

ADDRESS:

xxx.x        Startaddress (Byte.Bit) from Output/Input

Proprietary. Send bug reports to support@wago.com
```

Via the OPTIONS of this example program it is possible to select whether to read or to write data. The ADDRESS field determines the corresponding address.

The addresses follow the process image. The /proc/driver/kbus/pab file of the internal bus driver shows the configuration of the process image:

Example: `cat /proc/driver/kbus/pab`

```
Slot Terminal       Output:       Input:        
               Byte.Bit Offset     Byte.Bit Offset
8     750-553/000-000   0.0              x.x
9     750-559/000-000   6.0               x.x

---------------------------------------------------------------------
1     750-4xx             x.x               0.0
2     750-4xx             x.x               0.4
3     750-4xx             x.x               0.6
4     750-4xx             x.x               1.0
5     750-4xx             x.x               1.2
6     750-4xx             x.x               1.4
7     750-4xx             x.x               1.6

---------------------------------------------------------------------
FAB In:  00ffe060     FAB Size: 2040 bytes
FAB Out: 00ffe858     FAB Size: 2040 bytes
```
Every OPTION of the kbusdemo requires an address since every kbusdemo action, which is specified by the OPTION, is linked to a certain address in the process image.

Example:

```
./kbusdemo  -n  0.0
The bit for the output address 0.0 in the process image is set to TRUE

./kbusdemo  -o  1.7
The bit for the output address 1.7 in the process image is set to FALSE

./kbusdemo  -r  0
Reads the data at input byte 0
Output: IN[0]: 0b10100000  ||  IN[0]: 0xa0
The first byte has the following format: IN[Byte] binary - || IN[Byte] hexadecimal output
```
**Source code description: kbusdemo.h**

This header file contains process image access definitions. For performance reasons, the process image has a RAM address that cannot be accessed by the kernel. Definitions of these addresses are contained in the header file in order to access the process image.

**Source code description: kbusdemo.c**

The *kbusdemo.c* file contains the `main()` function. The console entries are analyzed and the internal bus read and write commands are executed.

Process image pointers, which are initialized by the use of definitions from the kbusdemo.h file, are used to read and write internal bus data. The KBusUpdate() function from the *kbusapi.h* file is used to update the process image. The console output is generated like the help text in that file.

**Source code description: kbusapi.h**

This header file contains definitions that make it easier to access the kbus.o driver.

**Source code description: kbusapi.c**

The accesses from the *kbusapi.h* file are coded in the *kbusapi.c* file.
8.3 Example of a Modbus TCP Server

Program name: mb_tcp

Installation directory: /bin/mb_tcp

Source code files:
~/.ucpdist/user/modbus/mb_tcp.h
~/.ucpdist/user/modbus/mb_tcp.c
~/.ucpdist/user/modbus/mbfunc.h
~/.ucpdist/user/modbus/mbfunc.c
~/.ucpdist/user/modbus/mbtypes.h
~/.ucpdist/user/modbus/mbserver.h
~/.ucpdist/user/modbus/mb_udp.c

Driver/Kernel modules: kbus.o (internal bus support)

Note
This example for a Modbus implementation does not completely comply with the Modbus specification. It is only an example for the implementation of a more complex program. Please find more information on MODBUS and MODBUS definitions under: http://www.modbus.org/.
In this example, a remote Modbus master can establish only one Modbus TCP connection. The user space server can only manage one connection at the same time.

Function description

This example program allows to read the state of the input modules and to set outputs via Ethernet. The data can be accessed byte by byte or bit by bit.

The following Modbus TCP function codes are available:

<table>
<thead>
<tr>
<th>Function code</th>
<th>Function name</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FC1: 0x01</td>
<td>Read Coils</td>
<td>Reading one single bit</td>
</tr>
<tr>
<td>FC2: 0x02</td>
<td>Read Input</td>
<td>Discrete reading of several input bits</td>
</tr>
<tr>
<td>FC3: 0x03</td>
<td>Read Multiple Registers</td>
<td>Reading several input registers</td>
</tr>
<tr>
<td>FC4: 0x04</td>
<td>Read InputRegisters</td>
<td>Reading several input registers</td>
</tr>
<tr>
<td>FC5: 0x05</td>
<td>Write Coil</td>
<td>Writing a single output bit</td>
</tr>
<tr>
<td>FC6: 0x06</td>
<td>Write SingleRegister</td>
<td>Writing a single output register</td>
</tr>
</tbody>
</table>
The Modbus addresses are:

Register Read Access:

<table>
<thead>
<tr>
<th>Start address</th>
<th>End address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0x01FF</td>
<td>Inputs of the process image</td>
</tr>
<tr>
<td>0x0200</td>
<td>0x03FF</td>
<td>Outputs of the process image</td>
</tr>
</tbody>
</table>

Register Write Access:

<table>
<thead>
<tr>
<th>Start address</th>
<th>End address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>0x01FF</td>
<td>Outputs of the process image</td>
</tr>
<tr>
<td>0x0200</td>
<td>0x03FF</td>
<td>Outputs of the process image</td>
</tr>
</tbody>
</table>

The dynamic loadable module "kbus.o" is used for the internal bus support.

In addition, `kbusapi` is used which makes it easier to access the internal bus module. This API (Application Interface) is added to the program during compilation and linking. The API interfaces are also described in this chapter.

The dynamic loadable module `kbus.o` serves to access the connected modules from a user program environment. In order to illustrate the driver interface, the Linux coupler incorporates the `kbusdemo` example program. This program can read or write the state/data of the internal bus via the Linux console.

**Source code description**

A more detailed description of the source code is not given now. The Modbus TCP server is only supposed to illustrate a more complex application on the Linux fieldbus coupler. Furthermore, the program is used to conduct a functional test of the Linux coupler and is hence installed on every coupler.

The customer can change or extend the source code arbitrarily.
8.4 Example Program for a CGI Application

Program names:
web, info, ether, kernelmsg, eeprom, config, restart

Installation directory: /var/www
index.html
upperframe.html
menuframe.html

Installation directory: /var/www/cgi-bin
web
info (symbolic link web)
ether (symbolic link web)
kernergmsg (symbolic link web)
eeprom (symbolic link web)
config (symbolic link web)
restart (symbolic link web)

Source code files:
~/uclinux-dist/user/wagoweb/html/index.html
~/uclinux-dist/user/wagoweb/html/upperframe.html
~/uclinux-dist/user/wagoweb/html/menuframe.html
~/uclinux-dist/user/wagoweb/web.c
~/uclinux-dist/user/wagoweb/info.c
~/uclinux-dist/user/wagoweb/ether.c
~/uclinux-dist/user/wagoweb/kernelmsg.c
~/uclinux-dist/user/wagoweb/eeprom.c
~/uclinux-dist/user/wagoweb/config.c
~/uclinux-dist/user/wagoweb/restart.c

Function description

This example program implements the web-based management (WBM) of the Linux fieldbus coupler.
The web server BOA uses the CGI files/programs. The HTML pages come directly from the web server.

The CGI file web can be called via different symbolic links. The name of the link is analyzed in the file and has a direct effect on the HTML output. The links described above are essential in order to execute the functions. This type of CGI call administration is ideal if you only want to need little memory in the file system for the files to be called. It is furthermore possible to exchange the entire functionality of the CGI calls by only replacing one file (see also chapter 4.5.5).
In the source code files, the `web.c` file only serves to connect code files. The actual functionality is in the file that has the same name as its symbolic call. The `func.c` file contains general functions.

**Source code description: *.html**

The above-mentioned HTML pages contain the general web page layout of the WAGO internet pages. The `Index.html` file divides the web page into three sections: one part for the headline, one part for the left selection menu and one for the dynamic part in the middle.

The headline of the page is stored in the `upperframe.html` file, the menu is stored in the `menuframe.html` file.

In order to load the status page, the `Index.html` file executes the CGI call as dynamic part.

**Source code description: web.c**

The name of the symbolic call is analyzed in the `web.c` file and the appropriate function is called.

**Source code description: info.c**

The status page of the Linux fieldbus coupler is generated in the `info.c` file. The file includes four functions:

- `read_envvar()`
- `read_BKTable()`
- `read_IP()
- `main_info()

The static HTML code is stored in the string `pucHtmlEthernet[]`, the dynamic values "%%s" being replaced by variable values directly in the `printf` command.

The function `read_envvar()` reads values from the highest flash memory sector, which, in this file, is the MAC address.

The function `read_BKTable()` reads firmware loader (FWL) or u-boot values from the table of the Linux fieldbus coupler. In this example, these are device description, item number, firmware version and serial number.
The function `read_IP()` reads the IP address that is currently used from the Ethernet device eth0. It is irrelevant whether the address was assigned dynamically, for example via DHCP, or if it is a fixed IP address.

`main_info()` calls the above-mentioned functions and writes the HTML code to the standard console stdout.

**Source code description: ether.c**

The TCP/IP configuration page of the Linux fieldbus coupler is generated in the `ether.c` file. The file includes five functions:

- `read_envvar()`
- `write_envvar()`
- `read_file()`
- `write_file()`
- `main_ether()`

The static HTML code is stored in the string `pucHtmlEthernet[]`, the dynamic values "%s" being replaced by a variable value directly in the `printf` command.

The function `read_envvar()` reads values from the highest flash memory sector. If a fixed IP address is used, these are the IP address, the subnet mask, the gateway and the host name. If the IP address is assigned dynamically via DHCP or BootP it is only the host name. The other values are set to default values.

The function `write_envvar()` writes values into the highest flash memory sector. If a fixed IP address is used, these are the IP address, the subnet mask, the gateway and the host name. In case of a dynamic IP address assignment via DHCP or BootP only the host name is saved. The other parameters keep their old values. All parameters are entered as a string into the environment variable `ipmode` which belongs to the start parameters of the Linux kernel. These environment variables are in the highest flash memory sector and are read by the boot loader (U-Boot).

`read_file()` reads the IP addresses of the name servers from the `/etc/resolv.conf` file.

The function `write_file()` writes the IP addresses of the name servers into the `/etc/resolv.conf` file.
The function `main_ether()` calls the above-mentioned functions and writes the HTML code to the standard console stdout. HTML shows the network configuration in a form. The form has the SUBMIT button `SAVE CONFIGURATION` which transfers a changed network configuration to the Linux fieldbus coupler in a QUERY_STRING. You have to enter a user with root privileges and the corresponding password.

These entries are authenticated in the `crypt.c` file. After successful authentication, the `SAVE CONFIGURATION` query string is analyzed and, if a value is changed, the appropriate `write_...-` function is called.

**Source code description: restart.c**

The restart page of the Linux fieldbus coupler is generated in the `restart.c` file. The file includes the function

```c
main_restart()
```

`main_restart()` writes the HTML code to the standard console stdout. Furthermore, the query string including the password is analyzed. After entering a user with root privileges and the corresponding password (default: user name: root, password: wago) and clicking the `REBOOT` button, `reboot` will be called which will reboot the operating system.
Source code description: *kernelmsg.c*

The *kernelmsg.c* file displays the kernel messages on the screen. The file includes the function

```c
main_kernelmsg()
```

*main_kernelmsg()* reads the kernel message ring buffer and displays the data on the screen.

Source code description: *eeprom.c*

The *eeprom.c* file displays the EEPROM messages on the screen. The file includes the function *main_eeprom()*.

*main_eeprom()* reads the */dev/eeprom* device file and displays the data on the screen in a table.

Source code description: *config.c*

The *config.c* file displays the IO modules (sorted by type) that are connected to the Linux fieldbus coupler on the screen.

The file includes the functions:

- `printStandard()`
- `printAnalogOutput()`
- `printAnalogInput()`
- `printDigitalOutput()`
- `printDigitalInput()`
- `printSpecial()`
- `main_config()`

The *printStandard()* function outputs the standard parameters of an IO module (type, slot number etc.).

The *printAnalogOutput()* or *Input()* function displays the special parameters of the individual analog modules separately in pop-up tables.
The `printDigitalOutput()` or `...Input()` function displays the special parameters of the individual digital modules separately in pop-up tables.

The `printSpecial()` function displays the special parameters of the individual complex modules (specialty modules) separately in pop-up tables.

The `main_config()` function initializes that part of the pop-up table which needs to be opened. For that purpose, the submit buttons are analyzed.
8.5 Example of Reading and Writing from the NVRAM

The NVRAM (Non Volatile Random Access Memory) is a non-volatile memory that retains its information even after the Linux fieldbus coupler has been switched off.

Benefit of the NVRAM:

- Short access times similar to RAM.
- Can be written to like a RAM
- Storage of data in the event of power loss similar to flash memory.

Access to the NVRAM

Access to the NVRAM is done via the `nvram` device file which is located in the `dev` directory of the Linux fieldbus coupler. The NVRAM can be written and read byte by byte. The following functions allow user space access to the NVRAM:

- `int open (const char *pathname, int flags);`
  This function opens the indicated file and returns the file descriptor. The file descriptor is a reference to the open file.

  `pathname:` Path of the device file: `/dev/nvram`
  `flags:` `O_WRONLY`, `O_RDONLY`, `O_RDWR`

- `ssize_t read(int fd, void *buf, size_t count);`
  This function reads a specified number of bytes from the referenced file and returns the number of bytes read.

  `fd:` File descriptor returned by the `open` function.
  `buf:` A pointer to the buffer into which to read the data.
  `count:` Number of bytes to be moved to the target buffer.

- `ssize_t write(int fd, const char *buf, size_t count);`
  This function writes a specified number of bytes to the referenced file and returns the number of bytes written.

  `fd:` File descriptor returned by the `open` function.
  `buf:` A pointer to the buffer from which to write the data to the memory.
  `count:` Number of bytes to be written.
Example of Reading and Writing from the NVRAM

- int close(int fd);
  This function closes the open file which is a reference to the file descriptor.

  *fd*: File descriptor returned by the open function.

- off_t lseek(int fd, off_t offset, int mode);
  This function sets the position pointer to any byte within the file and returns the current position in the file.

  *fd*: File descriptor returned by the open function.
  *offset*: Position within the file from which the bytes are read and written.
  *mode*:
  SEEK_SET: Within the file, both read/write operations start from the offset value (in bytes).

  SEEK_CUR: Within the file, both read/write operations start from the current position + offset value.

  SEEK_END: Within the file, both read/write operations start from the last byte + offset value (here the offset is usually "0" or negative value).
Example Program

```c
#include <fcntl.h>
#include <stdio.h>

int main()
{
    int i;
    int file;
    ssize_t sstmp;
    char buf[2048];

    /*Open Device*/
    if((file = open("/dev/nvram") < 0)
    {
        printf("ERROR: Can’t open device
("/dev/nvram")
        close(file);
        exit(1);
    }

    /*Read Device*/
    sstmp = read(file, buf, 8);
    printf("read %d bytes\n", sstmp);
    if(sstmp != 8)
    {
        printf("ERROR: Reading NVRAM failed\n");
    }
    else
    {
        printf("read(%x %x %x %x %x %x %x %x)\n", buf[0],
                buf[1], buf[2], buf[3], buf[4], buf[5], buf[6],
                buf[7]);
    }

    /*Trivial change of values*/
    for(i=0;i<8;i++)
    { buf[i] += 1;

    /*Write into Device*/
    if(-1 == lseek(file, 0, SEEK_SET))
        printf("ERROR: lseek failed\n");
    sstmp = write(file, buf, 8);
    printf("Write (%d) bytes\n", sstmp);
    if(sstmp != 8)
        printf("ERROR: Writing into Device failed\n");

    /*close device*/
    close(file);
}
```
9 Bootloader (U-Boot)

U-Boot and the Linux kernel boot up process are controlled via environment variables within the boot loader that are stored in the last sector of the flash memory. These variables are verified by a CRC32 checksum. The environment variables can be changed via the boot loader, the web server or through an example application in Linux. To use the boot loader, you have to enter a password during start-up of the Linux fieldbus coupler which will then give access to the configuration interface (see chapter 5.10.2). The environment variable can now be displayed using

```
LFBK # printenv
```

or be changed:

```
LFBK # setenv <variablenname> <variablenwert>
```

There is no equal sign between variable name and variable value. All changes that are made are not permanent and are lost when the Linux fieldbus coupler is rebooted. Use the `saveenv` call, in order to get permanent changes in the flash memory.

```
LFBK # saveenv
```

Saving Environment to Flash...

```
LFBK #
```

A list of all possible Shell commands can be viewed using

```
Help
```

9.1 Boot Loader Toolchain

The boot loader requires a special toolchain which is called "DENX Embedded Linux Development Kit" (ELDK). A pre-compiled version for x86 host systems is available under: `ftp://ftp.leo.org/pub/eldk/2.1.0/eldk-arm-linux-x86/` or on the WAGO distribution CD-ROM under `CD:/toolchain/uboot/`.

You install the toolchain and the other utility programs in the super user mode with the following command:

```
> rpm -i gcc-arm-2.95.4-4j_3.i386.rpm gcc-doc-2.95.4-4j_3.i386.rpm binutils-arm-2.11.93.0.2-3b_2.i386.rpm binutils-doc-2.11.93.0.2-3b_2.i386.rpm cpp-arm-2.95.4-4j_3.i386.rpm
```
9.2 TFTP Server Setup

In order to download the Linux kernel, the file system or the boot loader, it is essential to have a TFTP server on the development computer. Also, the user needs to have access rights for the `/tftpboot` directory in the root directory:

```plaintext
> su
Password

> mkdir /tftpboot
> chmod 777 /tftpboot

You have to install `inetd` and `atftp` via yast2 (S.U.S.E) before you can install a TFTP server on the development PC. After the installation, make the following entry in the `/etc/inetd.conf`:

```plaintext
<host-ip>:tftp dgram udp wait root /usr/sbin/in.tftpd -s /tftpboot
```

inetd starts automatically after a reboot or it can be started manually:

```plaintext
> /etc/init.d/inetd start

or restarted with

> /etc/init.d/inetd stop
> /etc/init.d/inetd start
```
9.3 Unpack the Source Code

The boot loader (U-Boot) source code is on the CD in the file: 
CD:/uboot/sources/. The installation is done via

```bash
> cd /media/cdrom/uboot/sources
> sh install.sh
```

A directory called `~/uboot` is created in the user's home directory, in which all files that are required for the boot loader compilation are stored.

9.4 Compile Boot Loader

The source code is configured and compiled via

```bash
> cd ~/uboot
> make modnet50_config
> make
```

The ELF object file is called `ub.elf`,
the binary image for a Linux coupler update is called `ub.bin`,
which is copied into the `/tftpboot` directory as `ub.bin`.

Create Flash Images

Images of the kernel and of the file system can be created during the compilation of the Linux distribution (see chapter 5.3). A boot loader image is created during the compilation of the boot loader U-Boot. The files are automatically copied into the `/tftpboot` directory which has to be created beforehand. See chapter 9.2.
9.5 Environment Variables

A list of all environment variables set can be viewed using

`> printenv`

An environment variable can contain values, further environment variables or U-Boot commands. In the latter case, the environment variable works similar to a macro which processes the commands one after the other.

The relevant environment variables for the Linux fieldbus coupler boot process are:

- **ipmode**
  Defines the type of IP configuration. Possible values are `bootp`, `dhcp`, a static IP in the format `xxx.xxx.xxx.xxx` or a completely static configuration in the format
  `10.1.1.8:10.1.2:1:10.1.1.254:255.255.255.0:lfbk:eth0`

- **bootargs**
  Contains further parameters for the kernel, for example the root file system (`root=/dev/mtdblock3`)

- **bootcmd**
  Contains a macro that creates a kernel command line from `ipmode` and `bootargs` and starts the kernel. This macro is called automatically 3 seconds after starting the Linux fieldbus coupler (stored in the `bootdelay` variable).

- **bootA/bootB**
  Macros that, depending on the selector switch position of the Linux fieldbus coupler, are copied to `bootcmd` when U-Boot is started.

- **bootdelay**
  The time the system waits for the abort password to arrive

- **ethaddr**
  MAC address of the Linux fieldbus coupler
The following environment variables are available for a software update:

- `ipaddr`
  IP address of the Linux fieldbus coupler

- `ethaddr`
  MAC address of the Linux fieldbus coupler

- `serverip`
  IP address of the TFTP server

- `netmask`
  Netmask of the network

- `update-rootfs`
  Macro for a root file system update

- `update-kernel`
  Macro for a Linux kernel update

**Note**
Please refer to chapter 9.2 for details on a TFTP server setup (required for a software update).

The following environment variables can be used to determine the standard interfaces:

- `stdin`
  Standard input interface

- `stdout`
  Standard output interface

- `stderr`
  Standard error output interface

- `baudrate`
  Baud rate of the serial interface

An update of the environment variables via the Linux operating system is executed via the device interface `/dev/mtdblock5`. The wagoset example program was developed for this purpose.
10 Fieldbus Communication

10.1 ETHERNET

10.1.1 General

ETHERNET is a technology, which has been proven and established as an effective means of data transmission in the field of information technology and office communication. Within a short time ETHERNET has also made a successful breakthrough in the area of private PC networks throughout the world.

This technology was developed in 1972 by Dr. Robert M. Metcalfe, David R. Boggs, Charles Thacker, Butler W. Lampson, and Xerox (Stanford, Ct.). Standardization (IEEE 802.3) took place in 1983.

ETHERNET predominantly uses coaxial cables or twisted pair cables as a transmission medium. Connection to ETHERNET, often already existing in networks, (LAN, Internet) is easy and the data exchange at a transmission rate of 10 Mbps or for some couplers/controllers also 100 Mbps is very fast.

ETHERNET has been equipped with higher level communication software in addition to standard IEEE 802.3, such as TCP/IP (Transmission Control Protocol / Internet Protocol) to allow communication between different systems. The TCP/IP protocol stack offers a high degree of reliability for the transmission of information.

In the ETHERNET based (programmable) fieldbus couplers and controllers developed by WAGO, usually various application protocols have been implemented on the basis of the TCP/IP stack.

These protocols allow the user to create applications (master applications) with standardized interfaces and transmit process data via an ETHERNET interface.

In addition to a series of management and diagnostic protocols, fieldbus specific application protocols are implemented for control of the module data, depending upon the coupler or controller, e. g. MODBUS TCP (UDP), EtherNet/IP, BACnet, KNXNET/IP, PROFINET, Powerlink, Sercos III or others.

Information such as the fieldbus node architecture, network statistics and diagnostic information is stored in the ETHERNET (programmable) fieldbus couplers and controllers and can be viewed as HTML pages via a web browser (e.g., Microsoft Internet-Explorer, Netscape Navigator) being served from the HTTP server in the couplers and controllers.

Furthermore, depending on the requirements of the respective industrial application, various settings such as selection of protocols, TCP/IP, internal clock and security configurations can be performed via the web-based management system. However, you can also load web pages you have created yourself into the couplers/controllers, which have an internal file system, using FTP.
The WAGO ETHERNET TCP/IP fieldbus node does not require any additional master components other than a PC with a network card. So, the fieldbus node can be easily connected to local or global networks using the fieldbus connection. Other networking components such as hubs, switches or repeaters can also be used. However, to establish the greatest amount of “determinism” a switch is recommended.

The use of ETHERNET as a fieldbus allows continuous data transmission between the plant floor and the office. Connection of the ETHERNET TCP/IP fieldbus node to the Internet even enables industrial processing data for all types of applications to be called up world-wide. This makes site independent monitoring, visualization, remote maintenance and control of processes possible.

10.1.2 Network Architecture – Principles and Regulations

A simple ETHERNET network is designed on the basis of one PC with a network interface card (NI), one crossover connection cable (if necessary), one ETHERNET fieldbus node and one 24 V DC power supply for the coupler/controller voltage source.

Each fieldbus node consists of a (programmable) fieldbus coupler or controller and a number of needed I/O modules.

Sensors and actuators are connected to the digital or analog I/O modules on the field side. These are used for process signal acquisition or signal output to the process, respectively.

Fieldbus communication between master application and (programmable) fieldbus coupler or controller takes place using the implemented fieldbus specific application protocol, e. g. MODBUS TCP (UDP), EtherNet/IP, BACnet, KNXNET/IP, PROFINET, Powerlink, Sercos III or others.
10.1.2.1 Transmission Media

General ETHERNET transmission standards

For transmitting data the ETHERNET standard supports numerous technologies with various parameters (e.g., transmission speed, medium, segment length and type of transmission).

<table>
<thead>
<tr>
<th>Technology</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10Base5</td>
<td>Uses a 24 AWG UTP (twisted pair cable) for a 1Mbps baseband signal for distances up to 500 m (250 m per segment) in a physical star topology.</td>
</tr>
<tr>
<td>10Base2</td>
<td>Uses a 5 mm 50 Ohm coaxial cable for a 10Mbps baseband signal for distances of up to 185 m in a physical bus topology (often referred to as Thin ETHERNET or ThinNet).</td>
</tr>
<tr>
<td>10Base5</td>
<td>Uses a 10 mm 50 Ohm coaxial cable for a 10Mbps baseband signal for distances of up to 500 m in a physical bus topology (often referred to as Thick ETHERNET).</td>
</tr>
<tr>
<td>10Base-F</td>
<td>Uses a fiber-optic cable for a 10Mbps baseband signal for distances of up to 4 km in a physical star topology. (There are three sub-specifications: 10Base-FL for fiber-optic link, 10Base-FB for fiber-optic backbone and 10Base-FP for fiber-optic passive).</td>
</tr>
<tr>
<td>10Base-T</td>
<td>Uses a 24 AWG UTP or STP/UTP (twisted pair cable) for a 10Mbps baseband signal for distances up to 100 m in a physical star topology.</td>
</tr>
<tr>
<td>10Broad36</td>
<td>Uses a 75 Ohm coaxial cable for a 10Mbps baseband signal for distances of up to 1800 m (or 3600 m with double cables) in a physical bus topology.</td>
</tr>
<tr>
<td>100BaseTX</td>
<td>Specifies a 100 Mbps transmission with a twisted pair cable of Category 5 and RJ45-connectors. A maximum segment of 100 meters may be used.</td>
</tr>
</tbody>
</table>

Tab. 10-1: ETHERNET Transmission Standards

Beyond that there are still further transmission standards, for example: 100Base-T4 (Fast ETHERNET over twisted conductors), 100Base-FX (Fast ETHERNET over fiber-optic cables) or P802.11 (Wireless LAN) for a wireless transmission.

The media types are shown with their IEEE shorthand identifiers. The IEEE identifiers include three pieces of information. The first item, for example, “10”, stands for the media. The third part of the identifier provides a rough indication of segment type or length. For thick coaxial cable, the “5” indicates a 500 meter maximum length allowed for individual thick coaxial segments. For thin coaxial cable, the “2” is rounded up from the 185 meter maximum length for individual thin coaxial segments. The “T” and “F” stand for ‘twisted pair’ and ‘fiber optic’, and simply indicate the cable type.
10Base-T, 100BaseTX

Either the 10BaseT standard or 100BaseTX can be used for the WAGO ETHERNET fieldbus node. The network architecture is very easy and inexpensive to assemble with S-UTP cable as transmission medium or with cables of STP type. Both types of cable can be obtained from any computer dealer.

S-UTP cable (screened unshielded twisted pair) is single-shielded cable of Category 5 with overall shield surrounding all twisted unshielded conductor pairs and an impedance of 100 ohm. STP cable (shielded twisted pair) is cable of Category 5 with stranded and individually shielded conductor pairs; no overall shield is provided.

Wiring of the fieldbus nodes

Maybe, a crossover cable is required for direct connection of a fieldbus node to the network card of the PC.

An ETHERNET switch is a device that allows all connected devices to transmit and receive data with each other. The switch can also be viewed as a “data traffic cop” where the hub “polices” the data coming in and going out of the individual ports, so the data will only be transmitted to the required node. WAGO recommends using a switch rather than a hub, this will allow for a more deterministic architecture.
### Attention
The cable length between the node and the hub cannot be longer than 100 m (328 ft.) without adding signal conditioning systems (i.e., repeaters). Various possibilities are described in the ETHERNET standard for networks covering larger distances.

---

**10.1.2.2 Network Topologies**

In the case of 10Base-T, or 100BaseTX several stations (nodes) are connected using a star topology according to the 10Base-T ETHERNET Standard.

Therefore, this manual only deals with the star topology, and the tree topology for larger networks in more detail.

**Star Topology**

A star topology consists of a network in which all nodes are connected to a central point via individual cables.

![Star Topology Diagram](image)

Fig. 10-4: Star Topology

A star topology offers the advantage of allowing the extension of an existing network. Stations can be added or removed without network interruption. Moreover, in the event of a defective cable, only the network segment and the node connected to this segment is impaired. This considerably increases the fail-safe of the entire network.
Tree Topology

The tree topology combines characteristics of linear bus and star topologies. It consists of groups of star-configured workstations connected to a linear bus backbone cable. Tree topologies allow for the expansion of an existing network, and enables schools, etc. to configure a network to meet their needs.

![Tree Topology Diagram](image-url)
5-4-3 Rule

A consideration in setting up a tree topology using ETHERNET protocol is the 5-4-3 rule. One aspect of the ETHERNET protocol requires that a signal sent out on the network cable must reach every part of the network within a specified length of time. Each concentrator or repeater that a signal goes through adds a small amount of time. This leads to the rule that between any two nodes on the network there can only be a maximum of 5 segments connected through 4 repeaters/concentrators. In addition, only 3 of the segments may be populated (trunk) segments if they are made of coaxial cable. A populated segment is one that has one or more nodes attached to it. In Figure 5-5, the 5-4-3 rule is adhered to. The furthest two nodes on the network have 4 segments and 3 repeaters/concentrators between them.

This rule does not apply to other network protocols or ETHERNET networks where all fiber optic cabling or a combination of a backbone with UTP cabling is used. If there is a combination of fiber optic backbone and UTP cabling, the rule is simply translated to 7-6-5 rule.

Cabling guidelines

"Structured Cabling" specifies general guidelines for network architecture of a LAN, establishing maximum cable lengths for the grounds area, building and floor cabling.

The "Structured Cabling" is standardized in EN 50173, ISO 11801 and TIA 568-A. It forms the basis for a future-orientated, application-independent and cost-effective network infrastructure.

The cabling standards define a domain covering a geographical area of 3 km and for an office area of up to 1 million square meters with 50 to 50,000 terminals. In addition, they describe recommendations for setting up of a cabling system.

Specifications may vary depending on the selected topology, the transmission media and coupler modules used in industrial environments, as well as the use of components from different manufacturers in a network. Therefore, the specifications given here are only intended as recommendations.
10.1.2.3 Coupler Modules

There are a number of hardware modules that allow for flexible arrangement for setting up an ETHERNET network. They also offer important functions, some of which are very similar.

The following table defines and compares these modules and is intended to simplify the correct selection and appropriate application of them.

<table>
<thead>
<tr>
<th>Module</th>
<th>Characteristics/application</th>
<th>ISO/OSI layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeater</td>
<td>Amplifier for signal regeneration, connection on a physical level.</td>
<td>1</td>
</tr>
<tr>
<td>Bridge</td>
<td>Segmentation of networks to increase the length.</td>
<td>2</td>
</tr>
<tr>
<td>Switch</td>
<td>Multiport bridge, meaning each port has a separate bridge function. Logically separates network segments, thereby reducing network traffic. Consistent use makes ETHERNET collision-free.</td>
<td>2 (3)</td>
</tr>
<tr>
<td>Hub</td>
<td>Used to create star topologies, supports various transmission media, does not prevent any network collisions.</td>
<td>2</td>
</tr>
<tr>
<td>Router</td>
<td>Links two or more data networks. Matches topology changes and incompatible packet sizes (e.g. used in industrial and office areas).</td>
<td>3</td>
</tr>
<tr>
<td>Gateway</td>
<td>Links two manufacturer-specific networks which use different software and hardware (i.e., ETHERNET and Interbus-Loop).</td>
<td>4-7</td>
</tr>
</tbody>
</table>

Tab. 10-2: Comparison of Coupler Modules for Networks

10.1.2.4 Transmission Mode

Some ETHERNET based WAGO couplers/controllers support both 10Mbit/s and 100Mbit/s for either full or half duplex operation. To guarantee a safe and fast transmission, both these couplers/controllers and their link partners must be configured for the same transmission mode.

**Note**

A faulty configuration of the transmission mode may result in a link loss condition, a poor network performance or a faulty behavior of the coupler/controller.

The IEEE 802.3u ETHERNET standard defines two possibilities for configuring the transmission modes:

- Static configuration
- Dynamic configuration
10.1.2.4.1 Static Configuration of the Transmission Mode

Using static configuration, both link partners are set to static transmission rate and duplex mode. The following configurations are possible:

- 10 Mbit/s, half duplex
- 10 Mbit/s, full duplex
- 100 Mbit/s, half duplex
- 100 Mbit/s, full duplex

10.1.2.4.2 Dynamic Configuration of the Transmission Mode

The second configuration option is the autonegotiation mode which is defined in the IEEE 802.3u standard. Using this mode, the transmission rate and the duplex mode are negotiated dynamically between both communication partners. Autonegotiation allows the device to automatically select the optimum transmission mode.

**Note**

To ensure a correct dynamic configuration process, the operation mode for the autonegotiation of both communication partners must be supported and activated.

10.1.2.4.3 Errors Occurring when Configuring the Transmission Mode

Invalid configurations are listed below:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Cause</th>
<th>Symptoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mismatch of the transmission rate</td>
<td>Occurs when configuring one link partner with 10 Mbit/s and the other one with 100 Mbit/s.</td>
<td>Link failure</td>
</tr>
<tr>
<td>Duplex mode mismatch</td>
<td>Occurs when one link partner is running in full-duplex and the other in half-duplex mode.</td>
<td>Faulty or discarded data packets as well as collisions on the medium.</td>
</tr>
<tr>
<td>Mismatch using autonegotiation</td>
<td>Occurs when one link partner is running in auto-negotiation mode and the other one is using a static configuration of the transmission mode in full-duplex operation.</td>
<td>The link partner, which is in autonegotiation mode, determines the network speed via the parallel detection procedure and sets the duplex mode to half-duplex. If the device is operating in full-duplex mode with static configuration, a duplex mode mismatch will occur (see above).</td>
</tr>
</tbody>
</table>
10.1.2.5 Important Terms

Data security

If an internal network (Intranet) is to be connected to the public network (e.g., the Internet) then data security is an extremely important aspect.

Undesired access can be prevented by a **Firewall**. Firewalls can be implemented in software or network components. They are interconnected in a similar way to routers as a switching element between Intranets and the public network. Firewalls are able to limit or completely block all access to the other networks, depending on the access direction, the service used and the authenticity of the network user.

Real-time ability

Transmission above the fieldbus system level generally involves relatively large data quantities. The permissible delay times may also be relatively long (0.1...10 seconds).

However, real-time behavior within the fieldbus system level is required for ETHERNET in industry.

In ETHERNET it is possible to meet the real-time requirements by restricting the bus traffic (< 10 %), by using a master-slave principle, or also by implementing a switch instead of a hub. MODBUS/TCP is a master/slave protocol in which the slaves only respond to commands from the master. When only one master is used, data traffic over the network can be controlled and collisions avoided.

Shared ETHERNET

Several nodes linked via a hub share a common medium. When a message is sent from a station, it is broadcast throughout the entire network and is sent to each connected node. Only the node with the correct target address processes the message. Collisions may occur and messages have to be repeatedly transmitted as a result of the large amount of data traffic. The delay time in a Shared ETHERNET cannot be easily calculated or predicted.

Deterministic ETHERNET

![Diagram of Shared ETHERNET](image)
The TCP/IP software or the user program in each subscriber can limit transmittable messages to make it possible to determine real-time requirements. At the same time the maximum medium message rate (datagrams per second), the maximum medium duration of a message, and the minimum time interval between the messages (waiting time of the subscriber) is limited.

Therefore, the delay time of a message is predictable.

**Switched ETHERNET**

In the case of Switched Ethernet, several fieldbus nodes are connected by a switch. When data from a network segment reaches the switch, it saves the data and checks for the segment and the node to which this data is to be sent. The message is then only sent to the node with the correct target address. This reduces the data traffic over the network, extends the bandwidth and prevents collisions. The runtimes can be defined and calculated, making the Switched Ethernet deterministic.

![Principle of Switched ETHERNET](image)

**Fig. 10-7: Principle of Switched ETHERNET**
10.1.3 Network Communication

Fieldbus communication between master application and (programmable) fieldbus coupler or controller usually takes place using an implemented fieldbus specific application protocol, e.g. MODBUS TCP (UDP), EtherNet/IP, BACnet, KNXNET/IP, PROFINET, Powerlink, Sercos III or others.

The protocol layer model helps with an example (MODBUS and EtherNet/IP) to explain the classification and interrelationships between the communication and application protocols.
In this example, the fieldbus communication can take place using either the MODBUS protocol or EtherNet/IP.

10.1.3.1 Protocol layer model

(1) Ethernet:

The Ethernet hardware forms the basis for the physical exchange of data. The exchanged data signals and the bus access procedure CSMA/CD are defined in a standard.

(2) IP:

For the communication the Internet Protocol (IP) is positioned above the Ethernet hardware. This bundles the data to be transmitted in packets along with sender and receiver address and passes these packets down to the Ethernet layer for physical transmission. At the receiver end, IP accepts the packets from the Ethernet layer and unpacks them.

(3) TCP, UDP:

a) TCP: (Transmission Control Protocol)
The TCP protocol, which is positioned above the IP layer, monitors the transport of the data packets, sorts their sequence and sends repeat requests for missing packets. TCP is a connection-oriented transport protocol.
The TCP and IP protocol layers are also jointly described as the TCP/IP protocol stack or TCP/IP stack.
b) UDP: (User Datagram Protocol)
The UDP layer is also a transport protocol like TCP, and is arranged above the IP layer. In contrast to the TCP protocol, UDP is not connection oriented. That means there are no monitoring mechanisms for data exchange between sender and receiver.
The advantage of this protocol is in the efficiency of the transmitted
data and the resultant increase in processing speed. Many programs use both protocols. Important status information is sent via the reliable TCP connection, while the main stream of data is sent via UDP.

---

**Diagram: TCP, UDP, IP, Ethernet (physical interface, CSMA/CD)**

**4) Management, Diagnostic and Application Protocols:**

Positioned above the TCP/IP stack or UDP/IP layer are correspondingly implemented management, diagnostic and application protocols that provide services that are appropriate for the application. For the management and diagnostic, these are, for example, SMTP (Simple Mail Transport Protocol) for e-mails, HTTP (Hypertext Transport Protocol) for www browsers and some others.

In this example, the protocols MODBUS/TCP (UDP) and EtherNet/IP are implemented for use in industrial data communication. Here the MODBUS protocol is also positioned directly above TCP (UDP)/IP; EtherNet/IP, on the other hand, basically consists of the protocol layers Ethernet, TCP and IP with an encapsulation protocol positioned above it. This serves as interface to CIP (Control and Information Protocol).

DeviceNet uses CIP in the same way as EtherNet/IP. Applications with DeviceNet device profiles can therefore be very simply transferred to EtherNet/IP.
10.1.3.2 Communication Protocols

In addition to the ETHERNET standard, the following important communication protocols are implemented in the WAGO ETHERNET based (programmable) fieldbus couplers and controllers:

- IP Version 4 (Raw-IP and IP-Multicast)
- TCP
- UDP
- ARP

The following diagram is intended to explain the data structure of these protocols and how the data packets of the communication protocols Ethernet, TCP and IP with the adapted application protocol MODBUS nested in each other for transmission. A detailed description of the tasks and addressing schemes of these protocols is contained in the following.

Fig. 10-8: Communication Protocols
10.1.3.2.1 ETHERNET

ETHERNET address (MAC-ID)

Each WAGO ETHERNET (programmable) fieldbus coupler or controller is
provided from the factory with a unique and internationally unambiguous
physical ETHERNET address, also referred to as MAC-ID (Media Access
Control Identity). This can be used by the network operating system for
addressing on a hardware level.

The address has a fixed length of 6 Bytes (48 Bit) and contains the address
type, the manufacturer’s ID, and the serial number.

Examples for the MAC-ID of a WAGO ETHERNET fieldbus coupler
(hexadecimal): 00H.30H.DEH.00H.00H.01H.

ETHERNET does not allow addressing of different networks.
If an ETHERNET network is to be connected to other networks, higher-
ranking protocols have to be used.

Note
If you wish to connect one or more data networks, routers have to be used.

ETHERNET Packet

The datagrams exchanged on the transmission medium are called
“ETHERNET packets” or just “packets”. Transmission is connectionless; i.e.
the sender does not receive any feedback from the receiver. The data used is
packed in an address information frame. The following figure shows the
structure of such a packet.

<table>
<thead>
<tr>
<th>Preamble</th>
<th>ETHERNET-Header</th>
<th>ETHERNET_Data</th>
<th>Check sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 Byte</td>
<td>14 Byte</td>
<td>46-1500 Byte</td>
<td>4 Byte</td>
</tr>
</tbody>
</table>

Fig. 10-9: ETHERNET-Packet

The preamble serves as a synchronization between the transmitting station and
the receiving station. The ETHERNET header contains the MAC addresses of
the transmitter and the receiver, and a type field.
The type field is used to identify the following protocol by way of
unambiguous coding (e.g., 0800hex = Internet Protocol).

10.1.3.3 Channel access method

In the ETHERNET Standard, the fieldbus node accesses the bus using
CSMA/CD (Carrier Sense Multiple Access/ Collision Detection).

- Carrier Sense: The transmitter senses the bus.
- Multiple Access: Several transmitters can access the bus.
- Collision Detection: A collision is detected.

Each station can send a message once it has established that the transmission
medium is free. If collisions of data packets occur due to several stations
transmitting simultaneously, CSMA/CD ensures that these are detected and
the data transmission is repeated.

However, this does not make data transmission reliable enough for industrial
requirements. To ensure that communication and data transmission via ETHERNET is reliable, various communication protocols are required.

10.1.3.3.1 IP-Protocol

The Internet protocol divides datagrams into segments and is responsible for their transmission from one network subscriber to another. The stations involved may be connected to the same network or to different physical networks which are linked together by routers. Routers are able to select various paths (network transmission paths) through connected networks, and bypass congestion and individual network failures. However, as individual paths may be selected which are shorter than other paths, datagrams may overtake each other, causing the sequence of the data packets to be incorrect. Therefore, it is necessary to use a higher-level protocol, for example, TCP to guarantee correct transmission.

**IP addresses**

To allow communication over the network each fieldbus node requires a 32 bit Internet address (IP address).

**Attention**

Internet addresses have to be unique throughout the entire interconnected networks.

As shown below there are various address classes with net identification (net ID) and subscriber identification (subscriber ID) of varying lengths. The net ID defines the network in which the subscriber is located. The subscriber ID identifies a particular subscriber within this network. Networks are divided into various network classes for addressing purposes:

- **Class A**: (Net-ID: Byte1, Host-ID: Byte2 - Byte4)
  
  e.g.: 101 . 16 . 232 . 22
  
  01100101 00010000 11101000 00010110
  
  0  Net-ID          Host-ID
  
  The highest bit in Class A networks is always ‘0’. Meaning the highest byte can be in a range of ‘0 0000000’ to ‘0 1111111’.
  
  Therefore, the address range of a Class A network in the first byte is always between 0 and 127.
Fieldbus Communication

ETHERNET

- **Class B**: (Net-ID: Byte1 - Byte2, Host-ID: Byte3 - Byte4)
  
  e.g.: 181 . 16 . 232 . 22

  | 10110101 | 00010000 | 11101000 | 00010110 |
  | Net-ID   | Host-ID  |

  The highest bits in Class B networks are always ‘10’.
  Meaning the highest byte can be in a range of ‘10 000000’ to ‘10 111111’.

  Therefore, the address range of Class B networks in the first byte is always between 128 and 191.

- **Class C**: (Net-ID: Byte1 - Byte3, Host-ID: Byte4)
  
  e.g.: 201 . 16 . 232 . 22

  | 11000101 | 00010000 | 11101000 | 00010110 |
  | Net-ID   | Host-ID  |

  The highest bits in Class C networks are always ‘110’.
  Meaning the highest byte can be in a range of ‘110 00000’ to ‘110 11111’.

  Therefore, the address range of Class C networks in the first byte is always between 192 and 223.

  Additional network classes (D, E) are only used for special tasks.

Key data

<table>
<thead>
<tr>
<th>Address range of the subnet network</th>
<th>Possible number of networks</th>
<th>Subscribers per network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A 1.XXX.XXX.XXX - 126.XXX.XXX.XXX</td>
<td>127 (2^7)</td>
<td>Ca. 16 Million (2²⁷)</td>
</tr>
<tr>
<td>Class B 128.000.XXX.XXX - 191.255.XXX.XXX</td>
<td>Ca. 16 thousand (2¹⁰)</td>
<td>Ca 65 thousand (2¹⁰)</td>
</tr>
<tr>
<td>Class C 192.000.000.XXX - 223.255.255.XXX</td>
<td>Ca. 2 million (2²¹)</td>
<td>254 (2⁸)</td>
</tr>
</tbody>
</table>

Each WAGO ETHERNET (programmable) fieldbus coupler or controller can be easily assigned an IP address via the implemented BootP protocol. For small internal networks we recommend selecting a network address from Class C.

**Attention**

Never set all bits to equal 0 or 1 in one byte (byte = 0 or 255). These are reserved for special functions and may not be allocated. Therefore, the address 10.0.10.10 may not be used due to the 0 in the second byte.
If a network is to be directly connected to the Internet, only registered, internationally unique IP addresses allocated by a central registration service may be used. These are available from InterNIC (International Network Information Center).

**Attention**
Direct connection to the Internet should only be performed by an authorized network administrator and is therefore not described in this manual.

**Subnets**

To allow routing within large networks a convention was introduced in the specification *RFC 950*. Part of the Internet address, the subscriber ID is divided up again into a subnetwork number and the station number of the node. With the aid of the network number it is possible to branch into internal subnetworks within the partial network, but the entire network is physically connected together. The size and position of the subnetwork ID are not defined; however, the size is dependent upon the number of subnets to be addressed and the number of subscribers per subnet.

![Fig. 10-10: Class B address with Field for Subnet ID](image)

**Subnet mask**

A subnet mask was introduced to encode the subnets in the Internet. This involves a bit mask, which is used to mask out or select specific bits of the IP address. The mask defines the subscriber ID bits used for subnet coding, which denote the ID of the subscriber. The entire IP address range theoretically lies between 0.0.0.0 and 255.255.255.255. Each 0 and 255 from the IP address range are reserved for the subnet mask.

The standard masks depending upon the respective network class are as follows:

- **Class A Subnet mask:**
  
  255 .0 .0 .0

- **Class B Subnet mask:**
  
  255 .255 .0 .0

- **Class C Subnet mask:**
  
  255 .255 .255 .0

Depending on the subnet division the subnet masks may, however, contain other values beyond 0 and 255, such as 255.255.255.128 or 255.255.255.248. Your network administrator allocates the subnet mask number to you.
Together with the IP address, this number determines which network your PC and your node belongs to.

The recipient node, which is located on a subnet initially, calculates the correct network number from its own IP address and the subnet mask. Only then does it check the node number and delivers the entire packet frame, if it corresponds.

Example of an IP address from a class B network:

<table>
<thead>
<tr>
<th>IP address:</th>
<th>172.16.233.200</th>
<th>10101100 00010000 11101001 11001000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet mask:</td>
<td>255.255.255.128</td>
<td>11111111 11111111 11111111 10000000</td>
</tr>
<tr>
<td>Net-ID:</td>
<td>172.16.00</td>
<td>10101100 00010000 00000000 00000000</td>
</tr>
<tr>
<td>Subnet-ID:</td>
<td>0.0.233.128</td>
<td>00000000 00000000 11101001 10000000</td>
</tr>
<tr>
<td>Host-ID:</td>
<td>0.0.0.72</td>
<td>00000000 00000000 00000000 01001000</td>
</tr>
</tbody>
</table>

**Attention**

Specify the network mask defined by the administrator in the same way as the IP address when installing the network protocol.

**Gateway**

The subnets of the Internet are normally connected via gateways. The function of these gateways is to forward packets to other networks or subnets.

This means that in addition to the IP address and network mask for each network card, it is necessary to specify the correct IP address of the standard gateway for a PC or fieldbus node connected to the Internet. You should also be able to obtain this IP address from your network administrator. The IP function is limited to the local subnet if this address is not specified.

**IP Packet**

In addition to the data units to be transported, the IP data packets contain a range of address information and additional information in the packet header.

<table>
<thead>
<tr>
<th>IP-Header</th>
<th>IP-Data</th>
</tr>
</thead>
</table>

Fig. 10-11: IP Packet

The most important information in the IP header is the IP address of the transmitter and the receiver and the transport protocol used.
10.1.3.3.1 RAW IP

Raw IP manages without protocols such as PPP (point-to-point protocol). With RAW IP, the TCP/IP packets are directly exchanged without handshaking, thus enabling the connection to be established more quickly.

However, the connection must beforehand have been configured with a fixed IP address. The advantages of RAW IP are high data transfer rate and good stability.

10.1.3.3.2 IP Multicast

Multicast refers to a method of transmission from a point to a group, which is a point-to-multipoint transfer or multipoint connection. The advantage of multicast is that messages are simultaneously transferred to several users or closed user groups via one address.

IP multicasting at Internet level is realised with the help of the Internet Group Message Protocol IGMP; neighbouring routers use this protocol to inform each other on membership to the group.

For distribution of multicast packets in the sub-network, IP assumes that the datalink layer supports multicasting. In the case of Ethernet, you can provide a packet with a multicast address in order to send the packet to several recipients with a single send operation. Here, the common medium enables packets to be sent simultaneously to several recipients. The stations do not have to inform each other on who belongs to a specific multicast address – every station physically receives every packet. The resolution of IP address to Ethernet address is solved by the use of algorithms, IP multicast addresses are embedded in Ethernet multicast addresses.

10.1.3.3.2 TCP Protocol

As the layer above the Internet protocol, TCP (Transmission Control Protocol) guarantees the secure transport of data through the network.

TCP enables two subscribers to establish a connection for the duration of the data transmission. Communication takes place in full-duplex mode (i.e., transmission between two subscribers in both directions simultaneously).

TCP provides the transmitted message with a 16-bit checksum and each data packet with a sequence number.

The receiver checks that the packet has been correctly received on the basis of the checksum and then sets off the sequence number. The result is known as the acknowledgement number and is returned with the next self-sent packet as an acknowledgement.

This ensures that the lost TCP packets are detected and resent, if necessary, in the correct sequence.
TCP port numbers

TCP can, in addition to the IP address (network and subscriber address), respond to a specific application (service) on the addressed subscriber. For this the applications located on a subscriber, such as a web server, FTP server and others are addressed via different port numbers. Well-known applications are assigned fixed ports to which each application can refer when a connection is built up.

Examples:

<table>
<thead>
<tr>
<th>Application</th>
<th>Port number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telnet</td>
<td>23</td>
</tr>
<tr>
<td>HTTP</td>
<td>80</td>
</tr>
</tbody>
</table>

A complete list of "standardized services" is contained in the RFC 1700 (1994) specifications.

TCP segment

The packet header of a TCP data packet is comprised of at least 20 bytes and contains, among others, the application port number of the transmitter and the receiver, the sequence number and the acknowledgement number.

The resulting TCP packet is used in the data unit area of an IP packet to create a TCP/IP packet.

10.1.3.3 UDP

The UDP protocol, like the TCP protocol, is responsible for the transport of data. Unlike the TCP protocol, UDP is not connection-oriented; meaning that there are no control mechanisms for the data exchange between transmitter and receiver. The advantage of this protocol is the efficiency of the transmitted data and the resulting higher processing speed.

10.1.3.4 ARP

ARP (Address Resolution Protocol).
This protocol combines the IP address with the physical MAC address of the respective Ethernet card. It is always used when data transfer to an IP address takes place in the same logical network in which the sender is located.
10.1.3.4 Administration and Diagnosis Protocols

In addition to the communication protocols described above, various fieldbus specific application protocols and a view protocols for system administration and diagnosis can be implemented.

- BootP
- HTTP
- DHCP
- DNS
- SNTP
- FTP
- SMTP.

More information

You can find a list of the exact available implemented protocols in the chapter "Technical Data" to the fieldbus coupler and/or controller.

10.1.3.4.1 BootP (Bootstrap Protocol)

The BootP protocol defines a request/response mechanism with which the MAC-ID of a fieldbus node can be assigned a fix IP address.

For this a network node is enabled to send requests into the network and call up the required network information, such as the IP address of a BootP server. The BootP server waits for BootP requests and generates the response from a configuration database.

The dynamic configuration of the IP address via a BootP server offers the user a flexible and simple design of his network. The WAGO BootP server allows any IP address to be easily assigned for the WAGO (programmable) fieldbus coupler or controller. You can download a free copy of the WAGO BootP server over the Internet at:  http://www.wago.com

More information

The procedure for address allocation with the WAGO BootP Server is described in detail in the Chapter “Starting up a Fieldbus Node”.

The BOOTP Client allows for dynamic configuring of the network parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address of the client</td>
<td>Network address of the (programmable) fieldbus coupler or controller</td>
</tr>
<tr>
<td>IP address of the router</td>
<td>If communication is to take place outside of the local network, the IP address of the routers (gateway) is indicated in this parameter.</td>
</tr>
<tr>
<td>Subnet mask</td>
<td>The Subnet mask makes the (programmable) fieldbus coupler or controller able to differentiate, which parts of the IP address determine the network and which the</td>
</tr>
<tr>
<td>IP addresses of the DNS servers</td>
<td>Here the IP addresses can be entered by maximally 2 DNS servers.</td>
</tr>
<tr>
<td>Host name</td>
<td>Name of the host</td>
</tr>
</tbody>
</table>

When using the bootstrap protocol for configuring the node, the network parameters (IP address, etc...) are stored in the EEPROM.

**Note**

The network configuration is only stored in the EEPROM when the BootP protocol is used, although not if configuration is done via DHCP.

The BootP protocol is activated in the (programmable) fieldbus coupler or controller by default.

When the BootP protocol is activated, the (programmable) fieldbus coupler or controller expects a BootP server to be permanently present.

If, however, there is no BootP server available after a power-on reset, the network remains inactive.

To operate the (programmable) fieldbus coupler or controller with the IP configuration stored in the EEPROM, you must first deactivate the BootP protocol.

This is done via the web-based management system on the appropriate HTML page saved in the (programmable) fieldbus coupler or controller, which is accessed via the “Port” link.

If the BootP protocol is deactivated, the (programmable) fieldbus coupler or controller uses the parameters stored in the EEPROM at the next boot cycle.

If there is an error in the stored parameters, a blink code is output via the IO LED and configuration via BootP is automatically switched on.

### 10.1.3.4.2 HTTP (HyperText Transfer Protocol)

HTTP is a protocol used by WWW (World Wide Web) servers for the forwarding of hypermedia, texts, images, audiodata, etc.

Today, HTTP forms the basis of the Internet and is also based on requests and responses in the same way as the BootP protocol.

The HTTP server implemented in the (programmable) fieldbus coupler or controller is used for viewing the HTML pages saved in the coupler/controller. The HTML pages provide information about the coupler/controller (state, configuration), the network and the process image.

On some HTML pages, (programmable) fieldbus coupler or controller settings can also be defined and altered via the web-based management system (e.g. whether IP configuration of the coupler/controller is to be performed via the DHCP protocol, the BootP protocol or from the data stored in the EEPROM).

The HTTP server uses port **number 80**.
10.1.3.4.3 DHCP (Dynamic Host Configuration Protocol)

The coupler’s/controller’s built-in HTML pages provide an option for IP configuration from a DHCP server, a BootP server, or the data stored in its EEPROM by default.

**Note**
The network configuration via DHCP is not stored in the EEPROM, this only occurs when using the BootP protocol.

The DHCP client allows dynamic network configuration of the coupler/controller by setting the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP address of the client</td>
<td>Network address of the coupler/controller</td>
</tr>
<tr>
<td>IP address of the router</td>
<td>If communication is to take place outside of the local network, the IP address of the routers (gateway) is indicated in this parameter.</td>
</tr>
<tr>
<td>Subnet mask</td>
<td>The Subnet mask makes the coupler/controller able to differentiate, which parts of the IP address determine the network and which the network station.</td>
</tr>
<tr>
<td>IP addresses of the DNS servers</td>
<td>Here the IP addresses can be entered by maximally 2 DNS servers.</td>
</tr>
<tr>
<td>Lease time</td>
<td>Here the maximum duration can be defined, how long the coupler/controller keeps the assigned IP address. The maximum lease time is 24.8 days. This results from the internal resolution of timer.</td>
</tr>
<tr>
<td>Renewing time</td>
<td>The Renewing time indicates, starting from when the coupler/controller must worry about the renewal of the leasing time.</td>
</tr>
<tr>
<td>Rebinding time</td>
<td>The Rebinding time indicates, after which time the coupler/controller must have gotten its new address.</td>
</tr>
</tbody>
</table>

In the case of configuration of network parameters via the DHCP protocol, the coupler/controller automatically sends a request to a DHCP server after initialisation. If there is no response, the request is sent again after 4 seconds, a further one after 8 seconds and again after 16 seconds. If all requests remain unanswered, a blink code is output via the “IO” LED. Transfer of the parameters from the EEPROM is not possible.

Where a lease time is used, the values for the renewing and rebinding time must also be specified. After the renewing time expires, the coupler/controller attempts to automatically renew the lease time for its IP address. If this continually fails up to the rebinding time, the coupler/controller attempts to obtain a new IP address. The time for the renewing should be about one half of the lease time. The rebinding time should be about \( \frac{7}{8} \) of the lease time.
10.1.3.4.4 DNS (Domain Name Systems)

The DNS client enables conversion of logical Internet names such as [www.wago.com](http://www.wago.com) into the appropriate decimal IP address represented with separator stops, via a DNS server. Reverse conversion is also possible. The addresses of the DNS server are configured via DHCP or web-based management. Up to 2 DNS servers can be specified. The host identification can be achieved with two functions, an internal host table is not supported.

10.1.3.4.5 SNTP-Client (Simple Network Time Protocol)

The SNTP client is used for synchronization of the time of day between a time server (NTP and SNTP server Version 3 and 4 are supported) and the clock module integrated in the (programmable) fieldbus coupler or controller. The protocol is executed via a UDP port. Only unicast addressing is supported.

**Configuration of the SNTP client**

The configuration of the SNTP client is performed via the web-based management system under the “Clock” link. The following parameters must be set:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address of the Time server</td>
<td>The address assignment can be made either over a IP address or a host name.</td>
</tr>
<tr>
<td>Time zone</td>
<td>The time zone relative to GMT (Greenwich Mean time). A range of -12 to +12 hours is acceptable.</td>
</tr>
<tr>
<td>Update Time</td>
<td>The update time indicates the interval in seconds, in which the synchronization with the time server is to take place.</td>
</tr>
<tr>
<td>Enable Time Client</td>
<td>It indicates whether the SNTP Client is to be activated or deactivated.</td>
</tr>
</tbody>
</table>

10.1.3.4.6 FTP-Server (File Transfer Protocol)

The file transfer protocol (FTP) enables files to be exchanged between different network stations regardless of operating system.

In the case of the ETHERNET coupler/controller, FTP is used to store and read the HTML pages created by the user, the IEC61131 program and the IEC61131 source code in the (programmable) fieldbus coupler or controller.

A total memory of 1.5 MB is available for the file system. The file system is mapped to RAM disk. To permanently store the data of the RAM disk, the information is additionally copied into the flash memory. The data is stored in the flash after the file has been closed. Due to the storage process, access times during write cycles are long.
Note
Up to 1 million write cycles are possible for writing to the flash memory for the file system.

The following table shows the supported FTP commands for accesses to the file system:

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER</td>
<td>Identification of the user</td>
</tr>
<tr>
<td>PASS</td>
<td>User password</td>
</tr>
<tr>
<td>ACCT</td>
<td>Account for access to certain files</td>
</tr>
<tr>
<td>REIN</td>
<td>Server reset</td>
</tr>
<tr>
<td>QUIT</td>
<td>Terminates the connection</td>
</tr>
<tr>
<td>PORT</td>
<td>Addressing of the data link</td>
</tr>
<tr>
<td>PASV</td>
<td>Changes server in the listen mode</td>
</tr>
<tr>
<td>TYPE</td>
<td>Determines the kind of the representation for the transferred file</td>
</tr>
<tr>
<td>STRU</td>
<td>Determines the structure for the transferred file</td>
</tr>
<tr>
<td>MODE</td>
<td>Determines the kind of file transmission</td>
</tr>
<tr>
<td>RETR</td>
<td>Reads file from server</td>
</tr>
<tr>
<td>STOR</td>
<td>Saves file on server</td>
</tr>
<tr>
<td>APPE</td>
<td>Saves file on server (Append mode)</td>
</tr>
<tr>
<td>ALLO</td>
<td>Reservation of the necessary storage location for the file</td>
</tr>
<tr>
<td>RNFR</td>
<td>Renames file from (with RNTO)</td>
</tr>
<tr>
<td>RNTO</td>
<td>Renames file in (with RNFR)</td>
</tr>
<tr>
<td>ABOR</td>
<td>Stops current function</td>
</tr>
<tr>
<td>DELE</td>
<td>Deletes file</td>
</tr>
<tr>
<td>CWD</td>
<td>Changes directory</td>
</tr>
<tr>
<td>LIST</td>
<td>Gives the directory list</td>
</tr>
<tr>
<td>NLST</td>
<td>Gives the directory list</td>
</tr>
<tr>
<td>RMD</td>
<td>Deletes directory</td>
</tr>
<tr>
<td>PWD</td>
<td>Gives the actually path</td>
</tr>
<tr>
<td>MKD</td>
<td>Puts on a directory</td>
</tr>
</tbody>
</table>

The TFTP (Trivial File Transfer Protocol) is not supported by some of the couplers/controllers.

More information
You can find a list of the exact available implemented protocols in the chapter "Technical Data" to the fieldbus coupler and/or controller.
10.1.3.4.7 SMTP (Simple Mail Transfer Protocol)

The Simple Mail Transfer Protocol (SMTP) enables sending of ASCII text messages to mail boxes on TCP/IP hosts in a network. It is therefore used for sending and receiving e-mails.

The e-mail to be sent is created with a suitable editor and placed in a mail outbasket.

A send SMTP process polls the out-basket at regular intervals and therefore finds mail waiting to be sent. It then establishes a TCP/IP connection with the target host, to which the message is transmitted. The receive SMTP process on the target host accepts the TCP connection. The message is then transmitted and finally placed in an in-basket on the target system. SMTP expects the target system to be online, otherwise no TCP connection can be established. Since many desktop computers are switched off at the end of the day, it is impractical to send SMTP mail there. For that reason, in many networks special SMTP hosts are installed in many networks, which are permanently switched on to enable distribution of received mail to the desktop computers.

10.1.3.5 Application Protocols

If fieldbus specific application protocols are implemented, then the appropriate fieldbus specific communication is possible with the respective coupler/controller. Thus the user is able to have a simple access from the respective fieldbus on the fieldbus node. There are based on ETHERNET couplers/controllers available developed by WAGO, with the following possible application protocols:

- MODBUS TCP (UDP)
- EtherNet/IP
- BACnet
- KNXnet/IP
- PROFINET
- Powerlink
- Sercos III

More information

You can find a list of the exact available implemented protocols in the chapter "Technical Data" to the fieldbus coupler and/or controller. If fieldbus specific application protocols are implemented, then these protocols are individual described in the following chapters.
11 Use in Hazardous Environments

11.1 Foreword

Today’s development shows that many chemical and petrochemical companies have production plants, production, and process automation machines in operation which use gas-air, vapor-air and dust-air mixtures which can be explosive. For this reason, the electrical components used in such plants and systems must not pose a risk of explosion resulting in injury to persons or damage to property. This is backed by law, directives or regulations on a national and international scale. WAGO-I/O-SYSTEM 750 (electrical components) is designed for use in zone 2 explosive environments. The following basic explosion protection related terms have been defined.

11.2 Protective Measures

Primarily, explosion protection describes how to prevent the formation of an explosive atmosphere. For instance by avoiding the use of combustible liquids, reducing the concentration levels, ventilation measures, to name but a few. But there are a large number of applications, which do not allow the implementation of primary protection measures. In such cases, the secondary explosion protection comes into play. Following is a detailed description of such secondary measures.

11.3 Classification Meeting CENELEC and IEC

The specifications outlined here are valid for use in Europe and are based on the following standards: EN50... of CENELEC (European Committee for Electrotechnical Standardization). On an international scale, these are reflected by the IEC 60079-... standards of the IEC (International Electrotechnical Commission).

11.3.1 Divisions

Explosive environments are areas in which the atmosphere can potentially become explosive. The term explosive means a special mixture of ignitable substances existing in the form of air-borne gases, fumes, mist or dust under atmospheric conditions which, when heated beyond a tolerable temperature or subjected to an electric arc or sparks, can produce explosions. Explosive zones have been created to describe the concentrations level of an explosive atmosphere. This division, based on the probability of an explosion occurring, is of great importance both for technical safety and feasibility reasons. Knowing that the demands placed on electrical components permanently employed in an explosive environment have to be much more stringent than those placed on electrical components that are only rarely and, if at all, for short periods, subject to a dangerous explosive environment.
Explosive areas resulting from gases, fumes or mist:

- Zone 0 areas are subject to an explosive atmosphere (> 1000 h /year) continuously or for extended periods.
- Zone 1 areas can expect the occasional occurrence of an explosive atmosphere (> 10 h ≤ 1000 h /year).
- Zone 2 areas can expect the rare or short-term occurrence of an explosive atmosphere (> 0 h ≤ 10 h /year).

Explosive areas subject to air-borne dust:

- Zone 20 areas are subject to an explosive atmosphere (> 1000 h /year) continuously or for extended periods.
- Zone 21 areas can expect the occasional occurrence of an explosive atmosphere (> 10 h ≤ 1000 h /year).
- Zone 22 areas can expect the rare or short-term occurrence of an explosive atmosphere (> 0 h ≤ 10 h /year).
11.3.2 Explosion Protection Group

In addition, the electrical components for explosive areas are subdivided into two groups:

Group I: Group I includes electrical components for use in fire-damp endangered mine structures.

Group II: Group II includes electrical components for use in all other explosive environments. This group is further subdivided by pertinent combustible gases in the environment. Subdivision IIA, IIB and IIC takes into account that different materials/substances/gases have various ignition energy characteristic values. For this reason the three subgroups are assigned representative types of gases:

IIA – Propane
IIB – Ethylene
IIC – Hydrogen

Tab. 11-1: Minimal ignition energy of representative types of gases

<table>
<thead>
<tr>
<th>Explosion group</th>
<th>I</th>
<th>IIA</th>
<th>IIB</th>
<th>IIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gases</td>
<td>Methane</td>
<td>Propane</td>
<td>Ethylene</td>
<td>Hydrogen</td>
</tr>
<tr>
<td>Ignition energy (µJ)</td>
<td>280</td>
<td>250</td>
<td>82</td>
<td>16</td>
</tr>
</tbody>
</table>

Hydrogen being commonly encountered in chemical plants, frequently the explosion group IIC is requested for maximum safety.
11.3.3 Unit Categories

Moreover, the areas of use (zones) and the conditions of use (explosion groups) are subdivided into categories for the electrical operating means:

Tab. 11-2: Unit categories

<table>
<thead>
<tr>
<th>Unit category</th>
<th>Explosion group</th>
<th>Area of use</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>I</td>
<td>Fire-damp protection</td>
</tr>
<tr>
<td>M2</td>
<td>I</td>
<td>Fire-damp protection</td>
</tr>
<tr>
<td>1G</td>
<td>II</td>
<td>Zone 0 Explosive environment by gas, fumes or mist</td>
</tr>
<tr>
<td>2G</td>
<td>II</td>
<td>Zone 1 Explosive environment by gas, fumes or mist</td>
</tr>
<tr>
<td>3G</td>
<td>II</td>
<td>Zone 2 Explosive environment by gas, fumes or mist</td>
</tr>
<tr>
<td>1D</td>
<td>II</td>
<td>Zone 20 Explosive environment by dust</td>
</tr>
<tr>
<td>2D</td>
<td>II</td>
<td>Zone 21 Explosive environment by dust</td>
</tr>
<tr>
<td>3D</td>
<td>II</td>
<td>Zone 22 Explosive environment by dust</td>
</tr>
</tbody>
</table>
11.3.4 Temperature Classes

The maximum surface temperature for electrical components of explosion protection group I is 150 °C (danger due to coal dust deposits) or 450 °C (if there is no danger of coal dust deposit).

In line with the maximum surface temperature for all ignition protection types, the electrical components are subdivided into temperature classes, as far as electrical components of explosion protection group II are concerned. Here the temperatures refer to a surrounding temperature of 40 °C for operation and testing of the electrical components. The lowest ignition temperature of the existing explosive atmosphere must be higher than the maximum surface temperature.

Tab. 11-3: Temperature classes

<table>
<thead>
<tr>
<th>Temperature Classes</th>
<th>Maximum Surface Temperature</th>
<th>Ignition Temperature of the Combustible Materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>450 °C</td>
<td>&gt; 450 °C</td>
</tr>
<tr>
<td>T2</td>
<td>300 °C</td>
<td>&gt; 300 °C to 450 °C</td>
</tr>
<tr>
<td>T3</td>
<td>200 °C</td>
<td>&gt; 200 °C to 300 °C</td>
</tr>
<tr>
<td>T4</td>
<td>135 °C</td>
<td>&gt; 135 °C to 200 °C</td>
</tr>
<tr>
<td>T5</td>
<td>100 °C</td>
<td>&gt; 100 °C to 135 °C</td>
</tr>
<tr>
<td>T6</td>
<td>85°C</td>
<td>&gt; 85 °C to 100 °C</td>
</tr>
</tbody>
</table>

The following table represents the division and attributes of the materials to the temperature classes and material groups in percent:

Tab. 11-4: Material groups in percent

<table>
<thead>
<tr>
<th>Temperature classes</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>26.6 %</td>
<td>42.8 %</td>
<td>25.5 %</td>
<td>94.9 %</td>
<td>4.9 %</td>
<td>0 %</td>
<td>0.2 %</td>
<td>432</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Explosion group</th>
<th>IIA</th>
<th>IIB</th>
<th>IIC</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>85.2 %</td>
<td>13.8 %</td>
<td>1.0 %</td>
<td>Total 501</td>
<td></td>
</tr>
</tbody>
</table>

* Number of classified materials
11.3.5 Types of Ignition Protection

Ignition protection defines the special measures to be taken for electrical components in order to prevent the ignition of surrounding explosive atmospheres. For this reason a differentiation is made between the following types of ignition protection:

Table 11-5: Types of Ignition Protection

<table>
<thead>
<tr>
<th>Identification</th>
<th>CENELEC standard</th>
<th>IEC standard</th>
<th>Explanation</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEx o</td>
<td>EN 50 015</td>
<td>IEC 79-6</td>
<td>Oil encapsulation</td>
<td>Zone 1 + 2</td>
</tr>
<tr>
<td>EEx p</td>
<td>EN 50 016</td>
<td>IEC 79-2</td>
<td>Overpressure encapsulation</td>
<td>Zone 1 + 2</td>
</tr>
<tr>
<td>EEx q</td>
<td>EN 50 017</td>
<td>IEC 79-5</td>
<td>Sand encapsulation</td>
<td>Zone 1 + 2</td>
</tr>
<tr>
<td>EEx d</td>
<td>EN 50 018</td>
<td>IEC 79-1</td>
<td>Pressure resistant encapsulation</td>
<td>Zone 1 + 2</td>
</tr>
<tr>
<td>EEx e</td>
<td>EN 50 019</td>
<td>IEC 79-7</td>
<td>Increased safety</td>
<td>Zone 1 + 2</td>
</tr>
<tr>
<td>EEx m</td>
<td>EN 50 028</td>
<td>IEC 79-18</td>
<td>Cast encapsulation</td>
<td>Zone 1 + 2</td>
</tr>
<tr>
<td>EEx i</td>
<td>EN 50 020 (unit)</td>
<td>IEC 79-11</td>
<td>Intrinsic safety</td>
<td>Zone 0 + 1 + 2</td>
</tr>
<tr>
<td></td>
<td>EN 50 039 (system)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEx n</td>
<td>EN 50 021</td>
<td>IEC 79-15</td>
<td>Electrical components for zone 2 (see below)</td>
<td>Zone 2</td>
</tr>
</tbody>
</table>

Ignition protection “n” describes exclusively the use of explosion protected electrical components in zone 2. This zone encompasses areas where explosive atmospheres can only be expected to occur rarely or short-term. It represents the transition between the area of zone 1, which requires an explosion protection and safe area in which for instance welding is allowed at any time.

Regulations covering these electrical components are being prepared on a world-wide scale. The standard EN 50 021 allows electrical component manufacturers to obtain certificates from the corresponding authorities for instance KEMA in the Netherlands or the PTB in Germany, certifying that the tested components meet the above mentioned standards draft.

Type “n” ignition protection additionally requires electrical components to be marked with the following extended identification:

- A – non spark generating (function modules without relay /without switches)
- AC – spark generating, contacts protected by seals (function modules with relays / without switches)
- L – limited energy (function modules with switch)
Further Information
For more detailed information please refer to the national and/or international standards, directives and regulations!

11.4 Classifications Meeting the NEC 500

The following classifications according to NEC 500 (National Electric Code) are valid for North America.

11.4.1 Divisions

The "Divisions" describe the degree of probability of whatever type of dangerous situation occurring. Here the following assignments apply:

<table>
<thead>
<tr>
<th>Explosion endangered areas due to combustible gases, fumes, mist and dust:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Division 1</td>
<td>Encompasses areas in which explosive atmospheres are to be expected occasionally ($&gt; 10 \text{ h } \leq 1000 \text{ h/year}$) as well as continuously and long-term ($&gt; 1000 \text{ h/year}$).</td>
</tr>
<tr>
<td>Division 2</td>
<td>Encompasses areas in which explosive atmospheres can be expected rarely and short-term ($&gt;0 \text{ h } \leq 10 \text{ h/year}$).</td>
</tr>
</tbody>
</table>

11.4.2 Explosion Protection Groups

Electrical components for explosion endangered areas are subdivided in three danger categories:

<table>
<thead>
<tr>
<th>Class I (gases and fumes):</th>
<th>Group A (Acetylene)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Group B (Hydrogen)</td>
</tr>
<tr>
<td></td>
<td>Group C (Ethylene)</td>
</tr>
<tr>
<td></td>
<td>Group D (Methane)</td>
</tr>
<tr>
<td>Class II (dust):</td>
<td>Group E (Metal dust)</td>
</tr>
<tr>
<td></td>
<td>Group F (Coal dust)</td>
</tr>
<tr>
<td></td>
<td>Group G (Flour, starch and cereal dust)</td>
</tr>
<tr>
<td>Class III (fibers):</td>
<td>No sub-groups</td>
</tr>
</tbody>
</table>
11.4.3 Temperature Classes

Electrical components for explosive areas are differentiated by temperature classes:

<table>
<thead>
<tr>
<th>Temperature classes</th>
<th>Maximum surface temperature</th>
<th>Ignition temperature of the combustible materials</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>450 °C</td>
<td>&gt; 450 °C</td>
</tr>
<tr>
<td>T2</td>
<td>300 °C</td>
<td>&gt; 300 °C to 450 °C</td>
</tr>
<tr>
<td>T2A</td>
<td>280 °C</td>
<td>&gt; 280 °C to 300 °C</td>
</tr>
<tr>
<td>T2B</td>
<td>260 °C</td>
<td>&gt; 260 °C to 280 °C</td>
</tr>
<tr>
<td>T2C</td>
<td>230 °C</td>
<td>&gt; 230 °C to 260 °C</td>
</tr>
<tr>
<td>T2D</td>
<td>215 °C</td>
<td>&gt; 215 °C to 230 °C</td>
</tr>
<tr>
<td>T3</td>
<td>200 °C</td>
<td>&gt; 200 °C to 215 °C</td>
</tr>
<tr>
<td>T3A</td>
<td>180 °C</td>
<td>&gt; 180 °C to 200 °C</td>
</tr>
<tr>
<td>T3B</td>
<td>165 °C</td>
<td>&gt; 165 °C to 180 °C</td>
</tr>
<tr>
<td>T3C</td>
<td>160 °C</td>
<td>&gt; 160 °C to 165 °C</td>
</tr>
<tr>
<td>T4</td>
<td>135 °C</td>
<td>&gt; 135 °C to 160 °C</td>
</tr>
<tr>
<td>T4A</td>
<td>120 °C</td>
<td>&gt; 120 °C to 135 °C</td>
</tr>
<tr>
<td>T5</td>
<td>100 °C</td>
<td>&gt; 100 °C to 120 °C</td>
</tr>
<tr>
<td>T6</td>
<td>85 °C</td>
<td>&gt; 85 °C to 100 °C</td>
</tr>
</tbody>
</table>
11.5 Identification

11.5.1 For Europe

According to CENELEC and IEC

![Diagram explaining the identification label of a hazardous environment device](image-url)

Fig. 11-1: Example for lateral labeling of bus modules
(750-400, 2 channel digital input module 24 V DC)
11.5.2 For America

According to NEC 500

Fig. 11.5.2-1: Example for lateral labeling of bus modules
(750-400, 2 channel digital input module 24 V DC)
11.6 Installation Regulations

In the Federal Republic of Germany, various national regulations for the installation in explosive areas must be taken into consideration. The basis being the ElexV complemented by the installation regulation DIN VDE 0165/2.91. The following are excerpts from additional VDE regulations:

<table>
<thead>
<tr>
<th>DIN VDE 0100</th>
<th>Installation in power plants with rated voltages up to 1000 V</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN VDE 0101</td>
<td>Installation in power plants with rated voltages above 1 kV</td>
</tr>
<tr>
<td>DIN VDE 0800</td>
<td>Installation and operation in telecommunication plants including information processing equipment</td>
</tr>
<tr>
<td>DIN VDE 0185</td>
<td>lightning protection systems</td>
</tr>
</tbody>
</table>

The USA and Canada have their own regulations. The following are excerpts from these regulations:

<table>
<thead>
<tr>
<th>NFPA 70</th>
<th>National Electrical Code Art. 500 Hazardous Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSI/ISA-RP 12.6-1987</td>
<td>Recommended Practice</td>
</tr>
<tr>
<td>C22.1</td>
<td>Canadian Electrical Code</td>
</tr>
</tbody>
</table>

Use in Hazardous Environments
Installation Regulations

**Danger**
When using the WAGO-I/O SYSTEM 750 (electrical operation) with Ex approval, the following points are mandatory:

The field bus independent I/O System Modules Type 750-xxx are to be installed in enclosures that provide for the degree of ingress protection of at least IP54.
For use in the presence of combustible dust, the above mentioned modules are to be installed in enclosures that provide for the degree of ingress protection of at least IP64.

The field bus independent I/O system may only be installed in hazardous areas (Europe: Group II, Zone 2 or America: Class I, Division 2, Group A, B, C, D) or in non-hazardous areas!

Installation, connection, addition, removal or replacement of modules, field bus connectors or fuses may only take place when the system supply and the field supply are switched off, or when the area is known to be non-hazardous.

Ensure that only approved modules of the electrical operating type will be used. The Substitution or Replacement of modules can jeopardize the suitability of the system in hazardous environments!

Operation of intrinsically safe EEx i modules with direct connection to sensors/actuators in hazardous areas of Zone 0 + 1 and Division 1 type requires the use of a 24 V DC Power Supply EEx i module!

DIP switches and potentiometers are only to be adjusted when the area is known to be non-hazardous.

---

**Further Information**
Proof of certification is available on request.
Also take note of the information given on the module technical information sheet.
12 Glossary

B

**Bit**
Smallest information unit. Its value can either be 1 or 0.

**Bit rate**
Number of bits transmitted within a time unit.

**BootP**
the bootstrap protocol is a protocol which specifies how system and network information is to be transmitted from a server to work stations.

**Bridge**
A network bridge serves to transmit messages independent of the destination of the message. It operates at the data link layer (layer 2) of the ISO/OSI model.
Bridges divide networks into segments which allows to increase the number of nodes. Incomplete telegrams are removed. Telegrams are only sent if the node's target address is in the connected segment. A bridge creates a data base which contains all station addresses (MAC addresses). On the basis of this data, the bridge decides whether or not the received data packets are forwarded to another network segment. By and by, the bridge will be better able to decide in which segment the data belongs. Today, switches are often used instead of bridges.

**Broadcast**
A message that is sent to all station connected to the network.

**Bus**
Line for bit serial or bit parallel clocked data transfer. A bus for bit parallel data transfer consists of address, data, control and supply bus. The width of the data bus (8, 16, 32, 64 bits) and its frequency are the determining factors for the data transmission speed. The width of the address bus limits the network expansion.

**Byte**
Binary Yoked Transfer Element. A byte generally contains 8 bits.
Client

A system that requests the services of another. With the aid of the service request, the client can access objects (data) on the server. The service is provided by the server.

CSMA/CD

Carrier Sense Multiple Access with Collision Detection. When a collision is detected, all subscribers back off. After waiting a random delay time, the subscribers attempt to re-transmit the data.

Data bus

see Bus.

Deterministic ETHERNET

The ETHERNET data is transferred at a defined time constant. The ETHERNET network can be defined and calculated. A Switched ETHERNET architecture makes this possible.

Driver

Software code which communicates with a hardware device. This communication is normally performed by internal device registers.

DHCP

Dynamic Host Configuration Protocol. This protocol allows to configure the network of a computer automatically as well as to assign addresses or to set parameters. The DHCP uses a fixed IP address pool when assigning a temporary IP address to a networked computer (client). This is done automatically and saves a lot of network configuration work. The client also obtains other information, such as the gateway address (router) and the IP address of the name server (DNS).
ETHERNET

Specifies a Local Area Network (LAN), which was developed by Xerox, Intel and DEC in the 70’s. The bus access process takes place according to the CSMA/CD method.

ETHERNET Standard

In 1983 ETHERNET was standardized by IEEE 802.3 10Base-5. ISO took over the standardization in the ISO Standard 8802/3. The essential differences between ETHERNET and the IEEE standard are to be found in the frame architecture and treatment of pad characters.

Fieldbus

System for serial information transmission between devices of automation technology in the process-related field area.

Firewall

Collective term for solutions which protect LANs connection to the Internet from unauthorized access. They are also able to control and regulate the traffic from the LAN into the Internet. The crucial part of firewalls are static routers which have an access control list used to decide which data packets can pass from which subscriber.

Frame

Unit of data transferred at the Data-Link layer. It contains the header and addressing information.

FTP

(File Transfer Protocol) A standard application for TCP/IP which allows users on one machine to transfer files to/from another.

Function

Module that always returns the same result (as a function value), prerequisite being identical input values; it has no local variables that store values beyond an invoke.
**Gateway**

Device for connecting two different networks. It converts the different protocols.

**Header**

A portion of the data packet, containing, among others, the address information of the receiver.

**Host computer / Subscriber**

Originally used to describe a central mainframe computer accessed from other systems. The services provided by the subscriber can be called up by means of local and remote request. Today, this term is also used to refer to simple computers which provide particular central Services (i.e. UNIX-Subscribers on the Internet).

**HTML**

Abbreviation of hypertext markup language

HTML is the description language for documents on the World Wide Web. It contains language elements for the design of hypertext documents.

**HTTP**

(Hyper Text Transfer Protocol) client server TCP/IP protocol which is used on the Internet or Intranets for exchanging HTML documents. It normally uses port 80.

**Hub**

A device which allows communication between several network users via twisted pair cable.

Similar to a repeater, but with many outputs, a hub is used to form a star topology.

**Hypertext**

Document format used by HTTP. Hypertext documents are text files which allow links to other text documents via particularly highlighted keywords.
ICMP-Protocol

TA protocol for the transmission of status information and error messages of the IP, TCP and UDP protocols between IP network nodes. ICMP offers, among others, the possibility of an echo (ping) request to determine whether a destination is available and is responding.

IEEE

Institute of Electrical and Electronic Engineers.

IEEE 802.3

IEEE 802.3 is a IEEE standard. ETHERNET only supports the yellow cable as a medium. IEEE 802.3 also supports S-UTP and wideband coaxial cable. The segment lengths range from 500 m for yellow cable, 100 m for TP and 1800 m for wideband coaxial cable. A star or a bus topology is possible. ETHERNET (IEEE 802.3) uses CSMA/CD as a channel access method.

Intel format

Set configuration of the fieldbus coupler / controller to establish the process image. In the coupler/controller memory, the module data is aligned in different ways, depending on the set configuration (Intel/Motorola-Format, word-alignment,...). The format determines whether or not high and low bytes are changed over. They are not changed over with the Intel format.

Internet

A collection of networks interconnected to each other throughout the world. Its most well known area is the World Wide Web.

Intranet

A network concept with private network connections over which data can be exchanged within a company.

IP

Internet Protocol. The connectionless network layer, which relies on upper protocols to provide reliability.

ISO/OSI-Reference Model

Reference model of the ISO/OSI for networks with the objective of creating open communication. It defines the interface standards of the respective software and hardware requirements between computer manufacturers. The model treats communication removed from specific implementations, using seven layers.
**LAN**

Local Area Network

**Manchester encoding**

In this encoding system, a 1 is encoded as a transition from low to high and a 0 as a transition from high to low.

**MIB**

Short form for "Management Information Base". MIB is a selection of information on all parameters, which can be handed over to the management software with a request via SNMP. Thus can be made remote maintenance, a monitoring and a control of nets by SNMP protocol.

**Open MODBUS/TCP Specification**

Specification which establishes the specific structure of a MODBUS/TCP data packet. This is dependant upon the selected function code.

**Operating system**

Software which links the application programs to the hardware.

**Ping command**

When a ping command (ping <IP address>) is entered, the ping program *ICMP* generates echo request packets. It is used to test whether a node is available.

**Port number**

The port number, together with the IP address, forms an unambiguous connection point between two processes (applications).

**Predictable ETHERNET**

The delay time of a message on an ETHERNET network can be predicted. The measures which have been taken in predictable ETHERNET make it virtually possible to realize realtime requirements.
Proxy gateway

A proxy gateway (or proxy server, too) allows systems which do not have direct access to the Internet, indirect access to the network. These can be systems which are excluded from direct access by a firewall for security reasons.

A proxy can filter out individual data packets between the Internet and a local network to increase security. Proxies are also used to limit access to particular servers.

In addition, proxy gateways can also have a cache function, in which case they check whether the respective URL address is already available locally and return it immediately, if necessary. This saves time and costs when there are multiple accesses. If the URL is not in the cache, the proxy forwards the request as normal.

The user should not notice the proxy gateway apart from the single configuration in the web browser. Most web browsers can be configured so that they use different or no proxy gateways per access method (FTP, HTTP).

Repeater

Repeaters are physical amplifiers without their own processing function. They refresh data without detecting damaged data and forward all signals. Repeaters are used for longer transmission distances or when the maximum number of nodes of 64 devices per twisted pair segment is exceeded. A request from a client to server is a provision to act on a service or function call.

Request

A service request from a client which requests the provision of a service from a server.

Response

The server’s reply to a client’s request.

RFC specifications

Specifications, suggestions, ideas and guidelines regarding the Internet are published in the form of RFCs (Request For Comments).

RJ45 connector

Also referred to as a Western connector. This connector allows the connection of two network controllers via twisted pair cables.
Router

Connects neighboring subnets, the router operating with addresses and protocols of the third ISO/OSI layer. As this layer is hardware independent, the routers allow transition to another transmission medium. To transmit a message the router evaluates the logical address (source and destination address) and finds the best path if there are several possibilities. Routers can be operated as repeaters or bridges.

Routing

Method of selecting the best path over which to send data to a distant network.

SCADA

Abbreviation for Supervisory Control and Data Acquisition. SCADA software is a program for the control and visualization of processes.

Segment

Typically, a network is divided up into different physical network segments by way of routers or repeaters.

Server

Device providing services within a client/server system. The service is requested by the Client.

Service

Object operation (read, write).

SMTP

Short form for „Simple Mail Transfer Protocol“. Standard protocol, with which E-mails are sent away in the internet.

SNMP

Short form for „Simple Network Management Protocol“. SNMP serves remote maintenance of servers. Thus leave themselves e.g. rout directly from the office of the network carrier out to configure, without someone must drive for this to the customer.
SOAP

Short form for “Simple Object Access Protocol“. XML is a standard for Meta data, the access on the XML objects takes place via SOAP. The standard defines, how transactions via internet and XML can be done and how dynamic Web services over distributed networks can be used.

Socket

Is a software interface introduced with BSD-UNIX for inter-process communication. Sockets are also possible in the network via TCP/IP. As from Windows 3.11, they are also available in Microsoft operating systems.

STP

With the STP cable (Shielded twisted pair) it acts around a symmetrical cable with in pairs stranded and protected veins. The classical STP cable is a multi-core cable, whose stranded conductors are isolated. The conductors of the STP cable are individually protected. It has no total screen.

S-STA

Beside the STP cables there is cable, which has total shielding from foil or network shielding additionally to the single shielding of the conductors still another. These cables are called S/STP cables: Screened/Shielded twisted pair.

Subnet

A portion of a network that shares the same network address as the other portions. These subnets are distinguished through the subnet mask.

Subnet mask

The subnet mask can be used to manipulate the address areas in the IP address room with reference to the number of subnets and subscribers. A standard subnet mask is, for example, 255.255.255.0.

S-UTP

Screened unshielded twisted pair cable which only has one external shield. However, the twisted pair cables are not shielded from each other.
Switch

Switches are comparable to bridges, but with several outputs. Each output uses the full ETHERNET bandwidth. A switch switches a virtual connection between an input port and an output port for data transmission. Switches learn which nodes are connected and filter the information transmitted over the network accordingly. Switches are intelligent devices that learn the node connections and can transfer data at the switch and not have to send it back to the main server.

Switched ETHERNET

The segments of this type of ETHERNET are connected by switches. There are many applications for switching technologies. ETHERNET switching is becoming increasingly popular in local networks as it allows the realization of a deterministic ETHERNET.

TCP

Transport Control Protocol.

TCP/IP Protocol Stack

Network protocols which allow communication between different networks and technologies.

Telnet

The Telnet protocol fulfills the function of a virtual terminal. It allows remote access from the user’s computer to other computer systems on the network.

Traps

Traps are unsolicited messages, which are sent by an agent to a management system, as soon as somewhat unexpected and for the management system interesting happens. Traps is with from the hardware admitted interrupts comparably. A well-known example of a Trap message is the „Blue screen“ with Win95/98.

Twisted Pair

Twisted pair cables (abbreviated to TP).
uClinux
The uClinux (microcontroller Linux) is a Linux kernel for microprocessors and microcontrollers without MMU (Memory Management Unit). In 1996 the project started with kernel version 2.0. Since kernel version 2.6, the major parts have been integrated into the Linux kernel. The development of the C standard library uClibc was also part of the uClinux project. Furthermore, several UserLand programs belong to the uClinux distribution. uClinux runs on many embedded systems. Embedded systems are devices in which a small computer is encapsulated by the device it controls, for example a DSL router, DVD player, etc.

UDP protocol
The user datagram protocol is a transport protocol (layer 4) of the ISO/OSI-reference model which supports data exchange between computers without a connection. UDP runs directly on top of the underlying IP protocol.

URL
Abbreviation for uniform resource locator. Address form for Internet files which are mostly applied within the World Wide Web (WWW). The URL format makes the unambiguous designation of all documents on the Internet possible by describing the address of a document or object which can be read by a web browser. URL includes the transmission type (http, ftp, news etc.), the computer which contains the information and the path on the computer. URL has the following format: Document type//Computer name/List of contents/File name.

UTP
The UTP cable is a symmetrical, not-protected cable with twisted colored wires in pairs. This type of cable, which there is in execution two-in pairs and four-in pairs, is the dominating type of cable in the floor wiring and the terminal wiring.

Web browser
Program for reading hypertext. The browser allows the various documents to be viewed in hypertext and navigation between documents.

Word-alignment
Set configuration of the fieldbus coupler/controller for the creation of a process image. Word-alignment is used to establish the process image word-by-word (2 bytes).

World Wide Web
HTTP server on the Internet.
## Literature List

<table>
<thead>
<tr>
<th>Title</th>
<th>Author</th>
<th>Publisher</th>
<th>Year</th>
<th>ISBN</th>
<th>Part Number</th>
</tr>
</thead>
</table>
14 Index

1

10Base-T · 164

A

Address

  Ethernet · 174
  Hardware · 174
  IP · 175
  TCP · 180

B

BootP · 181
Bridge · 168, 198, 207
Bus access procedure · 174

C

Cable

  Category 5 · 164
  Crossover · 164
  Impedance · 164
  -length · 90
  Parallel · 164
  carrier rail · 16, 19
  contacts
    data · 20
    power · 27
  Coupler modules · 168
  Crossover cable · 164
  CSMA/CD · 174

D

data contacts · 20
Data packet · 174
  Ethernet · 174
  IP · 178
  TCP · 180
  TCP/IP · 180
  Data security · 169
  Delay time · 169
  Diagnosis information · 161

E

Electrical Isolation · 42
Error
during start-up · 78
Error
-message · 82
Ethernet · 161
  Address · 174
  Industry · 169
  Network · 168
  Network architecture · 162
  Predictable · 170
  Shared · 169
  Standard · 163, 174
  Switched · 170
  Ethernet standard · 163, 165

F

Fieldbus node
  Architecture · 162
  Firewall · 169

G

Gateway · 168, 178, 201, 204

H

HTML pages · 182
HTTP · 182
hub · 162, 164, 168, 169
Hub · 90, 165, 201

I

I/O Modules · 130
Internal bus
  -error · 89
  Internet · 162, 169, 177, 182
  Intranet · 169
  IP address · 82, 175, 176, 178, 181
    Key data · 176

L

LED
  blink code · 78
  red · 78
  LEDs · 43
  locking disc · 18

M

MAC-ID · 46, 174
Modbus/TCP
  Specification · 203

N

Network card · 162, 164, 178
Network class · 175
node
  max. number · 176

O

Operating mode
  STOP · 45

P

Ping command · 82, 203
Port number · 180, 204
Port number 80 · 182
Power contacts · 21, 27
  not carried out · 28
Predicatable Ethernet · 170
Process
  Visualisation · 162
  Process data · 161
  Protocols · 161
  BootP · 176
  Proxy · 204

R

Realtime behaviour · 169
Realtime requirements · 170
Repeater · 162, 168, 204
Request · 204
Response · 204
Router · 168, 169, 174, 175, 200, 205
S
SCADA · 205
Segment length · 163
Sequence number · 179
Shared Ethernet · 169
Socket · 206
Standard
   Cabling · 167
      EN 50173, ISO 11801, TIA 568-A · 167
Standardization
   IEEE 802.3 · 161
Structured cabling · 167
Subnet · 178, 206
   mask · 177, 206
Subscriber ID · 177
Switch · 162, 168, 207
Switched Ethernet · 170, 199, 207

T
TCP/IP · 161
Topology · 163, 165, 167, 168, 201, 202
   Star · 165
Transmission
   media · 163
   rate · 161
   speed · 163

U
unlocking lug · 18

W
word-alignment · 137
WWW · 182