Relevant products

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Model</th>
<th>Part number</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCAN-RS-232</td>
<td>Plastic casing, screw terminal block</td>
<td>IPEH-002100</td>
</tr>
</tbody>
</table>

PCAN® is a registered trademark of PEAK-System Technik GmbH. All other product names mentioned in this document may be the trademarks or registered trademarks of their respective companies. They are not explicitly marked by “™” or “®”.

Copyright © 2019 PEAK-System Technik GmbH
Duplication (copying, printing, or other forms) and the electronic distribution of this document is only allowed with explicit permission of PEAK-System Technik GmbH. PEAK-System Technik GmbH reserves the right to change technical data without prior announcement. The general business conditions and the regulations of the license agreement apply. All rights are reserved.

PEAK-System Technik GmbH
Otto-Roehm-Strasse 69
64293 Darmstadt
Germany

Phone: +49 (0)6151 8173-20
Fax: +49 (0)6151 8173-29

www.peak-system.com
info@peak-system.com

Document version 1.3.0 (2019-05-27)
Contents

1 Introduction 5
  1.1 Properties at a Glance 5
  1.2 Scope of Supply 6
  1.3 Prerequisites for Operation 6

2 Connectors and Coding Solder Jumpers 7
  2.1 Screw Terminal Block 8
  2.2 J5 Connector Panel: JTAG Ports 8
  2.3 Coding Solder Jumpers 10

3 Operation 12

4 Software 13
  4.1 Installing the GNU ARM Toolchain 13
  4.2 Library 14
  4.3 Firmware Examples 14
    4.3.1 Compiling a Firmware Example 15

5 Firmware Upload 16
  5.1 Uploading Firmware via CAN 16
    5.1.1 System Requirements 16
    5.1.2 Preparing Hardware and Software 17
    5.1.3 Uploading the Firmware 18
  5.2 Uploading Firmware via the Serial Connections 21

6 Technical Specifications 23

Appendix A CE Certificate 25

Appendix B Dimension Drawings 26
Appendix C  Port Assignment of the Microcontroller
1 Introduction

The PCAN-RS-232 converter has one CAN channel and one RS-232 port. The data traffic between CAN and RS-232 is controlled by a freely programmable microcontroller. Thus, devices without CAN connection (e.g. PLC systems) can be connected to a CAN bus via almost any RS-232 protocol.

The provided library and Yagarto toolchain can be used to produce your own firmware and then transfer it to the converter via CAN. This gives a whole range of options for manipulation, evaluation, filtering, and routing of the data traffic.

On delivery the PCAN-RS-232 converter is supplied with an example firmware that routes from CAN to RS-232 and vice versa. It allows to configure the data transfer as well as the hardware with serial control commands. The corresponding source code is included.

1.1 Properties at a Glance

- Microcontroller of the NXP LPC21 series (16/32-bit ARM CPU)
- External 32-kByte EEPROM
- High-speed CAN channel (ISO 11898-2) with 40 kbit/s to 1 Mbit/s (lower bit rates on request)
- Data transfer between CAN and RS-232 with a maximum bit rate of 115,200 bit/s
- Additional digital input (Low-active) and output (Low-side switch)
- Duo LED for status indication
- 10-pole screw terminal block (Phoenix), exposed connector contacts on request
Supply voltage from 8 to 30 V
Extended operating temperature range from -40 to +85 °C
(-40 to +185 °F)
Import of a new firmware via CAN
4-bit coding of the hardware by solder jumpers
Plastic casing

1.2 Scope of Supply
PCAN-RS-232 in a plastic casing
10-pin screw terminal block
Windows development software (Yagarto GNU ARM toolchain, flash program)

1.3 Prerequisites for Operation
Power supply in the range of 8 to 30 V DC
For uploading a new firmware via CAN:
  • CAN interface of the PCAN series for the computer (e.g. PCAN-USB)
  • Operating system Windows 10, 8, 7 (32/64-bit)
2 Connectors and Coding
Solder Jumpers

The PCAN-RS-232 converter has a 10-pole screw terminal block for connecting the following components:

- Supply voltage
- CAN
- RS-232
- Digital input and digital output
- CAN bootloader activation

For direct access to the debugging ports (JTAG) of the microcontroller, an additional – yet not equipped – connector panel is available on the circuit board of the converter.

Furthermore the board has four coding solder jumpers in order to assign a fixed status to the corresponding input bits of the microcontroller. A concrete application is to identify a PCAN-RS-232 converter on the CAN bus for a firmware upload, especially if there are several converters connected and in operation.

The following subsections describe each connector assignment.
2.1 Screw Terminal Block

![Screw terminal block image]

Figure 1: Screw terminal block

<table>
<thead>
<tr>
<th>Terminal</th>
<th>Identifier</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+Vb</td>
<td>Power supply 8 - 30 V DC</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>3</td>
<td>CAN_L</td>
<td>Differential CAN signal</td>
</tr>
<tr>
<td>4</td>
<td>CAN_H</td>
<td>Differential CAN signal</td>
</tr>
<tr>
<td>5</td>
<td>DOut</td>
<td>Digital output, Low-side switch</td>
</tr>
<tr>
<td>6</td>
<td>DIn</td>
<td>Digital input, Low-active</td>
</tr>
<tr>
<td>7</td>
<td>Boot CAN</td>
<td>CAN bootloader activation, High-active</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>9</td>
<td>RS-232 RxD</td>
<td>RS-232 interface</td>
</tr>
<tr>
<td>10</td>
<td>RS-232 TxD</td>
<td>RS-232 interface</td>
</tr>
</tbody>
</table>

For further connection details that are not needed for programming of the PCAN-RS-232 converter because of implementation in a library, see also Appendix C Port Assignment of the Microcontroller on page 27.

2.2 J5 Connector Panel: JTAG Ports

The unpopulated connector panel J5 on the PCAN-RS-232 converter’s circuit board provides an access option to the JTAG...
ports of the LPC2194/01 microcontroller (μC) for hardware debugging.

If constant internal pull-down wiring of the TCK or RTCK signals is not suitable for your purposes, you can remove the respective pull-down resistor on the circuit board of the PCAN-RS-232 converter by soldering it out.
2.3 Coding Solder Jumpers

The four positions for coding solder jumpers (ID0 - ID3) are each assigned to one port of the LPC2194/01 microcontroller (μC).
The status of the ports is relevant in the following cases:

- The loaded firmware is programmed so that it reads the status at the corresponding ports of the microcontroller. For example, the activation of certain functions of the firmware or the coding of an ID is conceivable here.

- For a firmware upload via CAN the PCAN-RS-232 converter is identified by a 4-bit ID which is determined by solder jumpers. A bit is set (1) when the corresponding solder jumper position is open (default setting: ID 15, all positions open).

<table>
<thead>
<tr>
<th>Position</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary digit</td>
<td>0001</td>
<td>0010</td>
<td>0100</td>
<td>1000</td>
</tr>
<tr>
<td>Decimal equivalent</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

See also section 5.1 *Uploading Firmware via CAN* on page 16.
3 Operation

The PCAN-RS-232 converter is activated by applying the supply voltage to the respective input pins (see chapter 2 Connectors and Coding Solder Jumpers on page 7). The firmware in the flash memory is subsequently run.

The status indication of the LED depends on the used firmware.

On delivery the PCAN-RS-232 is supplied with an example firmware that routes from CAN to RS-232 and vice versa. It allows to configure the data transfer as well as the hardware with serial control commands.

The LED is green when the converter is switched on, and blinks during an established connection to a serial host (e.g. the shareware program CANHacker).

You can find the documentation about the example firmware on the supplied DVD in the directory branch

/Develop/Microcontroller hardware/PCAN-RS-232/Example/

and there as HTML pages under

6_CAN_TO_SER_BY_COMMAND/help/index.html
4 Software

This chapter covers the installation of the Yagarto GNU ARM tool-chain and gives notes about the software library and the firmware examples.

Software, source code, and additional information are included on the supplied DVD in the following directory branch:

/Develop/Microcontroller hardware/PCAN-RS-232/

4.1 Installing the GNU ARM Toolchain

To compile the code examples and the custom firmware code under Windows, install Yagarto on your computer. Yagarto is a collection of tools to develop applications for ARM processors and micro-controllers on Windows platforms. The collection includes the GNU GCC compiler for C and C++, Make, and further tools. Further information about Yagarto: www.yagarto.de

System requirement: Windows 10/8.1//7 (32/64-bit)

Do the following to install Yagarto:

1. From the directory branch on the provided DVD mentioned above, change to the Compiler subdirectory.

   The directory contains the two installation programs yagarto-*.exe and yagarto-tools-*.exe.

2. Execute the first installation program and follow its instructions.

   If you don’t want to use the default destination folder, make sure that your customized path doesn't contain any spaces. Otherwise compile operations will not work later.
3. Afterwards, execute the second installation program and follow its instructions.

In the system environment, the installation programs create search paths for the executable files. These new search paths are effective only for programs and command prompts that are started afterwards.

4.2 Library

The development of applications for the PCAN-RS-232 is supported by the library libPCAN-RS-232-GNU*s.a (* stands for version number), a binary file. You can access all resources of the converter by means of this library. The library is documented in the header files (*.h). The files are located in each example directory.

4.3 Firmware Examples

On the DVD, the Example subdirectory contains source code for several firmware examples that you can use and test directly and that you can reuse for custom firmware.

On delivery the PCAN-RS-232 converter is supplied with the example firmware 6_CAN_TO_SER_BY_COMMAND that routes from CAN to RS-232 and vice versa. It allows to configure the data transfer as well as the hardware with serial control commands.
4.3.1 Compiling a Firmware Example

Do the following to compile a firmware example under Windows:

1. From the provided DVD, copy the subdirectory of the desired example from the Example directory to the local hard disk.

2. Open a command prompt by using the Windows Start menu. Alternatively you can press the key combination \( \text{Ctrl} + \text{R} \) and enter \texttt{cmd.exe} as program to be executed.

3. At the command prompt change to the previously copied directory.

4. Execute the following command in order to clean-up the target directories (e.g. \texttt{.out}) from files that have been generated earlier:

   \texttt{make clean}

5. Execute the following command to compile the example firmware:

   \texttt{make all}

   If the compiler has finished without errors (“Errors: none”), you can find the firmware file with the extension \texttt{.bin} in the subdirectory \texttt{.out}. This file is then used for firmware upload to the PCAN-RS-232 converter.
5 Firmware Upload

The microcontroller in the PCAN-RS-232 converter can be equipped with new firmware in two different ways:

- Via CAN. The scope of supply includes the Windows program PCAN-Flash to transfer the firmware from the computer to the PCAN-RS-232 converter. This is the recommended method for a firmware upload.

- Via the serial ports of the microcontroller (only in special cases). For this the access to the circuit board of the PCAN-RS-232 converter is needed.

5.1 Uploading Firmware via CAN

5.1.1 System Requirements

The following prerequisites must be given, so that the PCAN-RS-232 converter can be updated with new firmware:

- CAN interface of the PCAN series for the computer (e.g. PCAN-USB)
- CAN cabling between the CAN interface and the PCAN-RS-232 converter with proper termination (120 Ω on each end of the CAN bus)
- Operating system Windows 10/8.1/7 (32/64-bit)
- If you want to update several PCAN-RS-232 converters connected to the same CAN bus, you must assign a unique ID to each converter. See section 2.3 Coding Solder Jumpers on page 10.
5.1.2 Preparing Hardware and Software

Perform the following steps for preparation of the hardware:

1. Switch the PCAN-RS-232 converter off by disconnecting it from the power supply.
2. Establish a connection between “Boot CAN” and “U_b” at the connectors of the converter.

![Figure 5: Connection at the screw terminal block between terminals 1 and 7](image)

This measure later applies the “Boot CAN” connection with a high level.
3. Connect the CAN bus of the converter with a CAN interface connected to the computer. Pay attention to the proper termination of the CAN cabling (2 x 120 Ω).

Perform the following steps for preparation of the software:

1. On the supplied DVD, change to the following directory:
   /Develop/Microcontroller hardware/PCAN-RS-232/
2. Copy the subdirectory PcanFlash to the local hard disk.
   The contained Windows software that copies the Firmware via CAN (PcanFlash.exe) can only be started from a data carrier that is writable.
5.1.3 Uploading the Firmware

The process of transferring a new firmware to the PCAN-RS-232 converter is as follows:

1. Ensure that a connection is established between the “Boot CAN” and “Ub” connections of the converter (details: see above).

2. Switch on the converter by applying a voltage supply.
   
   Due to the High level at the “Boot CAN” connection, the converter starts the CAN bootloader. This is indicated by the LED blinking orange.

3. Run the program PcanFlash.exe under Windows from the local hard drive.

4. Click on the (Options) button in order to call up the dialog box.

5. From the Hardware Profile dropdown list, select the PCAN-RS-232 entry.
6. Click on the ... button next to the **File name** field in order to select the desired firmware file (*.bin) to be uploaded.

7. Click on the **OK** button.

8. Make sure that the PCAN-Flash program is connected with 500 kbit/s to the available CAN interface at the computer.
If not, click the ✪ (Connect) button in order to change the selection in the according dialog box.

![Connect dialog box](image)

**Figure 8: Connect dialog box for selecting a CAN interface**

9. Click the ✪ (Detect) button in order to detect the PCAN-RS-232 converter connected to the CAN bus.

An entry for the converter appears in the main window.

![PCAN-Flash](image)

**Figure 9: PCAN-Flash**

10. Select the entry for the PCAN-RS-232 converter.
11. Click the (Program) button in order to start uploading the new firmware to the converter.

Observe the status indication at the bottom of the window. The process was successful if the last message to appear is “Flashing of module(s) finished!”.

12. Disconnect the power supply from the converter.

13. At the converter, disconnect “Boot CAN” from “U_b”.

You can now use the PCAN-RS-232 converter with the new firmware.

### 5.2 Uploading Firmware via the Serial Connections

This section shows how to activate the microcontroller’s bootloader. The actual upload process depends on the upload software used which is supplied by a third party and is not described here.

**Important note:** When uploading a firmware via the RS-232 interface, the CAN bootloader may be overwritten. Afterwards a firmware upload via CAN is not possible anymore.

Do the following to activate the microcontroller’s bootloader:

1. Switch the PCAN-RS-232 converter off by disconnecting it from the power supply.

2. Open the casing of the converter by removing the screws in order to gain access to the circuit board.

3. Shorten the jumper JP4 „sboot“, e.g. with a temporary solder connection.
4. Establish a serial connection to the computer or the programming adapter via the RS-232 connectors RxD (9) and TxD (10).

5. Switch on the PCAN-RS-232 converter by applying a voltage supply.

   Due to the Low level on port P0.14 of the microcontroller, the converter starts the bootloader for serial transfer. The LED stays off.
6 Technical Specifications

**Functionality**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>NXP LPC2194/01 clocked with 60 MHz</td>
</tr>
<tr>
<td>Add-on memory</td>
<td>32 kByte, EEPROM Atmel AT24C32B (via I²C)</td>
</tr>
<tr>
<td>CAN</td>
<td>High-speed CAN ISO 11898-2</td>
</tr>
<tr>
<td></td>
<td>Transceiver NXP TJA1040T</td>
</tr>
<tr>
<td></td>
<td>Bit rates 40 kbit/s - 1 Mbit/s</td>
</tr>
<tr>
<td></td>
<td>(lower bit rates on request)</td>
</tr>
<tr>
<td></td>
<td>No termination</td>
</tr>
<tr>
<td>RS-232</td>
<td>RxD and TxD serial connections with RS-232 levels</td>
</tr>
<tr>
<td>Digital input (Din)</td>
<td>Low-active, max. level $U_b$</td>
</tr>
<tr>
<td>Digital output (Dout)</td>
<td>Low-side switch, max. 60 V/0.7 A</td>
</tr>
<tr>
<td>Status indication</td>
<td>Duo LED</td>
</tr>
<tr>
<td>Connectors</td>
<td>Screw terminal block, 10-pin, pitch 3.5 mm (Phoenix Contact MC 1,5/10-ST-3,5 - 1840447)</td>
</tr>
</tbody>
</table>

**Power supply**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply voltage ($U_b$)</td>
<td>8 - 30 V DC</td>
</tr>
<tr>
<td>Current consumption</td>
<td>max. 70 mA at 12 V</td>
</tr>
</tbody>
</table>

**Measures**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Casing: 68 x 57 x 21 mm (W x D x H)</td>
</tr>
<tr>
<td></td>
<td>Circuit board: 51 x 54 mm (W x D)</td>
</tr>
<tr>
<td></td>
<td>See also dimension drawings in Appendix A on page 25</td>
</tr>
<tr>
<td>Weight</td>
<td>36 g</td>
</tr>
</tbody>
</table>

**Environment**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating temperature</td>
<td>-40 - +85 °C (-40 - +185 °F)</td>
</tr>
<tr>
<td>Temperature for storage and transport</td>
<td>-40 - +85 °C (-40 - +185 °F)</td>
</tr>
<tr>
<td>Relative humidity</td>
<td>15 - 90 %, not condensing</td>
</tr>
<tr>
<td>Ingress protection (IEC 60529)</td>
<td>IP20</td>
</tr>
</tbody>
</table>
## Conformity

<table>
<thead>
<tr>
<th></th>
<th>Directive 2014/30/EU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DIN EN 61326-1:2013-07</td>
</tr>
<tr>
<td>EMV</td>
<td></td>
</tr>
<tr>
<td>RoHS 2</td>
<td>Directive 2011/65/EU</td>
</tr>
<tr>
<td></td>
<td>DIN EN 50581 VDE 0042-12:2013-02</td>
</tr>
</tbody>
</table>
Appendix A CE Certificate

EU Declaration of Conformity

This declaration applies to the following product:
Product name: PCAN-RS-232
Item number(s): IPEH-0021.00
Manufacturer: PEAK-System Technik GmbH
Otto-Roehm-Strasse 69
64293 Darmstadt
Germany

We declare under our sole responsibility that the mentioned product is in conformity with the following directives and the affiliated harmonized standards:

EU Directive 2011/65/EU (RoHS 2)
DIN EN 50581 VDE 0042-12:2013-02
Technical documentation for the assessment of electrical and electronic products with respect to the restriction of hazardous substances;
German version EN 50581:2012

EU Directive 2014/30/EU (Electromagnetic Compatibility)
DIN EN 61326-1:2013-07
Electrical equipment for measurement, control and laboratory use - EMC requirements - Part 1:
General requirements (IEC 61326-1:2012);
German version EN 61326-1:2013

Darmstadt, 22 February 2019

Uwe Wilhelm, Managing Director
Appendix B  Dimension Drawings

The figures do not show the actual size of the product.
Appendix C Port Assignment of the Microcontroller

The following table lists the used inputs and outputs (ports) of the LPC2194/01 microcontroller (μC) and their function in the PCAN-RS-232 converter. It is meant as supplemental information. The converter’s functionality is implemented by the supplied library.

Note: Get additional information about the LPC2194/01 microcontroller on the homepage of NXP (www.nxp.com).

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O</th>
<th>μC function</th>
<th>Signal</th>
<th>Active (μC)</th>
<th>Function/connection¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0.0</td>
<td>O</td>
<td>TxD UART0</td>
<td>TxD0</td>
<td></td>
<td>Serial communication, Transmit, STB:10 (RS-232 levels)</td>
</tr>
<tr>
<td>P0.1</td>
<td>I</td>
<td>RxD UART0</td>
<td>RxD0</td>
<td></td>
<td>Serial communication, Receive, STB:9 (RS-232 levels)</td>
</tr>
<tr>
<td>P0.2</td>
<td>I, O</td>
<td>SCL</td>
<td>SCL</td>
<td></td>
<td>I²C bus to the Atmel AT24C256B EEPROM</td>
</tr>
<tr>
<td>P0.3</td>
<td>I, O</td>
<td>SDA</td>
<td>SDA</td>
<td></td>
<td>Coding solder jumpers on board (ID 0 - 3), bridged = Low</td>
</tr>
<tr>
<td>P0.4</td>
<td>I</td>
<td>Port pin</td>
<td>ID0</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>P0.5</td>
<td>I</td>
<td>Port pin</td>
<td>ID1</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>P0.6</td>
<td>I</td>
<td>Port pin</td>
<td>ID2</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>P0.7</td>
<td>I</td>
<td>Port pin</td>
<td>ID3</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>P0.12</td>
<td>O</td>
<td>Port pin</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>P0.13</td>
<td>I, O</td>
<td>Port pin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P0.14</td>
<td>I</td>
<td>Port pin</td>
<td>/Boot_ser</td>
<td>Low</td>
<td>Activate flashing via serial interface, JP4</td>
</tr>
</tbody>
</table>

¹ STB:n Terminal n on the screw terminal block
J4/5:n Pin n of the respective connector panel on the circuit board
JPx Jumper position on the circuit board (set = active)
<table>
<thead>
<tr>
<th>Port</th>
<th>I/O</th>
<th>μC function</th>
<th>Signal</th>
<th>Active (μC)</th>
<th>Function/connection&lt;sup&gt;1&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0.15</td>
<td>I</td>
<td>Port pin</td>
<td>/Boot_CAN</td>
<td>Low</td>
<td>Activate flashing via CAN with 500 kbit/s, STB:7 (due to wiring High-active)</td>
</tr>
<tr>
<td>P0.17</td>
<td>O</td>
<td>Port pin</td>
<td>V24_en</td>
<td>High</td>
<td>Deactivate the RS-232 component by Low level (activated by default); possibility for energy saving</td>
</tr>
<tr>
<td>P0.19</td>
<td>I</td>
<td>Port pin</td>
<td>Switch</td>
<td>High</td>
<td>Digital input Din, STB:6 (due to wiring Low-active)</td>
</tr>
<tr>
<td>P0.22</td>
<td>O</td>
<td>Port pin</td>
<td>CAN_en_1</td>
<td>Low</td>
<td>Activate the CAN transceiver&lt;sup&gt;2&lt;/sup&gt;</td>
</tr>
<tr>
<td>P0.25</td>
<td>I</td>
<td>RD1</td>
<td>CAN_RxD</td>
<td></td>
<td>CAN receive</td>
</tr>
<tr>
<td>TD1</td>
<td>O</td>
<td>TD1</td>
<td>CAN_TxD</td>
<td></td>
<td>CAN transmit</td>
</tr>
<tr>
<td>P0.28</td>
<td>I</td>
<td>Analog input</td>
<td>V-Power1</td>
<td></td>
<td>Measure voltage U&lt;sub&gt;b&lt;/sub&gt;, maximum value (0x03FF) corresponds to 33.1 V</td>
</tr>
<tr>
<td>P0.29</td>
<td>I</td>
<td>Analog input</td>
<td></td>
<td></td>
<td>Lies on GND</td>
</tr>
<tr>
<td>P0.30</td>
<td>I</td>
<td>Analog input</td>
<td></td>
<td></td>
<td>Lies on 1.8 V (microcontroller supply)</td>
</tr>
<tr>
<td>P1.16</td>
<td>O&lt;sup&gt;3&lt;/sup&gt;</td>
<td>Port pin</td>
<td>Low</td>
<td>LED red</td>
<td></td>
</tr>
<tr>
<td>P1.17</td>
<td>O&lt;sup&gt;3&lt;/sup&gt;</td>
<td>Port pin</td>
<td>Low</td>
<td>LED green</td>
<td></td>
</tr>
<tr>
<td>P1.21</td>
<td>O</td>
<td>Port pin</td>
<td></td>
<td>Low</td>
<td>Digital output Dout, STB:5 (Low-side switch)</td>
</tr>
<tr>
<td>P1.26</td>
<td></td>
<td>JTAG interface</td>
<td>RTCK</td>
<td></td>
<td>Debugging, J5:9</td>
</tr>
<tr>
<td>P1.27</td>
<td></td>
<td>JTAG interface</td>
<td>TDO</td>
<td></td>
<td>Debugging, J5:7</td>
</tr>
<tr>
<td>P1.28</td>
<td></td>
<td>JTAG interface</td>
<td>TDI</td>
<td></td>
<td>Debugging, J5:8</td>
</tr>
<tr>
<td>P1.29</td>
<td></td>
<td>JTAG interface</td>
<td>TCK</td>
<td></td>
<td>Debugging, J5:5</td>
</tr>
<tr>
<td>P1.30</td>
<td></td>
<td>JTAG interface</td>
<td>TMS</td>
<td></td>
<td>Debugging, J5:6</td>
</tr>
<tr>
<td>P1.31</td>
<td></td>
<td>JTAG interface</td>
<td>TRST</td>
<td></td>
<td>Debugging, J5:10</td>
</tr>
</tbody>
</table>

<sup>2</sup> After resetting the microcontroller, the CAN transceiver is deactivated and must be reactivated to use it.

<sup>3</sup> It may occur that the LED glows slightly when the output is inactive. If you would like to prevent this, your firmware must change the port type to input (I). Before switching on the LED again, the respective port type must be set to output (O).