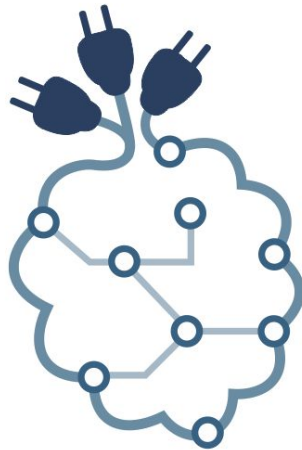


PiXtend V2

Software Manual



PiXtend®



Revision 10/08/18, V1.12

Qube Solutions GmbH
Arbachtalstr. 6, 72800 Eningen, Germany
<https://www.pixtend.com>



PiXtend V2 Software Manual

Version History

Version	Date	Description	Editor
1.00	13/10/2017	Document created, first release	RT
1.01	13/11/2017	CODESYS Package renamed to <i>PiXtend V2 Professional for CODESYS</i>	RT
1.02	20/01/2018	Signaling for error LED L1 added	RT
1.03	26/01/2018	Name of the SD image adapted from "Linux Tools" to "Basic". The Basic Image contains more than just the classic PiXtend Linux tools	TG
1.04	13/02/2018	Information about OpenPLC added	RT
1.05	20/02/2018	CODESYS I/O mapping overview and device parameter description added	RT
1.06	26/02/2018	CODESYS Retain memory usage example introduced	RT
1.07	05/03/2018	Change Python Library V2 link to version 0.1.1, the previous version had a problem where the GPIO PullUps could not be used as intended	RT
1.08	28/03/2018	Raspberry Pi 3 B+ added as a compatible model	RT
1.09	18/07/2018	Information for the <i>PiXtend V2 -L-</i> added	RT
1.10	31/07/2018	- PiXtend V2 -L- Process data + Control- & Statusbytes section added - UG (limited liability) changed to GmbH	TG
1.11	06/08/2018	OpenPLC section updated to version 3 of OpenPLC, including information on PiXtend V2 -L-	RT
1.12	10/08/2018	Added information on how to find out a Raspberry Pi's IP address and how to use the PiXtend V2 DAC in CODESYS	RT



Table of Contents

1. About this Documentation.....	7
1.1. Scope.....	7
1.2. Copyright.....	7
1.3. Names, Brands, Wordmarks and Images.....	8
1.4. Symbols.....	9
2. Important Information.....	10
2.1. Subject to Change.....	10
2.2. Intended Use.....	10
2.3. Technical Condition.....	11
2.3.1. PiXtend V2 -S- model.....	11
2.3.2. PiXtend V2 -L- model.....	12
2.4. Certificates.....	13
2.5. Safety Instructions.....	14
2.6. Disclaimer.....	15
2.7. Contact information.....	15
2.8. Assistance.....	15
3. Overview.....	16
4. Prerequisites.....	16
5. Licenses.....	17
6. Basic knowledge.....	18
6.1. Definition "Safe State".....	18
6.2. SPI Communication, Data Transmission and Cycle Time.....	18
6.3. Retain Memory.....	20
6.4. Preparing a SD card.....	22
6.5. Find network address (IP address) of a Raspberry Pi.....	24
6.6. Raspbian Login Information (Login).....	26
6.7. Turning off the Raspberry Pi.....	27
6.8. Compatibility of the Software Components.....	28
6.9. Enable Serial Interface.....	28
7. CODESYS.....	30
7.1. Information.....	31
7.1.1. Copyright of texts and images:.....	31
7.1.2. Compatibility.....	31
7.2. Prerequisites.....	32
7.2.1. System Requirements for CODESYS.....	32
7.2.2. Required Hardware.....	32
7.3. Installation of the required Software Components.....	33
7.3.1. CODESYS Development System.....	33
7.3.2. CODESYS Control for Raspberry Pi installation.....	35



PiXtend V2 Software Manual

7.3.3. Create a bootable SD-Card for the Raspberry Pi.....	37
7.3.4. PiXtend V2 CODESYS device driver installation.....	38
7.4. Further steps.....	38
7.5. CODESYS – Create a project.....	39
7.5.1. Step by Step to the first PiXtend V2 CODESYS program.....	39
7.5.2. Step by Step to your first CODESYS Webvisu.....	56
7.6. Step by step to your first DAC program.....	60
7.6.1. Create CODESYS Standard Project for PiXtend.....	60
7.6.2. Add SPI device.....	62
7.6.3. Add PiXtend V2 DAC device.....	65
7.6.4. Create Global Variable List.....	67
7.6.5. Mapping of variables.....	68
7.6.6. Task Configuration.....	70
7.6.7. Creating the main program.....	70
7.6.8. Connect to PiXtend V2 and download program.....	71
7.6.9. Create CODESYS Webvisu.....	72
7.7. CODESYS Serial Communication.....	77
7.7.1. Prerequisites.....	77
7.7.2. RS232 Interface - Connecting the cable.....	77
7.7.3. RS485 Interface - Connecting the cable (only PiXtend V2 -L-).....	78
7.7.4. Preparing Linux.....	79
7.7.5. Terminal Program.....	81
7.7.6. CODESYS.....	82
7.8. CAN-Bus Communication.....	85
7.8.1. introduction.....	85
7.8.2. Hardware connection to the CAN bus.....	86
7.8.3. Preparation and test under Linux.....	87
7.8.4. Preparations for CODESYS.....	93
7.8.5. CODESYS project with CANopen.....	96
7.9. CODESYS – PiXtend Retain Memory.....	113
7.9.1. Preparation.....	113
7.9.2. Create program.....	114
7.9.3. Further information.....	117
7.10. CODESYS – Shutting down the Raspberry Pi.....	118
7.11. PiXtend V2 -S- SPI Device.....	121
7.11.1. SPI Device Parameter.....	121
7.11.2. I/O Overview.....	123
7.12. PiXtend V2 -L- SPI Device.....	126
7.12.1. SPI Device Parameter.....	126
7.12.2. I/O Overview.....	128
7.13. FAQ – Frequently Asked Questions and Troubleshooting.....	132
7.13.1. CODESYS Raspberry Pi Runtime and PiXtend V2.....	132
7.13.2. Serial Communication.....	133



PiXtend V2 Software Manual

8. pxdev – Linux Tools & Library.....	135
8.1. Information.....	136
8.1.1. Usage of pixtendtool2(s/l) - Single commands for PiXtend V2.....	136
8.1.2. Usage of pxauto2(s/l) – GUI for PiXtend V2.....	136
8.1.3. Usage of the PiXtend/PiXtend V2 C-Library.....	137
8.2. Prerequisites.....	137
8.3. Usage of the PiXtend V2 SD-Image „Basic“.....	138
8.4. Manual installation of pxdev.....	139
8.4.1. Preparation and Installation.....	139
8.4.2. Enable SPI Bus.....	140
8.5. Working with pixtendtool2(s/l).....	141
8.5.1. Examples.....	142
8.5.2. Further steps.....	144
8.6. Using pxauto2(s/l).....	145
8.6.1. Navigation.....	147
8.6.2. Edit fields.....	147
9. C-Library „pxdev“.....	149
9.1. Prerequisites.....	150
9.2. Preparation.....	151
9.2.1. Create a new folder and a C-file.....	151
9.2.2. Including the libraries.....	152
9.3. Programming.....	152
9.4. Example Program for PiXtend V2 -S-.....	160
9.5. Example Program for PiXtend V2 -L-.....	162
9.6. Compile and run the program.....	164
9.7. Frequently Asked Questions (FAQ).....	165
10. FHEM.....	166
10.1. Prerequisites.....	166
10.2. Installation.....	167
10.2.1. Installation of FHEM.....	167
10.2.2. Integration of the module.....	168
10.2.3. Configuration of the system.....	168
10.3. Description of the module.....	169
10.3.1. Create a new PiXtend V2 device.....	169
10.3.2. Internals.....	169
10.3.3. Set-commands.....	169
10.3.4. Get-commands.....	172
10.3.5. Readings.....	174
10.3.6. Attributes.....	176
10.4. Examples.....	176
10.4.1. Presentation of sensor values.....	176
10.4.2. Debouncing and combining of in- and outputs.....	179
10.4.3. Control servo motors.....	180



PiXtend V2 Software Manual

10.5. Frequently Asked Questions (FAQ).....	181
11. PiXtend Python Library V2 (PPLV2).....	182
11.1. Prerequisites.....	183
11.2. Installation with PiXtend V2 -S- Image.....	184
11.3. Installation on original Raspbian.....	184
11.3.1. Preparation.....	184
11.3.2. PiXtend Python Library V2 installation.....	185
11.4. Programming.....	186
11.4.1. Example Folder.....	186
11.4.2. Execute Example.....	186
11.4.3. Create your own program.....	188
11.4.4. Short Overview.....	189
11.4.5. Start Python automatically at boot.....	191
11.4.6. Using the Serial Interface.....	192
11.4.7. Frequently Asked Questions (FAQ).....	192
12. OpenPLC Project.....	193
12.1. Prerequisites.....	194
12.2. Installation.....	195
12.2.1. Preparation.....	195
12.2.2. OpenPLC Runtime Installation.....	196
12.2.3. Programming Enviroment (IDE).....	198
12.3. Programming.....	199
12.3.1. Hello World Program.....	200
12.3.2. First start and Transfer program to PiXtend.....	201
12.4. Further Information.....	210
12.4.1. Available PiXtend V2 -S- I/Os.....	210
12.4.2. Available PiXtend V2 -L- I/Os.....	211
12.4.3. Handling Tasks in the OpenPLC Editor.....	212
12.4.4. OpenPLC Runtime autostart.....	213
12.5. Frequently Asked Questions (FAQ).....	214
13. FourZero™ Support.....	215
14. Appendix.....	216
14.1. PiXtend V2 -S-.....	217
14.1.1. Process Data.....	217
14.1.2. Control- & Status-Bytes.....	244
14.2. PiXtend V2 -L-.....	268
14.2.1. Process Data.....	268
14.2.2. Control- & Status-Bytes.....	291
14.3. Error LED "L1"-Signals.....	311
15. Figure Index.....	312
16. Table Index.....	315



1. About this Documentation

Save this Documentation!

This documentation is part of the product and is to be kept for the entire duration of the products usage. If the product is passed on or sold, this document must be handed over to the next user, this also include any extensions to this documentation.

1.1. Scope

This documentation applies only to the software components specified in the table of contents and to *PiXtend V2* devices of the following types:

- *PiXtend V2* Extension Board (QS-Order number: 600)
- *PiXtend V2* ePLC Basic (QS-Order number: 601)
- *PiXtend V2* ePLC Pro (QS-Order number: 602)
- *PiXtend V2 -L-* Extension Board (QS-Order number: 620)
- *PiXtend V2 -L-* ePLC Basic (QS-Order number: 621)
- *PiXtend V2 -L-* ePLC Pro (QS-Order number: 622)

Notice:

When the name *PiXtend V2* is found in this document on it's own without any additional letter, the given information is valid for all *PiXtend V2* models (*PiXtend V2 -S-* and *PiXtend V2 -L-* at this moment).

The documents for the predecessor series "PiXtend V1.x" can be found on our homepage at <https://www.pixtend.com>

1.2. Copyright

This documentation, including all texts and pictures, is protected by copyright. The written approval of Qube Solutions GmbH, Eningen u.A., must be obtained for any other use, translation into other languages, archiving or other alteration.

Copyright 2018 © Qube Solutions GmbH






1.3. Names, Brands, Wordmarks and Images

- "Raspberry Pi" and its logo are registered trademarks of the Raspberry Pi Foundation - www.raspberrypi.org
- "CODESYS" and its logo are registered trademarks of 3S-Smart Software GmbH - www.codesys.com
- "PiXtend", "ePLC" and the associated logo are protected trademarks of Qube Solutions GmbH - www.qube-solutions.de
- "AVR", "ATmega" and the associated logo are registered trademarks of Atmel Corporation - www.atmel.com or Microchip Technology Corporation www.microchip.com
- "Debian" and "Raspbian" are protected trademarks of the Debian Project - www.debian.org
- "I2C" or "I²C" are registered trademarks of NXP Semiconductors - www.nxp.com
- "Arduino" is a registered trademark of Arduino AG - www.arduino.cc

The rights of all companies and company names mentioned herein as well as products and product names lie with the respective companies.



1.4. Symbols

	<p>Important NOTE!</p> <p>This symbol identifies important information to avoid malfunctions and accidental misuse.</p>
	<p>Warning!</p> <p>This symbol indicates a warning and can result in damage to property if not observed.</p>
	<p>Danger!</p> <p>This symbol indicates a high hazard which can lead to serious personal injury or death, if not observed.</p>



2. Important Information

This chapter contains information about the subject to change topic, the intended use, the technical condition of the *PiXtend V2* series, the disclaimer, the safety notices and where to get help.

2.1. Subject to Change

Qube Solutions GmbH reserves the right to revise or amend this documentation in whole or in part if this serves the technical progress or if existing software components have to be changed or new ones have been created. The user will always find the latest version of this documentation at <https://www.pixtend.de/downloads/>.

2.2. Intended Use

PiXtend V2 is used in combination with the single-board computer "Raspberry Pi" (Raspberry Pi Foundation, UK registered charity 1129409), which is already included in the product (*PiXtend V2* ePLC Basic / Pro) or has to be bought by the customer as an accessory (*PiXtend V2* Extension Board).

The *PiXtend V2* system fulfills the function of a programmable logic controller (PLC) or an electrical measuring, control, regulating and laboratory device. It can read and evaluate sensors and control actuators. The logic programming of inputs and outputs can be done, among other things, with the software "CODESYS V3" from the company 3S Smart Software Solutions GmbH. Qube Solutions also provides components for other programming languages and systems from the areas of IT and home automation that the customer can use. Instructions and examples have been created by Qube Solutions for this purpose.

The *PiXtend V2* devices are designed for dry indoor environments - protection classes IP20 (ePLC Pro) and IP00 (Extension Board and ePLC Basic). Operation outside and in humid/wet rooms is not permitted, except the *PiXtend* units are installed in a suitable housing. The devices are not designed for hazardous areas or safety critical systems / installations.

PiXtend V2 devices can be used equally in the industrial/commercial environment as in educational facilities and in residential areas.

PiXtend V2 offers the possibility, under certain conditions, of switching hazardous voltages. Work on dangerous voltages is only permitted for qualified personnel (the requirements may differ from country to country). If in doubt, the use of dangerous voltages is prohibited.

Apart from this, *PiXtend* is suitable for all persons aged 14 and over who have read and



PiXtend V2 Software Manual

understood the safety data sheet and the manuals. Training facilities must be supervised by qualified and authorized personnel. Power supplies and accessories used must be approved for the country in which the *PiXtend V2* system is to be installed and used.

2.3. Technical Condition

2.3.1. PiXtend V2 -S- model

PiXtend V2 -S- is supplied with a pre-defined configuration, independent of its model:

- "SPI_EN" Switch activated → Communication between PiXtend & Raspberry Pi is activated
- „PI_5V "Switch activated → PiXtend & Raspberry Pi are supplied by the same voltage regulator on PiXtend (input voltage 12 - 24 V DC). No separate power supply needs to be connected to the Raspberry Pi
- All digital inputs are configured for a 24 V range (no jumper is set)
- All analog inputs are configured for a 0..10 V range (no jumper is set)
- The microcontroller firmware is always the latest version released by Qube Solutions. The current version can be found on our homepage

The ePLC models include additional configurations:

ePLC Basic & Pro

- Contents of SD-Card
 - the content is stated in the order, e.g. "CODESYS Control Demo" or "PiXtend Basic" (in the QS-Shop)
 - Our dealers always receive and ship the SD card "CODESYS Control Demo"

ePLC Pro

- Housing → Stainless steel hood and DIN rail housing pre-assembled

If you need a different version or a different hardware and software combination, please contact us directly (info@pixtend.com).



PiXtend V2 Software Manual

2.3.2. PiXtend V2 -L- model

PiXtend V2 -L- is supplied with a pre-defined configuration, independent of its model:

- "SPI_EN" Switch activated → Communication between PiXtend & Raspberry Pi is activated
- „PI_5V "Switch activated → PiXtend & Raspberry Pi are supplied by the same voltage regulator on PiXtend (input voltage 12 - 24 V DC). No separate power supply needs to be connected to the Raspberry Pi
- All digital inputs are configured for a 24 V range (no jumper is set)
- All analog inputs are configured for a 0..10 V range (no jumper is set)
- "CAN" / "AO" Jumper is set to "AO" → analog outputs active, CAN inactive
- RS485 "A" / "M" Jumper is set to "A" → automatic send/receive switching
- Termination resistors for RS485 and CAN are not set (no jumper)
- The microcontroller firmware is always the latest version released by Qube Solutions. The current version can be found on our homepage

The ePLC models include additional configurations:

ePLC Basic & Pro

- Contents of SD-Card
 - the content is stated in the order, e.g. "CODESYS Control Demo" or "PiXtend Basic" (in the QS-Shop)
 - Our dealers always receive and ship the SD card "CODESYS Control Demo"

ePLC Pro

- Housing → Stainless steel hood and DIN rail housing pre-assembled

If you need a different version or a different hardware and software combination, please contact us directly (info@pixtend.com).



2.4. Certificates



This product has been designed and manufactured in accordance with applicable European directives and is therefore marked with the CE sign. The intended use is described in this document. A safety data sheet is supplied with each product in paper form (multilingual).

Warning:

Changes and modifications to the product, as well as a non-compliance with the information contained in the manuals and safety data sheets, will lead to the loss of certification and approval for the European Economic Area.



WEEE Registration No .: DE 15868016

The symbol of the crossed-out waste bin (WEEE symbol) means that this product must be recycled separately from any household waste as electrical waste. Where you can find the nearest recycling station, ask your local municipal administration.

You can also return your device to us and we will handle the correct disposal for you.



2.5. Safety Instructions



***PiXtend V2* must not be used in safety-critical systems.**

Before use, check the suitability of the Raspberry Pi and *PiXtend V2* for accordance with your application. The default settings have been selected to meet the requirements of most users.



Caution should be exercised when dealing with, and particularly when experimenting with, the process data. Connected sensors and actuators can enter undefined states if incorrectly handled or output incorrect values.

If a machine, a device or a process is controlled or regulated with *PiXtend V2*, dangerous conditions can occur. Be aware of potential hazards at the earliest possible stage.

In case of doubt, disconnect the connections to the devices, sensors, motors, etc. from the power supply in order to minimize the risks to human life and machine.

The control bytes are not stored permanently in the microcontroller. After a reset or power cycle, all previous settings have been cleared and the *PiXtend V2* will not execute any further action until the onboard microcontroller receives a command.



It is recommended to not let the control program for the *PiXtend V2* start automatically while it is in development.



PiXtend V2 Software Manual

2.6. Disclaimer

The information contained in this documentation has been compiled, checked and tested with the software and hardware described herein. Nevertheless, deviations can not be ruled out completely. Qube Solutions GmbH shall not be liable for any damages that may result from the use of the software, software components, hardware or the steps described in this documentation.

2.7. Contact information

Our postal address:

Qube Solutions GmbH
Arbachtalstr. 6
D-72800 Eningen
Germany

How to reach us:

Tel.: +49 (0) 7121 8806920
Fax: +49 (0) 7121 2848662
info@pixtend.com
www.pixtend.com

2.8. Assistance

A lot of information, tips and tricks can be found in our support forum at:

<https://www.pixtend.de/forum/>

If any questions remain unanswered, please first check the FAQ's on our homepage. If the question still is not answered, let us know by e-mail (support@pixtend.com). You will receive an answer as soon as possible.

The latest versions of all documents and software components can be found in the download section of our homepage: <https://www.pixtend.de/downloads/>



3. Overview

PiXtend is a programmable logic controller (PLC) based on the powerful Raspberry Pi single board computer. It's numerous digital and analog inputs and outputs allow connections to a wide range of sensors and actuators found in industrial applications and in the Maker scene. The connection to other devices, controls and computer systems is established via standard serial interfaces (RS232/RS485, Ethernet, WiFi, CAN). All interfaces and I/Os have a robust design and comply with the PLC standard (IEC 61131-2).

4. Prerequisites

The exact prerequisites for hardware and software are listed separately at the beginning of each chapter, as these may vary depending on the software component in question.

In general, for most tasks, the following prerequisites have to be met:

- System requirements for development computer (PC)
 - *Microsoft Windows 7 / 8 / 10 (32 / 64 Bit)*
 - suitable PC hardware for the corresponding Microsoft Windows platform
 - Sufficient disk space to install new programs
 - Applicable rights (e.g., Administrator rights) allowing the installation of new programs
- Required Hardware
 - *PiXtend V2 Board* (www.pixtend.com)
 - Raspberry Pi Model B+ / 2 B / 3 B / 3 B+
 - Standard RJ-45 network cable
- For the first commissioning of a new Raspberry Pi:
 - SD card reader, internal or external
 - SD memory card, recommended min. 4GB (better 8GB)
 - microSD for Raspberry Pi model B+ / 2 B / 3 B / 3 B+
 - optional: HDMI-capable monitor
 - optional: USB keyboard
 - optional: USB mouse



5. Licenses

Linux Tools:

The software components *PiXtend V2* C-Library, *pxauto2s* and *pixtendtool2s* have been published by Qube Solutions GmbH under the [GNU GPL v3 license](#) and can be freely used and modified. The use in own projects as well as in commercial applications is permitted.



Our Linux programs and drivers are based on [Gordon Henderson's WiringPi-Library](#). Observe the respective license terms.

CODESYS:

The "PiXtend V2 Professional for CODESYS Package" for CODESYS V3.5 is provided by Qube Solutions GmbH free of charge and can be used for private and commercial purposes. The package can be freely distributed and used on any number of computers.

A modification of the "PiXtend V2 Professional for CODESYS Package" or one of the PiXtend V2 driver libraries contained therein is not permitted. If you need a customized version of one of the *PiXtend V2* drivers, contact Qube Solutions GmbH.

The CODESYS examples and the RC-Plug library also included in the software package are released under the [GNU GPL v3 license](#) and can be freely used and modified. The use in own projects as well as in commercial applications is permitted.

Our CODESYS programs, devices and libraries are based on the "CODESYS Control for Raspberry Pi" and other software components from [3S-Smart Software Solutions GmbH](#). The corresponding license terms of the company 3S-Smart Software Solutions GmbH & CODESYS apply.

FHEM:

The provided sources for the FHEM system have been published by Qube Solutions GmbH under the [GNU GPL v3 license](#). Our programs and drivers use the [WiringPi-Library](#) by [Gordon Henderson](#) and [FHEM](#) modules by [Rudolf König](#). The license terms of the respective copyright owner apply.



6. Basic knowledge

6.1. Definition "Safe State"

The *PiXtend V2* has a user activatable Watchdog. If the user has activated it and a communication failure occurs or the SPI bus is interrupted, the Watchdog triggers after the user-defined waiting time. If the Watchdog triggers, the microcontroller stops its work and switches to a defined state. This state is referred to as "Safe State" and has the following characteristics:

- All digital outputs & relays are switched off / put into the rest /idle state
- PWM outputs are switched to high-impedance (tri-state)
- Retain data is stored when Retain option is activated
- The "L1" status LED flashes after a fault if the LED was not deactivated (Error LED "L1"-Signals, page 311)
- The microcontroller or the PiXtend system must be restarted afterwards

If switching off the digital outputs does not correspond to the safe state of your application, external measures must be taken to prevent any critical or dangerous situation.

6.2. SPI Communication, Data Transmission and Cycle Time

Communication between the Raspberry Pi and the *PiXtend V2* (microcontroller) is done by using the SPI interface and with a standard transmission speed of 700 kHz. The consideration of the time required for the data transmission alone is not sufficient to be able to determine how fast the communication with the microcontroller can be done, i.e. how fast can the next transmission and the next and the next occur after the first SPI data transmission has completed. We call this "Cycle Time".

After each data transfer, the microcontroller needs some time to process the received data and only then can the next data transmission take place, otherwise it can be that the microcontroller does not accept any new data.



PiXtend V2 Software Manual

The following cycle times are recommended for the *PiXtend V2 -S-*:

Name	Time	Comment
Minimal	2.5 ms	The fastest possible cycle time, going below this limit will result in errors during communication. No DHT11 / 22 sensors can be used.
Optimized	10 ms	Recommended cycle time if no DHT11 / 22 sensors are used.
Default	30 ms	Recommended cycle time when all functions are used.

The following cycle times are recommended for the *PiXtend V2 -L-*:

Name	Time	Comment
Minimal	5 ms	The fastest possible cycle time, going below this limit will result in errors during communication. No DHT11 / 22 sensors can be used.
Optimized	10 ms	Recommended cycle time if no DHT11 / 22 sensors are used.
Default	30 ms	Recommended cycle time when all functions are used.

The “Default” cycle time has been preset in all software components so that all functions are available to the user without restriction.

The cycle time can be changed by the user, but the necessary steps for such a change are different for each software component.

The Linux tools for the *PiXtend V2*, for example, must be adapted in the source code, as well as the PiXtend Python Library V2.

All CODESYS users, however, can simply change the interval of the corresponding task and see the effects immediately after the next login.



The CODESYS driver of the *PiXtend V2* has a protective mechanism which prevents the driver from communicating with microcontroller too fast.

If the “**Cycle Guard**” detects a Bus Cycle interval that is too short, a “BusCycleError” is shown in the IO mapping of *PiXtend V2*, and no communication with the microcontroller takes place. In this case, select a longer (slower) interval for the Bus Cycle Task.



6.3. Retain Memory

Retain Memory is a special memory on the *PiXtend V2*, it allows the user to maintain values of variables in the event of a power failure, and to resume work after a restart. A technology developed by Qube Solutions allows the Raspberry Pi to utilize this special memory through the PiXtend V2. Such a memory technique is mainly a known function in automation technology, which is now also available to every *PiXtend V2* user, no matter which software component is used.

The PiXtend microcontroller continuously monitors the input supply voltage ("VCC"). If the Retain System is activated and the supply voltage falls below 19 V DC, the controller changes to the Safe State and stores the Retain Data securely. During the time of the detection of the undervoltage to the complete storage of all data, *PiXtend V2* and the Raspberry Pi are supplied with energy from an internal capacitor.

The Retain system can only be activated and used if a nominal supply voltage of 24 V DC is used. If, for example, PiXtend is operated with 12 V DC, the Retain Memory can not be activated and used.

Property	Value	Comment
Type of memory	Flash EEPROM	in the PiXtend microcontroller
Number of storage cycles	min. 10.000	
Memory size	- 32 Bytes (<i>PiXtend V2 -S-</i>) - 64 Bytes (<i>PiXtend V2 -L-</i>)	
Voltage threshold	19 V DC (+/- 0,6 V)	between VCC and GND
Rated supply voltage	24 V DC	to use Retain Memory
Max. Input power via VCC	10 W	for the guaranteed storage of all data

Table 1: Technical specifications – Retain System

For more information on the Retain System, see the corresponding *PiXtend V2* hardware manual.



PiXtend V2 Software Manual

Typical applications for Retain Memory are among others:

- Hour meters
- Device configuration / parameters
- Language settings of the user interface
- motor position
- Component or work piece counter
- Data flow protection
- Maintaining the current work step
- and much more.



PiXtend V2 Software Manual

6.4. Preparing a SD card

If you have ordered a pre-installed SD card from our shop, you can start right away using *PiXtend V2* and creating your own SD card is not necessary.

However, if you want to create a SD card for your Raspberry Pi nevertheless, you will find information on how to do this in this chapter.

We provide various SD card images in our download area at <https://www.pixtend.de/downloads>. Depending on the type of image you want, the "Images" always include a current version of the Raspbian operating system and also one of the following preconfigured options:

- **Basic Image** with PiXtend Linux Tools including the PiXtend C-library (pxdev), PiXtend Python Library V2 with examples, Node-RED and OpenPLC
- **CODESYS Control Demo** with CODESYS V3.5 Runtime Extension for the Raspberry Pi & PiXtend

Always use a SD card or SD image from Qube Solutions for your first tests. Our images are tested extensively and are pre-configured with all required settings. This way you will achieve the fastest progress.

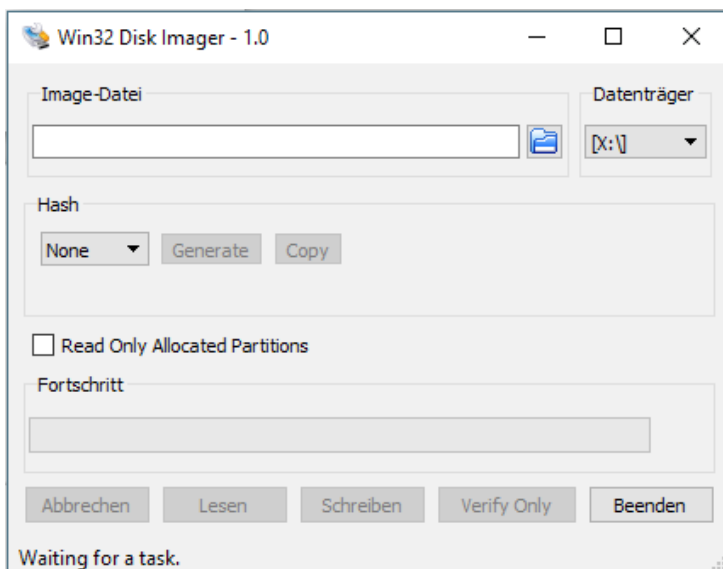


Figure 1: Software - Win32DiskImager

To transfer one of our images to a blank SD card, a PC with a SD card reader is required. We recommend the open source program "Win32DiskImager" as software for writing SD cards. The program runs under Microsoft Windows and can be downloaded free of charge from the following address:

<http://sourceforge.net/projects/win32diskimager/>



PiXtend V2 Software Manual

Notice:



The program Win32DiskImager needs to be run with Administrator rights or under an Administrator account to get access to local drives, otherwise there is a possibility that the SD card is not written to at all or is written incorrectly!

To create a SD card or transfer one of our images to your own SD card, perform the following steps:

- Insert the SD card into the card reader
- Wait for Windows to recognize the media
- Open the workspace/This PC or Windows Explorer and note the drive letters of the SD card, this must be specified later on in the Win32DiskImager program
- The zip file from our download area, in which the SD card image is located, must be opened and unpacked. Store the image file at an easy to reach place, i.e. the desktop
- Start the Win32DiskImager program with Administrator rights
- Select the unzipped image file (*.img) in Win32DiskImager under "Image file"
- Under Disk, select the drive letter of the SD card

Warning:



If you select the wrong disk, all data can be lost as soon as you have started the writing process in the Win32DiskImager program.

It is best to remove all removable media (such as USB sticks or USB hard disks) before you start the writing process, maybe even before you start the Win32DiskImager program.

- Once you have made all settings, start the writing process with a click on "Write"
- Wait until writing has completed



PiXtend V2 Software Manual

After a successful transfer of the image to the SD card, it can be ejected from the PC and plugged into the SD card slot of the Raspberry Pi.

If you see windows opening during or after the creation of the SD card, with or without content, or an error message appears that says that a drive can not be accessed, or you are prompted to format the media, on these dialogs always click on the "Cancel" button and close any windows that may have appeared.

This is no error situation. You get these messages because Microsoft Windows is not able to read the SD card because it is intended for the Raspbian operating system. As mentioned, remove the SD card from the PC and insert it into the SD card slot of the Raspberry Pi. Turn on the power and your Raspberry Pi starts the Raspbian operating system.

6.5. Find network address (IP address) of a Raspberry Pi

After creating a bootable SD card for the Raspberry Pi and its successful launch, a question often arises, which network address (IP address) does the Raspberry Pi have? This applies to all those who want to run the Raspberry Pi without a screen and keyboard, so called *headless* operation.

The factory setting of the network interface is always:

DHCP *Dynamic Host Configuration Protocol*

An automatic assignment of the network address by the house or company's own Internet router or DHCP server of the IT department.

There are several ways to find out the IP address of the RPi.

1. For everyone:
For one time only connect screen and keyboard and quickly enter the command:

```
ifconfig
```

on the Raspberry Pi check which IP address it has. Most Internet routers remember the so-called MAC address (device address) of the RPi and assign it the same IP address each time it boots. After that, the screen and keyboard can be removed and we can continue on the PC using an SSH client for example (e.g., Putty).



PiXtend V2 Software Manual

2. For CODESYS developers:

If you want to use the Raspberry Pi and especially *PiXtend V2* as a controller (PLC), you can use the Raspberry Pi search function in CODESYS built into the Raspberry Pi Update window by 3S Smart Software Solutions. This search function searches the entire home or corporate network for Raspberry Pi computers. If one or more devices were found, then all IP addresses of the found devices appear in a list. In the list you can either select the desired RPi and then write down the IP address or, for example, get information about the system.

Please note: The Raspberry Pi Update window can only be opened if the "*CODESYS Control for Raspberry Pi*" package is installed in CODESYS. If necessary, also look into the chapter 7.3.2 *CODESYS Control for Raspberry Pi installation*.

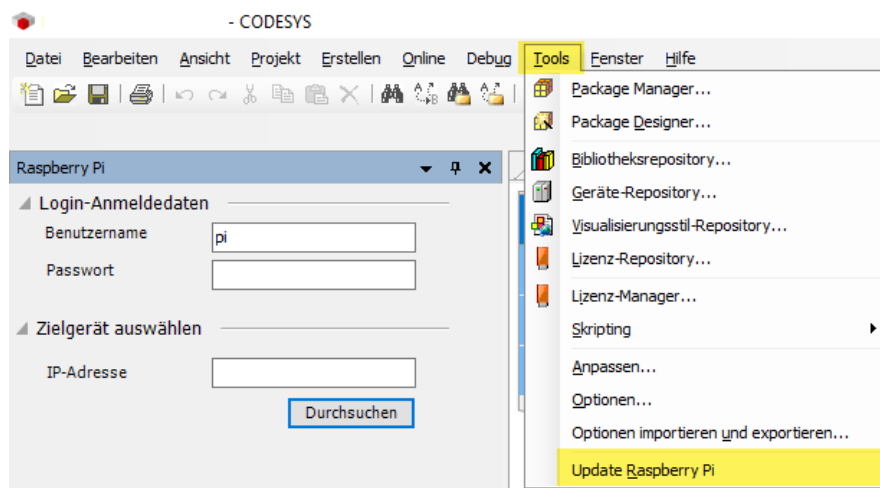


Figure 2: Basic knowledge - Find RPi network address

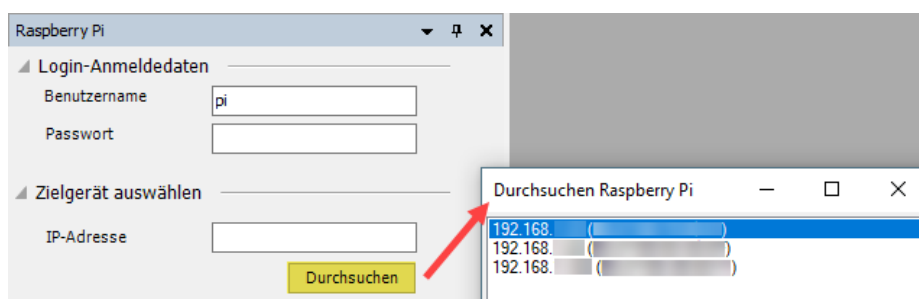


Figure 3: Basic knowledge – All found Raspberry Pi's shown in one list



PiXtend V2 Software Manual

If it is not clear which RPi is which, then it is best turn off all but the one you want to work with, close CODESYS, reboot the Gateway service of CODESYS and then try the search again. Sometimes it takes a while for CODESYS to forget the no longer active Raspberry Pis, if necessary restart the PC/Windows and then try the search again. Only one device should appear in the list.

3. If nothing else helps:

Then maybe a visit to the info page of the Raspberry Pi Foundation to determine the IP address of a Raspberry Pis might help.

You can reach the page via the following address

<https://www.raspberrypi.org/documentation/remote-access/ip-address.md>

The page lists various options, including programs for Windows or macOS, even smartphone apps are mentioned.

6.6. Raspbian Login Information (Login)



Our images have the default Raspbian login settings as defined by the Raspberry Pi Foundation.

The login data is:

- User: pi
- Password: raspberry



6.7. Turning off the Raspberry Pi

The Raspberry Pi always raises the question:
"Can I just unplug the power to turn off the Raspberry Pi?"

This question can be answered something like this:

The Raspberry Pi is a "full-featured" computer on a circuit board and like any other computer with an operating system, the sudden loss of power can lead to data loss.

Therefore, pulling the "plug" is possible at any time, but should be avoided to ensure the integrity of the file system and the data on the SD card. Similar to a bigger computer, it can happen that the Raspberry Pi does not start and the SD card has to be re-made in order to make it usable again.

To restart the Raspberry Pi, use the following command:

```
sudo reboot
```

If you want to turn it off, try this command:

```
sudo shutdown -h now
```

After a few seconds, the operating system has stopped all services and brought the operating system to a safe state. Now you can safely disconnect the power and all data on the SD card remains intact.

Remark:

Pulling the plug or interrupting the power supply to the Raspberry Pi does not necessarily lead to data loss or an unusable SD card. The information on this page is intended to be an indication that, as with any other computer, there is a possibility of failure, but no certainty that it will occur.

Many users of the Raspberry Pi have never experienced this problem at all, even though they always disconnect the power supply of the Raspberry Pi without shutting down the operating system.

If you have increased requirements, please contact us and we will help you to optimize your PiXtend application and achieve better data security.



6.8. Compatibility of the Software Components

There are many software components for the *PiXtend V2* which allow a wide usage of the platform. When using these components, the following must be observed.



The programs and software components listed in this document can not be used in parallel!

The "SPI bus" used for the communication with the microcontroller must only be used from one single program at a time. If the communication with microcontroller is done simultaneously from different programs, and also with different commands, this may lead to defective parts of the hardware, e.g. the relays and other problems.

A use of different programs "successively" is possible, if it is the "pxdev – Linux Tools & Library" or the "PiXtend Python Library V2 (PPLV2)". All programs which terminate automatically after each call and have to be manually restarted after each execution are suitable in this situation.

If the CODESYS Runtime extension has been installed on the SD card or you are using our CODESYS SD card image, no other program can be used to send the *PiXtend V2* additional commands or to get further information. The use is reserved exclusively for CODESYS, furthermore, CODESYS is automatically started every time the Raspberry Pi boots.

If you want to use the FHEM system and control your *PiXtend V2* this way, the same restrictions apply as for CODESYS, since FHEM also starts automatically after its installation, problems may occur otherwise. A parallel use of FHEM and CODESYS is not possible, since there is overlap in the usage of the SPI bus.

6.9. Enable Serial Interface

In order to use the serial interface of the Raspberry Pi computer for ones own purposes, it must be activated or configured first. In other words, if you are using the serial port already, the following commands will enable the serial interface and it will not send any data anymore by itself, this also disables access to the Linux console through the serial line.

The serial interface of the Raspberry Pi can be activated or changed with the program "raspi-config":

```
sudo raspi-config
```



PiXtend V2 Software Manual

In the program, go to:

5 Interfacing Options --> *P6 Serial* --> *<No>* --> *<Yes>* --> *<Ok>*

The following entry can now be found in the file `/boot/config.txt`:

```
enable_uart=1
```

Bluetooth is automatically deactivated and UART activated. The program "raspi-config" does all the work for us.

Then restart the Raspberry Pi and you can use the serial interface exclusively in your own applications.

You can reboot by using the following command:

```
sudo reboot
```

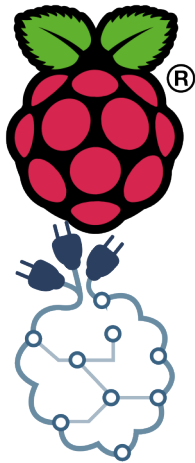


7. CODESYS

This chapter describes the installation of all software components required to program *PiXtend V2* (www.pixtend.de) with CODESYS (www.codesys.de).



The CODESYS development environment is designed for professional hardware-independent programming of control systems according to IEC 61131-3 and is developed by 3S-Smart Software Solutions.



In order to program a Raspberry Pi in CODESYS the Raspberry Pi Runtime extension for CODESYS, also developed by 3S, is required.

Special device drivers for *PiXtend V2* are required to get direct access to the I / O hardware and interfaces of the *PiXtend V2* board using CODESYS. The *PiXtend V2* drivers are provided free of charge by Qube Solutions GmbH.

This chapter is intended for all who want to create a project for *PiXtend V2* using CODESYS.

Notice:



The use of CODESYS is the same for PiXtend V2 -S and PiXtend V2 -L-, only for controlling the I / O hardware (also called I / O mapping), the appropriate SPI driver must be selected, since the distinction between both devices takes place here.



7.1. Information

7.1.1. Copyright of texts and images:

Texts and pictures, which are marked with the abbreviation (3S), are from the company 3S-Smart Systems GmbH in Kempten - www.codesys.com

Texts and images bearing the abbreviation (RPI) are from the Raspberry Pi Foundation - www.raspberrypi.org

Texts and pictures that are not marked or marked with the abbreviation (QS) are from Qube Solutions GmbH – www.pixtend.de.

7.1.2. Compatibility

CODESYS Projects and programs written for or on other CODESYS controllers can often be transferred to other CODESYS compatible controllers with little effort.

The CODESYS Runtime extension for the Raspberry Pi computer makes it a full-fledged CODESYS PLC, i.e. all compatible CODESYS programs can run on it.

Please pay attention to the hardware compatibility, it is necessary to check the extent to which there are matches and deviations and, if necessary, make adjustments to the hardware configuration in the program, so that the correct signals get to the correct variable in the CODESYS project.

If you are unsure whether your existing CODESYS PLC project works with *PiXtend V2*, please contact us. We will be happy to advise you of necessary changes and can help you to transfer the program.



7.2. Prerequisites

7.2.1. System Requirements for CODESYS

- Microsoft Windows 7 / 8 / 10 (32 / 64 Bit)
- suitable PC hardware for the corresponding Microsoft Windows platform

7.2.2. Required Hardware

- *PiXtend V2* Board (www.pixtend.com)
- *Raspberry Pi* Modell B+ / 2 B / 3 B / 3 B+
- Standard RJ-45 network cable
- For the first commissioning of a new Raspberry Pi:
 - SD card reader, internal or external
 - SD memory card, recommended min. 4GB (better 8GB)
 - microSD for Raspberry Pi model B+ / 2 B / 3 B / 3 B+
 - optional: HDMI-capable monitor
 - optional: USB keyboard
 - optional: USB mouse



7.3. Installation of the required Software Components

7.3.1. CODESYS Development System

7.3.1.1 Introduction of CODESYS

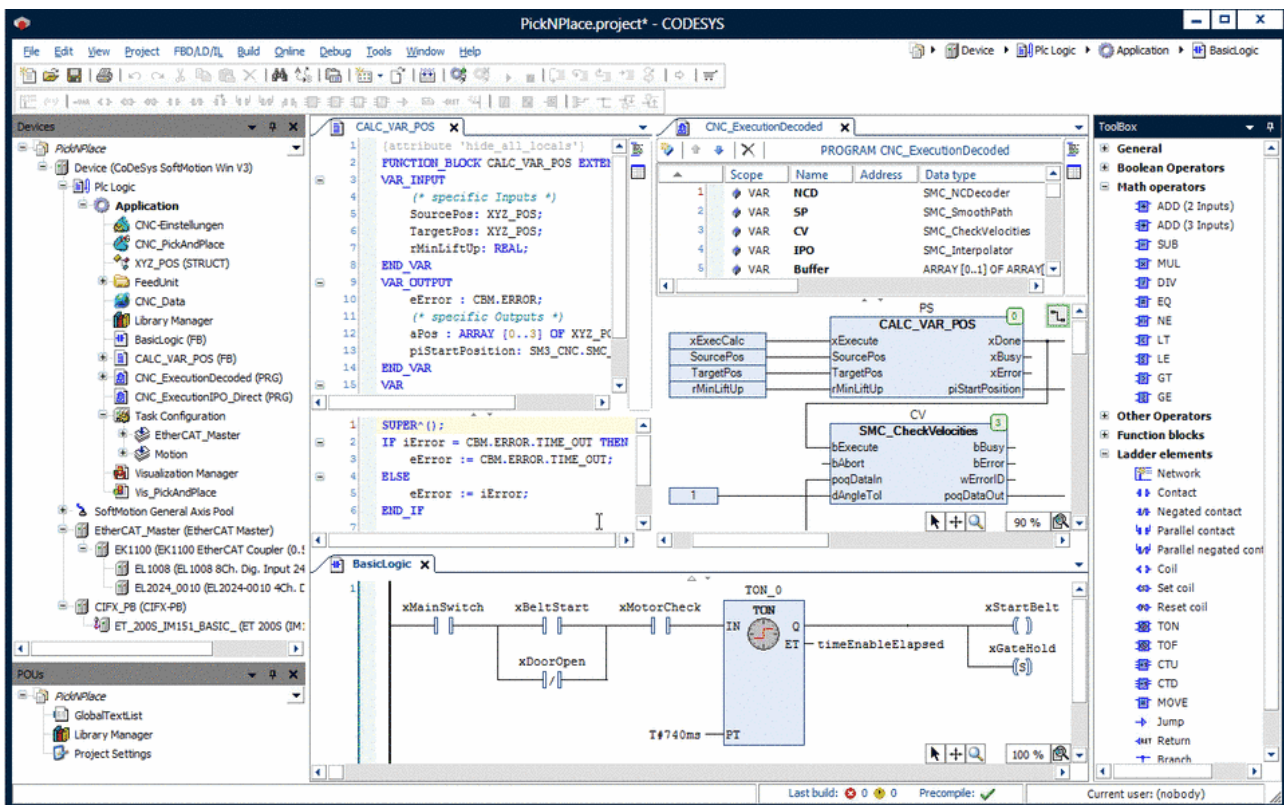


Figure 4: CODESYS Development System (Source: 3S)

CODESYS is a software platform for many tasks in the field of industrial automation technology. The basis is the IEC 61131-3 CODESYS Development System V3 programming tool. The tool provides the user with integrated solutions for his work - with the aim of providing him with practical support in the realization of his task. (Source: 3S)

The hardware-independent programming system CODESYS from the company 3S-Smart Software GmbH is ideally suited for use with Raspberry Pi and PiXtend V2. Apart from programming in all automation languages, it is also possible to create web visualizations for programmable logic controllers (PLC) according to the IEC 61131-3 standard. With the so-called "Webvisu", content and controls of your program can easily be brought to a website (CODESYS webserver runs on the Raspberry Pi). You do not need any knowledge of web programming (HTML, PHP, CGI, etc.) to create and work with Webvisu.



PiXtend V2 Software Manual

With the modern graphical interface of CODESYS, you are optimally equipped for the program creation for your control system and Webvisu.

You can access your Webvisu with any current smartphone, tablet or PC / MAC. All you need is a current web browser. We recommend the latest versions of browsers like Google Chrome, Microsoft Internet Explorer or Mozilla Firefox.

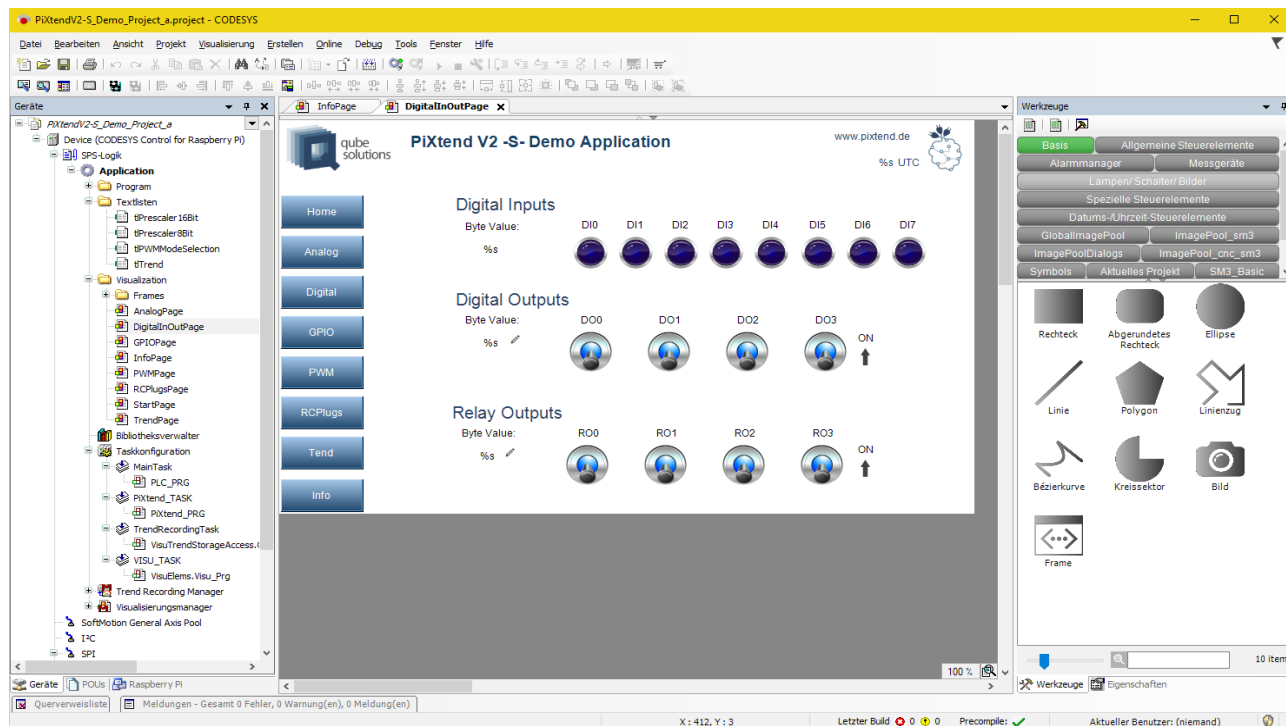


Figure 5: CODESYS - Visualization of the PiXtend V2 Demo Project

7.3.1.2 CODESYS Download

The CODESYS development environment can be downloaded free of charge from 3S. Only a free and uncomplicated registration is necessary.

- To do this, open the <https://store.codesys.com/> page in a browser of your choice.
- Click Login and register in the "CODESYS Store".
- Fill out the form completely and truthfully and note that a correct e-mail address is required.
- Click "Register Now" and follow the instructions on the store page or the instructions you receive by e-mail, you may need to confirm your e-mail address and wait until your account has been activated.
- With your self-assigned password and user name, you can login to the CODESYS



PiXtend V2 Software Manual

store and download the latest version of CODESYS (CODESYS V3.5 SP13, as of 31/07/2018).

7.3.1.3 CODESYS Installation

Install the CODESYS development system with the default settings, and preferably as an Administrator user, or perform the installation with Administrator rights. CODESYS itself can be started later without extended rights, a "normal" Windows user is sufficient.

You now have a fully-fledged CODESYS development system installed with which you can program the *PiXtend V2* PLC according to the industry standard IEC 61131-3.

7.3.2. CODESYS Control for Raspberry Pi installation

7.3.2.1 CODESYS Control for Raspberry Pi - Download

To use a Raspberry Pi with CODESYS, you need the free CODESYS Control Runtime extension for the Raspberry Pi from the CODESYS Store.

The CODESYS Store includes both free and paid products, which can be integrated into CODESYS as additional features.

To download files from the CODESYS Store, you must login to the CODESYS Store. You have already created the required account in a previous chapter.

- Immediately after the login, you can download all products labeled "Free", free of charge
- Click on the "CODESYS Control for Raspberry Pi SL" or follow the direct link:
<http://store.codesys.com/codesys-control-for-raspberry-pi-sl.html>
- Click the download button and wait for the download to complete
- A "First Steps" guide can be found in the CODESYS Online help at:
https://help.codesys.com/webapp/_rbp_f_help;product=CODESYS_Control_f_or_Raspberry_Pi_SL;version=3.5.13.0

Notes on the demo version:

- The free version is limited to 2 hours running time. After that, the runtime environment automatically ends until it is restarted. After each restart, the Raspberry Pi can be used again for full 2 hours without restrictions. This is usually sufficient for initial tests.
- For a time unlimited version, a license is available at the current price of 50 € plus VAT. You can then perform the licensing later on in the CODESYS Store.



PiXtend V2 Software Manual

7.3.2.2 CODESYS Control for Raspberry Pi – Installation

Double-clicking on the "CODESYS Control for Raspberry PI 3.5.X.X.package" file starts the CODESYS Package Manager, which automatically performs the installation of the Raspberry Pi runtime components for CODESYS.

Alternatively, you can start CODESYS manually and open the Package Manager via the Tools → Package Manager menu.

In the Package Manager, click "Install" and select the package you just downloaded.

Perform the installation using the default options.

Notice:



You must have Administrator rights on the computer for the installation, otherwise it may fail. In case of doubt, start CODESYS by right-clicking → "Run as Administrator" and start the "Package Manager" to install the Raspberry Pi Package.

The package includes:

- CODESYS Extension for the Raspberry Pi (drivers and libraries)
- Sample programs (default is the user's directory under "CODESYS Control for Raspberry PI")
- Debian Package codesyscontrol_arm_raspberry_V3.5.X.X.deb for the installation of the Runtime on the Linux system of the Raspberry Pi



PiXtend V2 Software Manual

Notice:



You can find a manual with more detailed technical information and installation instructions in the CODESYS Online help at:

https://help.codesys.com/webapp/_rbp_f_help;product=CODESYS_Control_for_Raspberry_Pi_SL;version=3.5.13.0

At top of the page you can choose your language if needed.

The document also explains the licensing options for the CODESYS Raspberry Pi Runtime extension.

7.3.3. Create a bootable SD-Card for the Raspberry Pi

Download the latest "*PiXtend V2 Image CODESYS*" SD card image from the download area at <https://www.pixtend.de/downloads/>.

The image is based on the Raspbian Linux operating system and is already pre-configured for use with CODESYS and *PiXtend V2*.

Alternatively, you can download and install the latest version of Raspbian as described at <https://www.raspberrypi.org/downloads/raspbian/>.

Then follow the installation instructions which you find in the CODESYS Online help:

- [CODESYS Online help for CODESYS Control for Raspberry Pi SL](#)

For beginners, we strongly recommend using our *PiXtend V2 Image "CODESYS Control"* to achieve a fast and easy start.

For more information on creating a SD card for the Raspberry Pi, see Chapter 6.4 Preparing a SD card.



PiXtend V2 Software Manual

7.3.4. PiXtend V2 CODESYS device driver installation

In order to access the *PiXtend* V2 hardware in CODESYS, you need the *PiXtend* V2 CODESYS device drivers.

You can download them either from the download area of our homepage (<https://www.pixtend.de/downloads/>) or in the CODESYS Store (always free of charge).



7.3.4.1 PiXtend V2 CODESYS Device driver - Download

The package "PiXtend V2 Professional for CODESYS" contains the CODESYS device drivers, sample programs and a detailed manual for the use of CODESYS with the *PiXtend* V2.

7.3.4.2 PiXtend V2 CODESYS Device driver – Installation

A double-click on the file "PiXtend_V2_Professional_for_CODESYS_V2_X_X.package" starts the CODESYS package manager, which automatically performs the installation of the *PiXtend* V2 device drivers, sample projects and *PiXtend* V2 documentation for CODESYS. By default, the contents is stored in the user's directory under "PiXtend V2 Professional for CODESYS".

Important:



You must have Administrator rights on the computer for installation, otherwise the installation may fail. In case of doubt, start CODESYS by right-clicking → "Run as Administrator" and start the "Package Manager" to install the PiXtend V2 Package via the Tools menu.

7.4. Further steps

Your CODESYS development system is now ready for use with *PiXtend* V2.

To create an empty CODESYS project with *PiXtend* V2 support, refer to chapter "7.5 CODESYS – Create a project".



7.5. CODESYS – Create a project

This chapter describes all the steps necessary to create a new *PiXtend V2* project (www.pixtend.de) in CODESYS (www.codesys.com). You will learn how to use *PiXtend V2* as a CODESYS device and how to create a simple visualization for your project.

7.5.1. Step by Step to the first PiXtend V2 CODESYS program

7.5.1.1 Create a CODESYS Standard Project for *PiXtend V2*

Start CODESYS.

Create a new project by clicking File → New Project in the main menu (Shortcut Ctrl-N)

Select "Standard Project" from the "Projects" category and give the project a name (here "PiXtendTest") and confirm with "OK"

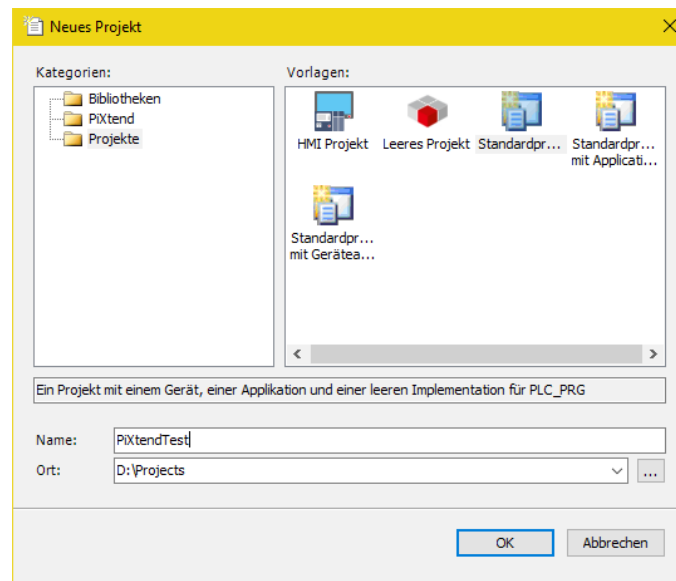


Figure 6: CODESYS - New Project



PiXtend V2 Software Manual

Select "CODESYS Control for Raspberry Pi" as the device and select "Continuous Function Chart (CFC)" as the programming language for the main PLC_PRG program.

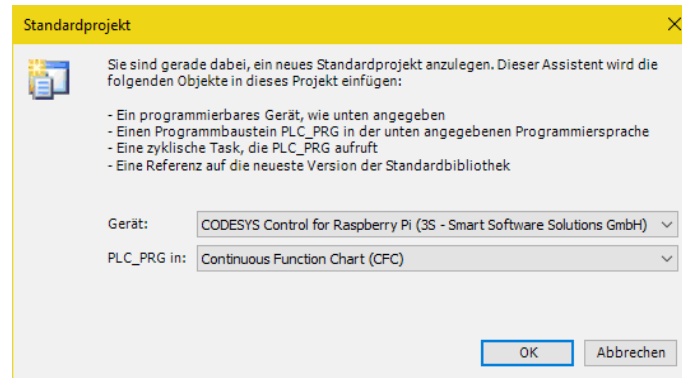


Figure 7: CODESYS - New Project - Device selection

After CODESYS has created the standard project, you will have a project with the following structure:

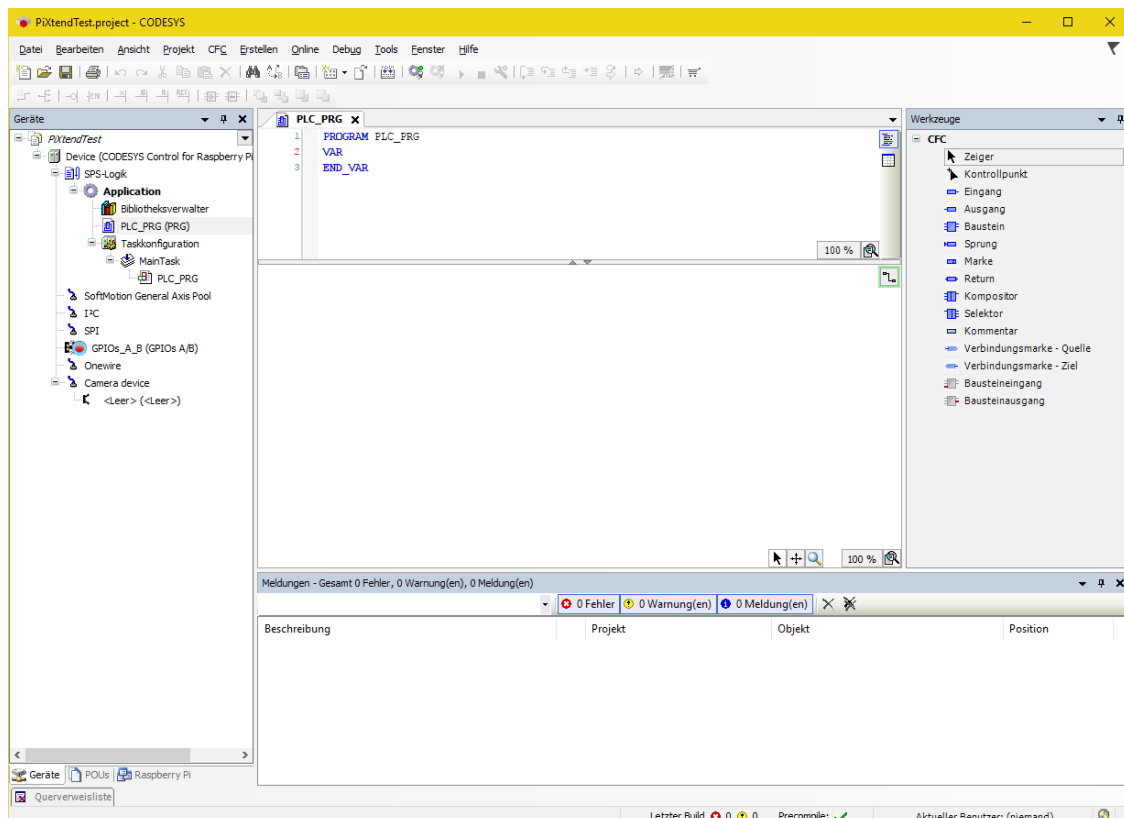


Figure 8: CODESYS - PiXtendTest Project



PiXtend V2 Software Manual

7.5.1.2 Attach SPI device

In the project tree, right-click on the entry "SPI" and select "Add device"

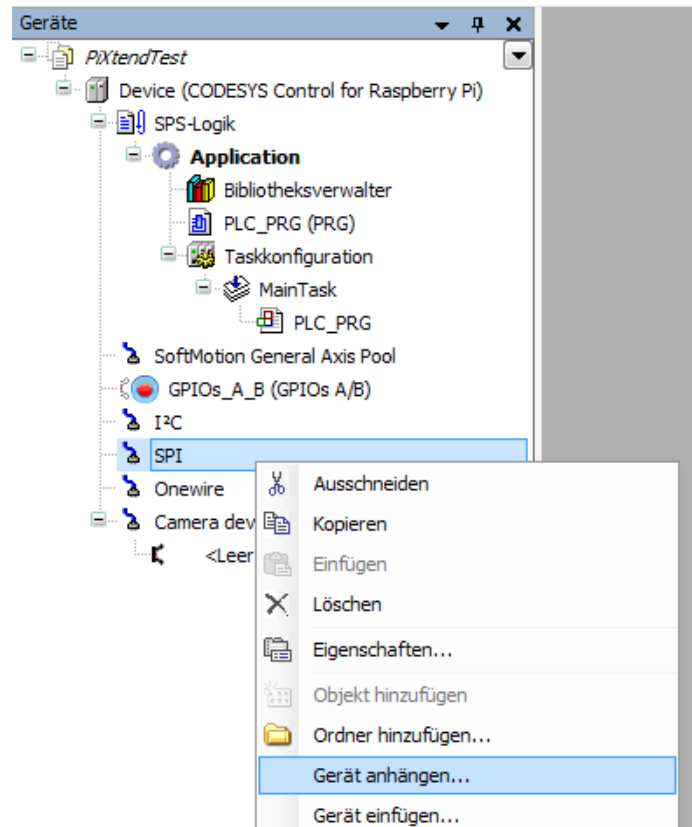


Figure 9: CODESYS - Add Device



PiXtend V2 Software Manual

Select "SPI master" as the device and click the "Add Device" button. Leave the window open.

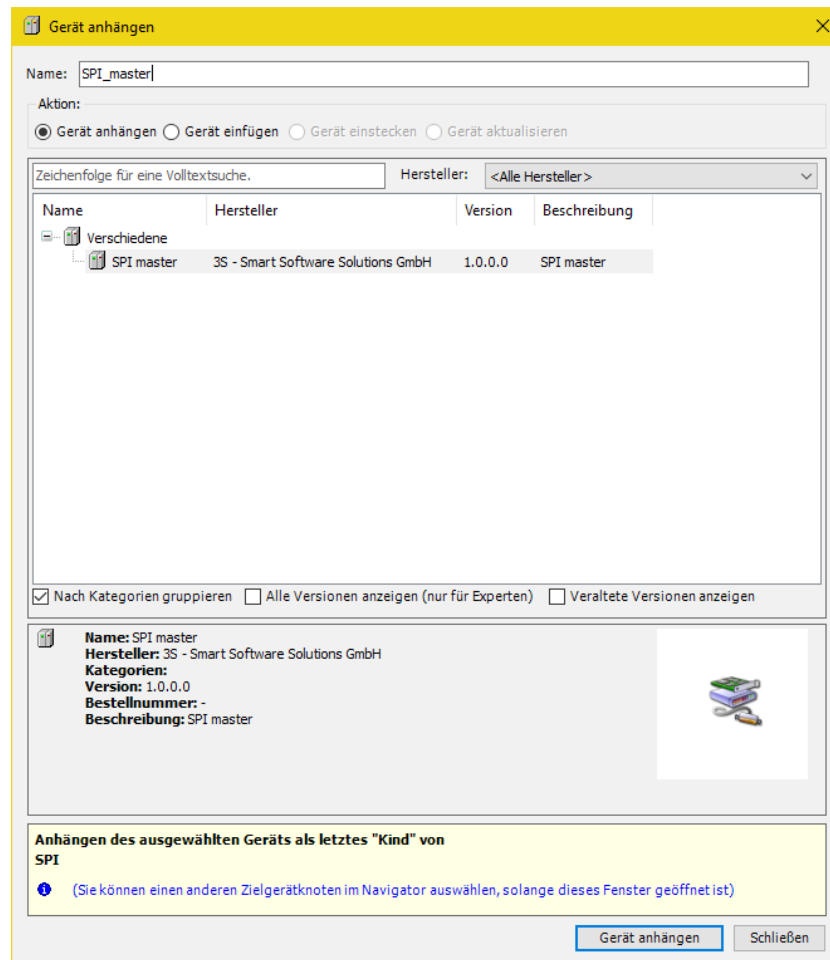


Figure 10: CODESYS - Add Device - SPI master

A new entry "SPI_master (SPI Master)" now exists in the project tree under SPI.



PiXtend V2 Software Manual

7.5.1.3 Add the PiXtend V2 -S- device

While the window is open, click the SPI Master you just added with the left mouse button. The contents of the "Add Device" window is updated.

Select "Qube Solutions GmbH" from the device manufacturer's drop-down menu, select the device "*PiXtend V2 -S-**" (not *PiXtend V2 -S- DAC*) and click on the "Add Device" button in the lower right corner and close the window.

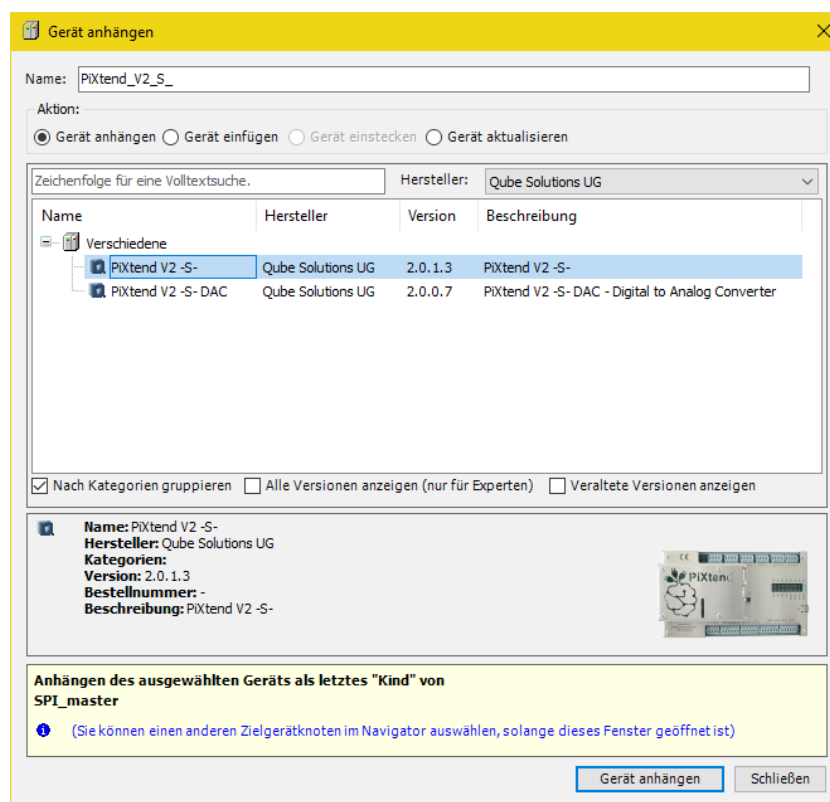


Figure 11: CODESYS - Add Device - PiXtend V2 -S-

Now PiXtend_V2_S_ appears as the device under "SPI_master (SPI Master)".

*Note:

As an alternative to the *PiXtend V2 -S-* you can also use a *PiXtend V2 -L-* device at this point. The example described here applies equally to both *PiXtend V2* devices, but note that the device names may be different.



PiXtend V2 Software Manual

Double-clicking on PiXtend_V2_S_ in the project tree opens the configuration page for the device. The "SPI devices I/O Mapping" tab shows which inputs and outputs of *PiXtend V2 -S-* are available for use in CODESYS.

Later, during operation ("Online"), you can observe the entire process image of *PiXtend V2 -S-* in this window. Input values can be monitored and output values can be set directly. However, since we want to use the values in our main program as well as in the visualization, we create a "Global Variable List" to assign variables to the inputs and outputs.

7.5.1.4 Create Global Variable List

On the right, click the "Application" entry in the project tree and add a new Global Variable List with the name "GVL" using "Add Object" -> "Global Variable List"

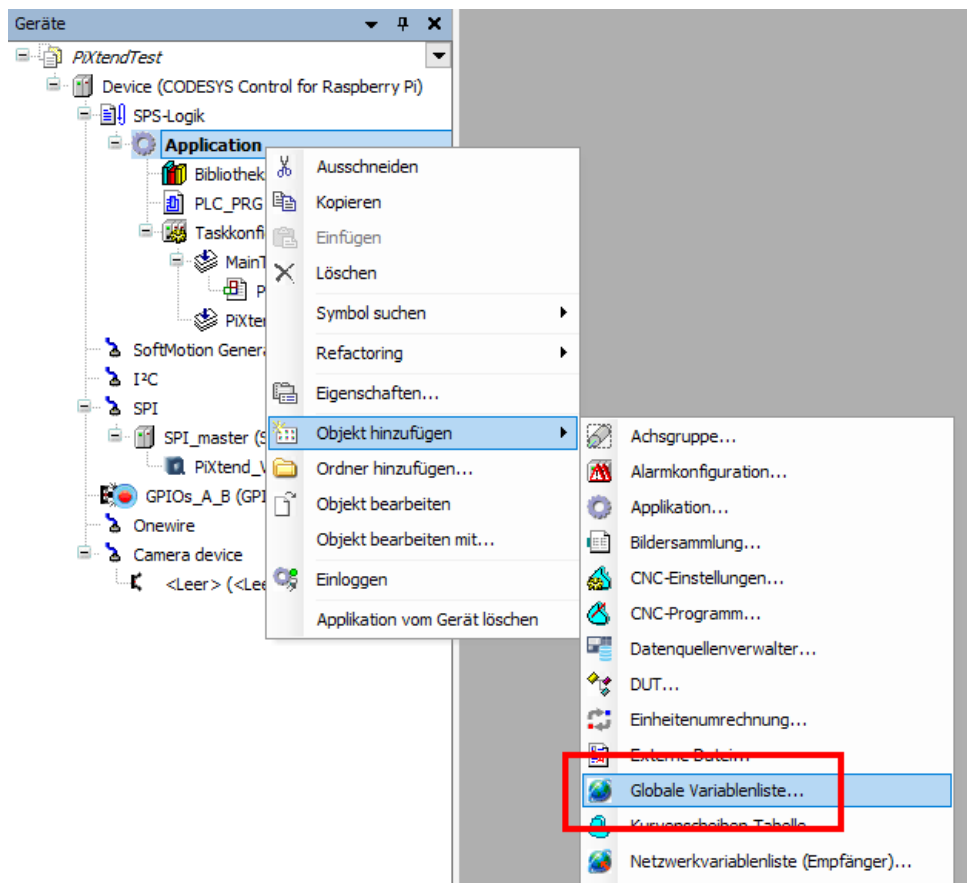


Figure 12: CODESYS - Add Global Variable List



PiXtend V2 Software Manual

Open the GVL and add the following entries:

```
//Digital Inputs
xDI0: BOOL;
xDI1: BOOL;
//Digital Outputs
xDO0: BOOL;
xDO1: BOOL;
//Status
xRun: BOOL;
byHardware: BYTE;
byFirmware: BYTE;
```

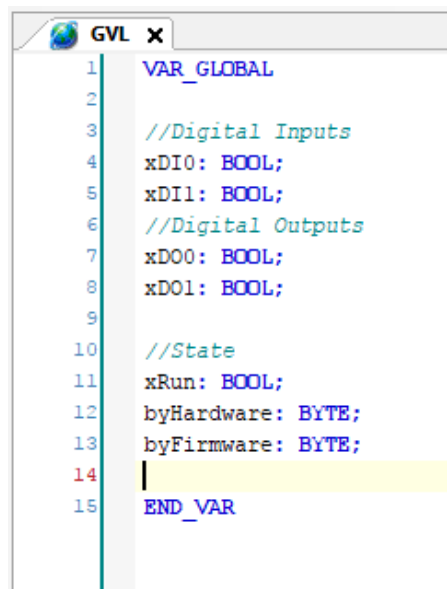


Figure 13: CODESYS - GVL - Declarations

In order to keep this chapter simple we restrict ourselves to the use of only 2 digital inputs and 2 digital outputs.

xDI0 stands for the first digital input, i.e. digital input 0, and xDO0 for digital output 0. Notice the difference between O (letter) and 0 (number).

Prefixes for variable names have proven particularly useful in larger projects.

We recommend using the prefix x for BOOL variables (true or false) to prevent confusion with Byte (prefix by) variables.



PiXtend V2 Software Manual

Other prefixes:

x → BOOL
by → BYTE
w → WORD
dw → DWORD
r → REAL
s → STRING

The status Bit xRun provides information on the status of the microcontroller.

The bytes byHardware and byFirmware provide information on the hardware revision of *PiXtend V2 -S-* and the firmware version of the microcontroller.

The naming of the variables in the GVL is left to you, but we use the same convention as the *PiXtend V2 -S-* demo project to make your start easier.

7.5.1.5 Mapping of Variables

After you have created the required variables in the GVL, they must be assigned – "mapped" - to, i.e. the respective hardware inputs and outputs of *PiXtend V2 -S-*.

To do so, open the already known "SPI devices I/O Mapping" tab by double-clicking on the *PiXtend V2 -S-* device. Open the folder "Digital Inputs" and also the channel "DigitalInputs" (in the picture of type Byte with address %IB22):

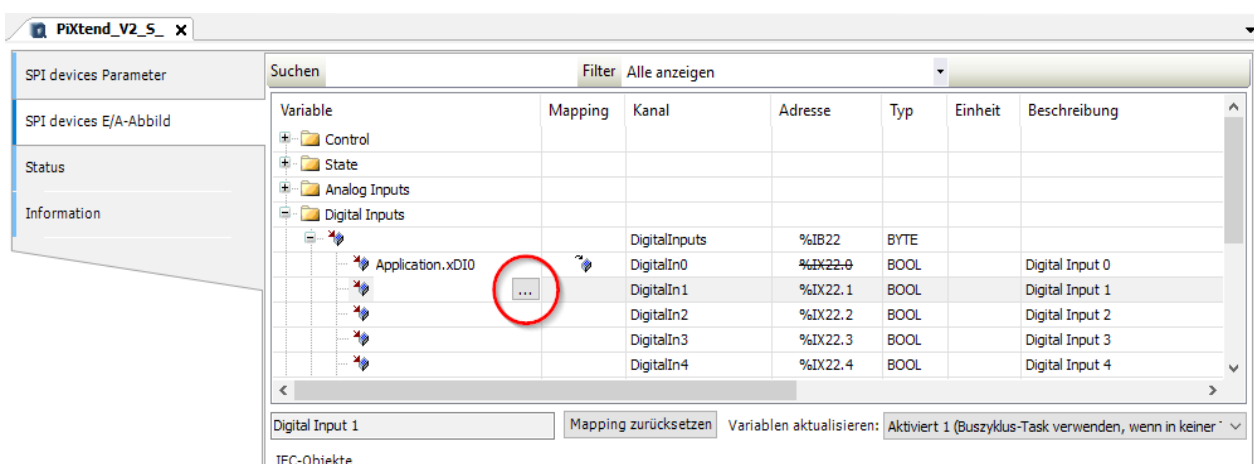


Figure 14: CODESYS - PiXtend V2 -S- I/O-Mapping

Now the existing input variables can be mapped bit by bit, after double clicking on the empty left column, a button with three points (...) appears. A click on it opens the input help with which the required variable is selected.



PiXtend V2 Software Manual

Select the variable Application → GVL → xDI0 for DigitalIn Bit 0

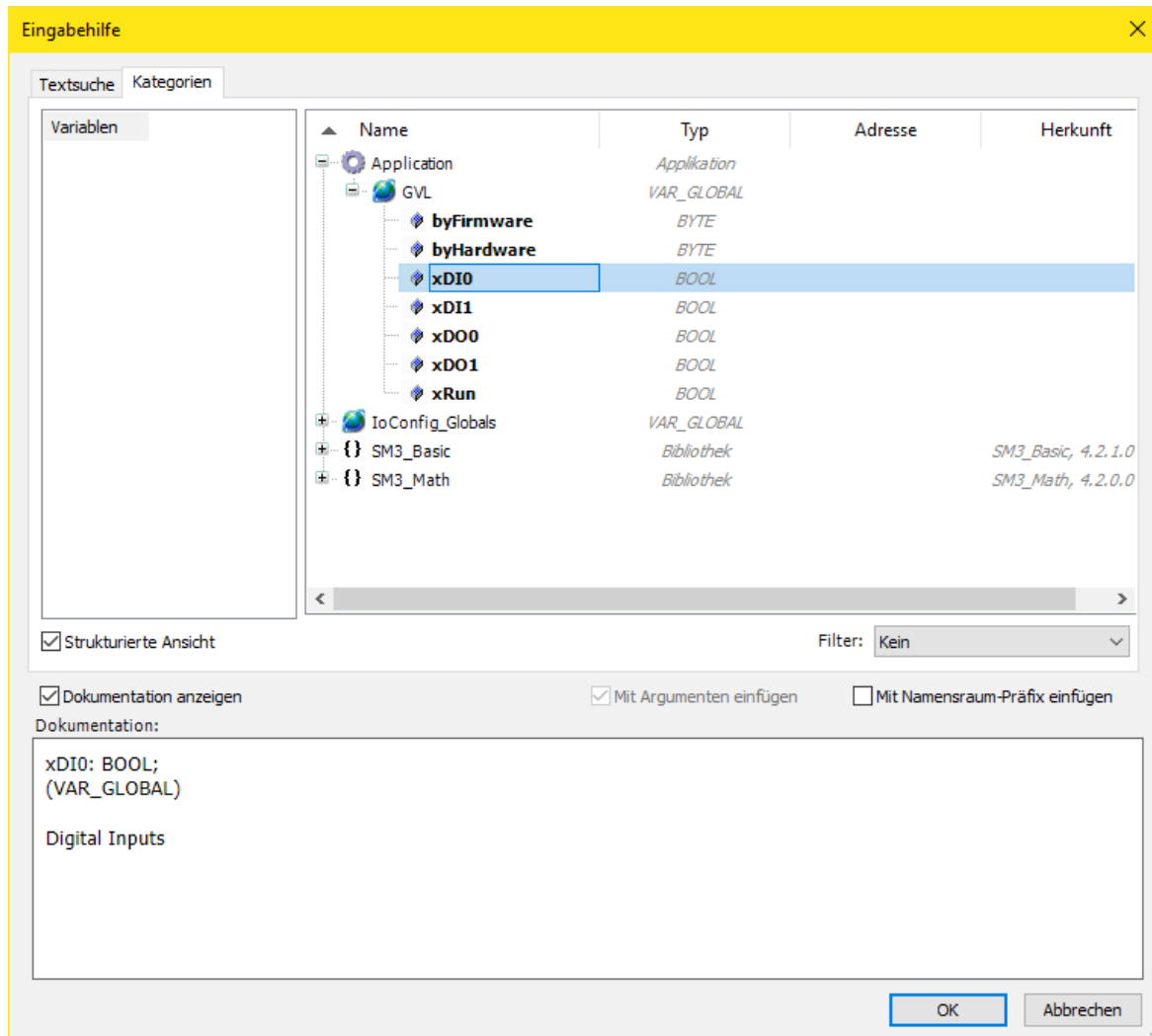


Figure 15: CODESYS - Input Assistance



PiXtend V2 Software Manual

Repeat the process for the remaining variables:

Application → GVL → xDI0 for Digital Input → DigitalIn → Bit 0

Application → GVL → xDI1 for Digital Input → DigitalIn → Bit 1

Application → GVL → xDO0 for Digital Out → DigitalOut → Bit 0

Application → GVL → xDO1 for Digital Out → DigitalOut → Bit 1

Bytes and bits (BOOLs) can be mapped according to the same principle:

State					
Application.byFirmware		Firmware	%IB0	BYTE	
Application.byHardware		Hardware	%IB1	BYTE	
		ModelIn	%IB2	BYTE	
		RetainCRCErr	%IX3.0	BIT	
		RetainVoltageError	%IX3.1	BIT	
		Error0	%IX3.2	BIT	
		Error1	%IX3.3	BIT	
		Error2	%IX3.4	BIT	
		Error3	%IX3.5	BIT	
Application.xRun		Run	%IX3.6	BIT	

Figure 16: CODESYS - PiXtend V2 -S- I/O-Mapping - State

Application → GVL → xRun → Run

Application → GVL → byFirmware → Firmware

Application → GVL → byHardware → Hardware

Finally, the GPIO bit 24 of the Raspberry Pi must be configured as an output. To do this, open the Raspberry Pi GPIO configuration by double-clicking on "GPIOs_A_B" in the project tree. Select "Output" for GPIO24:

GPIOs_A_B X						
GPIOs Parameter	Parameter	Typ	Wert	Standar...	Einheit	Beschreibung
GPIOs E/A-Abbild	GPIO4	Enumeration of BYTE	not used	not used		configuration of GPIO4
Status	GPIO17	Enumeration of BYTE	not used	not used		configuration of GPIO17
Information	GPIO18	Enumeration of BYTE	not used	not used		configuration of GPIO18
	GPIO22	Enumeration of BYTE	not used	not used		configuration of GPIO22
	GPIO23	Enumeration of BYTE	not used	not used		configuration of GPIO23
	GPIO24	Enumeration of BYTE	Output	not used		configuration of GPIO24
	GPIO25	Enumeration of BYTE	not used	not used		configuration of GPIO25
	GPIO27	Enumeration of BYTE	not used	not used		configuration of GPIO27
	GPIO28	Enumeration of BYTE	not used	not used		configuration of GPIO28

Figure 17: CODESYS - GPIO Parameter - GPIO24



PiXtend V2 Software Manual

Then change to the "GPIO I/O Mapping" tab and click on "Outputs". As a variable for bit 24, enter "rpi_gpio24".

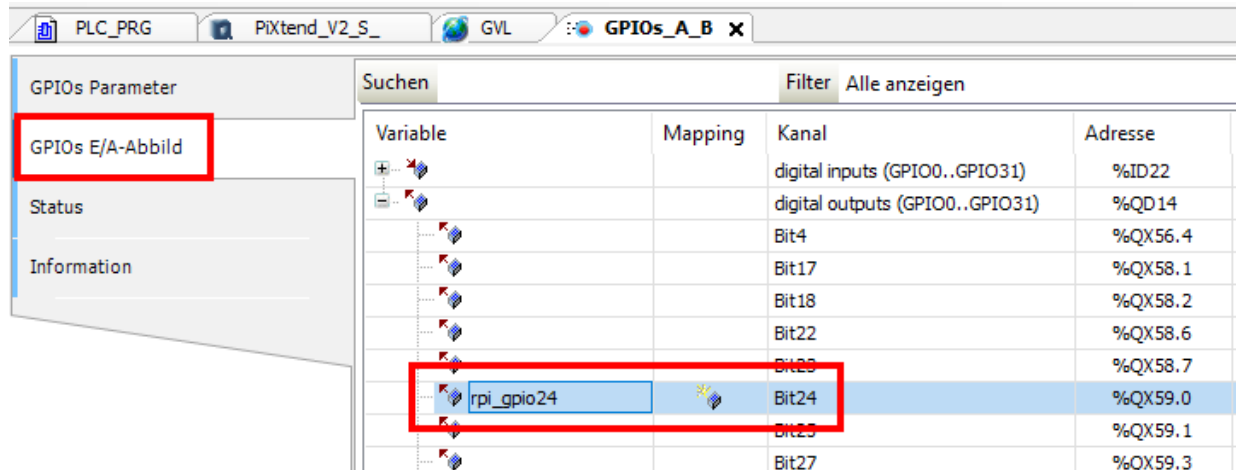


Figure 18: CODESYS - GPIO I/O-Mapping - rpi_gpio24 variable

7.5.1.6 Creating the Main Program

Now it is time to create a program which monitors the two inputs DI0 and DI1 and sets the two outputs DO0 and DO1 according to our programs logic.

In contrast to sequential programs (eg. in C or Python), which are usually executed only once, from top to bottom, it is necessary for programmable logic controllers (PLC) that certain program parts are cyclical in nature, ie can be called at fixed intervals to check whether any input variables have changed in the meantime and whether the outputs must be set or reset according to program logic.

A so-called task is used to determine the order in which a program is called and how often.

The typical sequence within a PLC cycle is:

- Retrieve and store all hardware input values
- Program logic to calculate new output values
- Assign all hardware output values



PiXtend V2 Software Manual

Click Task Configuration → MainTask to set the cycle time to t # 100ms. This cycle time is independent of the hardware used, since it has its own task. The automatically created PiXtend task has a cycle time of 30 ms and should not be changed. For more information, see chapter 6.2 SPI Communication, Data Transmission and Cycle Time.

The notation t # symbolizes that the value is a time specification and is usually used when programming controllers according to IEC 61131-3.

Priorität (0..31):
1

Typ	Intervall (z.B. t#200ms)
Zyklisch	t#100ms

Watchdog

☐ Aktivieren

Zeit (z.B. t#200ms):

Empfindlichkeit: 1

+ Aufruf hinzufügen ✕ Aufruf entfernen ✎ Aufruf ändern ⬆ Nach oben ⬇ Nach unten ➡

POU	Kommentar
PLC_PRG	

Figure 19: CODESYS - Task configuration - MainTask

The settings made are as follows: In the "MainTask" task, the PLC_PRG program is called every 100 ms, which then runs sequentially (from top to bottom). This is repeated every 100 ms.



PiXtend V2 Software Manual

In our example, the main program is named "PLC_PRG" and is created in the "Continuous Function Chart" programming language, since we have chosen this setting for input when creating the CODESYS project.

Complex PLC programs are usually distributed over several POU (Program Organization Units) in order to improve the maintainability and clarity of the code.

For the time being, we restrict ourselves to a program unit, namely PLC_PRG, which has already been generated automatically by CODESYS when the standard project was created.

Double-click PLC_PRG to open the editor and begin entering the program code.

CFC is graphically programmed. First drag and drop three "input" blocks from the toolbox (right) and place them under each other in the work area:

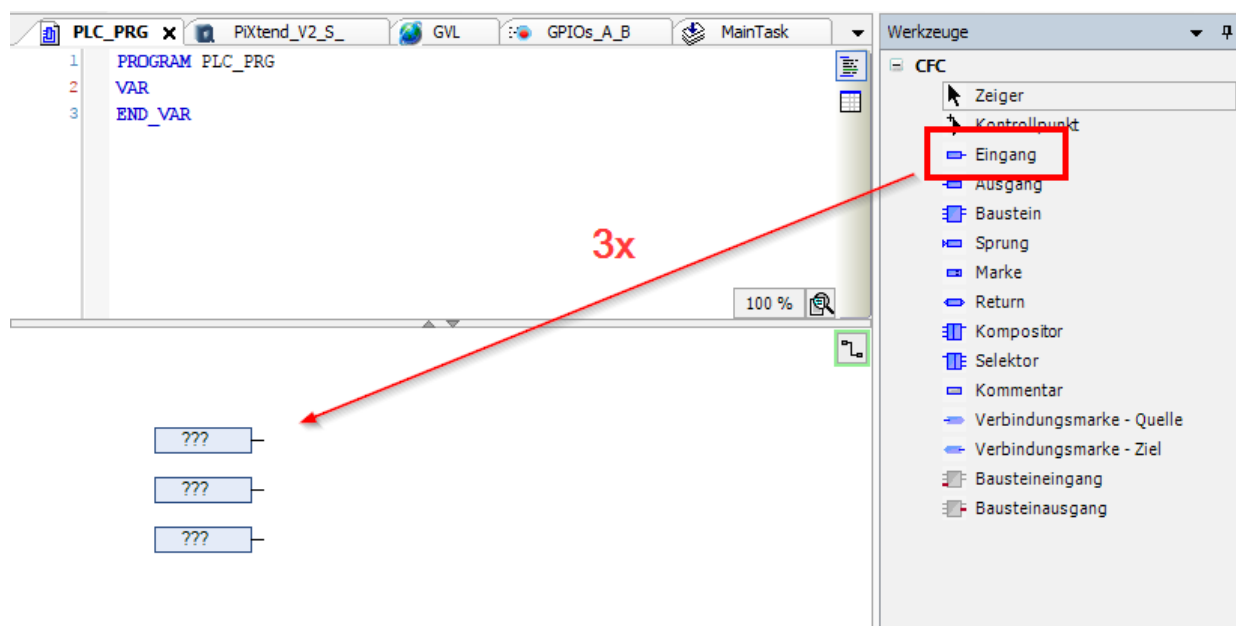


Figure 20: CODESYS - PLC_PRG (CFC) - Input



PiXtend V2 Software Manual

Repeat this with three "output" blocks and drag and drop a line from an input with to an output block.

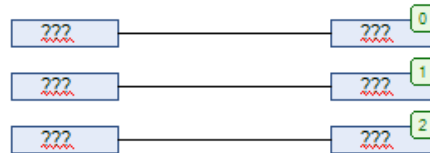


Figure 21: CODESYS - CFC - 3 Inputs connected to 3 outputs

Click in the middle of a block to assign variables or constants as shown in figure 22. A click on the three points ... opens the already known input help for variables. You can either type in the variable name and use the auto complete feature of CODESYS or select it from the list.

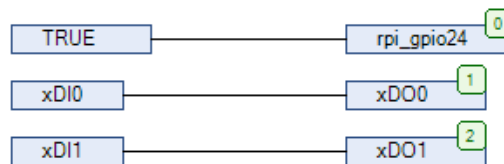


Figure 22: CODESYS - CFC - Parameterized

You have just created your first program. Here is a brief explanation:

- The Raspberry Pi GPIO bit 24 is set to TRUE to enable the SPI communication with the *PiXtend V2 -S-* microcontroller
- Input xDI0 is written to output xDO0
- Input xDI1 is written to output xDO1

On the menu bar, click Create → Compile to compile the program.



PiXtend V2 Software Manual

7.5.1.7 Connection to PiXtend V2 -S- and Program Download

If the program is compiled without errors, double-click "Device (CODESYS Control for Raspberry Pi)" in the project tree and then click the "Scan Network" button in the "Communication Settings" tab.

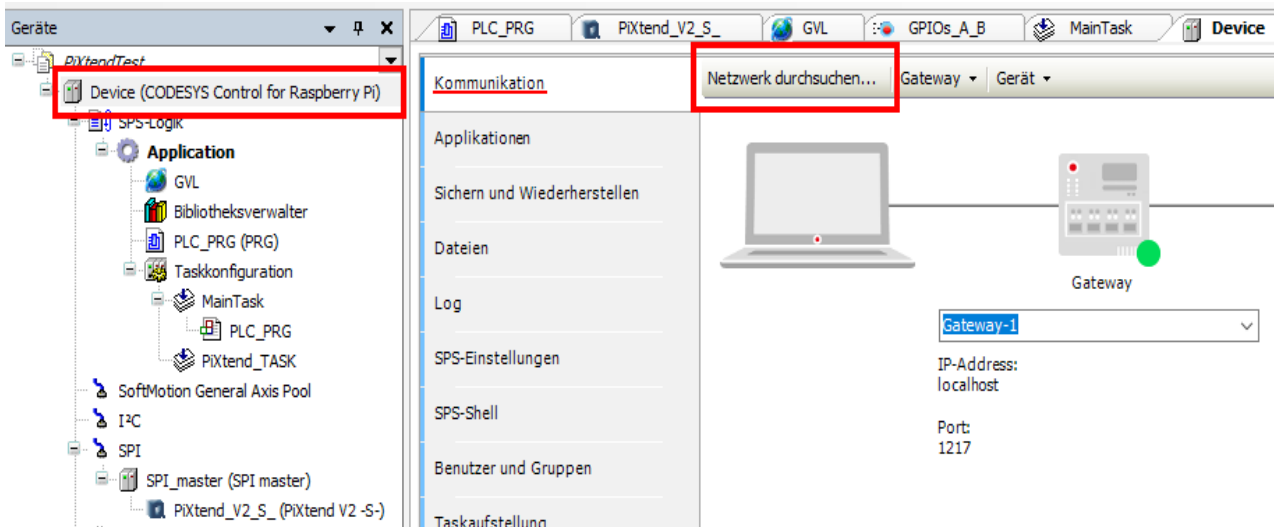


Figure 23: CODESYS - Communication - Scan Network...

CODESYS is now searching in your local network for a Raspberry Pi on which the CODESYS Runtime extension is running.

If no device is found, please check the following:

- Is the correct SD card with the image for the CODESYS runtime extension inserted? You can find a ready-made image in our download area (<https://www.pixtend.de/downloads/>). To create your own CODESYS image for the Raspberry Pi, please refer to the PDF guide which you can find in the CODESYS Store on the download page of the CODESYS Control for Raspberry Pi SL.
- Is the Raspberry Pi switched on and does it have a valid IP that you can ping from your PC? (via ping command from the Windows command line)
- The IP of your Raspberry Pi can be found with the command "ifconfig" in the Raspberry Pi command line. If you do not know the IP, you will need a screen and keyboard to look directly at the Raspberry Pi or use the search feature from within CODESYS to look for the IP of the Raspberry Pi.
- If you see a valid IP but the ping is not successful, check the network connection to your Raspberry Pi
- If the Raspberry Pi has been switched on for more than 2 hours, the CODESYS



PiXtend V2 Software Manual

Runtime extension will stop automatically and the Raspberry Pi must be restarted. You can do this for example, conveniently using the Linux console:

```
sudo shutdown -r now
```

If the Raspberry Pi has been found and selected, click on Online → Login on the main menu and download the program to the Raspberry Pi.

As soon as you are online, you can switch to "Run Mode" with the F5 key and see the live values in the *PiXtend V2 -S-* I/O Mapping tab. Click *PiXtend V2 -S-* in the device tree and select the "SPI devices I/O Mapping" tab.

Under State → "Firmware" and "Hardware" you will see the firmware version of the microcontroller and the revision of the hardware:

Variable	Mapping	Kanal	Adresse	Typ	Aktueller Wert	Vorher
Application.byFirmware	Firmware	Firmware	%IB0	BYTE	4	
Application.byHardware	Hardware	Hardware	%IB1	BYTE	21	
ModelIn			%IB2	BYTE	0	
RetainCRCErr			%IB3.0	BIT	FALSE	
RetainVoltageError			%IB3.1	BIT	FALSE	
Error0			%IB3.2	BIT	FALSE	
Error1			%IB3.3	BIT	FALSE	
Error2			%IB3.4	BIT	FALSE	
Error3			%IB3.5	BIT	FALSE	
Run	Run	Run	%IB3.6	BIT	TRUE	
BusCycleError			%IB3.7	BIT	FALSE	
CRCHdrInError			%IB4.0	BIT	FALSE	
CRCDatInError			%IB4.1	BIT	FALSE	
ModelInError			%IB4.2	BIT	FALSE	

Figure 24: CODESYS - PiXtend V2 -S- I/O-Mapping online

This is a *PiXtend V2* board revision 1 (hardware = 21) and the microcontroller firmware version 4. If you have a revised *PiXtend V2*, for example after an improvement or extension, the versions can differ.

The "Run" channel should be "True" in normal operation. This means that the microcontroller on *PiXtend V2* is in "Run Mode".



PiXtend V2 Software Manual

7.5.1.8 Further steps

Test the functionality of your program by applying a HIGH level (24 V) to the digital input 0 and 1 (for example by means of a button, switch, or a wire bridge). Please also pay attention to the notes on commissioning and the technical data sheet in the hardware manual.

If everything works, you can modify your PLC_PRG program to your liking. Try for example. following:

Add a module from the toolbar and assign it the type "AND" by clicking on the three question marks (?) and typing AND.

Connect the two inputs of the AND block with xDI0 and xDI1. The output of the AND block is therefore only TRUE if both inputs are simultaneously TRUE.

To make things even more interesting, add a "Timer On" module "TON" and give it an instance name (here "timer_delay"). The timer expects a time (here t # 2s - 2 seconds) and an input signal.

Connect the IN signal to the output of the AND block. The output Q is assigned to the digital output xDO1.

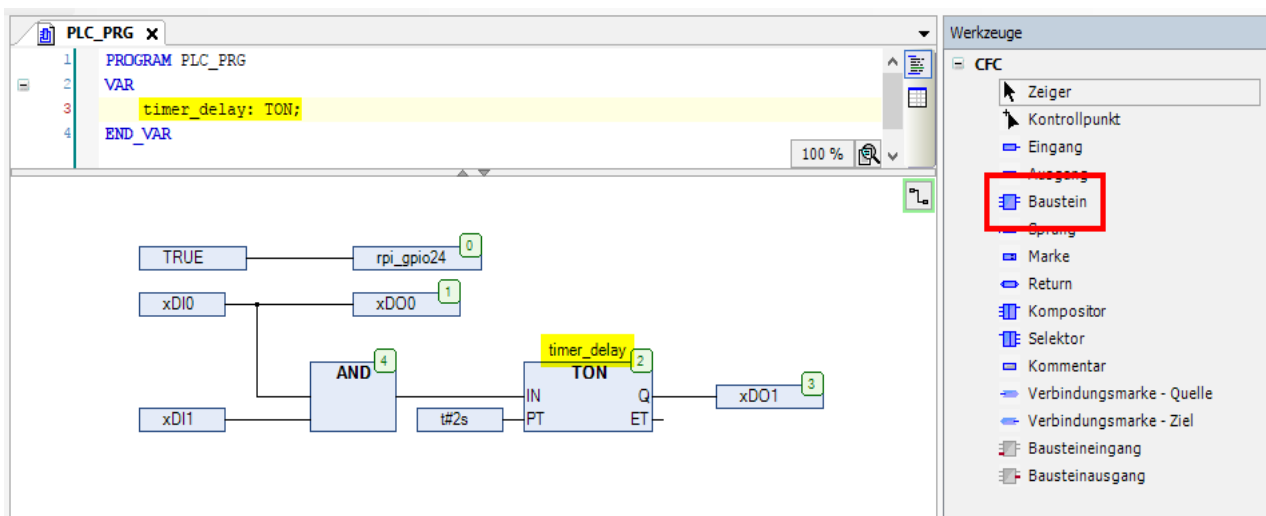


Figure 25: CODESYS - PLC_PRG (CFC) - Demo-Program

A high level on both inputs xDI0 and xDI1 for at least 2 seconds thus results in the output xDO1 being set. If only one input is HIGH, the output remains LOW. Similarly, the output remains LOW as long as both inputs are HIGH and the 2 seconds have not elapsed.



PiXtend V2 Software Manual

7.5.2. Step by Step to your first CODESYS Webvisu

We now want to add a visualization to the project so that you can monitor your control from any PC (also smartphone / tablet).

To do this, right-click on Application → Add Object → Visualization to the project tree.

CODESYS now automatically creates the "Visualization Manager" and a blank visualization called "Visualization".

In the manager, parameters for the Webvisu, e.g. the name of the visualization and preferred resolution. We leave it at the default settings.

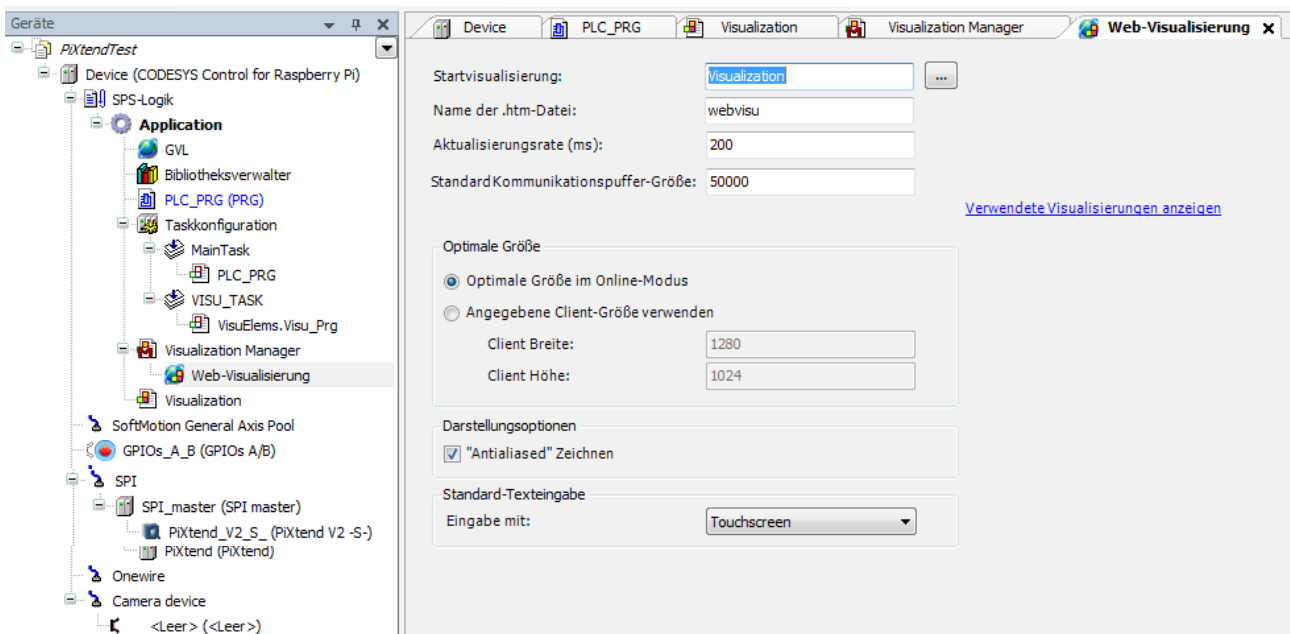


Figure 26: CODESYS - Web-Visualization - Configuration



PiXtend V2 Software Manual

A double click on "Visualization" opens the editor for the visualization. CODESYS already has a number of pre-made controls. Open the group Switches/Lamps/Images in the toolbar and place four lamps in the workspace.

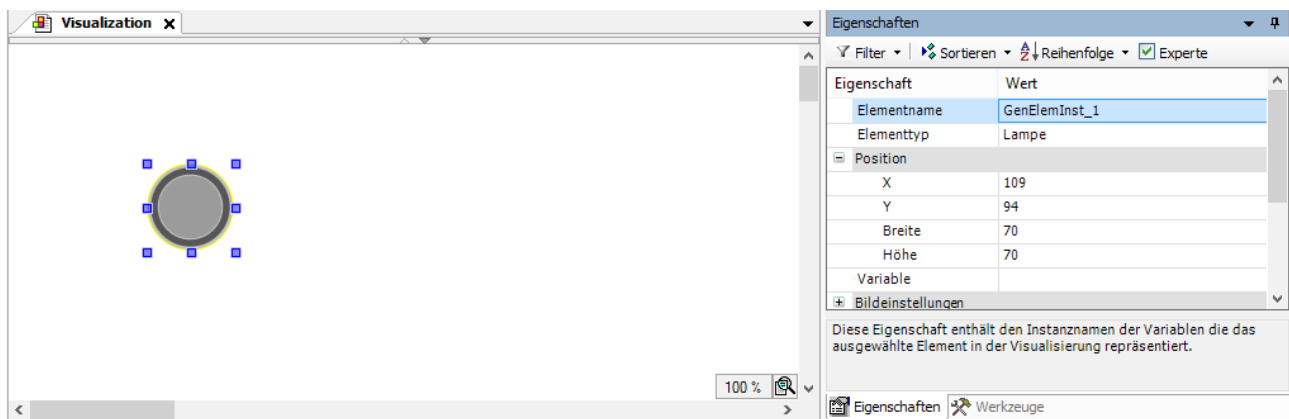


Figure 27: CODESYS - Visualization

Add 4 labels and assign the variables to the lamps under the "Variable" property (xDI0, xDI1, xDO0, xDO1).

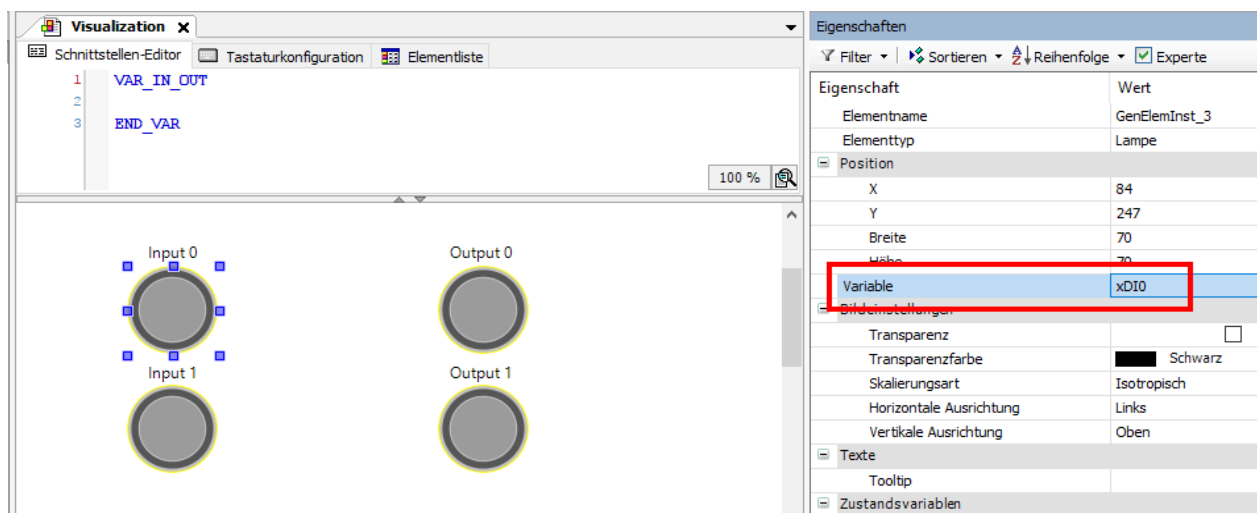


Figure 28: CODESYS - Visualization - Element property

Labels are always static. To display contents of variables, CODESYS uses text fields (rectangles).



PiXtend V2 Software Manual

Drag two new rectangles into the work area and enter into the "Text" property for the first rectangle "Firmware:%s" and into the second "Hardware:%s". This creates placeholders for two byte variables.

Set the checkbox "Expert" and insert the two variables for firmware version and hardware revision, byFirmware and byHardware, under "Text variables" for the respective rectangle.

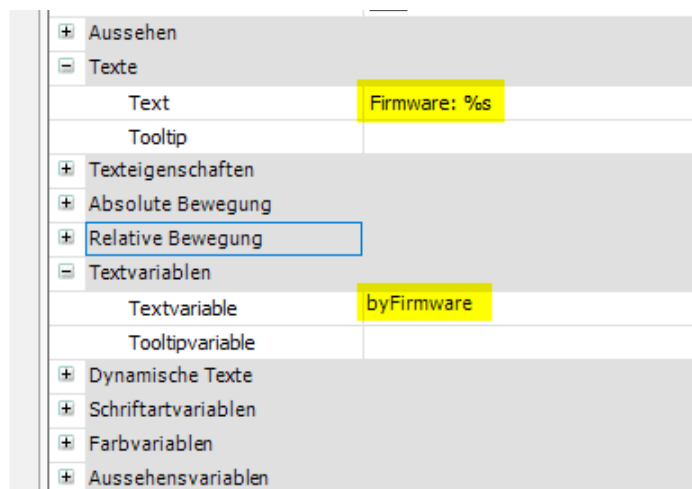
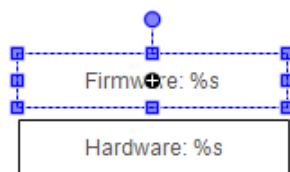


Figure 29: CODESYS - Visualization - Configuration of a rectangle

In live mode, the two placeholders %s are filled with the values from the variables. Further placeholders are %d for numbers, %f for floating point numbers (REAL, LREAL). Formatting options as in the C language are possible.



PiXtend V2 Software Manual

Finally, we add a heading and an animated CODESYS logo (special controls - wait symbol cube) to make the whole visu look nicer:

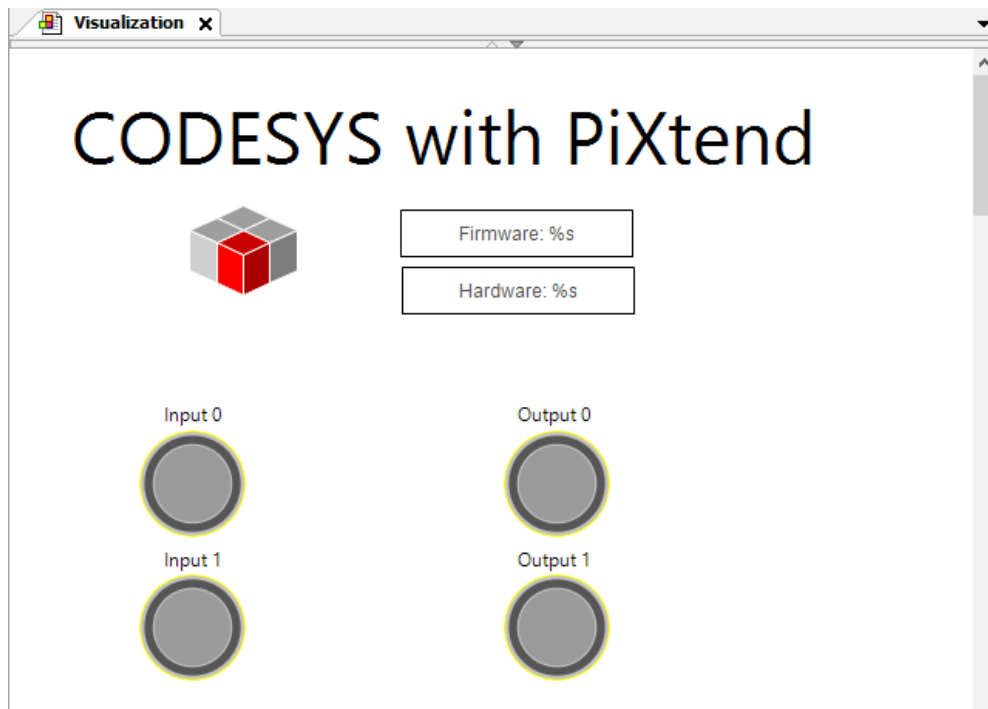


Figure 30: CODESYS - Visualization - Finished

Compile the project again using Build → Compile and perform a complete download. Run the program.

Open a browser of your choice on your PC or Smartphone / Tablet and enter the IP of your Raspberry Pi followed by ":8080/webvisu.htm", eg:

<http://192.168.178.99:8080/webvisu.htm>

We hope you enjoy using *PiXtend V2* with CODESYS and good luck with your projects!



7.6. Step by step to your first DAC program

7.6.1. Create CODESYS Standard Project for PiXtend

→ Start CODESYS.

Create a new project by clicking File → New Project in the main menu (Shortcut Ctrl + N)

Select "Standard Project" from the category "Projects" and give the project a name (here "PiXtendDACTest") and confirm with "OK".

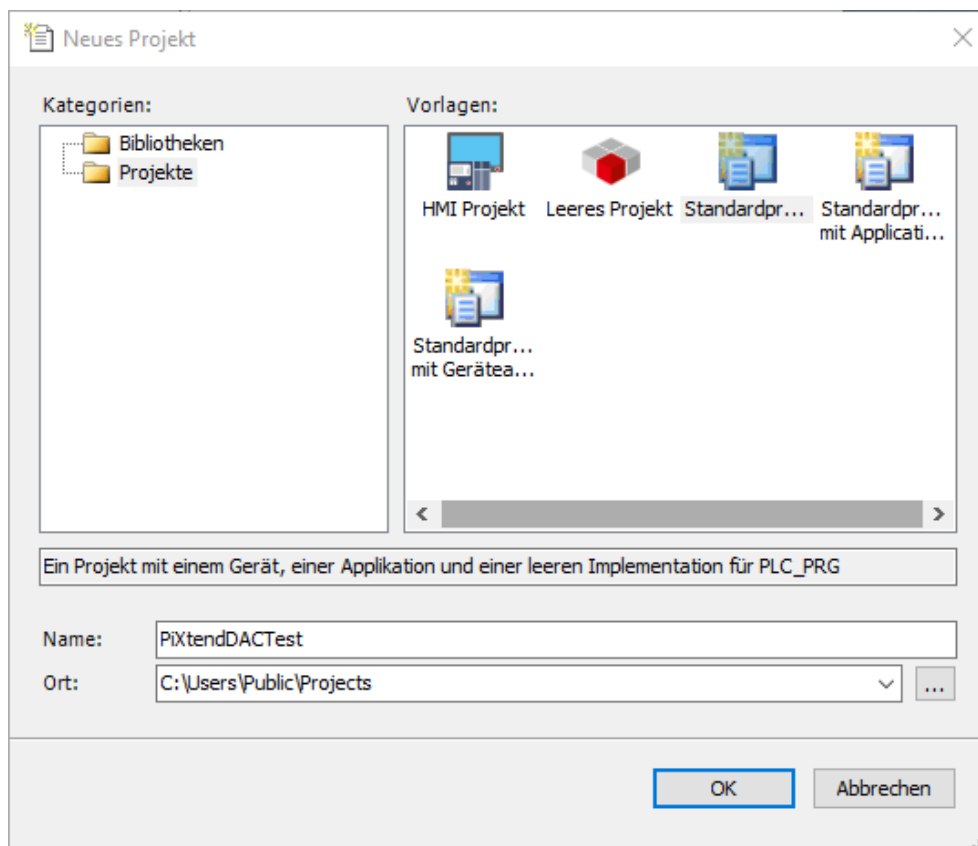


Figure 31: CODESYS - DAC - Create new project

Select "CODESYS Control for Raspberry Pi" as the device and select "Continuous Function Chart (CFC)" as the programming language for the main program PLC_PRG.



PiXtend V2 Software Manual

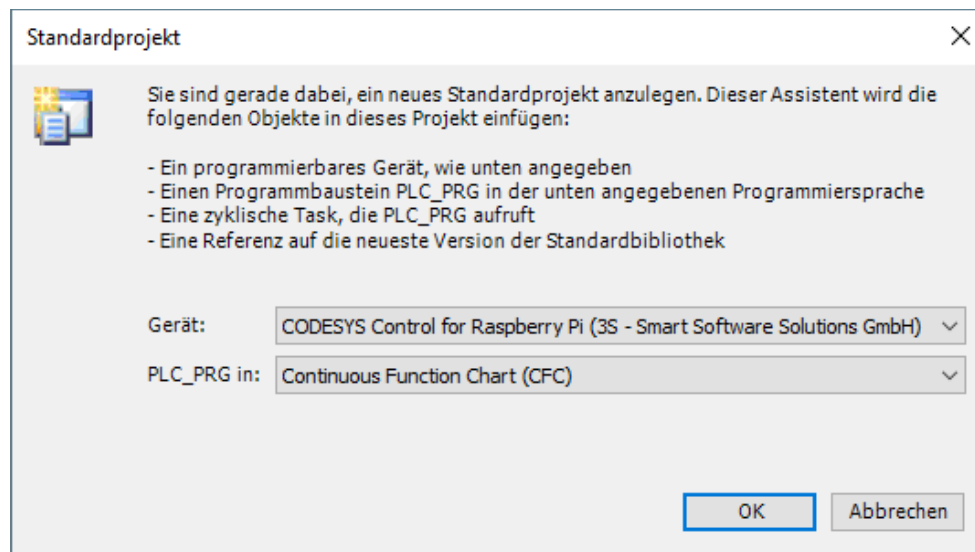


Figure 32: CODESYS - DAC - Choose device

After CODESYS has created the standard project, you receive a project with the following structure:

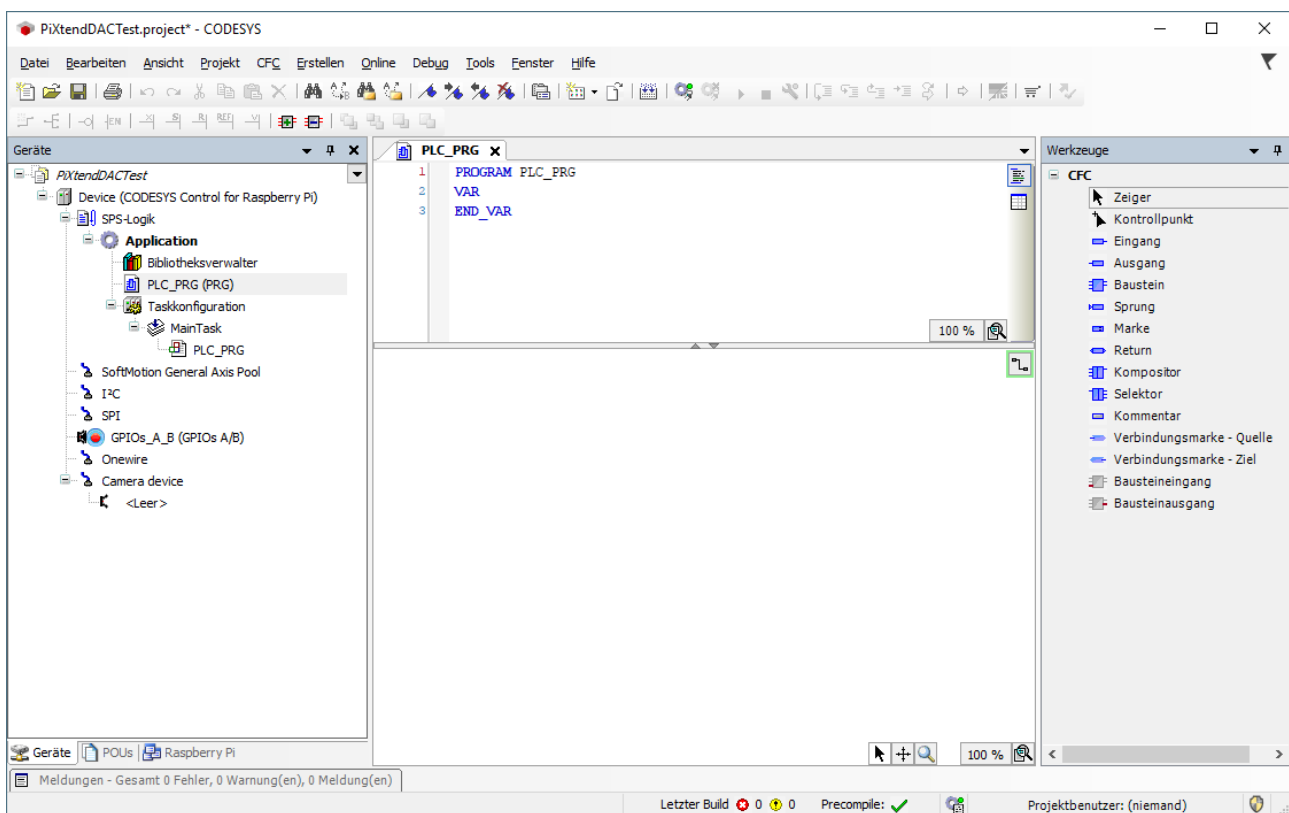


Figure 33: CODESYS - DAC – Project overview



PiXtend V2 Software Manual

7.6.2. Add SPI device

In the project tree, right-click on the entry "SPI" and select "Add device":

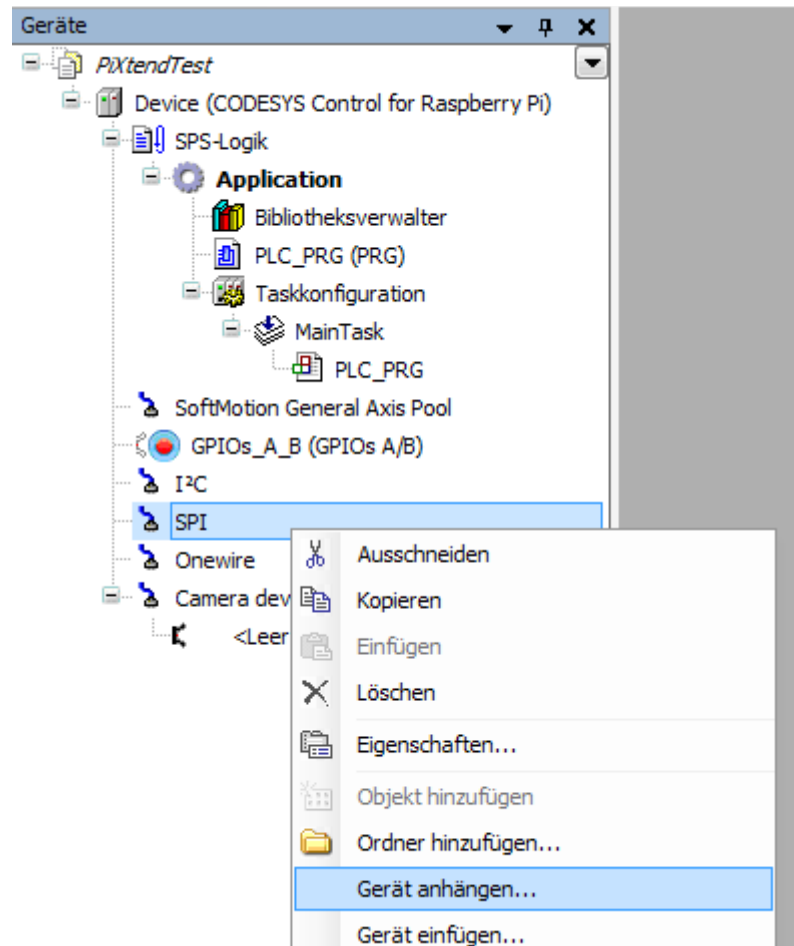


Figure 34: CODESYS - DAC - Add SPI device



PiXtend V2 Software Manual

Select "SPI master" as the device and click on the "Add device" button.

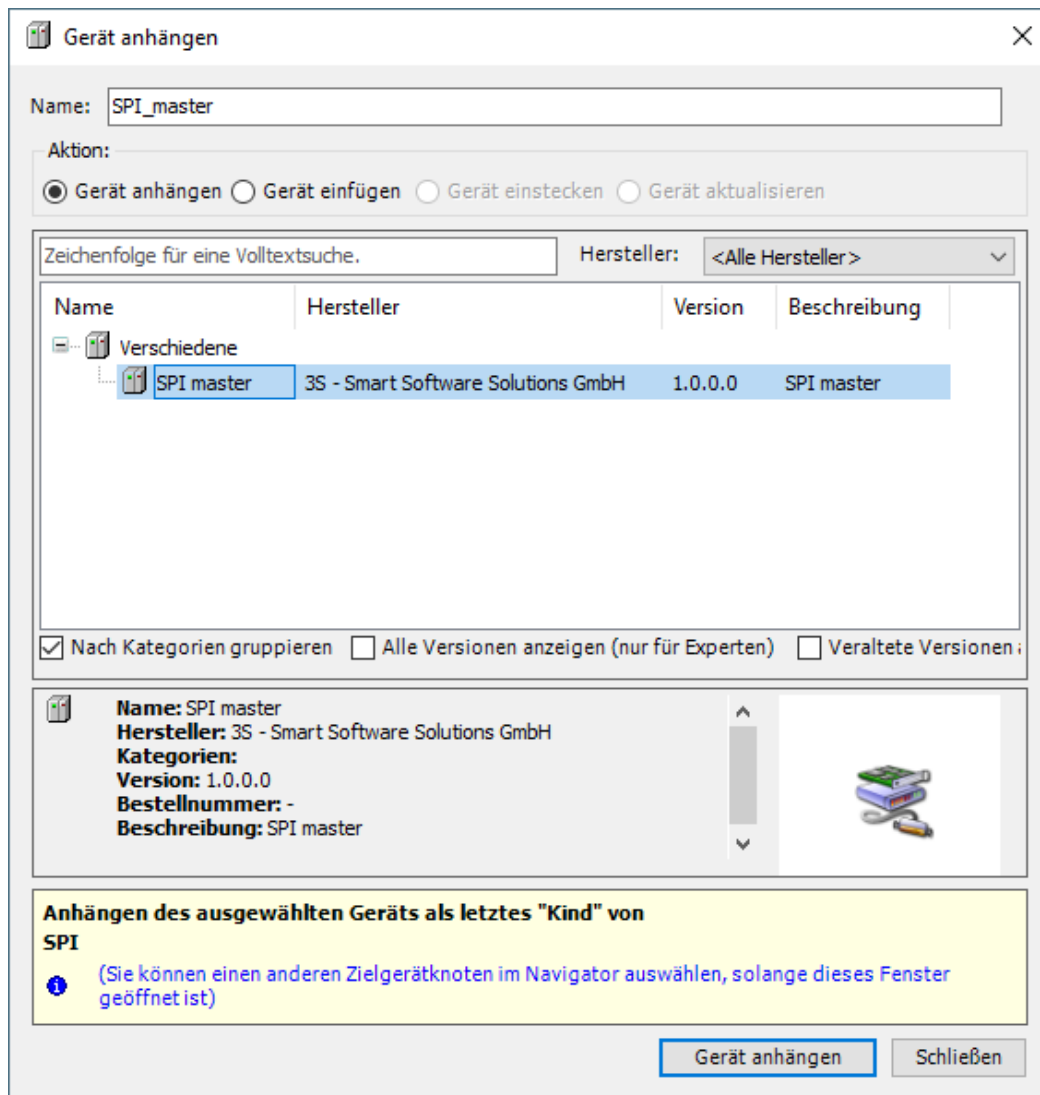


Figure 35: CODESYS - DAC - Add SPI master



PiXtend V2 Software Manual

There is now a new entry "SPI_master (SPI Master)" in the project tree under SPI.

Since the PiXtend V2 DAC is connected to the Raspberry Pi via the SPI port 0.1, you must adjust the parameters of the SPI master. Double click on the entry "SPI_master" and change the SPI port to `"/dev/spidev0.1"`:

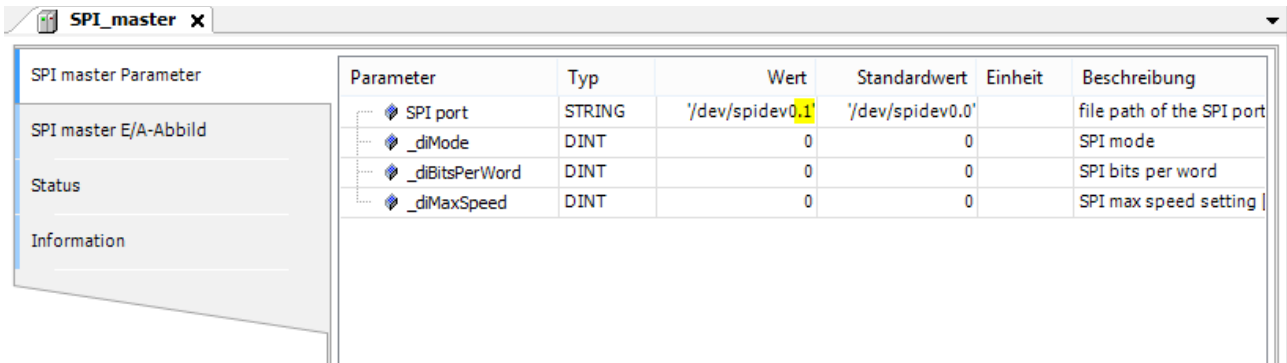


Figure 36: CODESYS - DAC - Configure SPI master

Notice:



The "SPI port" must be set to the device path

`/dev/spidev0.1`

otherwise there will be no communication with the DAC later on the PiXtend V2 board.



PiXtend V2 Software Manual

7.6.3. Add PiXtend V2 DAC device

Select the "SPI_master" entry in the project tree and add another device (right click> Add device).

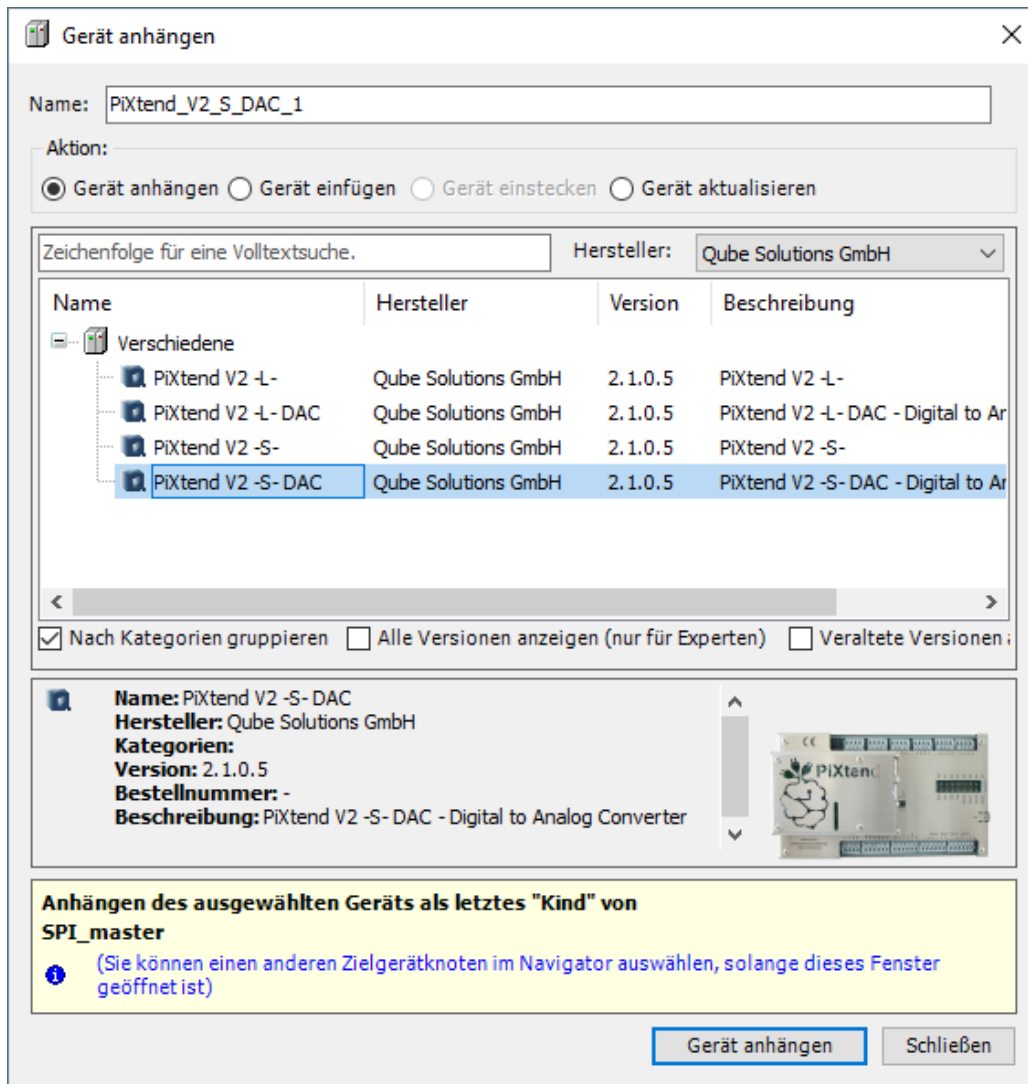


Figure 37: CODESYS - DAC - Add PiXtend V2 DAC device

Select the entry "Qube Solutions GmbH" in the device vendor DropDown menu, select as device "PiXtend V2 -S- DAC", for a PiXtend V2 -S- board or "PiXtend V2 -L- DAC" for a PiXtend V2 -L- board (not "PiXtend V2 -S-" or "PiXtend V2 -L-") and click on the "Add device" button on the bottom right and close the window.



PiXtend V2 Software Manual

Now the PiXtend V2 -S- DAC or PiXtend V2 -L- DAC appears as device under "SPI_master (SPI Master)".

A double-click on the "PiXtend_V2_S_DAC" (here in our example, the name may differ for PiXtend V2 -L- DAC) device in the project tree opens the configuration page for the device. In the "SPI devices I/O Mapping" tab, you can see the two analog outputs that the PiXtend V2 DAC provides for use in CODESYS.

In order for this process image to be cyclically exchanged, two changes are necessary. For "Update variables", please select the entry "Enabled 1 (Use bus cycle task, if not used in any task)"

and for bus cycle task the "PiXtend_Task", if this is not already the case.

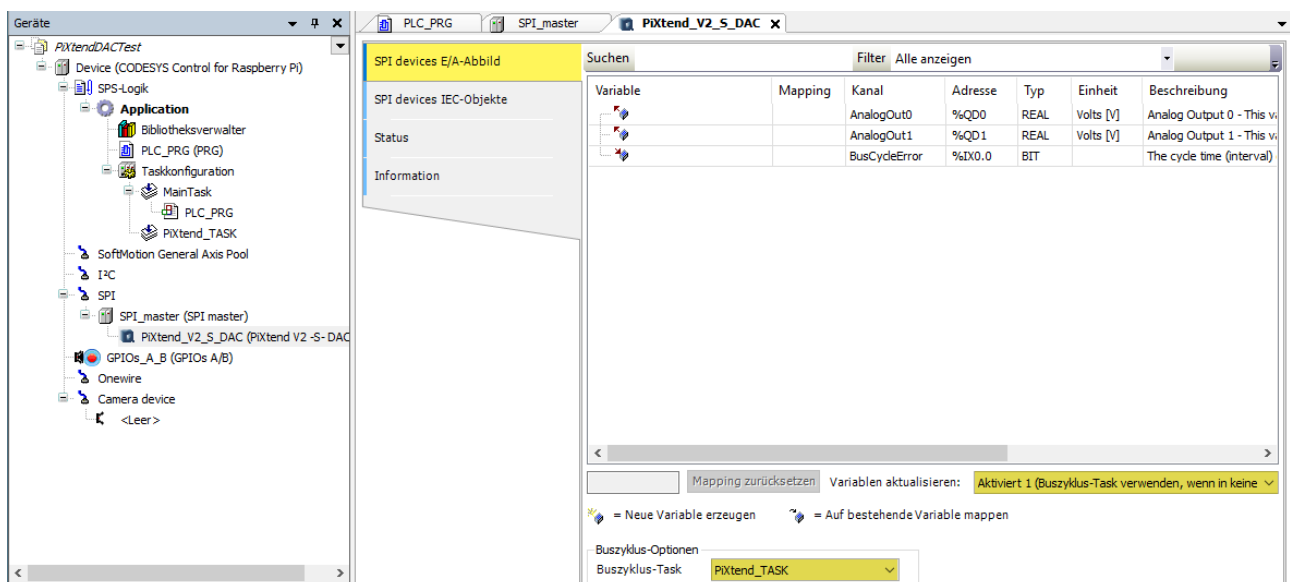


Figure 38: CODESYS - DAC – I/O Mapping settings

Later during operation ("Online control") you can directly assign values to the outputs in this window (using F7 "forcing" values).

However, since we want to access the outputs from the visualization, we create a Global Variables List (GVL) so we can assign variables to the outputs.



7.6.4. Create Global Variable List

Right-click on the "Application" entry in the project tree and add a new Global Variable List named "GVL" with "Add Object" -> "Global Variables List".

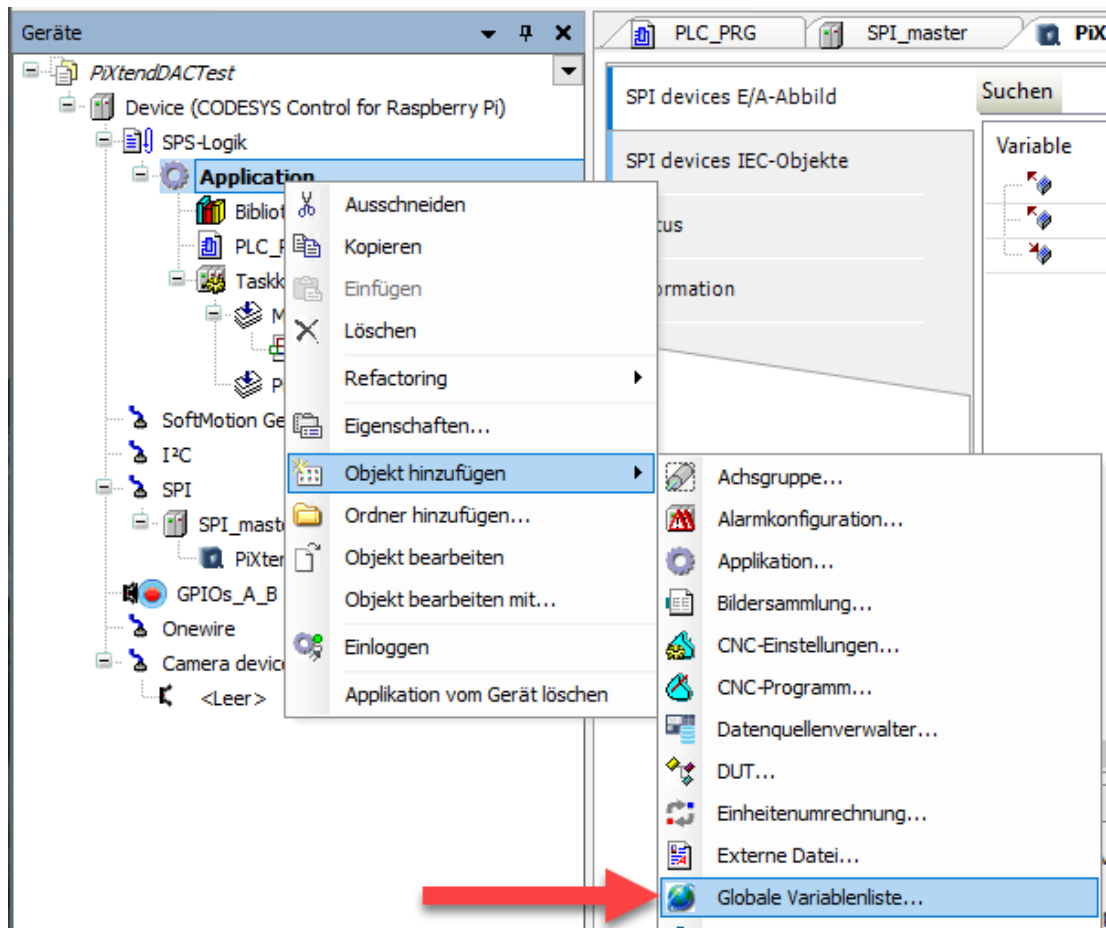


Figure 39: CODESYS - DAC - Add Global Variable List

Open the GVL and add the following entries:

```
//Analog Outputs  
rAOut0: REAL;  
rAOut1: REAL;
```

```
1 VAR_GLOBAL  
2 //Analog Outputs  
3 rAOut0: REAL;  
4 rAOut1: REAL;  
5 END VAR
```

Figure 40: CODESYS - DAC - GVL variables



7.6.5. Mapping of variables

After you have created the required variables in the GVL, they must be "mapped" corresponding to the outputs of the PiXtend V2 -S- DAC (here in our example for a PiXtend V2 -S- board).

Open the already known "SPI devices I/O Mapping" tab again by double-clicking on the "PiXtend_V2_S_DAC" device. Assign the two variables to the outputs

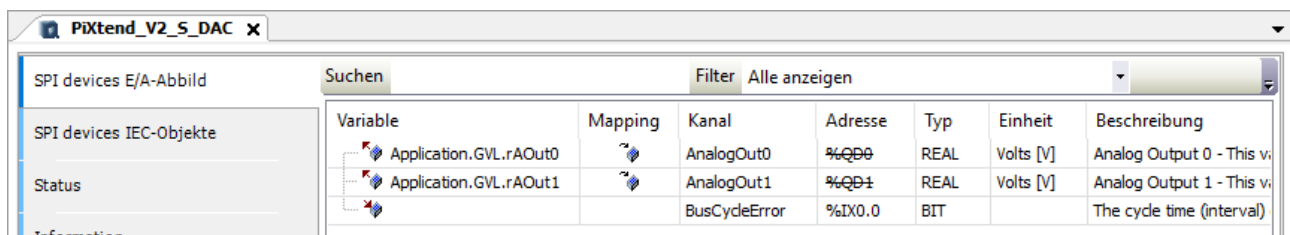


Figure 41: CODESYS - DAC - Variable mapping

by double-clicking on the empty left column, 3 dots appear (...). A click on them opens the input help with which the desired variable is selected.

Select the variable *Application.GVL.rAOut0* for *AnalogOut0*

Select the variable *Application.GVL.rAOut1* for *AnalogOut1*

Finally, the "GPIO bit 24" of the Raspberry Pi must be configured as an output. To do this, open the Raspberry Pi GPIO configuration by double-clicking on "GPIOs_A_B" in the project tree. Select "Output" for "GPIO24".

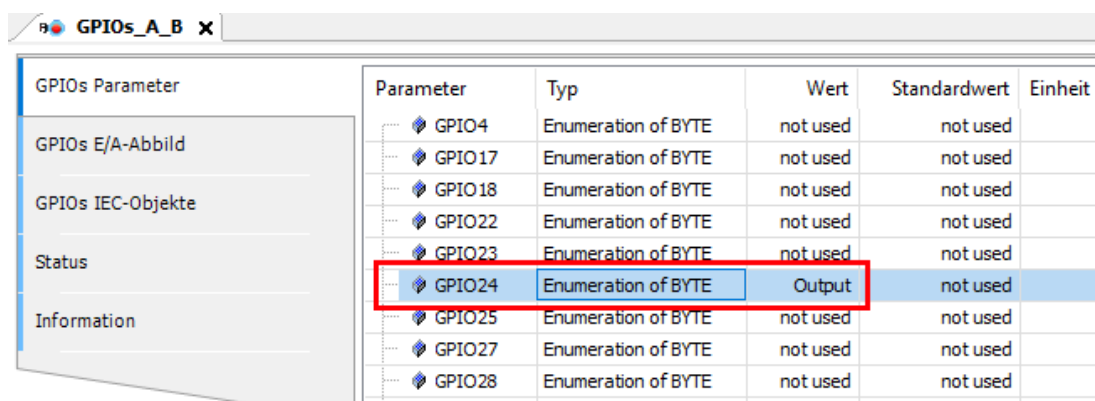
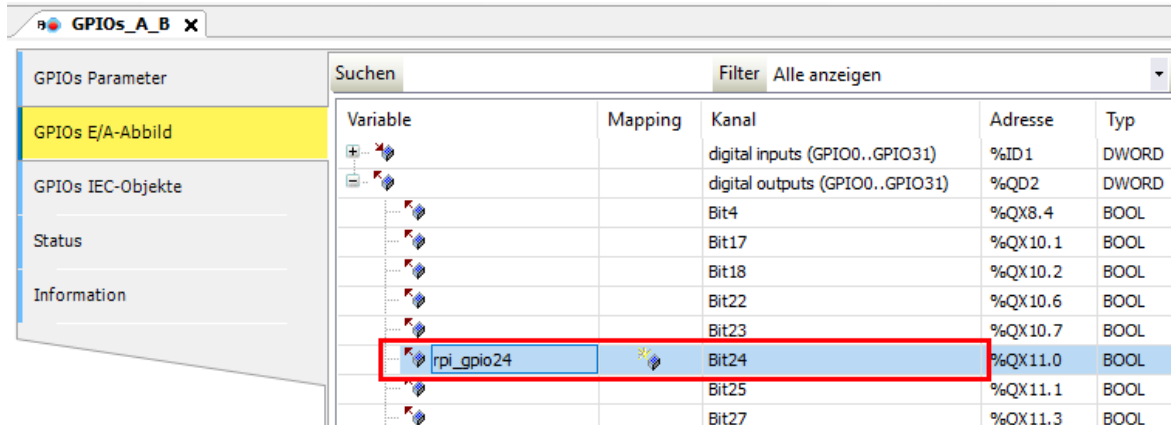


Figure 42: CODESYS - DAC - GPIO 24 as output



PiXtend V2 Software Manual

Then switch to the tab "GPIO I/O Mapping" and click on "**digital outputs**". As variable for "Bit24" enter "rpi_gpio24".



Variable	Mapping	Kanal	Adresse	Typ
		digital inputs (GPIO0..GPIO31)	%ID1	DWORD
		digital outputs (GPIO0..GPIO31)	%QD2	DWORD
		Bit4	%QX8.4	BOOL
		Bit17	%QX10.1	BOOL
		Bit18	%QX10.2	BOOL
		Bit22	%QX10.6	BOOL
		Bit23	%QX10.7	BOOL
rpi_gpio24		Bit24	%QX11.0	BOOL
		Bit25	%QX11.1	BOOL
		Bit27	%QX11.3	BOOL

Figure 43: CODESYS - DAC - GPIO 24

Notice:



The "GPIO 24" must always be set to "TRUE" later in the program so that communication with the PiXtend V2 DAC is possible.

It is best to set the variable "rpi_gpio24" to "TRUE" as the first instruction in your program so that it will not be forgotten later on.



7.6.6. Task Configuration

If desired, you can adjust the cycle time of the PiXtend_TASK. To do this, click Task Configuration → PiXtend_TASK. In principle, the cycle time can be reduced down to $\#1\text{ms}$, but for this test project the preset of 30 ms is sufficient. In addition, if you later want to control the PiXtend V2 board itself, then this value should remain unchanged.

7.6.7. Creating the main program

Double-click PLC_PRG to open the editor and start writing the program code.

In CFC, programming is done graphically. First, drag and drop an "input" block from the toolbox (right side) and place it in the workspace. Then add an "output" block and connect them together.

Click in the middle of a block to assign variables or constants as shown in figure 44:

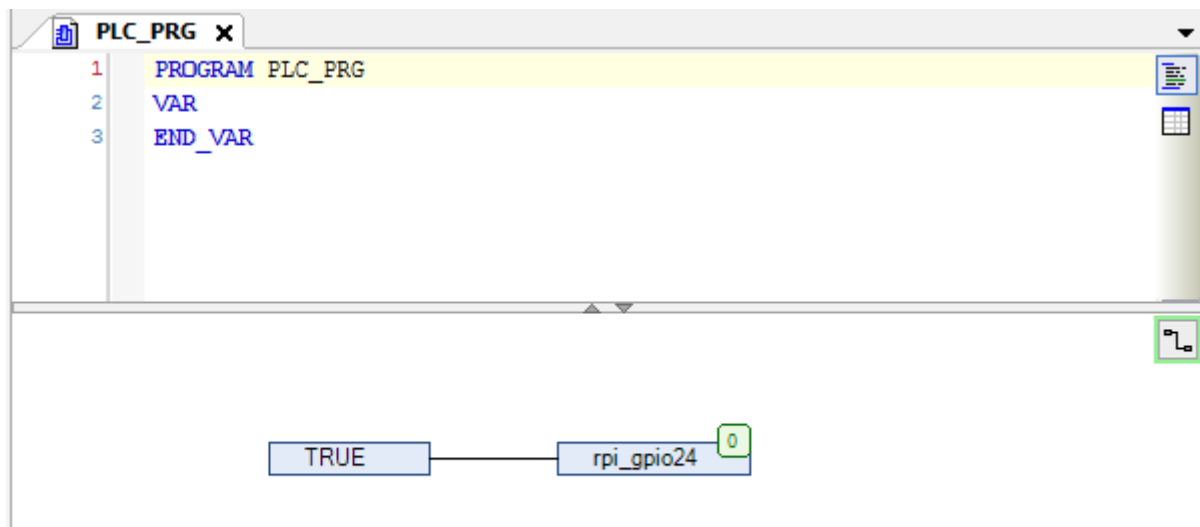


Figure 44: CODESYS - DAC - Assign variable rpi_gpio24

In the menu bar, click on Build → Build to compile the program.



PiXtend V2 Software Manual

7.6.8. Connect to PiXtend V2 and download program

If the program is compiled without errors, double-click on "Device (CODESYS Control for Raspberry Pi)" in the project tree and then on the "Scan Network" button.

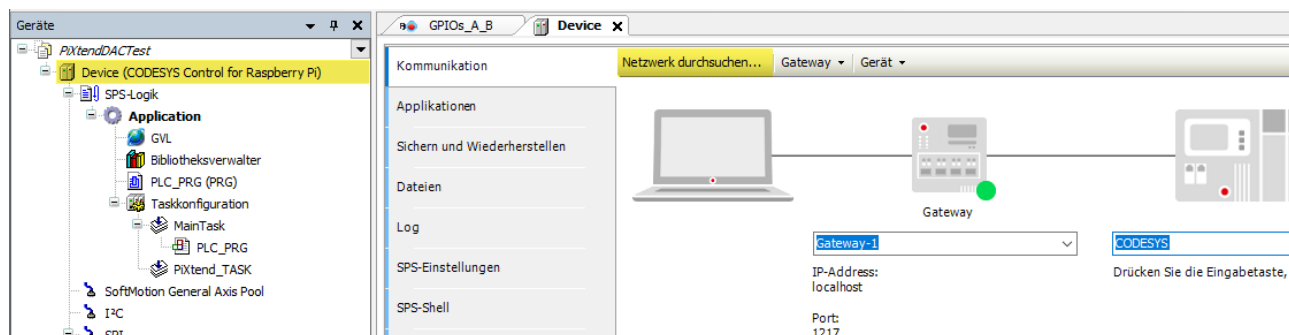


Figure 45: CODESYS - DAC - Scan network

CODESYS searches now in your local network for a Raspberry Pi on which the CODESYS Runtime extension is running.

If no device is found please check the following points:

- Is the correct SD card with the image for the CODESYS runtime extension inserted? A ready-made image can be found in the [download area](#)
- To create your own CODESYS image for the Raspberry Pi, please read the corresponding chapter in the CODESYS online help. Section "Add-ons".
- Is the Raspberry Pi powered on and does it have a valid IP address that you can ping from your PC? (via ping command from the Windows command line)
- The IP address of your Raspberry Pi can also be found with the command "ifconfig" directly in the Raspberry Pi command line, via keyboard and screen.
- If you see a valid IP here, but the ping is not successful, check the network connection to your Raspberry Pi
- If the Raspberry Pi has been on for more than 2 hours, the CODESYS Runtime extension automatically stops and the Raspberry Pi needs to be restarted. (sudo reboot)

Once the Raspberry Pi has been found and selected, click "Online → Login" in the main menu and download the program to the Raspberry Pi.



7.6.9. Create CODESYS Webvisu

We now want to add a simple visualization to the project. To do this, right-click on in the project tree:

"Application → Add object → Visualization"

CODESYS now automatically generates the "Visualization Manager" and an empty visualization called "Visualization". In the manager, parameters for the Webvisu, e.g. the name of the visualization and the preferred resolution can be set.

Change the scaling options to Isotropic:

The screenshot shows the 'WebVisu' settings dialog box. It contains several configuration fields and sections:

- Startvisualisierung:** A text field containing 'Visualization' with a dropdown arrow to its right.
- Name der .htm-Datei:** A text field containing 'webvisu'.
- Aktualisierungsrate (ms):** A text field containing '200'.
- Standardgröße Kommunikationspuffer:** A text field containing '50000'.
- Als Standardseite verwenden:** A checked checkbox.
- Verwendete Visualisierungen anzeigen:** A blue hyperlink.
- Skalierungsoptionen:** A section containing three radio buttons: 'Fest' (unselected), 'Isotropisch' (selected and highlighted in yellow), and 'Anisotropisch' (unselected). Below them is a checked checkbox 'Skalierungsoptionen für Dialoge verwenden'.
- Client Breite:** A text field containing '1280'.
- Client Höhe:** A text field containing '1024'.
- Darstellungsoptionen:** A section containing a checked checkbox 'Zeichnen mit Antialiasing'.
- Standardtexteingabe:** A section containing a dropdown menu 'Eingabe mit:' set to 'Touchscreen'.

Figure 46: CODESYS - DAC - WebVisu settings



PiXtend V2 Software Manual

Double-clicking on "Visualization" opens the editor for the visualization. CODESYS already has a number of ready-made controls. In the tool bar, open the "Common Controls" group and place two "sliders" in the workspace:

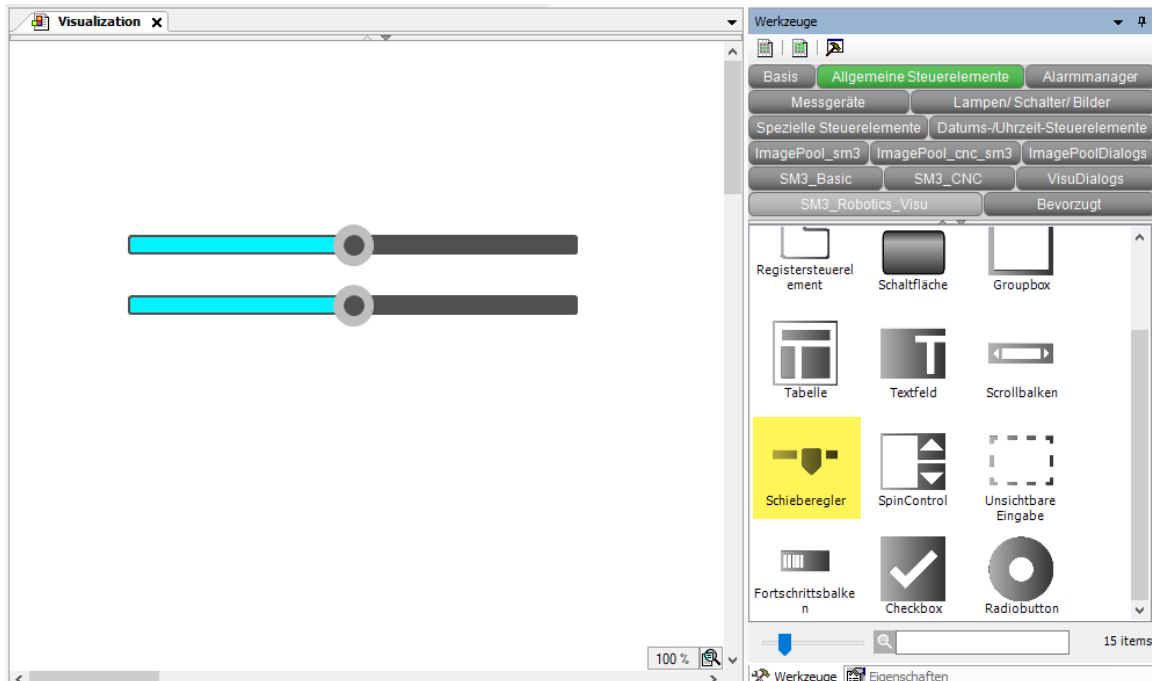


Figure 47: CODESYS - DAC - WebVisu - Sliders

Add two labels and assign the respective variables to the sliders under the "Variable" properties (GVLrAOut0, GVLrAOut1). Change the scale end of both sliders to "10":

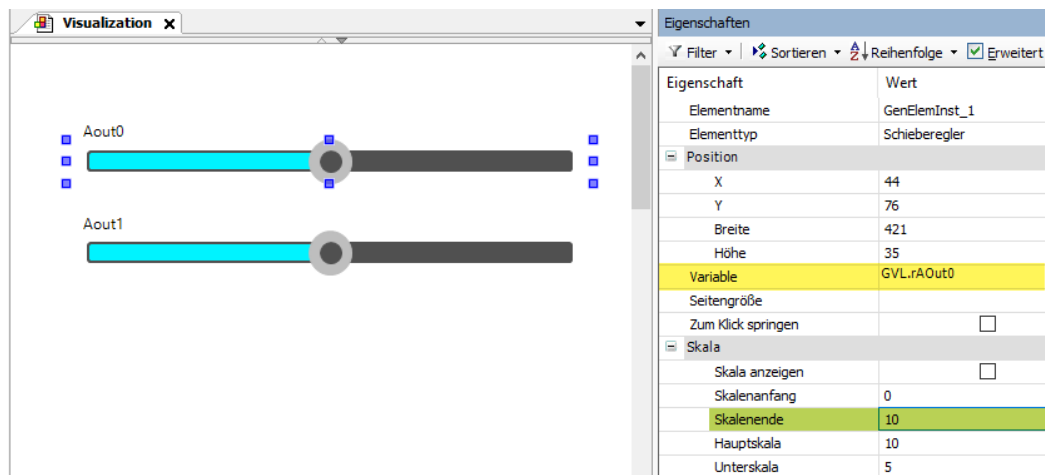


Figure 48: CODESYS - DAC - Sliders - Set scale end



PiXtend V2 Software Manual

Now add two text boxes (rectangles) and enter "% 2.1f" for the "Text" property. This creates a placeholder for a REAL variable with two digits and one decimal place.

Assign the value "GVL.rAOut0" and "GVL.rAOut1" to the property "Text variable".

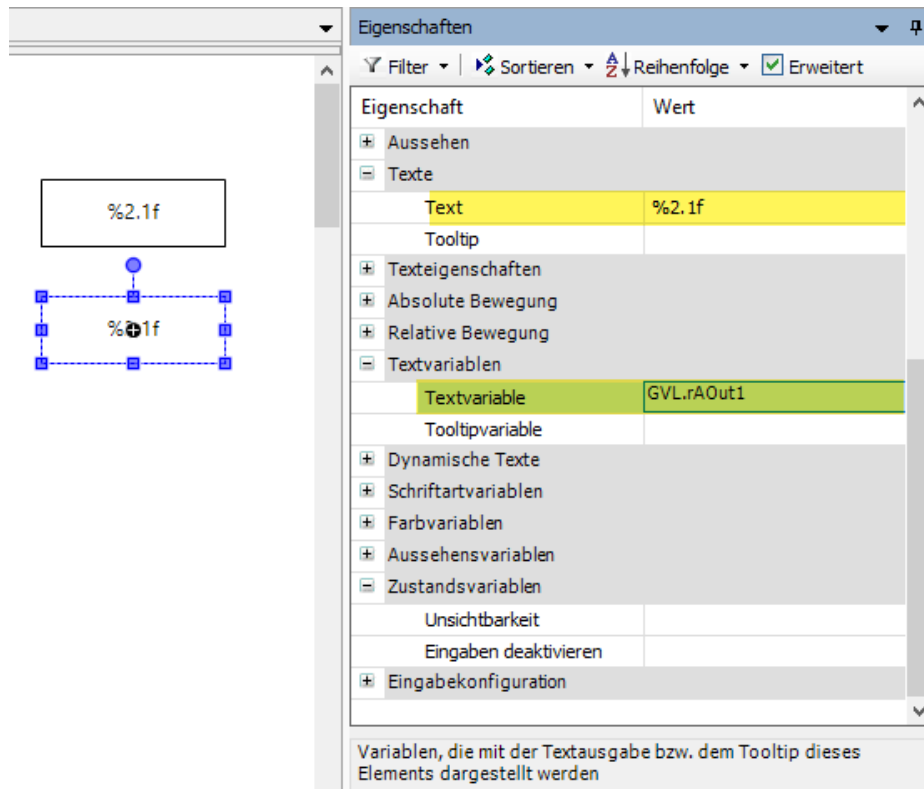


Figure 49: CODESYS - DAC - WebVisu - Create text boxes

Finally, we add a headline and an animated CODESYS logo (Special Controls - Waiting symbol cube) to make the whole thing look nice.



PiXtend V2 Software Manual

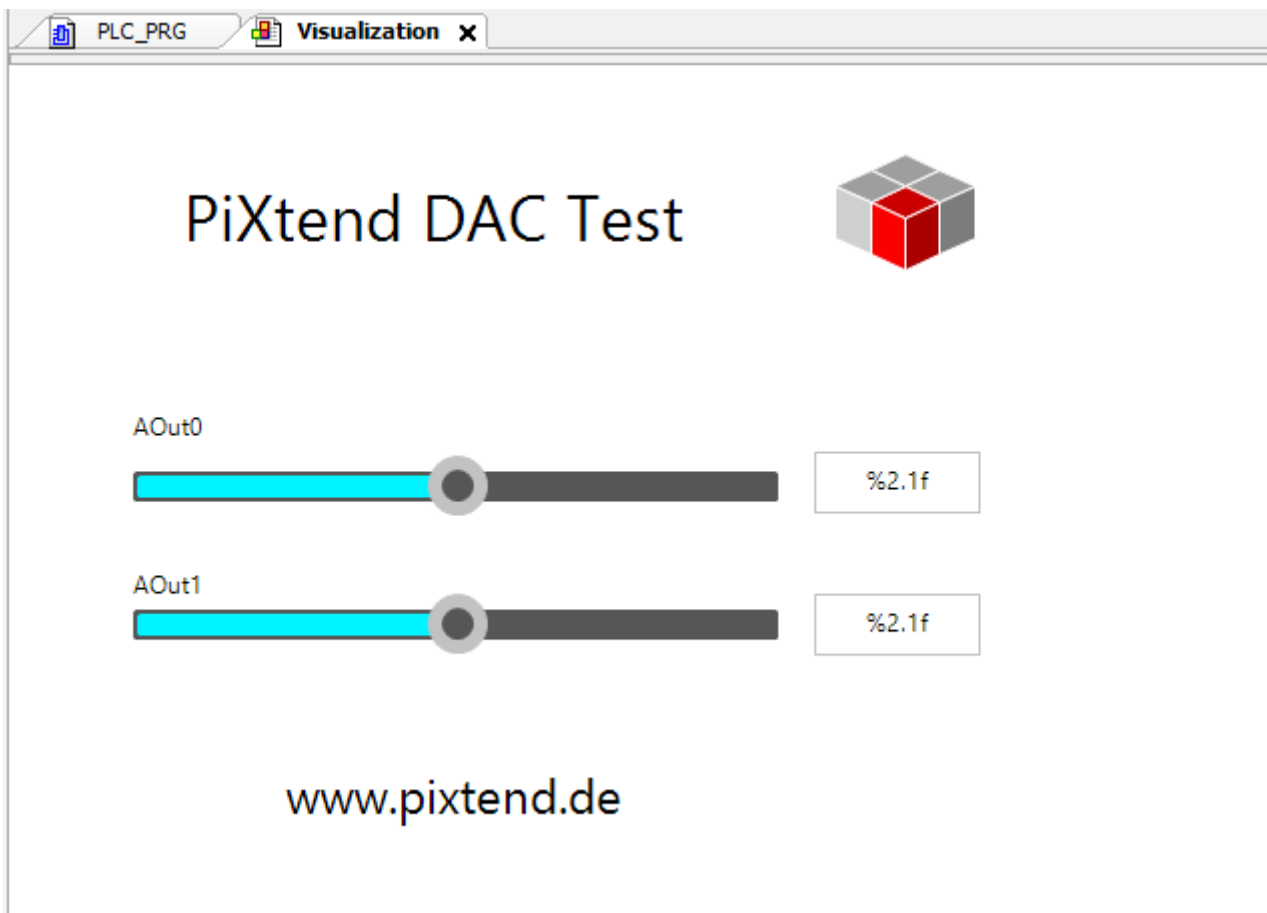


Figure 50: CODESYS - DAC - WebVisu - Finished

Compile the project again using "Build → Build" and perform a complete download.

Open a browser of your choice on your PC or Smartphone/Tablet and enter the IP address of your Raspberry Pi, followed by ":8080/webvisu.htm", ie.

<http://192.168.137.99:8080/webvisu.htm>

You can now move the sliders to change the voltage of the two analog outputs.



PiXtend V2 Software Manual

Notice:



If the PiXtend V2 board is to be used parallel to the PiXtend V2 DAC, make sure that you attach the PiXtend V2 device to it's own (additional) SPI master and not to the same one where the DAC is already connected.

Example for PiXtend V2 -S- (all specifications apply also to the PiXtend V2 -L-):

Parameter	Typ	Wert
SPI port	STRING	"/dev/spidev0.1"
_diMode	DINT	0
_diBitsPerWord	DINT	0
_diMaxSpeed	DINT	0

Parameter	Typ	Wert
SPI port	STRING	"/dev/spidev0.0"
_diMode	DINT	0
_diBitsPerWord	DINT	0
_diMaxSpeed	DINT	0

Figure 51: CODESYS - Notice on how to use both PiXtend V2 SPI devices

On the first SPI master (*SPI_Master*, highlighted in green) the PiXtend V2 -S- DAC is present, because we added it first to the SPI bus in this CODESYS DAC example. The PiXtend V2 -S- DAC device is highlighted in yellow. Also note the *SPI port* setting for the DAC, the communication uses here the second *Chip Select* of the Raspberry Pi, therefore, the device **"/dev/spidev0.1"** (yellow highlighted on the right side) must be used here.

The PiXtend V2 -S- board itself must be controlled via another, second SPI master. Highlighted in red in the picture above with the name "*SPI_master_1*". The PiXtend V2 -S- device can be added to this second SPI master (highlighted again in yellow) and used here. The communication uses the first *Chip Select* of the Raspberry Pi, therefore the device **"/dev/spidev0.0"** (highlighted yellow on the right side, lower part) must be used here.

Of course you can also add the PiXtend V2 -S- to the first and the DAC to the second SPI master, but remember to adjust the *SPI port* setting for each device accordingly.



7.7. CODESYS Serial Communication

This chapter describes all the steps necessary to establish communication between the Raspberry Pi (CODESYS) and a Windows PC (terminal), via the RS232 interface of *PiXtend V2*.

7.7.1. Prerequisites

The steps in this chapter assume that you have read the previous chapters, in particular Chapter 7.2 Prerequisites and 7.3 Installation of the required Software Components, and the CODESYS Development System with the 3 components CODESYS, CODESYS Control for Raspberry Pi and PiXtend V2 Professional for CODESYS are already installed successfully.

7.7.2. RS232 Interface - Connecting the cable

To use the RS232 interface, the data lines must be cross-connected, i.e. RX to TX.

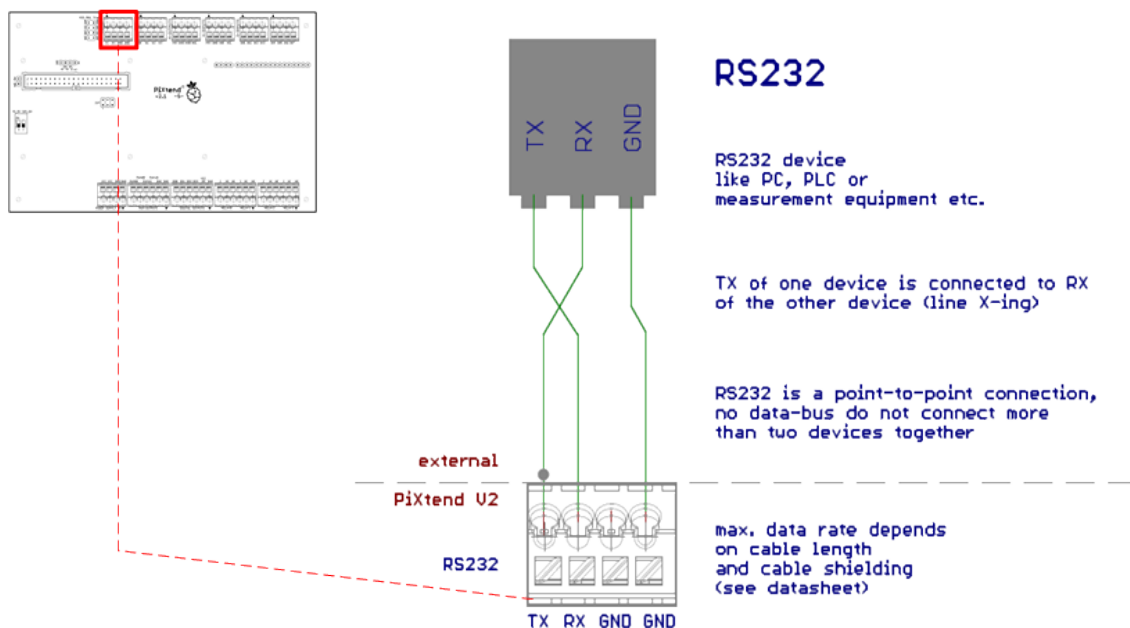


Figure 52: CODESYS - Circuit Diagram - RS232

A USB-to-Serial adapter is required to connect to a PC.

Further information on RS232 can be found in the data sheet (*PiXtend V2 -S-* or *PiXtend V2 -L-* technical data sheet in the hardware manual).



PiXtend V2 Software Manual

7.7.3. RS485 Interface - Connecting the cable (only PiXtend V2 -L-)

To use the RS485 interface, the data lines must be connected 1: 1, i.e. each A (\bar{A}) at A (\bar{A}) and B (\bar{B}) at B (\bar{B}).

To be able to use the RS485 interface, the GPIO 18 of the Raspberry Pi must be activated, whereby the serial interface of the Raspberry Pi is switched over to the RS485 hardware chip. Use of the RS232 interface is no longer possible after switching.

Notice:



The RS485 interface is only available on the *PiXtend V2 -L-*!

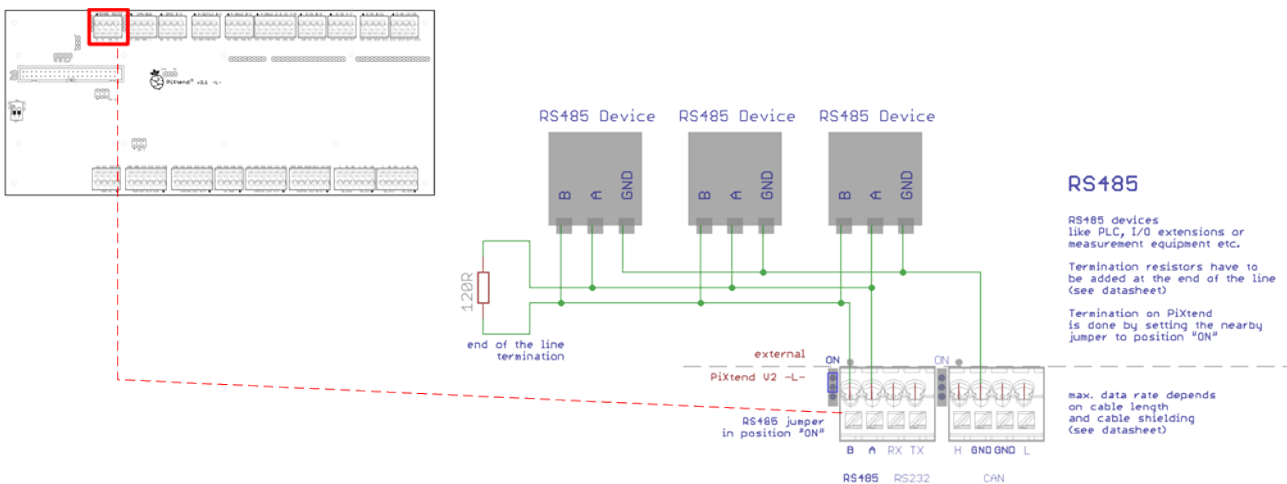


Figure 53: CODESYS - Circuit Diagram - RS232



7.7.4. Preparing Linux

7.7.4.1 Adjustment of the interface settings

First, the interface settings of the Raspberry Pi must be adjusted, so that the serial interface can be used with CODESYS. This means that the serial interface no longer has the console on it and no longer sends data, the Linux console via the serial interface is deactivated.

The serial interface of the Raspberry Pi can be activated or changed via the program "raspi-config":

```
sudo raspi-config
```

In the program you switch to:

5 Interfacing Options --> P6 Serial --> <No> --> <Yes> --> <Ok>

The following entry is found in the file /boot/config.txt:

```
enable_uart=1
```

Bluetooth is then automatically deactivated and UART activated. The program "raspi-config" does all the work for us here.

If you are using our CODESYS SD card image, you can now restart the Raspberry Pi and then continue with chapter 7.7.5 Terminal Program.

7.7.4.2 Activate the Interface in the CODESYS Configuration

Next, the interface must be "made known" to CODESYS. The interface must be entered into the CODESYS configuration so that CODESYS can access it.

Since CODESYS V3.5 SP11 (V3.5.11.x):

For SP11, the two named lines must be "commented out". This is achieved with a semicolon ";" before the rows. Note also that the changes no longer have to be carried out in the CODESYSControl.cfg, but in the CODESYSControl_User.cfg.

Use the following command:

```
sudo nano /etc/CODESYSControl_User.cfg
```

then add the lines at the very end of the file:

```
[SysCom]
;Linux.Devicefile=/dev/ttyS
;portnum := COM.SysCom.SYS_COMPORT1
```



PiXtend V2 Software Manual

On the SD image "PiXtend Image CODESYS V2.X.X" with SP13 runtime, the lines are already entered. You do not have to make an adjustment to the CODESYSControl_User.cfg if you work with this image!

However, if a serial USB interface is to be used, e.g. another RS232/RS485 dongle, then the semicolons must be removed and the name of the Linux device must be adapted:

```
[SysCom]
Linux.Devicefile=/dev/ttyUSB
portnum := COM.SysCom.SYS_COMPORT1
```

Make sure that there is no number behind ttyUSB, this must be omitted, otherwise access from CODESYS will not work.

You can find the device or the device name with the following command:

```
dmesg | grep -i tty
```

7.7.4.3 Raspberry Pi Restart

Now only the Raspberry Pi has to be restarted, so the settings are activated.

```
sudo reboot
```




PiXtend V2 Software Manual

7.7.5. Terminal Program

If a connection with a PC is to be established, a terminal program is required for the communication.

7.7.5.1 Terminal Program Installation

If no terminal program is available, the next step should be completed. In our example, we use HTerm (<http://www.derhammer.info/terminal/>).

7.7.5.2 Open Terminal and Settings

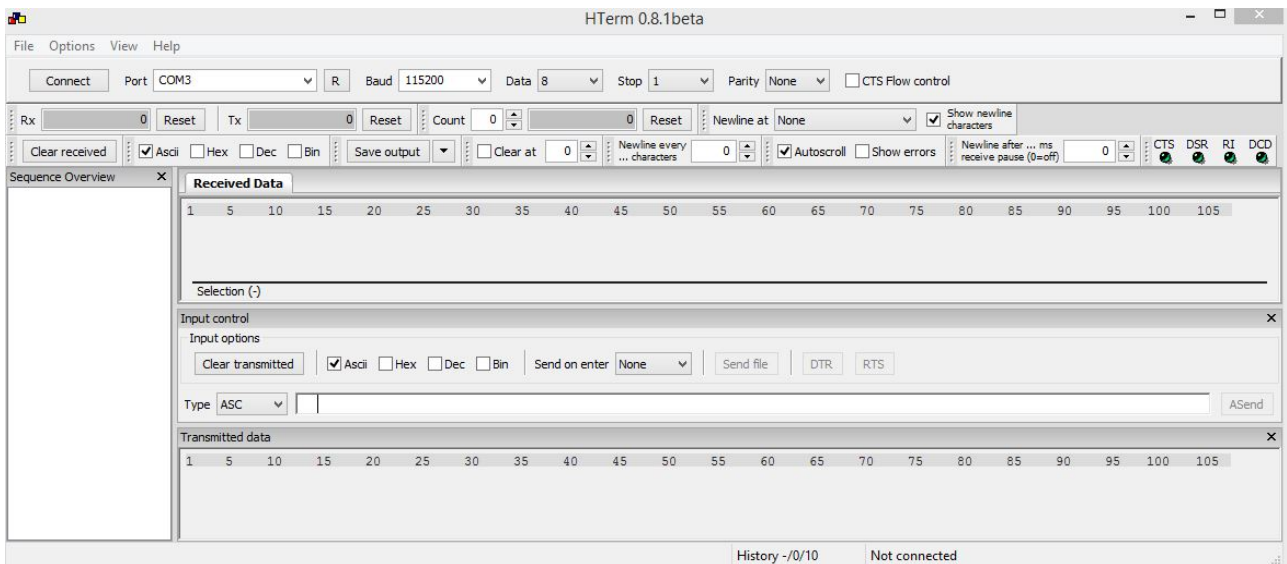


Figure 54: CODESYS - HTerm 0.8.1beta - RS232 Testprogram

Settings:

- Port: In the Device Manager under Connectors (COM & LPT)
- Baudrate: as described in CODESYS, e.g. 9600 Baud
- Data: 8
- Stop: 1
- Parity: none

Now you only have to connect using the "Connect" button. Now data can be received and sent.



PiXtend V2 Software Manual

7.7.6.CODESYS

7.7.6.1 CODESYS - 'PiXtendSerialTest' project

The first step is to open a project called "PixtendSerialTest", transfer the program to the Raspberry Pi and start it. The test project can be downloaded from <https://www.pixtend.de/downloads>.



Note that this project is also designed for PiXtend V1.2 / 1.3, on the *PiXtend V2 -S-* no RS485 interface is available. Ignore the "Enable RS485" switch.

The RS232 part of the project is fully compatible with *PiXtend V2 -S-*.

The *PiXtend V2 -L-* has an RS485 interface and the RS485 test can be performed.

7.7.6.2 Visualization



Serial Communication via RS232/RS485



Receive ☒ auto scroll

Timestamp | Message

RX: 0 Bytes TX: 0 Bytes

Send

Auto send ☐ enable interval [s] 5.0

CONNECT

DISCONNECT

Status

Disconnected

CLEAR ALL

SEND

OpenError

CloseError

ReadError

WriteError

Error Code

0

Baudrate

9600

Enable RS485

1 0

PiXtend
www.pixtend.de

Figure 55: CODESYS - Visualization - Test program - Serial Communication



PiXtend V2 Software Manual

Quick Start:

1. Set the baud rate (must match the communication partner settings)
2. Press "CONNECT" button
Data is now automatically received and displayed. To send a message:
3. Enter message in the "Send" field
4. Press the "SEND" button

Function of the buttons, switches and selection fields:

„CONNECT / DISCONNECT“

Connect or disconnect to the PC terminal or any other device.

„CLEAR ALL“

The received messages, received (RX) and transmitted (TX) bytes and errors are deleted. In addition, the automatic sending of messages is disabled.

„SEND“

Sends the message entered in the input field.

„Baudrate“

The desired baud rate (bits / second) can be selected here.

„Enable RS485“ (only PiXtend V1.2/V1.3 and PiXtend V2 -L-!)

This switch can be used to switch between the RS232 and the RS485 interface. For this purpose, however, a disconnect must first be carried out, since both interfaces can not be operated simultaneously.

„auto scroll“

If this checkbox is active (checked), the display of the received messages always scrolls automatically to the most recent message.

„enable“ Auto send

Activates the automatic transmission of the entered information at the set interval.



PiXtend V2 Software Manual

Input / output fields:

„Receive“

Here the received data is displayed in tabular form with time stamps.

„Send“

The data to be sent is entered here.

„Auto send“

Data will be sent automatically.

„interval“

Interval in which the data is to be sent automatically. Minimum is 1 second.

Status indicators:

„Error Code“

If an error occurs during opening, closing, reading or writing during operation, this is indicated by the error LEDs. The corresponding error code is displayed in the "Error code" field.

„Status“

Displays the status of the program.



7.8. CAN-Bus Communication

7.8.1. introduction

The CAN bus originally came from the automotive sector (developed by Bosch). Today, the robust and reliable bus is also used in many other areas, e.g. industrial automation. Automation technology often uses the CANopen protocol (OSI layer 7 - application layer), which, fortunately, is also supported by CODESYS and is available for the Raspberry Pi. So it's worth taking a closer look at the CAN bus and the CANopen stack.

In this chapter all steps are described which are necessary to set up a CANopen communication under CODESYS with *PiXtend V2 -L-*.

For the steps in this chapter you need an SD card with the PiXtend V2 SD card image "CODESYS Control". Then the CANopen configuration is described in CODESYS.

Notice:



The CAN bus interface is only available on the PiXtend V2 -L-!

You will also find many more information, tips and tricks in our support forum at: <https://www.pixtend.de/forum/>

Should any questions remain unanswered, please let us know by e-mail (support@pixtend.com). You will receive an answer as soon as possible and further information.

The latest versions of all documents and software components can be found in the download area of our homepage: <https://www.pixtend.de/pixtend/downloads/>



7.8.1.1 Requirements

For the activation of the CAN-Bus functionality basics are necessary in the handling of the Raspbian operating system (Linux), as well as with the operation of CODESYS itself.

In order to test the CAN communication under Linux, the *PiXtend V2 -L-* must be connected to a CAN bus with at least one active CAN device. Of course, it can also be connected to one or more other PiXtends (PiXtend V1.3 and *PiXtend V2 -L-* alike).

For the CODESYS CANopen master / slave communication two *PiXtend V2 -L-* boards are used in this chapter.

7.8.1.2 Restrictions

If the *PiXtend V2 -L-* CAN controller is used, the *PiXtend V2 -L-* DAC can not be used simultaneously. The two chips "share" the same SPI chip-select channel.

7.8.2. Hardware connection to the CAN bus

The correct pin assignment of the *PiXtend V2 -L-* CAN port can be found in the *PiXtend V2 -L-* hardware manual.

Warning:



In order to be able to use the CAN bus hardware on *PiXtend V2 -L-*, the jumper "CAN" / "AO" must be set to position "CAN".

Otherwise, the CAN bus does not work.

The DAC is disabled by this action and can no longer be used.

Please also note the jumper for the "*termination*" of the CAN bus.



7.8.3. Preparation and test under Linux

So that the CAN controller can later be used in CODESYS, some adjustments to the Raspbian (Linux) operating system are necessary.

If you have already made your own adjustments to your image, we recommend creating a new SD card for the first CAN tests.

We recommend using our "CODESYS Control" SD card image however later on for working with CODESYS itself, which you will always find in the latest version in our [download area](#).

Notice:



All steps described here refer to the Raspbian Image "Stretch" and are performed as user "pi".

The support of the CAN controller MCP2515 used on the *PiXtend V2 -L-* is only guaranteed as of kernel version 4.0.8. If you still use a "Wheezy" image, you must first upgrade to a kernel version 4.0.8 (or later).

Warning:



If you use an SD card with "CODESYS Control" image, then you must stop the CODESYS runtime component in order to be able to commission the CAN bus.

Use the following command:

```
sudo service codesyscontrol stop
```

After restarting the Raspberry Pi, CODESYS will be available as usual without the need for further intervention. Until the basic work is done, the CODESYS runtime has to be stopped again after each reboot.



PiXtend V2 Software Manual

7.8.3.1 Preparation of the SD card image

Download the Raspbian "Stretch" image from the official download page:

<https://www.raspberrypi.org/downloads/raspbian/>

Transfer the downloaded image to an SD card (8 GB recommended) with a tool of your choice. Under Windows, for this task, e.g. the free program Win32DiskImager can be used. See also chapter 6.4 Preparing a SD card.

Boot the new image and do the usual customizations using raspi-config:

- Extend the file system
- Setting your time zone
- Regional settings
- Language settings
- Keyboard layout
- Enable SSH access

Now restart the Raspberry Pi with the command

```
sudo reboot
```

and then log in via SSH in the command line as a user pi (for example, with Putty) or work directly with the screen and keyboard.

Tip:

If you use our SD card image "CODESYS Control", you can skip sections 7.8.3.3 and 7.8.3.4, as we have already prepared everything for you.

Only a small adjustment is necessary, which is described in 7.8.3.2. With the difference that the lines needed already exist and you just have to remove the # sign in front of the corresponding lines.



PiXtend V2 Software Manual

7.8.3.2 Configuration of the Device Tree Overlay

The following adjustments to the file `/boot/config.txt` are necessary to be able to operate the MCP2515 CAN chip of the PiXtend board from the kernel. Open the editor with the command

```
sudo nano /boot/config.txt
```

and add the following entries at the end:

```
dtparam=spi=on
dtoverlay=mcp2515-can1
dtparam=oscillator=20000000
dtparam=interrupt=4
dtparam=spimaxfrequency=1000000
dtoverlay=spi-bcm2835-overlay
dtoverlay=spi-dma
dtdebug=on
```

Save the changes with `Ctrl-O` and exit the editor with `Ctrl-X`.

Restart the Raspberry Pi now:

```
sudo reboot
```

7.8.3.3 Edit blacklist

So that the kernel module "mcp251x" does not load automatically when booting, add it to the blacklist.

Open the file `/etc/modprobe.d/raspi-blacklist.conf`

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

and add the following line:

```
blacklist mcp251x
```

The file is initially completely empty, but can still be used.

Save the changes with `Ctrl-O` and exit the editor with `Ctrl-X`.

Restart the Raspberry Pi now:

```
sudo reboot
```



7.8.3.4 Create CAN script

In order to activate the *PiXtend V2 -L-* CAN controller, the following actions must be taken:

- GPIO 24 configure as output and set to TRUE (SPI Enable)
- GPIO 27 configure as output and set to TRUE (SPI CS1 is passed on to the CAN controller, the DAC is deactivated)
- Load the kernel module "mcp251x"
- Configure and enable the interface can0

Since the same steps are always necessary, we create a new script in the home directory of the user pi:

```
nano activateCAN.sh
```

Insert the following lines:

```
#!/bin/sh
if [ ! -d "/sys/class/gpio/gpio24" ]; then
    echo "24" > /sys/class/gpio/export
fi
echo "out" > /sys/class/gpio/gpio24/direction
echo "1" > /sys/class/gpio/gpio24/value
if [ ! -d "/sys/class/gpio/gpio27" ]; then
    echo "27" > /sys/class/gpio/export
fi
echo "out" > /sys/class/gpio/gpio27/direction
echo "1" > /sys/class/gpio/gpio27/value
sleep 1
sudo modprobe mcp251x
sudo /sbin/ip link set can0 up type can bitrate 125000
sudo ip -s -d link show can0
```

Save the changes with Ctrl-O and exit the editor with Ctrl-X.

Make the script executable by typing the following command:

```
chmod +x activateCAN.sh
```



PiXtend V2 Software Manual

7.8.3.5 Activate CAN

If you now run the script using

```
sudo ./activateCAN.sh
```

the CAN controller will be activated.

With a call of

```
ifconfig
```

we can check if there is an entry for a *can0* interface.

7.8.3.6 Installing and using CAN utilities

Now install the "can-utils", a collection of useful user-space programs to access the CAN interface directly with SocketCAN under Linux:

```
sudo apt-get install can-utils
```

With the command

```
cansend can0 123#DEADBEEF
```

a CAN message with ID 123 and the content 0xDEADBEEF is sent.

With the command

```
candump can0
```

All traffic on can0 can be listened to:

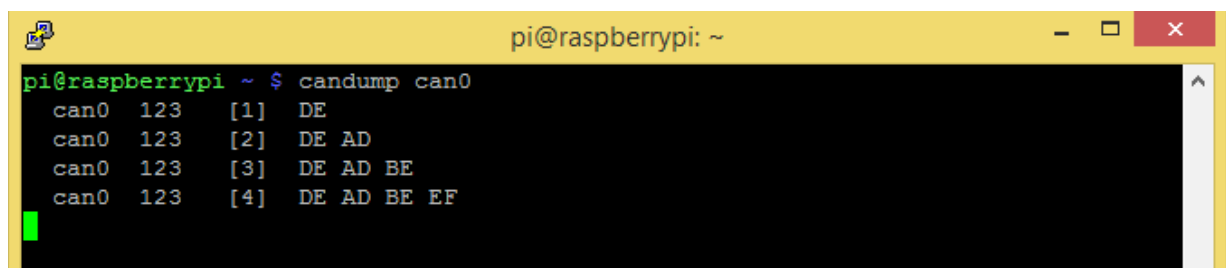


Figure 56: CODESYS - CAN-Bus Test under Linux

With the program "cangen" CAN messages can be generated continuously for test purposes.



PiXtend V2 Software Manual

Further useful functions and parameters of can-utils can be found in the respective help pages.

With these tools you can already participate in a CAN bus communication under Linux.

Please continue with the next steps (preparation for CODESYS) only if you have successfully received and sent data with the "can-utils".



7.8.4. Preparations for CODESYS

For the steps in this chapter, you need an SD card with CODESYS Runtime installed. Regardless of whether you are just starting with the CAN bus or have already built up your own image and system, we recommend that you first use our SD card image "CODESYS Control", which you can download from our [download area](#) begin.

7.8.4.1 Create start script

So that the kernel module for the *PiXtend V2 -L-* CAN controller can be loaded correctly, we need a startup script that performs the following actions in exactly this order:

- Stop CODESYS Control
- Activate CAN using the previously created script "activateCAN.sh" (See topic 7.8.3.4)
- Start CODESYS Control

Open a console on the Raspberry Pi (for example, via SSH with the tool Putty) and create the file "startPLCCAN.sh" in the home directory:

```
nano startPLCCAN.sh
```

Insert the following lines:

```
#!/bin/sh
sudo /etc/init.d/codesyscontrol stop
sudo /home/pi/activateCAN.sh
sudo /etc/init.d/codesyscontrol
```

Save the changes with Ctrl-O and exit the editor with Ctrl-X.

Make the script executable by typing the following command:

```
chmod +x startPLCCAN.sh
```



PiXtend V2 Software Manual

7.8.4.2 Creating a baud rate script

Now create the following baud rate script:

```
sudo nano /var/opt/codesys/rts_set_baud.sh
```

and add the following lines:

```
#!/bin/sh
BITRATE=`expr $2 \\* 1000`
ifconfig $1 down
echo ip link set $1 type can bitrate $BITRATE
ip link set $1 type can bitrate $BITRATE
ifconfig $1 up
```

Save the changes with Ctrl-O and exit the editor with Ctrl-X.

Make the script executable by typing the following command:

```
sudo chmod +x /var/opt/codesys/rts_set_baud.sh
```

7.8.4.3 Starting CODESYS Control with PiXtend V2 -L- CAN support

Restart the Raspberry Pi:

```
sudo reboot
```

After booting, the CODESYS Control Runtime is started automatically.

In order to be able to use CODESYS with CAN support, the script "startPLCCAN.sh" must be called additionally after each start:

```
cd ~
sudo ./startPLCCAN.sh
```

By calling the script, as mentioned above, first the CODESYS runtime is ended, the GPIOs are configured accordingly, the mcp251x kernel module is loaded, the can0 interface is configured, and then CODESYS Control is started again.



PiXtend V2 Software Manual

Of course, if you want to have the CAN support enabled permanently, you can automatically run the startup script every time you restart, edit the file

```
sudo nano /etc/rc.local
```

and add the following lines before "exit 0":

```
# Start CODESYS Control with PiXtend CAN Support  
sudo /home/pi/startPLCCAN.sh
```

Test the configuration by restarting the Raspberry Pi:

```
sudo reboot
```

With a call of "lsmod" you can check if the "mcp251x" module was loaded after the restart and with a call of "top" if the CODESYS Control is running.



7.8.5.CODESYS project with CANopen

If the preparations and tests under Linux were successful, a CANopen test project can now be created in CODESYS. We use two *PiXtend V2 -L-* devices and let them communicate with each other via CANopen.

We create an empty project to which two *PiXtend V2 -L-* devices are added. One device is the CANopen master and the other is the CANopen slave. Instead of using a second *PiXtend V2 -L-*, you can also use any other CAN bus device as a slave. For configuration, however consult the manual of your CAN slave.

In our example, for the sake of simplicity, we initially only use 1 byte, which is transferred from the master to the slave, but the example can then be expanded as required.

First create an empty project (File -> New Project -> Empty Project) and name it "PiXtendCAN_MasterSlave". Also, specify the location where your project should be saved if you do not agree with the preset.

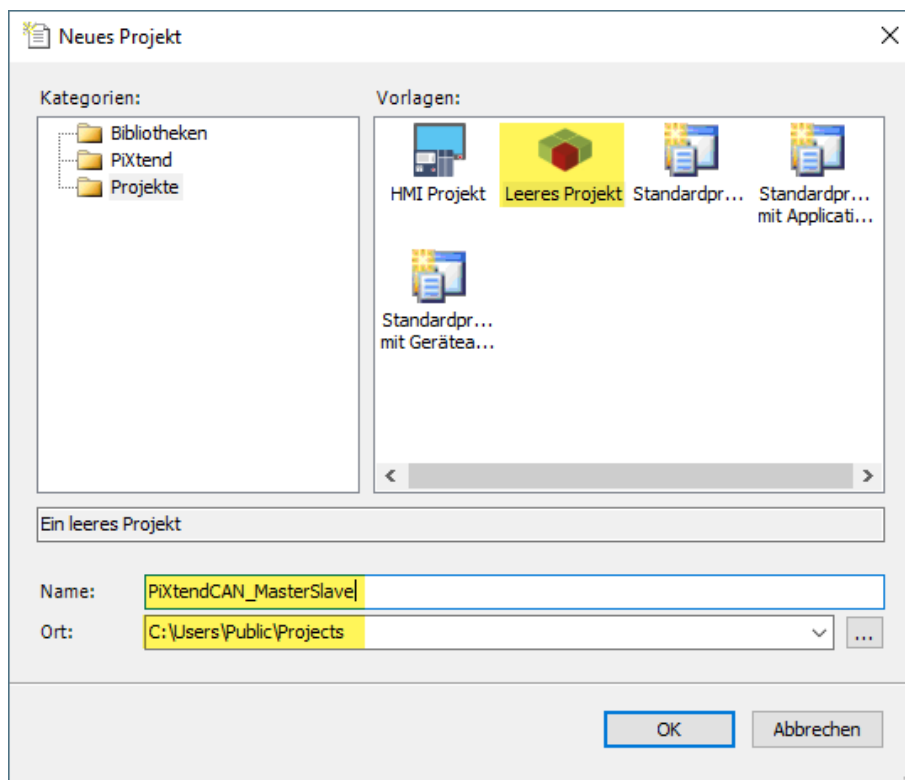


Figure 57: CODESYS - CAN-Bus - Create new project



7.8.5.1 PiXtend V2 -L- as CAN-Slave

Now add a new device to the project by right-clicking the project in the project tree and selecting "Add Device".

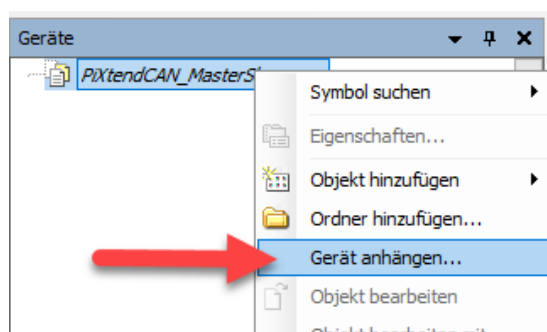


Figure 58: CODESYS - CAN-Bus Slave - Add PLC device

First change the manufacturer to "3S - Smart Software Solutions GmbH", then select the "CODESYS Control for Raspberry Pi" as the device, name it "PiXtendCAN_Slave".

When finished, click "Add Device" and then "Close".

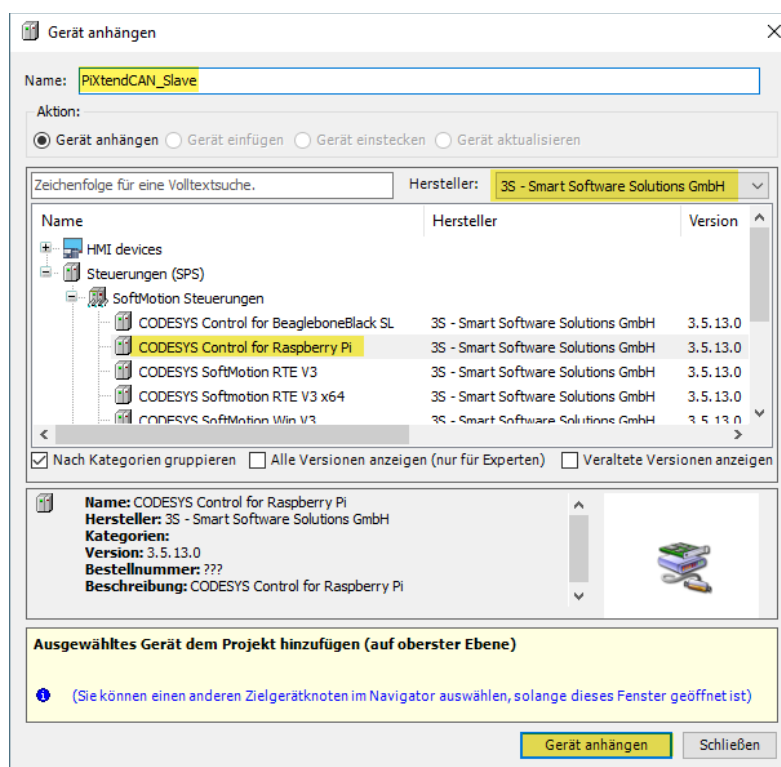


Figure 59: CODESYS - CAN-Bus Slave- Add Raspberry Pi



PiXtend V2 Software Manual

In the project tree, right-click on the "PiXtendCAN_Slave" device just added:

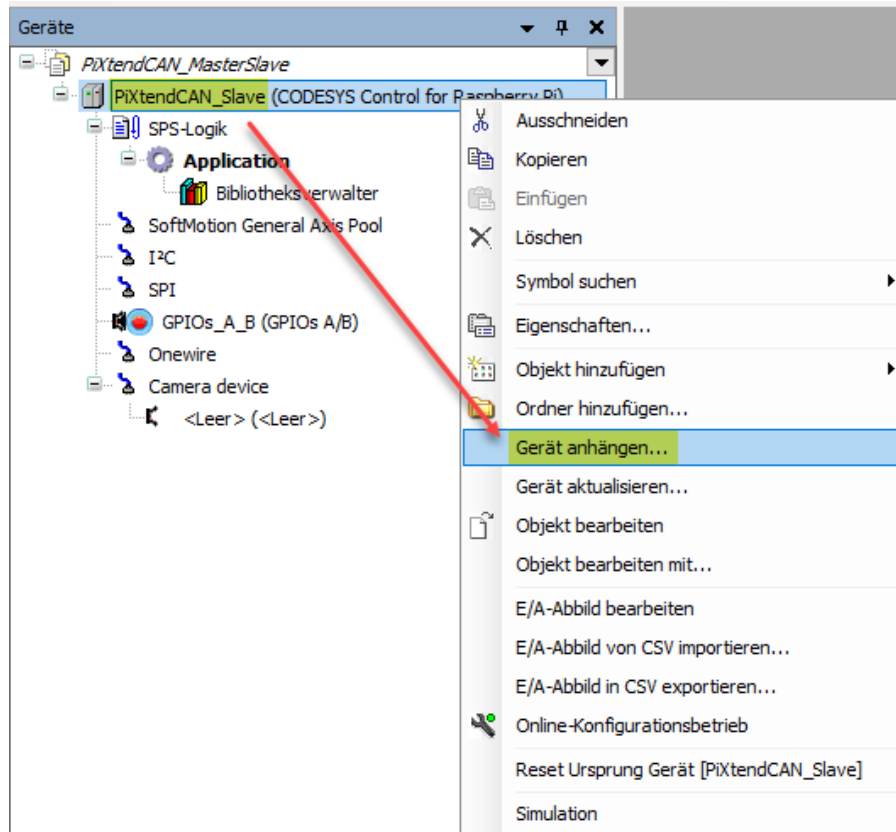


Figure 60: CODESYS - CAN-Bus Slave - Add device to PLC



PiXtend V2 Software Manual

In the dialog, select "CANbus" and add the device. The selection is faster if you select the company "3S - Smart Software Solutions GmbH" under "Vendor".

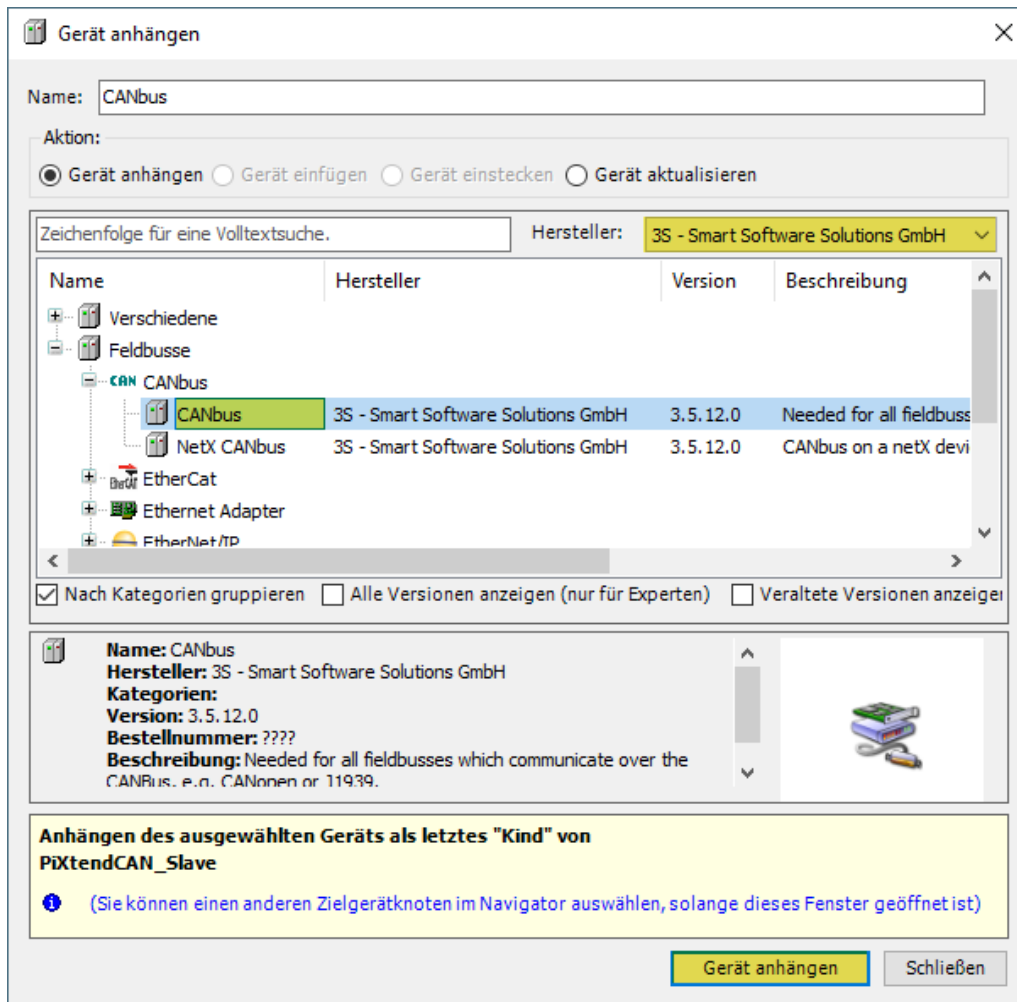


Figure 61: CODESYS - CAN-Bus Slave - Add CAN-Bus device

Leave the window open, we'll need it again on the next page.



PiXtend V2 Software Manual

Now left click on the just added "CANbus" device in the device tree, while the window "Add devices" is still open. After the click, the content of the "Add Devices" window changes and you can add a "CANopen Device" below the CANbus. The selection becomes easier if you select the company "3S - Smart Software Solutions GmbH" under "Vendor".

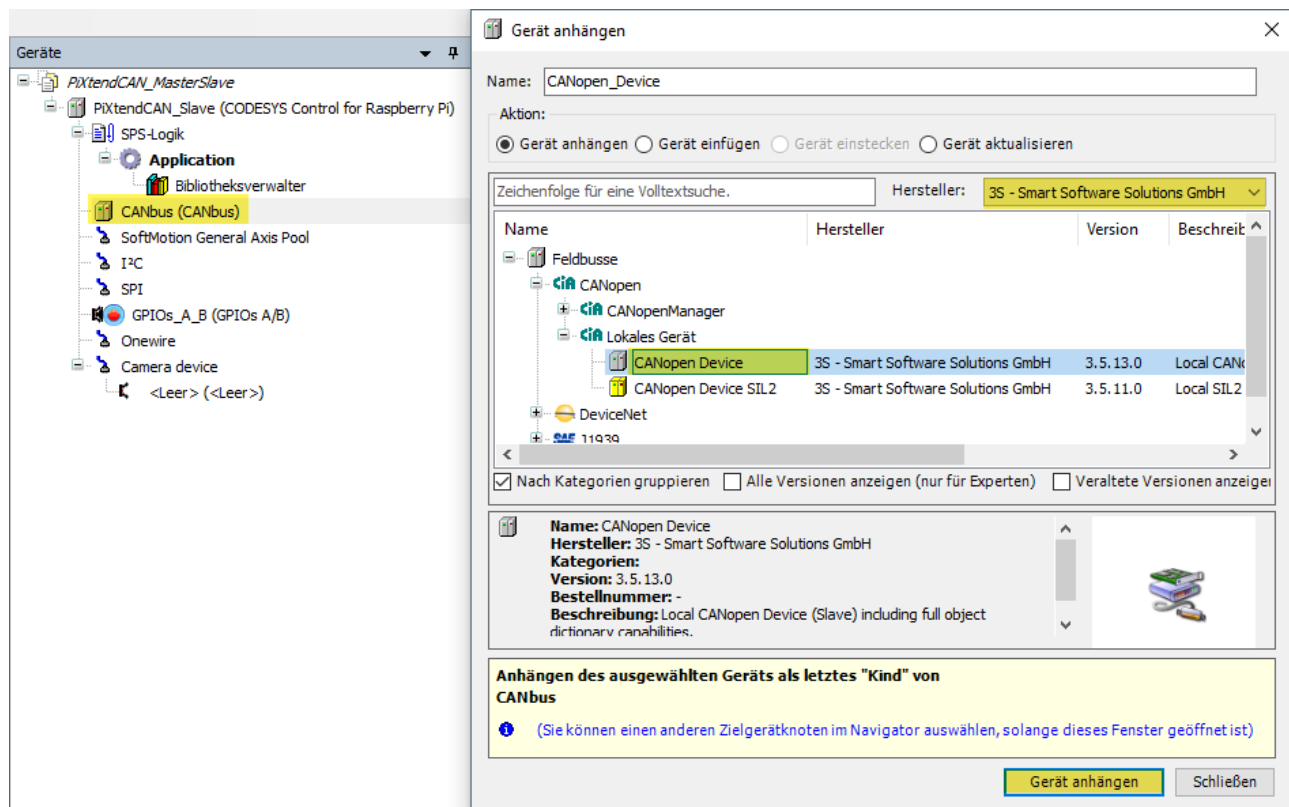


Figure 62: CODESYS - CAN-Bus Slave - Add CANopen device

After the addition you can close the "Add Devices" window.



PiXtend V2 Software Manual

Now double-click on the added CANopen device in the device tree and then click on the "Edit I/O area" button.

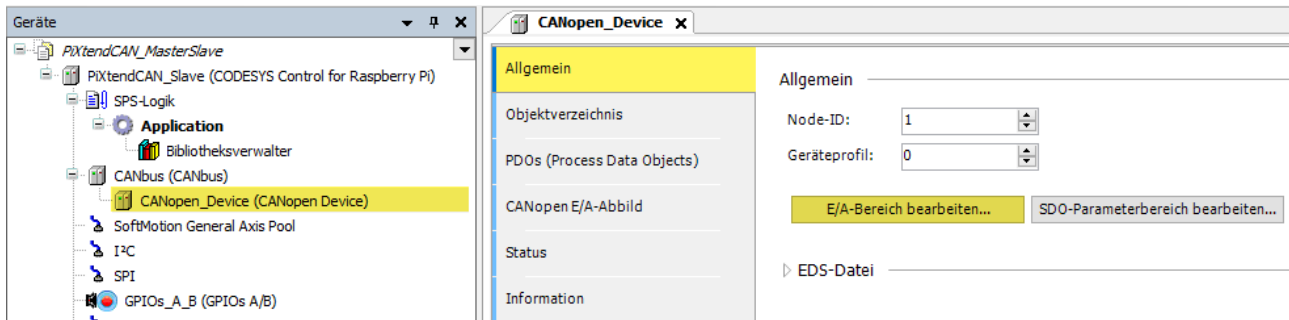


Figure 63: CODESYS - CAN-Bus Slave - Configure I/O area

Now add a new section and select "Receive". The remaining fields can remain unchanged since the master sends only a single USINT (BYTE) to the slave.

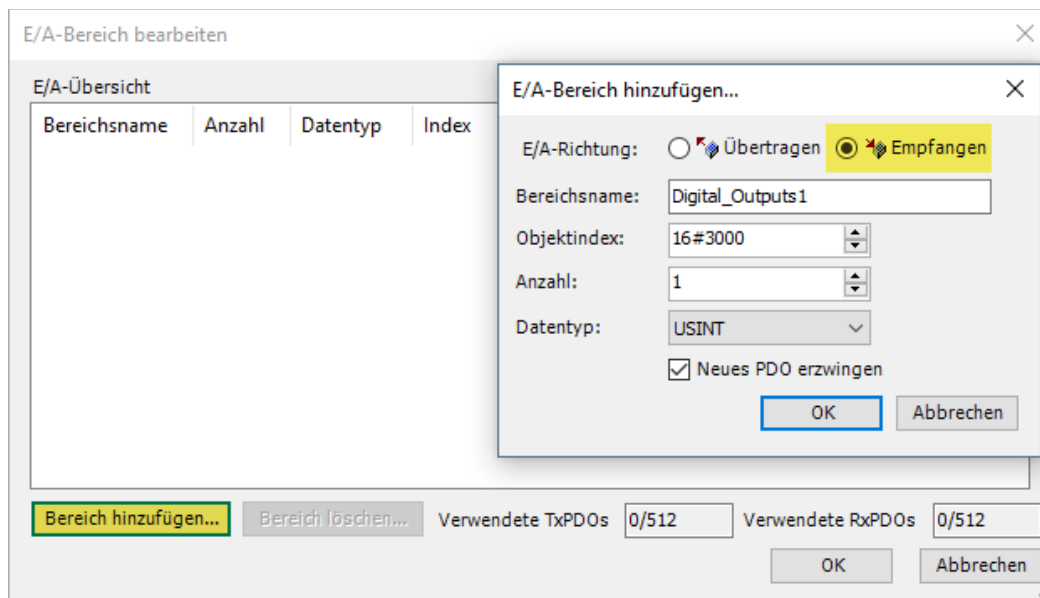


Figure 64: CODESYS - CAN-Bus Slave - Set PDO

Confirm your entries and close the dialog.



PiXtend V2 Software Manual

Now expand the "EDS-File" area and install the device just created in the device repository using the corresponding button. To be able to find your own device better later on, change Vendor name and / or the product name if necessary.

The screenshot shows the 'CANopen_Device' configuration window. The left sidebar has a tree view with 'Allgemein' selected. The main panel has two sections: 'Allgemein' and 'EDS-Datei'. In the 'Allgemein' section, 'Node-ID' is set to 1 and 'Geräteprofil' is set to 0. Below these are two buttons: 'E/A-Bereich bearbeiten...' and 'SDO-Parameterbereich bearbeiten...'. The 'EDS-Datei' section contains five input fields: 'Herstellername' (Qube Solutions GmbH), 'Herstellernummer' (801), 'Produktname' (MyCANopenDevice), 'Produktnummer' (1), and 'Revisionsnummer' (1). At the bottom of this section are two buttons: 'Ins Geräterepository installieren' and 'EDS-Datei exportieren...'.

Figure 65: CODESYS - CAN-Bus Slave - Install own CAN device

If you receive an indication that there is already a device with the same name or vendor number, then you can either go back and change the vendor number, or if you are sure that the other device is just a test device, you can overwrite it.

All information about our self-created CANopen device is now stored in the CODESYS device database and ready for later use, e.g. through the CANopen master.



PiXtend V2 Software Manual

Now add a Task Configuration to the application and a POU named "POU_Main".

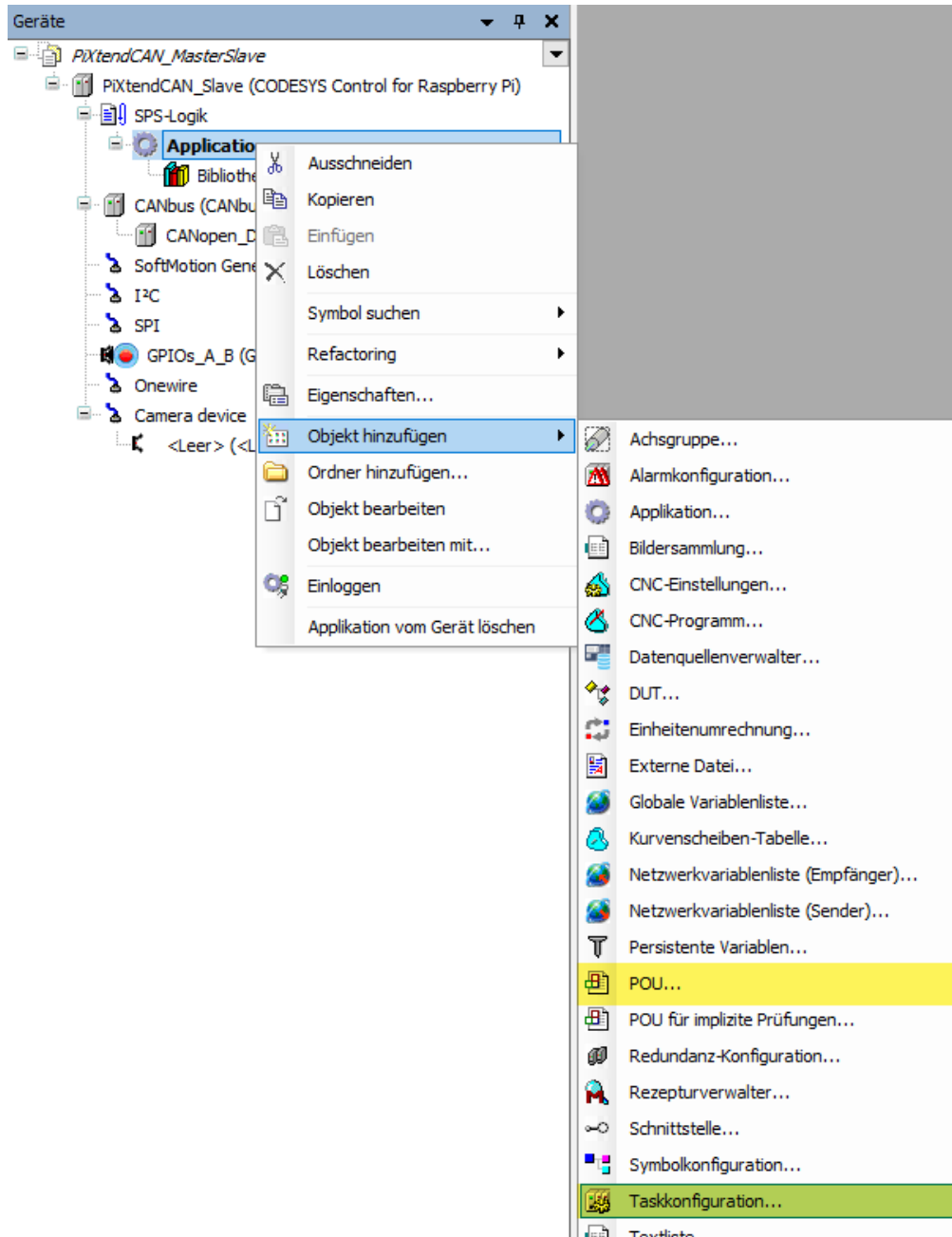


Figure 66: CODESYS - CAN-Bus Slave - POU and Task Configuration



PiXtend V2 Software Manual

Open the Task Configuration and add a call to the main program "POU_Main":

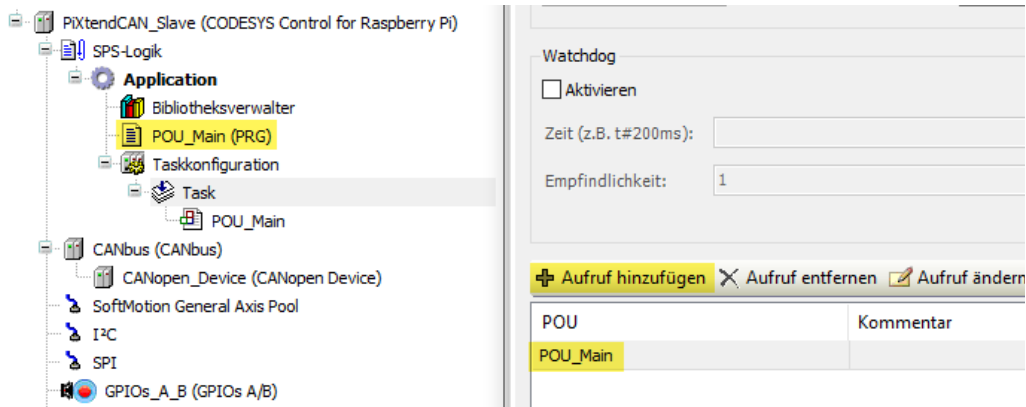


Figure 67: CODESYS - CAN-Bus Slave - Task Configuration

Switch back to the "CANopen_Device" and change there the settings for updating the variables under the tab "CANopen I/O Mapping":

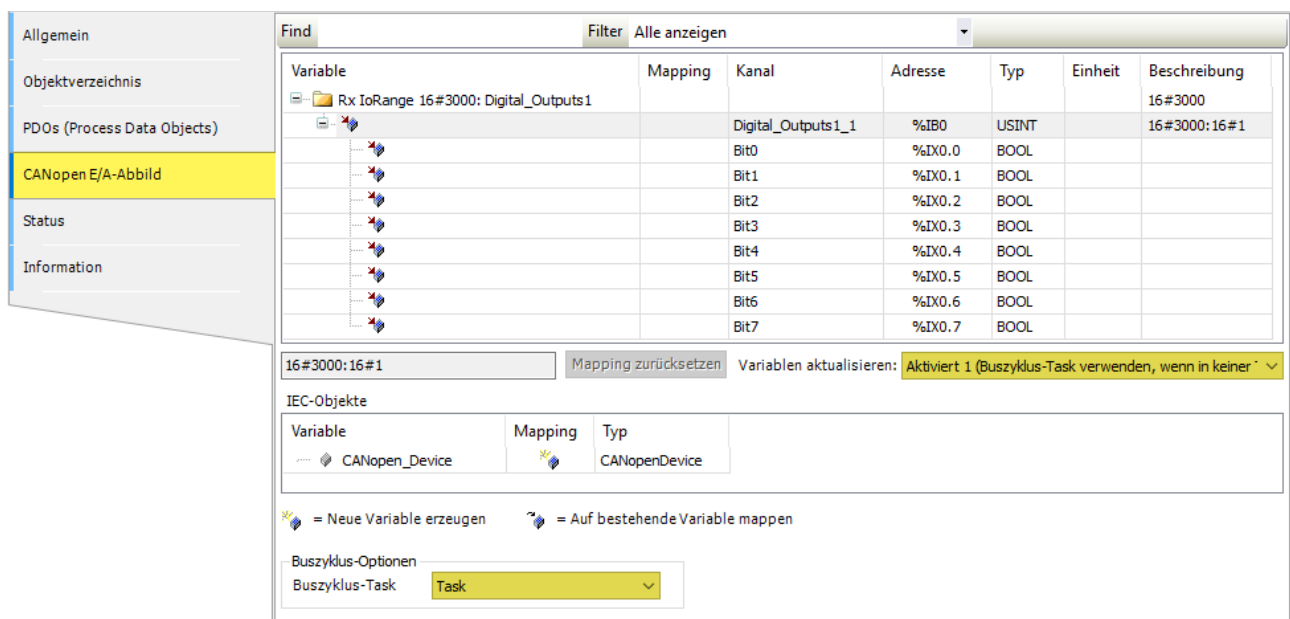


Figure 68: CODESYS - CAN-Bus Slave - CANopen I/O Mapping

Select the option "Enabled 1 - Use bus cycle task if not used in any task" and "Task" as bus cycle task.



PiXtend V2 Software Manual

Finally, the baud rate for the CAN bus should be reduced to "125 kBit/s":

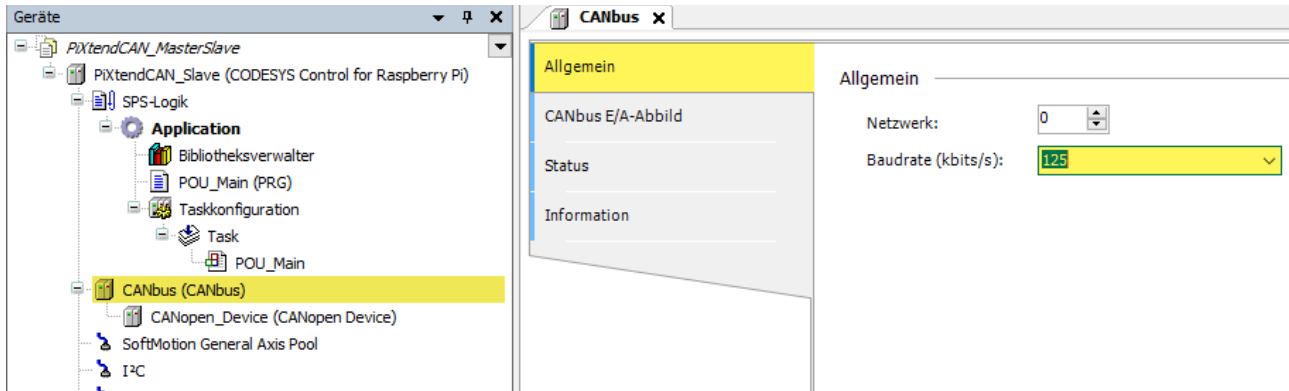


Figure 69: CODESYS - CAN-Bus Slave - Baud rate

This completes the configuration of the slave.



PiXtend V2 Software Manual

7.8.5.2 PiXtend V2 -L- as CANopen-Master

Now we add another *PiXtend V2 -L-* device to the project, which takes over the role of the CANopen master.

To do this, right-click on the project "PiXtendCAN_MasterSlave" in the project tree and add another "CODESYS Control for Raspberry Pi" device named "PiXtendCAN_Master":

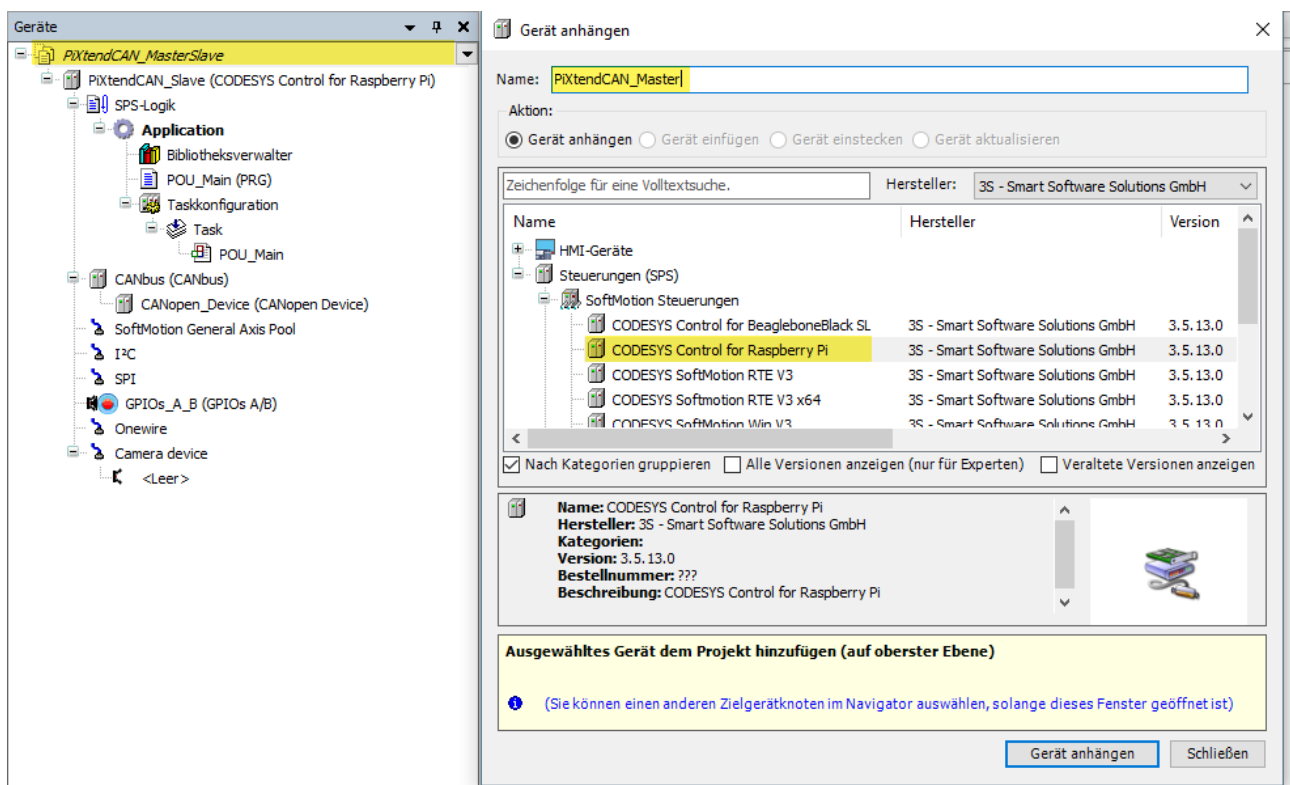


Figure 70: CODESYS - CAN-Bus Master - Add Raspberry Pi



PiXtend V2 Software Manual

Add to the "PiXtendCAN_Master" also a "CANbus" device:

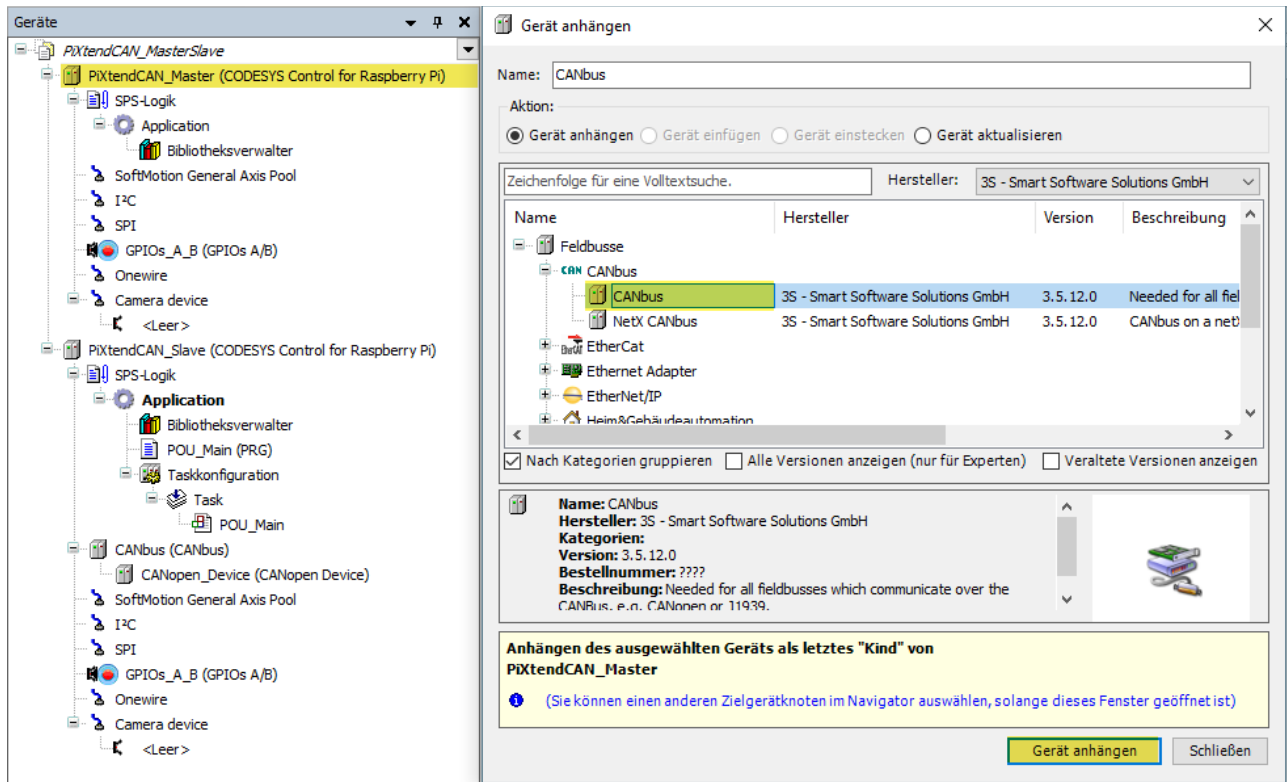


Figure 71: CODESYS - CAN-Bus Master - Add CANbus

Reduce or set the baud rate of the CANbus to "125 kBit/s" or to the same setting that was selected for the slave.



PiXtend V2 Software Manual

Right-click on the just added "CANbus" entry and add a "CANOpen_Manager" to it:

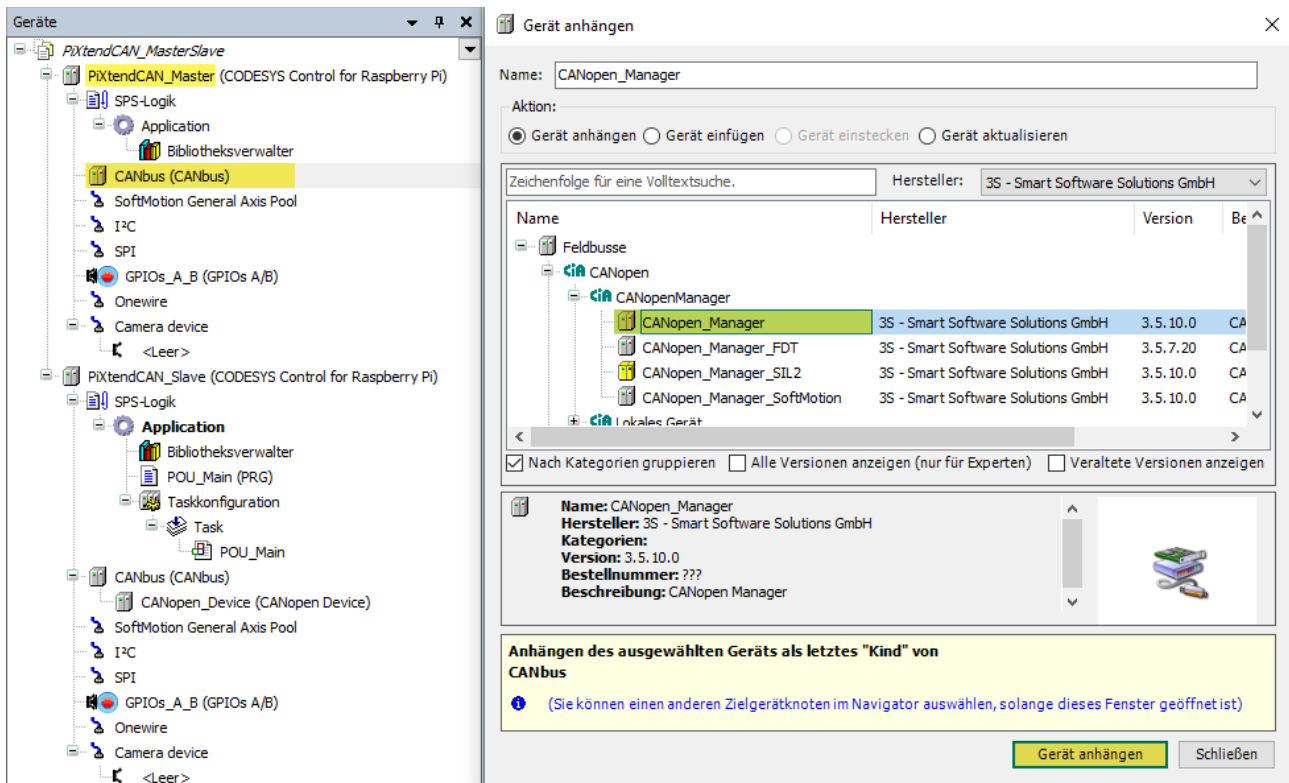


Figure 72: CODESYS - CAN-Bus Master - Add CANopen Manager

Leave the window open, we'll need it again on the next page.



PiXtend V2 Software Manual

Now select the just added "CANopen_Manager" and add the "MyCANopenDevice" to it which you created before or created yourself. If you have entered your own manufacturer name when creating the CANopen slave device, you can now select this name under "Vendor" and find your CANopen slave device even faster.

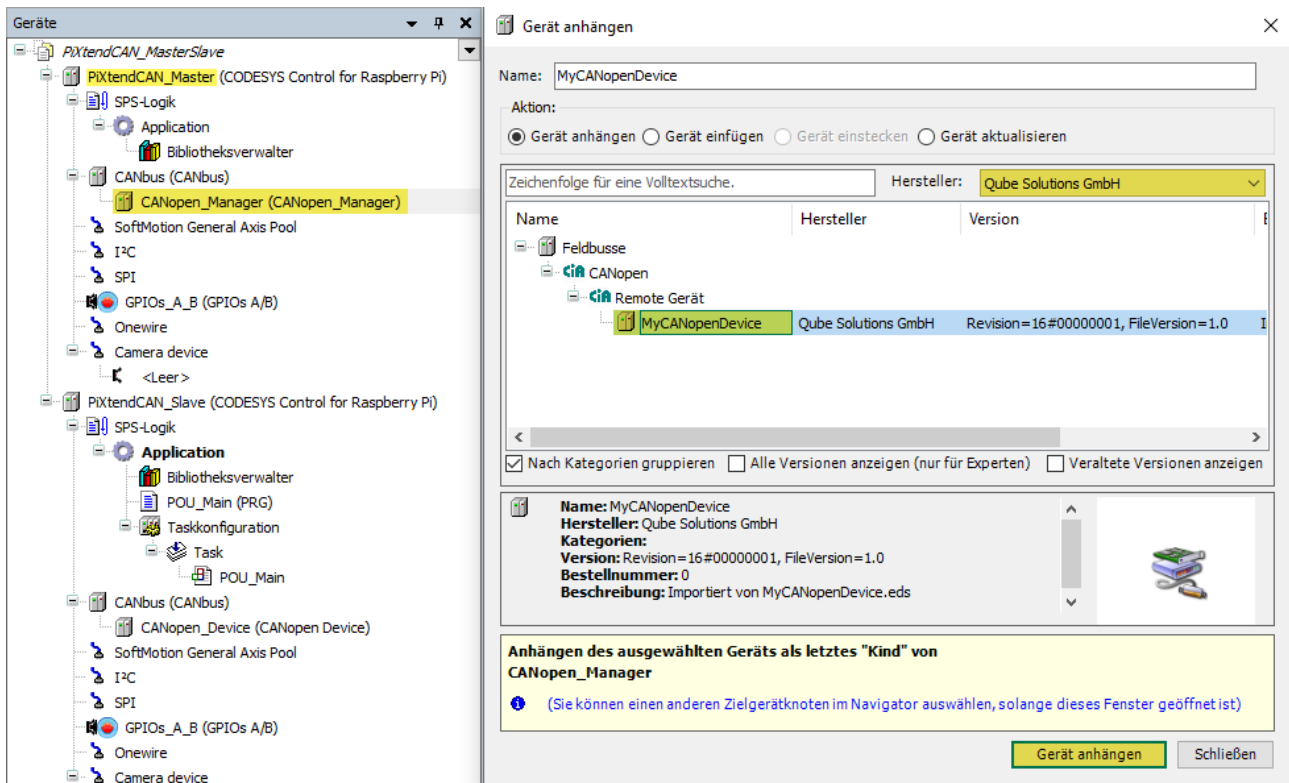


Figure 73: CODESYS - CAN-Bus Master - Add CAN-Bus slave

Add a "Task Configuration" and a "POU_Main" as the main program for the application to the "PiXtendCAN_Master" device and create a call for the user "POU_Main" for the entry "Task" in the "Task Configuration", similar to the one for the creation of the CANopen slave device.



PiXtend V2 Software Manual

Now open the I / O configuration (CANopen I / O image) of the device "MyCANopenDevice" and select the entry "Activated 1 - Use bus cycle task if not used in any task"

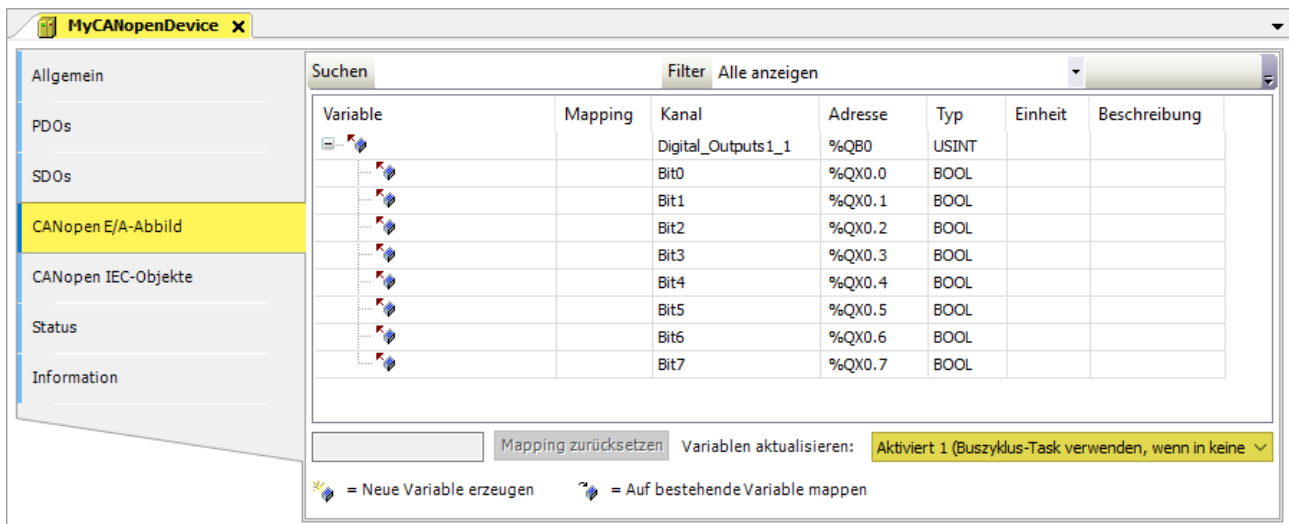


Figure 74: CODESYS - CAN-Bus Master - Configure CAN slave

7.8.5.3 Program Download and Test

Open the communication settings for the two devices "PiXtendCAN_Master" and "PiXtendCAN_Slave" one after the other and select the corresponding Raspberry Pi devices to be downloaded.

You can then use the function "Multiple Download" in the main menu "Online" to simultaneously load the applications onto the corresponding controllers.

Note: As soon as several applications are in the project tree, you can right-click on the application -> "Set Active Application" to select or activate the desired application.

Go "Online" to the applications one after the other and start them.

If everything has been configured correctly and both devices are connected correctly, the CAN entries in the project tree are displayed in green.



PiXtend V2 Software Manual

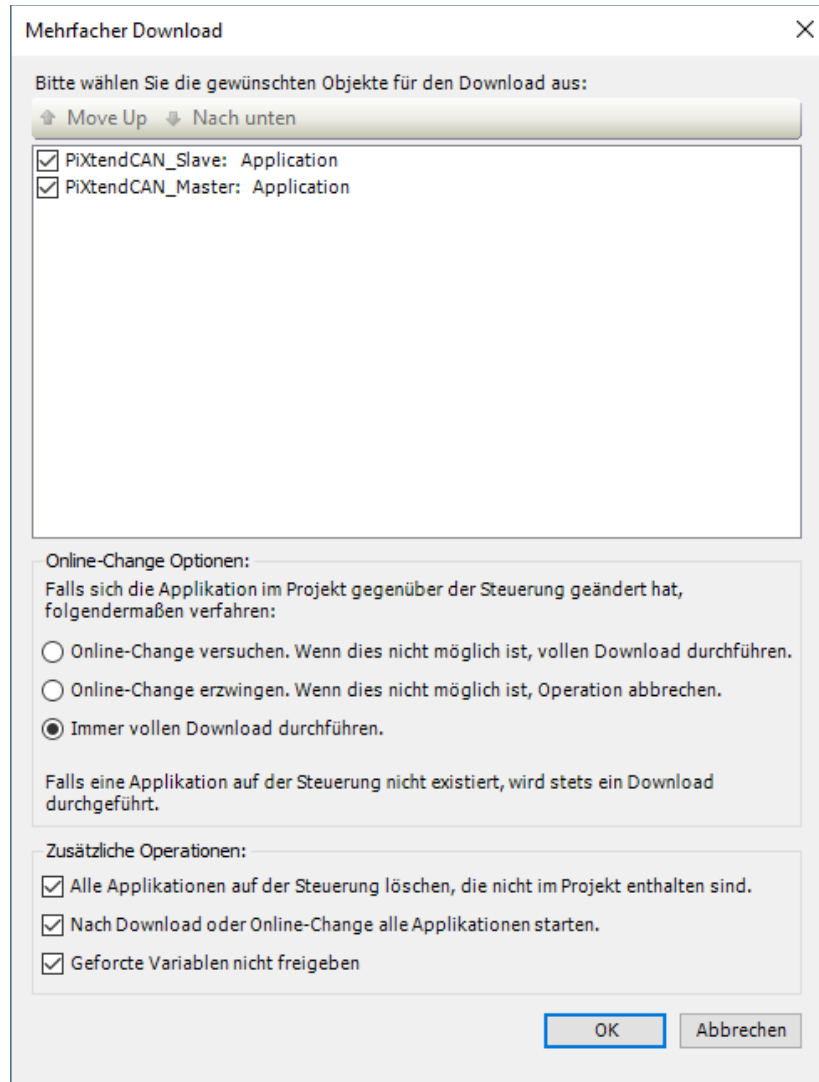


Figure 75: CODESYS - CAN-Bus Test - Multiple Download



PiXtend V2 Software Manual

If you now open the I/O area of the "CANopen Master" and change and write the values (Ctrl + F7), the values are automatically transferred from the master to the slave:

The screenshot shows two windows from the CODESYS software. The top window, titled 'MyCANopenDevice', displays a table of digital outputs. The bottom window, titled 'CANopen_Device', displays a similar table for the slave device. A red box highlights the 'Aktueller Wert' (Actual Value) column in the Master's table, and a red arrow points to the corresponding 'Aktueller Wert' column in the Slave's table, indicating the transfer of data.

Variable	Mapping	Kanal	Adresse	Typ	Aktueller Wert	Vorbereiteter Wert
Digital_Outputs1_1		Digital_Outputs1_1	%QB0	USINT	15	
Bit0		Bit0	%QX0.0	BOOL	TRUE	
Bit1		Bit1	%QX0.1	BOOL	TRUE	
Bit2		Bit2	%QX0.2	BOOL	TRUE	
Bit3		Bit3	%QX0.3	BOOL	TRUE	
Bit4		Bit4	%QX0.4	BOOL	FALSE	
Bit5		Bit5	%QX0.5	BOOL	FALSE	
Bit6		Bit6	%QX0.6	BOOL	FALSE	
Bit7		Bit7	%QX0.7	BOOL	FALSE	

Variable	Mapping	Kanal	Adresse	Typ	Aktueller Wert	Vorbereiteter Wert
Digital_Outputs1_1		Digital_Outputs1_1	%IB0	USINT	15	
Bit0		Bit0	%IX0.0	BOOL	TRUE	
Bit1		Bit1	%IX0.1	BOOL	TRUE	
Bit2		Bit2	%IX0.2	BOOL	TRUE	
Bit3		Bit3	%IX0.3	BOOL	TRUE	
Bit4		Bit4	%IX0.4	BOOL	FALSE	
Bit5		Bit5	%IX0.5	BOOL	FALSE	
Bit6		Bit6	%IX0.6	BOOL	FALSE	
Bit7		Bit7	%IX0.7	BOOL	FALSE	

Figure 76: CODESYS - CAN-Bus Test - Master and Slave exchange data

Of course, the variable could now be mapped and e.g. can be used directly in the main program. But this is all for now.



7.9. CODESYS – PiXtend Retain Memory

The *PiXtend V2 -S-* has a 32-byte flash memory (*PiXtend V2 -L-* has 64 bytes), which a user of CODESYS can use to store arbitrary values. This chapter is intended to illustrate the use of the Retain memory and to show which points are to be considered. We use a *PiXtend V2 -S-* here, but you can also use a *PiXtend V2 -L-*

7.9.1. Preparation

In preparation for this example, a new blank COEDSYS Standard Project is needed and the SPI bus must be configured with an SPI master and a *PiXtend V2 -S-*.

Furthermore, in the Raspberry Pi GPIOs, pin 24 should be defined as an output and should be given the variable name "*SPI_ENABLE*" in the I/O mapping.

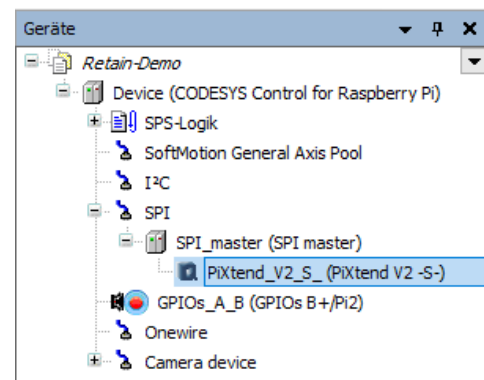


Figure 77: CODESYS - Retain-Demo - Device tree

Several variables are required in the I/O mapping of the *PiXtend V2 -S-*, which are listed below:

Folder	Channel	Variable	Description
Control	RetainDataEnable	xRetainDataEnable	Turn Retain function on / off
State	Firmware	byFirmware	Firmware version of the micro-controller
State	Hardware	byHardware	Hardware revision of the <i>PiXtend V2 -S-</i>
State	ModelIn	byModel	Model number of the <i>PiXtend V2 -S-</i>
State	RetainCRCError	xRetainCRCError	Retain CRC error bit
State	RetainVoltageError	xRetainVoltageError	Retain voltage is below 19 volts
State	Run	xRun	Micro-controller aktive signal
State	CRCHeaderInError	xCRCHeaderInError	CRC error in SPI header data found
State	CRCDataInError	xCRCDataInError	CRC error in SPI payload data found
Retain Data	RetainDataOut	arRetainDataOut	Retain data to the micro-controller
Retain Data	RetainDataIn	arRetainDataIn	Retain data from the micro-controller

The CODESYS project is now prepared, all necessary settings have been made. Now we can start programming!



7.9.2. Create program

In the program block "PLC_PRG" we need 4 new variables to control our test program. Create the following variables in the declaration section:

- xInit: BOOL
- xRetainInit: BOOL
- iRetainStep: INT
- xSetNewValue: BOOL

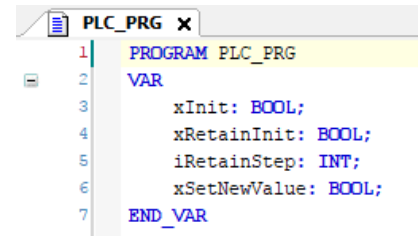


Figure 78: CODESYS - Retain - Variable declaration

In the program part we insert the following code:

```
IF xInit = FALSE THEN
    xInit := TRUE;
    SPI_ENABLE := TRUE;
END_IF

IF xRetainInit = FALSE THEN
    CASE iRetainStep OF
        0:
            IF xRun = TRUE AND byFirmware = 4 AND byHardware = 21 AND byModel = 83 AND
                xRetainCRCError = FALSE AND xRetainVoltageError = FALSE AND
                xCRCHeaderInError = FALSE AND xCRCDataInError = FALSE
            THEN
                iRetainStep := 1;
            END_IF
        1:
            arRetainDataOut[0] := arRetainDataIn[0];
            iRetainStep := 2;
        2:
            xRetainDataEnable := TRUE;
            iRetainStep := 3;
        3:
            xRetainInit := TRUE;
    END_CASE
END_IF

IF xSetNewValue THEN
    xSetNewValue := FALSE;
    arRetainDataOut[0] := arRetainDataOut[0] + 1;
END_IF
```

The numbers (4, 21 and 83) for the comparisons of byFirmware, byHardware and byModel may have to be replaced by the actual values of the used PiXtend V2 -S-. These numbers are just sample values.



PiXtend V2 Software Manual

```
1  IF xInit = FALSE THEN
2      xInit := TRUE;
3      SPI_ENABLE := TRUE;
4  END_IF
5
6  //Retain init sequence
7  IF xRetainInit = FALSE THEN
8      CASE iRetainStep OF
9          0: //Startup check - We have the Run bit, now lets check
10             //the PiXtend state
11             IF xRun = TRUE AND byFirmware = 4 AND byHardware = 21 AND byModel = 83 AND
12                 xRetainCRCError = FALSE AND xRetainVoltageError = FALSE AND
13                 xCRCHeaderInError = FALSE AND xCRCDataInError = FALSE
14             THEN
15                 //All OK - go to next step
16                 iRetainStep := 1;
17             END_IF
18
19             1: //Start - Get retain data from the micro-controller
20                 arRetainDataOut[0] := arRetainDataIn[0];
21                 //Go to next step - Enable retain data storage
22                 iRetainStep := 2;
23             2: //Activate retain data function
24                 xRetainDataEnable := TRUE;
25                 iRetainStep := 3;
26             3: //Done - Retain data setup & restore complete
27                 xRetainInit := TRUE;
28             END_CASE
29         END_IF
30
31     IF xSetNewValue THEN
32         xSetNewValue := FALSE;
33         //Increment the retain output byte 0 by 1
34         arRetainDataOut[0] := arRetainDataOut[0] + 1;
35     END_IF
```

Figure 79: CODESYS - Retain - Example program

Explanation of the program sequence:

In the first section with the line "*If xInit = False then*" a very short initialization is carried out, only the SPI communication with "*SPI_ENABLE := True*" is switched on.

In the next section, existing retain data will be read into CODESYS via a sequencer, provided that the checking of various variables, see Preparation, succeeds. In step 1, the first byte of the RetainDataIn array variable is written to the first byte of the RetainDataOut array variable, and in step 3, the retain function of the *PiXtend V2 -S-* is activated.

It is recommended to always enable the retain function as the last step after the "old" (previous) data has been restored.

We can use the last section to increment the first byte of the RetainDataOut array variable by one increment by setting the variable *xSetNewValue* to true. Then we can unplug the power from the *PiXtend V2 -S-* and everything turns off. After a restart, we should find in the RetainDataOut array variable the same value that was before the power cycle.



Example run:

1. After starting the program `arRetainDataOut[0]` and `arRetainDataIn[0]` have the same value, here in this example it is number 13.

```
1: //Start - Get retain data from the micro-controller
   arRetainDataOut[0][13] := arRetainDataIn[0][13];
   //Go to next step - Enable retain data storage
   iRetainStep[3] := 2;
2: //Activate retain data function
   xRetainDataEnable[TRUE] := TRUE;
   iRetainStep[3] := 3;
```

Figure 80: CODESYS - Retain - Start value 13

2. After setting the variable `xSetNewValue` to `true`, the value in `arRetainDataOut[0]` increases by one to 14. The variable `arRetainDataIn[0]` still shows us the "old" value.

```
1: //Start - Get retain data from the micro-controller
   arRetainDataOut[0][14] := arRetainDataIn[0][13];
   //Go to next step - Enable retain data storage
   iRetainStep[3] := 2;
2: //Activate retain data function
   xRetainDataEnable[TRUE] := TRUE;
   iRetainStep[3] := 3;
```

Figure 81: CODESYS - Retain - `arRetainDataOut[0]` increased by 1 to 14

3. We disconnect the supply of the *PiXtend V2 -S-*, wait briefly and restore the power (power cycle).

4. After the restart, the old value was restored from the flash memory of the micro-controller and immediately written to the first byte of the `RetainDataOut` array variable. From now on we can easily continue working with this value.

```
1: //Start - Get retain data from the micro-controller
   arRetainDataOut[0][14] := arRetainDataIn[0][14];
   //Go to next step - Enable retain data storage
   iRetainStep[3] := 2;
2: //Activate retain data function
   xRetainDataEnable[TRUE] := TRUE;
   iRetainStep[3] := 3;
```

Figure 82: CODESYS - Retain - After Power-Cycle

This concludes the example for the Retain memory, on the next page you will find more information on advanced programming techniques, for example, how to increase the data security of retain data.



7.9.3. Further information

7.9.3.1 Increase data security

The retain data is stored in the micro-controller flash memory with a CRC and read and checked at startup. It may be useful to perform such a check also in CODESYS to ensure the correctness of the persistent data.

For this purpose, a checksum (CRC), e.g. with 16 bits, can be created. In CODESYS there are such functions in the libraries *CAA Memory* and *CmpChecksum*. But also in the CODESYS open source library, short [OSCAT](#), such functions can be found.

A 16-bit checksum can be split into two bytes and written to the last two bytes of the retain memory. After rebooting, CODESYS allows you to check the restored values based on this checksum.

7.9.3.2 Working with structures

The outbound and inbound Retain data is provided as an array of 32 bytes. Each byte in this array can be addressed individually, allowing for easy looping and direct access. However, if you want to store 16-bit, 32-bit or even 64-bit values in the Retain memory, this can be a bit tedious since you have to assign each byte individually.

To simplify such situations, CODESYS offers two practical approaches. With Structs, data can be organized and summarized in CODESYS. If you combine a structure with a *Union* data type, advanced users may already know this DUT, this is also referred to as an overlay and gives you the option of addressing the Retain data via a structure. The big advantage of this approach is that you no longer have to worry about organization of the individual bytes within the array.

Example:

A DWORD variable is 4 bytes in size, if this variable is combined in a Union data type with an array of 4 bytes, each value written to the DWORD variable is automatically split into the 4 bytes which appear in the array. This is possible because the DWORD variable and the 4-byte array occupy the same memory space in CODESYS (overlay).

```
TYPE uDword :  
  UNION  
    dwValue : DWORD;  
    arValue : ARRAY[0..3] OF BYTE;  
  END_UNION  
END_TYPE
```



```
uValue.dwValue = 2000000;  
uValue.arValue[0] = 128;  
uValue.arValue[1] = 132;  
uValue.arValue[2] = 30;  
uValue.arValue[3] = 0;
```



7.10. CODESYS – Shutting down the Raspberry Pi

CODESYS itself does not have its own function to shut down the Raspberry Pi ("graceful shutdown") in order to switch it off or disconnect it from the power supply. Also see chapter 6.7 Turning off the Raspberry Pi for more information.

Shutting down the Raspberry Pi is not only advantageous in terms of data security, in addition, you can also use the CODESYS own retain function in this way, because only when shutting down the system the values of the CODESYS retain function are actually written to the SD card and read in at the next start. This is can be interessting, e.g. if you need more than 32 or 64 bytes of retain memory.

The following short example shows how to shut down the Raspberry Pi from CODESYS.

Add 2 new libraries to your project.

These are the libraries "SysProcess" and "SysTypes2 Interfaces". When adding, if necessary, use the search bar in the library manager and enter the names of the libraries directly to find them faster. Also remember to click on the plus symbol when adding, this will make the system libraries visible.

In the "SysProcess" library is the function "SysProcessExecuteCommand" which we need for doing a shutdown.

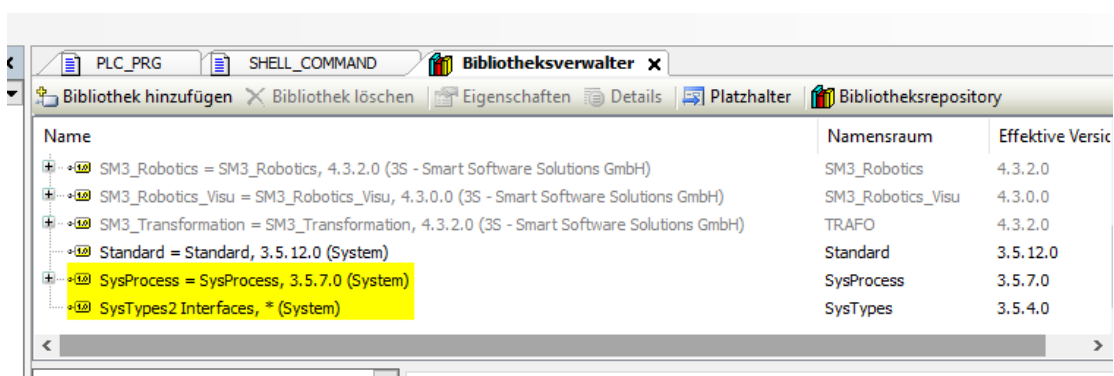


Figure 83: CODESYS - Shutdown - Library Manager



PiXtend V2 Software Manual

When adding libraries, the search bar (2) at the top of the dialog can be very helpful, also remember to enable the advanced view (1).

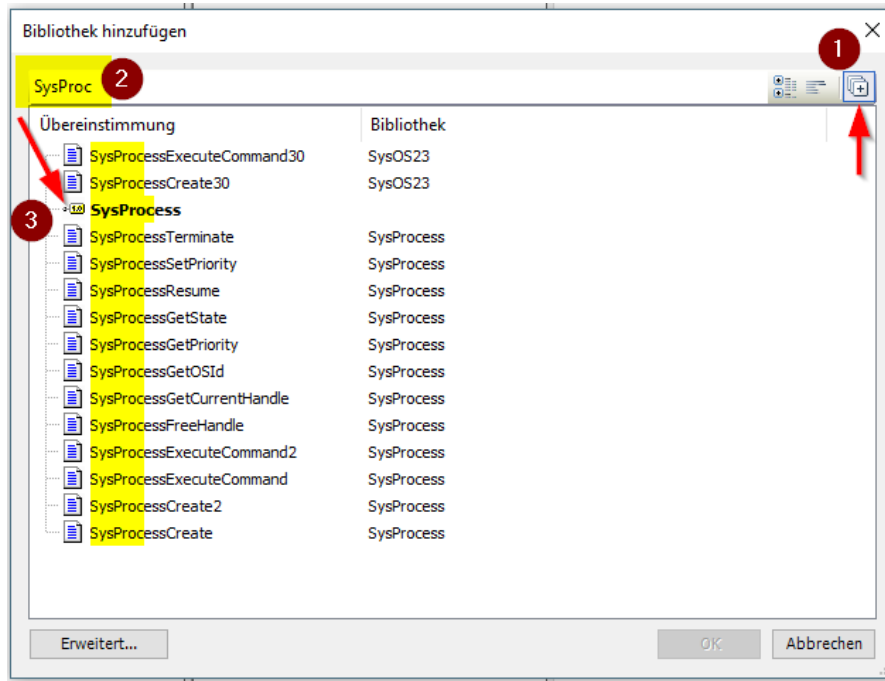


Figure 84: CODESYS - Shutdown - Add library

If both libraries have been added to the project, it is best to create a new POU and in addition 4 new variables.

You need 4 variables with the following types:

- 1 x BOOL
- 1 x DINT
- 1 x SysTypes.RTS_IEC_RESULT
- 1 x STRING

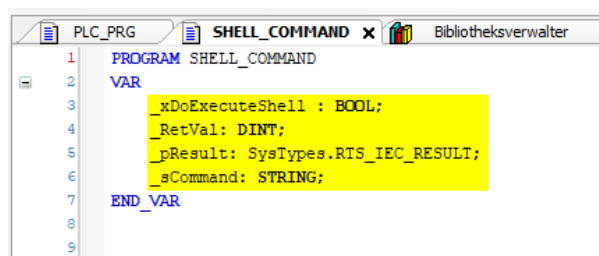


Figure 85: CODESYS - Shutdown - Variables



PiXtend V2 Software Manual

With an IF statement in the new program block you can encapsulate the execution of the "SysProcessExecuteCommand" function and it is important that we execute the shutdown command only once. If the execution is repeated, CODESYS can run into problems.

The command we have to give the operating system is:

```
sudo shutdown -h now & disown
```

The "&" character and the word "disown" are not printing errors, they are part of the command and must be present.

If the Boolean variable is TRUE, then we go into the IF block, reset the Bool immediately, then put our desired command in the String variable and then pass everything to the "SysProcessExecuteCommand" function. We have to pass the result of the function as a pointer with the operator ADR.

This way, the Raspberry Pi should simply shut down, you have to wait a short time after calling the command, similar to Windows, until all services and CODESYS itself have been terminated and the operating system has gone into a save state, then the supply can be turned off safely.

```
1
2 IF _xDoExecuteShell THEN
3   _xDoExecuteShell := FALSE;
4
5   //Kommando zum Herunterfahren setzen
6   _sCommand := 'sudo shutdown -h now & disown';
7
8   //Kommando dem Betriebssystem übergeben und ausführen
9   _RetVal := SysProcess.SysProcessExecuteCommand(pszCommand:= _sCommand, pResult:= ADR(_pResult));
10
11 END_IF
12
13
```

Figure 86: CODESYS - Shutdown - Example program



7.11. PiXtend V2 -S- SPI Device

7.11.1. SPI Device Parameter

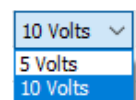
The *PiXtend V2 -S- SPI* device in CODESYS has some settings which can be changed in *SPI devices Parameter* tab. These parameters cannot be changed later on during runtime, only during desing time.

Parameter	Typ	Wert	Standardwert	Einheit
5 Volts/10 Volts Jumper Setting				
JumperSettingAI0	Enumeration of BOOL	10 Volts	10 Volts	Volts [V]
JumperSettingAI1	Enumeration of BOOL	10 Volts	10 Volts	Volts [V]
GPIO Configuration				
GPIO0Ctrl	Enumeration of BYTE	Input	Input	
GPIO1Ctrl	Enumeration of BYTE	Input	Input	
GPIO2Ctrl	Enumeration of BYTE	Input	Input	
GPIO3Ctrl	Enumeration of BYTE	Input	Input	
GPiOPullupsEnable	Enumeration of BOOL	Off	Off	
Microcontroller Settings				
WatchdogEnable	Enumeration of BYTE	Disabled	Disabled	
StateLEDDisable	Enumeration of BOOL	False	False	

Figure 87: CODESYS PiXtend V2 -S- SPI devices Parameter tab

7.11.1.15 Volt/10 Volt Jumper Setting

This parameter is used to tell the CODESYS driver for the *PiXtend* whether the analog signals at AnalogIn0 and AnalogIn1 are in the range of 5 volts (jumper set) or 10 volts (jumper not set, factory setting). Depending on this setting, the driver uses a different conversion factor to calculate the voltage values. The *PiXtend* itself can not tell if a jumper is set or not, the user must match the setting in CODESYS with the actual (physically) available jumpers on *PiXtend* manually and make an appropriate setting for this parameter.



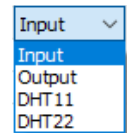


PiXtend V2 Software Manual

7.11.1.2 GPIO Configuration

The GPIO configuration allows the user to set each GPIO to one of four possible settings:

- Input – Default setting
- Output
- DHT11 - (Input for temperature and humidity measurement)
- DHT22 - (Input for temperature and humidity measurement)



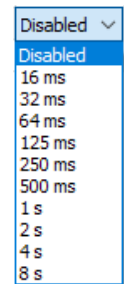
Furthermore, the GPIO configuration offers the possibility to activate the GPIO PullUps. If this setting is set to *On*, the GPIO PullUps can be activated by setting the respective GPIO output to *True* while a GPIO is configured as input.

7.11.1.3 Microcontroller Settings

The micro-controller of the *PiXtend V2 -S-* has two settings which the user can change.

1. *WatchdogEnable*

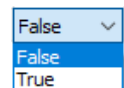
With this setting, the watchdog of the micro-controller can be switched on and off. The position *Disabled* means the watchdog is off (inactive), the selection of a waiting time activates the watchdog. There are times ranging from 16 milliseconds up to 8 seconds to choose from.



2. *StateLEDDisable*

The *PiXtend V2 -S-* has a status LED that signals an error if a problem is detected in the microcontroller. If necessary, this LED can be disabled.

The setting *False* means the LED is active (on), setting it to *True* deactivates the LED.





PiXtend V2 Software Manual

7.11.2. I/O Overview

In this chapter you will find a listing of all Ios available in CODESYS for the *PiXtend V2 -S-*, together with their data type and a short description.

Name	Type	Data type	Description
Control			
PWM0Ctrl1	Output	WORD	The effect of the PWM0Ctrl1 value depends on the selected Mode in PWM0Ctrl0. This value applies to channels A & B of PWM 0.
PWM1Ctrl1	Output	BYTE	The effect of the PWM1Ctrl1 value depends on the selected Mode in PWM1Ctrl0. This value applies to channels A & B of PWM 1.
PWM0Ctrl0	Output	BYTE	PWM 0 Control 0 - Allows to set the PWM mode, channel enable and prescaler. PWM 0 uses 16 Bit values.
PWM1Ctrl0	Output	BYTE	PWM 1 Control 0 - Allows to set the PWM mode, channel enable and prescaler. PWM 1 uses 8 Bit values.
GPIODebounce01	Output	BYTE	Debounces the GPIO Inputs 0 and 1, value * Bus-Cycle interval = debounce time.
GPIODebounce23	Output	BYTE	Debounces the GPIO Inputs 2 and 3, value * Bus-Cycle interval = debounce time.
DigitalInDebounce01	Output	BYTE	Debounces the Digital Inputs 0 and 1, value * Bus-Cycle interval = debounce time.
DigitalInDebounce23	Output	BYTE	Debounces the Digital Inputs 2 and 3, value * Bus-Cycle interval = debounce time.
DigitalInDebounce45	Output	BYTE	Debounces the Digital Inputs 4 and 5, value * Bus-Cycle interval = debounce time.
DigitalInDebounce67	Output	BYTE	Debounces the Digital Inputs 6 and 7, value * Bus-Cycle interval = debounce time.
SafeState	Output	BIT	Switch microcontroller to a safe state, in case the PLC needs to reboot or shutdown. True = Go to Safe State, False = Stay ON.
RetainDataEnable	Output	BIT	Activate the retain data storage function in the microcontroller. TRUE = On, FALSE = Off.
RetainCopy	Output	BIT	If TRUE then the current RetainDataOut is copied to RetainDataIn within the microcontroller. If FALSE the currently stored data in the microcontrollers memory is returned in RetainDataIn.
State			
Firmware	Input	BYTE	Firmware Version of the microcontroller on the PiXtend V2 -S- board.
Hardware	Input	BYTE	PiXtend V2 -S- board revision (hardware), 20 = 2.0, 21 = 2.1, etc.
ModelIn	Input	BYTE	Model number as reported by the PiXtend V2 -S- board.
Error	Input	BYTE	Error byte from the microcontroller.
RetainCRCErr	Input	BIT	The CRC of the retain area on the microcontroller is wrong, the data could be corrupted.



PiXtend V2 Software Manual

RetainVoltageError	Input	BIT	If TRUE then retain function cannot be used, the supply voltage is below 19 volts.
Run	Input	BIT	The communication with the microcontroller is running.
BusCycleError	Input	BIT	The cycle time (interval) of the Bus Cycle Task is too fast.
CRCHeaderInError	Input	BIT	A CRC error was detected in the SPI header, communication error.
CRCDataInError	Input	BIT	A CRC error in the PiXtend V2 -S- SPI data was detected, no usable data available.
ModelInError	Input	BIT	Model mismatch detected, the configured PiXtend V2 -S- model in CODESYS V3 and the actual hardware do not match.
Sensor0Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Sensor1Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Sensor2Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Sensor3Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Analog Inputs			
AnalogIn0	Input	REAL	Analog Input 0 is measured in Volts [V], range is 0 Volt to +10 Volts.
AnalogIn0Raw	Input	WORD	Analog Input 0 raw value, this is the actual 16 bit value from the ADC.
AnalogIn1	Input	REAL	Analog Input 1 is measured in Volts [V], range is 0 Volt to +10 Volts.
AnalogIn1Raw	Input	WORD	Analog Input 1 raw value, this is the actual 16 bit value from the ADC.
Digital Inputs			
DigitalInputs	Input	BYTE	Digital Inputs 0-7 as Byte, Bit access is also possible.
GPIOInputs	Input	BYTE	GPIO Inputs 0-3 as Byte, Bit access is also possible. GPIO inputs will only work if so configured in the Parameter section of the PiXtend device.

Digital Outputs			
------------------------	--	--	--



PiXtend V2 Software Manual

DigitalOutputs	Output	BYTE	Digital Outputs 0-3 as Byte, Bit access is also possible.
GPIOOutputs	Output	BYTE	PiXtend GPIO Outputs 0-3 as Byte, Bit access is also possible. GPIO outputs will only work if so configured in the Parameter section of the PiXtend device.
RelayOutputs	Output	BYTE	Relay outputs 0-3 a Byte, Bit access is also possible.
PWM Outputs			
PWM0A	Output	WORD	PWM 0, channel A output value, the actual PWM behaviour depends on the mode setting in PWM0Ctrl0.
PWM0B	Output	WORD	PWM 0, channel B output value, the actual PWM behaviour depends on the mode setting in PWM0Ctrl0.
PWM1A	Output	BYTE	PWM 1, channel A output value, the actual PWM behaviour depends on the mode setting in PWM1Ctrl0.
PWM1B	Output	BYTE	PWM 1, channel B output value, the actual PWM behaviour depends on the mode setting in PWM1Ctrl0.
Humidity Inputs			
Humidity measured by a sensor connected to GPIO 0-3, if activated in the SPI devices Parameter tab. The values are given in % [RH].			
Humid0	Input	REAL	
Humid1	Input	REAL	
Humid2	Input	REAL	
Humid3	Input	REAL	
Temperature Inputs			
Temperature of a sensor connected to GPIO 0-3, if activated in the SPI devices Parameter tab. The values are given in degrees Celcius (°C).			
Temp0	Input	REAL	
Temp1	Input	REAL	
Temp2	Input	REAL	
Temp3	Input	REAL	
Retain Data			
Retain data can be used to save information in the microcontrollers memory in case of a sudden powerloss. The Retain data is 32 bytes in size.			
RetainDataOut	Output	ARRAY[0..31] OF BYTE	
RetainDataIn	input	ARRAY[0..31] OF BYTE	

Table 2: PiXtend V2 -S- CODESYS I/Os overview



7.12. PiXtend V2 -L- SPI Device

7.12.1. SPI Device Parameter

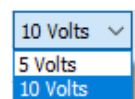
The *PiXtend V2 -L- SPI* device in CODESYS has some settings which can be changed in *SPI devices Parameter* tab. These parameters cannot be changed later on during runtime, only during desing time.

Parameter	Typ	Wert	Standar...	Einheit
5 Volts/10 Volts Jumper Setting				
JumperSettingAI0	Enumeration of BOOL	10 Volts	10 Volts	Volts [V]
JumperSettingAI1	Enumeration of BOOL	10 Volts	10 Volts	Volts [V]
JumperSettingAI2	Enumeration of BOOL	10 Volts	10 Volts	Volts [V]
JumperSettingAI3	Enumeration of BOOL	10 Volts	10 Volts	Volts [V]
GPIO Configuration				
GPIO0Ctrl	Enumeration of BYTE	Input	Input	
GPIO1Ctrl	Enumeration of BYTE	Input	Input	
GPIO2Ctrl	Enumeration of BYTE	Input	Input	
GPIO3Ctrl	Enumeration of BYTE	Input	Input	
GPIOPullupsEnable	Enumeration of BOOL	Off	Off	
Microcontroller Settings				
WatchdogEnable	Enumeration of BYTE	Disabled	Disabled	
StateLEDDisable	Enumeration of BOOL	False	False	

Figure 88: CODESYS PiXtend V2 -L- SPI devices Parameter Reiter

7.12.1.1 5 Volt/10 Volt Jumper Setting

This parameter is used to tell the CODESYS driver for the *PiXtend* whether the analog signals at AnalogIn0 and AnalogIn3 are in the range of 5 volts (jumper set) or 10 volts (jumper not set, factory setting). Depending on this setting, the driver uses a different conversion factor to calculate the voltage values. The *PiXtend* itself can not tell if a jumper is set or not, the user must match the setting in CODESYS with the actual (physically) available jumpers on *PiXtend* manually and make an appropriate setting for this parameter.



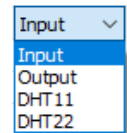


PiXtend V2 Software Manual

7.12.1.2 GPIO Configuration

The GPIO configuration allows the user to set each GPIO to one of four possible settings:

- Input – Default setting
- Output
- DHT11 - (Input for temperature and humidity measurement)
- DHT22 - (Input for temperature and humidity measurement)



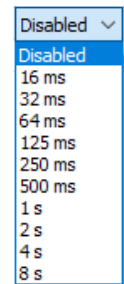
Furthermore, the GPIO configuration offers the possibility to activate the GPIO PullUps. If this setting is set to *On*, the GPIO PullUps can be activated by setting the respective GPIO output to *True* while a GPIO is configured as input.

7.12.1.3 Microcontroller Settings

The micro-controller of the *PiXtend V2 -L-* has two settings which the user can change.

1. *WatchdogEnable*

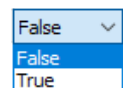
With this setting, the watchdog of the micro-controller can be switched on and off. The position *Disabled* means the watchdog is off (inactive), the selection of a waiting time activates the watchdog. There are times ranging from 16 milliseconds up to 8 seconds to choose from.



2. *StateLEDDisable*

The *PiXtend V2 -L-* has a status LED that signals an error if a problem is detected in the microcontroller. If necessary, this LED can be disabled.

The setting *False* means the LED is active (on), setting it to *True* deactivates the LED.





7.12.2. I/O Overview

In this chapter you will find a listing of all Ios available in CODESYS for the *PiXtend V2 -L-*, together with their data type and a short description.

Name	Type	Data type	Description
Control			
PWM0Ctrl1	Output	WORD	The effect of the PWM0Ctrl1 value depends on the selected Mode in PWM0Ctrl0. This value applies to channels A & B of PWM 0.
PWM1Ctrl1	Output	WORD	The effect of the PWM1Ctrl1 value depends on the selected Mode in PWM1Ctrl0. This value applies to channels A & B of PWM 1.
PWM2Ctrl1	Output	WORD	The effect of the PWM2Ctrl1 value depends on the selected Mode in PWM2Ctrl0. This value applies to channels A & B of PWM 1.
PWM0Ctrl0	Output	BYTE	PWM 0 Control 0 - Allows to set the PWM mode, channel enable and prescaler. PWM 0 uses 16 Bit values.
PWM1Ctrl0	Output	BYTE	PWM 1 Control 0 - Allows to set the PWM mode, channel enable and prescaler. PWM 1 uses 16 Bit values.
PWM2Ctrl0	Output	BYTE	PWM 2 Control 0 - Allows to set the PWM mode, channel enable and prescaler. PWM 2 uses 16 Bit values.
GPIODebounce01	Output	BYTE	Debounces the GPIO Inputs 0 and 1, value * Bus-Cycle interval = debounce time.
GPIODebounce23	Output	BYTE	Debounces the GPIO Inputs 2 and 3, value * Bus-Cycle interval = debounce time.
DigitalInDebounce01	Output	BYTE	Debounces the Digital Inputs 0 and 1, value * Bus-Cycle interval = debounce time.
DigitalInDebounce23	Output	BYTE	Debounces the Digital Inputs 2 and 3, value * Bus-Cycle interval = debounce time.
DigitalInDebounce45	Output	BYTE	Debounces the Digital Inputs 4 and 5, value * Bus-Cycle interval = debounce time.
DigitalInDebounce67	Output	BYTE	Debounces the Digital Inputs 6 and 7, value * Bus-Cycle interval = debounce time.
DigitalInDebounce89	Output	BYTE	Debounces the Digital Inputs 8 and 9, value * Bus-Cycle interval = debounce time.
DigitalInDebounce1011	Output	BYTE	Debounces the Digital Inputs 10 and 11, value * Bus-Cycle interval = debounce time.
DigitalInDebounce1213	Output	BYTE	Debounces the Digital Inputs 12 and 13, value * Bus-Cycle interval = debounce time.
DigitalInDebounce1415	Output	BYTE	Debounces the Digital Inputs 14 and 15, value * Bus-Cycle interval = debounce time.
SafeState	Output	BIT	Switch microcontroller to a safe state, in case the PLC needs to reboot or shutdown. True = Go to Safe State, False = Stay ON.
RetainDataEnable	Output	BIT	Activate the retain data storage function in the microcontroller. TRUE = On, FALSE = Off.



PiXtend V2 Software Manual

RetainCopy	Output	BIT	If TRUE then the current RetainDataOut is copied to RetainDataIn within the microcontroller. If FALSE the currently stored data in the microcontrollers memory is returned in RetainDataIn.
State			
Firmware	Input	BYTE	Firmware Version of the microcontroller on the PiXtend V2 -L- board.
Hardware	Input	BYTE	PiXtend V2 -L- board revision (hardware), 20 = 2.0, 21 = 2.1, etc.
Modelln	Input	BYTE	Model number as reported by the PiXtend V2 -L- board.
Error	Input	BYTE	Error byte from the microcontroller.
RetainCRCError	Input	BIT	The CRC of the retain area on the microcontroller is wrong, the data could be corrupted.
RetainVoltageError	Input	BIT	If TRUE then retain function cannot be used, the supply voltage is below 19 volts.
Run	Input	BIT	The communication with the microcontroller is running.
BusCycleError	Input	BIT	The cycle time (interval) of the Bus Cycle Task is too fast.
CRCHeaderInError	Input	BIT	A CRC error was detected in the SPI header, communication error.
CRCDataInError	Input	BIT	A CRC error in the PiXtend V2 -L- SPI data was detected, no usable data available.
ModellnError	Input	BIT	Firmware Version of the microcontroller on the PiXtend V2 -L- board.
I2CError	Input	BIT	An error was detected on the I2C bus or the microcontroller could not find a slave MC.
Sensor0Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Sensor1Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Sensor2Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Sensor3Error	Input	BIT	If TRUE, the microcontroller could not read data from the connected sensor at GPIO 0-3 therefore the corresponding values Temp and Humid are not valid.
Analog Inputs			
AnalogIn0	Input	REAL	Analog Input 0 is measured in Volts [V], range is 0 Volt to +10 Volts.
AnalogIn0Raw	Input	WORD	Analog Input 0 raw value, this is the actual 16 bit value from the ADC.
AnalogIn1	Input	REAL	Analog Input 1 is measured in Volts [V], range is 0 Volt to +10 Volts.



PiXtend V2 Software Manual

AnalogIn1Raw	Input	WORD	Analog Input 1 raw value, this is the actual 16 bit value from the ADC.
AnalogIn2	Input	REAL	Analog Input 2 is measured in Volts [V], range is 0 Volt to +10 Volts.
AnalogIn2Raw	Input	WORD	Analog Input 2 raw value, this is the actual 16 bit value from the ADC.
AnalogIn3	Input	REAL	Analog Input 3 is measured in Volts [V], range is 0 Volt to +10 Volts.
AnalogIn3Raw	Input	WORD	Analog Input 3 raw value, this is the actual 16 bit value from the ADC.
AnalogIn4	Input	REAL	Analog input 4 measured in milliamps [mA], range is 0 mA to +20 mA.
AnalogIn4Raw	Input	WORD	Analog input 4 raw value, actual 16 bit value from the ADC.
AnalogIn5	Input	REAL	Analog input 5 measured in milliamps [mA], range is 0 mA to +20 mA.
AnalogIn5Raw	Input	WORD	Analog input 5 raw value, actual 16 bit value from the ADC.
Digital Inputs			
DigitalInputs0	Input	BYTE	Digital Inputs 0-7 as Byte, Bit access is also possible.
DigitalInputs1	Input	BYTE	Digital Inputs 8-15 as Byte, Bit access is also possible.
GPIOInputs	Input	BYTE	GPIO Inputs 0-3 as Byte, Bit access is also possible. GPIO inputs will only work if so configured in the Parameter section of the PiXtend device.
Digital Outputs			
DigitalOutputs0	Output	BYTE	Digital Outputs 0-7 as Byte, Bit access is also possible.
DigitalOutputs1	Output	BYTE	Digital Outputs 8-11 as Byte, Bit access is also possible.
GPIOOutputs	Output	BYTE	PiXtend GPIO Outputs 0-3 as Byte, Bit access is also possible. GPIO outputs will only work if so configured in the Parameter section of the PiXtend device.
RelayOutputs	Output	BYTE	Relay outputs 0-3 a Byte, Bit access is also possible.



PiXtend V2 Software Manual

PWM Outputs			
PWM0A	Output	WORD	PWM 0, channel A output value, the actual PWM behaviour depends on the mode setting in PWM0Ctrl0.
PWM0B	Output	WORD	PWM 0, channel B output value, the actual PWM behaviour depends on the mode setting in PWM0Ctrl0.
PWM1A	Output	WORD	PWM 1, channel A output value, the actual PWM behaviour depends on the mode setting in PWM1Ctrl0.
PWM1B	Output	WORD	PWM 1, channel B output value, the actual PWM behaviour depends on the mode setting in PWM1Ctrl0.
PWM2A	Output	WORD	PWM 2, channel A output value, the actual PWM behaviour depends on the mode setting in PWM2Ctrl0.
PWM2B	Output	WORD	PWM 2, channel B output value, the actual PWM behaviour depends on the mode setting in PWM2Ctrl0.
Humidity Inputs	Humidity measured by a sensor connected to GPIO 0-3, if activated in the SPI devices Parameter tab. The values are given in % [RH].		
Humid0	Input	REAL	
Humid1	Input	REAL	
Humid2	Input	REAL	
Humid3	Input	REAL	
Temperature Inputs	Temperature of a sensor connected to GPIO 0-3, if activated in the SPI devices Parameter tab. The values are given in degrees Celcius (°C).		
Temp0	Input	REAL	
Temp1	Input	REAL	
Temp2	Input	REAL	
Temp3	Input	REAL	
Retain Data	Retain data can be used to save information in the microcontrollers memory in case of a sudden powerloss. The Retain data is 64 bytes in size.		
RetainDataOut	Output	ARRAY[0..63] OF BYTE	
RetainDataIn	Input	ARRAY[0..63] OF BYTE	

Table 3: PiXtend V2 -L- CODESYS I/Os overview



7.13. FAQ – Frequently Asked Questions and Troubleshooting

7.13.1. CODESYS Raspberry Pi Runtime and PiXtend V2

Problem

The green LED "+ 5 V" lights up and the Raspberry Pi starts up normally. However, the data exchange between Raspberry Pi and *PiXtend V2* does not seem to work (for example, relays or outputs can not be set).

Troubleshooting

Check that the "SPI_EN" switch is in the "ON" position. This makes communication between Raspberry Pi and *PiXtend V2* possible.

If you are working under CODESYS and have created your own project, always make sure that the GPIO24 (on the Raspberry Pi) is configured as an output and set to "TRUE" (see Image. 89). The GPIO24 is the signal which is transmitted via the "SPI_EN" switch and activates the data transmission.

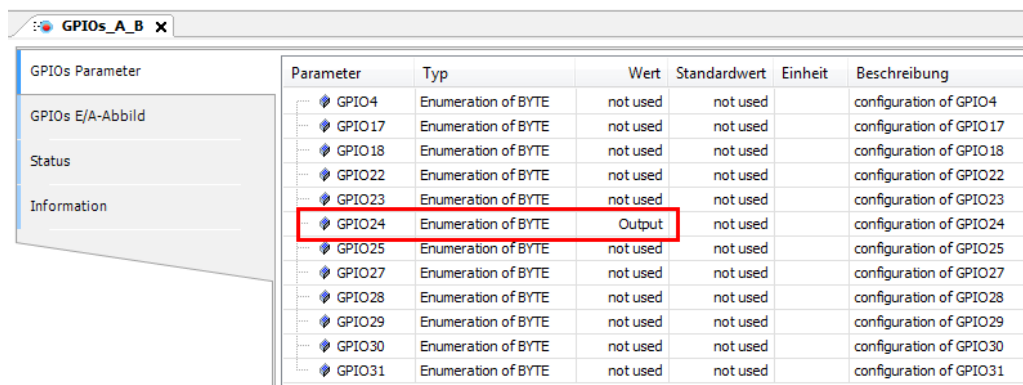


Figure 89: CODESYS - GPIO24 configured as output



7.13.2. Serial Communication

7.13.2.1 Not all characters are transmitted or arrive one after the other

If too many characters are sent, it may be that not all characters can be transmitted at once. A cycle is then not enough to send all characters. A remedy can be to increase the baud rate.

In order to roughly calculate how many characters can be transferred, the following calculation can be carried out:

Number of possible characters = (cycle time * baud rate) / bits to be transmitted

- ▶ the cycle time is expressed in seconds (10 ms = 0,010 s)
- ▶ the bits to be transmitted also contain, for example, a start or stop bit
- ▶ a character consists of 8 Bits (= 1 Byte)

7.13.2.2 Why are GPIOs 18 and 22 switched over in the program?

Applies only to PiXtend V1.2 / V1.3 and *PiXtend V2 -L-*!

GPIO_18:	TRUE	=>	RS485
	FALSE	=>	RS232
GPIO_22:	TRUE	=>	RS485 Send-Mode
	FALSE	=>	RS485 Receive-Mode

The switching of sending and receiving for the RS485 has to be carried out manually, since the Raspberry Pi does not provide the necessary pins (RTS / CTS).

Notice:



The *PiXtend V2 -L-* also has an automatic send / receive mode switch, which is preset as a factory default. The GPIO 22 is not needed in this case.



PiXtend V2 Software Manual

7.13.2.3 Scrolling the table does not work properly

If the mouse pointer is over the table, scrolling may not work properly. Just move the mouse pointer out of the table.

7.13.2.4 The page / user interface is not displayed correctly

Because the page is a canvas element, it may not appear correctly. In this case reload the page (F5). In addition, a browser is needed which understands HTML5.

7.13.2.5 The first message is not transmitted correctly

When two *PiXtend V2* are connected, the first message sent is not transmitted correctly, but only the first character. This behavior can occur when both *PiXtend V2* and Raspberry Pi's are simultaneously started. The reason for this is the kernel message "Uncompressing Linux ..." which is always output at startup on the serial interface, even if it is actually deactivated. This is a problem of the Raspian Jessie version of 21.11.2015, which should be fixed with the next version.

7.13.2.6 Why is there no "Auto send" function on RS485?

Applies only to PiXtend V1.2 / V1.3 and PiXtend V2 -L-!

This function is prepared in the program but is not used. Since RS485 is a bus protocol, only one slave can send at the same time. If the "auto-send" function is used and more than one slave is transmitting at the same time, a collision may occur and data will not be transmitted correctly.

7.13.2.7 Not all baud rates are displayed

In this example, only some baud rates in the range of 9600 to 115200 are used. If other baud rates are required, these must also be entered into the program (in the text list "Baudrates" as well as in the main program in "state 5").

If the selection list is not displayed correctly, a change in the aspect ratio of the window can help or a reload of the page.



8. pxdev – Linux Tools & Library

This guide describes all the steps necessary to install **pxdev**, the Linux development tools for the *PiXtend V2* system (www.pixtend.com) – For a manual install see chapter: 8.4 Manual installation of pxdev

Alternatively, you can easily start with our prepared SD card images. We recommend this option especially for beginners and for a first commissioning of *PiXtend V2*, see also chapter: 8.3 Usage of the PiXtend V2 SD-Image „Basic“

The following components are contained in pxdev:

pixtend – The PiXtend / *PiXtend V2* C library allows access to *PiXtend V2* I/O hardware. The data exchange between microcontroller and Raspberry Pi takes place via SPI (Serial Peripheral Interface). The wiringPi-Library is used for this purpose.
(<https://projects.drogon.net/raspberry-pi/wiringpi/>)

pixtendtool2s - A command line tool to access the *PiXtend V2 -S-* I/O hardware and configuration bytes using simple console commands.

pxauto2s –A simple console application with a graphical user interface. *PiXtend V2 -S-* can be continuously monitored. The application is suitable for fast commissioning and I/O tests.

pixtendtool2l - A command line tool to access the *PiXtend V2 -L-* I/O hardware and configuration bytes using simple console commands.

pxauto2l –A simple console application with a graphical user interface. *PiXtend V2 -L-* can be continuously monitored. The application is suitable for fast commissioning and I/O tests.



8.1. Information

8.1.1. Usage of pixtendtool2(s/l) - Single commands for PiXtend V2

The `pixtendtool2(s/l)` provides direct access to the *PiXtend V2* hardware from the Linux command line. It is therefore suitable, among other things, for the following applications:

- Setting up individual commands via the command line: locally or also via SSH remote access over the network, e.g. outputs and inputs
- Call `pixtendtool2s` commands from your own shell scripts
- Cyclic invocation of `pixtendtool2(s/l)` instructions to periodically log data and provide it for other services, e.g. a MySQL database.
- Used as a reference for using the PiXtend / *PiXtend V2* C library in "manual mode" (single commands)
- Can be adapted to your own needs

8.1.2. Usage of pxauto2(s/l) – GUI for PiXtend V2

The `pxauto2(s/l)` tool has a graphical interface and is suitable for the quick commissioning of the *PiXtend V2* hardware. The states of the I/Os can be cyclically monitored.

- `pxauto2(s/l)` can be used both locally and remotely, i.e. with a SSH terminal
- I/O tests of connected hardware, sensors & actuators
- Testing the functionality of *PiXtend V2*
- Basis for own projects with GUI (Graphical User Interface)
- Used as a reference for the use of the PiXtend / *PiXtend V2* C library in "Auto Mode" (cyclic process image in C)



PiXtend V2 Software Manual

8.1.3. Usage of the PiXtend/PiXtend V2 C-Library

The PiXtend / *PiXtend* V2 C library is used by all prior listed programs and forms the interface between Raspberry Pi and the microcontroller used on *PiXtend* V2. Changes to the library are usually not necessary.

8.2. Prerequisites

Required Software

- *PiXtend* V2 SD-Image “Basic” (C / Python / Node-RED / OpenPLC) (**recommended!**)

Download from: <https://www.pixtend.de/downloads>

or:

- Current Raspbian (Strech) Linux-Distribution on SD Card
- wiringPi (<https://projects.drogon.net/raspberry-pi/wiringpi/>)
- ncurses Libraries `libncurses5-dev libncursesw5-dev`
- Optional SSH-Client (e.g. putty.exe – www.putty.org)

Required Hardware

PiXtend V2 Board (www.pixtend.com)

Raspberry Pi Model B+ / 2 B / 3 B / 3 B+



8.3. Usage of the PiXtend V2 SD-Image „Basic“

The convenient way for first tests and the further work with pxdev is the use of one of our preinstalled SD card or the SD card image.

The latest image can be downloaded free of charge from our homepage:

<https://www.pixtend.de/downloads/>

How to write the image to an SD card can be found in chapter 6.4 Preparing a SD card.

After booting, change to the pxdev folder with the following command:

```
cd pxdev
```

or into the folder of "pixtendtools2s":

```
cd pxdev/PiXtend_V2/pixtendtool2s/
```

or for *PiXtend V2 -L-* into the folder "pixtendtools2l":

```
cd pxdev/PiXtend_V2/pixtendtool2l/
```

The chapter 8.4 Manual installation of pxdev can be skipped and continue directly with the use of the tools or the library as found in chapter 8.5 Working with pixtendtool2(s/l).



8.4. Manual installation of pxdev

8.4.1. Preparation and Installation

The latest version of the pxdev package can be downloaded from our homepage in the download area as a .zip archive or checked out with GIT from the pxdev repository. GIT is used for version management and in this case allows to download complete repositories from the Internet with simple commands. We recommend the following procedure:

Enter the following commands:

```
sudo apt-get update
sudo apt-get install git-core
```

in the Raspberry Pi command line to install GIT on the Raspberry Pi (if not already present).

Pxdev uses internally the wiringPi library (<https://projects.drogon.net/raspberry-pi/wiringpi/>) as its basis.

Install the wiringPi library using GIT as follows:

```
cd ~
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
```

pxauto2(s/l) uses the *ncurses* library.

Install the required packages:

```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

Install the pxdev package into the home directory of your Raspberry Pi.

```
cd ~
```



PiXtend V2 Software Manual

Clone the pxdev repository as follows:

```
git clone git://git.code.sf.net/p/pixtend/pxdev pxdev
```

Use the `ls` command to display the contents of your home directory.

Change to the folder pxdev:

```
cd pxdev
```

with `ls -la` you can see the contents of the pxdev package:

It contains a simple build script.

Make the script executable:

```
chmod +x build
```

You can now start the build script:

```
./build
```

The PiXtend / *PiXtend V2* library, the *pixtendtool2s*, the *pxauto2s*, the *pixtendtool2l*, and the *pxauto2l* tool are now build in sequence. The shell script also creates the programs for PiXtend V1.x, which you can ignore.

8.4.2.Enable SPI Bus

Communication between the Raspberry Pi and the *PiXtend V2* microcontroller uses SPI. Make sure that the SPI module on the Raspberry Pi is activated. In the following, we will show you how the SPI bus is activated.

Open *raspi-config*:

```
sudo raspi-config
```

and enable the SPI bus under „5 Interfacing Options“ → P4 SPI → „Yes“ → „Ok“

After that, *raspi-config* should offer to restart the system, if not do so manually.

```
sudo reboot
```



8.5. Working with pixtendtool2(s/l)

The following instructions and examples are based on the *PiXtend V2 -S-*, but they also apply to the *PiXtend V2 -L-* if the corresponding program is used. Instead of "*pixtendtool2s*" then use "*pixtendtool2l*".

Change to the directory that has the *pixtendtool2s* program e.g. with

```
cd ~/pxdev/PiXtend_V2/pixtendtool2s
```

Enter the following command:

```
sudo ./pixtendtool2s -h
```

This command returns all possible values and options / parameters:

```
PiXtend Tool V2 -S- http://www.pixtend.de - Version 0.5.5
usage: sudo ./pixtendtool2s [OPTION] [VALUE(s)]

Available options:
-h                                Print this help
-do VALUE                        Set the digital output byte to VALUE[0-255]
-do BIT VALUE                    Set the digital output BIT[0-3] to VALUE [0/1]
-dor                             Get the digital output byte
-dor BIT                        Get the digital output BIT[0-3]
-di                             Get the digital input byte
-di BIT                         Get the digital input BIT[0-7]
-ai CHANNEL                     Get the analog input CHANNEL[0-1] raw value
-ai CHANNEL REF                 Get the analog input CHANNEL[0-1] value based on REF[5V/10V]
-ao CHANNEL VALUE               Set the analog output CHANNEL[0-1] to VALUE[0-1023]
-rel VALUE                      Set the relay output byte to VALUE[0-255]
-rel BIT VALUE                  Set the relay output BIT[0-3] to VALUE[0/1]
-relr                           Get the relay output byte
-relr BIT                       Get the relay output BIT[0-3]
-gw VALUE                       Set GPIO output byte to VALUE[0-255]
-gw BIT VALUE                   Set GPIO output BIT[0-3] to VALUE[0/1]
-gr                             Get GPIO input byte
-gr BIT                         Get GPIO input BIT[0-3]
-gc VALUE                       Set GPIO control to VALUE[0-255]
-tr CHANNEL TYPE                Get temperature from CHANNEL[0-3] of TYPE[DHT11/DHT22]
-hr CHANNEL TYPE                Get humidity from CHANNEL[0-3] of TYPE[DHT11/DHT22]
-srv0 CHANNEL VALUE             Set servo 0 CHANNEL[0-1] to VALUE[0-65535]
-srv1 CHANNEL VALUE             Set servo 1 CHANNEL[0-1] to VALUE[0-255]
-pwm0 CHANNEL VALUE             Set PWM 0 CHANNEL[0-1] to VALUE[0-65535]
```



PiXtend V2 Software Manual

<code>-pwm1 CHANNEL VALUE</code>	Set PWM 1 CHANNEL[0-1] to VALUE[0-255]
<code>-pwm0c VALUE0 VALUE1</code>	Set PWM 0 control VALUE0[0-255] and VALUE1[0-65535]
<code>-pwm1c VALUE0 VALUE1</code>	Set PWM 1 control VALUE0[0-255] and VALUE1[0-255]
<code>-swc SERIALDEVICE BAUDRATE CHAR</code>	Write a CHAR on SERIALDEVICE
<code>-sws SERIALDEVICE BAUDRATE STRING</code>	Write a STRING on SERIALDEVICE (max 255)
<code>-sr SERIALDEVICE BAUDRATE</code>	Read data from SERIALDEVICE until Ctrl^C
<code>-ucc0 VALUE</code>	Set the microcontroller control 0 to VALUE[0-255]
<code>-ucc1 VALUE</code>	Set the microcontroller control 1 to VALUE[0-255]
<code>-ucs</code>	Get microcontroller status register
<code>-ucr</code>	Reset the microcontroller
<code>-ucv</code>	Get microcontroller version
<code>-ucw</code>	Get microcontroller warnings

Note for PWM 0/1 and Servo 0/1: 0 = channel A and 1 = channel B

The output may differ if there is a new version of the pxtendtool2s. The output shown here refers to version 0.5.5.

8.5.1.Examples

The command

```
sudo ./pxtendtool2s -di
```

provides information on the status of the digital input byte at the time of the call:

```
pi@raspberrypi:~/pxdev/PiXtend_V2/pxtendtool2s $ sudo ./pxtendtool2s -di
Digital input byte is [0]
```

Figure 90: Linux-Tools - pxtendtool2s - Switch '-di'

```
sudo ./pxtendtool2s -do 15
```

Sets the digital outputs to dec. 15. This represents binary 00001111, thus setting the outputs 0, 1, 2 and 3.

To query analog inputs, use the following:

```
sudo ./pxtendtool2s -ai 0
```

```
pi@raspberrypi:~/pxdev/PiXtend_V2/pxtendtool2s $ sudo ./pxtendtool2s -ai 0
Analog input 0 value is [516]
```

Figure 91: Linux-Tools - pxtendtool2s - Switch '-ai'



PiXtend V2 Software Manual

Analog input 0 gives us the value 516. For a reference voltage of 10 V and a resolution of 10 bits ($2 \times 10 = 1024$), the value thus corresponds to $10 \text{ V} / 1024 * 516 = 5.04 \text{ volts}$.

Serial transmissions via RS232 can also be done very easily with the `pixtendtool2s`.

A string is sent with the following command via the UART interface of the Raspberry Pi:

```
sudo ./pixtendtool2s -sws /dev/ttyS0 115200 test123
```

```
pi@raspberrypi:~/pxdev/PiXtend_V2/pixtendtool2s $ sudo ./pixtendtool2s -sws /dev/ttyS0 115200 test123
Opened serial device [fd=6]
put string test123
Closed serial device
```

Figure 92: Linux-Tools - `pixtendtool2s` - Switch '-sws'

With the Raspberry Pi 3 B (3 B+), the UART interface is named `"/dev/ttyS0"`. Here, the string (string) "test123" is transmitted with a baud rate of 115.200 baud.

PiXtend V2 -S- has a UART interface which connects to a RS232 converter.¹

Of course, serial interfaces can also receive / read:

```
sudo ./pixtendtool2s -sr /dev/ttyS0 115200
```

```
pi@raspberrypi:~/pxdev/PiXtend_V2/pixtendtool2s $ sudo ./pixtendtool2s -sr /dev/ttyS0 115200
Opened serial device [fd=6]
Recv: Hallo PiXtend V2 -S-!
```

Figure 93: Linux-Tools - `pixtendtool2s` - Switch '-sr'

Additional notes on the UART interface of the Raspberry Pi:

The serial interface of the Raspberry Pi can be used as a remote console. If, however, the interface is to be used exclusively for data transfers, the program `raspi-config` must be used:

```
sudo raspi-config
```

Then select:

5 *Interfacing Options* --> P6 *Serial* --> <No> --> <Yes> --> <Ok>

Afterwards a reboot is necessary:

```
sudo reboot
```

¹ The *PiXtend V2 -L-* also has an RS485 converter that can be activated via GPIO 18. The RS232 interface is then no longer available on the *PiXtend V2 -L-*.



PiXtend V2 Software Manual

8.5.2. Further steps

Experiment with the options and parameters.

After you are familiar with the use and parameters of the `pixtendtool2(s/l)`, you can write your own shell scripts in which you use the `pixtendtool2(s/l)` program.

You can send the output of `pixtendtool2(s/l)` e.g. directly into a text file:

PiXtend V2 -S-:

```
sudo ./pixtendtool2s -h > Help.txt
```

PiXtend V2 -L-:

```
sudo ./pixtendtool2l -h > Help.txt
```

or append data to an existing file:

PiXtend V2 -S-:

```
sudo ./pixtendtool2s -ai 0 >> analog0.log
```

PiXtend V2 -L-:

```
sudo ./pixtendtool2l -ai 0 >> analog0.log
```

Set up a cronjob², which cyclically calls a script and stores the states of all analog inputs in a file.

- Or a script that enters the analog values into a MySQL database.
- Or a script that turns on a relay at a certain time every day, and off at another time.

Perhaps you were wondering how exactly the control registers should be used or certain values are processed. For this purpose, there is further information in the appendix and its sub chapters (chapter 14 Appendix and following).

In these chapters you will find bit-precise all the information and possibilities the *PiXtend V2 -S-* and *PiXtend V2 -L-* offers.

² <https://en.wikipedia.org/wiki/Cron>



8.6. Using pxauto2(s/l)

The following instructions and examples are based on the *PiXtend V2 -S-*, but they also apply to the *PiXtend V2 -L-* if the corresponding program is used. Instead of "*pxauto2s*" then use "*pxauto2l*".

Change to the directory containing pxauto2s with:

```
cd ~/pxdev/PiXtend_V2/pxauto2s
```

Now enter the command

```
sudo ./pxauto2s
```

to start pxauto2s.



The use of the "sudo" command is very important here. If pxauto2(s/l) is started without "sudo", unexpected behavior can occur, since the hardware can not be accessed. In addition, the console output might be interrupted.

pxauto2s currently has the following menu items:

- DIIn - Digital inputs
- AIIn - Analog Inputs / Temperature / Humidity - DHT11 / 22 Setting
- GPIO - General Purpose Input / Output
- DOut - Digital Outputs
- AOut - Analog Outputs
- PWM - PWM Outputs
- Ctrl - Control, Debounce, and Jumper-Bytes
- Stat - Status Bytes
- RetIn - Retain Input (see chapter 6.3 Retain Memory)
- RetOut - Retain Output (see chapter 6.3 Retain Memory)



Figure 94: Linux-Tools - pxauto2s - Main menu



8.6.1. Navigation

After starting *pxauto2(s/l)*, you are initially in the menu (MENU-MODE)

Use the UP and DOWN arrow keys to select another menu item.

Press RETURN to move to the currently selected window (EDIT-MODE).

Color-highlighted fields (yellow) can be edited by typing a value.

To delete values, use the keyboard keys DEL / BACKSPACE.

Boolean values (e.g., digital outputs) and selections can be selected with the LEFT / RIGHT arrow keys.

To accept changes, leave the field with the UP / DOWN buttons.

To return to the menu, press RETURN again. This operation automatically takes the value in the current field.

Pressing the "q" key on the main menu will exit the program.

8.6.2. Edit fields

The highlighted fields (yellow) can be changed. Use the arrow keys to navigate to the desired field and use LEFT / RIGHT to select between the available values, or type a new value directly.

(Note: num-block is not supported in SSH-Terminal)



Figure 95: Linux-Tools - pxauto2s - Menu DOut



With the PiXtend / *PiXtend* V2 C-Library ("pxdev"), you will be able to program your own Linux programs for your *PiXtend* V2 board or integrate its functionalities into your own programs.

This chapter explains the steps required to create your own programs. In addition, you will get an overview of the possibilities offered by the C-Library. At the end of this chapter you will find two documented sample programs that you can test or use as a basis for your own developments.

[illegible]

```
Pixtend Tool V2 -> <- http://www.pixtend.de - Version 0.9.5.5
usage: sudo ./pixtendtool2 [OPTION] [VALUE(s)]

Available options:
  -b                               Print this help
  -do VALUE                        Set the digital output byte to VALUE[0-255]
  -dso BIT VALUE                  Set the digital output BIT[0-3] to VALUE[0/1]
  -dor                           Get the digital output byte
  -dsor BIT                      Get the digital output BIT[0-3]
  -di                             Get the digital input byte
  -disr BIT                      Get the digital input BIT[0-7]
  -ai CHANNEL                    Get the analog input CHANNEL[0-1] raw value
  -asr CHANNEL REF               Get the analog input CHANNEL[0-1] value based on REF[SU/LOV]
  -ao CHANNEL VALUE              Set the analog output CHANNEL[0-1] to VALUE[0-1023]
  -rel VALUE                     Set the relay output byte to VALUE[0-255]
  -relr BIT VALUE               Set the relay output BIT[0-3] to VALUE[0/1]
  -relr                          Get the relay output byte
  -relsr BIT                   Get the relay output BIT[0-3]
  -gw VALUE                     Set GPIO output byte to VALUE[0-255]
  -gsr BIT VALUE               Set GPIO output BIT[0-3] to VALUE[0/1]
  -gi                           Get GPIO input byte
  -gisr BIT[0-3]              Get GPIO input BIT[0-3]
  -gc VALUE                    Set GPIO control to VALUE[0-255]
  -ct CHANNEL TYPE             Get temperature from CHANNEL[0-3] or TYPE[DHT11/DHT22]
  -hr CHANNEL TYPE             Get humidity from CHANNEL[0-3] or TYPE[DHT11/DHT22]
  -arv 0 CHANNEL[0-1]         Set servo 0 CHANNEL[0-1] to VALUE[0-65535]
  -arv 1 CHANNEL[0-1]         Set servo 1 CHANNEL[0-1] to VALUE[0-255]
  -pmw 0 CHANNEL[0-1]         Set PWM 0 CHANNEL[0-1] to VALUE[0-65535]
  -pmw 1 CHANNEL[0-1]         Set PWM 1 CHANNEL[0-1] to VALUE[0-255]
  -pmwc 0 VALUE VALUE         Set PWM 0 control VALUE[0-255] and VALUE[1] [0-65535]
  -pmwc 1 VALUE VALUE         Set PWM 1 control VALUE[0-255] and VALUE[1] [0-255]
  -w CHAR ON SERIALDEVICE     Write a CHAR on SERIALDEVICE
  -ws SERIALDEVICE BAUDRATE STRING
                                Write a STRING on SERIALDEVICE (max 255)
  -rs SERIALDEVICE BAUDRATE    Read data from SERIALDEVICE until CtrlC
  -mc MICROCONTROLLER          Set the microcontroller control 0 to VALUE[0-255]
  -mcr                         Get the microcontroller control 1 to VALUE[0-255]
  -ucs                         Get microcontroller status register
  -urc                         Reset the microcontroller
  -uv                            Get microcontroller version
  -uw                            Get microcontroller warnings
```





9.1. Prerequisites

This chapter assumes that you have already installed the C-Library pxdev. If you have not already, see Chapter 8 pxdev – Linux Tools & Library for more information.

The easiest way is to use our SD card image "*PiXtend V2 Basic Image*" (C / Python / Node-RED / OpenPLC) which you can download free of charge from our homepage. In this image, pxdev and the required tools are already installed.

You should also familiarize yourself with chapter 14 (*Appendix*) and the sub chapters depending on which *PiXtend V2* board you have.

Basic knowledge in C programming is required. The handling of different number systems (decimal, binary, hexadecimal) and the conversion between these must also be known.

You can always find the latest source code of the PiXtend / *PiXtend V2* library at <https://sourceforge.net/projects/pixtend/>. If you are programming on Linux, it can be useful to display the source code on a PC or to print them out.

This avoids a constant change between your program code and the source code of pxdev.



9.2. Preparation

The data exchange between *PiXtend V2* and Raspberry Pi is done cyclic via the SPI bus. This process is also called Auto Mode, following PiXtend V1.3. This makes it possible to exchange all input and output values at once. Information on SPI and cycle times can be found in chapter 6.2 SPI Communication, Data Transmission and Cycle Time.

The Auto Mode offers advantages such as an optional watchdog timer and a check for transmission errors (CRC checksum). Our example programs pxauto2(s/l) use the Auto Mode, as the name suggests.

Next, we look at the approach of Auto Mode and for that we create files and folders. In chapter 9.6 *Compile and run the program*, we make an executable program from our code.

Notice:

All instructions and examples are shown here using a *PiXtend V2 -S-* but work just as well with a *PiXtend V2 -L-*. At the end of a statement or command simply replace the "S" with an "L" to be able to use a *PiXtend V2 -L-*.

We always work in the user's home directory, "pi". This avoids problems with read and write permissions, which can occur in other folders.

After logging on via SSH or after booting and using keyboard and mouse, connected directly to the Raspberry Pi, you always end up automatically in this directory. If you have changed the directory, return to it with the following command:

```
cd /home/pi
```

9.2.1. Create a new folder and a C-file

```
mkdir pixCEExample  
cd pixCEExample
```

Now create a new C-file pixCEExample and edit it with the editor "nano":

```
touch pixCEExample.c  
nano pixCEExample.c
```

The nano editor will start and you will see that the pixCEExample.c file is empty.



9.2.2. Including the libraries

Now we can integrate the required PiXtend / *PiXtend* V2 C-Library (pxdev) into our C-file:

```
#include <pixtend.h>
```

You should also include the following standard C libraries:

```
#include <stdlib.h>
```

```
#include <stdio.h>
```

Now the real programming can start.

9.3. Programming

In this section the various C-functions, which pxdev offers, are presented and explained in more detail. The naming of the transfer parameters used in this documentation differ in part from the header file of the PiXtend / PiXtend V2 library. This is for the sake of clarity.

The SPI Auto Mode function allows cyclic and fast processing of the various PiXtend / *PiXtend* V2 C functions. The cyclic redundancy check (cyclic redundancy check CRC) checks the transmission of the SPI for errors. Furthermore, the use of the function offers the advantage that the watchdog of the *PiXtend* V2 microcontroller can be used. In case of a software error, this prevents an undefined state of the system. The software regularly notifies the watchdog that it is running properly. If the current program does not return after a certain time, the watchdog triggers a reset of the microcontroller and the program stops (see chapter 6.1 Definition "Safe State").

In order to set the cycle time and ensure the permanent transfer between Raspberry Pi and *PiXtend* V2, an infinite loop ("infinity loop") must be implemented in the Auto Mode. The loop is called again depending on a condition or after a certain time. To set the cycle time, different waiting times values can be used. For an overview of possible values, see chapter 6.2 SPI Communication, Data Transmission and Cycle Time.



PiXtend V2 Software Manual

Used functions:

```
int Spi_AutoModeV2S(struct pixtOutV2S *OutputData, struct pixtInV2S *InputData)
```

```
int Spi_AutoModeDAC(struct pixtOutDAC *OutputDataDAC)
```

Requirements:

The function Spi_SetupV2 (0) must be called for communication with the microcontroller.

The function Spi_SetupV2 (1) must be called for communication with the DAC (for analog outputs).

The program of the microcontroller starts automatically.

If the watchdog timer is to be used as well, the function must be called as follows:

```
OutputData.byUcCtrl0 = 0x07; //Watchdog set to 1 sec.
```

The function Spi_AutoModeV2S (struct OutputData, struct InputData) initiates a complete data exchange between PiXtend V2 -S- and Raspberry Pi. In order to use the function, a new structure of type pixtInV2S and pixtOutV2S must be created.

```
struct pixtInV2S InputData;  
struct pixtOutV2S OutputData;
```

Notice:



For the output data, here in this example “*OutputData*”, the variable *OutputData.byModelOut* must always be set to the value of the applicable PiXtend V2 model. The numbers are derived from the ASCII table for the letters “S” and “L”.

- „S“ has the ASCII decimal number „83“, this means for PiXtend V2 -S-: ***OutputData.byModelOut = 83***
- „L“ has the ASCII decimal number 76, this means for PiXtend V2 -L-: ***OutputData.byModelOut = 76***



PiXtend V2 Software Manual

The following variables are available in the *pixtOutV2S* structure for the *PiXtend V2 -S-*. For more information on the individual bits and bytes, please refer to chapter 14 Appendix.

OutputData.byModelOut;	//One Byte Model as handshake
OutputData.byUCMode	//Reserved
OutputData.byUCCtrl0;	//One Byte for uC Control 0
OutputData.byUCCtrl1;	//One Byte for uC Control 1
OutputData.byDigitalInDebounce01;	//One Byte for Digital Inputs 0 und 1 debounce
OutputData.byDigitalInDebounce23;	//One Byte for Digital Inputs 2 und 3 debounce
OutputData.byDigitalInDebounce45;	//One Byte for Digital Inputs 4 und 5 debounce
OutputData.byDigitalInDebounce67;	//One Byte for Digital Inputs 6 und 7 debounce
OutputData.byDigitalOut;	//One Byte for setting digital outputs
OutputData.byRelayOut;	//One Byte for setting relays
OutputData.byGPIOCtrl;	//One Byte for GPIO Control
OutputData.byGPIOOut;	//One Byte for GPIO Inputs 0 und 1 debounce
OutputData.byGPIODebounce23;	//One Byte for GPIO Inputs 2 und 3 debounce
OutputData.byPWM0Ctrl0;	//One Byte for PWM 0 Control 0
OutputData.wPWM0Ctrl1;	//Two Bytes for PWM 0 Control 1
OutputData.wPWM0A;	// Two Bytes for PWM 0 Channel A Value
OutputData.wPWM0B;	// Two Bytes for PMW 0 Channel B Value
OutputData.byPWM1Ctrl0;	//One Byte for PWM 1 Control 0
OutputData.byPWM1Ctrl1;	//One Byte for PWM 1 Control 1
OutputData.byPWM1A;	//One Byte for PWM 1 Channel A Value
OutputData.byPWM1B;	//One Byte for PWM 1 Channel B Value
OutputData.byJumper10V;	//One Byte for Jumper setting Analog In 0,1
OutputData.byGPIO0Dht11;	//One Byte DHT11 / 22 selection for GPIO 0
OutputData.byGPIO1Dht11;	//One Byte DHT11 / 22 selection for GPIO 1
OutputData.byGPIO2Dht11;	//One Byte DHT11 / 22 selection for GPIO 2
OutputData.byGPIO3Dht11;	//One Byte DHT11 / 22 selection for GPIO 3
OutputData.abbyRetainDataOut[32];	//One Array with 32 Bytes Retain Data Output



PiXtend V2 Software Manual

The following variables are available in the *pixtOutV2L* structure for the *PiXtend V2 -L-*. For more information on the individual bits and bytes, please refer to chapter 14 Appendix.

OutputData.byModelOut;	//One Byte Model as handshake
OutputData.byUCMode	//Reserved, not used
OutputData.byUCCtrl0;	//One Byte for uC Control 0
OutputData.byUCCtrl1;	//One Byte for uC Control 1
OutputData.byDigitalInDebounce01;	//One Byte for Digital Inputs 0 und 1 debounce
OutputData.byDigitalInDebounce23;	//One Byte for Digital Inputs 2 und 3 debounce
OutputData.byDigitalInDebounce45;	//One Byte for Digital Inputs 4 und 5 debounce
OutputData.byDigitalInDebounce67;	//One Byte for Digital Inputs 6 und 7 debounce
OutputData.byDigitalInDebounce89;	//One Byte for Digital Inputs 8 und 9 debounce
OutputData.byDigitalInDebounce1011;	//One Byte for Digital Inputs 10 und 11 debounce
OutputData.byDigitalInDebounce1213;	//One Byte for Digital Inputs 12 und 13 debounce
OutputData.byDigitalInDebounce1415;	//One Byte for Digital Inputs 14 und 14 debounce
OutputData.byDigitalOut0;	//One Byte for setting the digital outputs 0 – 7
OutputData.byDigitalOut1;	//One Byte for setting the digital outputs 8 – 11
OutputData.byRelayOut;	//One Byte for setting the relays
OutputData.byGPIOCtrl;	//One Byte for GPIO Control
OutputData.byGPIOOut;	//One Byte for setting the GPIO Outputs
OutputData.byGPIODebounce01;	//One Byte for GPIO Inputs 0 and 1 debounce
OutputData.byGPIODebounce23;	//One Byte for GPIO Inputs 2 and 3 debounce
OutputData.byPWM0Ctrl0;	//One Byte for PWM 0 Control 0
OutputData.wPWM0Ctrl1;	//Two Bytes for PWM 0 Control 1
OutputData.wPWM0A;	//Two Bytes for PWM 0 channel A value
OutputData.wPWM0B;	//Two Bytes for PMW 0 channel B value
OutputData.byPWM1Ctrl0;	//One Byte for PWM 1 Control 0
OutputData.wPWM1Ctrl1;	//Two Bytes for PWM 1 Control 1
OutputData.wPWM1A;	//Two Bytes for PWM 1 channel A value
OutputData.wPWM1B;	//Two Bytes for PWM 1 channel B value
OutputData.byPWM2Ctrl0;	//One Byte for PWM 2 Control 0
OutputData.wPWM2Ctrl1;	//Two Bytes for PWM 2 Control 1
OutputData.wPWM2A;	// Two Bytes for PWM 2 channel A value
OutputData.wPWM2B;	// Two Bytes for PWM 2 channel B value
OutputData.byJumper10V;	//One Byte for Jumper setting Analog In 0,1
OutputData.byGPIO0Dht11;	//One Byte DHT11 / 22 Selection for GPIO 0
OutputData.byGPIO1Dht11;	//One Byte DHT11 / 22 Selection for GPIO 1
OutputData.byGPIO2Dht11;	//One Byte DHT11 / 22 Selection for GPIO 2
OutputData.byGPIO3Dht11;	//One Byte DHT11 / 22 Selection for GPIO 3
OutputData.abbyRetainDataOut[64];	//One Array with 64 Bytes Retain Data Output



PiXtend V2 Software Manual

With `OutputData.byJumper10V` the jumper setting of the analog inputs AI0 and AI1 can be set to 5 V or 10 V. At lower voltages than 5 V, it makes sense to use the 5 V jumper setting, as this can be measured with a higher resolution.³

If the AI0 is set to 10 V, the variable must be filled as follows:

```
OutputData.byJumper10V = 0x01;
```

If AI1 is set to a jumper setting of 10 V, use:

```
OutputData.byJumper10V = 0x02;
```

If both analog inputs should be set to 10 V:

```
OutputData.byJumper10V = 0x03;
```

For a jumper setting of 5 V, the bits need not to be set or have to be written to with the value "0".

For more information, see the corresponding chapters in the Appendix for your *PiXtend V2* device.

³ Please remember to set the physical jumper on the PiXtend board before switching from 10 V to 5 V, otherwise you will get wrong values later in the program.



PiXtend V2 Software Manual

The input data of *PiXtend V2 -S-* can be taken from the following variables found in the structure *pixtInV2S*:

```
InputData.byFirmware;      // One byte for the microcontroller firmware version
InputData.byHardware;      // One byte for the hardware revision
InputData.byModelln;       // One byte model for handshake
InputData.byUCState;       // One byte for the microcontroller state
InputData.byUCWarnings;    // One byte for microcontroller warnings
InputData.byDigitalIn;     // One byte for the digital inputs
InputData.byGPIOIn;        // One byte for the GPIO inputs
InputData.wAnalogIn0;      // Two bytes for analog input 0
InputData.wAnalogIn1;      // Two bytes for analog input 1
InputData.wTemp0           // Two bytes for temperature value DHT11 / 22 of GPIO0
InputData.wTemp1           // Two bytes for temperature value DHT11 / 22 of GPIO1
InputData.wTemp2           // Two bytes for temperature value DHT11 / 22 of GPIO2
InputData.wTemp3           // Two bytes for temperature value DHT11 / 22 of GPIO3
InputData.wHumid0          // Two bytes for humidity value DHT11 / 22 of GPIO0
InputData.wHumid1          // Two bytes for humidity value DHT11 / 22 of GPIO1
InputData.wHumid2          // Two bytes for humidity value DHT11 / 22 of GPIO2
InputData.wHumid3          // Two bytes for humidity value DHT11 / 22 of GPIO3
InputData.rAnalogIn0;      // float variable for voltage of analog input 0
InputData.rAnalogIn1;      // float variable for voltage of analog input 1
InputData.rTemp0;          // float variable for temperature of GPIO0
InputData.rTemp1;          // float variable for temperature of GPIO1
InputData.rTemp2;          // float variable for temperature of GPIO2
InputData.rTemp3;          // float variable for temperature of GPIO3
InputData.rHumid0;         // float variable for humidity of GPIO0
InputData.rHumid1;         // float variable for humidity of GPIO1
InputData.rHumid2;         // float variable for humidity of GPIO2
InputData.rHumid3;         // float variable for humidity of GPIO3
InputData.abRetainDataIn[32]; // Array with 32 bytes of Retain Data Input
```



PiXtend V2 Software Manual

The input data of *PiXtend V2 -L-* can be taken from the following variables found in the structure *pixtlnV2L*:

```
InputData.byFirmware;    //One byte for the microcontroller firmware version
InputData.byHardware;    //One byte for the hardware revision
InputData.byModelIn;     //One byte model for handshake
InputData.byUCState;     //One byte for the microcontroller state
InputData.byUCWarnings; //One byte for microcontroller warnings
InputData.byDigitalIn0;  //One byte for the digital inputs 0 - 7
InputData.byDigitalIn1;  //One byte for the digital inputs 8 - 15
InputData.byGPIOIn;      //One byte for the GPIO inputs
InputData.wAnalogIn0;    //Two bytes for analog input 0
InputData.wAnalogIn1;    //Two bytes for analog input 1
InputData.wAnalogIn2;    //Two bytes for analog input 2
InputData.wAnalogIn3;    //Two bytes for analog input 3
InputData.wAnalogIn4;    //Two bytes for analog input 4
InputData.wAnalogIn5;    //Two bytes for analog input 5
InputData.wTemp0         //Two bytes for temperature value DHT11 / 22 of GPIO0
InputData.wTemp1         //Two bytes for temperature value DHT11 / 22 of GPIO1
InputData.wTemp2         //Two bytes for temperature value DHT11 / 22 of GPIO2
InputData.wTemp3         //Two bytes for temperature value DHT11 / 22 of GPIO3
InputData.wHumid0        //Two bytes for humidity value DHT11 / 22 of
InputData.wHumid1        //Two bytes for humidity value DHT11 / 22 of GPIO1
InputData.wHumid2        //Two bytes for humidity value DHT11 / 22 of GPIO2
InputData.wHumid3        //Two bytes for humidity value DHT11 / 22 of GPIO3
InputData.rAnalogIn0;    //float variable for voltage of analog input 0
InputData.rAnalogIn1;    //float variable for voltage of analog input 1
InputData.rAnalogIn2;    //float variable for voltage of analog input 2
InputData.rAnalogIn3;    //float variable for voltage of analog input 3
InputData.rAnalogIn4;    //float variable for voltage of analog input 4
InputData.rAnalogIn5;    //float variable for voltage of analog input 5
InputData.rTemp0;        //float variable for temperature of GPIO0
InputData.rTemp1;        //float variable for temperature of GPIO1
InputData.rTemp2;        //float variable for temperature of GPIO2
InputData.rTemp3;        //float variable for temperature of GPIO3
InputData.rHumid0;       //float variable for humidity of GPIO0
InputData.rHumid1;       //float variable for humidity of GPIO1
InputData.rHumid2;       //float variable for humidity of GPIO2
InputData.rHumid3;       //float variable for humidity of GPIO3
InputData.abRetainDataIn[64]; //Array with 64 bytes of Retain Data Input
```



PiXtend V2 Software Manual

To pass the *Input* and *Output* structures into the function, the following line is used:

```
Spi_AutoModeV2S(&OutputData, &InputData);
```

The structure *OutputData* contains the values for the control bytes and process data (output values from the viewpoint of the Raspberry Pi to *PiXtend V2 -S-*).

In the *InputData* structure, the values of the inputs are written during the programs execution which came from the *PiXtend V2 -S-* microcontroller).

A short example:

With the following code, the temperature can be read out from a DHT sensor via GPIO0 and output to the console:

```
dht0Temp=InputData.rTemp0;  
printf("Temperature C: %.2f\n", dht0Temp);
```

The function *Spi_AutoModeDAC* (struct *OutputDataDAC*) is only used for communication with the digital-analog converter (DAC). The function requires a structure of type *pixtOutDAC* as the transfer parameter. This can be created as follows:

```
struct pixtOutDAC OutputDataDAC
```

The structure must contain the values for the two analog outputs.

```
OutputDataDAC.wAOut0 = 1023; //1023 corresponds to the max. value, 0 min. value  
OutputDataDAC.wAOut1 = 1023;
```

Here, the analog outputs are set to 100%.

With the following call, we pass the structure *OutputDataDAC* to the function.

```
Spi_AutoModeDAC(&OutputDataDAC);
```



9.4. Example Program for PiXtend V2 -S-

```
//Include the libraries
#include <pixtend.h>
#include <stdio.h>

//Function prototype
int GetPixData();

//Define some variables
float dht1Temp=0.0;
float dht1Hum=0.0;

//PiXtend V2 -S- input data
struct pixtInV2S InputData;

//PiXtend V2 -S- output data
struct pixtOutV2S OutputData;

//PiXtend V2 -S- DAC Output Data
struct pixtOutDAC OutputDataDAC;

//Read Temperature and Humidity from GPIO1 (DHT22)
int GetPixData()
{
    dht1Temp=InputData.rTemp1;
    dht1Hum=InputData.rHumid1;

    //Only print out valid data
    if(dht1Hum>0.0)
    {
        //Write out the data on the linux console
        printf("Temperature [°C]: %.2f\n", dht1Temp);
        printf("Humidity [%%]: %.2f\n", dht1Hum);
    }

    return 0;
}

int main(void)
{
    //Configure SPI
    Spi_SetupV2(0);
    Spi_SetupV2(1);

    //Write Data in Structure OutputData
    OutputData.byModelOut = 83; // Set model as handshake
    OutputData.byGPIOCtrl = 32; //GPIO1 is used with DHT22 sensor
    OutputDataDAC.wAOut0 = 1023; //pre-load analog outputs
    OutputDataDAC.wAOut1 = 1023;
```




PiXtend V2 Software Manual

```
//Auto Mode in infinity loop
while(1)
{
    //Do the data transfer!
    Spi_AutoModeV2S(&OutputData, &InputData);
    Spi_AutoModeDAC(&OutputDataDAC);

    //Set all Relays if at least one digital input is High
    if(InputData.byDigitalIn>0)
    {
        printf("Input detected - setting Relays!\n");
        OutputData.byRelayOut = 15;
    }
    else
    {
        OutputData.byRelayOut = 0;
    }

    //Toggle Analog Outputs
    if(OutputDataDAC.wAOut0 == 1023 || OutputDataDAC.wAOut1 == 1023)
    {
        OutputDataDAC.wAOut0 = 512;
        OutputDataDAC.wAOut1 = 512;
    }
    else
    {
        OutputDataDAC.wAOut0 = 1023;
        OutputDataDAC.wAOut1 = 1023;
    }

    //Read data and print it out
    GetPixData();

    //Take a 100 ms break between the cycles
    delay(100);
}
return 0;
}
```



9.5. Example Program for PiXtend V2 -L-

```
//Include the libraries
#include <pixtend.h>
#include <stdio.h>

//Function prototype
int GetPixData();

//Define some variables
float dht1Temp=0.0;
float dht1Hum=0.0;

//PiXtend V2 -L- input data
struct pixtInV2L InputData;

//PiXtend V2 -L- output data
struct pixtOutV2L OutputData;

//PiXtend V2 -L- DAC Output Data
struct pixtOutDAC OutputDataDAC;

//Read Temperature and Humidity from GPIO1 (DHT22)
int GetPixData()
{
    dht1Temp=InputData.rTemp1;
    dht1Hum=InputData.rHumid1;

    //Only print out valid data
    if(dht1Hum>0.0)
    {
        //Write out the data on the linux console
        printf("Temperature [°C]: %.2f\n", dht1Temp);
        printf("Humidity [%%]: %.2f\n", dht1Hum);
    }

    return 0;
}

int main(void)
{
    //Configure SPI
    Spi_SetupV2(0);
    Spi_SetupV2(1);

    //Write Data in Structure OutputData
    OutputData.byModelOut = 76; // Set model as handshake, PiXtend V2 -L- = 76
    OutputData.byGPIOCtrl = 32; //GPIO1 is used with DHT22 sensor
    OutputDataDAC.wAOut0 = 1023; //pre-load analog outputs
    OutputDataDAC.wAOut1 = 1023;
```



PiXtend V2 Software Manual

```
//Auto Mode in infinity loop
while(1)
{
    //Do the data transfer!
    Spi_AutoModeV2L(&OutputData, &InputData);
    Spi_AutoModeDAC(&OutputDataDAC);

    //Set all Relays if at least one digital input is High
    if(InputData.byDigitalIn0>0 || InputData.byDigitalIn1>0)
    {
        printf("Input detected - setting Relays!\n");
        OutputData.byRelayOut = 15;
    }
    else
    {
        OutputData.byRelayOut = 0;
    }

    //Toggle Analog Outputs
    if(OutputDataDAC.wAOut0 == 1023 || OutputDataDAC.wAOut1 == 1023)
    {
        OutputDataDAC.wAOut0 = 512;
        OutputDataDAC.wAOut1 = 512;
    }
    else
    {
        OutputDataDAC.wAOut0 = 1023;
        OutputDataDAC.wAOut1 = 1023;
    }

    //Read data and print it out
    GetPixData();

    //Take a 100 ms break between the cycles
    delay(100);
}
return 0;
}
```



9.6. Compile and run the program

To run the program, it must first be compiled. The following line must be entered for the compilation:

```
sudo gcc -Wall -o "pixCEExample" "pixCEExample.c" -lpixtend -lwiringPi
```

Now pixCEExample.c is compiled and stored in pixCEExample program.

With the command:

```
sudo ./pixCEExample
```

the program will be executed.

If you do not want to use sudo, the compiled file pixCEExample can receive the needed rights. For this purpose, the following command can be used:

```
sudo chmod -R 0777 pixCEExample
```

The *sudo*⁴ command can now be omitted when running the C program.

To exit the program, simply use the key combination CTRL + C.

⁴ *sudo* is necessary here, because the SPI interface is addressed via the wiringPi-Library (<http://wiringpi.com/>). This access still requires superuser privileges, depending on the version.



9.7. Frequently Asked Questions (FAQ)

What do I need to take into account when using the PiXtend / PiXtend V2 C library?

The SPI interface must be configured with the function `Spi_SetupV2 (int device)`. The `Spi_SetupV2` function can be called up to two times (once for device 0, once for device 1). Therefore, the calls should be moved to an initialization routine that is executed only once (immediately after the program is started).

The output byte `byModelOut` must be transmitted to *PiXtend V2* of the handshake, which is always the decimal number 83.

There is sometimes a transfer problem between PiXtend and Raspberry Pi. What can I do?

In your program, check whether you call the function `Spi_SetupV2 (int device)` several times or cyclically. If this is the case, this is the reason for the transmission errors. If there is no transmission at all, check whether the switch `SPI_EN` on the *PiXtend V2* is ON. If DHT sensors are used, the cycle time must be at least 30 ms. Generally, the cycle time should not be less or faster than 2.5 ms (*PiXtend V2 -S-*) or 5 ms (*PiXtend V2 -L-*).



10. FHEM

FHEM is a PERL-based server program for home automation, which enables the user to integrate different sensors and actuators into a single system and to automate processes. These sensors and actuators can for example be switches, lamps, heaters or temperature and humidity sensors.

Therefore this FHEM module gives you full access to the functions of the *PiXtend V2* within the FHEM environment.

The FHEM module for the *PiXtend V2* developed by Qube Solutions GmbH is a stand-alone module for the Raspberry Pi and has no software dependencies to other modules or auxiliary programs.

10.1. Prerequisites

For the installation of FHEM on the *PiXtend V2* controller, basic knowledge of the Raspberry Pi and its Linux operating system is required. In addition to a *PiXtend V2* and a Raspberry Pi, the following parts are required:

- SD card with the latest Raspbian image. See chapter 6.4 Preparing a SD card to create such an SD card.
- SSH client software (e.g. putty.exe – www.putty.org)
- FTP client software (e.g. WinSCP – www.winscp.net)

For the subsequent application it is useful to get an overview of the available functions and possibilities the *PiXtend V2* offers.



Also note the hints for software compatibility in section 6.8 Compatibility of the Software Components.



10.2. Installation

For the installation of FHEM and the FHEM module for the *PiXtend V2* the following steps have to be taken. If FHEM is already installed the first step can be skipped and you can start directly with the integration of the module.

10.2.1. Installation of FHEM

The latest instructions for the installation of FHEM on the Raspberry Pi can be found under <https://debian.fhem.de/>.

It is recommended to execute the installation according to “The easy way: use apt-get” (listed in the left menu). For the version (21.09.2017 – Rev 5.8.15110) the instructions are as follows:

Update the repository-key for FHEM with the following command:

```
wget -qO - http://debian.fhem.de/archive.key | sudo apt-key add -
```

Now the address of the repository has to be added to the file ***sources.list***. Therefore the file has to be opened:

```
sudo nano /etc/apt/sources.list
```

and the next line is inserted at the end of the file. Save the file (*ctrl+x* and close with *y* and *enter*). This entry is automatically deleted when FHEM is installed.

```
deb http://debian.fhem.de/nightly/ /
```

After that the installed packages have to be updated depending of the new repository:

```
sudo apt-get update
```

In the next step FHEM can be installed (confirm the question of whether FHEM should be installed with ‘y’):

```
sudo apt-get install fhem
```



PiXtend V2 Software Manual

10.2.2. Integration of the module

To integrate the FHEM module for the *PiXtend V2* in to the FHEM environment the module has to be copied into the FHEM directory. The module can be found as a ZIP file in the [download section](#) of our homepage. Since the FHEM directory belongs to the user “fhem”, no files can be stored here without granting the permission. As a result the users of the group temporarily require the necessary write-permissions:

```
sudo chmod g+w /opt/fhem/FHEM
```

After that the module “**97_PiXtendV2.pm**” can be copied with an FTP client to the directory “**/opt/fhem/FHEM**”. Finally the permissions of the group should be reset:

```
sudo chmod g-w /opt/fhem/FHEM
```

10.2.3. Configuration of the system

The data transfer between Raspberry Pi and the *PiXtend V2* board uses SPI as its interface. Therefore SPI has to be enabled in the settings of the raspberry Pi. With

```
sudo raspi-config
```

the settings menu is opened and the interface can be enabled via “5 Interfacing Options” and “P4 SPI”. Exit the menu after that.

As a default the user “fhem” doesn’t have the permission to access the SPI or the GPIOs, but the access is required for the module to work and therefore the user has to be added to these groups:

```
sudo adduser fhem gpio; sudo adduser fhem spi; sudo reboot
```

This line also performs a required restart.

After the Raspberry Pi restarted the following address can be copied to the web browser to open the FHEM website of the Raspberry Pi:

```
http://<RaspberryIP>:8083/fhem
```

<RaspberryIP> has to be replaced by the IP-address of the Raspberry Pi. The current IP-address can be found with the following command:

```
ifconfig
```

The installation of FHEM and the integration of the module are done and you can start with the home automation.



10.3. Description of the module

The *PiXtend V2* module for FHEM has a variety of functions which can be used. On the following pages it is described how such a module can be created and how the functions can be used. The functions can be executed either via the FHEM web-interface of the module or by writing the appropriate command to the command line of FHEM.

The following description covers the independent sections of the module. An overview of the functions can be found directly in FHEM in the “device specific help” of this module as well.

10.3.1. Create a new PiXtend V2 device

A new *PiXtend V2 -S-* module can be created by writing the following command to the command line of FHEM. <name> must be replaced with the name the device should have.

```
define <name> PiXtendV2
```

To create a *PiXtend V2 -L-* device use:

```
define <name> PiXtendV2 L
```

Although it is not explicitly excluded to create several *PiXtend V2* modules in FHEM for only one physically existing device, it is not recommended! The multiple definition doesn't make sense and can lead to problems because the same hardware functions are accessed from several points.

10.3.2. Internals

The internals which belongs exclusively to this module are the “RetainSize” and the field “STATE”. RetainSize represents the size of retain data in bytes. The value of STATE can be “defined”, “active” or “error”. The value “defined” is just a short time after the definition of the module valid and then changes to “active”. If an error occurs the value changes to “error” and the field “ErrorMsg” contains a more accurate error information of the failure that caused the error. In most of the cases this is due to a faulty communication between *PiXtend V2* and the Raspberry Pi, e.g. if *PiXtend V2* is in safe state.

10.3.3. Set-commands

The *PiXtend V2* module has a variety of Set-commands to control digital outputs or relays and more. Commands to configure the basic settings start with an “_” and can be set via the attribute „PiXtend_Parameter“ too.

If a command supports multiple channels the “#”-sign has to be replaced with the channel number. All Set-commands are case insensitive to guarantee an easy use.



PiXtend V2 Software Manual

_GPIO#Ctrl [input,output,DHT11,DHT22]

With this setting the GPIO can be configured as [input], [output] or as [DHT11] or [DHT22] as well, if a DHT sensor is connected to this GPIO. If a DHT is selected and connected, the GPIO can't simultaneously be used as a normal GPIO but the temperature and humidity is read automatically.

_GPIOPullupsEnable [yes,no]

This setting enables [yes] or disables [no] the possibility to set the internal pullup resistors via GPIOOut setting for all GPIOs.

_JumperSettingAI# [5V,10V]

This setting affects the calculation of the voltage on the analog inputs and refers to the actual setting of the physical jumper on the *PiXtend V2*-Board [5V,10V]. The default value is [10V] if no jumper is used.

_StateLEDDisable [yes,no]

This setting disables [yes] or enables [no] the state LED on the *PiXtend V2*. If the LED is disabled it won't light up in case of an error.

_WatchdogEnable [disable,125ms,1s,8s]

This setting allows to configure the watchdog timer. If the watchdog is configured, the *PiXtend V2* will go to safe state if no valid data transfer has occurred within the selected timeout and thus can't be accessed anymore without a reset ("Reset").

AnalogOut# []

Sets the analog output to the selected voltage. The value can be a voltage between 0 V and 10 V or a raw value between 0 and 1023. To set the value to a voltage the value has to include a "." even if it is an even number.

DigitalDebounce# [0-255]

Allows to debounce the digital inputs. A setting always affects two channels. So DigitalDebounce01 affects DigitalIn0 and DigitalIn1. The resulting delay is calculated by



PiXtend V2 Software Manual

(selected value)*(100 ms). The selected value can be any number between 0 and 255. Debouncing can be useful if a switch or button is connected to this input.

DigitalOut# [on,off,toggle]

Set the digital output HIGH [on] or LOW [off] or [toggle]s it.

GPIODebounce# [0-255]

Allows to debounce the GPIO inputs. A setting always affects two channels. So GPIODebounce01 affects GPIOIn0 and GPIOIn1. The resulting delay is calculated by (selected value)*(100 ms). The selected value can be any number between 0 and 255. Debouncing can be usefull if a switch or button is connected to this input.

GPIOOut# [on,off,toggle]

Set the GPIO to HIGH [on] or LOW [off] or [toggle]s it, if it is configured as an output. If it is configured as an input, this command can enable [on] or disable [off] or [toggle] the internal pullup resistor for that GPIO, but therefore pullups have to be enabled globally via _GPIOPullupsEnable.

PWM

The *PiXtend V2* supports various PWM-Modes, which can be configured with these settings starting with "PWM". A description of the accurate meaning of these values and how the PWM modes can be changed can be found in chapter 14.

RelayOut# [on,off,toggle]

Set the relay to HIGH [on] or LOW [off] or [toggle]s it.

Reset

Resets the controller on the *PiXtend V2*, for example if it is in safe state and allows to access it again.

RetainCopy [on,off]

If RetainCopy is enabled [on] the RetainDataOut that is written to the *PiXtend V2* will be received in RetainDataIn again. This can be useful in some situations to see which data was send to the *PiXtend V2*. If it is disabled [off] the last stored data will be received.



RetainDataOut [0-(RetainSize-1)] [0-255]

PiXtend V2 supports the storage of retain data. If enabled, this data is stored in case of a power failure or if it is triggered by a watchdog timeout or the safe state command. The retain data is organized in bytes and each byte can be written individually with a value between 0 and 255. As first parameter the command needs the byte index which is between 0 and the (RetainSize-1). The RetainSize is shown in the "Internals". As the second parameter a value is expected which should be stored.

RetainEnable [on,off]

The function of storing retain data on the *PiXtend V2* has to be enabled [on], otherwise no data is stored [off]. The memory in which the data is stored supports 10.000 write-cycles. So the storage of retain data should only be used if it is really necessary.

SafeState

This setting allows to force the *PiXtend V2* to enter the safe state. If retain storage is enabled the data will be stored. In safe state the *PiXtend V2* won't communicate with FHEM and can't be configured. To restart the *PiXtend V2* a reset has to be done.

10.3.4. Get-commands

Get-commands allow to get the Readings or the system state. The format of the returned value can be changed with the attribute "PiXtend_GetFormat" and is represented as a raw value or in a message text.

If a command supports multiple channels the "#" -sign has to be replaced with the channel number. All Get-commands are case insensitive to guarantee an easy use.

AnalogIn#

Returns the Value of the selected analog input. The result depends on the selected _JumperSettingAI# and the actual physical jumper position on the board.

DigitalIn#

Returns the state 1 (HIGH) or 0 (LOW) of the digital input.

GPIOIn#



PiXtend V2 Software Manual

Returns the state 1 (HIGH) or 0 (LOW) of the GPIO, independent of its configuration (input, output, ..).

RetainDataIn [0-(RetainSize-1)]

Returns the value of the selected RetainDataIn-byte.

Sensor# [temperature,humidity]

If a DHT-sensor is connected to the corresponding GPIO and the `_GPIO#Ctrl` is set to DHT11 or DHT22 the temperature and humidity are measured and can be read.

SysState

Returns the system state [defined, active, error] of the FHEM module.

UCState

Returns the state of the *PiXtend V2*. If it is 1 everything is fine. If it is greater than 1 an error occurred or is present and the *PiXtend V2* can't be configured. More information can be found in chapter 14.1.2.4 and 14.2.2.4.

UCWarnings

Returns a value that represents the *PiXtend V2* warnings. For more information see chapter 14.1.2.4 and 14.2.2.4.

Version

Returns the FHEM module version and the microcontroller version [Model-Hardware-Firmware].



10.3.5. Readings

Readings are values which are provided by the controller, e.g. sensor values or the state of inputs. The meaning of the readings is similar to the Get-commands. Most of these readings trigger an event on change which allows FHEM and the user to react to this change.

AnalogIn#

Shows the result of the measurement on the analog inputs in V.

DigitalIn#

Shows the state 1 (HIGH) or 0 (LOW) of the digital inputs.

Firmware

Shows the firmware version.

GPIOIn#

Shows the state 1 (HIGH) or 0 (LOW) of the GPIOs, independant of their configuration (input, output, ..).

Hardware

Shows the hardware version.

Model

Shows the model.



PiXtend V2 Software Manual

RetainDataIn

Shows the values of the RetainDataIn. The values of RetainDataIn are combined in one row, whereby the most left value represents Byte0 / RetainDataIn0. Each value is separated by an " " and thus can be parsed very easy in Perl:

```
my ($str) = ReadingsVal(pix, "RetainDataIn", "?")
if($str ne "?"){
    my @val = split(/ /, $str);    => $val[0] contains Byte0,
                                   $val[1] Byte1, etc.
    ...
}
```

Sensor#T/H

Shows the temperature (T) in °C and the humidity (H) in % of the sensor that is connected to the corresponding GPIO.

UCState

Returns the state of the *PiXtend V2*. If it is 1 everything is fine. If it is greater than 1 an error occurred or is present and the *PiXtend V2* can't be configured. More information can be found in chapter 14.1.2.4 and 14.2.2.4.

UCWarnings

Returns a value that represents the *PiXtend V2* warnings. For more information see chapter 14.1.2.4 and 14.2.2.4.



10.3.6. Attributes

The attribute name is case-sensitive.

PiXtend_GetFormat [text,value]

Changes the style in which the values of the Get-commands are returned. They can be returned as a message [text] or as a raw [value]. Default is the presentation as a message.

PiXtend_Parameter

With this attribute the base configuration (Set-commands with a leading "_") can be saved as an attribute. Attributes are stored in the config file. Single commands are separated with a space " " and each value is separated by a ":", e.g.:

```
attr pix PiXtend_Parameter _gpio0ctrl:dht11 _gpio3ctrl:dht22
```

10.4. Examples

The starting point for the following examples always is a defined *PiXtend V2 -S-* oder *PiXtend V2 -L-* module with the name "pix". The module is defined by this command:

```
define pix PiXtendV2
```

or

```
define pix PiXtendV2 L
```

10.4.1. Presentation of sensor values

In FHEM values can be displayed in a diagram (plot). This allows the graphical processing of sensor values which are connected to the *PiXtend V2*. In this example a DHT11-sensor is connected to GPIO0 and a DHT22-sensor is connected to GPIO3. To supply the sensors any 5 V output can be used.

To activate the sensor functionality the device "pix" has to be opened in FHEM. After that select the Set-command "*_GPIO0Ctrl*" and on the second field "*DHT11*" and confirm the setting by pressing the set-button. Alternatively the following command can be written directly to the command line (see Fig. 98):

```
set pix _gpio0ctrl dht11
```




PiXtend V2 Software Manual

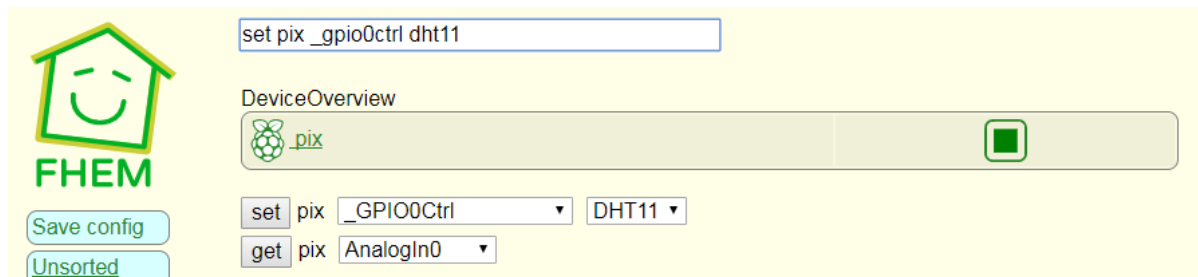


Figure 98: FHEM - Activation of the sensor functionality

Repeat this step for the DHT22-sensor and GPIO3.

Optionally this configuration (and other settings) can be loaded automatically on start-up of FHEM. Therefore the commands have to be set via the attribute “PiXtend_Parameter”, e.g.:

```
attr pix PiXtend_Parameter _gpio0ctrl:dht11 _gpio3ctrl:dht22
```

The value of a command is separated by a colon (':') and multiple commands can be separated by a space. With this method all set-commands with a leading underline can be loaded on start-up of FHEM.

After the configuration the measured values of the sensors can be found in the Readings-section of the device “pix” in the fields “Sensor0” and “Sensor3”.

To display the measured values the following tutorial was used:

https://wiki.fhem.de/wiki/Buderus_Web_Gateway#Mit_FileLog

For the first step a LogFile is created to which the measured values are stored:

```
define pixlog FileLog ./log/pix-%Y-%m.log pix:Sensor0H:.*|  
pix:Sensor0T:.*|pix:Sensor3H:.*|pix:Sensor3T:.*
```

The second step creates an SVG plot out of this LogFile by clicking the appropriate button (see Fig. 99).



PiXtend V2 Software Manual

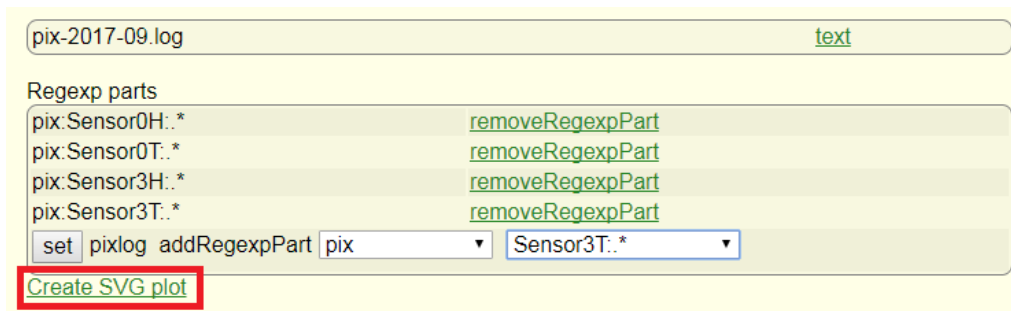


Figure 99: FHEM - Create Plot via LogFile

Now the plot can be configured, e.g. like the one in Fig. 100. After the configuration the button “write .gplot file” has to be pressed to create the plot.

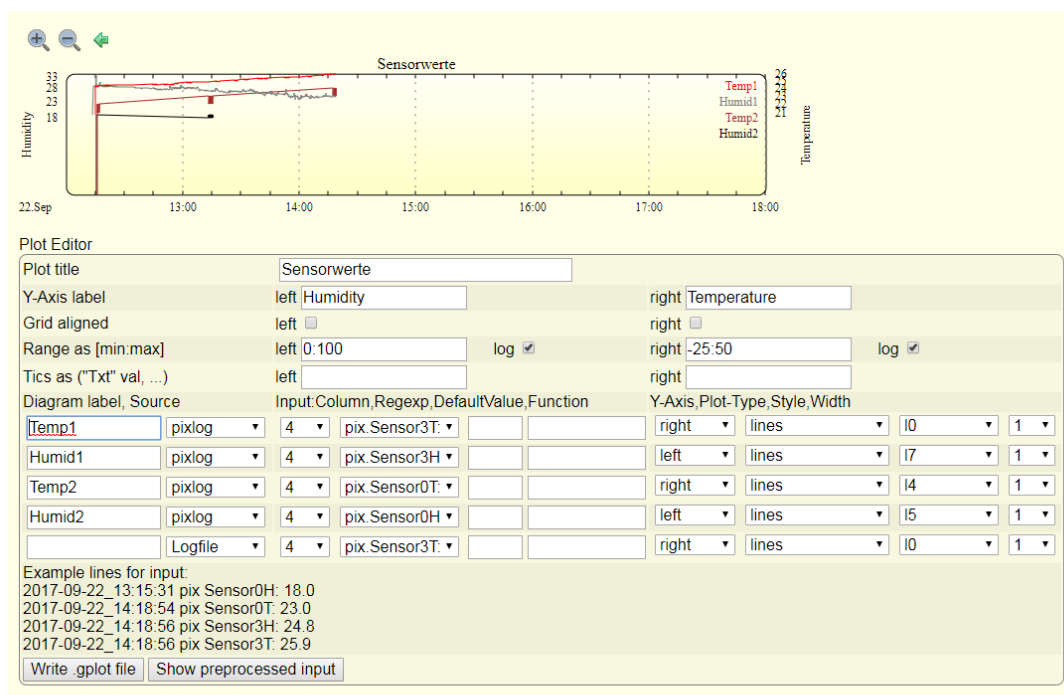


Figure 100: FHEM - Plot for the sensor values



The DHT11-sensor (Temp2 & Humid2) only has an accuracy of 1°C or 1%. As a result it may seem like the value is bouncing. Furthermore a new measurement point is only created if a new value is present, respectively the Reading has changed.



10.4.2. Debouncing and combining of in- and outputs

For the first step a digital input (DI0) should be debounced and turn on a relay (Relay0) afterwards. For debouncing the input the following command is used. Thereby the digital inputs DI0 and DI1 are debounced by $10 \times 100 \text{ ms} = 1 \text{ s}$:

```
set pix digitaldebounce01 10
```

To turn on a relay the “notify” dummy of FHEM can be used:

```
define n_relay notify pix:DigitalIn0:.* set pix RelayOut0  
$EVTPART1
```

The relay is set to the same state (\$EVTPART1) that DigitalIn0 has. For the ideal case the relay reacts 1 s delayed to the digital input.

In the next step a turn-off delay should be implemented using the same debounce function. In this case digital input DI2 and Relay1 are used. First of all the digital input is debounced again, but this time only 100 ms.

```
set pix digitaldebounce23 1
```

After that a “DOIF” module with the name “TimedSW” is defined:

```
define TimedSW DOIF ([pix:DigitalIn2:"(on)"]) (set pix relayout1  
on,set pix digitaldebounce23 50) DOELSE (set pix relayout1 off,set  
pix digitaldebounce23 1)
```

If the condition is true, that DigitalIn2 is “on” than Relay1 is turned “on” as well and the debounce value for DI2 is set to 50 (= 5 s). If DigitalIn2 is “off” the relay is turned off and the debounce value is reset (0,1 s). As a result the relay turns on nearly immediately with the digital input but is turned off 5 s delayed.

For the third step two conditions should be combined. Therefore Relay2 should be turned on only if the voltage of the analog input AI0 is higher than 5 V and digital Input DI4 is active (“on”). Again a “DOIF” module is used for this example:

```
define Condi DOIF ([pix:DigitalIn4:"(on)"] and [pix:AnalogIn0] >  
5.0) (set pix relayout2 on) DOELSE (set pix relayout2 off)
```



10.4.3. Control servo motors

This example is intended to illustrate the control of a servo motor in FHEM. By default, the servo mode is active in the PWM settings and therefore does not have to be configured. However, the outputs of the servo channels must be activated so that a signal comes out of them. According to the description in chapter 14 for the configuration of the PWM channels, the value "24" must be written in PWMxCtrl0. For example:

```
set pix pwm0ctrl0 24
```

to enable the two PWM channels PWM0A and PWM0B. The servo motor can now be controlled via the fields PWM0A and PWM0B, whereby it should be noted that the value 0 corresponds to the minimum limit value and the value 16000 corresponds to the maximum decisive, e.g.:

```
set pix pwm0a 9000
```

As a default value 0 is used for PWMxA/B in servo mode. If a different value should be used before activating the channels the two commands can be swapped in its order.



10.5. Frequently Asked Questions (FAQ)

The Readings for sensor values shows „Sensor not connected“ or „Function not enabled“. Why?

In this case the function to read out the sensor values wasn't enabled for the corresponding GPIO channel (“_GPIOxCtrl”) or this setting was done but the sensor isn't properly connected. Enable the sensor functionality or check the cabling of the sensors.

The “ioctl.ph is not available but needed” error appears during the device's creation (define). What should be done?

The *PiXtend V2* module for FHEM requires ioctl.ph to access the SPI, but this file wasn't found. There is probably an error in the installation of Raspbian. We recommend the usage of the images we have tested. These images can be found in the [download section](#) of our homepage.

During the definition of the device the error "Error! Device not xxx. Please change access rights for fhem user ..." occurs. What's the problem?

The user “fhem” needs the permission to access the SPI and the GPIOs of the Raspberry Pi. Grant these permissions to the user by adding the user to the following groups:

```
sudo adduser fhem gpio; sudo adduser fhem spi; sudo reboot
```

The value of “STATE” is “error” and doesn't change. What can I do?

The communication between *PiXtend V2* and the Raspberry Pi is faulty. Check if these two devices are connected properly, the switches on the *PiXtend V2* are set to the correct position, the *PiXtend V2* isn't in Safe State and SPI is enabled on the Raspberry Pi.



11. PiXtend Python Library V2 (PPLV2)

The programming language Python is described as a universal and interpretable language. It should have a concise and readable form and facilitate the programming in this way. A major feature of Python is, for example, that the program code is structured by indenting the code⁵.



Figure 101: Python - Source: <https://www.python.org/>,
The Python Brochure

The enthusiasm and spread of the language Python is not least due to a very large library basis, there is something for almost any application. Also, the very large community that grew around Python makes this language very popular.

Furthermore, Python is freely available to everyone and can thus be used at vocational training, at university, at home, but also commercially. The programming language can be used by

beginners as well as by advanced programmers to create computer programs for different systems.

Therefore, it is not surprising that the Raspberry Pi Foundation decided to integrate Python into their Raspbian image⁶.

Immediately after the first start, the Python programming on the Raspberry Pi can commence immediately, everything is already preinstalled. Access to, for example, the on-board GPIOs or the SPI bus functions "Out of the Box".

The PPLV2 is open source and can be used, changed and expanded by everyone.

Further information, tips and tricks can also be found in our support forum at:

<https://www.pixtend.de/forum/>

The latest versions of all documents and software components can be found in our download section on our homepage: <https://www.pixtend.de/downloads/>

⁵ Source Wikipedia April 2017: [https://de.wikipedia.org/wiki/Python_\(Programmiersprache\)](https://de.wikipedia.org/wiki/Python_(Programmiersprache))

⁶ <https://www.raspberrypi.org/documentation/usage/python/> and <https://www.raspberrypi.org/downloads/raspbian/>



11.1. Prerequisites

The PPLV2 driver support refers to the *PiXtend V2* for Python 2.7.9 and later (not Python 3).

We recommend one of these Raspberry Pi models: B+, 2 B, 3 B, 3 B+.

Download the *PiXtend V2 Basic Image* SD card from our download area and use it as a starting point for your projects. Alternatively, you can use an original Raspbian Stretch Image.

On the Raspberry Pi you can access either a directly connected keyboard and monitor or via SSH (TeraTerm / Putty) from a PC.

If you are using an original Raspbian Image directly, the Raspberry Pi needs an active Internet connection so you can follow the steps in this chapter.

We recommend to read the corresponding chapters for Process Data and Control- & Status-Bytes, suited for your *PiXtend V2*, found in the Appendix. These contain information on the configuration of the PWM outputs and the analog inputs, as well as useful information about the GPIOs, the digital inputs and outputs, and the relays on *PiXtend V2*.

For information about SSH clients, the Putty program, WinSCP, or a suitable programming tool, download the PiXtend V1.3 App-Note for Python, [APP-PX-401 PiXtend Python Library](#). Many sections of this App-Note also provide useful hints for working with the PPLV2.



11.2. Installation with PiXtend V2 -S- Image

In the PiXtend V2 Basic Image, the PiXtend Python Library V2 is already preinstalled and can be used immediately. See Chapter 6.4 *Preparing a SD card* for more information on how to create an SD card for the Raspberry Pi.

On the Raspberry Pi you will find all files in the folder `/home/pi/pplv2/`.

You can start programming right away.

11.3. Installation on original Raspbian

11.3.1. Preparation

We start with an original Raspbian Stretch Image (Release: 07.09.17). With this image, the PIXEL interface automatically starts after the first boot. Since we do not need it here, we disable it. A monitor, keyboard and mouse are required for subsequent steps. Logging via Ethernet using SSH does not work, because the SSH server is disabled by default.

Click on the terminal icon in the PIXEL interface's menu bar and open a console window. Maximize the window for best visibility.



Figure 102: Python - Raspbian PIXEL - Terminal icon found in the Taskbar

Now we can open the configuration on the Raspberry Pi:

```
sudo raspi-config
```

The configuration program for the Raspberry Pi is executed. In under "3 Boot Options" menu item, select "B1 Desktop / CLI" and then in the submenu "B2 Console Autologin". The selection is as follows:

„3 Boot Options“ → „B1 Desktop / CLI“ → „B2 Console Autologin“

At this point we can also activate the SPI bus. This is also done in the raspi-config configuration program:

„5 Interfacing Options“ → „P4 SPI“ → <Yes> → <Ok>



PiXtend V2 Software Manual

So that we can later access the Raspberry Pi via network using SSH, for example, to run our own Python programs, the SSH server has to be activated at this point:

„5 Interfacing Options“ → „P2 SSH“ → <Yes> → <Ok>

Now we can leave the configuration program, simply press the Tab key twice, until the word <Finish> is highlighted in red and then press the Enter key.

After these changes, a reboot must be carried out, should raspi-config not automatically offer this when exiting the program, enter the following command:

```
sudo reboot
```

Now we can download and install the required components in the next steps. To download the necessary files from the Internet, the Raspberry Pi must have an active Internet connection.

11.3.2. PiXtend Python Library V2 installation

For the installation of the PPLV2, we first create a directory for the PPLV2 package, then download the package and unpack it into the created directory. The last step is to install the PPLV2 package. Once this is done, the PiXtend Python Library V2 can be used globally in all Python 2.7.9 programs.

Installing a Python package can cause many text outputs on the console, which is completely normal.

Carry out the following steps in sequence:

```
mkdir pplv2
wget https://www.pixtend.de/files/downloads/pplv2_v0.1.3.zip
unzip pplv2_v0.1.3.zip -d ./pplv2/
cd pplv2
sudo python setup.py install
```

If the last command was successfully executed, the PiXtend Python Library V2 is now installed and can be used in your own Python programs.

The installation of the PPLV2 package can be checked with pip freeze. The list of installed packages now contains the entry `pixtendliv2 == 0.1.3`.



PiXtend V2 Software Manual

11.4. Programming

In the first step we use a sample program, which is already on the Raspberry Pi. It is part of the PiXtend Python Library V2 and is located in the folder “/home/pi/pplv2/examples”.

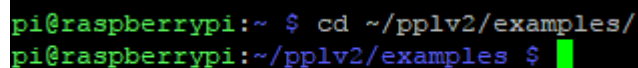
Those who work directly at Raspberry Pi can get started right away. Everyone who wants to connect to the Raspberry Pi via a network must do so first. One SSH client program which is often used by us is the program “Putty”.

11.4.1. Example Folder

After logging in, we are in the pi user’s home directory (/home/pi/). Here is already the folder pplv2, which was created at the beginning or already exists on our SD card and contains the source code of the PiXtend Python Library V2 and the subfolder examples with the examples.

We change to the directory with the examples:

```
cd ~/pplv2/examples/
```



```
pi@raspberrypi:~ $ cd ~/pplv2/examples/  
pi@raspberrypi:~/pplv2/examples $
```

Figure 103: Python - Console command - PPLV2 folder

11.4.2. Execute Example

To start with the *PiXtend V2* board, we are using a *PiXtend V2 -S-* as an example, we launch the Python PiXtend V2 demo program. All digital inputs are read and all digital outputs are switched alternately. The clacking of the relays is easy to hear and the LEDs indicate the condition.

The following command starts the example:

```
sudo python pixtendv2s_demo.py
```

If you have a *PiXtend V2 -L-* try the following:

```
sudo python pixtendv2l_demo.py
```



Figure 104: Python - PiXtend Python Library V2 Demo Program

The key combination Ctrl + C can be used to terminate the Python program.



PiXtend V2 Software Manual

11.4.3. Create your own program

Open an editor or program that allows you to write or edit Python programs and create a new empty Python file. Save this file, e.g. under the name "programm1.py".

We use a *PiXtend V2 -S-* as an example here, but the same functions are also available on the *PiXtend V2 -L-*. Replace the "S" with an "L".

In our first program, only relay 0 should be cycled every second, that's all.

The program could look like this:

```
#!/usr/bin/env python
```

```
from pxtendv2s import PiXtendV2S
```

```
import time
```

```
p = PiXtendV2S()
```

```
while True:
```

```
    if p.relay0 == p.OFF:
```

```
        p.relay0 = p.ON
```

```
    else:
```

```
        p.relay0 = p.OFF
```

```
    time.sleep(1)
```

```
1  #!/usr/bin/env python
2
3  from pxtendv2s import PiXtendV2S
4  import time
5
6  p = PiXtendV2S()
7
8  while True:
9      if p.relay0 == p.OFF:
10         p.relay0 = p.ON
11      else:
12         p.relay0 = p.OFF
13         time.sleep(1)
14
```

Figure 105: Python - Short Demo Program

Program explanation:

- In the first line we announce that it is a Python program, the program runs later on the Raspberry Pi.
- After that the *PiXtendV2S* class has to be imported, so we get access to the features and functions of *PiXtend V2 -S-* and a communication with the microcontroller and DAC is possible.
- To wait at the end of the program, we also import the time class.
- With `p = PiXtendV2S ()` an object is created with which we can work with, the communication with the microcontroller is automatically executed in the background.
- In a while loop we check relay 0 whether it is On or Off and respond accordingly.
- The constants ON and OFF can be replaced by True and False. Working with



PiXtend V2 Software Manual

names is intended help to clarify the actual procedure.

- At the end, we wait a second to reduce the strain on relay 0.

Transfer the Python program to the Raspberry Pi and run it with the following call:

```
sudo python programml.py
```

After starting, the relay 0 starts to turn on and off, as programmed by us and this happens now every second.

To end the program, press the key combination Ctrl + C 2 times in succession. At the first press, the while loop is exited, the second press terminates the background communication with the *PiXtend V2* board. In your own programs you can call the "close" function of the *PiXtendV2S* class before exiting the program, then the communication with the microcontroller is terminated properly and the SPI driver is closed. Furthermore, the asynchronous communication stops.

11.4.4. Short Overview

The PPLV2 consists of multiple Python classes, the *PiXtendV2Core*, *PiXtendV2S* and *PiXtendV2L*. The *PiXtendV2Core* class represents a basic class, which contains many basic functions and properties, but is not executable by itself.

The *PiXtendV2S* / *PiXtendV2L* classes inherit these properties and functions and forms all the remaining properties of *PiXtend V2*.

After the instantiation of the *PiXtendV2S* / *PiXtendV2L* class, the communication with the microcontroller on the *PiXtend V2* board is automatically executed in the background. The user can easily use all digital and analog inputs and outputs without having to worry about the communication. An explicit wait after each cycle is not required.

In the zip file (pplv2_v0.1.x.zip) or in the folder *pplv2* on the Raspberry Pi, the subfolder "doc" contains two HTML files that contain the API documentation for the named Python classes. In particular the help file for the *PiXtendV2S* / *PiXtendV2L* classes shows all usable functions and properties. If you want to use the GPIOs, the temperature and humidity inputs or the PWM outputs, for example, look here.



PiXtend V2 Software Manual

The following table provides a brief overview of the most important and common features and properties of the Python class *PiXtendV2S*⁷:

Name	Type	Description
		Instantiation of the PiXtendV2S class and start of communication with the <i>PiXtend V2</i> microcontroller. Example: <code>p = PiXtendV2S ()</code> Instantiation of the PiXtendV2L class and start of communication with the <i>PiXtend V2</i> microcontroller. Example: <code>p = PiXtendV2L ()</code>
close	Function	If the Python program is to be terminated, it is recommended to call the close function and then delete the PiXtendV2S instance. The close function resets all internal variables and objects, closes the SPI driver and terminates the background communication. Example: <code>p.close ()</code> <code>p = None</code>
digital_in0 .. 7	Property (r)	With these 8 read only properties (digital_in0 to digital_in7), the states of the digital inputs can be read by the <i>PiXtend V2 -S-</i> . The properties provide either False for value (OFF) or True for value (ON).
digital_out0 .. 3	Property (rw)	The 4 digital outputs can be read or written via the properties digital_out0 to digital_out3. If the property is assigned the value False, the corresponding digital output goes off (OFF) or when the value True is set the output turns on (ON).
relay0 .. 3	Property (rw)	The 4 relays can be read or written to via the relay0 to relay3 properties. If the property is assigned the value False, the corresponding relay is switched off (OFF) or when the value True is set, it turns on (ON).
hardware	Property (r)	The hardware version can be read via this read-only property. The value 21 is, for example, Board version 2.1.
firmware	Property (r)	The firmware version of the microcontroller can be determined via this read-only property.
ON	Constant	Corresponds to the value True.
OFF	Constant	Corresponds to the value False.

⁷ Meaning of (r) and (rw) behind a property: r = read only and rw = read and write
Please find the functions and properties for the *PiXtend V2 -L-* in the corresponding API documentation in the *doc* folder.



11.4.5. Start Python automatically at boot

After a reboot / power-up of the Raspberry Pi, a Python program does not start automatically. As already mentioned, a Python program can be started with the following command: replace the myprogram.py with your own programs name:

```
sudo python myprogram.py
```

We can also run this command automatically when the Linux system is booted.

The change is done quickly in the Linux console:

```
sudo nano /etc/rc.local
```

The file "rc.local" opens with some content. We add two new lines before the line "exit 0" with the assumption that the program to be started is located in the home directory:

```
cd /home/pi/  
sudo python myprogram.py &
```

The "&" is not a typo. This starts the Python program as a process in the background.

Save and exit with CTRL + X → Reboot



WARNING: Please test the program beforehand very carefully! If the program has an error, e.g. that the digital outputs are activated in an uncontrolled manner or very fast, this may damage the connected peripherals, and because the faulty program is set up as an "Autostart program", it does not help to restart the Raspberry Pi to solve the problem. Always make backup copies of your work.



11.4.6. Using the Serial Interface

The use of the serial interface is not included in the PPLv2 package. This topic has more to do with the Raspberry Pi and Python than with the *PiXtend* V2. For the Raspbian version mentioned in this documentation, the serial interface runs, for example, via the Linux device `/dev/ttyS0`. Use the Python module [pySerial](#) to use the serial interface in your own programs.

Please note that the output of the Linux console is output here, so you must edit the `/boot/cmdline.txt` file beforehand so you use it, the line `console = serial0,115200` has to be deleted from the first line or use the `raspi-config` to disable the console output.

11.4.7. Frequently Asked Questions (FAQ)

I can not start my program, Python gives an error message about an indentation?

There is probably a problem with the indentation of the different lines of code in the program. Once again look carefully that the indentation is correct for every line. We use e.g. always 4 spaces per indentation level, only one space is missing, or spaces and tabs have been mixed, Python issues a failure message. In Notepad ++ e.g. you can see the "non-printable" characters and you see very quickly where something is wrong, in addition, Python says in which line it has detected the error.

Can I run my program without sudo?

This seems to be actually possible with Python in this release of the Raspbian OS, shown in this document and also used for our SD images, but we still recommend using *sudo* so that Python can also execute system functions. If *sudo* is not used, in this case, problems can occur and the cause of which may not be easy to explain or to find.



12. OpenPLC Project

The [OpenPLC Project](#) has set itself the goal of being a completely free and standard-compliant⁸ software basis for programmable logic controllers (PLC). We find the realization of the project to be very successful and fitting to the concept of PiXtend.

Especially for vocational training / schools and use in technical study we see clear advantages in contrast to the "professional tools" because of the simplicity of the programming. That's why we worked together with the project creator to develop a driver to support the *PiXtend V2* series.

**DESIGNED TO BE
OPEN SOURCE**

The OpenPLC is the first fully functional standardized open source PLC. We believe that opening the black-box of a PLC will create opportunities for people to study its concepts, create new technologies and share resources.

Figure 106: OpenPLC - Designed to be Open Source - www.OpenPLCProject.com

The following chapters show the installation and setup on a separate SD card or who wants to start and test OpenPLC right away, can use the [PiXtend Basic Image V2.x](#) in which OpenPLC is already installed.

An overview of the usable IOs and functions can be found in the Chapter 12.4 Further Information.

You can find more information, tips and tricks in our support forum at:
<https://www.pixtend.de/forum/>

The latest versions of all documents and software components can be found in the download area of our homepage: <https://www.pixtend.de/downloads/>

8 Programming languages as defined by IEC/EN 61131-3: ST, IL, LD, FBD, SFC



12.1. Prerequisites

For the use of OpenPLC on PiXtend we recommend one of these Raspberry Pi models: B+, 2 B, 3 B, 3 B+.

If you start with a new project or would like to test OpenPLC, then download the *PiXtend Basic Image V2.x* SD card image from our [download area](#) and use it for testing or as a starting point for your own projects. Alternatively, you can also use an original Raspbian *Stretch* Image.

The Raspberry Pi you can access either directly using a connected keyboard and monitor or via SSH (TeraTerm / Putty) from a PC.

If you are using an original Raspbian Image directly, the Raspberry Pi needs an active Internet connection so you can follow the steps in this chapter.

We recommend to read the chapters Process Data and Control- & Status-Bytes for your exact PiXtend V2 device found in the Appendix. These contain information on the configuration of the PWM outputs and the analog inputs, as well as useful information about the GPIOs, the digital inputs and outputs, and the relays on *PiXtend V2*.



12.2. Installation

12.2.1. Preparation

Apart from this chapter, the installation of the required software is also explained on the homepage of the *OpenPLC Project*:

<http://www.openplcproject.com/getting-started> → Runtime → PiXtend

You can follow the steps on the mentioned website or in this chapter as you like.

If you use our SD card image *PiXtend Basic V2.x*, you can jump directly to chapter 12.3 Programming and start writing PLC programs.

We start with an original *Raspbian Stretch Image*. With this image, the new PIXEL interface starts automatically after the first boot. Since we do not need it, we disable it.

To do this, we access the Raspberry Pi via SSH:

```
sudo raspi-config
```

The configuration program for the RPi is executed. In the menu item "*3 Boot Options*" we select "*B1 Desktop / CLI*" and then menu item "*B1 Console*" or "*B2 Console Autologin*".

On this occasion, we can also activate the SPI bus right away. This is also done in the configuration program *raspi-config*:

„*5 Interfacing Options*“ → „*P4 SPI*“ → <Yes> → <Ok>

After this change, a reboot is necessary:

```
sudo reboot
```

Now we can download and install the required components in the next steps.



12.2.2. OpenPLC Runtime Installation

The *OpenPLC Project* relies on *wiringPi*⁹ (as do our *pxdev* Linux Tools) to access the interfaces and GPIOs of the Raspberry Pi. Let's start with the installation. It is assumed that you are in the home directory of the user Pi (/home/pi):

```
sudo apt-get update
sudo apt-get install git-core git
git clone git://git.drogon.net/wiringPi
cd wiringPi
./build
```

Next we need some Debian packages, which can be installed very comfortably via apt-get:

```
sudo apt-get install build-essential pkg-config bison flex
sudo apt-get install autoconf automake libtool make nodejs
```

Notice:



The basic installation of OpenPLC_v3 can take up to *1 hour* depending on the Raspberry Pi model.

Also make sure that there is enough space on the SD card to install the needed update and compile OpenPLC_v3.

Now we download the latest version of *OpenPLC* from *github* and compile it:

```
cd /home/pi
git clone https://github.com/thiagoralves/OpenPLC_v3.git
cd OpenPLC_v3
sudo ./install.sh rpi
```

9 WiringPi – OpenSource GPIO/SPI/I²C/Serial library for the Raspberry Pi – www.wiringpi.com



PiXtend V2 Software Manual

```
[FINALIZING]
Activating Blank driver
Setting Platform
Optimizing ST program...
Generating C files...
POUS.c
POUS.h
LOCATED_VARIABLES.h
VARIABLES.csv
Config0.c
Config0.h
Res0.c
Moving Files...
Compiling for Linux
Generating object files...
Generating glueVars...
Compiling main program...
Compilation finished successfully!
pi@raspberrypi:~/OpenPLC_v3 $
```

Figure 107: OpenPLC -
OpenPLC_v3 installation was
successful

For all those who know OpenPLC_v2, the selection of the hardware (now also called PIN mapping) now takes place in OpenPLC itself in the web interface. There is an own configuration point, called "Hardware" now.

Almost there! Now we just have to start the runtime, this is done with the following command:

```
sudo ./start_openplc.sh
```

After starting the OpenPLC_v3 Runtime, this is available on the Raspberry Pi under port "8080".

You can reach the web server with a browser via the following web address:

<http://ip-raspberry-pi:8080/>

```
pi@raspberrypi:~/OpenPLC_v3 $ sudo ./start_openplc.sh
* Serving Flask app "webserver" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
```

Figure 108: OpenPLC - OpenPLC_v3 runtime start



12.2.3. Programming Enviroment (IDE)

The programming environment "*PLCOpen Editor*" is free and available for Windows, Linux and MacOS. We use the version for Windows which is version 1.2.

Download the latest version here:

<http://www.openplcproject.com/plcopen-editor>



Figure 109: OpenPLC Editor download

After unpacking the ZIP archive, you can open the editor directly using the enclosed link. The programming environment starts:

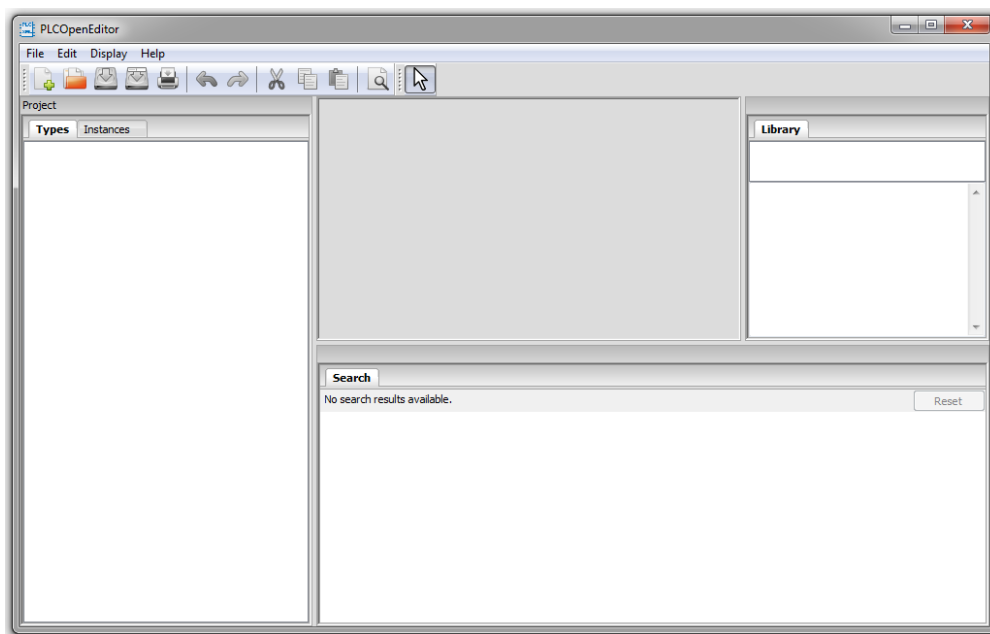


Figure 110: OpenPLC editor

Now all software components have been prepared and installed, we can continue with the next chapter, let's do some programming!



12.3. Programming

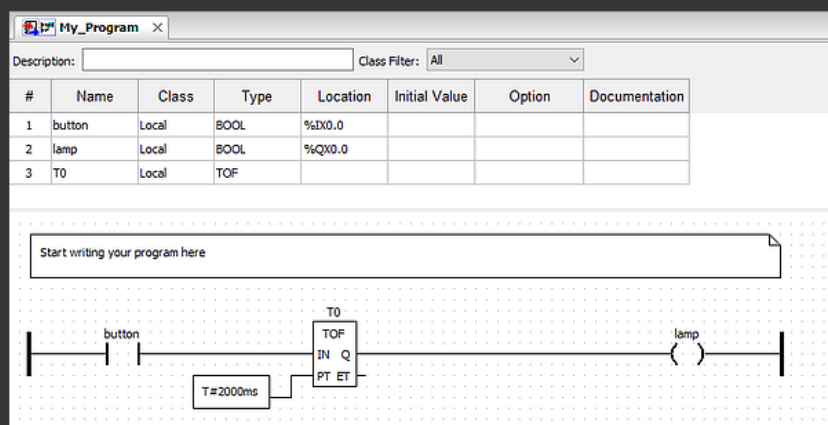
In the first step, we use a test program that can be downloaded from the homepage of the OpenPLC Project:

<http://www.openplcproject.com/getting-started>

- Runtime
- PiXtend
- scroll to the bottom to where the „Hello World“ programm can be downloaded

Running the first project

This is a simple hello world project for the OpenPLC. The button is attached to %IX0.0 and the lamp is attached to %QX0.0. When the button is pressed and released, the lamp remains on for 2 seconds and then turns off.



Download the project below and open it on the PLCopen Editor. Click on **File -> Generate Program** in order to compile the diagram into a ST file. Then upload the ST file to the OpenPLC using the web interface.



Hello World

Figure 111: OpenPLC - Download Hello World program

The program or project files in the OpenPLC Editor are saved as XML files.



12.3.1. Hello World Program

Start the PLCOpen Editor and select under "File" → "Open" the previously downloaded test program "Hello_World.xml".

The example program "My_Program" appears in the project tree under "Blank Project" (Programs):

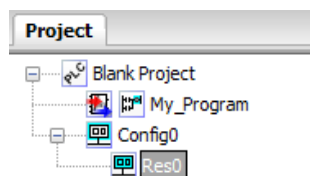


Figure 112: OpenPLC - Project overview

Double-click on "My_Program" to open the editor.

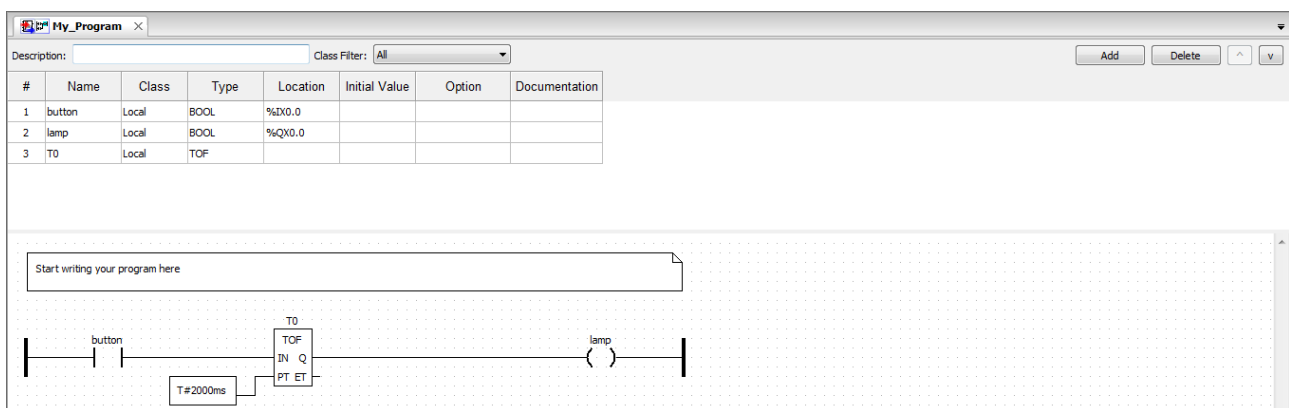


Figure 113: OpenPLC - Hello World program

The small program is programmed in Ladder Diagram (LD).

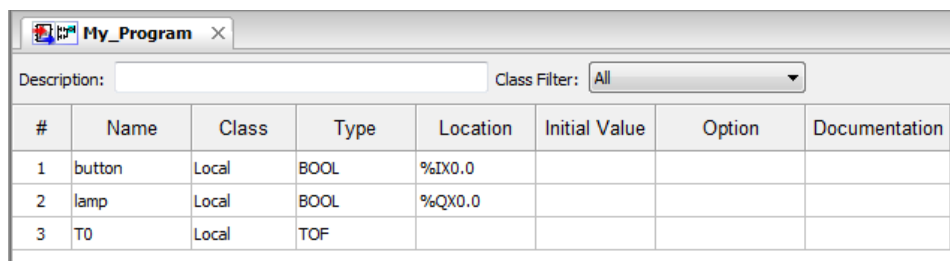
A button / switch, here represented with "button", is connected to the input of a TOF block, this block implements a switch-off delay, in our case of 2000 ms (2 s).

When the pushbutton / switch "button" is pressed, this immediately activates the output "lamp" at the output of the TOF block and is held for another 2 seconds after releasing "button", after which the output "lamp" goes off.



PiXtend V2 Software Manual

Let's look at the variables list:



#	Name	Class	Type	Location	Initial Value	Option	Documentation
1	button	Local	BOOL	%IX0.0			
2	lamp	Local	BOOL	%QX0.0			
3	T0	Local	TOF				

Figure 114: OpenPLC - Variables list

In the column "*Name*" is the variable name. The type is a BOOL for "*button*" and "*lamp*", ie 1 bit each. The bits are linked via the "*Location*" column with a hardware address. The button / switch "*button*" is located here at the address %IX0.0, which is the digital input 0 (DigitalIn0) for PiXtend V2 -S-. The output "*lamp*" is located at the address %QX0.0 - ie the digital output 0 (DigitalOut0).

More about the hardware addresses can be found in chapter 12.4 Further Information. Of course, variables that are not connected to a hardware input / output can also be defined.

12.3.2. First start and Transfer program to PiXtend

12.3.2.1 Generate program

Under "*File*" → "*Generate Program*", the program can be converted into a file, which we can load onto the controller (OpenPLC Runtime). If no error is found during program creation, the editor reports success:

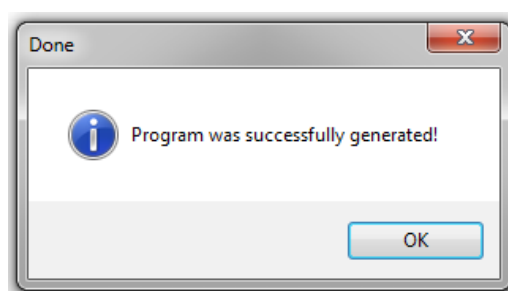


Figure 115: OpenPLC - Success notice



PiXtend V2 Software Manual

12.3.2.2 Start Runtime

In order to transfer the just created program to the controller, the *OpenPLC* Runtime has to be started first.

This can be done directly at the RPi or via SSH with the following commands:

```
cd ~/OpenPLC_v3/  
sudo ./start_openplc.sh
```

12.3.2.3 First Login

If the runtime has started, we can open a browser on our Windows computer. The *OpenPLC* Runtime automatically launches a web server that we can now access. It is assumed that you know the IP address of the Raspberry Pi.

In our case this is 192.168.1.27. The web server is running on port 8080:



Figure 116: OpenPLC - Browser address bar

Replace the IP address with the address of your Raspberry Pi. The *OpenPLC* server's login page opens:

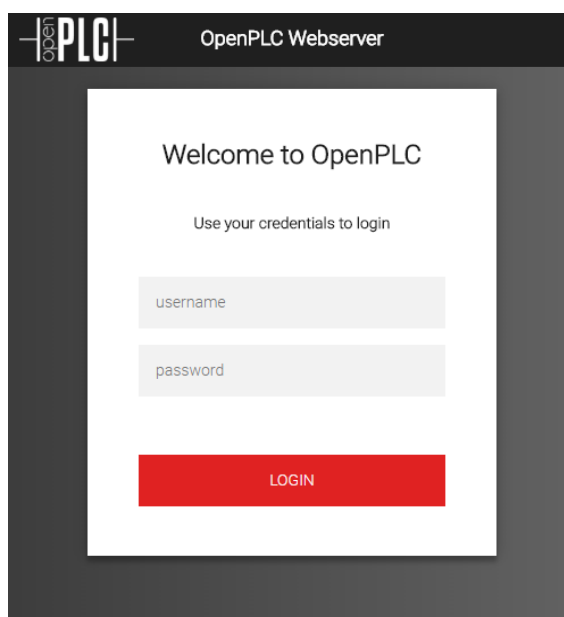


Figure 117: OpenPLC - Runtime web server



PiXtend V2 Software Manual

The default username and password are:

username: openplc

password: openplc

It is recommended after the first login, first to change the password of the "openplc" user. This is easily done via the menu on the left by clicking on the menu item "Users".

The screenshot shows the 'Edit User' interface in the OpenPLC software. On the left is a dark sidebar with a menu: Dashboard, Programs, Slave Devices, Monitoring, Hardware, Users (highlighted in red), Settings, and Logout. Below the menu, it says 'Status: Stopped' and has a 'Start PLC' button. The main content area is titled 'Edit User' and contains several input fields: 'Name' (with 'OpenPLC User'), 'Username' (with 'openplc'), 'Email' (with 'openplc@openplc.com'), 'Password' (masked with dots), and 'Picture' (with a 'Choose File' button and 'No file chosen' text). At the bottom of the form are two red buttons: 'Save changes' and 'Delete user'. The top of the interface has a status bar with 'Stopped: Blank Program' and 'OpenPLC User'.

Figure 118: OpenPLC - Change user data

Modify all information as desired and then save your changes by clicking on the "Save changes" button.

If you are then prompted to log in again, then do so simply with the newly/self assigned login credentials.

In the next step we still have to select our hardware, OpenPLC now calls this "PIN Mapping".



PiXtend V2 Software Manual

12.3.2.4 Choose Hardware (PIN Mapping)

Next, the PIN mapping, the existing hardware, must be set. At delivery, an empty driver (blank driver) is installed in OpenPLC.

In the "Hardware" menu, the hardware used by OpenPLC can be easily changed by selecting it from a list. The list displays all supported hardware platforms and devices.

Around the middle of the list, we see that *OpenPLC_v3* supports all available PiXtend devices. These are currently:

- PiXtend V1.3 (pixtend)
- PiXtend V2 -S- (pixtend 2s)
- PiXtend V2 -L- (pixtend 2l)

Select your PiXtend from the hardware list and click on the button "Save Changes".

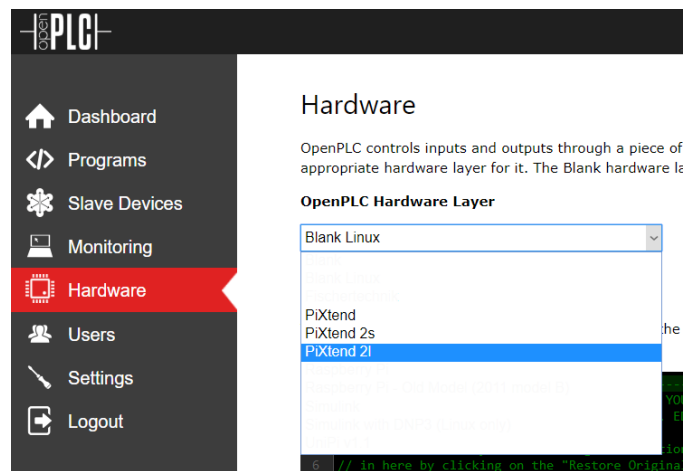


Figure 119: OpenPLC - Hardware selection

Afterwards OpenPLC starts immediately with the compilation of the currently set program, with a new installation or this is the first start of OpenPLC, the program is always called "Blank Program".

We have to wait until OpenPLC has checked and compiled everything.

Now we can finally transfer our own program to the controller and run the "Hello World" program.

With a click on the button "Go to Dashboard" we get back to the main page.

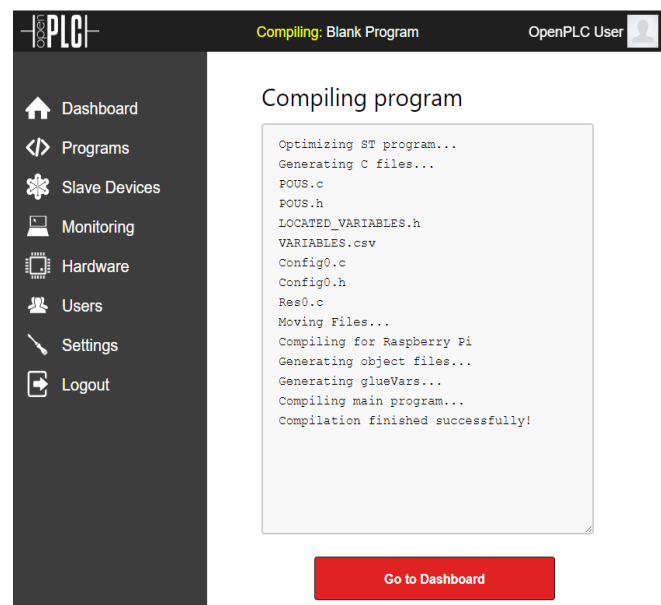


Figure 120: OpenPLC - Compiling Blank Program



12.3.2.5 Transfer Program

To be able to transfer a program to the controller, we need to switch to the "Programs" menu. We can already see that there is already a program available, the "Blank Program", which we have already encountered as an active program in PIN mapping.

With the button "Choose file" we now select the program file created by us with the extension ".st" and load it with the help of the button "Upload Program" on to the controller.

The screenshot shows the OpenPLC web interface. The top bar displays the 'openPLC' logo, the status 'Stopped: Blank Program', and the user 'OpenPLC User'. The left sidebar contains navigation links: Dashboard, Programs (highlighted), Slave Devices, Monitoring, Hardware, Users, Settings, and Logout. The main content area is titled 'Programs' and includes a description: 'Here you can upload a new program to OpenPLC or revert back to a previous uploaded program shown on the table.' Below this is a table with the following data:

Program Name	File	Date Uploaded
Blank Program	blank_program.st	May 24, 2018 - 08:02PM

Below the table is a link: [List all programs](#). The 'Upload Program' section contains a 'Choose File' button, the text 'No file chosen', and an 'Upload Program' button. At the bottom of the sidebar, the status 'Status: Stopped' is shown above a red 'Start PLC' button.

Figure 121: OpenPLC - Programs menu



PiXtend V2 Software Manual

The view changes and OpenPLC asks us for details about the program we just selected. We need to give our program at least a name so that we can finish the process and get the program into the OpenPLC program database. The description can optionally be filled out, this is mainly a help later, if many programs have been transferred to the controller or you have created many different versions of the same program, here entering a program description can be very helpful.

By clicking on the "Upload program" button, our program will be entered in the OpenPLC program database and will be translated immediately.

OpenPLC

Stopped: Blank Program

OpenPLC User

Dashboard

Programs

Slave Devices

Monitoring

Hardware

Users

Settings

Logout

Status: Stopped

Start PLC

Program Info

Name

Hello World

Description

Hello World test program

File

377823.st

Date Uploaded

Aug 03, 2018 - 09:40AM

Upload program

Figure 122: OpenPLC - Enter Program Info



PiXtend V2 Software Manual

The web interface should show us the following at the end:

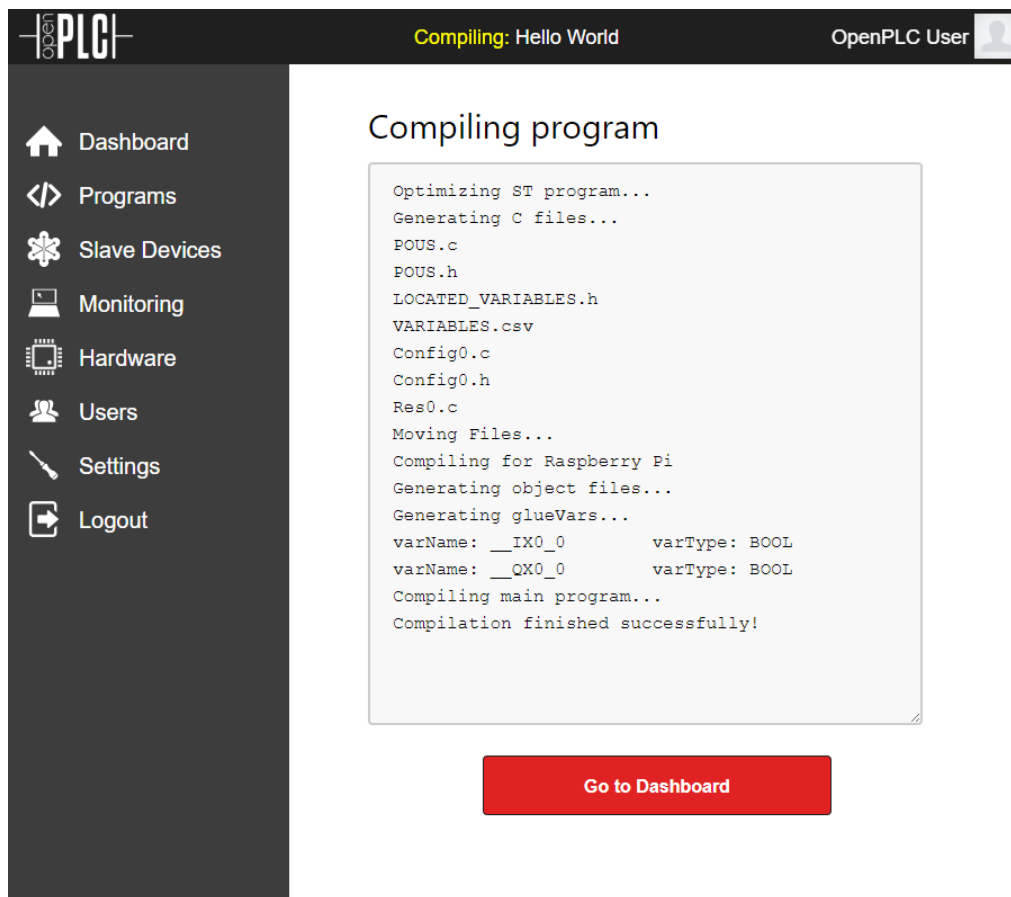


Figure 123: OpenPLC - Compiling Hello World program

If warnings occur during the compilation, then you can simply ignore them, it is important that the last line of the output says "Compilation finished successfully!".

With a click on the button "Go to Dashboard" we get back to the main page. Our program has already been selected as an active program by OpenPLC, but it has not started yet.

As a last step, we need to click the button in the main menu on the left side

Click "Start PLC" and our program starts.

In addition, we can track the activity of OpenPLC and our program in output "Runtime Logs".



PiXtend V2 Software Manual

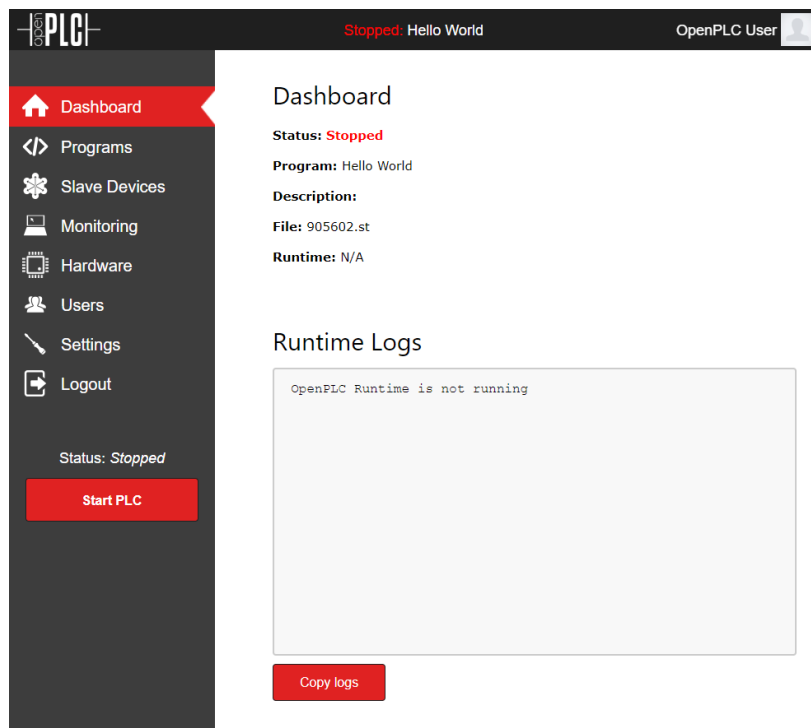


Figure 124: OpenPLC - Hello World program is stopped

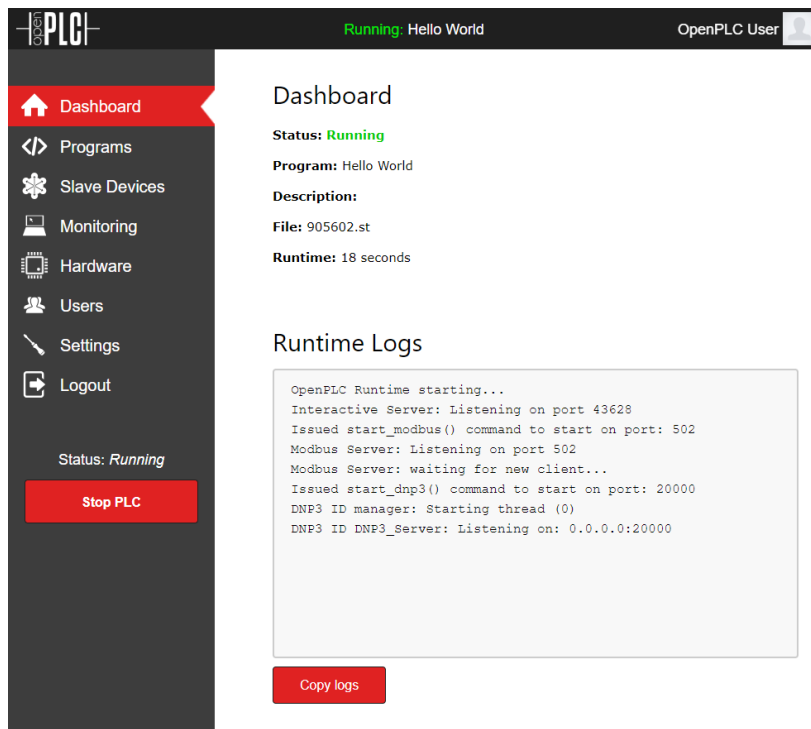


Figure 125: OpenPLC - Hello World program is running



PiXtend V2 Software Manual

To test the function of the program, set the PiXtend's DigitalIn0 to high level (depending on the jumper setting - at 5 V or 24 V). The output DigitalOut0 is activated.

If a low level (0 V) is then applied to DigitalIn0, DigitalOut0 remains active for another 2 seconds (switch-off delay).

You have commissioned your first PLC program with *PiXtend V2* and the software from the OpenPLC Project.

Further information for further work and to create more complex programs can be found in the following chapter.



12.4. Further Information

Below you will find the important information about *OpenPLC* and *PiXtend V2* so that you can write your own control programs.

12.4.1. Available PiXtend V2 -S- I/Os

In the following table you will find all available I/Os and their *OpenPLC* address together with type, description, value range and data type.

PiXtend description	Type	Description	Value range	Data type	Address
DigitalIn 0 - DigitalIn 7	Input	Digital inputs	False..True	BOOL	%IX0.0 - %IX0.7
GPIOIn 0 – GPIOIn 3	Input	GPIO inputs	False..True	BOOL	%IX1.0 - %IX0.3
Firmware Version	Input	Firmware version of the microcontroller	0..255	BYTE	%IB0
Hardware Version	Input	Hardware version of the PiXtend board	0..255	BYTE	%IB1
Model	Input	PiXtend V2 model information	0..255	BYTE	%IB2
uC State	Input	Microcontroller state	0..255	BYTE	%IB3
UC Warnings	Input	Microcontroller warnings	0..255	BYTE	%IB4
AnalogIn 0 - AnalogIn 1	Input	Analog inputs	0..1023	WORD	%IW0 - %IW1
Temp 0 – Temp 3	Input	Temperature inputs, raw value	0..65535	WORD	%IW2 - %IW5
Humid 0 – Humid 3	Input	Humidity inputs, raw value	0..65535	WORD	%IW6 - %IW9
DigitalOut 0 - DigitalOut 3	Output	Digital outputs	False..True	BOOL	%QX0.0 - %QX0.3
RelayOut 0 - RelayOut 3	Output	Relay outputs	False..True	BOOL	%QX0.4 - %Q0.7
GPIOOut 0 - GPIOOut 3	Output	GPIO outputs	False..True	BOOL	%QX1.0 - %QX1.3
uCCtrl 0	Output	Microcontroller Control-Register 0	0..255	BYTE	%QB0
uCCtrl 1	Output	Microcontroller Control-Register 1	0..255	BYTE	%QB1
GPIOCtrl	Output	GPIO Control-Register	0..255	BYTE	%QB2
AnalogOut 0 - AnalogOut 1	Output	Analog outputs (DAC)	0..1023	WORD	%QW0 - %QW1
PWM 0 Ctrl 0	Output	PWM 0 Control-Register 0	0..255	BYTE	%QB3
PWM 0 A / B	Output	PWM 0 Value for channel A / B	0..16000	WORD	%QW2 - %QW3
PWM 0 Ctrl 1	Output	PWM 0 Control-Register 1	0..65535	WORD	%QW4
PWM 1 Ctrl 0	Output	PWM 1 Control-Register 0	0..255	BYTE	%QB4
PWM 1 Ctrl 1	Output	PWM 1 Control-Register 1	0..255	BYTE	%QB5
PWM 1 A / B	Output	PWM 1 Value for channel A / B	0..255	BYTE	%QB6 - %QB7

Table 4: *OpenPLC* – Overview Hardware addresses of PiXtend V2 -S-



12.4.2. Available PiXtend V2 -L- I/Os

In the following table you will find all available *PiXtend V2 -L-* I/Os and their OpenPLC address together with type, description, value range and data type.

PiXtend description	Type	Description	Value range	Data type	Address
DigitalIn 0 - DigitalIn 15	Input	Digital inputs	False..True	BOOL	%IX0.0 - %IX1.7
GPIOIn 0 – GPIOIn 3	Input	GPIO inputs	False..True	BOOL	%IX2.0 - %IX2.3
Firmware Version	Input	Firmware version of the microcontroller	0..255	BYTE	%IB0
Hardware Version	Input	Hardware version of the PiXtend board	0..255	BYTE	%IB1
Model	Input	PiXtend V2 model information	0..255	BYTE	%IB2
uC State	Input	Microcontroller state	0..255	BYTE	%IB3
UC Warnings	Input	Microcontroller warnings	0..255	BYTE	%IB4
AnalogIn 0 - AnalogIn 5	Input	Analog inputs	0..1023	WORD	%IW0 - %IW5
Temp 0 – Temp 3	Input	Temperature inputs, raw value	0..65535	WORD	%IW6 - %IW9
Humid 0 – Humid 3	Input	Humidity inputs, raw value	0..65535	WORD	%IW10 - %IW13
DigitalOut 0 -DigitalOut 11	Output	Digital outputs	False..True	BOOL	%QX0.0 - %QX1.3
RelayOut 0 -RelayOut 3	Output	Relay outputs	False..True	BOOL	%QX1.4 - %Q1.7
GPIOOut 0 -GPIOOut 3	Output	GPIO outputs	False..True	BOOL	%QX2.0 - %QX2.3
uCCtrl 0	Output	Microcontroller Control-Register 0	0..255	BYTE	%QB0
uCCtrl 1	Output	Microcontroller Control-Register 1	0..255	BYTE	%QB1
GPIOCtrl	Output	GPIO Control-Register	0..255	BYTE	%QB2
AnalogOut 0 -AnalogOut 1	Output	Analog outputs (DAC)	0..1023	WORD	%QW0 - %QW1
PWM 0 Ctrl 0	Output	PWM 0 Control-Register 0	0..255	BYTE	%QB3
PWM 0 A / B	Output	PWM 0 Value for channel A / B	0..16000	WORD	%QW2 - %QW3
PWM 0 Ctrl 1	Output	PWM 0 Control-Register 1	0..65535	WORD	%QW8
PWM 1 Ctrl 0	Output	PWM 1 Control-Register 0	0..255	BYTE	%QB4
PWM 1 Ctrl 1	Output	PWM 1 Control-Register 1	0..65535	WORD	%QW9
PWM 1 A / B	Output	PWM 1 Value for channel A / B	0..255	BYTE	%QW5 - %QW5
PWM 2 Ctrl 0	Output	PWM 2 Control-Register 0	0..255	BYTE	%QB5
PWM 2 A / B	Output	PWM 2 Value for channel A / B	0..16000	WORD	%QW6 - %QW7
PWM 2 Ctrl 1	Output	PWM 2 Control-Register 1	0..65535	WORD	%QW10

Table 5: OpenPLC – Overview Hardware addresses of PiXtend V2 -L-

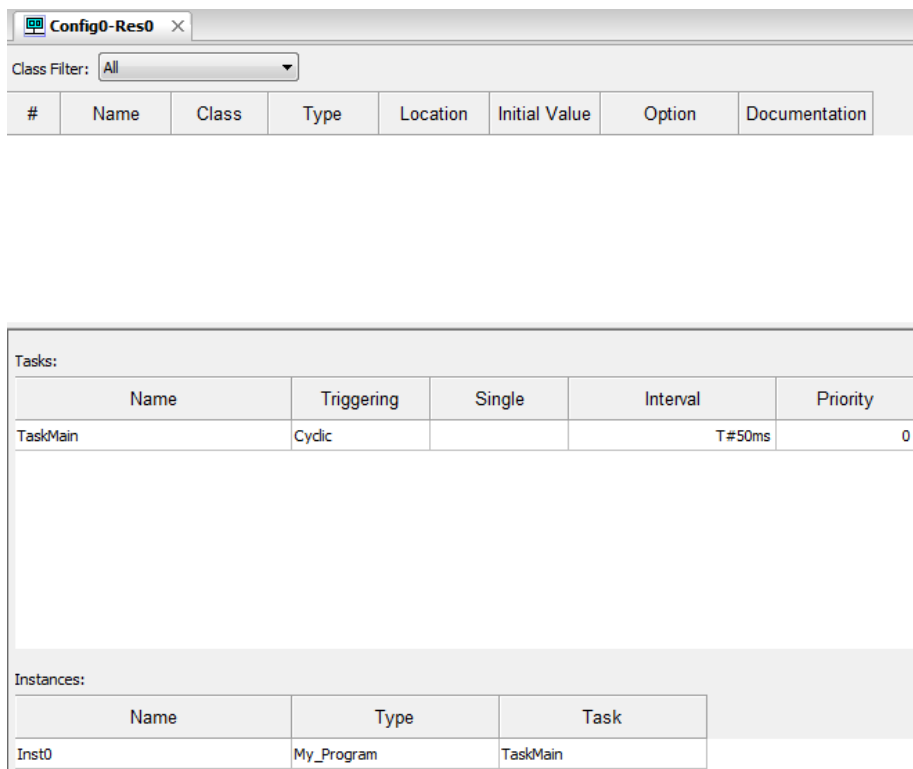
Note: CAN bus and serial communication are currently not part of OpenPLC.



12.4.3. Handling Tasks in the OpenPLC Editor

As you can see in the example program "Hello_World", there are special configurations for the project or the controller in the project tree: *Config0*

The sub-element "*Res0*" defines the cycle time of the PLC. In the example program, the content of "*Res0*" looks like this:



The screenshot shows the 'Config0-Res0' window in the OpenPLC editor. It features a 'Class Filter' dropdown set to 'All' and a table with columns: #, Name, Class, Type, Location, Initial Value, Option, and Documentation. Below this is a 'Tasks' section with a table containing one entry: 'TaskMain' with 'Cyclic' triggering, an interval of 'T#50ms', and a priority of '0'. At the bottom is an 'Instances' section with a table showing 'Inst0' of type 'My_Program' assigned to the 'TaskMain' task.

#	Name	Class	Type	Location	Initial Value	Option	Documentation
---	------	-------	------	----------	---------------	--------	---------------

Name	Triggering	Single	Interval	Priority
TaskMain	Cyclic		T#50ms	0

Name	Type	Task
Inst0	My_Program	TaskMain

Figure 126: OpenPLC - Config0 / Task overview

The TaskMain was set to "Cyclic" and the interval to 50 ms.

You can set the interval for *PiXtend V2 -S-* to a minimum of 2.5 ms and for *PiXtend V2 -L-* to a minimum of 5 ms. A setting smaller than 2.5 ms / 5 ms can cause problems and is therefore not recommended!

Further down you will see the section "*Instances*" where "*TaskMain*" is assigned to our program "My_Program".

If necessary, you can also use multiple tasks with different settings. For the first steps, we recommend using the default settings and setting up only one task.



12.4.4. OpenPLC Runtime autostart

After a reboot / power-up of the Raspberry Pi, the *OpenPLC* Runtime does not start automatically. As described during the installation, the runtime can be started with the following command:

```
sudo ./start_openplc.sh
```

We can have this command executed automatically when booting the Linux system.

The change is done quickly in the Linux console:

```
sudo nano /etc/rc.local
```

The file "*rc.local*" opens with some content. We enter two new lines before the line "*exit 0*":

```
cd /home/pi/OpenPLC_v3/  
sudo ./start_openplc.sh &
```

The "&" is not a typo. This starts the *OpenPLC* runtime as a process in the background.

Save changes made (Ctrl + O) and exit with Ctrl + X → Restart



12.5. Frequently Asked Questions (FAQ)

I can save, upload and run my program - but nothing happens - what can I do?

Usually in such cases the SPI bus is not active. Please take a look at the installation chapter. Otherwise, the dip switch *SPI_EN* on the PiXtend board could be set to "Off". Slide the *SPI_EN* switch to the "On" position to allow communication between the Raspberry Pi and *PiXtend*.

If this is your first *PiXtend* or your first setup, you can use one of our prepared SD card images (Basic Image or CODESYS Image) for the first steps. These are already preconfigured and tested by us. This should cover the software side things. If there are still difficulties, there is probably a hardware problem.

In this case check all switches and jumpers as well as the connection of the supply voltage, if everything is connected correctly.

For an exchange of information, the forums of *Qube Solutions* and the *OpenPLC Project* are available:

<http://www.pixtend.de/forum/>

<http://openplc.discussion.community/>



13. FourZero™ Support

Coming soon!

The *PiXtend V2* will support the platform FourZero™ from Automation of Things Europe GmbH. This makes it possible to program PiXtend V2 devices according to the IEC61499 standard.

The release is planned for 2018.

For more information on FourZero™, visit their homepage at <http://www.automationofthings.com>.



14. Appendix

In this chapter, you will find the process data exchanged between the PiXtend microcontroller and the Raspberry Pi (RPI). We speak here of process data, as it is about the data of inputs and outputs of the PiXtend board.

The information in this document are especially intended for people and institutions who wish to program their own software support for PiXtend V2.

With the PiXtend sample programs for CODESYS, the data is already correctly transmitted, evaluated and, if necessary, processed.

With the Linux tools and the use of the PiXtend library (pxdev) it is important to know how the data can be influenced and evaluated. This also applies the PiXtend Python Library V2.

The process data is defined in the PiXtend microcontroller and therefore independent of the programming language or Linux software used.

If questions remain unanswered, we would like to invite you to start a new topic in our community forum: www.pixtend.de/forum/.

The latest versions of all documents and software components can be found in the download section of our homepage: <https://www.pixtend.de/downloads/>



14.1. PiXtend V2 -S-

14.1.1. Process Data

14.1.1.1 Process Data overview

There are two types of process data:

- Process data transferred from the Raspberry Pi to PiXtend
- Process data transferred from PiXtend to the Raspberry Pi

14.1.1.1.1 *Process data transferred from RPI to PiXtend*

Data for digital outputs, relays and GPIOs (as outputs) PWM data, retain data

- DigitalOut (P. 218)
- RelayOut (P. 219)
- GPIOOut (P. 220)
- PWM0X (L/H) – (P. 221)
- PWM1X (L/H) – (P. 227)
- RetainDataOutX (P. 234)

14.1.1.1.2 *Process data transferred from the RPi to the PiXtend DAC*

Data for analog outputs

- AnalogOutX (L/H) – (P. 235)

14.1.1.1.3 *Process data transferred from PiXtend to the RPi*

Data of digital and analog inputs, GPIOs (as inputs)

Temperature and humidity of DHT11/22 or AM2302 sensors, Retain Data

- DigitalIn (P. 237)
- AnalogInX (L/H) – (P. 238)
- GPIOIn (P. 239)
- TempX (L/H) – (P. 240)
- HumidX (L/H) – (P. 242)
- RetainDataInX – (P. 243)



PiXtend V2 Software Manual

14.1.1.2 Output Data

The output data is the data that the Raspberry Pi transmits to the PiXtend microcontroller. An example are the digital outputs and relays.

14.1.1.2.1 DigitalOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	DO3	DO2	DO1	DO0
Startvalue	0	0	0	0	0	0	0	0

Table 6: Bits of the DigitalOut Byte

Bit 0..3

The four digital outputs are organized in one byte. The least significant bit (LSB) contains the state of DO0. Bit 7 (MSB), bit 6, bit 5 and bit 4 are not used.

The start value after the power-up or reset of the microcontroller is "0" for the entire DigitalOut byte. The outputs are deactivated at the Start and in SafeState.

By writing a "1" to one of the bits (0..3), the corresponding output is activated. The corresponding LED on the PiXtend board lights up.

The digital outputs on PiXtend V2 have a separate supply. On the light-emitting diode "VCC DO", you can see whether the supply is connected and is supplied with power.

The LEDs of the digital outputs also light up if the VCC-DO supply has not been connected, but no voltage is "visible" on the DO pins.

You can find more information in the hardware manual of PiXtend V2 -S-.

To test the function of the outputs, the value 255 (decimal) or 0xFF (hex) can be written on the DigitalOut byte. All four outputs are activated. The fact that bits 4 to 7 are also set here has no effect.



14.1.1.2.2 RelayOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	RELAY3	RELAY2	RELAY1	RELAY0
Startvalue	0	0	0	0	0	0	0	0

Table 7: Bits of the RelayOut Byte

Bit 0..3

The four relay outputs are organized in one byte. The least significant bit (LSB) contains the state of RELAY0. Bit 7 (MSB) to bit 4 are not used.

The entire RelayOut byte is "0" as the start value after the power-up or reset of the microcontroller. The relays are thus deactivated at the Start and in SafeState.

By writing a "1" to one of the bits (0..3), the corresponding relay is activated. The corresponding LED on the PiXtend board lights up and the relay switches audibly.

To test the function of the relays, the value 255 (decimal) or 0xFF (hex) can be written to the RelayOut byte. All four relays are activated. The fact that the bits 4..7 are also set here has no effect.



14.1.1.2.3 GPIOOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startvalue	0	0	0	0	0	0	0	0

Table 8: Bits of the GPIOOut Byte

Bit 0..3

The four GPIO outputs are organized in one byte. The least significant bit (LSB) contains the state of GPIO0. Bit 7 (MSB) to bit 4 are not used.

The entire GPIOout byte is "0" as the start value after the power-up or reset of the microcontroller. The GPIOs are disabled at the Start and in StafeState.

The GPIOs can perform three different tasks: control input, output or temperature / humidity sensors DHT11/22, AM2302. The configuration is done via the control byte GPIOCtrl. Further information can be found in the chapter: 14.1.2 *Control- & Status-Bytes*.

By writing a "1" to one of the bits (0..3) the corresponding GPIO is activated (if configured as output). If the GPIO or GPIOs are configured as input or for the sensors mentioned, changing the values in GPIOOut has no effect *.

To test the function of the GPIOs, they can be configured as outputs. The value 255 (decimal) or 0xFF (hex) can then be written to the GPIOOut byte. All four GPIO outputs are activated.

The fact that the bits 4..7 are also set here has no effect.

* If a GPIO is configured as an input with PullUps and the associated bit is set to "1" in GPIOOut, a pull-up resistor is activated on this GPIO. This gives the GPIO input a high-impedance (~ 35 k ohms) reference to 5 V and no longer floats. Without external wiring, the input now remains at high level. By default, the GPIO inputs are floating inputs with no defined levels (without external circuitry).

The pull-up resistors must generally be activated via the "GPIOPullUpEnable" bit - for further details see page 250.



14.1.1.2.4 PWM0X (L/H) – 16 Bit Resolution

The PWM units of PiXtend can be operated in four different modes. In each mode, the PWM process data has a different effect. For this reason, the four modes "Servo Mode", "Duty Cycle Mode", "Universal Mode" and "Frequency Mode" are discussed separately in this section.

For more information on the configuration and modes of the PWM0 outputs, see:

14.1.2.3.7 *PWM0Ctrl* – for 16 Bit PWMs from page 256 and following.

The PWM0X bytes are the process data for the two 16-bit PWM channels that carry the full designation PWM0A and PWM0B on the module.

14.1.1.2.4.1 Servo-Mode

The Servo Mode is specially designed for the requirements of model building servos. The period duration is set to 50 Hz (20 ms). At the beginning of the period, the signal is at least 1 ms (minimum) or maximum 2 ms (maximum). The remaining time is the signal always at low level.

The PWM0X-Bytes L and H contain a common 16 bit data word. Thereby PWM0XL the Low-Byte and PWM0XH the High-Byte. The „X“ is replaced by the letter of the respective PWM channel („A“ or „B“).

PWM0XL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 9: PWM0XL Byte

PWM0XH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	

Table 10: PWM0XH Byte

Adjustable value range

smallest value: 0 (decimal) → 1 ms, minimum position

maximum value: 16000 (decimal) → 2 ms, maximum position

Values greater than 16000 are limited by the controller and act as 16000. In the range between 0 and 16000 the movement of the servo is linear. The start value of the PWM0X byte after a power-up / reset is "0".



14.1.1.2.4.2 Duty-Cycle-Mode

In the Duty Cycle Mode, the bytes PWM0XL and PWM0XH contain a common 16 bit data word. This data word is used for setting the duty cycle. The „X“ is replaced by the letter of the respective PWM channel ("A" or "B"). A separate duty cycle can be assigned to each channel.

The frequency or period duration is set jointly for both channels.

The mode is thus ideally suited to control, for example, two drives or fans independently in their speed from 0% to 100%.

If it is important in your application to use several different frequencies, you will learn from page 225 and following more about the „Frequency Mode“.

PWM0XL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 11: PWM0XL Byte

PWM0XH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 12: PWM0XH Byte

The duty cycle resulting from a particular value depends not only on the PWM0XL / H values, but also on the control bytes PWM0Ctrl1L and PWM0Ctrl1H. The coarse setting of the frequency is done via the prescaler bits (PS0..2) in PWM0Ctrl0, the fine adjustment via PWM0Ctrl1L / H.

For more information on configuring the PWM0 channels, see the section:
14.1.2.3.7.114.1.2.3.7 PWM0Ctrl – for 16 Bit PWMs starting at page 256.

Calculation Formulas – PWM0 – 16 Bit

Frequency: $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWM0Ctrl1}$

Duty-Cycle: $\%on = 100 \% * \text{PWM0X} / \text{PWM0Ctrl1}$



PiXtend V2 Software Manual

Example

Duty Cycle Mode and the A & B channels, Prescaler to 1024:

PWM0Ctrl0 = 249 (decimal), or 11111001b (binary)

The following 16-bit value is written in PWM0Ctrl1L / H: 5000 (decimal).

The 16-bit value 2500 (decimal) is written in PWM0XL / H. The duty cycle is 50%.

If PWM0XL / H is written to a value greater than that in PWM0Ctrl1L / H, the PWM channel remains "1" throughout. To have a continuous logic "0" the value "0" has to be written in PWM0XL / H.

This gives us a frequency of 1.56 Hz in this example.

We recommend that you first test the implementation of the PWM functions in your own programs without connected devices or actuators and, if possible, with an oscilloscope.



Use the PWM values carefully!

Some actuators can be damaged or destroyed by incorrect frequencies or duty cycles.

Always check your programs with an oscilloscope before real devices are connected to the PWM channels.



14.1.1.2.4.3 Universal-Mode

In the Universal Mode, the bytes PWM0AL and PWM0AH contain a corresponding 16-bit data word. This data word is used to set the duty cycle of channel A.

The B channel can not be set in this mode. It outputs a PWM with 50% duty cycle and half the frequency of the A-channel.

The mode is suitable for applications where different frequencies are required, but one channel should also be configured to operate in duty cycle mode.

PWM0AL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 13: PWM0AL Byte

PWM0AH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 14: PWM0AH Byte

The configuration of the A channel is the same as for the Duty Cycle Mode.

A change in the bytes PWM0BL and PWM0BH has no effect. The B-channel depends solely on the frequency configuration of the A-channel.



14.1.1.2.4.4 Frequency-Mode

In the frequency Mode, the bytes PWM0XL and PWM0XH contain a corresponding 16-bit data word. This data word is used to set the frequency of the respective channel. The duty cycle is always 50% for channels A and B and can not be changed.

The mode is ideally suited if many different frequencies are required, but the duty cycle does not matter. This is, for example, the case when PWM channels are used to control stepper-motor drivers which only react on signal edges and do not require a variable duty cycle. With PiXtend, the speed of up to four stepper motors can be controlled (2x 16 Bit PWM0X, 2x 8 Bit PWM1X).

PWM0AL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 15: Bits of the PWM0AL Byte

PWM0AH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 16: Bits of the PWM0AH Byte

The rough setting of the frequency is done via the prescaler bits (PS0..2) in PWM0Ctrl0, the fine adjustment by PWM0XL / H.

The maximum frequency in this mode is 20 kHz. When higher frequencies are set, the output signal is limited to 20 kHz.

The frequency is calculated as follows:

Frequency channel X = 16 MHz / 2 / Prescaler / PWM0X



PiXtend V2 Software Manual

Example

A frequency of 500 Hz is to be output on channel A, and 250 Hz on channel B.

- Activate channel A and B, select frequency mode, configure prescaler to 64
→ PWM0Ctrl0 = 123 (decimal) or 01111011b (binary)

- PWM0A (16 Bit data word) = 250 (decimal)
→ Frequency channel A = $16 \text{ MHz} / 2 / 64 / 250 = 500 \text{ Hz}$

- PWM0B (16 Bit data word) = 500 (decimal)
→ Frequency channel B = $16 \text{ MHz} / 2 / 64 / 500 = 250 \text{ Hz}$

If the PWM signals should not be present at the outputs at all times, the channel (s) can be activated / deactivated in each cycle.

Notice:



The combination of Frequency Mode and DHT-sensors is not possible. If one or more DHT sensors have been activated on the PiXend GPIOs, the frequency mode can not be activated. The attempt to activate the frequency mode anyway, even though DHT sensors are used, results in the deactivation of the PWM outputs.



14.1.1.2.5 PWM1X (L/H) – 8 Bit Resolution

The PWM units of PiXtend can be operated in four different modes. In each mode, the PWM process data has a different effect. For this reason, the four modes "Servo Mode", "Duty Cycle Mode", "Universal Mode" and "Frequency Mode" are discussed separately in this section.

For more information on the configuration & modes of the PWM1 outputs, see:

14.1.2.3.8.114.1.2.3.8 PWM1Ctrl – for 8 Bit PWMs on page 260.

The PWM1X bytes are the process data for the two 8-bit PWM channels that carry the full designation PWM1A and PWM1B on the module.

14.1.1.2.5.1 Servo-Mode

The Servo Mode is specially designed for the requirements of model building servos. The period duration is set to 50 Hz (20 ms). At the beginning of the period, the signal is at least 1 ms (minimum) or maximum 2 ms (maximum). The remaining time is the signal low.

The PWM1 channels have two PWM1X bytes (L & H), but only the low byte (L) is used here. For example, only the low-byte is offered under CODESYS. The "X" is replaced by the letter of the respective PWM channel ("A" or "B").

PWM1XL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 17: Bits of the PWM1XL Byte

PWM1XH – not used

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 18: Bits of the PWM1XH Byte

Adjustable value range



PiXtend V2 Software Manual

smallest value: 0 (decimal) → 1 ms, minimum position

maximum value: 125 (decimal) → 2 ms, maximum position

Values greater than 125 lead to the maximum position (two milliseconds). In the range between 0 and 125 the movement of the servo is linear.

The start value of the PWM1XL byte, after a power-up / reset, is "0".



14.1.1.2.5.2 Duty-Cycle-Mode

The PWM1 channels have two PWM1X bytes (L & H), but only the low byte (L) is used here. For example, only the low byte is offered under CODESYS. The PWM1XL byte is used to set the duty cycle. The "X" is replaced by the letter of the respective PWM channel ("A" or "B"). A separate duty cycle can be assigned to each channel.

The frequency or period duration is set jointly for both channels.

The mode is thus ideally suited to control, for example, two drives or fans independently in their speed from 0% to 100%.

If it is important in your application to use several different frequencies, you will learn more about the "Frequency Mode" on page 232.

PWM1XL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 19: Bits of the PWM1XL Byte

PWM1XH – not used

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 20: Bits of the PWM1XH Byte

The duty cycle resulting from a particular value depends exclusively on the PWM1XL values. The frequency can not be set as precisely for the PWM1 channels as it can be done with the 16-bit PWM0 channels.

There are two factors that influence the frequency:

1. Set Prescaler in PWM1Ctrl0
2. Setting PWM1Ctrl1 to „1“, this doubles the frequency as set by the Prescaler

Calculation of the frequency: $f = 16 \text{ MHz} / 2 / \text{Prescaler} / 255$

The division by 2 is omitted when a "1" is written to PWM1Ctrl1.

For more information about the configuration, see: 14.1.2.3.8 PWM1Ctrl – for 8 Bit PWMs starting at page 260.



PiXtend V2 Software Manual

We recommend that you test the implementation of the Duty Cycle Mode in your own programs without connected devices or actuators and, if possible, to check the output with an oscilloscope.



Set the PWM values carefully!

Some actuators can be damaged or destroyed by incorrect frequencies or duty cycles.

Always check your programs with an oscilloscope before real devices are connected to the PWM channels.



14.1.1.2.5.3 Universal-Mode

The PWM1 channels have two PWM1X bytes (L & H), but only the low byte (L) is used here. For example, only the low byte is offered under CODESYS. The PWM1AL byte is used to set the duty cycle of channel A.

The B channel can not be set in this mode. It outputs a PWM signal with 50% duty cycle and half the frequency of channel A.

The mode is suitable for applications where different frequencies are required, but one channel should also be configured with regard to the duty cycle.

PWM1AL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 21: Bits of the PWM1AL Byte

PWM1AH – not used

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 22: Bits of the PWM1AH Byte

Calculation Formulas

Frequency: $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWM1Ctrl1}$

Duty-Cycle: $\%on = 100 \% * \text{PWM1AL} / \text{PWM1Ctrl1}$

A change to the byte PWM1BL has no effect. The B-channel depends solely on the frequency configuration of the A-channel.



14.1.1.2.5.4 Frequency-Mode

The PWM1 channels have two PWM1X bytes (L & H), but only the low byte (L) is used here. For example, only the low byte is offered under CODESYS. The PWM1AL byte is used to set the frequency of the respective channel. The duty cycle is always 50% for channels A and B and can not be changed.

The mode is ideally suited if as many different frequencies are required, but the duty cycle does not matter. This is the case, for example, if the PWM channels are used to control stepper motor drivers, which only react on signal edges and do not require a variable duty cycle. With PiXtend, up to four stepper motors can be controlled in their speed (2x 16 Bit PWM0X, 2x 8 Bit PWM1X).

PWM1AL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 23: Bits of the PWM1AL Byte

PWM1AH – not used

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 24: Bits of the PWM1AH Byte

A rough setting of the frequency is done via the Prescaler bits (PS0..2) in PWM1Ctrl0, the fine adjustment via PWM1XL.

The maximum frequency in this mode is 20 kHz. When higher frequencies are set, the output signal is limited to 20 kHz.

The frequency is calculated as follows:

Calculation of the frequency: $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWM1XL}$



PiXtend V2 Software Manual

Example

A frequency of 500 Hz is to be output on channel A, and 250 Hz on channel B.

- Activate channel A and B, select frequency mode, configure prescaler to 256
→ PWM1Ctrl0 = 219 (decimal) / 11011011 (binary)

- PWM1AL = 63 (decimal)

→ Frequency channel A = $16 \text{ MHz} / 2 / 256 / 63 = 496 \text{ Hz}$

- PWM1BL = 250 (decimal)

→ Frequency channel B = $16 \text{ MHz} / 2 / 256 / 250 = 125 \text{ Hz}$

If the PWM signals are not needed at the outputs all the time, the channel (s) can be activated / deactivated in each cycle.

As can be seen from this example, the frequency at channel A is not set perfectly to 500 Hz. PWM1AL should be set to 62.5, which is not possible with an integer data type.

Check the different prescaler settings for the desired frequency. It is possible to set the same or similar frequencies with different prescaler settings.

Notice:



The combination of Frequency Mode and DHT-sensors is not possible. If one or more DHT sensors have been activated on the PiXend GPIOs, the Frequency Mode can not be activated. The attempt to activate the Frequency Mode anyway, even though DHT sensors are used, results in the deactivation of the PWM outputs.



14.1.1.2.6 *RetainDataOutX*

Retain data consist of 32 bytes for each direction (*RetainDataOut* & *RetainDataIn*).

RetainDataOut, which is described in this section, is the data that is transferred from the Raspberry Pi to the PiXtend microcontroller for the actual backup. The storage location of the retain data is the microcontroller.

For the 32 bytes, there is no default for their data format. You can freely decide how to describe and use this data in your application. For example, under CODESYS, the 32 bytes are offered as a byte array.

The *RetainDataOutX* are transferred in each cycle. If the Retain Memory is activated (for further information, see page 250), the data is stored in the microcontroller in the event of a power failure / voltage interruption, and will be available again in the *RetainDataIn* bytes after the next start.



14.1.1.3 Outgoing Data – DAC

On PiXtend V2 -S- is a digital to analog converter (DAC), which is controlled directly by the Raspberry Pi and is responsible for the two analog outputs. The DAC is not part of the microcontroller and the data of the DAC is therefore not in the process image, which is exchanged between Raspberry Pi and microcontroller.

The DAC has a data format defined by the manufacturer of the chip, which is explained in the following. For further information, please refer to the data sheet of the chip¹⁰.

On PiXtend the 10 bit version of the chip is installed.

14.1.1.3.1 AnalogOutX (L/H)

The DAC receives a 16 bit data word per channel. The DAC does not provide a "response" - there is only one data direction. The X in AnalogOutX stands for channel A and B, respectively.

The chip is connected via SPI bus to the Raspberry Pi (ChipSelect - CS1). The high byte (AnalogOutXH) is transmitted first, then the low byte (AnalogOutXL).

On PiXtend V2 -S- channel A is responsible for AO0 and channel B is responsible for AO1.

AnalogOutXL

Bit	7	6	5	4	3	2	1	0
Name	D5	D4	D3	D2	D1	D0	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 25: Bits of the AnalogOutL Byte

The 10-bit value for the output is in both bytes AnalogOutXL and AnalogOutXH - D0 (LSB) to D9 (MSB).

The low-byte AnalogOutXL contains the lower 6 bits. The bits "0" and "1" of AnalogOutXL are not evaluated - "do not care".

¹⁰ Microchip MCP4812



AnalogOutXH

Bit	7	6	5	4	3	2	1	0
Name	/A-B	-	/GA	/SHDN	D9	D8	D7	D6
Startvalue	0	0	0	0	0	0	0	0

Table 26: Bits of the AnalogOutH Byte

The upper 4 bits of the 10 bit output value are located in AnalogOutXH.

There are also three configuration bits in the AnalogOutXH-Byte:

Bit 4 – /SHDN

The abbreviation SHDN means "Output Shutdown Control". The channel set via bit 7 (/A-B) can be activated or deactivated via the /SHDN bit. If a "1" is written in /SHDN, the channel is active.

Otherwise, or even after the PowerUp, the channel is set to "0" and thus disabled (start value). In the deactivated state, the analog outputs are switched on at PiXtend V2 -S- and a voltage of less than 50 mV is output. No matter what value the outputs are set to.

Bit 5 – /GA

Sets the "Output Gain Selection". The /GA bit must always be set to "0" (starting value) for the intended operation of PiXtend V2 -S- (0..10 V output voltage).

Bit 7 – /A-B

Selecting the channel - A or B. If the bit contains a "1", the 16 bit data word consisting of AnalogOutXL & AnalogOutXH is used for channel B. If bit 7 is set to "0" (start value), it is used for channel A.

All further information can be found in the data sheet of the DAC Chips – Microchip MCP4812.



PiXtend V2 Software Manual

14.1.1.4 Incoming Data

Input data is the data that the PiXtend microcontroller transmits to the Raspberry Pi. For example, the digital & analog inputs.

14.1.1.4.1 *DigitalIn*

Bit	7	6	5	4	3	2	1	0
Name	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Startvalue	0	0	0	0	0	0	0	0

Table 27: Bits of the DigitalIn Byte

The eight digital inputs (*DigitalIn*) are organized in one byte. The least significant bit (LSB) contains the state of DI0, the most significant (MSB) is the state of DI7.

The inputs always have the value "0" in idle state. In the active state (voltage applied), the value changes to "1". The level of a "1" or "0" can be found in the PiXtend V2 -S- hardware manual.

There is the possibility to debounce the digital inputs, which is useful in many applications. For more information, see:

14.1.2.3.4 *DigitalInDebounce* starting at page 252.



14.1.1.4.2 AnalogInX (L/H)

AnalogInXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 28: AnalogInXL

AnalogInXH

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	MSB	
Startvalue	0	0	0	0	0	0	0	0

Table 29: Bits of the AnalogInXH Byte

The analog / digital converter integrated in the PiXtend controller provides 10-bit values. The values are organized in the process data in a 16 bit data word consisting of two bytes. The most significant two bits, of the 10-bit value, are in the byte AnalogInXH.

The "X" is replaced by the number of the corresponding analog channel ("0" or "1").

The analog inputs are already digitally filtered in the microcontroller. 10 values are recorded, the mean value is formed and the result is stored in AnalogInX.

The value range is between "0" and "1023". The analog inputs are marked with AI0 and AI1 on the module or on the stainless steel housing.

In order to calculate voltages from these values, the following calculation formula is used:

Convert analog value to voltage

$\text{AnalogInX} * 10 / 1024 = \text{Voltage at channel X [V]}$

(Voltages are measured at AI0 and AI1)

The jumper must be observed for the voltage inputs. The "10" in the above calculation formula is used when no jumper is set (0..10 V range).

If the jumper is set for a channel (0..5 V range), a "5" instead of a "10" must be used in the formula.



14.1.1.4.3 GPIOIn

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startvalue	0	0	0	0	0/1	0/1	0/1	0/1

Table 30: Bits of the GPIOIn Byte

The four GPIO inputs (GPIO_IN) are organized in one byte. The least significant bit (LSB) contains the state of GPIO0. Bit 7 (MSB) to bit 4 are not used.

The GPIO inputs have no defined rest state * (floating input). Only by the external application of a voltage does a stable state result:

0 V → Value „0“

5 V → Value „1“

Since the inputs are floating, they can be "0" or "1" without an external connection or wobble between the values.

The GPIOs can perform three different tasks: be an input / output or read the temperature / humidity from sensors like DHT11/22, AM2302. The configuration is done via the control byte GPIOCtrl. For more information, see the chapter 14.1.2 *Control- & Status-Bytes*.

After a reset or power-up of the device, the GPIOs are configured as inputs.

* There is the possibility to configure the GPIOs as inputs and to activate via GPIOOut so-called pull-up resistors. This causes a "1" at the input in the idle state (without floating inputs). For more information, see: 14.1.1.2.3 *GPIOOut* starting at page 220.



14.1.1.4.4 TempX (L/H)

TempXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 31: TempXL Byte

TempXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 32: TempXH Byte

In the 1-Wire / DHT-Mode of the GPIOs, the TempXL and TempXH bytes contain a 16-bit data word. This data word contains the temperature information of a sensor (if a sensor is connected and configured). The "X" stands for the number of the GPIO, ie "0" to "3".

The GPIOs can perform three different tasks: be an input / output or read temperature / humidity from sensors like DHT11/22, AM2302. The configuration is done via the control byte GPIOCtrl. For more information, see chapter 14.1.2 Control- & Status-Bytes.

The containing value can very easily be converted to a temperature value (floating-point value). However, a distinction must be made between DHT11 and DHT22 (TempX represents the 16-bit value):

DHT11: $\text{TempX} / 256 = \text{Floating point } [^{\circ}\text{C}]$

Example: $5632 / 256 = 22,0\text{ }^{\circ}\text{C}$ – DHT11 Sensors do not provide a decimal point!

DHT22: $\text{TempX} / 10 = \text{Floating point } [^{\circ}\text{C}]$

Example: $224 / 10 = 22,4\text{ }^{\circ}\text{C}$



PiXtend V2 Software Manual

Depending on the programming language and data type, a "typecast" must be carried out. Otherwise, after dividing by 10 (DHT22), there is no floating-point value but an integer value (as an example 22 instead of 22.4).

It is also possible to divide by 256 with the DHT11 sensors using a "right shift" by 8 digits.

Negative temperatures can also be measured with DHT22 sensors. To measured this, the MSB must be evaluated in TempXH. If the MSB is a "1", then it is a negative temperature.



14.1.1.4.5 HumidX (L/H)

HumidXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 33: HumidXL Byte

HumidXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 34: HumidXH Byte

In the 1-Wire / DHT-Mode of the GPIOs, the HumidXL and HumidXH bytes contain a 16-bit data word. This data word contains the humidity information of a sensor (if a sensor is connected and configured). The "X" stands for the number of the GPIO, ie "0" to "3".

The GPIOs can perform three different tasks: be an input / output or read temperature / humidity from sensors like DHT11 / 22, AM2302. The configuration is done via the control byte GPIOCtrl. For more information, see chapter 14.1.2 Control- & Status-Bytes.

The value can easily be converted to a humidity value (floating-point value). However, a distinction must be made between DHT11 and DHT22 (HumidX represents the 16-bit value):

DHT11: $\text{HumidX} / 256 = \text{Floating point [\%]} - \text{relative humidity}$

Example: $10496 / 256 = 41,0 \%$ – DHT11 Sensors do not provide a decimal point!

DHT22: $\text{HumidX} / 10 = \text{Floating point [\%]} - \text{relative humidity}$

Example: $417 / 10 = 41,7 \%$

Depending on the programming language and data type, a "typecast" must be carried out. Otherwise, after dividing by 10 (DHT22), there is no floating-point value but an integer value (as an example 22 instead of 22.4).

It is also possible to divide by 256 with the DHT11 sensors using a "right shift" by 8 digits.



14.1.1.4.6 *RetainDataInX*

Retain data consist of 32 bytes for each direction (RetainDataOut & RetainDataIn). RetainDataIn, which is described in this section, is the previously stored data which, after a power failure / voltage interruption, is transferred back from the PiXtend microcontroller to the Raspberry Pi.

After restarting the PiXtend system, the previously stored data is available in RetainDataIn, if the Retain Memory has previously been activated, before the power failure. For further information, see page 250.

The data is retained until new retain data is stored. The data can also be retained spanning any number of system power cycles if the Retain functionality is not reactivated.

If the Retain memory is activated again and a power supply interruption occurs, the data is overwritten by the current data in RetainDataOut.



14.1.2. Control- & Status-Bytes

PiXtend or the PiXtend microcontroller can be configured via control bytes. In addition, the microcontroller informs the Raspberry Pi using status bytes and, if necessary, of errors and warnings.

The following pages provide an overview of the control and status bytes. Each byte is then described in detail. The various possibilities are shown and exemplary calculations are carried out.

Should any questions remain unanswered, please let us know by e-mail (support@pixtend.com). You will receive an answer as soon as possible.



14.1.2.1 Control-Bytes Overview

Control bytes are transferred from the Raspberry Pi to the PiXtend microcontroller. This way you can influence the behavior of the microcontroller.

The available control bytes are:

- **ModelOut (P. 247)**
indicates which hardware version (model) the RPi software expects (e.g., CODESYS or pxdev)
- **UCMode (P. 247)**
specifies which transmission mode to use
- **UCCtrl (P. 248)**
basic settings of the microcontroller (Retain, Watchdog, SafeState ...)
- **DigitalInDebounce (P. 252)**
indicates whether and how the digital inputs should be debounced
- **GPIOCtrl (P. 254)**
configures the PiXtend GPIOs (input, output, DHT-Mode)
- **GPIODebounce (P. 255)**
indicates whether and how the GPIO inputs should be debounced
- **PWM0Ctrl & PWM1Ctrl (P. 256 & 260)**
configures the PWM channels (mode, activation, frequencies)



14.1.2.2 Status-Bytes Overview

Status bytes are transferred from the PiXtend microcontroller to the Raspberry Pi. They contain important information about the state of PiXtend or the microcontroller.

The following status bytes are available:

- **Firmware (P. 265)**
informs about the microcontroller firmware version
- **Hardware (P. 265)**
informs about the PiXtend hardware version
- **Modelln (P. 265)**
informs about the PiXtend model
- **UCState (P. 266)**
contains the operating state & error codes of the microcontroller
- **UCWarnings (P. 267)**
contains warnings of the microcontroller



PiXtend V2 Software Manual

14.1.2.3 Description of the Control-Bytes

14.1.2.3.1 *ModelOut*

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	x	x	x	x	x	x	x	x

Table 35: *ModelOut* Byte

Using the byte *ModelOut*, the application software running on the Raspberry Pi informs the microcontroller of the PiXtend model it is expecting. This allows the microcontroller to decide whether it is the expected model and if not, return an error to the Raspberry Pi.

The content of *ModelOut* is an ASCII character. In the case of PiXtend V2 -S-, the byte contains the following value: ASCII character **S** (Capital letter S – 83 decimal / 53 hexadecimal).

For the user, there is usually no reason to change this value. As an example, drivers are provided with the CODESYS package "PiXtend V2 Professional for CODESYS", in which the model is fixed and can not be changed.

14.1.2.3.2 *UCMode*

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 36: *Bits of the UCMode* Byte

Currently (as of October 2017) there is only one transmission mode, the so-called "Auto-Mode". The byte is reserved for future use.

It is not necessary to enter a value for the operation of PiXtend. If a value is entered, this has no effect.



PiXtend V2 Software Manual

14.1.2.3.3 UC Ctrl

14.1.2.3.3.1 UC Ctrl0

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	WDE3	WDE3	WDE1	WDE0
Startvalue	0	0	0	0	0	0	0	0

Table 37: Bits of the UC Ctrl0 Byte

Bit 0..3 – WDE0..3 (WatchdogEnable0..3)

With the WDE0..3 bits the watchdog timer of the PiXtend microcontroller can be activated and set.

The default setting is that all bits are "0". Activation and setting of the watchdog is explained in the following table:

WDE 3	WDE 2	WDE 1	WDE 0	Status of Watchdog	Time
0	0	0	0	deactivated	deactivated
0	0	0	1	active	16 ms
0	0	1	0	active	32 ms
0	0	1	1	active	64 ms
0	1	0	0	active	0,125 s
0	1	0	1	active	0,25 s
0	1	1	0	active	0,5 s
0	1	1	1	active	1 s
1	0	0	0	active	2 s
1	0	0	1	active	4 s
1	0	1	0	active	8 s

Table 38: Watchdog Settings

If the watchdog is activated, communication between Raspberry Pi and PiXtend is monitored. If there is a pause longer than the set time between two valid cycles, the watchdog will trigger and put the microcontroller in the Safe State*. An invalid cycle (e.g.,



PiXtend V2 Software Manual

because of a CRC error) is categorized by the watchdog as if no cycle had been performed.

The use of the watchdog makes sense if the PiXtend control needs to be brought into a defined state in the event of a fault. An error may occur, e.g. when the application program stops responding on the Raspberry Pi and no longer transfers data to the microcontroller.

Recommendation

Do not enable the watchdog during the development of your user software. If, for example, a new program is transferred via CODESYS or the user software is debugged, then no transmission takes place for a certain time. So the watchdog always triggers again and again and a reboot of the system becomes necessary. This can lead to an unnecessary delay in your development process.

* Definition of "Safe State":

- All digital outputs & relays are switched off / put into the rest /idle state
- PWM outputs are switched to high-impedance (tri-state)
- Retain data is stored when Retain option is activated
- The "L1" status LED flashes after a fault if the LED was not deactivated (Error LED "L1"-Signals, page 311)
- The microcontroller or the PiXtend system must be restarted afterwards

If switching off the digital outputs for your application does not correspond to a safe state, external measures must be taken to counteract this behavior.



14.1.2.3.3.2 UC Ctrl1

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	GPUE	SLED	RE	RC	SAFE
Startvalue	0	0	0	0	0	0	0	0

Table 39: Bits of the UC Ctrl1 Byte

Bit 0 – SAFE (SafeState)

The SAFE bit, when activated by the value "1", can immediately set the microcontroller to the Safe State *. The default value is "0"

* Definition of "Safe State":

- All digital outputs & relays are switched off / put into the rest /idle state
- PWM outputs are switched to high-impedance (tri-state)
- Retain data is stored when Retain option is activated
- The "L1" status LED flashes after a fault if the LED was not deactivated (Error LED "L1"-Signals, page 311)
- The microcontroller or the PiXtend system must be restarted afterwards

If switching off the digital outputs for your application does not correspond to a safe state, external measures must be taken to counteract this behavior.

Bit 1 – RC (RetainCopy)

The RC bit can be used to configure which data is visible in the Retain input area (RetainDataIn0..31).

With the startvalue "0", the most recently stored data is transferred from the microcontroller to the Raspberry Pi (normal retention mode).

If the value "1" is set for the RC bit, the data sent last by the Raspberry Pi is returned. In the Retain area RetainDataIn0..31, then RetainDataOut0..31 is mirrored, but with a cycle delay (RetainCopy operation). The contents of the retain data is not lost, but is only displayed as long as the RC bit is "1".

For more information on PiXtend's Retain feature, see 6.3. Retain Memory.



Bit 2 – RE (RetainEnable)

With the RE bit, the Retain function of PiXtend can be activated. For this purpose, a "1" must be written into this bit. By default, RE is "0" (after a reset or power-up) and Retain is deactivated.

Recommendation

Activate Retain only if the functionality is really required. The number of Retain memory cycles can only be guaranteed up to 10,000 cycles.

Further information on the Retain function of PiXtend V2 -S- can be found in the chapter: 6 *Basic knowledge* on page 20 in this document.

Bit 3 – SLED (StatusLED)

The status LED "L1" can be deactivated by SLED bit if it is not required. By default, the status LED is active and can be deactivated by the value "1" in SLED.

Bit 4 – GPUE (GPIOPullUpEnable)

With the GPUE bit, the pull-up resistors of the PiXtend GPIOs can be enabled (these are, however, not yet active). The enable has an effect if the GPIOs are configured as inputs and a "1" is written in the byte GPIOOut for the respective GPIO.

A "1" in GPUE activates this functionality.

For more information about the PiXtend GPIOs, see: *GPIOOut* starting at page 220.



14.1.2.3.4 DigitalInDebounce

DigitalInDebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 40: Bits of the DigitalInDebounce01 Byte

DigitalInDebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 41: Bits of the DigitalInDebounce23 Byte

DigitalInDebounce45

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 42: Bits of the DigitalInDebounce45 Byte

DigitalInDebounce67

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 43: Bits of the DigitalInDebounce67 Byte

With the help of the DigitalInDebounce bytes the debouncing of the eight digital inputs of PiXtend can be configured. By default the debouncing of the inputs is not active.

The debouncing is activated / deactivated together for two channels at a time. As a result, no interactions occur between the two channels, but these only receive the same



PiXtend V2 Software Manual

debounce setting.

The number in a DigitalInDebounce byte corresponds to the number of cycles in which an electrical signal is present at the digital input and must remain constant, for the level change to be passed on to the application software (Raspberry Pi). This applies both to a change of the signal level from "0" (low) to "1" (high), as well as vice versa.

If the bytes are not changed or a "0" is written as a value, then no debouncing takes place and the application software receives a new value in each cycle.

Example:

The first two digital inputs (DI0 and DI1) should be debounced. Only changes to the digital inputs which are constantly applied for more than 100 ms should be visible. The cycle time (data exchange between PiXtend and Raspberry Pi) is 10 ms.

- The byte DigitalInDebounce01 is used
- Calculation of the value for DigitalInDebounce01: $100 \text{ ms} / 10 \text{ ms} = 10$

The byte is thus set to 10 (decimal) to get the desired effect. If the cycle time during the software development is changed, it must be remembered that the value for debouncing must also be adjusted.



14.1.2.3.5 GPIOCtrl

Bit	7	6	5	4	3	2	1	0
Name	SENS3	SENS2	SENS1	SENS0	IO3	IO2	IO1	IO0
Startvalue	0	0	0	0	0	0	0	0

Table 44: Bits of the GPIOCtrl Byte

Bits 0..3 – IO0..3

With the IO bits, the four PiXtend GPIOs can be configured as either a digital input or output. With the startvalue "0", all GPIOs are configured as inputs. For example, the bit IO3 is set to "1", GPIO3 becomes an output. The value is set via the data word GPIOOut.

Bits 4..7 – SENS0..3 (Sensor0..3)

The GPIOs can also be used for 1-wire sensors (DHT11, DHT22, AM2302) in addition to their input / output function. The upper four bits of the GPIOCtrl data word are available for this purpose. For example, the bit SENS3 is set to "1", then one of the afore mentioned sensors can be connected to GPIO3 and evaluated. The setting of the IO bit is no longer relevant. The SENSX bits have a higher priority compared to the IOX bits.

The results / measured values of the sensors are in TempXL / TempXH and HumidXL / HumidXH. The "X" corresponds to the number of the GPIO to which a sensor is connected.

Notice



Because time is required for communication with the sensors (approximately 25 ms), a minimum cycle time of 30 ms is needed if at least one sensor is used. It does not matter whether one or four sensors are queried.



14.1.2.3.6 GPIODebounce

GPIODebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 45: Bits of the GPIODebounce01 Byte

GPIODebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 46: Bits of the GPIODebounce23 Byte

The debouncing of the GPIOs only works if they are configured as inputs. The GPIODebounce bytes are otherwise used exactly the same as the DigitalInDebounce bytes. Further information & examples can be found in the section: *DigitalInDebounce* starting at page 252.



PiXtend V2 Software Manual

14.1.2.3.7 PWM0Ctrl – for 16 Bit PWMs

The following descriptions refers to the PWM0 channels (PWM0A & PWM0B) with 16 bit resolution. The information on the 8 bit PWM channels can be found in the following sections of 14.1.2.3.8 PWM1Ctrl – for 8 Bit PWMs.

14.1.2.3.7.1 PWM0Ctrl0

Bit	7	6	5	4	3	2	1	0
Name	PS2	PS1	PS0	EnableB	EnableA	-	MODE1	MODE0
Startvalue	0	0	0	0	0	0	0	0

Table 47: Bits of the PWM0Ctrl0 Byte

Bit 0..1 – MODE0..1

The PWM outputs can be operated in four different modes. The mode is configured via the MODE bits. By default, PiXtend starts in servo mode (MODE0..1 = 00).

MODE1	MODE0	Mode-Name	Description
0	0	Servo Mode	Both channels (A / B) are used to drive model-building servos - special signal form
0	1	Duty Cycle Mode	Both channels (A / B) have the same frequency, but independently adjustable duty cycle
1	0	Universal Mode	Freely adjustable frequency & duty cycle for channel A, half frequency for channel B and 50% duty cycle
1	1	Frequency Mode	Freely adjustable frequencies for both channels (A / B), duty cycle is always 50%

Table 48: Mode Bits of the PWM0Ctrl0 Byte

For the normal user, Servo Mode and Duty Cycle Mode are sufficient. Universal and Frequency Modes can be useful and used for specific applications, but are more complex to configure and use.

Bit 3..4 – EnableA, EnableB

With the EnableA and EnableB bits, the respective channel can be activated. By default, the PWM channels are disabled. By writing a "1" into the respective bit the channel is activated and the PWM is visible at the output.

The PWM can initially be configured for the desired task (mode, frequency / value ...) and then activated. The activation can also happen in the same cycle.



Bit 5..7 – PS0..PS2 (Prescaler0..3)

The prescaler bits (PS bits) allow a rough setting of the PWM frequency. Depending on the PWM mode (see MODE bits), the PS bits have different effects.

The following table provides information about the effects of the PS bits:

PS2	PS1	PS0	Description	Base Frequency
0	0	0	PWM-Outputs deactivated	-
0	0	1	Prescaler: 1	16 MHz
0	1	0	Prescaler: 8	2 MHz
0	1	1	Prescaler: 64	250 kHz
1	0	0	Prescaler: 256	62,5 kHz
1	0	1	Prescaler: 1024	15,625 kHz
1	1	0	Prescaler: 1024	15,625 kHz
1	1	1	Prescaler: 1024	15,625 kHz

Table 49: Prescaler-Bits in the PWM0Ctrl0 Byte

As mentioned, the prescaler settings have different effects, depending on the mode you have set:

Mode	Description
Servo Mode	No impact
Duty Cycle Mode	Setting the prescaler or the basic frequency
Universal Mode	Setting the prescaler or the basic frequency
Frequency Mode	Setting the prescaler or the basic frequency

Table 50: PWM Mode

The rough setting of the frequency is done via the prescaler bits (PS0..2) in PWM0Ctrl0, the fine adjustment via PWM0Ctrl1L / H.

How the frequency / period duration can be adjusted precisely can be read in the section: 14.1.1.2.4 PWM0X (L/H) – 16 Bit Resolution at page 221.



PiXtend V2 Software Manual

Notice



If the Frequency Mode is to be used (for advanced users), no DHT sensors may be connected or activated in the GPIOCtrl byte. The combination of Frequency Mode and DHT-sensors is not possible. If one or more DHT sensors have been activated on the PiXend GPIOs, the Frequency Mode can not be activated. The attempt to activate the Frequency Mode anyway, even though DHT sensors are used, results in the deactivation of the PWM outputs.



14.1.2.3.7.2 PWM0Ctrl1 (L/H)

The PWM0Ctrl1 bytes L and H contain a related 16 bit data word. PWM0Ctrl1L is the low byte and PWM0Ctrl1H is the high byte.

The PWM frequency / period duration can be set with the two bytes (in PWM mode "Duty-Cycle" & "Universal"). The frequency is valid for both channels.

In the Duty Cycle Mode, the duty cycle can be configured independently for each channel. In Universal Mode, the duty cycle can only be set for channel A.

The PWM1Ctrl1 bytes have no effect for the Frequency Mode.

PWM0Ctrl1L

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 51: PWM0Ctrl1L Byte

PWM0Ctrl1H

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 52: PWM0Ctrl1H Byte

Example of the PWM period of time in the "Duty Cycle" Mode:

The prescaler (PS0..2) is set to the value 64 and thus the basic frequency to 250 kHz. In addition, the Duty Cycle Mode and channel A are activated:

PWM0Ctrl0: 01101001b

The 16 bit data word PWM0Ctrl1 is set to value 1000:

PWM0Ctrl1L: 11101000b

PWM0Ctrl1H: 00000011b

$\text{PWM-Period} = \text{Microcontroller-Base Clock} / 2 / \text{Prescaler} / \text{PWM0Ctrl1}$

$\text{PWM-Period} = 16 \text{ MHz} / 2 / 64 / 1000 = \mathbf{125 \text{ Hz}}$



PiXtend V2 Software Manual

14.1.2.3.8 PWM1Ctrl – for 8 Bit PWMs

The descriptions below refers to the PWM1 channels (PWM1A & PWM1B) with 8 bit resolution. The information on the 16-bit PWM channels can be found in the previous chapter 14.1.2.3.7 PWM0Ctrl – for 16 Bit PWMs.

14.1.2.3.8.1 PWM1Ctrl0

Bit	7	6	5	4	3	2	1	0
Name	PS2	PS1	PS0	EnableB	EnableA	-	MODE1	MODE0
Startvalue	0	0	0	0	0	0	0	0

Bit 0..1 – MODE0..1

The PWM outputs can be operated in four different modes. The mode is configured via the MODE bits. By default, PiXtend starts in Servo Mode (MODE0..1 = 00).

MODE1	MODE0	Mode-Name	Description
0	0	Servo Mode	Both channels (A / B) are used to drive model-building servos - special signal form
0	1	Duty Cycle Mode	Both channels (A / B) have the same frequency, but independently adjustable duty cycle
1	0	Universal Mode	Freely adjustable frequency & duty cycle for channel A, half frequency for channel B and 50% duty cycle
1	1	Frequency Mode	Freely adjustable frequencies for both channels (A / B), duty cycle is always 50%

For the normal user, the Servo Mode and Duty Cycle Mode are sufficient. Universal and Frequency Mode can be useful for specific applications, but are more complex to configure and use.

Bit 3..4 – EnableA, Enable B

With the EnableA and EnableB bits, the respective channel can be activated. By default, the PWM channels are disabled. By writing a "1" into the respective bit the channel is activated and the PWM is visible at the output.

The PWM can initially be configured for the desired task (mode, frequency / value ...) and then activated. The activation can also happen in the same cycle.



PiXtend V2 Software Manual

Bit 5..7 – PS0..PS2 (Prescaler0..3)

The prescaler bits (PS bits) allow a rough setting of the PWM frequency. Depending on the PWM mode (see MODE bits), the PS bits have different effects.

The following table provides information about the effects of the PS bits:

CS2	CS1	CS0	Description	Base Frequency
0	0	0	PWM-Output deactivated	-
0	0	1	Prescaler: 1	16 MHz
0	1	0	Prescaler: 8	2 MHz
0	1	1	Prescaler: 32	0,5 MHz
1	0	0	Prescaler: 64	250 kHz
1	0	1	Prescaler: 128	125 kHz
1	1	0	Prescaler: 256	62,5 kHz
1	1	1	Prescaler: 1024	15,625 kHz

As mentioned, the prescaler settings have different effects, depending on the mode you have set:

Mode	Description
Servo Mode	No impact
Duty Cycle Mode	Setting the prescaler or the basic frequency
Universal Mode	Setting the prescaler or the basic frequency
Frequency Mode	Setting the prescaler or the basic frequency

The rough setting of the frequency is done via the prescaler bits (PS0..2) in PWM1Ctrl0, the fine adjustment via PWM1Ctrl1L / H.

How the frequency / period duration can be adjusted precisely can be found in section: 14.1.1.2.5 *PWM1X (L/H) – 8 Bit Resolution* found on page 227.



PiXtend V2 Software Manual

Notice:



The combination of Frequency Mode and DHT-sensors is not possible. If one or more DHT sensors have been activated on the PiXend GPIOs, the Frequency Mode can not be activated. The attempt to activate the Frequency Mode anyway, even though DHT sensors are used, results in the deactivation of the PWM outputs.



14.1.2.3.8.2 PWM1Ctrl1 (L/H)

The PWM1 channels (8-bit PWM) have two PWM1Ctrl bytes (L & H) but only the low-byte (L) is used here.

PWM1Ctrl1L can be used to set the PWM frequency / period (in PWM mode "Duty Cycle" & "Universal"). The frequency applies to both PWM1 channels (A & B).

In the Duty Cycle Mode, the duty cycle can be configured independently for each channel. In the Universal Mode, the duty cycle can only be set for channel A. The PWM1Ctrl1 bytes have no effect for the Frequency Mode.

PWM1Ctrl1L

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 53: Bits of the PWM1Ctrl1L Byte

PWM1Ctrl1H – not used

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 54: Bits of the PWM1Ctrl1H Byte

Example of the PWM period duration in "Universal Mode":

The prescaler (PS0..2) is set to the value 64 and thus the basic frequency to 250 kHz. In addition, the Universal Mode and channel A are activated:

PWM1Ctrl0: 10001010b

The 8 bit data byte PWM1Ctrl1L is set to the value 150:

PWM1Ctrl1L: 10010110b

PWM1Ctrl1H: is not used

PWM Period Duration = Microcontroller Clock / 2 / Prescaler / PWM0Ctrl1

PWM- Period Duration = 16 MHz / 2 / 64 / 150 = 0,833 kHz



Example of the PWM period duration in the „Duty-Cycle-Mode“:

The prescaler (PS0..2) is set to the value 64 and thus to 250 kHz. In addition, the Duty Cycle Mode and channel A are activated:

PWM1Ctrl0: 10001001b

A special feature of the 8-bit PWMs in the Duty Cycle Mode must be observed here:
there are only two different settings for PWM1Ctrl1L ("0" or "1")

PWM1Ctrl1L: 00000000b

PWM1Ctrl1H: not used

PWM- Period Duration = Microcontroller Clock / 2 / Prescaler / 255

PWM- Period Duration = 16 MHz / 64 / 255 / 2 = 490 Hz

PWM1Ctrl1L: 00000001b

PWM1Ctrl1H: not used

PWM- Period Duration = Microcontroller Clock / Prescaler / 255

PWM- Period Duration = 16 MHz / 64 / 255 = 980 Hz

An example for the Frequency Mode can be found in the section: 14.1.1.2.5.4 *Frequency-Mode* found on page 232.



PiXtend V2 Software Manual

14.1.2.4 Description of the Status-Bytes

The status bytes inform the user or the user program running on the Raspberry Pi about the status of the microcontroller.

The status bytes can be queried via CODESYS, PiXtend C-Library or PiXtend-Python-Library V2 (PPLV2).

14.1.2.4.1 *Firmware*

The firmware byte contains a number for the firmware version. The number can be between 0 and 255. If you require support for your PiXtend V2 device, always provide the firmware version.

14.1.2.4.2 *Hardware*

The hardware byte contains the version number of your PiXtend board. With the version number, the user software can check whether the software / drivers matches the hardware.

Example

Hardware-Byte contains number 21
→ PiXtend Hardware Version 2.1

14.1.2.4.3 *ModelIn*

In the ModelIn byte, the microcontroller informs the user software about the PiXtend V2 model. With the model identification the user software can check, if the software / drivers fits the hardware.

The content of ModelIn is an ASCII character. In the case of PiXtend V2 -S- the byte contains the following value: ASCII character **S** (uppercase S - 83 decimal / 53 hexadecimal).



14.1.2.4.4 UCState

Bit	7	6	5	4	3	2	1	0
Name	ERR3	ERR2	ERR1	ERR0	-	-	-	RUN
Startvalue	0	0	0	0	0	0	0	0

Table 55: Bits of the UCState Byte

Bit 0 – RUN

By querying this bit, the user can check whether PiXtend is ready for operation ("1") or if a successful data transmission has been carried out. This can make sense after the start of the PiXtend system.

If the controller is put into the Safe State (for example by the watchdog), the bit contains a "0". However, this can no longer be queried because no communication is possible (until a restart is performed).

Bit 4..7 – ERROR0..3

The ERR bits inform about problems that the microcontroller can recognize and pass on to the user program. The following table describes the error codes and their cause:

ERR 3	ERR 2	ERR 1	ERR 0	Error Description
0	0	0	0	No Error
0	0	0	1	-
0	0	1	0	CRC Error - Data the payload data obtained from the Raspberry Pi is faulty
0	0	1	1	Data Block is to short Driver fault / Wrong model
0	1	0	0	PiXtend model does not match the expected and the actual model do not match (ModelIn and ModelOut are not equal)
0	1	0	1	CRC Error - Header the header data received from Raspberry Pi is faulty
0	1	1	0	SPI Frequency is to high the chosen SPI frequency is too high, transmission errors occur

Table 56: Classification of the Error-Bits in the UCState Byte



14.1.2.4.5 UCWarnings

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	W1	W0	-
Startvalue	0	0	0	0	0	0	0	0

Table 57: Bits of the UCWarnings Byte

Bit 1..2 – W0..1 (RetainCRCErrror. RetainVoltageError)

The **WX** bits inform the user about warnings regarding the Retain functionality of PiXtend V2. In the user program, these bits can be checked to see whether the Retain memory functions correctly or not.

W0 – RetainCRCErrror

Is "1" if an error has been detected when checking the Retain memory.

W1 – RetainVoltageError

Is "1" if the current supply voltage of PiXtend is too low (less than 19 V). Retain storage can not be activated.



14.2. PiXtend V2 -L-

14.2.1. Process Data

14.2.1.1 Process Data overview

There are two types of process data:

- Process data transferred from the Raspberry Pi to PiXtend
- Process data transferred from PiXtend to the Raspberry Pi

14.2.1.1.1 *Process data transferred from RPI to PiXtend*

Data for digital outputs, relays and GPIOs (as outputs) PWM data, retain data

- DigitalOutX (P. 269)
- RelayOut (P. 271)
- GPIOOut (P. 272)
- PWMYX (L/H) – (P. 273)
- RetainDataOutX (P. 280)

14.2.1.1.2 *Process data transferred from the RPi to the PiXtend DAC*

Data for analog outputs

- AnalogOutX (L/H) – (S. 281)

14.2.1.1.3 *Process data transferred from PiXtend to the RPi*

Data of digital and analog inputs, GPIOs (as inputs)

Temperature and humidity of DHT11/22 or AM2302 sensors, Retain Data

- DigitalInX (P. 283)
- AnalogInX (L/H) – (P. 284)
- GPIOIn (P. 286)
- TempX (L/H) – (P. 287)
- HumidX (L/H) – (P. 289)
- RetainDataInX – (P. 290)



PiXtend V2 Software Manual

14.2.1.2 Output Data

The output data is the data that the Raspberry Pi transmits to the PiXtend microcontroller. An example are the digital outputs and relays.

14.2.1.2.1 DigitalOutX

The digital outputs are organized in two bytes. The byte DigitalOut0 contains the first eight outputs (DO0 - DO7). The byte DigitalOut1 the remaining four outputs (DO8 - DO11).

DigitalOut0

Bit	7	6	5	4	3	2	1	0
Name	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0
Startvalue	0	0	0	0	0	0	0	0

Table 58: Bits of the DigitalOut0 Byte

DigitalOut0 - Bit 0..7

The least significant bit (LSB) contains the state of DO0. The most significant bit (MSB) contains the state of DO7.

DigitalOut1

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	DO11	DO10	DO9	DO8
Startvalue	0	0	0	0	0	0	0	0

Table 59: Bits of the DigitalOut1 Byte

DigitalOut1 - Bit 0..3

The least significant bit (LSB) contains the state of DO8. Bit 7 (MSB), bit 6, bit 5 and bit 4 are not used.



PiXtend V2 Software Manual

The start value after the power-up or reset of the microcontroller is "0" for the entire DigitalOutX bytes. The outputs are deactivated at the Start and in StafeState.

By writing a "1" to one of the DO bits, the corresponding output is activated. The corresponding LED on the PiXtend board lights up.

The digital outputs on PiXtend V2 each have a separate infeed for four outputs:

DO0 – DO3: VCC-DO0-3

DO4 – DO7: VCC-DO4-7

DO8 – DO11: VCC-DO8-11

The supply connections are located on the terminal block, which also houses the four outputs. You can recognize at the LED "VCC-DOx-x" whether a supply voltage is connected to the supply.

The digital output LEDs will light up even if the VCC-DO supply is not connected, but no voltage will be "visible" on the DO pins.

Further information can be found in the hardware manual of *PiXtend V2 -L-*.

To test the function of the outputs, the value 255 (decimal) or 0xFF (hex) can be written to the DigitalOutX bytes. All twelve outputs are activated. The fact that bits 4 to 7 are also set in the DigitalOut1 byte has no effect.



14.2.1.2.2 RelayOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	RELAY3	RELAY2	RELAY1	RELAY0
Startvalue	0	0	0	0	0	0	0	0

Table 60: Bits of the RelayOut Byte

Bit 0..3

The four relay outputs are organized in one byte. The least significant bit (LSB) contains the state of RELAY0. Bit 7 (MSB) to bit 4 are not used.

The entire RelayOut byte is "0" as the start value after the power-up or reset of the microcontroller. The relays are thus deactivated at the Start and in StafeState.

By writing a "1" to one of the bits (0..3), the corresponding relay is activated. The corresponding LED on the PiXtend board lights up and the relay switches audibly.

To test the function of the relays, the value 255 (decimal) or 0xFF (hex) can be written to the RelayOut byte. All four relays are activated. The fact that the bits 4..7 are also set here has no effect.



14.2.1.2.3 GPIOOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startvalue	0	0	0	0	0	0	0	0

Table 61: Bits of the GPIOOut Byte

Bit 0..3

The four GPIO outputs are organized in one byte. The least significant bit (LSB) contains the state of GPIO0. Bit 7 (MSB) to bit 4 are not used.

The entire GPIOout byte is "0" as the start value after the power-up or reset of the microcontroller. The GPIOs are disabled at the Start and in SafeState.

The GPIOs can perform three different tasks: control input, output or temperature / humidity sensors DHT11/22, AM2302. The configuration is done via the control byte GPIOCtrl. Further information can be found in the chapter: 14.2.2 Control- & Status-Bytes.

By writing a "1" to one of the bits (0..3) the corresponding GPIO is activated (if configured as output). If the GPIO or GPIOs are configured as input or for the sensors mentioned, changing the values in GPIOOut has no effect *.

To test the function of the GPIOs, they can be configured as outputs. The value 255 (decimal) or 0xFF (hex) can then be written to the GPIOOut byte. All four GPIO outputs are activated.

The fact that the bits 4..7 are also set here has no effect.

* If a GPIO is configured as an input with PullUps and the associated bit is set to "1" in GPIOOut, a pull-up resistor is activated on this GPIO. This gives the GPIO input a high-impedance (~ 35 k ohms) reference to 5 V and no longer floats. Without external wiring, the input now remains at high level. By default, the GPIO inputs are floating inputs with no defined levels (without external circuitry).

The pull-up resistors must generally be activated via the "GPIOPullUpEnable" bit - for further details see page 297.



14.2.1.2.4 PWMYX – 16 Bit Resolution

The PWM units of PiXtend can be operated in four different modes. In each mode, the PWM process data has a different effect. For this reason, the four modes "Servo Mode", "Duty Cycle Mode", "Universal Mode" and "Frequency Mode" are discussed separately in this section.

For more information on the configuration and modes of the PWM0 outputs, see:

14.2.2.3.7 PWMYCtrl – für 16 Bit PWMs from page 304 and following.

The PWM bytes described below are the process data for the six 16-bit PWM channels that have the full name PWM0A (P0A), PWM0B (P0B), PWM1A (P1A), PWM1B (P1B) on the module. PWM2A (P2A) and PWM2B (P2B).

When we talk about two channels with the same number from a group - e.g. PWM0A & PWM0B form the first PWM group for example.

When a group is put into one specific mode. Then both channels A & B included in this PWM group are in this mode. A single channel in a group can not be put into another mode - only the entire group.

14.2.1.2.4.1 Servo-Mode

The Servo Mode is specially designed for the requirements of model building servos. The period duration is set to 50 Hz (20 ms). At the beginning of the period, the signal is at least 1 ms (minimum) or maximum 2 ms (maximum). The remaining time is the signal always at low level.

The PWMYX bytes L and H contain an associated 16-bit data word. Where PWMYXL is the low byte and PWMYXH is the high byte. The "Y" is replaced by the number of the group ("0", "1" or "2"). The "X" is replaced by the letter of the respective PWM channel ("A" or "B"). This means that all six PWM outputs can be mapped according to the same principle.



PWMYXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 62: PWMYXL Byte

PWMYXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 63: PWMYXH Byte

Adjustable value range

smallest value: 0 (decimal) → 1 ms, minimum position

maximum value: 16000 (decimal) → 2 ms, maximum position

Values greater than 16000 are limited by the controller and act as 16000. In the range between 0 and 16000 the movement of the servo is linear. The start value of the PWMYX byte after a power-up / reset is "0".



14.2.1.2.4.2 Duty-Cycle-Mode

In the Duty Cycle Mode, the bytes PWMYXL and PWMYXH contain a common 16 bit data word. This data word is used for setting the duty cycle.

The "Y" is replaced by the number of the group ("0", "1" or "2"). The "X" is replaced by the letter of the respective PWM channel ("A" or "B"). Each channel can be assigned a separate duty cycle. The frequency or period is set in common for both channels of a group.

The mode is thus ideally suited to control, for example, two drives or fans independently in their speed from 0% to 100%.

If it is important in your application to use several different frequencies, you will learn from page 278 and following more about the „Frequency Mode“.

PWMYXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 64: PWMYXL Byte

PWMYXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 65: PWMYXH Byte

The duty cycle resulting from a particular value depends not only on the PWMYXL / H values, but also on the control bytes PWMYCtrl1L and PWMYCtrl1H. The coarse setting of the frequency is done via the prescaler bits (PS0..2) in PWMYCtrl0, the fine adjustment via PWMYCtrl1L / H.

For more information on configuring the PWMYX channels, see the section: 14.2.2.3.7 PWMYCtrl – für 16 Bit PWMs from page 304.

Calculation Formulas – PWM0 – 16 Bit

Frequency: $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWMYCtrl1}$

Duty-Cycle: $\%on = 100 \% * \text{PWMYX} / \text{PWMYCtrl1}$



PiXtend V2 Software Manual

Example

Duty Cycle Mode and the A & B channels, Prescaler to 1024:

PWM0Ctrl0 = 249 (decimal), or 11111001b (binary)

The following 16-bit value is written in PWM0Ctrl1L / H: 5000 (decimal).

The 16-bit value 2500 (decimal) is written in PWM0XL / H. The duty cycle is 50%.

If PWM0XL / H is written to a value greater than that in PWM0Ctrl1L / H, the PWM channel remains "1" throughout. To have a continuous logic "0" the value "0" has to be written in PWM0XL / H.

This gives us a frequency of 1.56 Hz in this example.

We recommend that you first test the implementation of the PWM functions in your own programs without connected devices or actuators and, if possible, with an oscilloscope.



Use the PWM values carefully!

Some actuators can be damaged or destroyed by incorrect frequencies or duty cycles.

Always check your programs with an oscilloscope before real devices are connected to the PWM channels.



14.2.1.2.4.3 Universal-Mode

In universal mode, the bytes PWMYAL and PWMYAH contain an associated 16-bit data word. This data word is used to set channel A's duty cycle.

The B-channel is not adjustable in this mode. It outputs a PWM with 50% duty cycle and half the frequency of the A-channel.

The "Y" is replaced by the number of the group ("0", "1" or "2").

The mode is suitable for applications where different frequencies are required, but one channel should also be configured to operate in duty cycle mode.

PWMYAL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 66: PWMYAL Byte

PWMYAH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 67: PWMYAH Byte

The configuration of the A channel is the same as for the Duty Cycle Mode.

A change in the bytes PWMYBL and PWMYBH has no effect. The B-channel depends solely on the frequency configuration of the A-channel.



14.2.1.2.4.4 Frequency-Mode

In the frequency mode, the bytes PWMYXL and PWMYXH contain an associated 16-bit data word. This data word is used to set the frequency of the respective channel. The "Y" is replaced by the number of the group ("0", "1" or "2"). The "X" is replaced by the letter of the respective PWM channel ("A" or "B").

The duty cycle is always 50% for channels A and B and can not be changed.

The mode is ideally suited if many different frequencies are required, but the duty cycle does not matter. This is, for example, the case when PWM channels are used to control stepper-motor drivers which only react on signal edges and do not require a variable duty cycle. Thus, the speed of up to six stepper motors can be controlled with PiXtend V2 -L-.

PWMYXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 68: Bits of the PWMYXL Byte

PWMYXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 69: Bits of the PWMYXH Byte

The rough setting of the frequency is done via the prescaler bits (PS0..2) in PWMYCtrl0, the fine adjustment by PWMYXL / H.

The maximum frequency in this mode is 20 kHz. When higher frequencies are set, the output signal is limited to 20 kHz.

The frequency is calculated as follows:

Frequency channel X = 16 MHz / 2 / Prescaler / PWMYX



PiXtend V2 Software Manual

Example

A frequency of 500 Hz is to be output on channel A, and 250 Hz on channel B.

- Activate channel A and B, select frequency mode, configure prescaler to 64
→ PWM0Ctrl0 = 123 (decimal) or 01111011b (binary)

- PWM0A (16 Bit data word) = 250 (decimal)
→ Frequency channel A = $16 \text{ MHz} / 2 / 64 / 250 = 500 \text{ Hz}$

- PWM0B (16 Bit data word) = 500 (decimal)
→ Frequency channel B = $16 \text{ MHz} / 2 / 64 / 500 = 250 \text{ Hz}$

If the PWM signals should not be present at the outputs at all times, the channel (s) can be activated / deactivated in each cycle.

Notice:



The combination of Frequency Mode and DHT-sensors is not possible. If one or more DHT sensors have been activated on the PiXend GPIOs, the frequency mode can not be activated. The attempt to activate the frequency mode anyway, even though DHT sensors are used, results in the deactivation of the PWM outputs.



14.2.1.2.5 *RetainDataOutX*

Retain data consist of 64 bytes for each direction (*RetainDataOut* & *RetainDataIn*). *RetainDataOut*, which is described in this section, is the data that is transferred from the Raspberry Pi to the PiXtend microcontroller for the actual backup. The storage location of the retain data is the microcontroller.

For the 64 bytes, there is no default for their data format. You can freely decide how to describe and use this data in your application. For example, under CODESYS, the 64 bytes are offered as a byte array.

The *RetainDataOutX* are transferred in each cycle. If the Retain Memory is activated (for further information, see page 297), the data is stored in the microcontroller in the event of a power failure / voltage interruption, and will be available again in the *RetainDataIn* bytes after the next start.



14.2.1.3 Ausgangsdaten – DAC

On PiXtend V2 -L- is a digital to analog converter (DAC), which is controlled directly by the Raspberry Pi and is responsible for the two analog outputs. The DAC is not part of the microcontroller and the data of the DAC is therefore not in the process image, which is exchanged between Raspberry Pi and microcontroller.

The DAC has a data format defined by the manufacturer of the chip, which is explained in the following. For further information, please refer to the data sheet of the chip¹¹.

On PiXtend the 10 bit version of the chip is installed.

14.2.1.3.1 AnalogOutX (L/H)

The DAC receives a 16 bit data word per channel. The DAC does not provide a "response" - there is only one data direction. The **X** in *AnalogOutX* stands for channel A and B, respectively.

The chip is connected via SPI bus to the Raspberry Pi (ChipSelect - CS1). The high byte (AnalogOutXH) is transmitted first, then the low byte (AnalogOutXL).

On PiXtend V2 -L- channel A is responsible for AO0 and channel B is responsible for AO1.

AnalogOutXL

Bit	7	6	5	4	3	2	1	0
Name	D5	D4	D3	D2	D1	D0	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 70: Bits of the AnalogOutL Byte

The 10-bit value for the output is in both bytes AnalogOutXL and AnalogOutXH - D0 (LSB) to D9 (MSB).

The low-byte AnalogOutXL contains the lower 6 bits. The bits "0" and "1" of AnalogOutXL are not evaluated - "do not care".

11 Microchip MCP4812



AnalogOutXH

Bit	7	6	5	4	3	2	1	0
Name	/A-B	-	/GA	/SHDN	D9	D8	D7	D6
Startvalue	0	0	0	0	0	0	0	0

Table 71: Bits of the AnalogOutH Byte

The upper 4 bits of the 10 bit output value are located in AnalogOutXH.

There are also three configuration bits in the AnalogOutXH-Byte:

Bit 4 – /SHDN

The abbreviation SHDN means "Output Shutdown Control". The channel set via bit 7 (/A-B) can be activated or deactivated via the /SHDN bit. If a "1" is written in /SHDN, the channel is active.

Otherwise, or even after the PowerUp, the channel is set to "0" and thus disabled (start value). In the deactivated state, the analog outputs are switched on at PiXtend V2 -L- and a voltage of less than 50 mV is output. No matter what value the outputs are set to.

Bit 5 – /GA

Sets the "Output Gain Selection". The /GA bit must always be set to "0" (starting value) for the intended operation of PiXtend V2 -L- (0..10 V output voltage).

Bit 7 – /A-B

Selecting the channel - A or B. If the bit contains a "1", the 16 bit data word consisting of AnalogOutXL & AnalogOutXH is used for channel B. If bit 7 is set to "0" (start value), it is used for channel A.

All further information can be found in the data sheet of the DAC Chips – Microchip MCP4812.



PiXtend V2 Software Manual

14.2.1.4 Incoming Data

Input data is the data that the PiXtend microcontroller transmits to the Raspberry Pi. For example, the digital & analog inputs.

14.2.1.4.1 DigitalInX

DigitalIn0

Bit	7	6	5	4	3	2	1	0
Name	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Startvalue	0	0	0	0	0	0	0	0

Table 72: Bits of the DigitalIn0 Byte

The first eight digital inputs (DigitalIn0) are organized in one byte. The least significant bit (LSB) contains the state of DI0, the most significant (MSB) the state of DI7.

DigitalIn1

Bit	7	6	5	4	3	2	1	0
Name	DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8
Startvalue	0	0	0	0	0	0	0	0

Table 73: Bits of the DigitalIn1 Byte

The other eight digital inputs (DigitalIn1) are organized in one byte. The least significant bit (LSB) contains the state of DI8, the most significant (MSB) the state of DI15.

The inputs always have the value "0" in idle state. In the active state (voltage applied), the value changes to "1". The level of a "1" or "0" can be found in the PiXtend V2 -L- hardware manual.

There is the possibility to debounce the digital inputs, which is useful in many applications. For more information, see:

14.2.2.3.4 DigitalInDebounce starting at page 299.



14.2.1.4.2 AnalogInX (L/H)

AnalogInXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 74: AnalogInXL

AnalogInXH

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	MSB	
Startvalue	0	0	0	0	0	0	0	0

Table 75: Bits of the AnalogInXH Byte

The analog / digital converter integrated in the PiXtend controller provides 10-bit values. The values are organized in the process data in a 16 bit data word consisting of two bytes. The most significant two bits, of the 10-bit value, are in the byte AnalogInXH.

The "X" is replaced by the number of the corresponding analog channel ("0" or "1").

The analog inputs are already digitally filtered in the microcontroller. 10 values are recorded, the mean value is formed and the result is stored in AnalogInX.

The value range is between "0" and "1023". The analog inputs are marked on the module or on the stainless steel housing with AI0, AI1, AI2, AI3, AI4 and AI5. AI0 - AI3 are voltage inputs (0..5 V or 0..10 V), inputs AI4 and AI5 are current inputs (0..20 mA / 4..20 mA).



PiXtend V2 Software Manual

To calculate voltages from the values of AI0 - AI3, the following formula is used:

Convert analog value to voltage

$\text{AnalogInX} * 10 / 1024 = \text{Voltage at channel X [V]}$

(Voltages are measured at AI0 - AI3)

The jumper must be observed for the voltage inputs. The "10" in the above calculation formula is used when no jumper is set (0..10 V range).

If the jumper is set for a channel (0..5 V range), a "5" instead of a "10" must be used in the formula.

To calculate currents from the values of AI4 - AI5, the following calculation formula is used:

Convert analog value to current

$\text{AnalogInX} * 0.020158400229358 = \text{Current in channel X [mA]}$

(Currents are measured at AI4 - AI5)



14.2.1.4.3 GPIOIn

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startvalue	0	0	0	0	0/1	0/1	0/1	0/1

Table 76: Bits of the GPIOIn Byte

The four GPIO inputs (GPIO_IN) are organized in one byte. The least significant bit (LSB) contains the state of GPIO0. Bit 7 (MSB) to bit 4 are not used.

The GPIO inputs have no defined rest state * (floating input). Only by the external application of a voltage does a stable state result:

0 V → Value „0“

5 V → Value „1“

Since the inputs are floating, they can be "0" or "1" without an external connection or wobble between the values.

The GPIOs can perform three different tasks: be an input / output or read the temperature / humidity from sensors like DHT11/22, AM2302. The configuration is done via the control byte GPIOCtrl. For more information, see the chapter 14.2.2 Control- & Status-Bytes.

After a reset or power-up of the device, the GPIOs are configured as inputs.

* There is the possibility to configure the GPIOs as inputs and to activate via GPIOOut so-called pull-up resistors. This causes a "1" at the input in the idle state (without floating inputs). For more information, see: 14.2.1.2.3 GPIOOut starting at page 272.



14.2.1.4.4 TempX (L/H)

TempXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 77: TempXL Byte

TempXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 78: TempXH Byte

In the 1-Wire / DHT-Mode of the GPIOs, the TempXL and TempXH bytes contain a 16-bit data word. This data word contains the temperature information of a sensor (if a sensor is connected and configured). The "X" stands for the number of the GPIO, ie "0" to "3".

The GPIOs can perform three different tasks: be an input / output or read temperature / humidity from sensors like DHT11/22, AM2302. The configuration is done via the control byte GPIOCtrl. For more information, see chapter 14.2.2 Control- & Status-Bytes starting at page 291.

The containing value can very easily be converted to a temperature value (floating-point value). However, a distinction must be made between DHT11 and DHT22 (TempX represents the 16-bit value):

DHT11: TempX / 256 = Floating point [°C]

Example: $5632 / 256 = 22,0\text{ °C}$ – DHT11 Sensors do not provide a decimal point!

DHT22: TempX / 10 = Floating point [°C]

Example: $224 / 10 = 22,4\text{ °C}$



PiXtend V2 Software Manual

Depending on the programming language and data type, a "typecast" must be carried out. Otherwise, after dividing by 10 (DHT22), there is no floating-point value but an integer value (as an example 22 instead of 22.4).

It is also possible to divide by 256 with the DHT11 sensors using a "right shift" by 8 digits.

Negative temperatures can also be measured with DHT22 sensors. To measured this, the MSB must be evaluated in TempXH. If the MSB is a "1", then it is a negative temperature.



14.2.1.4.5 HumidX (L/H)

HumidXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 79: HumidXL Byte

HumidXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 80: HumidXH Byte

In the 1-Wire / DHT-Mode of the GPIOs, the HumidXL and HumidXH bytes contain a 16-bit data word. This data word contains the humidity information of a sensor (if a sensor is connected and configured). The "X" stands for the number of the GPIO, ie "0" to "3".

The GPIOs can perform three different tasks: be an input / output or read temperature / humidity from sensors like DHT11 / 22, AM2302. The configuration is done via the control byte GPIOCtrl. For more information, see chapter 14.2.2 Control- & Status-Bytes starting at page 291.

The value can easily be converted to a humidity value (floating-point value). However, a distinction must be made between DHT11 and DHT22 (HumidX represents the 16-bit value):

DHT11: HumidX / 256 = Floating point [%] - relative humidity

Example: $10496 / 256 = 41,0 \%$ – DHT11 Sensors do not provide a decimal point!

DHT22: HumidX / 10 = Floating point [%] - relative humidity

Example: $417 / 10 = 41,7 \%$

Depending on the programming language and data type, a "typecast" must be carried out. Otherwise, after dividing by 10 (DHT22), there is no floating-point value but an integer value (as an example 22 instead of 22.4).

It is also possible to divide by 256 with the DHT11 sensors using a "right shift" by 8 digits.



14.2.1.4.6 *RetainDataInX*

Retain data consist of 64 bytes for each direction (RetainDataOut & RetainDataIn). RetainDataIn, which is described in this section, is the previously stored data which, after a power failure / voltage interruption, is transferred back from the PiXtend microcontroller to the Raspberry Pi.

After restarting the PiXtend system, the previously stored data is available in RetainDataIn, if the Retain Memory has previously been activated, before the power failure. For further information, see page 297.

The data is retained until new retain data is stored. The data can also be retained spanning any number of system power cycles if the Retain functionality is not reactivated.

If the Retain memory is activated again and a power supply interruption occurs, the data is overwritten by the current data in RetainDataOut.



14.2.2. Control- & Status-Bytes

PiXtend or the PiXtend microcontroller can be configured via control bytes. In addition, the microcontroller informs the Raspberry Pi using status bytes and, if necessary, of errors and warnings.

The following pages provide an overview of the control and status bytes. Each byte is then described in detail. The various possibilities are shown and exemplary calculations are carried out.

Should any questions remain unanswered, please contact us in our forum or by e-mail (support@pixtend.com). You will receive an answer as soon as possible and further information.



14.2.2.1 Control-Bytes im Überblick

Control bytes are transferred from the Raspberry Pi to the PiXtend microcontroller. This way you can influence the behavior of the microcontroller.

The available control bytes are:

- **ModelOut (P. 294)**
indicates which hardware version (model) the RPi software expects (e.g., CODESYS or pxdev)
- **UCMode (P. 294)**
specifies which transmission mode to use
- **UCCtrl (P. 295)**
basic settings of the microcontroller (Retain, Watchdog, SafeState ...)
- **DigitalInDebounce (P. 299)**
indicates whether and how the digital inputs should be debounced
- **GPIOCtrl (P. 302)**
configures the PiXtend GPIOs (input, output, DHT-Mode)
- **GPIODebounce (P. 303)**
indicates whether and how the GPIO inputs should be debounced
- **PWMYCtrl (P. 304)**
configures the PWM channels (mode, activation, frequencies)



14.2.2.2 Status-Bytes Overview

Status bytes are transferred from the PiXtend microcontroller to the Raspberry Pi. They contain important information about the state of PiXtend or the microcontroller.

The following status bytes are available:

- **Firmware (P. 308)**
informs about the microcontroller firmware version
- **Hardware (P. 308)**
informs about the PiXtend hardware version
- **Modelln (P. 308)**
informs about the PiXtend model
- **UCState (P. 309)**
contains the operating state & error codes of the microcontroller
- **UCWarnings (P. 310)**
contains warnings of the microcontroller



PiXtend V2 Software Manual

14.2.2.3 Description of the Control-Bytes

14.2.2.3.1 *ModelOut*

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	x	x	x	x	x	x	x	x

Table 81: *ModelOut* Byte

Using the byte *ModelOut*, the application software running on the Raspberry Pi informs the microcontroller of the PiXtend model it is expecting. This allows the microcontroller to decide whether it is the expected model and if not, return an error to the Raspberry Pi.

The content of *ModelOut* is an ASCII character. In the case of PiXtend V2 -L-, the byte contains the following value: ASCII character **L** (Capital letter L – 76 decimal / 4C hexadecimal).

For the user, there is usually no reason to change this value. As an example, drivers are provided with the CODESYS package "PiXtend V2 Professional for CODESYS", in which the model is fixed and can not be changed.

14.2.2.3.2 *UCMode*

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startvalue	0	0	0	0	0	0	0	0

Table 82: *Bits of the UCMODE* Byte

Currently (as of June 2018) there is only one transmission mode, the so-called "Auto-Mode". The byte is reserved for future use.

It is not necessary to enter a value for the operation of PiXtend. If a value is entered, this has no effect.



14.2.2.3.3 UC Ctrl

14.2.2.3.3.1 UC Ctrl0

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	WDE3	WDE3	WDE1	WDE0
Startvalue	0	0	0	0	0	0	0	0

Table 83: Bits of the UC Ctrl0 Byte

Bit 0..3 – WDE0..3 (WatchdogEnable0..3)

With the WDE0..3 bits the watchdog timer of the PiXtend microcontroller can be activated and set.

The default setting is that all bits are "0". Activation and setting of the watchdog is explained in the following table:

WDE 3	WDE 2	WDE 1	WDE 0	Status of Watchdog	Time
0	0	0	0	deactivated	deactivated
0	0	0	1	active	16 ms
0	0	1	0	active	32 ms
0	0	1	1	active	64 ms
0	1	0	0	active	0,125 s
0	1	0	1	active	0,25 s
0	1	1	0	active	0,5 s
0	1	1	1	active	1 s
1	0	0	0	active	2 s
1	0	0	1	active	4 s
1	0	1	0	active	8 s

Table 84: Watchdog Settings

If the watchdog is activated, communication between Raspberry Pi and PiXtend is monitored. If there is a pause longer than the set time between two valid cycles, the watchdog will trigger and put the microcontroller in the Safe State*. An invalid cycle (e.g.,



PiXtend V2 Software Manual

because of a CRC error) is categorized by the watchdog as if no cycle had been performed.

The use of the watchdog makes sense if the PiXtend control needs to be brought into a defined state in the event of a fault. An error may occur, e.g. when the application program stops responding on the Raspberry Pi and no longer transfers data to the microcontroller.

Recommendation

Do not enable the watchdog during the development of your user software. If, for example, a new program is transferred via CODESYS or the user software is debugged, then no transmission takes place for a certain time. So the watchdog always triggers again and again and a reboot of the system becomes necessary. This can lead to an unnecessary delay in your development process.

* Definition of "Safe State":

- All digital outputs & relays are switched off / put into the rest /idle state
- PWM outputs are switched to high-impedance (tri-state)
- Retain data is stored when Retain option is activated
- The "L1" status LED flashes after a fault if the LED was not deactivated (Error LED "L1"-Signals, page 311)
- The microcontroller or the PiXtend system must be restarted afterwards

If switching off the digital outputs for your application does not correspond to a safe state, external measures must be taken to counteract this behavior.



14.2.2.3.3.2 UC Ctrl1

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	GPUE	SLED	RE	RC	SAFE
Startvalue	0	0	0	0	0	0	0	0

Table 85: Bits of the UC Ctrl1 Byte

Bit 0 – SAFE (SafeState)

The SAFE bit, when activated by the value "1", can immediately set the microcontroller to the Safe State *. The default value is "0"

* Definition of "Safe State":

- All digital outputs & relays are switched off / put into the rest /idle state
- PWM outputs are switched to high-impedance (tri-state)
- Retain data is stored when Retain option is activated
- The "L1" status LED flashes after a fault if the LED was not deactivated (Error LED "L1"-Signals, page 311)
- The microcontroller or the PiXtend system must be restarted afterwards

If switching off the digital outputs for your application does not correspond to a safe state, external measures must be taken to counteract this behavior.

Bit 1 – RC (RetainCopy)

The RC bit can be used to configure which data is visible in the Retain input area (RetainDataIn0..63).

With the startvalue "0", the most recently stored data is transferred from the microcontroller to the Raspberry Pi (normal retention mode).

If the value "1" is set for the RC bit, the data sent last by the Raspberry Pi is returned. In the Retain area RetainDataIn0..63, then RetainDataOut0..63 is mirrored, but with a cycle delay (RetainCopy operation). The contents of the retain data is not lost, but is only displayed as long as the RC bit is "1".

For more information on PiXtend's Retain feature, see 6.3. Retain Memory.



Bit 2 – RE (RetainEnable)

With the RE bit, the Retain function of PiXtend can be activated. For this purpose, a "1" must be written into this bit. By default, RE is "0" (after a reset or power-up) and Retain is deactivated.

Recommendation

Activate Retain only if the functionality is really required. The number of Retain memory cycles can only be guaranteed up to 10,000 cycles.

Further information on the Retain function of PiXtend V2 -L- can be found in the chapter: 6 *Basic knowledge* on page 20 in this document.

Bit 3 – SLED (StatusLED)

The status LED "L1" can be deactivated by SLED bit if it is not required. By default, the status LED is active and can be deactivated by the value "1" in SLED.

Bit 4 – GPUE (GPIOPullUpEnable)

With the GPUE bit, the pull-up resistors of the PiXtend GPIOs can be enabled (these are, however, not yet active). The enable has an effect if the GPIOs are configured as inputs and a "1" is written in the byte GPIOOut for the respective GPIO.

A "1" in GPUE activates this functionality.

For more information about the PiXtend GPIOs, see: *GPIOOut* starting at page 272.



14.2.2.3.4 DigitalInDebounce

DigitalInDebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 86: Bits of the DigitalInDebounce01 Byte

DigitalInDebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 87: Bits of the DigitalInDebounce23 Byte

DigitalInDebounce45

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 88: Bits of the DigitalInDebounce45 Byte

DigitalInDebounce67

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 89: Bits of the DigitalInDebounce67 Byte



PiXtend V2 Software Manual

DigitalInDebounce89

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 90: Bits of the DigitalInDebounce89 Byte

DigitalInDebounce1011

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 91: Bits of the DigitalInDebounce1011 Byte

DigitalInDebounce1213

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 92: Bits of the DigitalInDebounce1213 Byte

DigitalInDebounce1415

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 93: Bits of the DigitalInDebounce1415 Byte

With the help of the DigitalInDebounce bytes the debouncing of the 16 digital inputs of PiXtend can be configured. By default the debouncing of the inputs is not active.

The debouncing is activated / deactivated together for two channels at a time. As a result, no interactions occur between the two channels, but these only receive the same debounce setting.



PiXtend V2 Software Manual

The number in a DigitalInDebounce byte corresponds to the number of cycles in which an electrical signal is present at the digital input and must remain constant, for the level change to be passed on to the application software (Raspberry Pi). This applies both to a change of the signal level from "0" (low) to "1" (high), as well as vice versa.

If the bytes are not changed or a "0" is written as a value, then no debouncing takes place and the application software receives a new value in each cycle.

Example:

The first two digital inputs (DI0 and DI1) should be debounced. Only changes to the digital inputs which are constantly applied for more than 100 ms should be visible. The cycle time (data exchange between PiXtend and Raspberry Pi) is 10 ms.

- The byte DigitalInDebounce01 is used
- Calculation of the value for DigitalInDebounce01: $100 \text{ ms} / 10 \text{ ms} = 10$

The byte is thus set to 10 (decimal) to get the desired effect. If the cycle time during the software development is changed, it must be remembered that the value for debouncing must also be adjusted.



14.2.2.3.5 GPIOCtrl

Bit	7	6	5	4	3	2	1	0
Name	SENS3	SENS2	SENS1	SENS0	IO3	IO2	IO1	IO0
Startvalue	0	0	0	0	0	0	0	0

Table 94: Bits of the GPIOCtrl Byte

Bits 0..3 – IO0..3

With the IO bits, the four PiXtend GPIOs can be configured as either a digital input or output. With the startvalue "0", all GPIOs are configured as inputs. For example, the bit IO3 is set to "1", GPIO3 becomes an output. The value is set via the data word GPIOOut.

Bits 4..7 – SENS0..3 (Sensor0..3)

The GPIOs can also be used for 1-wire sensors (DHT11, DHT22, AM2302) in addition to their input / output function. The upper four bits of the GPIOCtrl data word are available for this purpose. For example, the bit SENS3 is set to "1", then one of the afore mentioned sensors can be connected to GPIO3 and evaluated. The setting of the IO bit is no longer relevant. The SENSX bits have a higher priority compared to the IOX bits.

The results / measured values of the sensors are in TempXL / TempXH and HumidXL / HumidXH. The "X" corresponds to the number of the GPIO to which a sensor is connected.

Notice



Because time is required for communication with the sensors (approximately 25 ms), a minimum cycle time of 30 ms is needed if at least one sensor is used. It does not matter whether one or four sensors are queried.



14.2.2.3.6 GPIODebounce

GPIODebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 95: Bits of the GPIODebounce01 Byte

GPIODebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startvalue	0	0	0	0	0	0	0	0

Table 96: Bits of the GPIODebounce23 Byte

The debouncing of the GPIOs only works if they are configured as inputs. The GPIODebounce bytes are otherwise used exactly the same as the DigitalInDebounce bytes. Further information & examples can be found in the section: 14.2.2.3.4 DigitalInDebounce starting at page 299.



14.2.2.3.7 PWMYCtrl – für 16 Bit PWMs

The following descriptions refer to all 16-bit PWMYX channels (PWM0A, PWM0B, PWM1A, PWM1B, PWM2A & PWM2B).

The "Y" is replaced by the number of the group ("0", "1" or "2"). The "X" is replaced by the letter of the respective PWM channel ("A" or "B").

14.2.2.3.7.1 PWMYCtrl0

Bit	7	6	5	4	3	2	1	0
Name	PS2	PS1	PS0	EnableB	EnableA	-	MODE1	MODE0
Startvalue	0	0	0	0	0	0	0	0

Table 97: Bits of the PWMYCtrl0 Byte

Bit 0..1 – MODE0..1

The PWM outputs can be operated in four different modes. The mode is configured via the MODE bits. By default, PiXtend starts in servo mode (MODE0..1 = 00).

MODE1	MODE0	Mode-Name	Description
0	0	Servo-Mode	Both channels (A / B) are used to drive model-building servos - special signal form
0	1	Duty-Cycle-Mode	Both channels (A / B) have the same frequency, but independently adjustable duty cycle
1	0	Universal-Mode	Freely adjustable frequency & duty cycle for channel A, half frequency for channel B and 50% duty cycle
1	1	Frequency-Mode	Freely adjustable frequencies for both channels (A / B), duty cycle is always 50%

Table 98: Mode Bits of the PWMYCtrl0 Byte

For the normal user, Servo Mode and Duty Cycle Mode are sufficient. Universal and Frequency Modes can be useful and used for specific applications, but are more complex to configure and use.

Bit 3..4 – EnableA, EnableB

With the EnableA and EnableB bits, the respective channel can be activated. By default, the PWM channels are disabled. By writing a "1" into the respective bit the channel is activated and the PWM is visible at the output.



PiXtend V2 Software Manual

The PWM can initially be configured for the desired task (mode, frequency / value ...) and then activated. The activation can also happen in the same cycle.

Bit 5..7 – PS0..PS2 (Prescaler0..3)

The prescaler bits (PS bits) allow a rough setting of the PWM frequency. Depending on the PWM mode (see MODE bits), the PS bits have different effects.

The following table provides information about the effects of the PS bits:

PS2	PS1	PS0	Description	Base Frequency
0	0	0	PWM-Ausgänge deaktiviert	-
0	0	1	Vorteiler (Prescaler): 1	16 MHz
0	1	0	Vorteiler (Prescaler): 8	2 MHz
0	1	1	Vorteiler (Prescaler): 64	250 kHz
1	0	0	Vorteiler (Prescaler): 256	62,5 kHz
1	0	1	Vorteiler (Prescaler): 1024	15,625 kHz
1	1	0	Vorteiler (Prescaler): 1024	15,625 kHz
1	1	1	Vorteiler (Prescaler): 1024	15,625 kHz

Table 99: Prescaler-Bits of the PWMYCtrl0 Byte

As mentioned, the prescaler settings have different effects, depending on the mode you have set:

Mode	Description
Servo Mode	No impact
Duty Cycle Mode	Setting the prescaler or the basic frequency
Universal Mode	Setting the prescaler or the basic frequency
Frequency Mode	Setting the prescaler or the basic frequency

Table 100: PWM Mode

The rough setting of the frequency is done via the prescaler bits (PS0..2) in PWMYCtrl0, the fine adjustment via PWMYCtrl1L / H.

How the frequency / period duration can be adjusted precisely can be read in the section: 14.2.1.2.4 PWMYX – 16 Bit Resolution ab Seite 273.



PiXtend V2 Software Manual

Notice:



If the Frequency Mode is to be used (for advanced users), no DHT sensors may be connected or activated in the GPIOCtrl byte. The combination of Frequency Mode and DHT-sensors is not possible. If one or more DHT sensors have been activated on the PiXend GPIOs, the Frequency Mode can not be activated. The attempt to activate the Frequency Mode anyway, even though DHT sensors are used, results in the deactivation of the PWM outputs.



14.2.2.3.7.2 PWMYCtrl1 (L/H)

The PWMYCtrl1 bytes L and H contain a related 16 bit data word. PWM0Ctrl1L is the low byte and PWMYCtrl1H is the high byte.

The PWM frequency / period duration can be set with the two bytes (in PWM mode "Duty-Cycle" & "Universal"). The frequency is valid for both channels.

In the Duty Cycle Mode, the duty cycle can be configured independently for each channel. In Universal Mode, the duty cycle can only be set for channel A.

The PWMYCtrl1 bytes have no effect for the Frequency Mode.

PWMYCtrl1L

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startvalue	0	0	0	0	0	0	0	0

Table 101: PWMYCtrl1L Byte

PWMYCtrl1H

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startvalue	0	0	0	0	0	0	0	0

Table 102: PWMYCtrl1H Byte

Example of the PWM period of time in the "Duty Cycle" Mode:

The prescaler (PS0..2) is set to the value 64 and thus the basic frequency to 250 kHz. In addition, the Duty Cycle Mode and channel A are activated:

PWM0Ctrl0: 01101001b

The 16 bit data word PWM0Ctrl1 is set to value 1000:

PWM0Ctrl1L: 11101000b

PWM0Ctrl1H: 00000011b

$\text{PWM-Period} = \text{Microcontroller-Base Clock} / 2 / \text{Prescaler} / \text{PWM0Ctrl1}$

$\text{PWM-Period} = 16 \text{ MHz} / 2 / 64 / 1000 = 125 \text{ Hz}$



PiXtend V2 Software Manual

14.2.2.4 Description of the Status-Bytes

The status bytes inform the user or the user program running on the Raspberry Pi about the status of the microcontroller.

The status bytes can be queried via CODESYS, PiXtend C-Library or PiXtend-Python-Library V2 (PPLV2).

14.2.2.4.1 *Firmware*

The firmware byte contains a number for the firmware version. The number can be between 0 and 255. If you require support for your PiXtend V2 device, always provide the firmware version.

14.2.2.4.2 *Hardware*

The hardware byte contains the version number of your PiXtend board. With the version number, the user software can check whether the software / drivers matches the hardware.

Example

Hardware-Byte contains number 21
→ PiXtend Hardware Version 2.1

14.2.2.4.3 *ModelIn*

In the ModelIn byte, the microcontroller informs the user software about the PiXtend V2 model. With the model identification the user software can check, if the software / drivers fits the hardware.

The content of ModelIn is an ASCII character. In the case of PiXtend V2 -L- the byte contains the following value: ASCII character **L** (uppercase L – 76 decimal / 4C hexadecimal).



14.2.2.4.4 UCState

Bit	7	6	5	4	3	2	1	0
Name	ERR3	ERR2	ERR1	ERR0	-	-	-	RUN
Startvalue	0	0	0	0	0	0	0	0

Table 103: Bits of the UCState Byte

Bit 0 – RUN

By querying this bit, the user can check whether PiXtend is ready for operation ("1") or if a successful data transmission has been carried out. This can make sense after the start of the PiXtend system.

If the controller is put into the Safe State (for example by the watchdog), the bit contains a "0". However, this can no longer be queried because no communication is possible (until a restart is performed).

Bit 4..7 – ERROR0..3

The ERR bits inform about problems that the microcontroller can recognize and pass on to the user program. The following table describes the error codes and their cause:

ERR 3	ERR 2	ERR 1	ERR 0	Error Description
0	0	0	0	No Error
0	0	0	1	-
0	0	1	0	CRC Error - Data the payload data obtained from the Raspberry Pi is faulty
0	0	1	1	Data Block is to short Driver fault / Wrong model
0	1	0	0	PiXtend model does not match the expected and the actual model do not match (ModelIn and ModelOut are not equal)
0	1	0	1	CRC Error - Header the header data received from Raspberry Pi is faulty
0	1	1	0	SPI Frequency is to high the chosen SPI frequency is too high, transmission errors occur

Table 104: Classification of the Error-Bits in the UCState Byte



14.2.2.4.5 UCWarnings

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	W2	W1	W0	-
Startvalue	0	0	0	0	0	0	0	0

Table 105: Bits of the UCWarnings Byte

Bit 1..2 – W0..1 (RetainCRCErrror. RetainVoltageError)

The **WX** bits inform the user about warnings regarding the Retain functionality of PiXtend V2. In the user program, these bits can be checked to see whether the Retain memory functions correctly or not.

W0 – RetainCRCErrror

Is "1" if an error has been detected when checking the Retain memory.

W1 – RetainVoltageError

Is "1" if the current supply voltage of PiXtend is too low (less than 19 V). Retain storage can not be activated.

Bit 3 – W2 (I2CErrror)

The PiXtend microcontroller and the PWM controller are connected via the I²C bus. W2 is "1" if an error has occurred during the check (I²C / TWI) of the PWM data for PWM1A, PWM1B, PWM2A and PWM2B.



14.3. Error LED “L1“-Signals

Some error situation do not allow the micro-controller to communicate it's state to the Raspberry Pi and the governing software component. For these situations an LED (L1) was placed on the PiXtend V2 board to help the user to determine the source of a problem if there is no communication with the micro-controller or for example if the watchdog has triggered.

The meaning of the different LED (L1) signals can be found below.

State (Signal)	Meaning	Possible solutions
Off	Ready, no error	-
On / Blinking very fast	Communication error (SPI)	Check switch “SPI_EN”, check used software component for correct operation, only one program at a time is allowed to communicate with the micro-controller. In C programs the Input and Output structure variables must be declared globally or locally the keyword “static” is needed.
Blinking fast (0,2s)	Watchdog was triggered	Shutdown RPi and turn of any power supply (power cycle). The micro-controller needs to restart in order to get it to work again.
Fading (3s)	AC/Retain error	The supply voltage is too low and the Retain function is activated, no Retain data can be saved. Check if the supply voltage has 24 volts.
Pulsing (5s)	Shutdown (safe state)	The micro-controller is going or is in safe state. The bit 0 within the control byte UC Ctrl1 (page 250) was set.

Table 106: LED L1 Signals



15. Figure Index

Figure 1: Software - Win32DiskImager.....	22
Figure 2: Basic knowledge - Find RPi network address.....	25
Figure 3: Basic knowledge – All found Raspberry Pi's shown in one list.....	25
Figure 4: CODESYS Development System (Source: 3S).....	33
Figure 5: CODESYS - Visualization of the PiXtend V2 Demo Project.....	34
Figure 6: CODESYS - New Project.....	39
Figure 7: CODESYS - New Project - Device selection.....	40
Figure 8: CODESYS - PiXtendTest Project.....	40
Figure 9: CODESYS - Add Device.....	41
Figure 10: CODESYS - Add Device - SPI master.....	42
Figure 11: CODESYS - Add Device - PiXtend V2 -S-.....	43
Figure 12: CODESYS - Add Global Variable List.....	44
Figure 13: CODESYS - GVL - Declarations.....	45
Figure 14: CODESYS - PiXtend V2 -S- I/O-Mapping.....	46
Figure 15: CODESYS - Input Assistance.....	47
Figure 16: CODESYS - PiXtend V2 -S- I/O-Mapping - State.....	48
Figure 17: CODESYS - GPIO Parameter - GPIO24.....	48
Figure 18: CODESYS - GPIO I/O-Mapping - rpi_gpio24 variable.....	49
Figure 19: CODESYS - Task configuration - MainTask.....	50
Figure 20: CODESYS - PLC_PRG (CFC) - Input.....	51
Figure 21: CODESYS - CFC - 3 Inputs connected to 3 outputs.....	52
Figure 22: CODESYS - CFC - Parameterized.....	52
Figure 23: CODESYS - Communication - Scan Network.....	53
Figure 24: CODESYS - PiXtend V2 -S- I/O-Mapping online.....	54
Figure 25: CODESYS - PLC_PRG (CFC) - Demo-Program.....	55
Figure 26: CODESYS - Web-Visualization - Configuration.....	56
Figure 27: CODESYS - Visualization.....	57
Figure 28: CODESYS - Visualization - Element property.....	57
Figure 29: CODESYS - Visualization - Configuration of a rectangle.....	58
Figure 30: CODESYS - Visualization - Finished.....	59
Figure 31: CODESYS - DAC - Create new project.....	60
Figure 32: CODESYS - DAC - Choose device.....	61
Figure 33: CODESYS - DAC – Project overview.....	61
Figure 34: CODESYS - DAC - Add SPI device.....	62
Figure 35: CODESYS - DAC - Add SPI master.....	63
Figure 36: CODESYS - DAC - Configure SPI master.....	64
Figure 37: CODESYS - DAC - Add PiXtend V2 DAC device.....	65
Figure 38: CODESYS - DAC – I/O Mapping settings.....	66
Figure 39: CODESYS - DAC - Add Global Variable List.....	67
Figure 40: CODESYS - DAC - GVL variables.....	67



PiXtend V2 Software Manual

Figure 41: CODESYS - DAC - Variable mapping.....	68
Figure 42: CODESYS - DAC - GPIO 24 as output.....	68
Figure 43: CODESYS - DAC - GPIO 24.....	69
Figure 44: CODESYS - DAC - Assign variable rpi_gpio24.....	70
Figure 45: CODESYS - DAC - Scan network.....	71
Figure 46: CODESYS - DAC - WebVisu settings.....	72
Figure 47: CODESYS - DAC - WebVisu - Sliders.....	73
Figure 48: CODESYS - DAC - Sliders - Set scale end.....	73
Figure 49: CODESYS - DAC - WebVisu - Create text boxes.....	74
Figure 50: CODESYS - DAC - WebVisu - Finished.....	75
Figure 51: CODESYS - Notice on how to use both PiXtend V2 SPI devices.....	76
Figure 52: CODESYS - Circuit Diagram - RS232.....	77
Figure 53: CODESYS - Circuit Diagram - RS232.....	78
Figure 54: CODESYS - HTerm 0.8.1beta - RS232 Testprogram.....	81
Figure 55: CODESYS - Visualization - Test program - Serial Communication.....	82
Figure 56: CODESYS - CAN-Bus Test under Linux.....	91
Figure 57: CODESYS - CAN-Bus - Create new project.....	96
Figure 58: CODESYS - CAN-Bus Slave - Add PLC device.....	97
Figure 59: CODESYS - CAN-Bus Slave- Add Raspberry Pi.....	97
Figure 60: CODESYS - CAN-Bus Slave - Add device to PLC.....	98
Figure 61: CODESYS - CAN-Bus Slave - Add CAN-Bus device.....	99
Figure 62: CODESYS - CAN-Bus Slave - Add CANopen device.....	100
Figure 63: CODESYS - CAN-Bus Slave - Configure I/O area.....	101
Figure 64: CODESYS - CAN-Bus Slave - Set PDO.....	101
Figure 65: CODESYS - CAN-Bus Slave - Install own CAN device.....	102
Figure 66: CODESYS - CAN-Bus Slave - POU and Task Configuration.....	103
Figure 67: CODESYS - CAN-Bus Slave - Task Configuration.....	104
Figure 68: CODESYS - CAN-Bus Slave - CANopen I/O Mapping.....	104
Figure 69: CODESYS - CAN-Bus Slave - Baud rate.....	105
Figure 70: CODESYS - CAN-Bus Master - Add Raspberry Pi.....	106
Figure 71: CODESYS - CAN-Bus Master - Add CANbus.....	107
Figure 72: CODESYS - CAN-Bus Master - Add CANopen Manager.....	108
Figure 73: CODESYS - CAN-Bus Master - Add CAN-Bus slave.....	109
Figure 74: CODESYS - CAN-Bus Master - Configure CAN slave.....	110
Figure 75: CODESYS - CAN-Bus Test - Multiple Download.....	111
Figure 76: CODESYS - CAN-Bus Test - Master and Slave exchange data.....	112
Figure 77: CODESYS - Retain-Demo - Device tree.....	113
Figure 78: CODESYS - Retain - Variable declaration.....	114
Figure 79: CODESYS - Retain - Example program.....	115
Figure 80: CODESYS - Retain - Start value 13.....	116
Figure 81: CODESYS - Retain - arRetainDataOut[0] increased by 1 to 14.....	116
Figure 82: CODESYS - Retain - After Power-Cycle.....	116
Figure 83: CODESYS - Shutdown - Library Manager.....	118



PiXtend V2 Software Manual

Figure 84: CODESYS - Shutdown - Add library.....	119
Figure 85: CODESYS - Shutdown - Variables.....	119
Figure 86: CODESYS - Shutdown - Example program.....	120
Figure 87: CODESYS PiXtend V2 -S- SPI devices Parameter tab.....	121
Figure 88: CODESYS PiXtend V2 -L- SPI devices Parameter Reiter.....	126
Figure 89: CODESYS - GPIO24 configured as output.....	132
Figure 90: Linux-Tools - pixtendtool2s - Switch '-di'.....	142
Figure 91: Linux-Tools - pixtendtool2s - Switch '-ai'.....	142
Figure 92: Linux-Tools - pixtendtool2s - Switch '-sws'.....	143
Figure 93: Linux-Tools - pixtendtool2s - Switch '-sr'.....	143
Figure 94: Linux-Tools - pxauto2s - Main menu.....	146
Figure 95: Linux-Tools - pxauto2s - Menu DOut.....	148
Figure 96: Linux-Tools - pxauto2s.....	149
Figure 97: Linux-Tools - pixtendtool2s.....	149
Figure 98: FHEM - Activation of the sensor functionality.....	177
Figure 99: FHEM - Create Plot via LogFile.....	178
Figure 100: FHEM - Plot for the sensor values.....	178
Figure 101: Python - Source: https://www.python.org/ , The Python Brochure.....	182
Figure 102: Python - Raspbian PIXEL- Terminal icon found in the Taskbar.....	184
Figure 103: Python - Console command - PPLV2 folder.....	186
Figure 104: Python - PiXtend Python Library V2 Demo Program.....	187
Figure 105: Python - Short Demo Program.....	188
Figure 106: OpenPLC - Designed to be Open Source - www.OpenPLCProject.com	193
Figure 107: OpenPLC - OpenPLC_v3 installation was successful.....	197
Figure 108: OpenPLC - OpenPLC_v3 runtime start.....	197
Figure 109: OpenPLC Editor download.....	198
Figure 110: OpenPLC editor.....	198
Figure 111: OpenPLC - Download Hello World program.....	199
Figure 112: OpenPLC - Projec overview.....	200
Figure 113: OpenPLC - Hello World program.....	200
Figure 114: OpenPLC - Variables list.....	201
Figure 115: OpenPLC - Success notice.....	201
Figure 116: OpenPLC - Browser address bar.....	202
Figure 117: OpenPLC - Runtime web server.....	202
Figure 118: OpenPLC - Change user data.....	203
Figure 119: OpenPLC - Hardware selection.....	204
Figure 120: OpenPLC - Compiling Blank Program.....	204
Figure 121: OpenPLC - Programs menu.....	205
Figure 122: OpenPLC - Enter Program Info.....	206
Figure 123: OpenPLC - Compiling Hello World program.....	207
Figure 124: OpenPLC - Hello World program is stopped.....	208
Figure 125: OpenPLC - Hello World program is running.....	208
Figure 126: OpenPLC - Config0 / Task overview.....	212



16. Table Index

Table 1: Technical specifications – Retain System.....	20
Table 2: PiXtend V2 -S- CODESYS I/Os overview.....	125
Table 3: PiXtend V2 -L- CODESYS I/Os overview.....	131
Table 4: OpenPLC – Overview Hardware addresses of PiXtend V2 -S-.....	210
Table 5: OpenPLC – Overview Hardware addresses of PiXtend V2 -L-.....	211
Table 6: Bits of the DigitalOut Byte.....	218
Table 7: Bits of the RelayOut Byte.....	219
Table 8: Bits of the GPIOOut Byte.....	220
Table 9: PWM0XL Byte.....	221
Table 10: PWM0XH Byte.....	221
Table 11: PWM0XL Byte.....	222
Table 12: PWM0XH Byte.....	222
Table 13: PWM0AL Byte.....	224
Table 14: PWM0AH Byte.....	224
Table 15: Bits of the PWM0AL Byte.....	225
Table 16: Bits of the PWM0AH Byte.....	225
Table 17: Bits of the PWM1XL Byte.....	227
Table 18: Bits of the PWM1XH Byte.....	227
Table 19: Bits of the PWM1XL Byte.....	229
Table 20: Bits of the PWM1XH Byte.....	229
Table 21: Bits of the PWM1AL Byte.....	231
Table 22: Bits of the PWM1AH Byte.....	231
Table 23: Bits of the PWM1AL Byte.....	232
Table 24: Bits of the PWM1AH Byte.....	232
Table 25: Bits of the AnalogOutL Byte.....	235
Table 26: Bits of the AnalogOutH Byte.....	236
Table 27: Bits of the DigitalIn Byte.....	237
Table 28: AnalogInXL.....	238
Table 29: Bits of the AnalogInXH Byte.....	238
Table 30: Bits of the GPIOIn Byte.....	239
Table 31: TempXL Byte.....	240
Table 32: TempXH Byte.....	240
Table 33: HumidXL Byte.....	242
Table 34: HumidXH Byte.....	242
Table 35: ModelOut Byte.....	247
Table 36: Bits of the UCMODE Byte.....	247
Table 37: Bits of the UCCTRL0 Byte.....	248
Table 38: Watchdog Settings.....	248
Table 39: Bits of the UCCTRL1 Byte.....	250
Table 40: Bits of the DigitalInDebounce01 Byte.....	252
Table 41: Bits of the DigitalInDebounce23 Byte.....	252



PiXtend V2 Software Manual

Table 42: Bits of the DigitalInDebounce45 Byte.....	252
Table 43: Bits of the DigitalInDebounce67 Byte.....	252
Table 44: Bits of the GPIOCtrl Byte.....	254
Table 45: Bits of the GPIODebounce01 Byte.....	255
Table 46: Bits of the GPIODebounce23 Byte.....	255
Table 47: Bits of the PWM0Ctrl0 Byte.....	256
Table 48: Mode Bits of the PWM0Ctrl0 Byte.....	256
Table 49: Prescaler-Bits in the PWM0Ctrl0 Byte.....	257
Table 50: PWM Mode.....	257
Table 51: PWM0Ctrl1L Byte.....	259
Table 52: PWM0Ctrl1H Byte.....	259
Table 53: Bits of the PWM1Ctrl1L Byte.....	263
Table 54: Bits of the PWM1Ctrl1H Byte.....	263
Table 55: Bits of the UCState Byte.....	266
Table 56: Classification of the Error-Bits in the UCState Byte.....	266
Table 57: Bits of the UCWarnings Byte.....	267
Table 58: Bits of the DigitalOut0 Byte.....	269
Table 59: Bits of the DigitalOut1 Byte.....	269
Table 60: Bits of the RelayOut Byte.....	271
Table 61: Bits of the GPIOOut Byte.....	272
Table 62: PWMYXL Byte.....	274
Table 63: PWMYXH Byte.....	274
Table 64: PWMYXL Byte.....	275
Table 65: PWMYXH Byte.....	275
Table 66: PWMYAL Byte.....	277
Table 67: PWMYAH Byte.....	277
Table 68: Bits of the PWMYXL Byte.....	278
Table 69: Bits of the PWMYXH Byte.....	278
Table 70: Bits of the AnalogOutL Byte.....	281
Table 71: Bits of the AnalogOutH Byte.....	282
Table 72: Bits of the DigitalIn0 Byte.....	283
Table 73: Bits of the DigitalIn1 Byte.....	283
Table 74: AnalogInXL.....	284
Table 75: Bits of the AnalogInXH Byte.....	284
Table 76: Bits of the GPIOIn Byte.....	286
Table 77: TempXL Byte.....	287
Table 78: TempXH Byte.....	287
Table 79: HumidXL Byte.....	289
Table 80: HumidXH Byte.....	289
Table 81: ModelOut Byte.....	294
Table 82: Bits of the UCMode Byte.....	294
Table 83: Bits of the UC Ctrl0 Byte.....	295
Table 84: Watchdog Settings.....	295



PiXtend V2 Software Manual

Table 85: Bits of the UC Ctrl1 Byte.....	297
Table 86: Bits of the DigitalInDebounce01 Byte.....	299
Table 87: Bits of the DigitalInDebounce23 Byte.....	299
Table 88: Bits of the DigitalInDebounce45 Byte.....	299
Table 89: Bits of the DigitalInDebounce67 Byte.....	299
Table 90: Bits of the DigitalInDebounce89 Byte.....	300
Table 91: Bits of the DigitalInDebounce1011 Byte.....	300
Table 92: Bits of the DigitalInDebounce1213 Byte.....	300
Table 93: Bits of the DigitalInDebounce1415 Byte.....	300
Table 94: Bits of the GPIO Ctrl Byte.....	302
Table 95: Bits of the GPIODebounce01 Byte.....	303
Table 96: Bits of the GPIODebounce23 Byte.....	303
Table 97: Bits of the PWMYCtrl0 Byte.....	304
Table 98: Mode Bits of the PWMYCtrl0 Byte.....	304
Table 99: Prescaler-Bits of the PWMYCtrl0 Byte.....	305
Table 100: PWM Mode.....	305
Table 101: PWMYCtrl1L Byte.....	307
Table 102: PWMYCtrl1H Byte.....	307
Table 103: Bits of the UCState Byte.....	309
Table 104: Classification of the Error-Bits in the UCState Byte.....	309
Table 105: Bits of the UCWarnings Byte.....	310
Table 106: LED L1 Signals.....	311