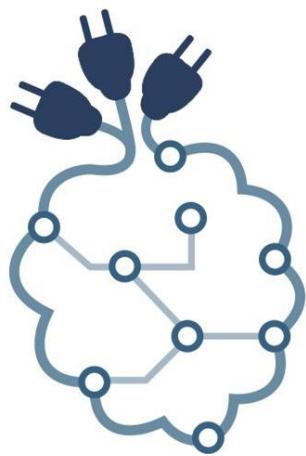


► SOFTWARE HANDBUCH



# PiXtend®

PIXTEND V2

## Versionshistorie

Version	Datum	Beschreibung	Bearbeiter
1.00	13.10.2017	Dokument erstellt, erste Version	RT
1.01	13.11.2017	CODESYS Package Name geändert auf PiXtend V2 Professional for CODESYS	RT
1.02	20.01.2018	Signalisierung für LED L1 hinzugefügt	RT
1.03	26.01.2018	Bezeichnung des SD-Image von „Linux Tools“ zu „Basis“ angepasst. Basis Image enthält mehr als nur die klassischen PiXtend Linux Tools	TG
1.04	13.02.2018	Informationen zu OpenPLC hinzugefügt	RT
1.05	20.02.2018	CODESYS E/A-Abbild Übersicht und Parameter Beschreibung für PiXtend V2 -S- hinzugefügt	RT
1.06	26.02.2018	CODESYS Retain Speicher Beispiel erstellt	RT
1.07	05.03.2018	Umstellung der Python Library V2 auf Version 0.1.1, hier wurde ein Fehler behoben, der die Verwendung der GPIO PullUps verhinderte.	RT
1.08	28.03.2018	Raspberry Pi 3 B+ wurde als kompatibles Modell hinzugefügt.	RT
1.09	18.07.2018	Informationen für PiXtend V2 -L- hinzugefügt	RT
1.10	31.07.2018	- PiXtend V2 -L- Prozessdaten + Control- & Statusbytes Abschnitt erstellt - UG (haftungsbeschränkt) in GmbH geändert	TG
1.11	09.08.2018	Aktualisierung des Kapitels zu OpenPLC von Version 2 zu Version 3 mit Unterstützung von PiXtend V2 -L- Erweiterung des CODESYS Kapitels um die Verwendung des PiXtend DAC Geräts	RT
1.12	10.08.2018	Korrekturen am Text und Änderung der Schriftart und Hervorhebung von Anweisungen für den Raspberry Pi	RT
1.13	09.01.2019	Raspberry Pi statische IP-Adresse zuweisen, In Kapitel 7.5 verwenden die Variablen jetzt den Modulnamen als Präfix „GVL“, Python Kurzübersicht um analoge Eingänge ergänzt, SPI Protokoll Übersicht im Anhang ergänzt für PiXtend V2 -S- und -L-	RT
1.14	02.12.2019	Raspberry Pi 4 aufgenommen, Aktivierung SPI-Kommunikation über GPIO24 ergänzt, Umstellung auf CODESYS Version V3.5 SP15 Patch 1, Hinweis eingefügt dass unter Python nur eine Instanz erlaubt ist, Anpassung des Installationsvorgangs zu wiringPi, Umstellung der seriellen Schnittstelle von ttyS0 zu ttyAMA0, Kapitel Basis Wissen erweitert um das Thema Cyber Security (IT-Sicherheit) Grundinformationen, die PPLv2 unterstützt auch Python 3	RT

# Inhaltsverzeichnis

1.	Hinweise zu dieser Dokumentation .....	8
1.1.	Gültigkeitsbereich.....	8
1.2.	Urheberschutz.....	8
1.3.	Wortmarken und Bildmarken.....	8
1.4.	Symbole.....	9
2.	Wichtige Erläuterungen .....	10
2.1.	Änderungsvorbehalt .....	10
2.2.	Bestimmungsgemäße Verwendung .....	10
2.3.	Technischer Zustand .....	11
2.3.1.	PiXtend V2 -S- Ausführung.....	11
2.3.2.	PiXtend V2 -L- Ausführung.....	12
2.4.	Zulassungen.....	13
2.5.	Sicherheitshinweise .....	14
2.6.	Haftungsausschluss.....	15
2.7.	Kontaktinformationen .....	15
2.8.	Hilfestellung erhalten.....	15
3.	Überblick .....	16
4.	Voraussetzungen .....	16
5.	Lizenzen .....	17
6.	Basis Wissen.....	18
6.1.	Definition „sicherer Zustand“ .....	18
6.2.	SPI-Kommunikation, Datenübertragung und Zykluszeit .....	18
6.3.	SPI-Kommunikation aktivieren .....	20
6.4.	Retain-Speicher (dt. Remanenz).....	21
6.5.	SD-Karte für Ihren Raspberry Pi vorbereiten .....	22
6.6.	Netzwerkadresse (IP-Adresse) vom Raspberry Pi ermitteln.....	24
6.7.	Raspberry Pi - Statische IP-Adresse einstellen .....	26
6.8.	Raspbian Anmeldedaten (Login) .....	27
6.9.	Den Raspberry Pi ausschalten.....	28
6.10.	Kompatibilität der Software-Komponenten.....	29
6.11.	Serielle Schnittstelle aktivieren.....	29
6.12.	PiXtend V2 - Hinweise zur seriellen Schnittstelle .....	30
6.13.	Hinweise zu Sicherheit und Cyber Security .....	32
6.13.1.	Passwort ändern .....	32
6.13.2.	Raspberry Pi im Internet.....	33
6.13.3.	Befehle und Skripte prüfen .....	33
6.13.4.	Backups des Systems erstellen.....	33
6.13.5.	Updates installieren .....	33
7.	CODESYS.....	34
7.1.	Hinweise .....	35
7.1.1.	Urheberrecht von Texten und Bildern:.....	35
7.1.2.	Kompatibilität.....	35
7.2.	Voraussetzungen .....	36
7.2.1.	Systemanforderungen für CODESYS .....	36
7.2.2.	Benötigte Hardware .....	36
7.3.	Installation der benötigten Software Komponenten .....	37
7.3.1.	CODESYS Entwicklungsumgebung .....	37
7.3.1.1	Vorstellung von CODESYS .....	37
7.3.1.2	CODESYS Download .....	38
7.3.1.3	CODESYS Installation .....	39
7.3.2.	CODESYS Control for Raspberry Pi installieren .....	39
7.3.2.1	CODESYS Control for Raspberry Pi - Download .....	39
7.3.2.2	CODESYS Control for Raspberry Pi – Installation .....	40
7.3.3.	Bootfähiges SD-Karten-Image für den Raspberry Pi erstellen .....	41
7.3.4.	PiXtend V2 CODESYS Gerätetreiber installieren .....	41
7.3.4.1	PiXtend V2 CODESYS Gerätetreiber - Download .....	41
7.3.4.2	PiXtend V2 CODESYS Gerätetreiber – Installation .....	41
7.4.	Weitere Schritte .....	41
7.5.	CODESYS – Projekt erstellen .....	42
7.5.1.	Schritt für Schritt zum ersten PiXtend V2 CODESYS Programm.....	42
7.5.1.1	CODESYS Standard Projekt für PiXtend V2 erzeugen.....	42
7.5.1.2	SPI Gerät anhängen .....	44

7.5.1.3	PiXtend V2 -S- Gerät anhängen .....	46
7.5.1.4	Globale Variablen Liste erzeugen.....	47
7.5.1.5	Mapping der Variablen .....	49
7.5.1.6	Erstellung des Hauptprogramms .....	52
7.5.1.7	Verbindung mit PiXtend V2 -S- und Programm Download.....	56
7.5.1.8	Weitere Schritte.....	58
7.5.2.	Schritt für Schritt zur ersten CODESYS Webvisu .....	59
7.6.	Schritt für Schritt zum ersten DAC Programm .....	63
7.6.1.	CODESYS Standard Projekt für PiXtend erzeugen .....	63
7.6.2.	SPI Gerät anhängen .....	65
7.6.3.	Globale Variablen Liste erzeugen .....	68
7.6.4.	Mapping der Variablen .....	69
7.6.5.	PiXtend V2 DAC Gerät anhängen.....	71
7.6.6.	Task Konfiguration.....	73
7.6.7.	Erstellung des Hauptprogramms .....	73
7.6.8.	Verbindung mit PiXtend V2 und Programm Download .....	74
7.6.9.	Erstellung der CODESYS Webvisu.....	75
7.7.	CODESYS Serielle Kommunikation .....	80
7.7.1.	Voraussetzungen .....	80
7.7.2.	RS232-Schnittstelle - Anschluss des Kabels .....	80
7.7.3.	RS485-Schnittstelle - Anschluss des Kabels (nur PiXtend V2 -L-) .....	81
7.7.4.	Vorbereitung Linux .....	82
7.7.4.1	Anpassung der Schnittstelleneinstellungen .....	82
7.7.4.2	Die Schnittstelle in der CODESYS-Konfiguration aktivieren .....	82
7.7.4.3	Raspberry Pi neu starten .....	83
7.7.5.	Terminal Programm .....	84
7.7.5.1	Terminal Programm installieren .....	84
7.7.5.2	Terminal öffnen und einstellen .....	84
7.7.6.	CODESYS .....	85
7.7.6.1	CODESYS Projekt „PiXtendSerialTest“ starten .....	85
7.7.6.2	Visualisierung .....	85
7.8.	CAN-Kommunikation .....	87
7.8.1.	Einführung .....	87
7.8.1.1	Voraussetzungen .....	88
7.8.1.2	Einschränkungen .....	88
7.8.2.	Hardware-Verbindung zum CAN-Bus .....	88
7.8.3.	Vorbereitung und Test unter Linux .....	89
7.8.3.1	Konfiguration des Device Tree Overlay .....	90
7.8.3.2	Blacklist bearbeiten .....	90
7.8.3.3	CAN Script anlegen .....	91
7.8.3.4	CAN aktivieren .....	92
7.8.3.5	CAN-Utilities installieren und verwenden .....	92
7.8.4.	Vorbereitungen für CODESYS .....	93
7.8.4.1	Start Skript anlegen .....	93
7.8.4.2	Baudrate Skript anlegen .....	93
7.8.4.3	CODESYS Control mit PiXtend V2 -L- CAN-Unterstützung starten .....	94
7.8.5.	CODESYS Projekt mit CANopen .....	95
7.8.5.1	PiXtend V2 -L- als CAN-Slave .....	96
7.8.5.2	PiXtend V2 -L- als CANopen-Master .....	105
7.8.5.3	Programm Download und Test .....	109
7.9.	CODESYS – PiXtend Retain Speicher .....	112
7.9.1.	Vorbereitung .....	112
7.9.2.	Programm erstellen .....	113
7.9.3.	Weitere Informationen .....	116
7.9.3.1	Datensicherheit erhöhen .....	116
7.9.3.2	Arbeiten mit Strukturen .....	116
7.10.	CODESYS – Raspberry Pi herunterfahren .....	117
7.11.	PiXtend V2 -S- SPI Device .....	119
7.11.1.	SPI Device Parameter .....	119
7.11.1.1	5 Volt/10 Volt Jumper Setting .....	120
7.11.1.2	GPIO Configuration .....	121
7.11.1.3	Mikrocontroller Settings .....	121
7.11.2.	I/O Übersicht .....	122
7.12.	PiXtend V2 -L- SPI Device .....	125

7.12.1. SPI Device Parameter.....	125
7.12.1.1 5 Volt/10 Volt Jumper Setting.....	125
7.12.1.2 GPIO Configuration.....	126
7.12.1.3 Mikrocontroller Settings.....	126
7.12.2. I/O Übersicht.....	127
7.13. FAQ – Häufig gestellte Fragen und Fehlerbehandlung.....	131
7.13.1. CODESYS Raspberry Pi Runtime und PiXtend V2.....	131
7.13.2. Serielle Kommunikation .....	132
7.13.2.1 Es werden nicht alle Zeichen übertragen oder kommen nacheinander an.....	132
7.13.2.2 Wozu werden im Programm die GPIOs 18 und 22 umgeschaltet? .....	132
7.13.2.3 Das Scrollen der Tabelle funktioniert nicht korrekt .....	133
7.13.2.4 Die Seite/Oberfläche wird nicht richtig angezeigt.....	133
7.13.2.5 Die erste Nachricht wird nicht richtig übertragen.....	133
7.13.2.6 Warum gibt es bei RS485 die „Auto send“-Funktion nicht?.....	133
7.13.2.7 Es werden nicht alle Baudraten angezeigt.....	133
8. pxdev – Linux Tools & Library .....	134
8.1. Hinweise .....	135
8.1.1. Einsatzbereiche pixtendtool2(s/l) – Einzelbefehle für PiXtend V2.....	135
8.1.2. Einsatzbereiche pxauto2(s/l) – GUI für PiXtend V2.....	135
8.1.3. Einsatzbereiche PiXtend/PiXtend V2 C-Library.....	136
8.2. Voraussetzungen .....	136
8.3. Verwenden des PiXtend V2 Basis Images .....	137
8.4. Manuelle Installation von pxdev.....	138
8.4.1. Vorbereitung und Installation .....	138
8.4.2. SPI-Bus aktivieren.....	139
8.5. Verwendung des pixtendtool2(s/l).....	140
8.5.1. Beispiele.....	141
8.5.2. Weitere Schritte.....	142
8.6. Verwendung von pxauto2(s/l).....	143
8.6.1. Navigation .....	145
8.6.2. Felder editieren.....	145
9. C-Library „pxdev“ .....	147
9.1. Voraussetzungen .....	148
9.2. Vorbereitung .....	148
9.2.1. Erstellen eines neuen Ordners und eines C-Files .....	148
9.2.2. Einbinden der Bibliotheken .....	149
9.3. Programmierung .....	149
9.4. Beispielprogramm für PiXtend V2 -S- .....	157
9.5. Beispielprogramm für PiXtend V2 -L- .....	159
9.6. Kompilieren und Ausführen des Programms.....	161
9.7. Frequently Asked Questions (FAQ).....	162
10. FHEM .....	163
10.1. Voraussetzungen .....	163
10.2. Installation.....	164
10.2.1. Installation von FHEM .....	164
10.2.2. Integration des Moduls .....	164
10.2.3. Anpassung des Systems .....	165
10.3. Modulbeschreibung.....	165
10.3.1. Anlegen eines neuen PiXtend V2-Geräts .....	165
10.3.2. Internals .....	165
10.3.3. Set-Kommandos .....	165
10.3.4. Get-Kommandos .....	167
10.3.5. Readings .....	168
10.3.6. Attribute .....	170
10.4. Beispiele .....	170
10.4.1. Darstellen von Sensorwerten .....	171
10.4.2. Ein- und Ausgänge entprellen und verknüpfen .....	173
10.4.3. Servo-Motor ansteuern .....	173
10.5. Frequently Asked Questions (FAQ).....	174
11. PiXtend Python Library V2 (PPLV2).....	175
11.1. Voraussetzungen .....	175
11.2. Installation mit PiXtend V2 Image .....	176
11.3. Installation auf dem Original-Raspbian .....	176
11.3.1. Vorbereitung .....	176
11.3.2. PiXtend Python Library V2 installieren .....	177
11.4. Programmierung .....	177

11.4.1.	Beispiel-Verzeichnis.....	177
11.4.2.	Beispiel ausführen.....	177
11.4.3.	Eigenes Programm erstellen.....	179
11.4.4.	Kurzübersicht.....	180
11.4.5.	Python Programm automatisch starten.....	181
11.4.6.	Verwendung der seriellen Schnittstelle.....	183
11.4.7.	Frequently Asked Questions (FAQ).....	183
12.	Anhang.....	185
12.1.	PiXtend V2 -S- .....	186
12.1.1.	Prozessdaten.....	186
12.1.1.1	Prozessdaten im Überblick.....	186
12.1.1.1.1	Prozessdaten die vom RPi an PiXtend übertragen werden.....	186
12.1.1.1.2	Prozessdaten die vom RPi an den PiXtend-DAC übertragen werden.....	186
12.1.1.1.3	Prozessdaten die von PiXtend an den RPi übertragen werden.....	186
12.1.1.2	SPI-Bus Protokoll Übersicht.....	187
12.1.1.3	SPI-Bus Konfiguration.....	188
12.1.1.4	Ausgangsdaten.....	189
12.1.1.4.1	DigitalOut.....	189
12.1.1.4.2	RelayOut.....	189
12.1.1.4.3	GPIOOut.....	190
12.1.1.4.4	PWM0X (L/H) – 16 Bit Auflösung.....	190
12.1.1.4.5	PWM0Ctrl – für 16 Bit PWMs.....	190
12.1.1.4.5.1.	Frequency-Mode .....	191
12.1.1.4.5.2.	Servo-Mode .....	192
12.1.1.4.5.3.	Duty-Cycle-Mode.....	193
12.1.1.4.5.4.	Universal-Mode .....	194
12.1.1.4.6	PWM1X (L/H) – 8 Bit Auflösung .....	194
12.1.1.4.6.1.	Servo-Mode .....	195
12.1.1.4.6.2.	Duty-Cycle-Mode.....	196
12.1.1.4.6.3.	Universal-Mode .....	197
12.1.1.4.6.4.	Frequency-Mode .....	198
12.1.1.4.7	RetainDataOutX.....	199
12.1.1.5	Ausgangsdaten – DAC .....	199
12.1.1.5.1	AnalogOutX (L/H) .....	199
12.1.1.6	Eingangsdaten .....	201
12.1.1.6.1	DigitalIn .....	201
12.1.1.6.2	AnalogInX (L/H) .....	202
12.1.1.6.3	GPIOIn .....	203
12.1.1.6.4	TempX (L/H).....	204
12.1.1.6.5	HumidX (L/H).....	205
12.1.1.6.6	RetainDataInX .....	205
12.1.2.	Control- & Status-Bytes.....	206
12.1.2.1	Control-Bytes im Überblick .....	206
12.1.2.2	Status-Bytes im Überblick .....	207
12.1.2.3	Beschreibung der Control-Bytes .....	208
12.1.2.3.1	ModelOut.....	208
12.1.2.3.2	UCMode.....	208
12.1.2.3.3	UCCtrl.....	209
12.1.2.3.3.1.	UCCtrl0 .....	209
12.1.2.3.3.2.	UCCtrl1 .....	211
12.1.2.3.4	DigitalInDebounce .....	212
12.1.2.3.5	GPIOCtrl .....	214
12.1.2.3.6	GPIODebounce .....	215
12.1.2.3.7	PWM0Ctrl – für 16 Bit PWMs.....	215
12.1.2.3.7.1.	PWM0Ctrl0 .....	216
12.1.2.3.7.2.	PWM0Ctrl1 (L/H) .....	218
12.1.2.3.8	PWM1Ctrl – für 8 Bit PWMs.....	219
12.1.2.3.8.1.	PWM1Ctrl0 .....	219
12.1.2.3.8.2.	PWM1Ctrl1 (L/H) .....	221

12.1.2.4	Beschreibung der Status-Bytes .....	223
12.1.2.4.1	Firmware .....	223
12.1.2.4.2	Hardware .....	223
12.1.2.4.3	ModelIn .....	223
12.1.2.4.4	UCState .....	224
12.1.2.4.5	UCWarnings .....	225
12.2.	PiXtend V2 -L- .....	226
12.2.1.	Prozessdaten .....	226
12.2.1.1	Prozessdaten im Überblick .....	226
12.2.1.1.1	Prozessdaten die vom RPi an PiXtend übertragen werden .....	226
12.2.1.1.2	Prozessdaten die vom RPi an den PiXtend-DAC übertragen werden .....	226
12.2.1.1.3	Prozessdaten die von PiXtend an den RPi übertragen werden .....	226
12.2.1.2	SPI-Bus Protokoll Übersicht .....	227
12.2.1.3	SPI-Bus Konfiguration .....	229
12.2.1.4	Ausgangsdaten .....	230
12.2.1.4.1	DigitalOutX .....	230
12.2.1.4.2	RelayOut .....	231
12.2.1.4.3	GPIOOut .....	232
12.2.1.4.4	PWMYX – 16 Bit Auflösung .....	232
12.2.1.4.4.1.	Servo-Mode .....	233
12.2.1.4.4.2.	Duty-Cycle-Mode .....	234
12.2.1.4.4.3.	Universal-Mode .....	236
12.2.1.4.4.4.	Frequency-Mode .....	237
12.2.1.4.5	RetainDataOutX .....	238
12.2.1.5	Ausgangsdaten – DAC .....	239
12.2.1.5.1	AnalogOutX (L/H) .....	239
12.2.1.6	Eingangsdaten .....	241
12.2.1.6.1	DigitalInX .....	241
12.2.1.6.2	AnalogInX (L/H) .....	242
12.2.1.6.3	GPIOIn .....	243
12.2.1.6.4	TempX (L/H) .....	244
12.2.1.6.5	HumidX (L/H) .....	245
12.2.1.6.6	RetainDataInX .....	246
12.2.2.	Control- & Status-Bytes .....	246
12.2.2.1	Control-Bytes im Überblick .....	247
12.2.2.2	Status-Bytes im Überblick .....	247
12.2.2.3	Beschreibung der Control-Bytes .....	248
12.2.2.3.1	ModelOut .....	248
12.2.2.3.2	UCMode .....	248
12.2.2.3.2.1.	UCCtrl .....	249
12.2.2.3.2.2.	UCCtrl0 .....	249
12.2.2.3.2.3.	UCCtrl1 .....	251
12.2.2.3.3	DigitalInDebounce .....	252
12.2.2.3.4	GPIOCtrl .....	254
12.2.2.3.5	GPIODebounce .....	255
12.2.2.3.6	PWMYCtrl – für 16 Bit PWMs .....	256
12.2.2.3.6.1.	PWMYCtrl0 .....	256
12.2.2.3.6.2.	PWMYCtrl1 (L/H) .....	258
12.2.2.4	Beschreibung der Status-Bytes .....	259
12.2.2.4.1	Firmware .....	259
12.2.2.4.2	Hardware .....	259
12.2.2.4.3	ModelIn .....	259
12.2.2.4.4	UCState .....	260
12.2.2.4.5	UCWarnings .....	261
12.3.	Fehler-LED „L1“ – Signalisierung .....	262
13.	Abbildungsverzeichnis .....	263
14.	Tabellenverzeichnis .....	265

## 1. Hinweise zu dieser Dokumentation

Bewahren Sie diese Dokumentation auf!

Diese Dokumentation ist Bestandteil des Produkts und während der gesamten Nutzungsdauer des Produkts aufzubewahren. Wird das Produkt weitergegeben oder veräußert, so ist dieses Dokument dem nachfolgenden Benutzer auszuhändigen; dies schließt gegebenenfalls erhaltene Erweiterungen zu dieser Dokumentation mit ein.

### 1.1. Gültigkeitsbereich

Die vorliegende Dokumentation gilt ausschließlich für die im Inhaltsverzeichnis genannten Softwarekomponenten und für PiXtend V2 Geräte in den folgenden Ausführungen:

- PiXtend V2 -S- Extension Board (Artikelnummer: 50199 004)
- PiXtend V2 -S- ePLC Basic (Artikelnummer: 50199 005 + 50199 013)
- PiXtend V2 -S- ePLC Pro (Artikelnummer: 50199 006 + 50199 014)
- PiXtend V2 -L- Extension Board (Artikelnummer: 50199 001)
- PiXtend V2 -L- ePLC Basic (Artikelnummer: 50199 002 + 50199 011)
- PiXtend V2 -L- ePLC Pro (Artikelnummer: 50199 003+50199 012)

#### Hinweis:

Wird im Dokumentationstext nur der Name PiXtend V2 verwendet, so gelten die genannten Informationen für beide Produkte (PiXtend V2 -S- und PiXtend V2 -L-) gleichermaßen.

Die Dokumente für die Vorgängerserie „PiXtend V1.x“ finden Sie auf unserer Homepage unter  
<https://www.pixtend.de/downloads/>

### 1.2. Urheberschutz

Diese Dokumentation, zusammen mit allen Texten und Bildern, ist urheberrechtlich geschützt. Für jede andere Verwenden, Übersetzung in andere Sprachen, Archivierung oder sonstige Veränderung muss die schriftliche Genehmigung der Kontron Electronics GmbH, D-72663 Großbettlingen, eingeholt werden.

Copyright 2018 © Kontron Electronics GmbH

### 1.3. Wortmarken und Bildmarken

- „Raspberry Pi“ und das zugehörige Logo sind geschützte Markenzeichen der Raspberry Pi Foundation – [www.raspberrypi.org](http://www.raspberrypi.org)
- „CODESYS“ und das zugehörige Logo sind geschützte Markenzeichen der Firma 3S-Smart Software GmbH – [www.codesys.com](http://www.codesys.com)
- „PiXtend“, „ePLC“ und das zugehörige Logo sind geschützte Markenzeichen der Firma Kontron Electronics GmbH – [www.kontron-electronics.de](http://www.kontron-electronics.de)
- „AVR“, „ATmega“ und die zugehörige Logo sind geschützte Markenzeichen der Atmel Corporation – [www.atmel.com](http://www.atmel.com) bzw. Microchip Technology Corporation [www.microchip.com](http://www.microchip.com)
- „Debian“ und „Raspbian“ sind geschützte Markenzeichen des Debian Project – [www.debian.org](http://www.debian.org)
- „I2C“ bzw. „I<sup>2</sup>C“ sind geschützte Markenzeichen von NXP Semiconductors – [www.nxp.com](http://www.nxp.com)
- „Arduino“ ist ein geschütztes Markenzeichen der Arduino AG – [www.arduino.cc](http://www.arduino.cc)

Die Rechte aller hier genannten Firmen und Firmennamen sowie Waren und Warennamen liegen bei den jeweiligen Firmen.

## 1.4. Symbole

**NOTICE**

NOTICE weist auf ein bestimmtes Merkmal hin.

**CAUTION**

CAUTION weist auf eine gefährliche Situation hin, die, wenn sie nicht vermieden wird, zu leichten oder mittleren Verletzungen führen kann.

**DANGER**

DANGER weist auf eine gefährliche Situation hin, die, wenn sie nicht vermieden wird, zu Tod oder schweren Verletzungen führt.

## 2. Wichtige Erläuterungen

Dieses Kapitel enthält Informationen zum Änderungsvorbehalt, der bestimmungsgemäßen Verwendung, dem technischen Zustand der PiXtend V2 Modelle, den Haftungsausschluss, den Sicherheitshinweisen und wo Sie Hilfestellung erhalten können.

### 2.1. Änderungsvorbehalt

Die Kontron Electronics GmbH behält sich vor diese Dokumentation in Teilen oder im Ganzen zu überarbeiten oder zu ändern, wenn dies dem technischen Fortschritt dient, bestehende Softwarekomponenten geändert werden müssen oder neue entstanden sind. Der Anwender findet die aktuelle Version dieser Dokumentation stets unter <https://www.pixtend.de/downloads/>.

### 2.2. Bestimmungsgemäße Verwendung

PiXtend V2 Geräte werden in Kombination mit dem Einplatinencomputer „Raspberry Pi“ (Raspberry Pi Foundation, UK registered charity 1129409) verwendet, der entweder bereits mit ausgeliefert wird (PiXtend V2 ePLC Basic / Pro) oder vom Kunden als Zubehör zu beschaffen ist (PiXtend V2 Extension Board).

Das PiXtend V2 System erfüllt die Funktion einer speicherprogrammierbaren Steuerung (SPS) bzw. einem elektrischen Mess-, Steuer-, Regel- und Laborgerät. Es kann Sensoren auswerten und Aktoren ansteuern. Die Logik-Programmierung zwischen Ein- und Ausgängen kann unter anderem mit der Software „CODESYS V3“ der Firma Smart Software Solutions GmbH geschehen. Kontron Electronics bietet noch weitere, aus den Bereichen IT und Home Automation stammende, Programmiersprachen und Systeme an, die der Kunde einsetzen kann. Dafür existieren jeweils Anleitungen und Beispiele von Kontron Electronics.

Die PiXtend V2 Geräte sind für trockene Innenräume ausgelegt – Schutzklasse IP20 (ePLC Pro), IP00 (Extension Board und ePLC Basic). Der Betrieb im Außenbereich und in Feuchträumen ist nicht gestattet, außer die PiXtend-Geräte werden in ein geeignetes Gehäuse eingebaut. Die Geräte sind nicht für explosionsgefährdete Bereiche oder sicherheitskritische Systeme/Anlagen ausgelegt.

PiXtend V2 Geräte dürfen gleichermaßen im industriellen/gewerblichen Umfeld, in Bildungseinrichtungen und im Wohnbereich eingesetzt werden.

PiXtend V2 bietet die Möglichkeit, unter bestimmten Voraussetzungen, gefährliche elektrische Spannungen zu schalten. Arbeit an gefährlichen Spannungen ist nur berechtigtem Fachpersonal erlaubt (die Voraussetzungen können länderspezifisch abweichen). Im Zweifel ist die Verwendung gefährlicher Spannungen untersagt. PiXtend ist für Personen ab 14 Jahren geeignet, die das Sicherheitsdatenblatt und die Handbücher gelesen und verstanden haben. In Bildungseinrichtungen ist der Betrieb von fachkundigem und berechtigtem Personal zu überwachen. Eingesetzte Stromversorgungen und Zubehörteile müssen eine Zulassung für das Land besitzen, in dem das Gesamtsystem mit PiXtend V2 eingebaut oder verwendet werden soll.

## 2.3. Technischer Zustand

### 2.3.1. PiXtend V2 -S- Ausführung

PiXtend V2 -S- wird, unabhängig von dessen Variante, mit einer definierten Konfiguration ausgeliefert:

- „SPI\_EN“ Schalter aktiviert → Kommunikation zwischen PiXtend und Raspberry Pi ist aktiviert
- „PI\_5V“ Schalter aktiviert → PiXtend und Raspberry Pi werden gemeinsam vom Spannungsregler auf PiXtend (Eingangsspannung 24V DC ±20%) versorgt. An den Raspberry Pi Computer wird keine separate Spannungsversorgung angeschlossen
- Alle digitalen Eingänge sind für den 24V Bereich konfiguriert (kein Jumper gesteckt)
- Alle analogen Eingänge sind für den 0...10V Bereich konfiguriert (kein Jumper gesteckt)
- Die Mikrocontroller-Firmware entspricht immer der neusten, von Kontron Electronics veröffentlichten Version. Die aktuelle Version kann auf unserer Homepage eingesehen werden

Bei den ePLC-Varianten sind zusätzliche Konfigurationen zu nennen:

ePLC Basic & Pro

- Inhalt SD-Karte
  - der Inhalt wird bei der Bestellung angegeben, z.B. „CODESYS Control Demo“ oder „Basis (C / Python / Node-RED)“ (im Shop)
  - Bei unseren Händlern wird immer die SD-Karte mit dem Inhalt „CODESYS Control Demo“ (2h begrenzte Laufzeit, zusätzliche Lizenz erforderlich) ausgeliefert
- ePLC Pro
- Gehäuse → Edelstahlhaube und Hutschienengehäuse vormontiert

Wenn Sie eine andere Ausführung oder eine andere Hardware- und Software-Kombination benötigen, richten Sie Ihre Anfrage bitte direkt an uns ( [info@pixtend.de](mailto:info@pixtend.de) ).

### 2.3.2. PiXtend V2 -L- Ausführung

PiXtend V2 -L- wird, unabhängig von dessen Variante, mit einer definierten Konfiguration ausgeliefert:

- „SPI\_EN“ Schalter aktiviert → Kommunikation zwischen PiXtend und Raspberry Pi ist aktiviert
- „PI\_5V“ Schalter aktiviert → PiXtend und Raspberry Pi werden gemeinsam vom Spannungsregler auf PiXtend (Eingangsspannung 24V DC ±20%) versorgt. An den Raspberry Pi Computer wird keine separate Spannungsversorgung angeschlossen
- Alle digitalen Eingänge sind für den 24V Bereich konfiguriert (kein Jumper gesteckt)
- Alle analogen Eingänge sind für den 0...10V Bereich konfiguriert (kein Jumper gesteckt)
- „CAN“ / „AO“ Jumper auf Stellung „AO“ → analoge Ausgänge aktiv, CAN inaktiv
- RS485 „A“ / „M“ Jumper auf Stellung „A“ → automatische Send-/Empfangsumschaltung
- Terminierungswiderstände für RS485 und CAN inaktiv (kein Jumper gesteckt)
- Die Mikrocontroller-Firmware entspricht bei Auslieferung immer der neusten, von Kontron Electronics veröffentlichten Version. Die aktuelle Version kann auf unserer Homepage eingesehen werden.

Bei den ePLC-Varianten sind zusätzliche Konfigurationen zu nennen:

ePLC Basic & Pro

- Inhalt SD-Karte
  - der Inhalt wird bei der Bestellung im Online Shop angegeben, z.B. „CODESYS Control Demo“ oder „Basis“ (C / Python / Node-RED)
  - Bei unseren Händlern wird immer die SD-Karte mit dem Inhalt „CODESYS Control Demo“ (2h begrenzte Laufzeit, zusätzliche Lizenz erforderlich) ausgeliefert

ePLC Pro

- Gehäuse → Edelstahlhaube und Hutschienengehäuse vormontiert

Wenn Sie eine andere Ausführung oder eine andere Hardware- und Software-Kombination benötigen, richten Sie Ihre Anfrage bitte direkt an uns ( [info@pixtend.de](mailto:info@pixtend.de) ).

## 2.4. Zulassungen



Dieses Produkt wurde in Übereinstimmung mit den geltenden europäischen Richtlinien entwickelt und hergestellt und trägt daher das CE-Zeichen. Der bestimmungsgemäße Gebrauch ist im vorliegenden Dokument beschrieben. Ein Sicherheitsdatenblatt liegt jedem Produkt in Papierform bei (mehrsprachig).

**Warnung:**

Änderungen und Modifikationen des Produkts, sowie die Nichteinhaltung der Angaben aus den Handbüchern und Sicherheitsdatenblättern führen zum Verlust der Zulassung.



Das Symbol der durchgestrichenen Mülltonne (WEEE-Symbol) bedeutet, dass dieses Produkt getrennt vom Hausmüll als Elektroschrott dem Recycling zugeführt werden muss. Wo Sie die nächste kostenlose Annahmestelle finden, erfahren Sie von Ihrer kommunalen Verwaltung.

## 2.5. Sicherheitshinweise

**CAUTION**

PiXtend V2 darf nicht in sicherheitskritischen Systemen eingesetzt werden.  
Prüfen Sie vor der Verwendung die Eignung von Raspberry Pi und PiXtend V2 für Ihre Anwendung. Die Standard-Einstellungen wurden so gewählt, dass die Anforderungen der meisten Anwender erfüllt werden.

**DANGER**

Beim Umgang und besonders beim Experimentieren mit den Prozessdaten ist Vorsicht geboten. Angeschlossene Sensoren und Aktoren können bei falscher Handhabung undefinierte Zustände einnehmen bzw. falsche Werte ausgeben.  
Wird mit PiXtend V2 eine Maschine, ein Gerät oder Prozess gesteuert oder geregelt, können so gefährliche Zustände eintreten. Machen Sie sich mögliche Gefahrenquellen frühzeitig bewusst.  
Trennen Sie im Zweifelsfall die Verbindungen zu den Geräten, Sensoren, Motoren usw. von der Spannungsversorgung ab, um Gefahren für Mensch und Maschine zu minimieren.  
Die Control-Bytes werden nicht dauerhaft gespeichert. Nach einem Reset oder Power-Cycle sind alle vorherigen Einstellungen gelöscht und das PiXtend V2 führt erst die nächste Aktion wieder aus, wenn der onboard Mikrocontroller dazu einen Befehl erhält.

**NOTICE**

Es wird empfohlen während der Entwicklung eines Steuerprogramms für das PiXtend V2 dieses nicht automatisch starten zu lassen.

## 2.6. Haftungsausschluss

Die Angaben in dieser Dokumentation wurden mit größter Sorgfalt zusammengetragen, geprüft und mit der hier beschriebenen Software und Hardware getestet. Dennoch können Abweichungen nicht ganz ausgeschlossen werden. Kontron Electronics GmbH haftet nicht für etwaige Schäden die unter Umständen durch die Verwendung der zur Verfügung gestellten Software, Softwarekomponenten, Hardware oder der in dieser Dokumentation beschriebenen Schritte entstehen können.

## 2.7. Kontaktinformationen

### Unsere Postanschrift:

Kontron Electronics GmbH  
Kantstraße 10  
D-72663 Großbettlingen

### So erreichen Sie uns:

Telefon: 07022 40570  
[info@kontron-electronics.de](mailto:info@kontron-electronics.de)  
[www.kontron-electronics.de](http://www.kontron-electronics.de)

## 2.8. Hilfestellung erhalten

Viele Informationen, Tipps und Tricks finden Sie in unserem Support-Forum unter: <https://www.pixtend.de/forum/>  
Sollten trotzdem Fragen offenbleiben, so bitten wir Sie zunächst in den FAQ's auf unserer Homepage nachzusehen.  
Wenn die Frage dort nicht geklärt wird, können Sie uns per E-Mail ([support@pixtend.de](mailto:support@pixtend.de)) in Kenntnis setzen. Sie erhalten schnellst möglich eine Antwort und weitere Informationen.

Die jeweils neusten Versionen aller Dokumente und Software-Komponenten finden Sie im Download-Bereich unserer Homepage: <https://www.pixtend.de/downloads/>

### 3. Überblick

PiXtend ist eine, auf dem leistungsfähigen Raspberry Pi Einplatinencomputer basierende, speicherprogrammierbare Steuerung (SPS / engl.: PLC). Das breite Spektrum digitaler und analoger Ein- & Ausgänge ermöglicht den Anschluss unterschiedlichster Sensoren und Aktoren aus der Industrie und dem Maker-Bereich. Die Verbindung zu anderen Geräten, Steuerungen und Computersystemen wird über serielle Standard-Schnittstellen (RS232/RS485, Ethernet, WiFi, CAN) hergestellt. Alle Schnittstellen und I/Os sind robust ausgeführt und entsprechen der SPS-Norm (IEC 61131-2).

### 4. Voraussetzungen

Die genauen Voraussetzungen an Hard- und Software finden Sie am Anfang jedes Kapitels separat aufgeführt, da diese je nach eingesetzter Softwarekomponente unterschiedlich sein können.

Im Allgemeinen wird für die meisten Arbeiten folgendes vorausgesetzt:

- Systemanforderungen für Entwicklungsrechner (PC)
  - Microsoft Windows 7/8/10 (32/64 Bit)
  - geeignete PC-Hardware für die entsprechende Microsoft Windows-Plattform
  - Ausreichend Festplattenplatz zur Installation neuer Programme
  - Entsprechende Rechte (z.B. Administratorrechte) die eine Installation eines neuen Programms erlauben
- Benötigte Hardware
  - PiXtend V2 Board ([www.pixtend.de](http://www.pixtend.de))
  - Raspberry Pi Modell B+/2 B/3 B/3 B+/4 B
  - Standard RJ-45 Netzwerkkabel
- Für die erste Inbetriebnahme eines neuen Raspberry Pi:
  - SD-Kartenleser, intern oder extern
  - SD-Speicherkarte, empfohlen mindestens 4 GB (besser 8 GB)  
micro SD für Raspberry Pi Modell B+/2 B/3 B/3 B+/4 B
  - optional: HDMI-fähiger Monitor
  - optional: USB-Keyboard
  - optional: USB-Maus

## 5. Lizenzen

Linux Tools:

Die Softwarekomponenten PiXtend V2 C-Library, pxauto2s, pxauto2l, pixtendtool2s und pixtendtool2l sind von Kontron Electronics GmbH unter der GNU GPL v3 Lizenz veröffentlicht worden und können frei genutzt und modifiziert werden. Der Einsatz in eigenen Projekten sowie zur kommerziellen Nutzung ist erlaubt.

Unsere Linux-Programme und Treiber bauen auf der WiringPi-Library von Gordon Henderson auf. Beachten Sie die jeweiligen Lizenzbestimmungen.



CODESYS:

Das „PiXtend V2 Professional for CODESYS Package“ für CODESYS V3.5 wird von Kontron Electronics GmbH kostenlos zur Verfügung gestellt und kann für private sowie kommerzielle Zwecke genutzt werden. Das Package kann frei verteilt und in beliebiger Anzahl verwendet werden.

Eine Veränderung des „PiXtend V2 Professional for CODESYS Packages“ oder einer der darin enthaltenen PiXtend V2 Treiberbibliothek ist nicht gestattet. Wenn Sie eine angepasste Version eines der PiXtend V2 Treibers benötigen, wenden Sie sich an Kontron Electronics GmbH.

Die CODESYS-Beispiele und die RC-Plug Bibliothek die ebenfalls im Package enthalten sind, wurden unter der GNU GPL v3 Lizenz veröffentlicht und können frei genutzt und modifiziert werden. Der Einsatz in eigenen Projekten sowie die kommerzielle Nutzung sind erlaubt.

Unsere CODESYS-Programme, Devices und Bibliotheken basieren auf der „CODESYS Control for Raspberry Pi“ und weitere Software-Komponenten der Firma 3S-Smart Software Solutions GmbH. Es gelten die entsprechenden Lizenzbedingungen der Firma 3S-Smart Software Solutions GmbH & CODESYS.

FHEM:

Die zur Verfügung gestellten Quelltexte für das FHEM System sind von Kontron Electronics GmbH unter der GNU GPL v3 Lizenz veröffentlicht worden. Unsere Programme und Treiber verwenden die WiringPi-Library von Gordon Henderson und FHEM-Module von Rudolf König. Es gelten die Lizenzbestimmungen des jeweiligen Rechteinhabers.

## 6. Basis Wissen

### 6.1. Definition „sicherer Zustand“

Das PiXtend V2 verfügt über einen vom Benutzer aktivierbaren Watchdog. Hat der Benutzer diesen aktiviert und es kommt zu einem Kommunikationsausfall bzw. der SPI-Bus ist gestört, dann löst der Watchdog nach der vom Benutzer eingestellten Wartezeit aus. Löst der Watchdog aus, stellt der Mikrocontroller seine Arbeit ein und bleibt in einem definierten Zustand stehen. Dieser Zustand wird in diesem Dokument als „sicherer Zustand“ bezeichnet und hat die nachfolgenden Eigenschaften:

- Alle digitalen Ausgänge und Relais werden abgeschaltet, in den Ruhezustand gebracht
- PWM-Ausgänge werden hochohmig (Tri-State) geschaltet
- Retain-Daten werden abgespeichert, wenn Retain aktiviert ist
- Die Status-LED „L1“ blinkt abhängig nach Fehlerursache (Fehler-LED „L1“ - Signalisierung), wenn die LED aktiviert ist
- Der Mikrocontroller bzw. das PiXtend V2 - System muss im Anschluss neu gestartet werden

Wenn das Ausschalten der digitalen Ausgänge für Ihre Anwendung nicht dem sicheren Zustand entspricht, so sind externe Maßnahmen vorzusehen, um dies abzufangen.

### 6.2. SPI-Kommunikation, Datenübertragung und Zykluszeit

Die Kommunikation zwischen Raspberry Pi und PiXtend V2 (Mikrocontroller) erfolgt mittels der SPI-Schnittstelle und wird standardmäßig mit einer Übertragungsgeschwindigkeit von 700 kHz ausgeführt. Die alleinige Betrachtung der benötigten Zeit der Datenübertragung ist nicht ausreichend, um festlegen zu können wie schnell tatsächlich mit dem Mikrocontroller kommuniziert werden kann. Das heißt, wie schnell kann nach der ersten SPI-Datenübertragung die nächste Übertragung sowie die weiteren Übertragungen erfolgen, dies bezeichnen wir als Zykluszeit.

Nach jeder Datenübertragung benötigt der Mikrocontroller etwas Zeit, um die empfangenen Daten zu verarbeiten. Erst im Anschluss darf die nächste Datenübertragung erfolgen, sonst besteht die Möglichkeit, dass der Mikrocontroller die neuen Daten nicht annimmt.

Für das PiXtend V2 -S- gelten folgende Zykluszeiten:

Bezeichnung	Zeit	Kommentar
Minimal	2,5 ms	Schnellstmögliche Zykluszeit, darf nicht unterschritten werden. Keine DHT11/22 Sensoren nutzbar.
Optimiert	10 ms	Empfohlene Zykluszeit, wenn keine DHT11/22 Sensoren verwendet werden.
Standard	30 ms	Empfohlene Zykluszeit, wenn alle Funktionen genutzt werden.

Für das PiXtend V2 -L- gelten folgende Zykluszeiten:

Bezeichnung	Zeit	Kommentar
Minimal	5 ms	Schnellstmögliche Zykluszeit, darf nicht unterschritten werden. Keine DHT11/22 Sensoren nutzbar.
Optimiert	10 ms	Empfohlene Zykluszeit, wenn keine DHT11/22 Sensoren verwendet werden.
Standard	30 ms	Empfohlene Zykluszeit, wenn alle Funktionen genutzt werden.

In allen Softwarekomponenten wurde die Zykluszeit Standard voreingestellt, damit ab Werk dem Anwender alle Funktionen ohne Einschränkung zur Verfügung stehen.

Die Zykluszeit kann durch den Anwender geändert werden. Die notwendigen Schritte für eine solche Änderung unterscheiden sich jedoch in Abhängigkeit der Softwarekomponente.

Die Linux-Tools für das PiXtend V2 müssen beispielsweise im Quellcode angepasst werden, ebenso die PiXtend Python Library V2.

Alle CODESYS Anwender hingegen können das Intervall des entsprechenden Tasks ändern und beim nächsten Einloggen sofort die Auswirkungen sehen.

### NOTICE

Der CODESYS Treiber des PiXtend V2 verfügt über einen Schutzmechanismus, der verhindert, dass der Mikrocontroller zu schnell angesprochen wird.

Stellt der Cycle Guard ein zu kurzes Buszyklustask-Intervall fest, wird im EA-Abbild von PiXtend V2 ein „BusCycleError“ angezeigt, es findet keine Kommunikation mit dem Mikrocontroller statt. Wählen Sie in diesem Fall ein längeres (langsameres) Intervall für den Buszyklustask.

### 6.3. SPI-Kommunikation aktivieren

Die SPI-Kommunikation zwischen Raspberry Pi und dem PiXtend V2 Board setzt eine Freigabe über den GPIO24 voraus, sonst kann keine Kommunikation stattfinden. Diese Vorrichtung wurde eingebaut, damit der Anwender entscheiden kann ob und wann ein Datenaustausch zwischen Raspberry Pi und PiXtend V2 erfolgen darf.

Der GPIO24 hat beim Raspberry Pi noch weitere Bezeichnungen, die je nach verwendeter Softwarekomponente genutzt werden. Die nachfolgende Tabelle zeigt diese Bezeichnungen, die von der Software wiringPi stammen.

Physikalischer Pin	wiringPi Name	wiringPi Nummer	BCM Anschlussname
18	GPIO.5	5	24 (GPIO24)

Tabelle 1: Raspberry Pi GPIO24 Bezeichnungen

Exemplarisch wird für die Sprachen C und Python, sowie für das Tool CODESYS gezeigt wie die Einstellungen vorzunehmen sind.

#### Beispiel C:

- wiringPiSetup();
- pinMode(5, OUTPUT);
- digitalWrite(5, 1);

#### Hinweis:

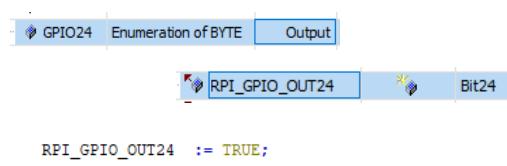
Für die Programmiersprachen C und Python stehen dem Anwender entsprechende Bibliotheken zur Verfügung, die diese Funktion bereits eingebaut haben. Es wird jedem Anwender empfohlen auf diese Bibliotheken zurückzugreifen. Weitere Informationen stehen in den entsprechenden Kapiteln in diesem Handbuch.

#### Beispiel Python:

- GPIO.setmode(GPIO.BCM)
- GPIO.setup(24, GPIO.OUT)
- GPIO.output(24, True)

#### Beispiel CODESYS V3.5:

- GPIO Gerät öffnen
- GPIO Parameter, GPIO24 als Ausgang
- GPIO E/A-Abbild, bei Bit24 im Kanal digital outputs einen Variablennamen eintragen
- Variable im Programm auf TRUE setzen



## 6.4. Retain-Speicher (dt. Remanenz)

Retain-Speicher bzw. Retain-Memory (engl.) wird im Deutschen ebenso als remanenter Speicher oder einfach als Remanenz bezeichnet. Dieser spezielle Speicher des PiXtend V2 ermöglicht es dem Anwender Werte von Variablen auch bei einem Spannungsausfall zu erhalten und nach einem Neustart bzw. Wiederanlauf dort fortzufahren, wo gestoppt wurde. Eine von Kontron Electronics entwickelte Technik ermöglicht es durch das PiXtend V2 Board den Raspberry Pi um diese besondere Funktion zu erweitern. Diese Speichertechnik ist eine bekannte Funktion in der Automatisierungstechnik, diese steht jedem PiXtend V2 Anwender zur Verfügung, unabhängig davon, welche Softwarekomponente zum Einsatz kommt.

Der PiXtend-Mikrocontroller überwacht die Spannung am Versorgungseingang („VCC“). Wurde das Retain System aktiviert und die Versorgungsspannung sinkt unter 19V DC, so wechselt der Controller in den sicheren Zustand und speichert die Retain-Daten ab. In der Zeitspanne zwischen dem Erkennen der Unterspannung bis zur vollständigen Speicherung der Daten wird PiXtend V2 und der Raspberry Pi aus einem internen Kondensator gespeist.

Das Retain System lässt sich nur aktivieren und verwenden, wenn mit einer nominellen Versorgungsspannung von 24V DC gearbeitet wird. Sofern PiXtend beispielsweise mit 12V DC betrieben wird, kann der Retain Speicher nicht genutzt werden.

Eigenschaft	Wert	Kommentar
Art des Speichers	Flash EEPROM	im PiXtend-Mikrocontroller
Anzahl Speicherzyklen	min. 10.000	
Speichergröße	- 32 Bytes (PiXtend V2 -S-) - 64 Bytes (PiXtend V2 -L-)	
Spannungsschwelle	19V DC (+/- 0,6V)	zwischen VCC und GND
Nenn-Versorgungsspannung	24V DC	um Retain Nutzen zu können
Max. Eingangsleistung über VCC	10W	für die garantierter Speicherung aller Daten

Tabelle 2: Technische Daten – Retain System

Weitere Informationen zur Verwendung des Retain Systems finden Sie im Hardware Handbuch zum entsprechenden PiXtend V2 Board.

Typische Anwendungen für den sogenannten Retain-Speicher sind unter anderem:

- Betriebsstundenzähler
- Gerätekonfiguration/Parameter
- Spracheinstellungen der Bedienoberfläche
- Motorenposition
- Bauteil- bzw. Werkstückzähler
- Werteverlauf Sicherung
- Festhalten des aktuellen Arbeitsschritts
- uvm.

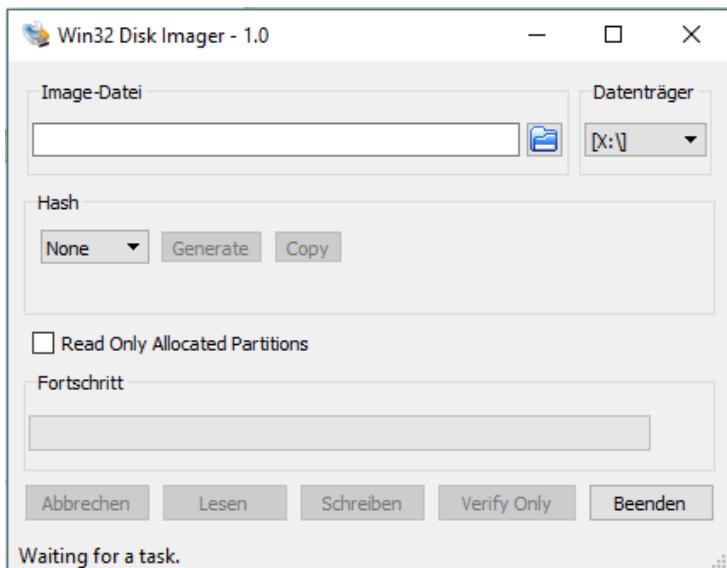
## 6.5. SD-Karte für Ihren Raspberry Pi vorbereiten

Haben Sie in unserem Shop eine vorinstallierte SD-Karte bestellt, dann können Sie sofort starten, das Erstellen einer eigenen SD-Karte ist nicht notwendig. Möchten Sie selbst eine SD-Karte für Ihren Raspberry Pi erstellen, finden Sie nachfolgend Informationen wie Sie dies tun können.

Wir stellen in unserem Downloadbereich unter <https://www.pixtend.de/downloads> verschiedene SD-Karten-Images zur Verfügung. Die „Images“ beinhalten je nach Typ eine aktuelle Version des Betriebssystems Raspbian und zusätzlich eine der folgenden vorkonfigurierten Optionen:

- PiXtend V2 Basis Image mit Linux-Tools inkl. der zugehörigen C-Library (pxdev), PiXtend Python Library V2 (PPLV2) inkl. Beispielen, Node-RED
- CODESYS Control Demo mit CODESYS V3 Runtime Erweiterung für den Raspberry Pi und PiXtend

Verwenden Sie für Ihre ersten Tests immer eine SD-Karte bzw. SD-Image von Kontron Electronics. Unsere Images werden ausführlich getestet und mit allen notwendigen Einstellungen vorbereitet. Auf diesem Weg erzielen Sie die schnellsten Fortschritte.



Um eines unserer Images auf eine leere SD-Karte zu übertragen, wird ein PC mit SD-Karten-Lesegerät benötigt. Als Software zum Schreiben von SD-Karten empfehlen wir das Open Source Programm „Win32DiskImager“. Das Programm läuft unter Microsoft Windows und kann unter folgender Adresse kostenlos heruntergeladen werden: <http://sourceforge.net/projects/win32diskimager/>

Abbildung 1: Software - Win32DiskImager

**NOTICE**

Zur Ausführung bzw. für den Laufwerkszugriff muss das Programm Win32DiskImager mit Administrator-Rechten bzw. unter einem Administratorkonto gestartet werden. Ist dies nicht der Fall, so besteht die Möglichkeit, dass die SD-Karte gar nicht oder fehlerhaft beschrieben wird!

Zur Erstellung einer SD-Karte bzw. zum Übertragen eines unserer Images auf Ihre SD-Karte führen Sie folgende Schritte aus:

- SD-Karte in das Kartenlesegerät einlegen oder einstecken
- Warten bis Windows das Medium erkannt hat
- Den Arbeitsplatz oder Windows Explorer öffnen und den Laufwerksbuchstaben notieren, dieser muss später im Win32DiskImager angegeben werden
- Die Zip-Datei, in der sich das SD-Karten Image von unserem Downloadbereich befindet, öffnen und entpacken.
- Das Programm Win32DiskImager mit Administrator-Rechten starten
- Die entpackte Image-Datei im Win32DiskImager unter „Image-Datei“ auswählen
- Unter Datenträger den Laufwerksbuchstaben der SD-Karte auswählen

**⚠ CAUTION**

Wählen Sie den falschen Datenträger aus, können alle Daten darauf verloren gehen, sobald Sie den Schreibvorgang mit Win32DiskImager gestartet haben.

Entfernen Sie am besten alle Wechseldatenträger (zum Beispiel USB-Sticks oder USB-Festplatten) bevor Sie den Schreibvorgang starten, sogar bevor Sie das Win32DiskImager Programm starten.

- Haben Sie alle Einstellungen vorgenommen, starten Sie den Schreibvorgang mit einem Klick auf „Schreiben“
- Warten Sie bis der Schreibvorgang abgeschlossen ist

Nach erfolgreicher Übertragung des Images auf die SD-Karte, kann diese am PC ausgeworfen und in den SD-Karten Slot des Raspberry Pi eingesteckt werden.

Sollten Sie während oder nach der Erstellung der SD-Karte ein Fenster sehen, mit oder ohne Inhalt, oder eine Fehlermeldung erhalten, dass auf ein Laufwerk nicht zugegriffen werden kann bzw. Sie werden aufgefordert das Medium zu formatieren, klicken Sie in solchen Fällen bei diesen Dialogen immer auf die „Abbrechen“ Schaltfläche und schließen Sie eventuell aufgegangene Fenster.

In einer solchen Situation liegt kein Fehler vor. Sie erhalten diese Meldungen, da Microsoft Windows nicht in der Lage ist die Daten der SD-Karte zu lesen, da diese für das Raspbian Betriebssystem bestimmt sind. Werfen Sie, wie erwähnt, die SD-Karte aus und stecken Sie diese in den SD-Karten Slot des Raspberry Pi. Schalten Sie die Versorgung ein und Ihr Raspberry Pi startet das Raspbian Betriebssystem.

## 6.6. Netzwerkadresse (IP-Adresse) vom Raspberry Pi ermitteln

Nach der Erstellung einer startfähigen SD-Karte für den Raspberry Pi und dessen erfolgreichen Start, stellt sich oft die Frage, welche Netzwerkadresse (IP-Adresse) hat der Raspberry Pi überhaupt? Das betrifft die Nutzer, die den Raspberry Pi ohne Bildschirm und Tastatur, also Headless (Kopflos), betreiben wollen.

Die Werkseinstellung der Netzwerkschnittstelle ist immer:

DHCP    Dynamic Host Configuration Protocol

Eine automatische Vergabe der Netzwerkadresse durch den haus- oder firmeneigenen Internet-Router oder DHCP-Server der IT.

Um die IP-Adresse des RPi herauszufinden gibt es mehrere Möglichkeiten.

1. Für alle:

Einmalig Bildschirm und Tastatur anschließen und schnell mit dem Befehl:

```
ifconfig
```

Auf dem Raspberry Pi nachschauen welche IP-Adresse er hat (Anschluss „eth0“).

Die meisten Internet Router merken sich die MAC Adresse (Geräte Adresse) des RPi und weisen ihm bei jedem Start die gleiche IP-Adresse zu.

Danach können Bildschirm und Tastatur entfernt werden, es kann am PC und SSH (z.B. Putty) weitergehen.

2. Für CODESYS Entwickler:

Wer den Raspberry Pi und vor allem PiXtend V2 als Steuerung (SPS) einsetzen möchte, kann die von 35 Smart Software Solutions eingebaute Raspberry Pi Suchfunktion im Raspberry Pi Update Fenster nutzen. Diese Suchfunktion durchsucht das ganze Heim- bzw. Firmennetzwerk nach Raspberry Pi Computern. Wurde ein oder mehrere Geräte gefunden, dann erscheinen alle IP-Adressen der gefundenen Geräte in einer Liste. In dieser Liste kann man entweder den gewünschten RPi auswählen und anschließend die IP-Adresse notieren oder zum Beispiel Informationen über das System abrufen.

Bitte beachten: Das Raspberry Pi Update Fenster lässt sich erst aufrufen, wenn das „CODESYS Control for Raspberry Pi“ Paket in CODESYS installiert ist. Schauen Sie dazu ins Kapitel 7.3.2 CODESYS Control for Raspberry Pi installieren.

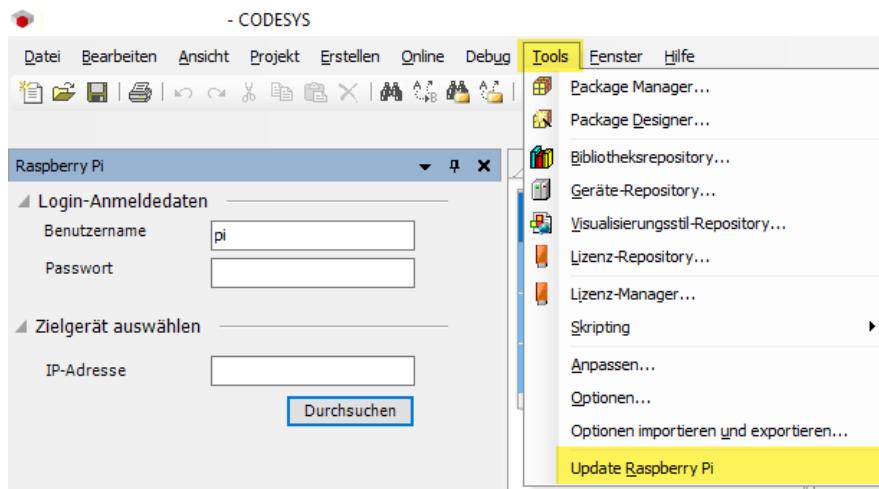


Abbildung 2: Basis Wissen - RPi IP-Adresse in CODESYS ermitteln

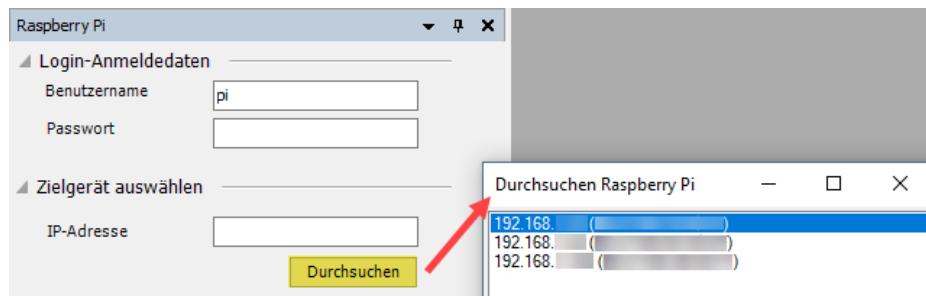


Abbildung 3: Basis Wissen - Alle im Netzwerk gefundenen Raspberry Pi's

Ist unklar welcher RPi welcher ist, dann am besten alle RPIs bis auf den gewünschten ausschalten. Im Anschluss CODESYS schließen, das Gateway neu starten und die Suche erneut starten. Manchmal dauert es ein wenig bis CODESYS die inaktiven Raspberry Pis vergisst, in diesen Fällen den PC/Windows neu starten und die Suche nochmals beginnen. Es sollte nur ein Gerät in der Liste erscheinen.

3. Wenn nichts weiterhilft:

Dann hilft vielleicht der Besuch der Info Seite der Raspberry Pi Foundation zur Ermittlung der IP Adresse eines Raspberry Pis. Sie können die Seite über folgende Adresse erreichen

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

Auf der Seite sind verschiedene Möglichkeiten aufgeführt, unter anderem Programme für Windows oder macOS, sogar Smartphone Apps werden genannt.

## 6.7. Raspberry Pi - Statische IP-Adresse einstellen

Der Raspberry Pi erhält seine Netzwerkeinstellungen ab Werk, von einem Router im lokalen Netzwerk oder von einem dedizierten DHCP Server. Dieser Zustand ist praktisch im Hausgebrauch, im professionellen Einsatz ist es dagegen wünschenswert, dass der Raspberry Pi eine feste und permanente Netzwerkadresse (IP-Adresse) erhält.

Die nachfolgenden Informationen stammen von der Raspberry Pi Learning Resources Webseite (in englischer Sprache). Wir empfehlen Ihnen immer zuerst auf dieser Webseite nachzuschauen, um die detailliertesten Informationen zu erhalten, da die Raspberry Pi Foundation ab und zu Änderungen an Betriebssystemeinstellungen vornimmt.

Verbinden Sie sich mit dem Raspberry Pi entweder direkt, mit Bildschirm und Tastatur oder per Netzwerk über ein SSH Client Programm (z.B. Putty).

**1. Die Datei „/etc/dhcpcd.conf“ bearbeiten:**

```
sudo nano /etc/dhcpcd.conf
```

**2. An das Ende der Datei folgendes anfügen:**

```
interface eth0

static ip_address=192.168.0.2/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

Fügen Sie anstelle der IP-Adresse 192.168.0.XXX die für Ihr Netzwerk gültige IP Adressen ein. Der Eintrag „ip\_address“ ist die feste IP-Adresse des Raspberry Pi, „routers“, ist die IP Adresse des Routers und „domain\_name\_servers“, ist die IP Adresse des DNS-Servers, meistens dieselbe IP Adresse wie die des Routers. Die 24 nach der IP-Adresse ist die Netzwerk-Maske und entspricht dem Wert 255.255.255.0.

**3. Änderungen speichern:**

Mit der Tastenkombination „STRG+O“ werden die Änderungen gespeichert und mit „STRG+X“ wird der nano Editor verlassen.

**4. Neustart durchführen:**

```
sudo reboot
```

Nach einem Neustart ist die fest eingestellte IP-Adresse des Raspberry-Pi aktiv geschaltet und ab jetzt über diese zu erreichen.

## 6.8. Raspbian Anmeldedaten (Login)

**NOTICE**

Bei unseren Images belassen wir die Raspbian Anmeldung so wie sie von der Raspberry Pi Foundation vorgegeben wird.

Die Login-Daten sind:

- Benutzer: pi
- Passwort: raspberry

## 6.9. Den Raspberry Pi ausschalten

Beim Raspberry Pi kommt immer wieder die Frage auf:  
„Kann ich einfach den Stecker ziehen, um den Raspberry Pi auszuschalten?“

Diese Frage lässt sich in etwa so beantworten:  
Der Raspberry Pi ist ein „vollständiger“ Computer auf einer Platine und wie jeder andere Computer mit Betriebssystem, kann der plötzliche Verlust der Stromversorgung zu Datenverlust führen.

Daher ist das Ziehen des „Steckers“ zwar jeder Zeit möglich, sollte aber vermieden werden, um die Integrität des Dateisystems und der Daten auf der SD-Karte zu gewährleisten. Ähnlich wie bei einem großen Computer kann es passieren, dass der Raspberry Pi nicht mehr startet und die SD-Karte neu bespielt werden muss, um sie wieder verwenden zu können.

Um den Raspberry Pi neu zu starten, kann folgender Befehl verwendet werden:

```
sudo reboot
```

Soll er hingegen ausgeschaltet werden, verwenden Sie diesen Befehl:

```
sudo shutdown -h now
```

Nach wenigen Sekunden hat das Betriebssystem alle Dienste gestoppt und einen sicheren Zustand eingenommen.  
Jetzt kann man bedenkenlos die Stromversorgung trennen, alle Daten bleiben auf der SD-Karte erhalten.

### Anmerkung:

Ein Ziehen des Steckers oder die plötzliche Unterbrechung der Stromversorgung beim Raspberry Pi muss nicht zwangsläufig zu Datenverlust beziehungsweise einer unbrauchbaren SD-Karte führen. Die Informationen auf dieser Seite geben lediglich den Hinweis, dass wie bei jedem anderen Computer die Möglichkeit eines Ausfalls besteht, jedoch nicht eintreten muss.

Sollten Sie erhöhte Anforderungen haben, nehmen Sie mit uns Kontakt auf, wir helfen Ihnen Ihre PiXtend Anwendung zu optimieren und eine höhere Datensicherheit zu erreichen.

## 6.10. Kompatibilität der Software-Komponenten

Für das PiXtend V2 gibt es viele Softwarekomponenten, die einen vielfältigen Einsatz der Plattform erlauben. Beim Einsatz dieser Komponenten muss folgendes beachtet werden.

### NOTICE

Die in diesem Dokument aufgeführten Programme und Softwarekomponenten können nicht parallel betrieben werden!

Der mit dem Mikrocontroller zur Kommunikation verwendete „SPI-Bus“ darf immer nur von einem Programm verwendet werden. Wird der Mikrocontroller gleichzeitig von verschiedenen Programmen angesprochen, mit unterschiedlichen Befehlen, kann dies zu fehlerhaften Daten führen sowie zu defekten Teilen der Hardware, zum Beispiel der Relais.

Eine Nutzung verschiedener Programme „hintereinander“ ist möglich, sofern es sich um die „pxdev-Linux Tools & Library“ handelt oder um die „PiXtend Phyton Library V2 (PPLV2)“. Das sind Programme, die sich nach ihrer Ausführung selbst beenden und manuell neugestartet werden müssen.

Wurde auf der SD-Karte die CODESYS Runtime Erweiterung installiert oder Sie verwenden unser CODESYS SD-Karten Image, lässt sich kein anderes Programm nutzen um PiXtend V2 zusätzliche Befehle zu schicken oder Informationen abzufragen. Die Nutzung ist hier exklusive für CODESYS reserviert, ferner wird CODESYS bei jedem Start des Raspberry Pi automatisch gestartet.

Setzen Sie das FHEM System ein und steuern Sie Ihr PiXtend V2 über FHEM, so gelten die gleichen Einschränkungen wie für CODESYS. FHEM wird nach seiner Installation ebenfalls automatisch gestartet da es sonst zu Komplikationen kommen kann. Eine parallele Nutzung von FHEM und CODESYS ist nicht möglich, es kommt hier zu Überschneidungen beim SPI-Bus.

## 6.11. Serielle Schnittstelle aktivieren

Damit die serielle Schnittstelle des Raspberry Pi Computers für eigene Zwecke genutzt werden kann, sollte diese zuerst aktiviert und konfiguriert werden. Das heißt, verwenden Sie die serielle Schnittstelle anderweitig, so wird mit den nachstehenden Befehlen die serielle Schnittstelle aktiviert beziehungsweise umgestellt. Sie sendet ab sofort selbstständig keine Daten aus, der Zugriff auf die Linux-Konsole über die serielle Schnittstelle wird durch diesen Vorgang deaktiviert.

Die serielle Schnittstelle des Raspberry Pi kann über das Programm „raspi-config“ aktiviert bzw. umgestellt werden:

```
sudo raspi-config
```

Im Programm wechseln Sie zu:

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

Im Anschluss findet sich in der Datei /boot/config.txt folgender Eintrag:

```
enable_uart=1
```

Die UART Schnittstelle wird aktiviert. Das Programm „raspi-config“ nimmt uns hier die komplette Arbeit ab.

Führen Sie im Anschluss einen Neustart des Raspberry Pi Computers durch, danach können Sie die serielle Schnittstelle exklusiv in eigenen Anwendungen nutzen.

Einen Neustart führen Sie durch folgenden Befehl aus:

```
sudo reboot
```

## 6.12. PiXtend V2 - Hinweise zur seriellen Schnittstelle

Um mit unseren PiXtend eIO Geräten sowie mit anderen seriellen Geräten uneingeschränkt zu arbeiten empfehlen wir die Verwendung der vollwertigen seriellen Schnittstelle des Raspberry Pi. Diese wird auch PL011 oder serial0 genannt und hat den Gerätenamen ttyAMA0. Nur diese Schnittstelle ermöglicht die Verwendung des Paritätsbit in der seriellen Kommunikation.

Alle unsere SD-Karten Images ab Version 2.1.6.0 für das Basis Image und das PiXtend V2 -S- CODESYS Image bzw. Version 2.1.1.L für das PiXtend V2 -L- CODESYS Image, verwenden die ttyAMA0 Schnittstelle zur Kommunikation. Die ttyS0 Schnittstelle wird nicht mehr verwendet.

Sie möchten prüfen, ob Sie bereits ein geändertes Image verwenden. Schauen Sie am besten in die config.txt Datei in der Raspbian Bootpartition.

Stecken Sie die SD-Karte unter Windows in einen Kartenleser ein und öffnen Sie die SD-Karte mit dem Windows Explorer. Prüfen Sie, ob die Datei config.txt ganz am Ende den Eintrag dtoverlay=pi3-disable-bt enthält.

Um dies am Raspberry Pi direkt zu prüfen, verwenden Sie folgenden Befehl:

→ sudo nano /boot/config.txt

Gehen Sie ans Ende der Datei und schauen Sie dort nach dem Eintrag dtoverlay=pi3-disable-bt. Ist er vorhanden, dann verwenden Sie bereits das Gerät ttyAMA0 als serielle Schnittstelle und müssen nichts weiter tun.

```
dtoverlay=audio-off
enable_uart=1
dtoverlay=pi3-disable-bt
```

Abbildung 4: RPi - Bluetooth ausschalten

Ist dieser Eintrag nicht vorhanden, dann fügen Sie ihn ein und führen folgende Schritte aus:

- sudo nano /boot/config.txt
- Ganz ans Ende der Datei gehen
- Fügen Sie ein: dtoverlay=pi3-disable-bt
- Speichern Sie die Änderungen mit Strg+O → Eingabetaste
- Verlassen Sie den Editor mit Strg+X
- Einen Neustart durchführen: sudo reboot

Wenn man mit CODESYS V3.5 arbeitet und serielle Geräte ansteuern möchte, so ist nach der vorherigen Änderung zusätzlich eine weitere Anpassung notwendig, damit CODESYS V3.5 die richtige serielle Schnittstelle verwendet.

- Folgenden Befehl ausführen: sudo nano /etc/CODESYSControl\_User.cfg
- In der Datei CODESYSControl\_User.cfg den Abschnitt [SysCom] suchen
- Ändern Sie den Abschnitt wie folgt ab von:  
`[SysCom]
;Linux.Devicefile=/dev/ttyS
;portnum := COM.SysCom.SYS_COMPORT1`
- hin zu  
`[SysCom]
Linux.Devicefile=/dev/ttyAMA
portnum := COM.SysCom.SYS_COMPORT1`
- D.h. entfernen Sie die beiden Semikolons und ändern Sie den Gerätenamen von ttyS hinzu ttyAMA, fügen Sie keinesfalls eine Zahl am Ende des Gerätenamens an, dies macht später CODESYS selbst.
- Speichern Sie die Änderungen mit Strg+O → Eingabetaste
- Verlassen Sie den Editor mit Strg+X

- Starten Sie den Raspberry Pi neu mit: sudo reboot

```
GNU nano 2.7.4          Datei: /etc/CODESYSControl_User.cfg

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

Abbildung 5: CODESYSControl\_User.cfg - Serieller Anschluss ttyAMA

Denken Sie ferner daran auf der serial0 bzw. ttyAMA0 Schnittstelle die sogenannte Login-Shell zu deaktivieren.

**NOTICE**

Bluetooth kann nach dieser Änderung nicht mehr verwendet werden!

## 6.13. Hinweise zu Sicherheit und Cyber Security

Wer heutzutage ein Computersystem, einen eigenen Rechner oder einen Raspberry Pi Computer betreibt, sollte sich Gedanken um die Sicherheit des Systems machen und Vorkehrungen treffen. Dies gilt insbesondere für Geräte, die über einen Netzwerkanschluss und/oder ein WLAN Modul verfügen. Es spielt keine Rolle ob das betreffende System im privaten Haushalt, im Büro einer Firma oder als Teil einer Anlage in einer Produktionseinrichtung betrieben wird.

Nachfolgend geben wir Empfehlungen und Denkanstöße, was Sie beim Einsatz eines Raspberry Pi in Bezug auf IT-Sicherheit tun ist, um die Systemsicherheit zu erhöhen oder zu verbessern. Bitte beachten Sie, wir zeigen nur einen kleinen Auszug an Möglichkeiten. Wir empfehlen allen Anwendern sich weiterhin ausführlich über das Internet zu informieren. Nutzen Sie unsere Empfehlungen und Denkanstöße beispielsweise als Ausgangspunkt für Ihr Suche. Beim industriellen Einsatz von PiXtend-Geräten ist Rücksprache mit der internen IT- bzw. Cyber Security Abteilung zu halten. Diese kennt sich in der Regel mit Linux-Systemen, dazu gehört der Raspberry Pi Computer, bestens aus. Ist dies nicht der Fall, so ist es sinnvoll externe Firmen oder Berater zu diesen Themengebieten zu befragen. Wird PiXtend im „Inselbetrieb“ verwendet, keine Verbindung zu einem Netzwerk hergestellt und die Wireless-Schnittstellen deaktiviert, so ist die Umsetzung einer sicheren IT-Lösung wesentlich einfacher.

### 6.13.1. Passwort ändern

Der erste und vermutlich wichtigste Punkt ist die Änderung des Passwortes des „pi“ Benutzers. Die Standardlogininformationen sind für jedes neu installierte Raspbian immer gleich, sie sind im Internet verfügbar und wir stellen sie in dieser Dokumentation bereit. Nutzen Sie nach dem ersten Start sofort den Linux Befehl „passwd“ um das Passwort des „pi“ Benutzers zu ändern.

Soll beispielsweise der RPi über SSH erreichbar sein, lässt sich der Login von einer Passwortauthentifizierung auf eine Zertifikatsauthentifizierung umstellen. Zusätzlich kann ein neuer Benutzer, ausgestattet mit den gleichen Rechten wie der „pi“ Benutzer, angelegt werden. Im Anschluss wird der „pi“ Benutzer gelöscht. Weitergehende Informationen finden Sie im Internet.

## 6.13.2. Raspberry Pi im Internet

Wenn der Raspberry Pi über das Internet erreichbar ist, dann ist höchste Vorsicht geboten. Unabhängig davon ob der Raspberry Pi im Heimnetzwerk oder in einem Firmennetz betrieben wird, vermeiden Sie die direkte Erreichbarkeit des RPi. Ist es dennoch notwendig, nutzen Sie beispielsweise die Port-Weiterleitungsfunktion der heimischen Fritzbox oder fragen Sie in Ihrer IT-Abteilung an, welche Möglichkeiten in Ihrer Firma vorhanden sind. Diese Lösung ist nur für sehr Fortgeschrittene zu empfehlen und ein Großteil der Firmen wird eine direkte Erreichbarkeit nicht erlauben.

Setzen Sie stattdessen auf eine VPN-Anbindung und nutzen Sie diese, um von unterwegs auf Ihren Raspberry Pi zuzugreifen. Heimanwender, die über eine Fritzbox verfügen, können sich recht einfach einen VPN-Zugang einrichten.

## 6.13.3. Befehle und Skripte prüfen

Das Internet bietet Informationen und Tipps rund um den Raspberry Pi, nicht zuletzt wegen der immer größer werdenden Community. Leider gibt es unter all diesen Angeboten auch nicht vertrauenswürdige Quellen. Beim Experimentieren und Ausprobieren mit dem Raspberry Pi neigt man gerne dazu, Befehle von einer Webseite oder einer anderen Quelle zu kopieren, ohne zu hinterfragen was dieser Befehl tut. Erfordert der Befehl zudem Superuser-Rechte (root-Rechte), das heißt man muss „sudo“ verwenden, kann schnell etwas schief gehen. Bei unbekannten Befehlen aus unbekannten Quellen sollte zuerst geprüft werden, um welchen Befehl es sich handelt und welche Auswirkungen mit dem Einsatz des Befehls verbunden sind. Wird man aufgefordert etwas herunterzuladen, zum Beispiel mit wget oder curl, sollte man überlegen, ob die Quelle vertrauenswürdig ist.

## 6.13.4. Backups des Systems erstellen

Sicherungen der SD-Karte des Raspberry Pi lassen sich recht einfach mit dem Programm Win32 Disk Imager (Windows) anlegen, siehe Kapitel 6.5 SD-Karte für Raspberry Pi vorbereiten. Da auf der SD-Karte alle Informationen gespeichert sind, die der Raspberry Pi zum Arbeiten benötigt, ist eine Wiederherstellung mit Hilfe einer Ersatzkarte leicht möglich. Dieses Vorgehen ist eine einfache und schnelle Möglichkeit das System, nach einem Vorfall, wieder in Betrieb zu nehmen.

Wir empfehlen fertige Projekte in Stichpunkten zu dokumentieren und alle Änderungen sowie die installierte Software festzuhalten. Somit hat man die Möglichkeit, Veränderungen nachzuvollziehen und kann das System jederzeit neu aufbauen.

## 6.13.5. Updates installieren

Ein wichtiger Punkt der Cyber Security ist die Installation von Updates, die Entwicklung des Linux-Kernels und der Raspbian Distribution wird aktiv betrieben. Dies geschieht, um das System permanent zu verbessern und um gefundene Sicherheitslücken zu schließen.

Es stellt sich die Frage, wie und unter welchen Umständen Updates installiert werden können, um ein Höchstmaß an Sicherheit zu erreichen. Anleitungen im Internet zeigen wie sich beispielsweise automatisch Updates installieren lassen, ähnlich wie bei Windows.

### Anmerkung:

Die Installation von Updates ist sehr wichtig. Im Hinblick auf Systemstabilität und Funktion jedoch nicht immer unkritisch, vor allem wenn es sich um Steuerungsanwendungen handelt.

Das Aktualisieren von Programmen kann in der Regel gefahrlos durchgeführt werden. Eine Aktualisierung von Betriebssystemteilen oder des gesamten Betriebssystems kann zu Inkompatibilitäten mit vorhandenen oder selbst geschriebenen Programmen führen. Sollten nach einem Update notwendige Programme nicht wie erwartet funktionieren oder das Update schlägt fehl, muss das Gerät einem Service unterzogen und instandgesetzt werden. Siehe dazu Punkt 6.13.4 Backups des Systems erstellen.

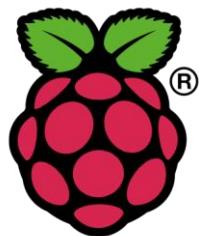
Wir empfehlen Updates nicht auf dem Produktivsystem zu installieren, sondern zunächst Versuche mit einem Testgerät durchzuführen. So lässt sich feststellen, ob nach einem Update alles wie erwartet funktioniert

## 7. CODESYS

Dieses Kapitel beschreibt die Installation aller benötigten Software Komponenten um PiXtend V2 ([www.pixtend.de](http://www.pixtend.de)) mit CODESYS ([www.codesys.de](http://www.codesys.de)) zu programmieren.



Die CODESYS Entwicklungsumgebung ist für die professionelle hardwareunabhängige Programmierung von Steuerungen nach IEC 61131-3 konzipiert und wird von der Firma 3S-Smart Software Solutions entwickelt.



Um einen Raspberry Pi in CODESYS zu programmieren wird die Raspberry Pi Runtime Erweiterung für CODESYS benötigt, die ebenfalls von 3S entwickelt wird.



Spezielle Geräte-Treiber für PiXtend V2 werden benötigt um einen direkten Zugriff auf die I/O-Hardware und Schnittstellen des PiXtend V2 Boards mittels CODESYS zu erhalten. Die PiXtend V2 Treiber stellt Kontron Electronics GmbH kostenfrei zur Verfügung.

Dieses Kapitel richtet sich an Personen, die ein Projekt für PiXtend V2 unter der Verwendung von CODESYS erstellen wollen.

### NOTICE

Die Verwendung von CODESYS ist für PiXtend V2 -S- und PiXtend V2 -L- gleich, lediglich für die Ansteuerung der I/O-Hardware (auch E/A-Mapping genannt) muss der passende SPI-Treiber gewählt werden, da hier die Unterscheidung zwischen den Geräten stattfindet.

## 7.1. Hinweise

### 7.1.1. Urheberrecht von Texten und Bildern:

Texte und Bilder, welche mit dem Kürzel (3S) versehen sind, stammen von der Firma 3S-Smart Systems GmbH in Kempten – [www.codesys.com](http://www.codesys.com)

Texte und Bilder, welche mit dem Kürzel (RPI) versehen sind, stammen von der Raspberry Pi Foundation – [www.raspberrypi.org](http://www.raspberrypi.org)

Texte und Bilder, welche nicht markiert oder mit dem Kürzel (QS) versehen sind, stammen von der Firma Kontron Electronics GmbH – [www.pixtend.de](http://www.pixtend.de)

### 7.1.2. Kompatibilität

CODESYS Projekte und Programme, die für andere CODESYS Steuerungen oder auf anderen CODESYS Steuerungen geschrieben wurden, lassen sich mit wenig Aufwand auf andere CODESYS kompatible Steuerungen übertragen. Die CODESYS Runtime Erweiterung für den Raspberry Pi Computer macht aus diesem eine vollwertige CODESYS SPS, alle kompatiblen CODESYS Programme können ausgeführt werden.

Zu beachten ist lediglich die Hardware-Kompatibilität, es muss überprüft werden ob es Übereinstimmungen oder Abweichungen gibt. Unter Umständen müssen Anpassungen an der Hardware-Konfiguration im Programm vorgenommen werden, damit im CODESYS Projekt die richtigen Signale an die richtige Stelle gelangen.

Sie sind unsicher sind ob Ihr bestehendes CODESYS SPS Projekt mit PiXtend V2 zusammenarbeitet. Nehmen Sie mit uns Kontakt auf, wir beraten Sie gerne bei notwendigen Änderungen und sind bei der Umstellung des Programms behilflich.

## 7.2. Voraussetzungen

### 7.2.1. Systemanforderungen für CODESYS

- Microsoft Windows 7/8/10(32/64Bit)
- Geeignete PC-Hardware für die entsprechende Microsoft Windows-Plattform

### 7.2.2. Benötigte Hardware

- PiXtend V2 Board ([www.pixtend.de](http://www.pixtend.de))
- Raspberry Pi Modell B+/2B/3B/3B+/4B
- Standard RJ-45 Netzwerkkabel  
Die Netzwerkeinstellungen des Raspberry Pi sind ab Werk auf DHCP eingestellt. Die IP-Adresse wird in den meisten Fällen vom installierten Internet-Router vergeben.
- Für die erste Inbetriebnahme eines neuen Raspberry Pi:
  - SD-Kartenleser, intern oder extern
  - SD-Speicherkarte, empfohlen min. 4GB (besser 8GB)  
microSD für Raspberry Pi Modell B+/2B/3B/3B+/4B
  - optional: HDMI fähiger Monitor
  - optional: USB-Keyboard
  - optional: USB-Maus

## 7.3. Installation der benötigten Software Komponenten

### 7.3.1. CODESYS Entwicklungsumgebung

#### 7.3.1.1 Vorstellung von CODESYS

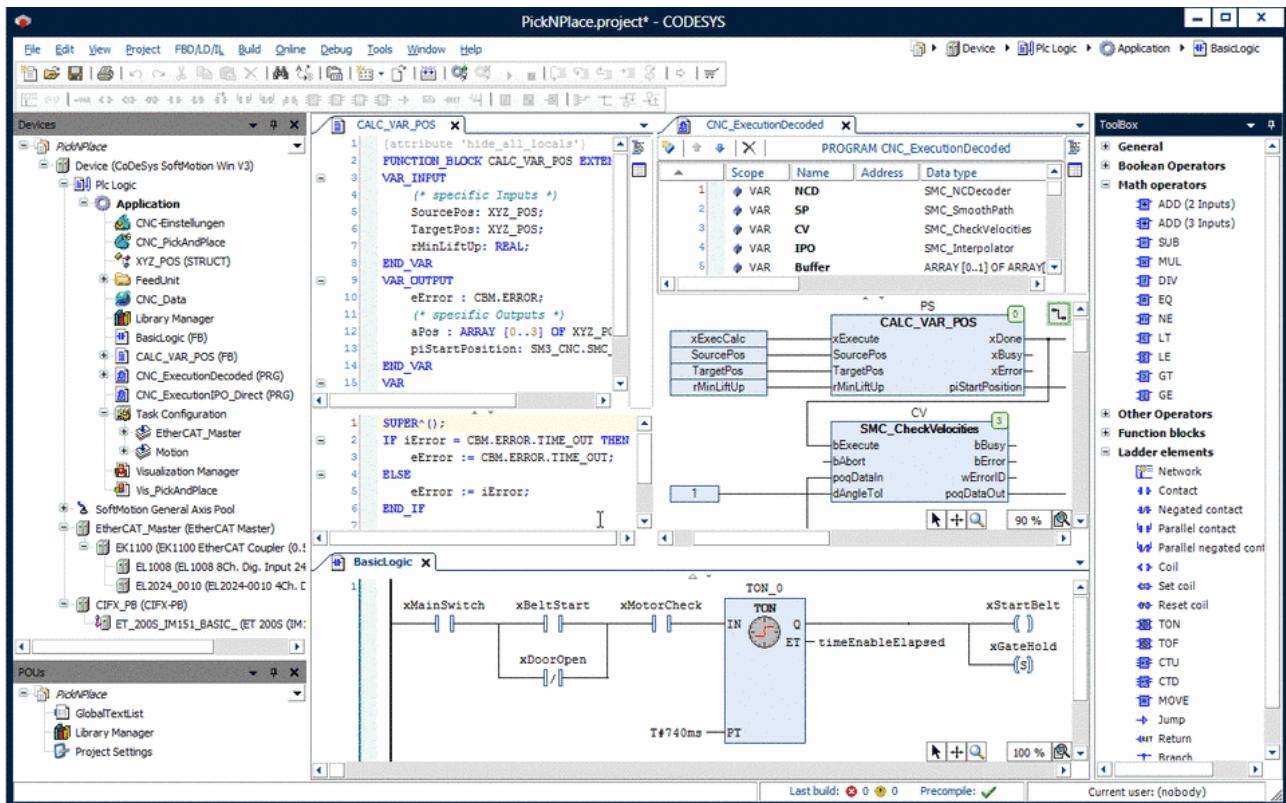


Abbildung 6: CODESYS Development System (Quelle: 3S)

CODESYS ist eine Software-Plattform für Aufgabenstellungen in der industriellen Automatisierungstechnik. Basis ist das IEC 61131-3 Programmierwerkzeug CODESYS Development System V3. Das Tool bietet dem Anwender integrierte Lösungen für seine Arbeit – mit dem Ziel, ihn praxisgerecht bei der Realisierung seiner Aufgabe zu unterstützen. (Quelle: 3S)

Das hardwareunabhängige Programmiersystem CODESYS von der Firma 3S-Smart Software GmbH eignet sich bestens für den Einsatz mit Raspberry Pi und PiXtend V2. Es ermöglicht neben der Programmierung in allen Sprachen für speicherprogrammierbare Steuerungen (SPS) nach der Norm IEC 61131-3 auch die Erstellung von Web-Visualisierungen. Mit der sogenannten „Webvisu“ lassen sich Inhalte und Steuerelemente Ihres Programms leicht auf eine Webseite übertragen (CODESYS-Webserver läuft auf dem Raspberry Pi). Für das Erstellen und Arbeiten mit der Webvisu benötigen Sie keinerlei Kenntnisse über Web-Programmierung (HTML, PHP, CGI o.ä.).

Mit der modernen grafischen Oberfläche von CODESYS sind Sie bestens für die Programmierung der Steuerung und Webvisu ausgerüstet. Der Zugriff auf die Webvisu ist mit jedem aktuellen Smartphone, Tablet oder PC/MAC möglich. Sie benötigen lediglich einen aktuellen Web-Browser. Wir empfehlen die aktuellen Versionen der Browser Google Chrome, Microsoft Internet Explorer oder Mozilla Firefox.

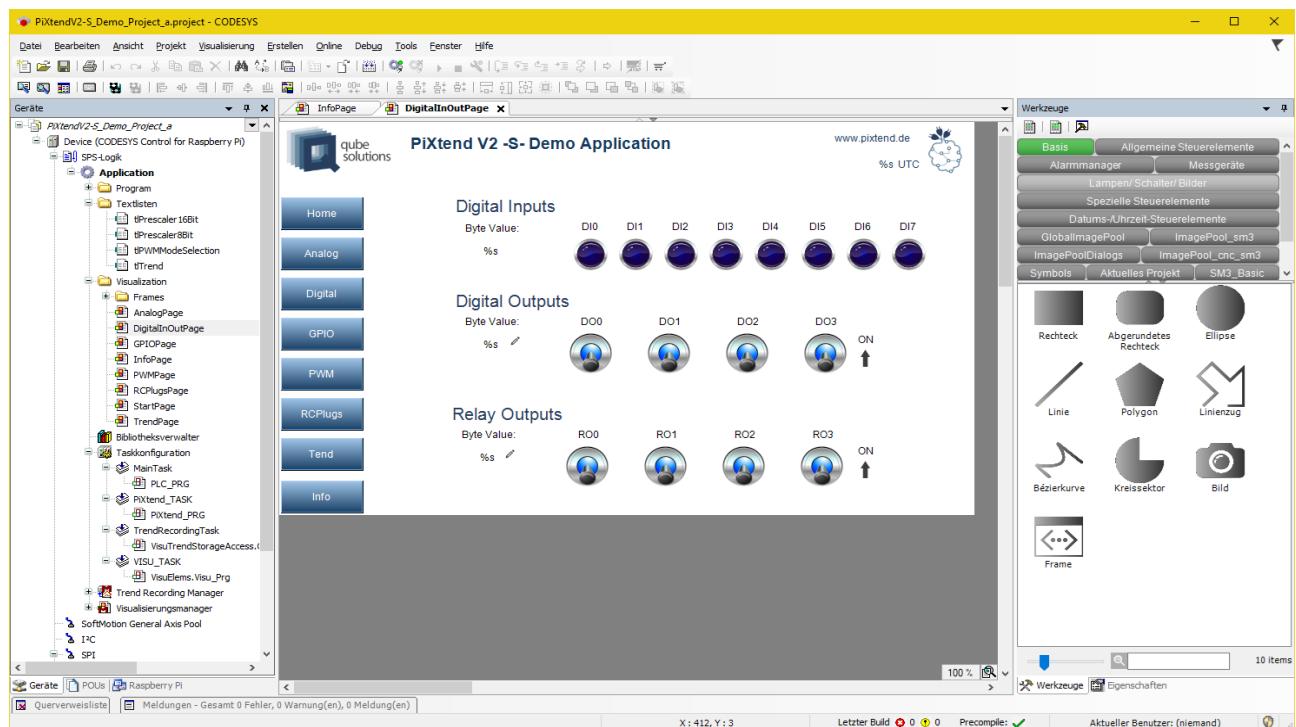


Abbildung 7: CODESYS - Visualisierung für das PiXtend V2 Demo Projekt

### 7.3.1.2 CODESYS Download

Die CODESYS Entwicklungsumgebung kann kostenlos von 3S heruntergeladen werden. Dazu ist lediglich eine unkomplizierte Registrierung notwendig.

- Öffnen Sie dazu die Seite <https://store.codesys.com/> in einem Browser Ihrer Wahl.
- Klicken Sie auf den Button Anmelden und registrieren Sie sich im „CODESYS Store“ (Benutzerkonto anlegen).
- Füllen Sie das Formular vollständig aus und beachten Sie, dass eine korrekte E-Mail-Adresse benötigt wird.
- Klicken Sie auf „Jetzt Registrieren“ und folgen Sie den Anweisungen auf der Store Seite oder den Hinweisen, die Sie per E-Mail erhalten, eventuell müssen Sie Ihre E-Mail-Adresse bestätigen und warten bis Ihr Konto freigeschaltet wurde.
- Mit Ihrem selbst vergebenen Passwort und Ihrem Benutzernamen können Sie sich im CODESYS Store einloggen und die aktuellste CODESYS Version herunterladen (Momentan CODESYS V3.5 SP15 Patch 1, Stand November 2019).

### 7.3.1.3 CODESYS Installation

Installieren Sie die CODESYS Entwicklungsumgebung mit den vorgegebenen Standard-Einstellungen und als Administrator-Benutzer oder führen Sie die Installation mit Administratorrechten aus. CODESYS selbst kann ohne erweiterte Rechte gestartet werden, ein „normaler“ Windows Benutzer ist ausreichend.

Nun steht Ihnen eine vollwertige CODESYS Entwicklungsumgebung zur Verfügung mit der Sie die PiXtend V2 SPS gemäß dem Industrie-Standard IEC 61131-3 programmieren können.

### 7.3.2. CODESYS Control for Raspberry Pi installieren

#### 7.3.2.1 CODESYS Control for Raspberry Pi - Download

Um einen Raspberry Pi in CODESYS zu verwenden, benötigen Sie die kostenlose CODESYS-Control Laufzeitumgebung für den Raspberry Pi aus dem CODESYS Store.

Der CODESYS Store beinhaltet kostenlose und kostenpflichtige Produkte die in CODESYS als zusätzliche Features eingebunden werden können.

Loggen Sie sich im CODESYS Store ein um Dateien herunterzuladen. Ein entsprechendes Konto haben Sie bereits gemäß den Vorgaben in einem vorangegangenen Kapitel angelegt.

- Direkt nach dem Login können Sie alle mit „Free“ gekennzeichneten Produkte kostenlos herunterladen
- Klicken Sie auf das Produkt „CODESYS Control for Raspberry Pi SL“ oder folgen Sie dem direkten Link:  
<http://store.codesys.com/codesys-control-for-raspberry-pi-sl.html>
- Klicken Sie auf den Download-Button und warten Sie bis der Download abgeschlossen wurde
- Eine „Erste Schritte“ Anleitung finden Sie in der CODESYS Online Hilfe:  
[https://help.codesys.com/webapp/\\_rbp\\_f\\_help;product=CODESYS\\_Control\\_for\\_Raspberry\\_Pi\\_SL;version=3.5.15.10](https://help.codesys.com/webapp/_rbp_f_help;product=CODESYS_Control_for_Raspberry_Pi_SL;version=3.5.15.10)

#### Hinweise zur Demo Version:

- Die kostenlose Version ist auf 2 Stunden Laufzeit beschränkt. Danach beendet sich die Laufzeitumgebung automatisch bis zum nächsten Neustart. Nach jedem Neustart kann der Raspberry Pi für 2 Stunden ohne Einschränkungen benutzt werden. Dies ist für erste Tests völlig ausreichend.
- Für eine zeitlich uneingeschränkte Version steht eine Lizenz zum aktuellen Preis von 50€ zzgl. MwSt. zur Verfügung. Sie können die Lizenzierung zu einem späteren Zeitpunkt im CODESYS Store durchführen.

### 7.3.2.2 CODESYS Control for Raspberry Pi – Installation

Ein Doppelklick auf die Datei „CODESYS Control for Raspberry PI 3.5.X.X.package“ startet den CODESYS Package Manager der automatisch die Installation der Raspberry Pi Laufzeit-Komponenten für CODESYS durchführt.

Alternativ können Sie CODESYS manuell starten und den Package Manager über das Menü Tools → Package Manager öffnen.

Klicken Sie im Package Manager auf „Installieren“ und wählen Sie das soeben heruntergeladene Package aus.

Führen Sie die Installation mit den Standard-Optionen durch.

**NOTICE**

Für die Installation benötigen Sie die Administrator-Rechte auf dem Computer, sonst kann die Installation fehlschlagen. Im Zweifelsfall starten Sie CODESYS mit einem Rechtsklick „Als Administrator ausführen“ und starten dann den „Package Manager“ zur Installation des Raspberry Pi Package.

Im Paket sind enthalten:

- CODESYS Erweiterung für das Raspberry Pi (Treiber und Bibliotheken)
- Beispiel Programme (Standardmäßig im Benutzer-Verzeichnis unter „CODESYS Control for Raspberry PI“)
- Debian Package codesyscontrol\_arm\_raspberry\_V3.5.X.X.deb  
für die Installation der Runtime auf dem Linux-System des Raspberry Pi

**NOTICE**

Eine Anleitung mit ausführlicheren technischen Informationen und Installationsanleitung finden Sie in der CODESYS Online Hilfe unter  
[Addons:https://help.codesys.com/webapp/\\_rbp\\_f\\_help;product=CODESYS\\_Control\\_for\\_Raspberry\\_Pi\\_SL;version=3.5.15.10](https://help.codesys.com/webapp/_rbp_f_help;product=CODESYS_Control_for_Raspberry_Pi_SL;version=3.5.15.10)

Wählen Sie am oberen Rand Ihre gewünschte Sprache aus.

Das Dokument erläutert unter anderem die Lizenzierungsoptionen der CODESYS Raspberry Pi Runtime Erweiterung.

### 7.3.3. Bootfähiges SD-Karten-Image für den Raspberry Pi erstellen

Laden Sie sich das neueste PiXtend V2 „CODESYS“ SD-Karten Image aus dem Download Bereich <https://www.pixtend.de/downloads/> herunter.

Das Image basiert auf dem Raspbian Linux-Betriebssystem und ist für die Verwendung mit CODESYS und PiXtend V2 bereits vorkonfiguriert.

Alternativ können Sie die neueste Version von Raspbian herunterladen und installieren, wie auf <https://www.raspberrypi.org/downloads/raspbian/> beschrieben.

Folgen Sie der Installationsanleitung, welche Sie in der CODESYS Online Hilfe finden unter Addon „CODESYS Control for Raspberry Pi SL“:

- CODESYS Online Hilfe zu CODESYS Control for Raspberry Pi SL

Einstiegern empfehlen wir ausdrücklich unser PiXtend V2 Image „CODESYS Control“ zu verwenden, um einen möglichst einfachen und problemlosen Start zu erzielen.

### 7.3.4. PiXtend V2 CODESYS Gerätetreiber installieren

Um auf die PiXtend V2 Hardware in CODESYS zugreifen zu können benötigen Sie die PiXtend V2 CODESYS Gerätetreiber. Diesen können Sie entweder im Download-Bereich unserer Homepage (<https://www.pixtend.de/downloads/>) oder im CODESYS Store herunterladen.



#### 7.3.4.1 PiXtend V2 CODESYS Gerätetreiber - Download

Das Package „PiXtend V2 Professional for CODESYS“ enthält die CODESYS Geräte-Treiber, Beispielprogramme und eine ausführliche Bedienungsanleitung für die Verwendung von CODESYS mit dem PiXtend V2.

#### 7.3.4.2 PiXtend V2 CODESYS Gerätetreiber – Installation

Ein Doppelklick auf die Datei „PiXtend\_V2\_Professional\_for\_CODESYS\_V2\_X\_X.package“ startet den CODESYS Paketmanager der automatisch die Installation der PiXtend V2 Gerätetreiber, Beispielprojekte und PiXtend V2 Dokumentation für CODESYS durchführt. Standardmäßig werden die Inhalte im Benutzerverzeichnis unter „PiXtend V2 Professional for CODESYS“ abgelegt.

#### NOTICE

Sie benötigen für die Installation die Administrator-Rechte auf dem Computer, sonst kann die Installation fehlschlagen. Im Zweifelsfall starten Sie CODESYS mit einem Rechtsklick → „Als Administrator ausführen“ und starten dann den „Package Manager“ zur Installation des PiXtend V2 Package über das Tools Menü.

## 7.4. Weitere Schritte

Ihre CODESYS Entwicklungsumgebung ist nun für die Verwendung mit PiXtend V2 vorbereitet. Um ein leeres CODESYS Projekt mit PiXtend V2 Unterstützung zu erstellen, lesen Sie im „Kapitel 7.5 CODESYS – Projekte erstellen weiter.“

## 7.5. CODESYS – Projekt erstellen

Dieses Kapitel beschreibt alle notwendigen Schritte um ein neues PiXtend V2 Projekt ([www.pixtend.de](http://www.pixtend.de)) in CODESYS ([www.codesys.com](http://www.codesys.com)) anzulegen. Sie lernen PiXtend V2 als CODESYS Gerät zu verwenden und eine einfache Visualisierung für Ihr Projekt zu erstellen.

### 7.5.1. Schritt für Schritt zum ersten PiXtend V2 CODESYS Programm

#### 7.5.1.1 CODESYS Standard Projekt für PiXtend V2 erzeugen

Starten Sie CODESYS.

Erstellen Sie ein neues Projekt indem Sie im Hauptmenü auf Datei → Neues Projekt klicken (Shortcut Strg-N / Ctrl-N)

Wählen Sie „Standard Projekt“ aus der Kategorie „Projekte“, geben Sie dem Projekt einen Namen (hier „PiXtendTest“) und bestätigen Sie mit „OK“

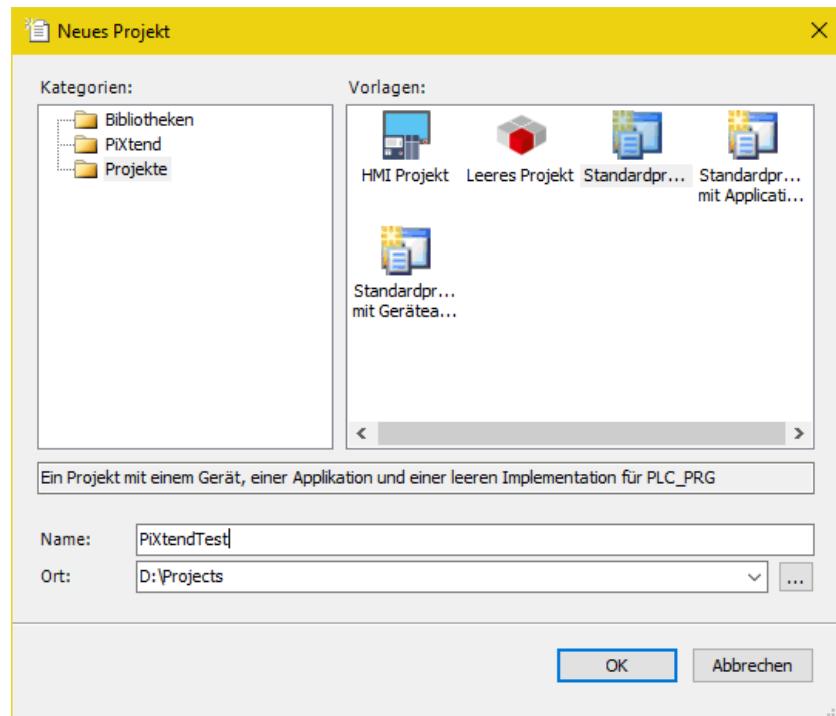


Abbildung 8: CODESYS - Neues Projekt

Als Gerät wählen Sie „CODESYS Control for Raspberry Pi“ und als Programmiersprache für das Hauptprogramm PLC\_PRG wählen Sie „Continuous Function Chart (CFC)“.

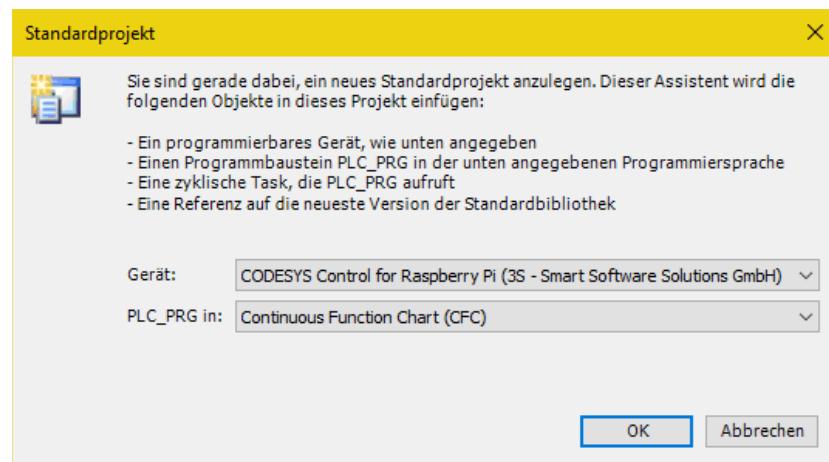


Abbildung 9: CODESYS - Neues Projekt - Auswahl Gerät

Nachdem CODESYS das Standard Projekt angelegt hat, erhalten Sie ein Projekt mit folgender Struktur:

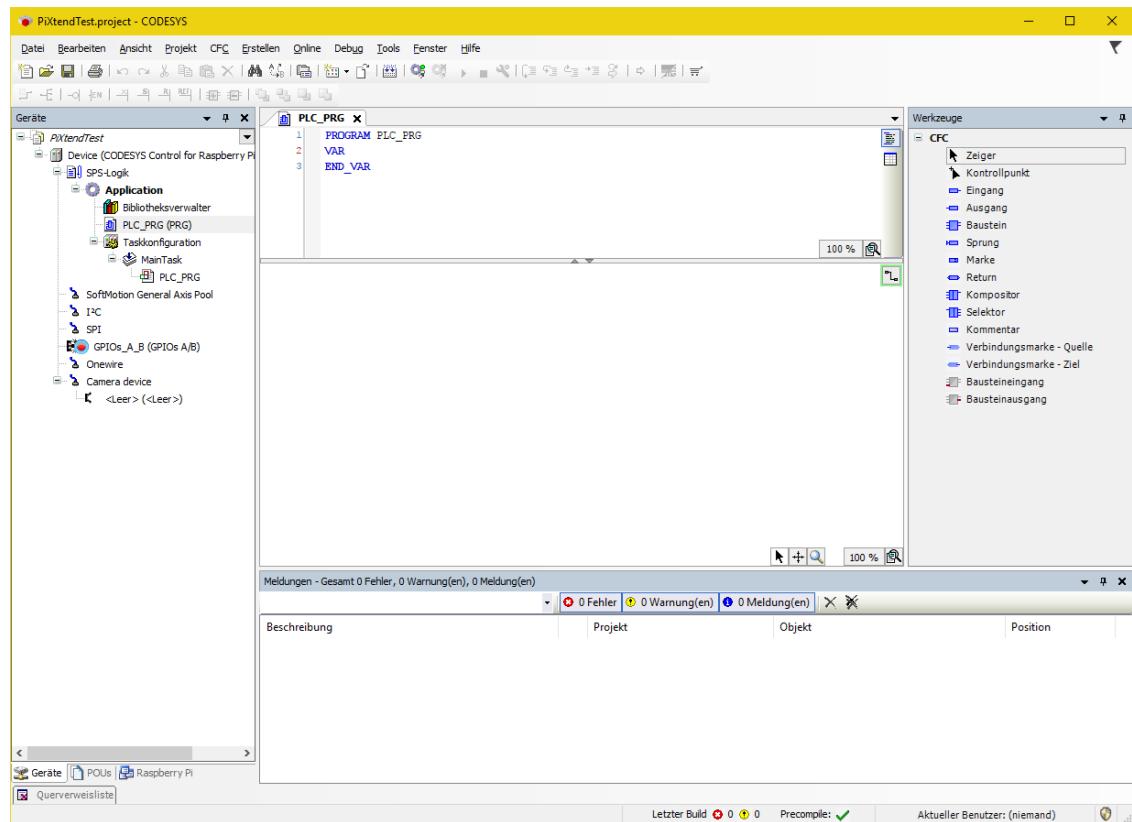


Abbildung 10: CODESYS - PiXtendTest Projekt

### 7.5.1.2 SPI Gerät anhängen

Im Projektbaum machen Sie einen Rechtsklick auf den Eintrag „SPI“ und wählen Sie „Gerät anhängen“

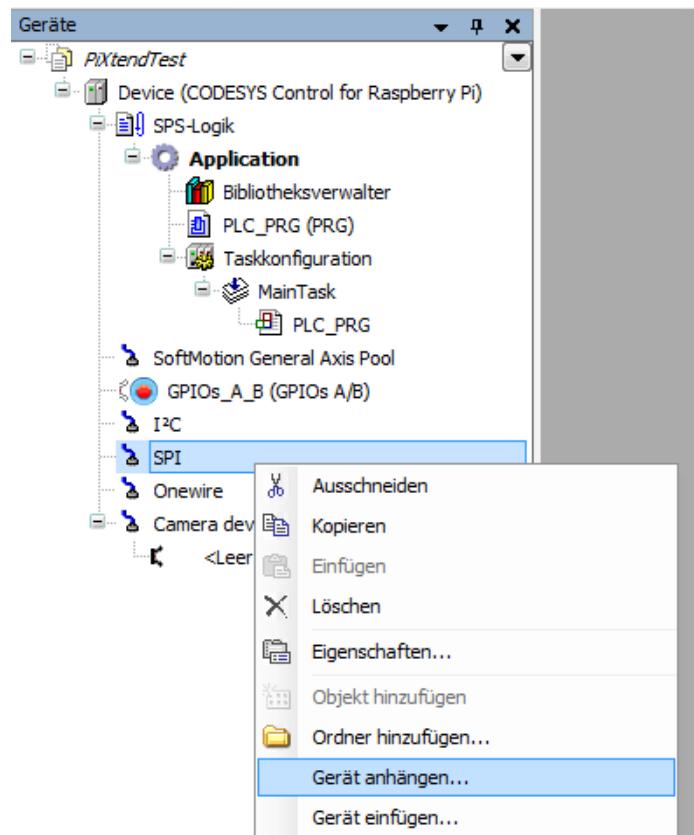


Abbildung 11: CODESYS - Gerät anhängen

Wählen Sie „SPI master“ als Gerät aus und klicken Sie auf den Button „Gerät anhängen“. Lassen Sie das Fenster geöffnet.

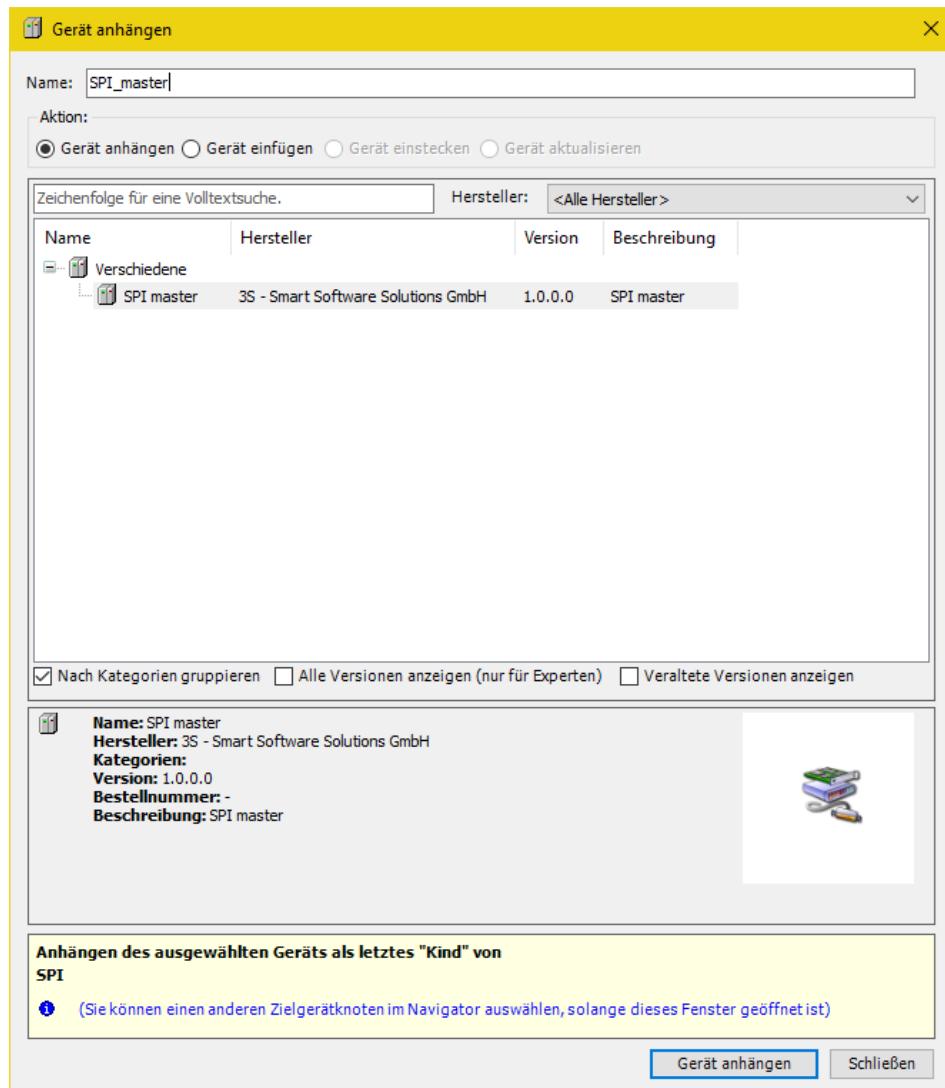


Abbildung 12: CODESYS - Gerät anhängen - SPI master

Im Projektbaum unter SPI existiert nun ein neuer Eintrag „SPI\_master (SPI Master)“.

### 7.5.1.3 PiXtend V2 -S- Gerät anhängen

Während das Fenster geöffnet ist klicken Sie mit der linken Maustaste auf den soeben erzeugten SPI Master. Die Inhalte des „Gerät anhängen“ Fensters werden aktualisiert.

Wählen Sie im Geräte Hersteller Drop-Down Menü den Eintrag „Kontron Electronics GmbH“ aus, wählen Sie anschließend das Gerät aus „PiXtend V2 -S-“\* (nicht PiXtend V2 -S- DAC) und klicken Sie rechts unten auf den Button „Gerät anhängen“ und schließen Sie das Fenster.

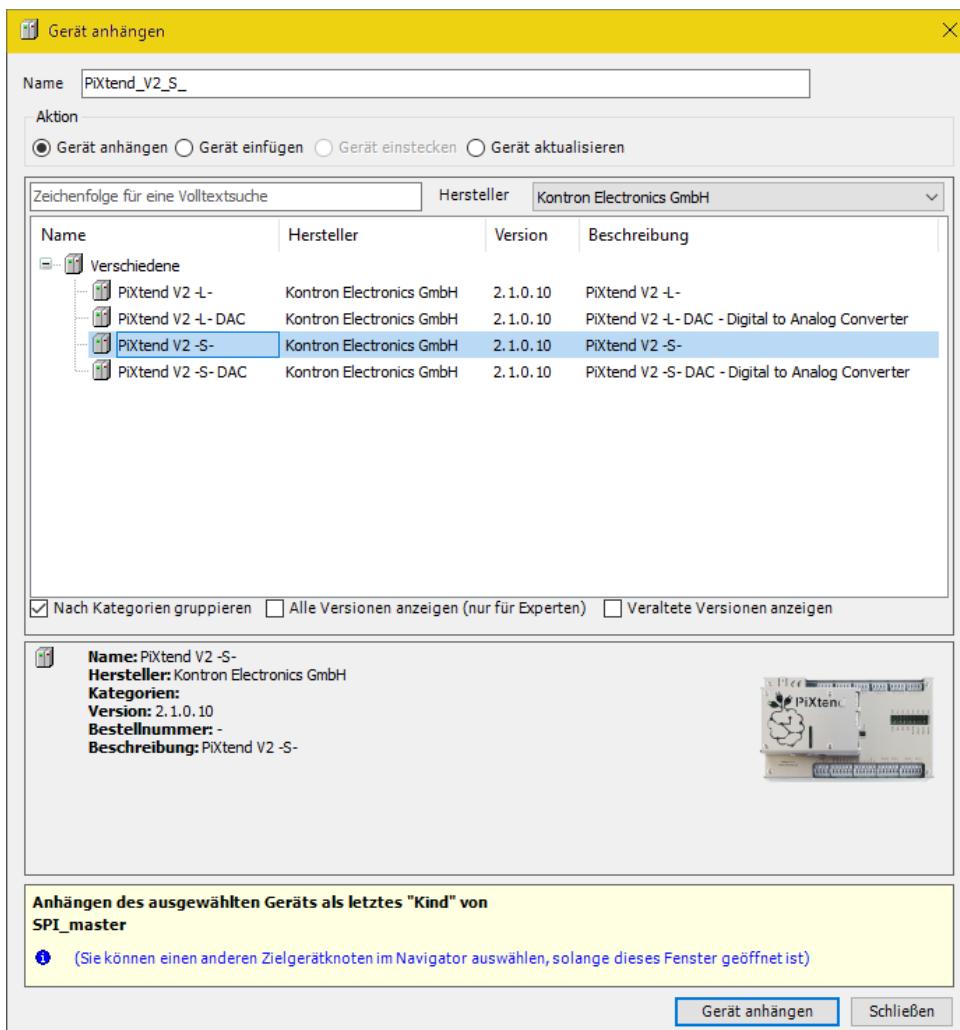


Abbildung 13: CODESYS - Gerät anhängen - PiXtend V2 -S-

Nun erscheint PiXtend\_V2\_S\_ als Gerät unter „SPI\_master (SPI Master)“.

#### NOTICE

Alternativ zum PiXtend V2 -S- können Sie an dieser Stelle gerne ein PiXtend V2 -L- Gerät verwenden. Das hier beschriebene Beispiel passt für beide PiXtend V2 Geräte gleichermaßen. Bitte beachten Sie, dass die Gerätenamen unterschiedlich sein können.

Ein Doppelklick auf PiXtend\_V2\_S\_ im Projektbaum öffnet die Konfigurationsseite für das Gerät. Im Reiter „SPI devices E/A-Abbild“ sehen Sie, welche Ein und Ausgänge von PiXtend V2 -S- für die Verwendung in CODESYS bereitgestellt werden. Im laufenden Betrieb („Online zur Steuerung“), lässt sich in diesem Fenster das gesamte Prozessabbild von PiXtend V2 -S- beobachten. Eingangswerte können überwacht und Ausgangswerte direkt gesetzt werden. Da die Werte in unserem Hauptprogramm sowie in der Visualisierung verwendet werden sollen, erzeugen wir eine „Globale Variablen Liste“ um den Ein- und Ausgängen Variablen zuzuweisen.

#### 7.5.1.4 Globale Variablen Liste erzeugen

Rechts-klicken Sie auf den Eintrag „Application“ im Projektbaum und fügen mit „Objekt hinzufügen“ -> „Globale Variablen Liste“ eine neue Globale Variablen Liste mit der Bezeichnung „GVL“ hinzu:

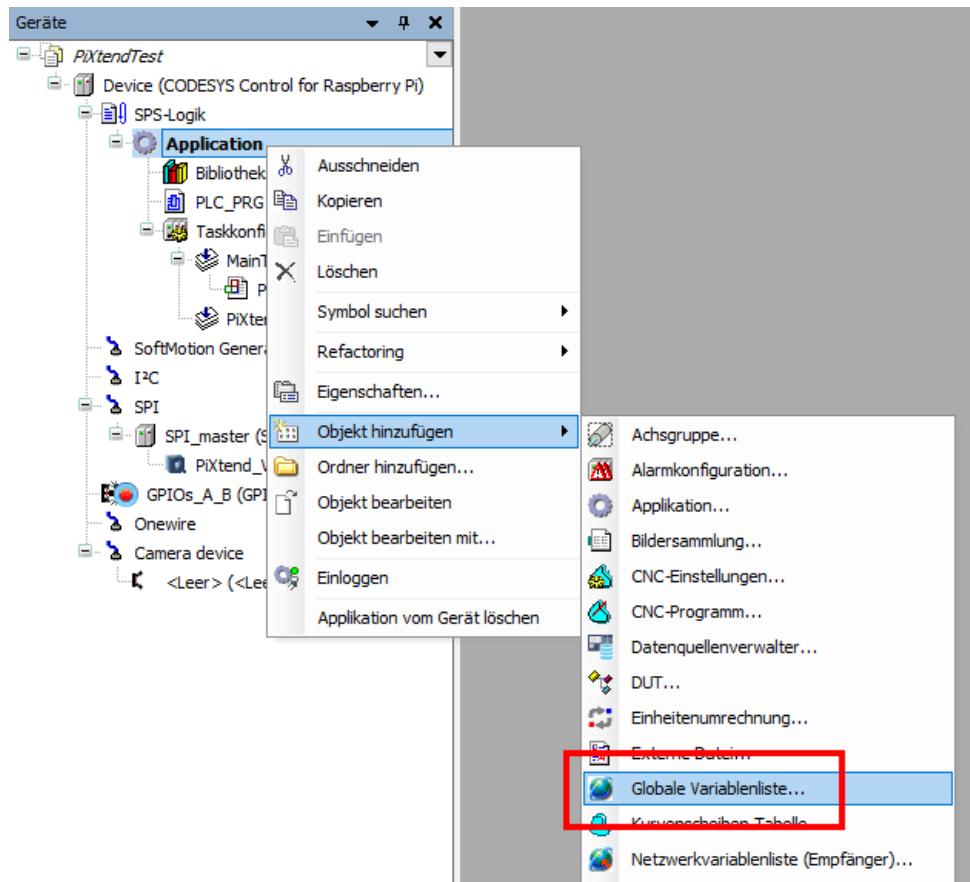
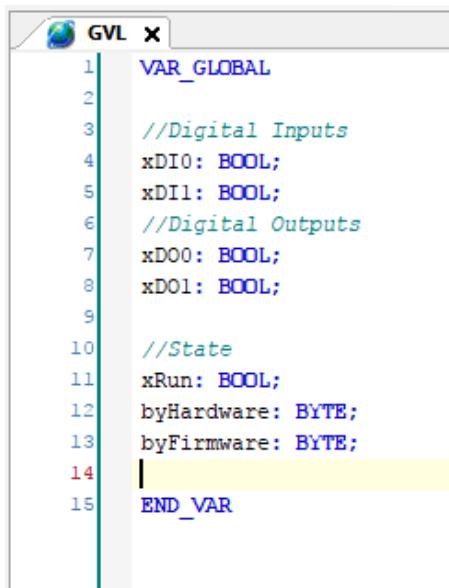


Abbildung 14: CODESYS – Globale Variablenliste einfügen

Öffnen Sie die GVL und fügen Sie folgende Einträge hinzu:

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



The screenshot shows the GVL (Global Variable Declaration) editor window. The code is displayed in a text area with line numbers on the left. The code defines global variables for digital inputs (xDI0, xDI1), digital outputs (xDO0, xDO1), status (xRun), and hardware/firmware versions (byHardware, byFirmware). The word 'VAR\_GLOBAL' is highlighted in blue at the top of the code.

```

1 VAR_GLOBAL
2
3 //Digital Inputs
4 xDI0: BOOL;
5 xDI1: BOOL;
6 //Digital Outputs
7 xDO0: BOOL;
8 xDO1: BOOL;
9
10 //State
11 xRun: BOOL;
12 byHardware: BYTE;
13 byFirmware: BYTE;
14
15 END_VAR

```

Abbildung 15: CODESYS - GVL - Deklarationen

Um dieses Kapitel einfach zu halten beschränken wir uns vorerst auf die Verwendung von 2 digitalen Eingängen und 2 digitalen Ausgängen.

xDI0 steht hierbei für den ersten digitalen Eingang, also Digital Input 0, und xDO0 für Digital Output 0. Beachten Sie den Unterschied zwischen 0 (Buchstabe) und 0 (Zahl).

Präfixe für Variablen Namen haben sich besonders in größeren Projekten bewährt.

Wir empfehlen die Verwendung des Präfixes x für BOOLsche Variablen (true oder false) um eine Verwechslung mit Variablen vom Typ Byte (Präfix by) vorzubeugen.

**Weitere Präfixe:**

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

Das Status Bit xRun gibt Auskunft darüber, in welchem Zustand sich der Mikrocontroller gerade befindet.

Die Bytes byHardware und byFirmware geben Auskunft über die Hardware Revision von PiXtend V2 -S- und die Firmwareversion des Mikrocontrollers.

Die Namensgebung der Variablen in der GVL bleibt Ihnen selbst überlassen, wir verwenden dieselben Konventionen wie im PiXtend V2 Demo Projekt, um Ihnen den Einstieg zu erleichtern.

### 7.5.1.5 Mapping der Variablen

Nachdem Sie die benötigten Variablen in der GVL angelegt haben, müssen Sie entsprechen „gemapped“ (zugewiesen) werden, sprich den jeweiligen Hardware Ein- und Ausgängen von PiXtend V2 -S- zugewiesen werden.

Öffnen Sie dazu das bereits bekannte „SPI devices E/A-Abbild“ indem Sie auf das PiXtend V2 -S- Gerät doppelklicken. Wählen Sie den Ordner „Digital Inputs“ und den Kanal „DigitalInputs“ aus (im Bild vom Typ Byte mit Adresse %IB22):

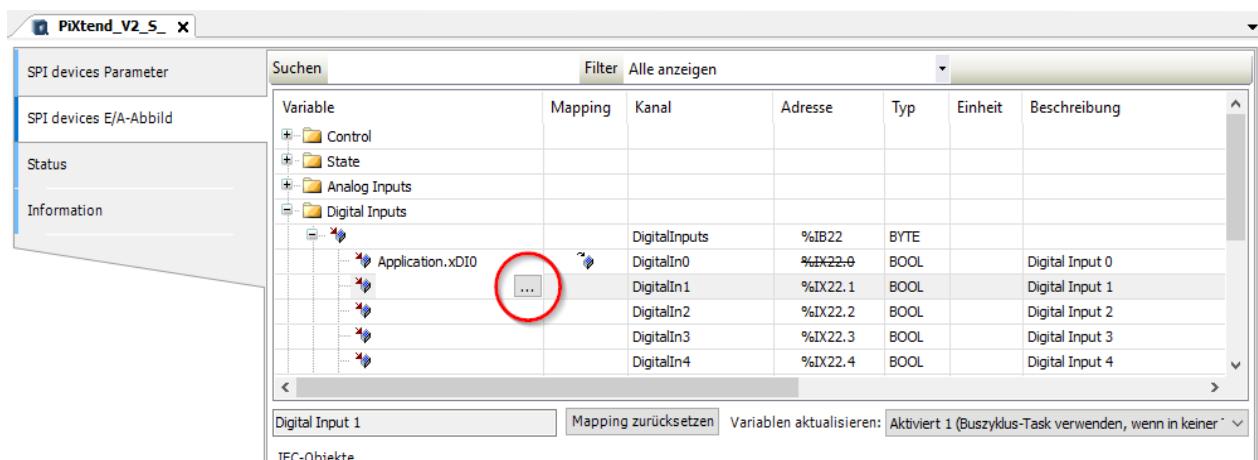


Abbildung 16: CODESYS - PiXtend V2 -S- E/A-Abbild

Nun können die existierenden Eingangsvariablen bit-weise gemapped werden. Wenn Sie auf die noch leere linke Spalte doppelklicken erscheinen drei Punkte (...). Ein Klick darauf öffnet die Eingabehilfe, mit der die gewünschte Variable ausgewählt wird.

Wählen Sie die Variable Application → GVL → xDIO für DigitalIn Bit 0

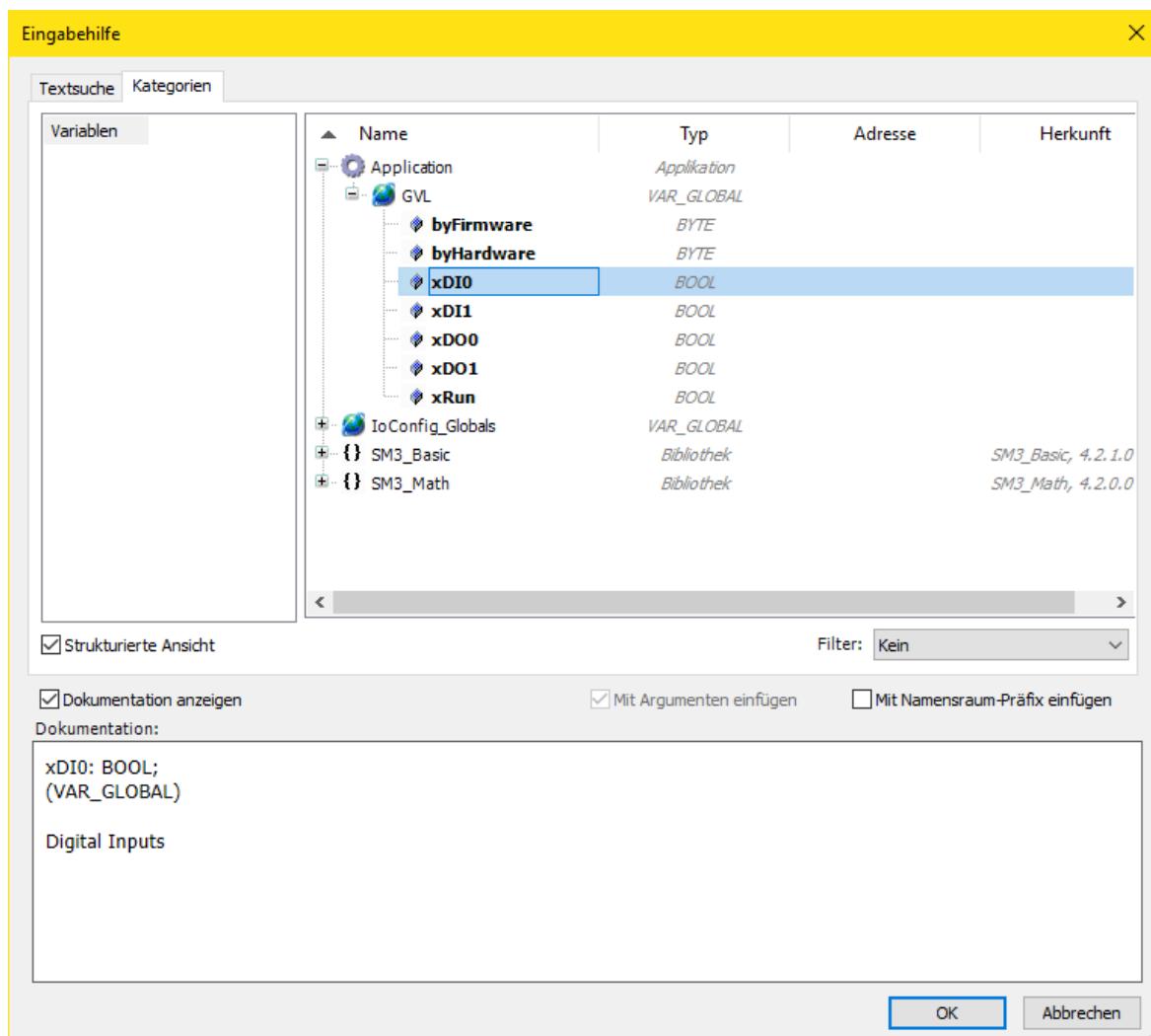


Abbildung 17: CODESYS - Eingabehilfe

Wiederholen Sie den Vorgang entsprechend für die restlichen Variablen:

Application → GVL → xDIO für Digital Input → DigitalIn → Bit 0

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

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

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

Bytes und Bits (BOOLs) können nach demselben Prinzip gemapped werden:

State					
Application.byFirmware		Firmware	%IB0	BYTE	
Application.byHardware		Hardware	%IB1	BYTE	
		ModelIn	%IB2	BYTE	
		RetainCRCError	%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	

Abbildung 18: CODESYS - PiXtend V2 -S- E/A-Abbild - State

Application → GVL → xRun für Status → Run

Application → GVL → byFirmware → Firmware

Application → GVL → byHardware → Hardware

Als Letztes muss das GPIO Bit 24 des Raspberry Pi als Ausgang konfiguriert werden. Öffnen Sie dazu die Raspberry Pi GPIO Konfiguration indem Sie auf „GPIOs\_A\_B“ im Projektbaum doppelklicken. Wählen Sie „Output“ für GPIO24:

GPIOs Parameter	Parameter	Typ	Wert	Standar...	Einheit	Beschreibung
GPIOs E/A-Abbild	GPIO4	Enumeration of BYTE	not used	not used		configuration of GPIO4
	GPIO17	Enumeration of BYTE	not used	not used		configuration of GPIO17
	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
	GPIO70	Enumeration of BYTE	not used	not used		configuration of GPIO70

Abbildung 19: CODESYS - GPIO Parameter - GPIO24

Wechseln Sie anschließend auf den Reiter „GPIO E/A-Abbild“ und klicken Sie auf „Outputs“. Als Variable für Bit 24 geben Sie „rpi\_gpio24“ ein.

Variable	Mapping	Kanal	Adresse
+ ...		digital inputs (GPIO0..GPIO31)	%ID22
- ...		digital outputs (GPIO0..GPIO31)	%QD14
Bit4		Bit4	%QX56.4
Bit17		Bit17	%QX58.1
Bit18		Bit18	%QX58.2
Bit22		Bit22	%QX58.6
Bit23		Bit23	%QX58.7
rpi_gpio24	Bit24	Bit24	%QX59.0
		Bit25	%QX59.1
		Bit27	%QX59.3

Abbildung 20: CODESYS - GPIO E/A-Abbild - rpi\_gpio24 Variable

### 7.5.1.6 Erstellung des Hauptprogramms

Nun erstellen Sie ein Programm, das die beiden Eingänge DIO und DI1 überwacht und entsprechend einer Programmlogik die beiden Ausgänge D00 und D01 bedient.

Sequenzielle Programme (zum Beispiel in C oder Python) werden in der Regel nur einmal, von oben bis unten ausgeführt. Bei speicherprogrammierbaren Steuerungen (SPS, bzw. PLC, „Programmable Logic Controller“ auf Englisch) ist es notwendig, dass bestimmte Programmteile zyklisch, in festen Intervallen aufgerufen werden. Hier wird überprüft, ob sich Eingangsvariablen geändert haben und ob die Ausgänge entsprechend der Programmlogik neu gesetzt oder rückgesetzt werden müssen.

In einem sogenannten Task wird festgelegt, in welcher Reihenfolge und wie häufig ein Programm aufgerufen wird.

Der Typische Ablauf innerhalb eines SPS Zyklus ist:

- Alle Hardware Eingangswerte abrufen und zwischenspeichern
- Programmlogik durchlaufen, um neue Ausgangswerte zu berechnen
- Alle Hardware Ausgangswerte zuweisen

Klicken Sie auf Taskkonfiguration → MainTask um die Zykluszeit auf t#100ms festzulegen. Diese Zykluszeit ist unabhängig von der verwendeten Hardware, da es hierfür einen eigenen Task gibt. Der automatisch erstellte PiXtend-Task hat eine Zykluszeit von 30 ms und sollte nicht verändert werden. Weitere Informationen finden Sie in Kapitel 6.2 SPI-Kommunikation, Datenübertragung und Zykluszeit.

Die Schreibweise t# symbolisiert, dass es sich bei dem Wert um eine Zeitangabe handelt, die bei der Programmierung von Steuerungen nach IEC 61131-3 gebräuchlich ist.

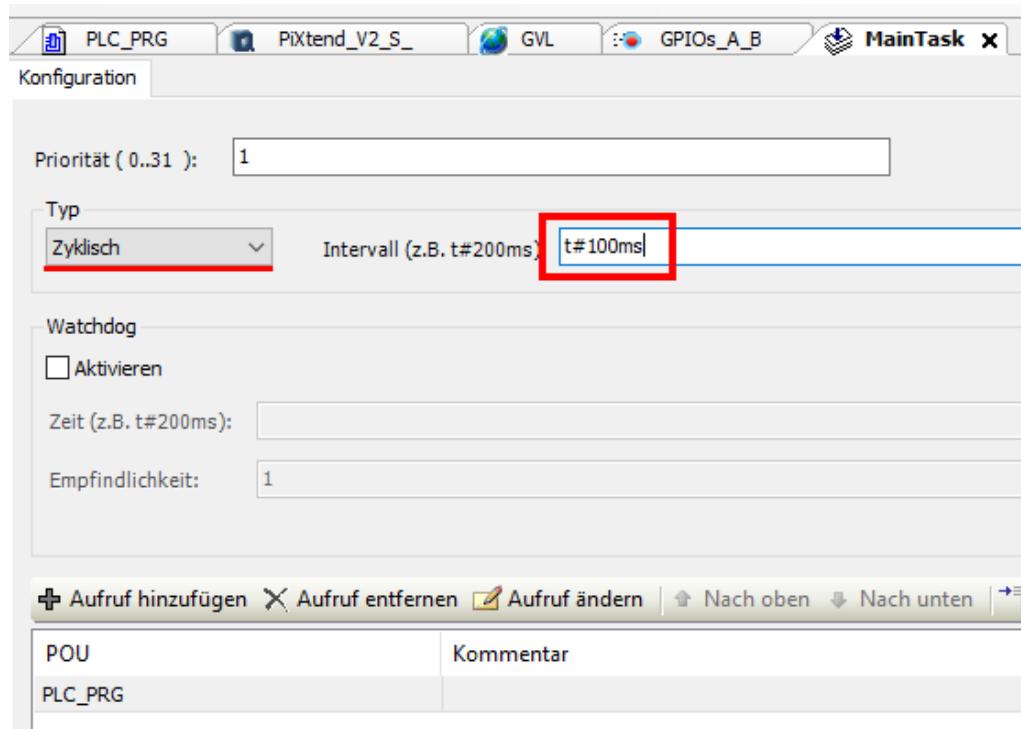


Abbildung 21: CODESYS - Taskkonfiguration - MainTask

Die vorgenommenen Einstellungen bedeuten: Im Task „MainTask“ wird alle 100 ms das Programm PLC\_PRG aufgerufen, es läuft sequenziell (von oben nach unten) ab und wird alle 100 ms wiederholt.

In unserem Beispiel ist das Hauptprogramm mit dem Namen „PLC\_PRG“ versehen. Es wird in der Programmiersprache „Continous Function Chart“ erstellt, da diese Einstellung bei der Erstellung des CODESYS-Projektes ausgewählt wurde.

Komplexe SPS-Programme werden in der Regel auf mehrere POUs (Program Organization Units, dt. Bausteine) verteilt, um die Wartbarkeit und Übersichtlichkeit des Codes zu verbessern.

Wir beschränken uns vorerst auf eine Programm Einheit, nämlich PLC\_PRG, welche von CODESYS bei der Erstellung des Standardprojektes automatisch erzeugt wurde.

Doppelklicken Sie auf PLC\_PRG, um den Editor zu öffnen und mit der Erstellung des Programm Codes zu beginnen.

In CFC wird grafisch programmiert. Ziehen Sie zunächst drei „Eingang“ Blöcke per Drag&Drop aus der Werkzeug Box (rechts) und platzieren Sie diese im Arbeitsbereich untereinander:

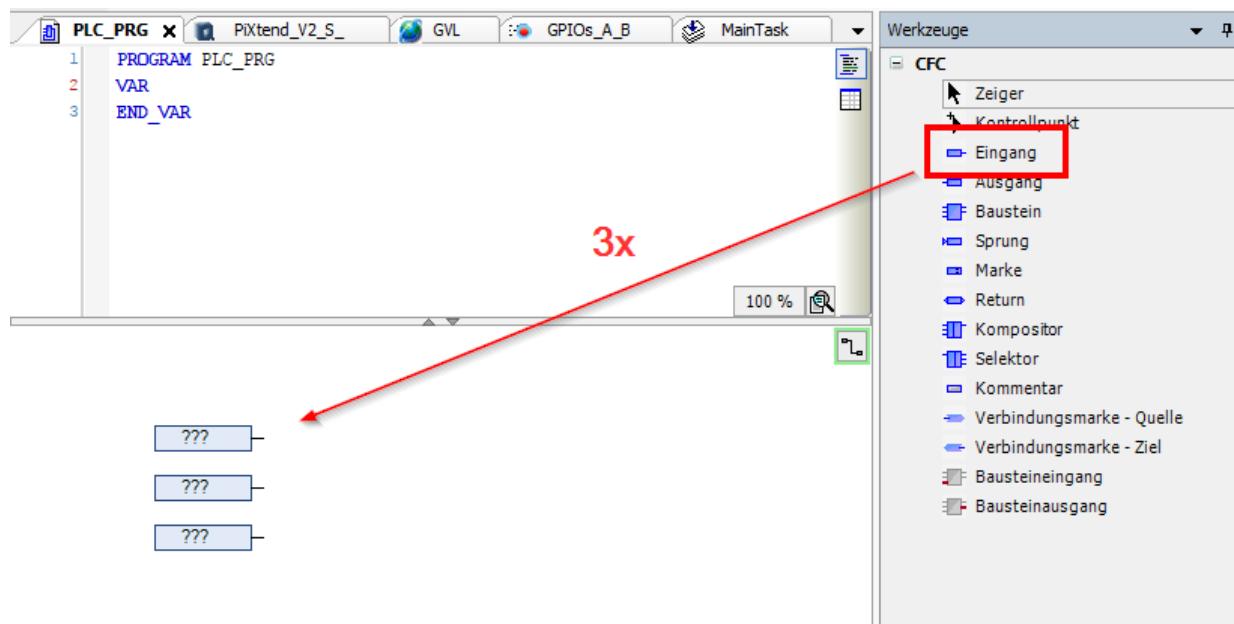


Abbildung 22: CODESYS - PLC\_PRG (CFC) - Eingang

Wiederholen Sie dies mit drei „Ausgang“ Blöcken und verbinden Sie per Drag&Drop jeweils einen Eingang mit einem Ausgangsblock.

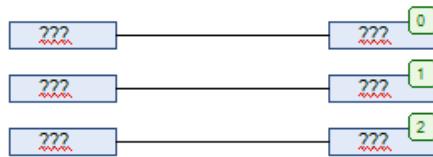


Abbildung 23: CODESYS - CFC - 3 Eingänge verbunden mit 3 Ausgängen

Klicken Sie in die Mitte eines Blockes um Variablen oder Konstanten wie in der Abbildung zuzuweisen. Ein Klick auf die drei Punkte ... öffnet die Ihnen schon bekannte, Eingabehilfe für Variablen. Sie können entweder den Variablennamen eintippen und die Auto vervollständigung von CODESYS nutzen oder aus der Liste auswählen. Bei direkter Eingabe achten Sie darauf die Vorsilbe „GVL.“, den Namen der Globalen Variablen Liste, zu verwenden

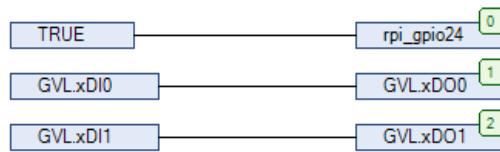


Abbildung 24: CODESYS - CFC - Parametriert

Sie haben soeben Ihr erstes Programm erstellt. Hier eine kurze Erklärung:

- Das Raspberry Pi GPIO Bit 24 wird auf TRUE gesetzt um die SPI Kommunikation mit dem PiXtend V2 -S-Mikrocontroller zu aktivieren
- Eingang GVL.xDIO wird auf Ausgang GVL.xD00 geschrieben
- Eingang GVL.xDI1 wird auf Ausgang GVL.xD01 geschrieben

Klicken Sie in der Menüleiste auf Erstellen → Übersetzen, um das Programm zu kompilieren.

### 7.5.1.7 Verbindung mit PiXtend V2 -S- und Programm Download

Wenn das Programm ohne Fehler kompiliert ist, doppelklicken Sie im Projektbaum auf „Device (CODESYS Control for Raspberry Pi)“ und anschließend auf den Button „Netzwerk durchsuchen“ im Reiter „Kommunikation“.

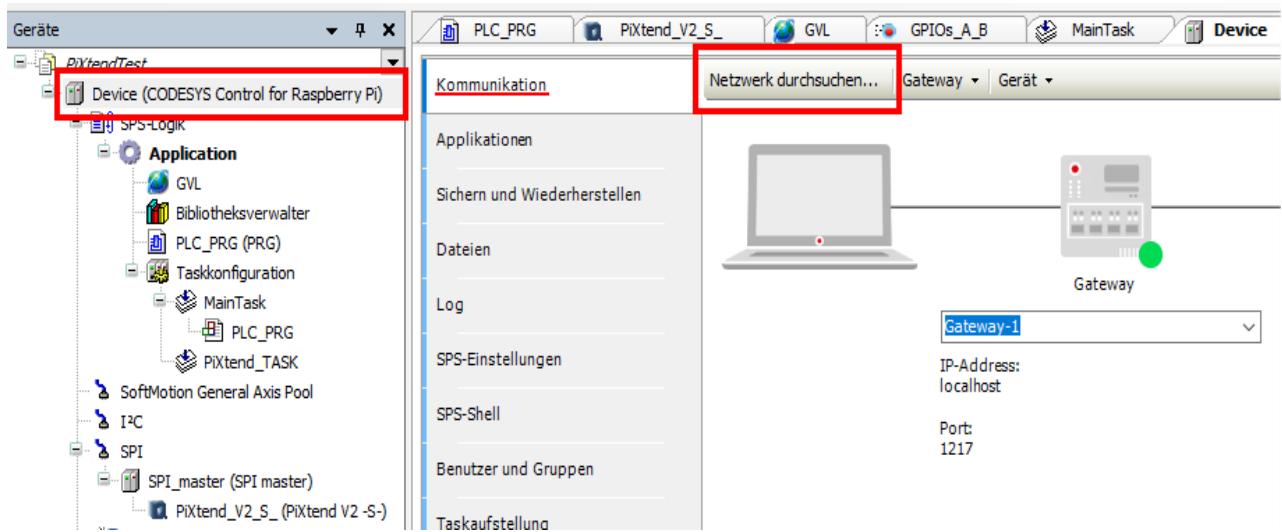


Abbildung 25: CODESYS - Kommunikation - Netzwerk durchsuchen...

CODESYS sucht nun in Ihrem lokalen Netzwerk nach einem Raspberry Pi auf dem die CODESYS Runtime Erweiterung läuft.

Wird kein Device gefunden, prüfen Sie bitte folgende Punkte:

- Ist die korrekte SD-Karte mit dem Image für die CODESYS Laufzeit Erweiterung eingelegt? Ein vorgefertigtes Image finden Sie im Download Bereich ([www.pixtend.de/downloads](http://www.pixtend.de/downloads)). Zur Erstellung eines eigenen CODESYS Images für den Raspberry Pi lesen Sie bitte die PDF-Anleitung, die Sie im CODESYS Store auf der Download-Seite des CODESYS Control for Raspberry Pi SL finden.
- Ist der Raspberry Pi eingeschaltet und besitzt er eine gültige IP, die Sie von ihrem PC aus pingen können? (via ping Befehl aus der Windows Kommandozeile)
- Die IP Ihres Raspberry Pi erfahren Sie mit dem Befehl „ifconfig“ in der Raspberry Pi Kommandozeile. Kennen Sie die IP nicht, benötigen Sie Bildschirm und Tastatur, um direkt auf dem Raspberry Pi nachzuschauen zu können.
- Falls Sie hier eine gültige IP sehen, der Ping jedoch nicht erfolgreich ist, prüfen Sie die Netzwerkverbindung zu Ihrem Raspberry Pi
- Wenn das Raspberry Pi länger als 2 Stunden eingeschaltet war, beendet sich die CODESYS Runtime Erweiterung automatisch und der Raspberry Pi muss neu gestartet werden. Den Befehl können Sie bequem über die Linux-Konsole ausführen:

```
sudo shutdown -r now
```

Wurde der Raspberry Pi gefunden und ausgewählt, klicken Sie im Hauptmenü auf Online → Einloggen und führen Sie einen Download auf das Raspberry Pi durch.

Sind Sie online, dann können Sie mit der Taste F5 in den „Run Modus“ wechseln und die Live-Werte im PiXtend V2 -S-E/A-Abbild einsehen. Klicken Sie im Gerätebaum auf PiXtend V2 -S- und wählen Sie den Reiter „SPI devices E/A-Abbild“.

Unter State → „Firmware“ und „Hardware“ sehen Sie die Firmwareversion des Mikrocontrollers und die Revision der Hardware:

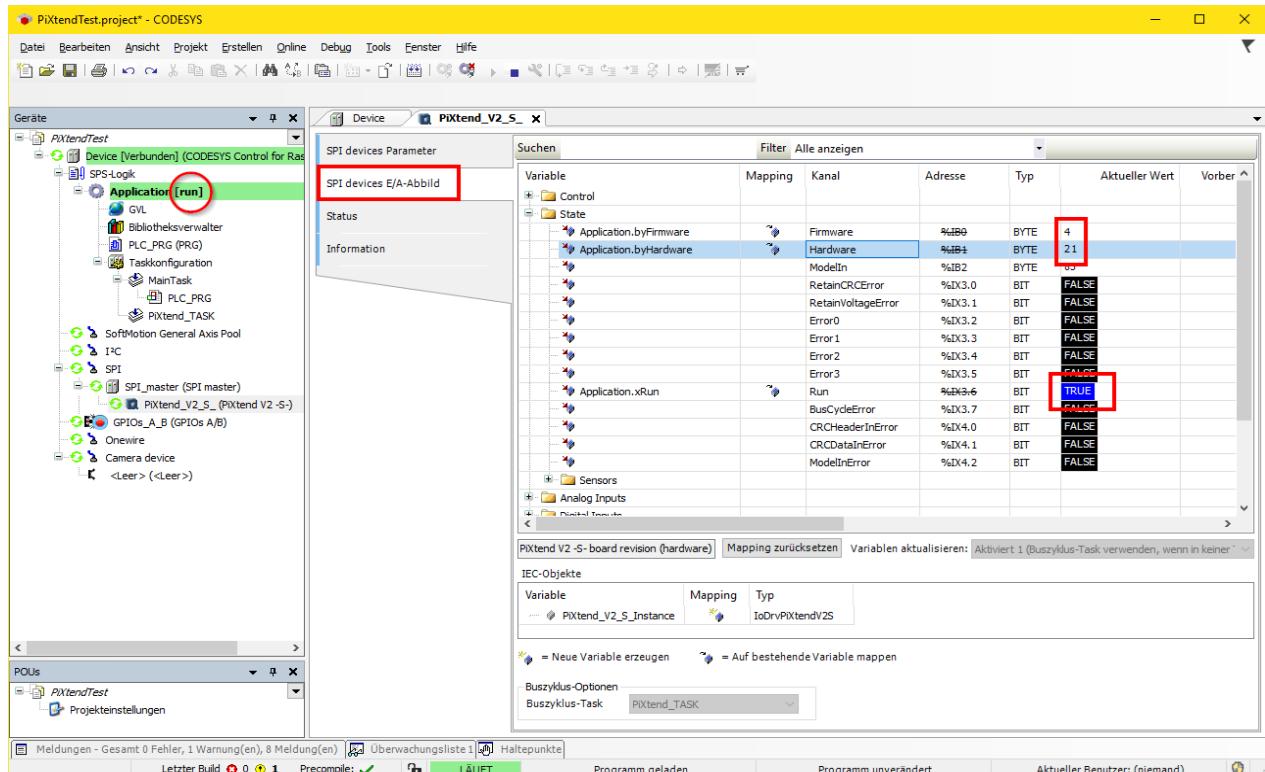


Abbildung 26: CODESYS – PiXtend V2 -S- E/A-Abbild online

Es handelt sich hier um ein PiXtend V2 -S- Board der Revision 1 (Hardware = 21) und um die Mikrocontroller Firmware-Version 4. Wenn Sie ein überarbeitetes PiXtend V2 besitzen, beispielsweise nach einer Optimierung oder Erweiterung, sind die Versionen entsprechend verschieden.

Der Kanal „Run“ sollte im Normalbetrieb „True“ sein. Dies bedeutet, dass sich der Mikrocontroller auf PiXtend V2 im „Run Mode“ befindet.

### 7.5.1.8 Weitere Schritte

Testen Sie die Funktionalität ihres Programms indem Sie einen HIGH-Pegel (24V) am digitalen Eingang 0 und 1 anlegen (zum Beispiel mittels eines Tasters, Schalters, oder einer Drahtbrücke). Beachten Sie dazu unbedingt die Hinweise zur Inbetriebnahme sowie das technische Datenblatt im Hardwarehandbuch.

Funktioniert alles, können Sie Ihr Programm PLC\_PRG nach Belieben abändern. Versuchen Sie folgendes:

Fügen Sie einen Baustein aus der Werkzeug Leiste hinzu und weisen Sie ihm den Typ „AND“ zu indem Sie auf die drei Fragezeichen (???) klicken und AND eintippen.

Verbinden Sie die beiden Eingänge des AND Blockes mit GVL.xDIO und GVL.xDI1. Der Ausgang des AND Blockes wird nur TRUE, wenn beide Eingänge gleichzeitig TRUE sind.

Um die Sache etwas spannender zu gestalten fügen Sie zusätzlich einen „Timer On“ Baustein „TON“ ein und geben Sie ihm einen Instanz-Namen (hier „timer\_delay“). Der Timer erwartet eine Zeit (hier t#2s - 2 Sekunden) und ein Eingangssignal.

Verbinden Sie das IN Signal mit dem Ausgang des AND Blocks. Der Ausgang Q wird dem digitalen Ausgang GVL.xD01 zugewiesen.

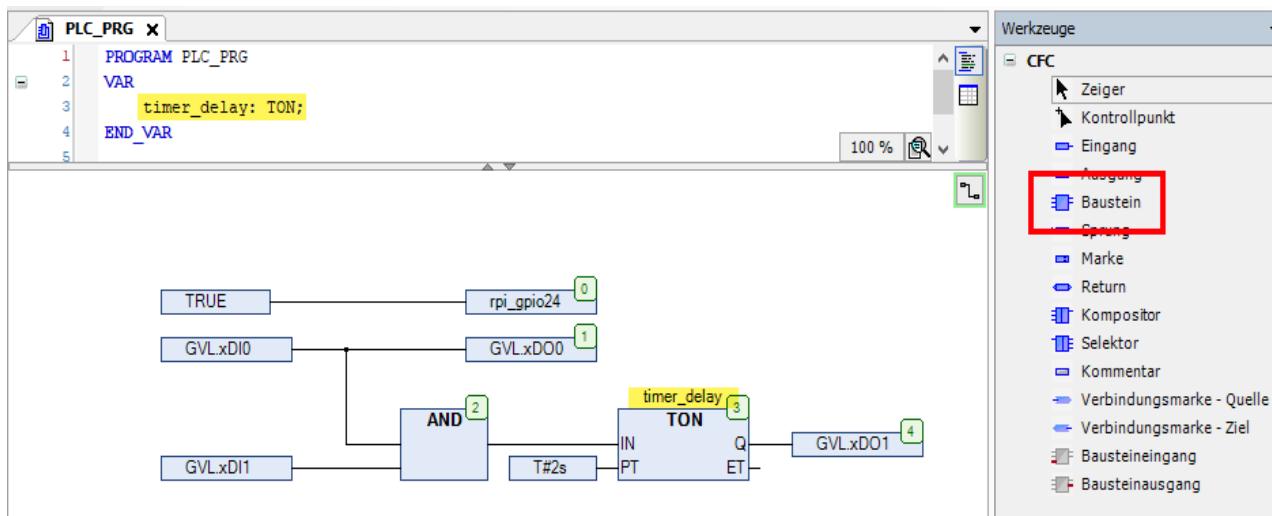


Abbildung 27: CODESYS - PLC\_PRG (CFC) - Demo-Programm

Ein High Pegel an beiden Eingängen GVL.xDIO und GVL.xDI1 für mindestens 2 Sekunden führt dazu, dass der Ausgang GVL.xD01 gesetzt wird. Ist nur ein Eingang HIGH bleibt der Ausgang LOW. Ebenso bleibt der Ausgang LOW solange beide Eingänge HIGH sind und die 2 Sekunden noch nicht abgelaufen sind.

## 7.5.2. Schritt für Schritt zur ersten CODESYS Webvisu

Nun fügen wir dem Projekt eine Visualisierung hinzu damit Sie ihre Steuerung von jedem PC (Smartphone /Tablet) überwachen können.

Klicken Sie rechts im Projektbaum auf Application → Objekt hinzufügen → Visualisierung.

CODESYS erzeugt automatisch den „Visualization-Manager“ und eine leere Visualisierung namens „Visualization“.

Im Manager können Parameter für die Webvisu, beispielsweise der Name der Visualisierung und bevorzugte Auflösung eingestellt werden. Wir belassen es bei den Standardeinstellungen.

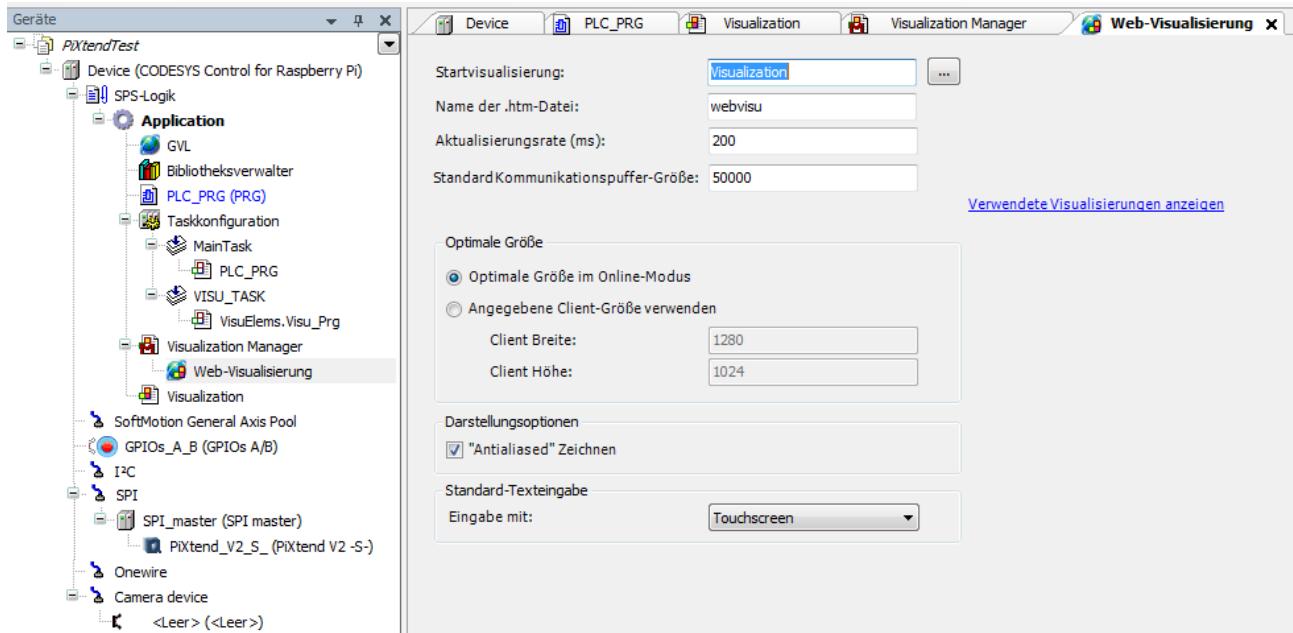


Abbildung 28: CODESYS - Web-Visualisierung - Konfiguration

Ein Doppelklick auf „Visualization“ öffnet den Editor für die Visualisierung. CODESYS besitzt eine Reihe vorgefertigter Steuerelemente. Öffnen Sie in der Werkzeug-Leiste die Gruppe Schalter/Lampen/Bilder und platzieren Sie vier Lampen im Arbeitsbereich.

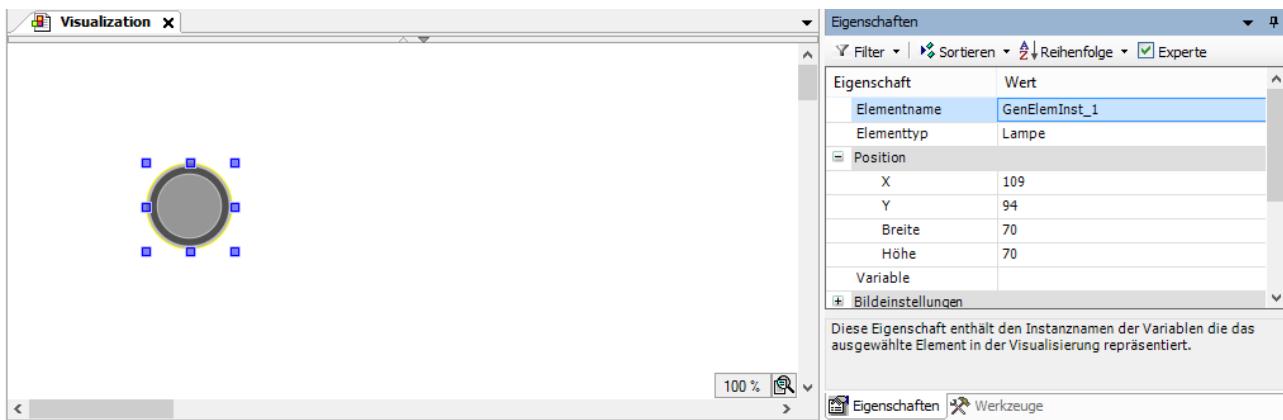


Abbildung 29: CODESYS - Visualisierung

Fügen Sie 4 Beschriftungen hinzu und weisen Sie den Lampen unter der Eigenschaft „Variable“ die jeweiligen Variablen zu (xDI0, xDI1, xD00, xD01).

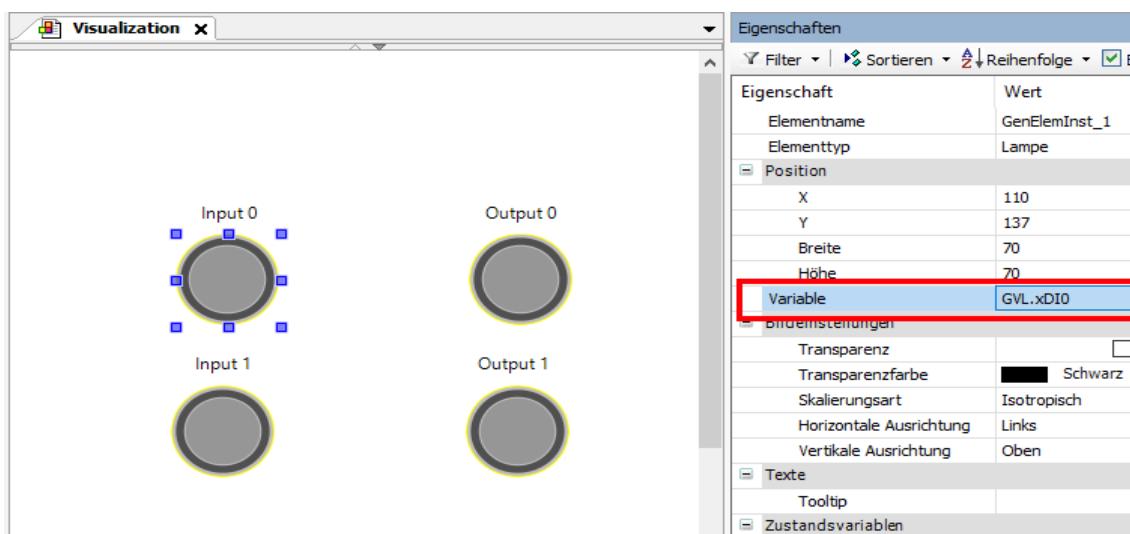


Abbildung 30: CODESYS - Visualisierung - Elementeigenschaften

Beschriftungen sind immer statisch. Um Inhalte von Variablen darzustellen werden in CODESYS Textfelder verwendet.

Ziehen Sie zwei neue Rechtecke in den Arbeitsbereich, geben Sie beim ersten „Firmware: %s“ und beim zweiten „Hardware: %s“ für die „Text“ Eigenschaft ein. Somit erzeugen Sie Platzhalter für zwei Byte Variablen.

Setzen Sie den Haken bei „Experte“. Nun fügen Sie für „Textvariablen“ die beiden Variablen für die Firmware Version und Hardware Revision, byFirmware und byHardware, beim jeweiligen Rechteck ein.

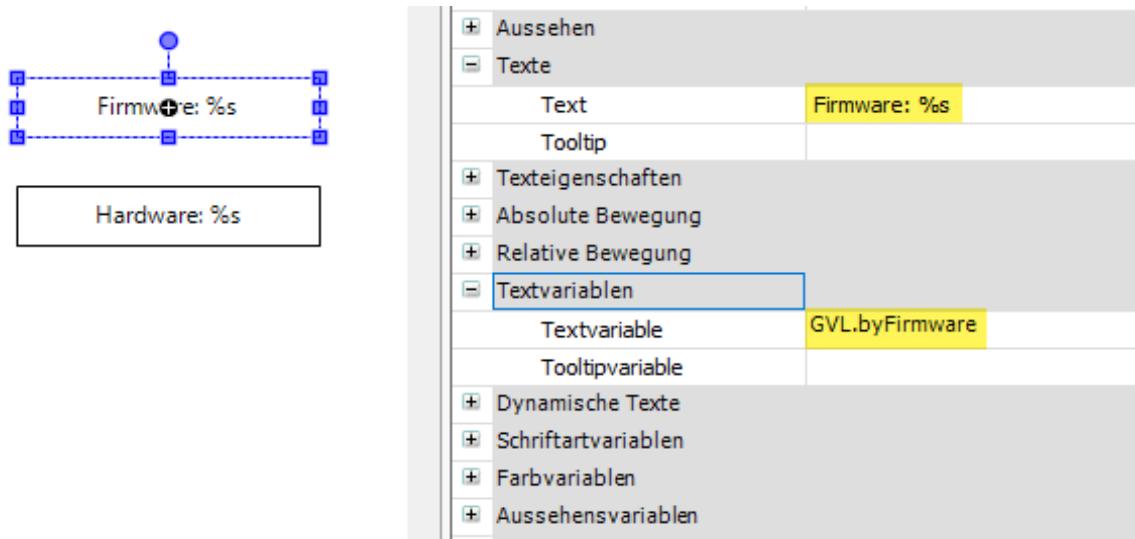


Abbildung 31: CODESYS - Visualisierung - Konfiguration eines Rechtecks

Im Live-Betrieb werden die beiden Platzhalter %s entsprechend mit den Textvariablen befüllt. Weitere Platzhalter sind %d für Zahlen, bzw. %f für Gleitkommazahlen (REAL, LREAL). Formatierungsoptionen wie in C sind möglich.

Abschließend fügen wir noch eine Überschrift und ein animiertes CODESYS-Logo (Spezielle Steuerelemente - Wartesymbol Würfel) hinzu um das ganze optisch aufzuwerten:

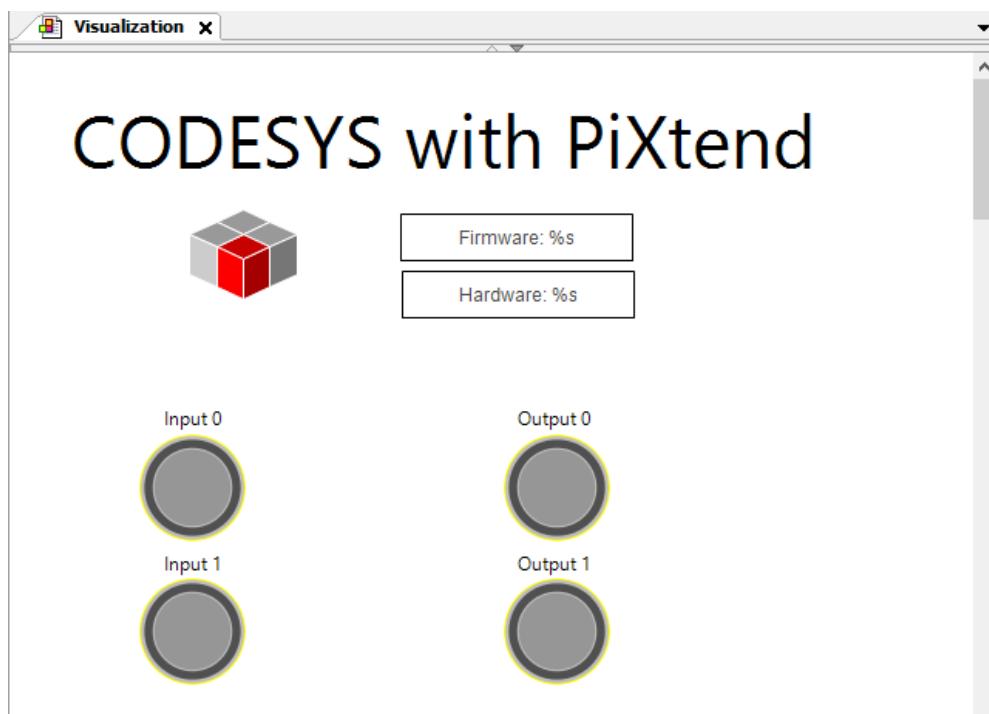


Abbildung 32: CODESYS - Visualisierung - Fertiggestellt

Kompilieren Sie das Projekt erneut mittels Erstellens → Übersetzen und führen Sie einen kompletten Download durch.

Öffnen Sie einen Browser Ihrer Wahl auf Ihrem PC oder Smartphone/Tablet und geben Sie die IP Ihres Raspberry Pi ein, gefolgt von „.:8080/webvisu.htm“, also z.B.:  
<http://192.168.1.99:8080/webvisu.htm>

## 7.6. Schritt für Schritt zum ersten DAC Programm

### 7.6.1. CODESYS Standard Projekt für PiXtend erzeugen

→ Starten Sie CODESYS.

Erstellen Sie ein neues Projekt indem Sie im Hauptmenü auf Datei → Neues Projekt klicken (Shortcut Strg+N)

Wählen Sie „Standard Projekt“ aus der Kategorie „Projekte“ und geben Sie dem Projekt einen Namen (hier „PiXtendDACTest“) bestätigen Sie anschließend mit „OK“.

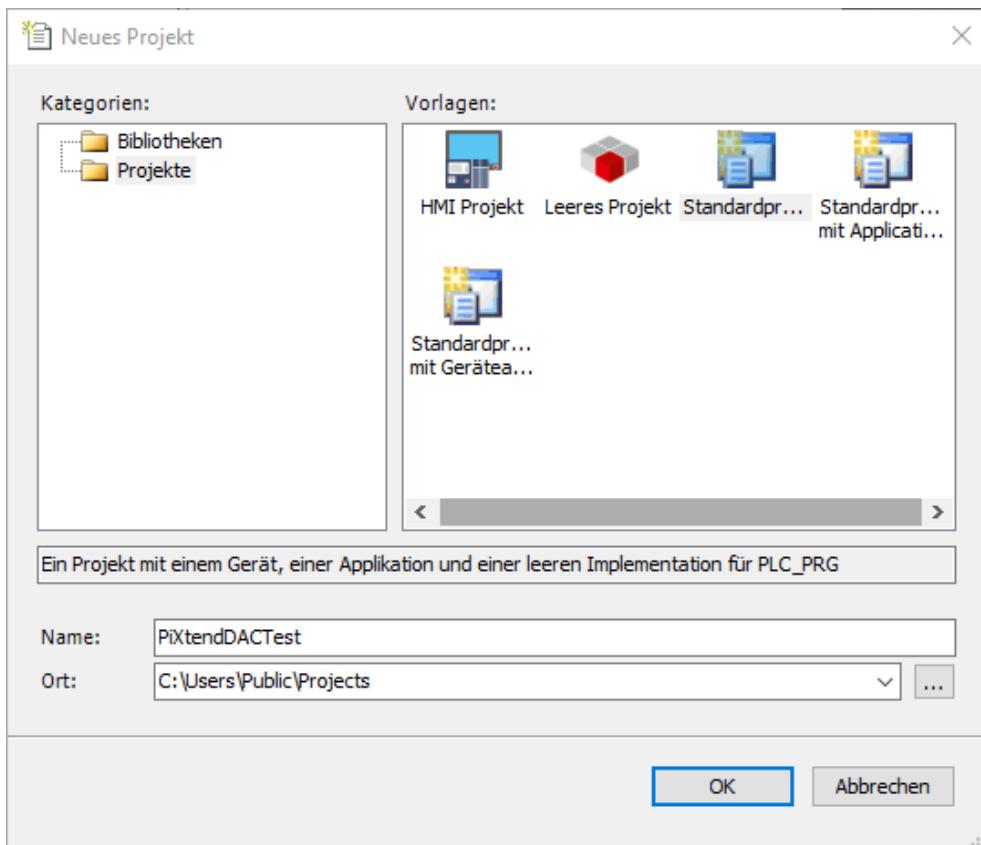


Abbildung 33: CODESYS - DAC - Neues Projekt erstellen

Als Gerät wählen Sie „CODESYS Control for Raspberry Pi“ und als Programmiersprache für das Hauptprogramm PLC\_PRG wählen Sie „Continous Function Chart (CFC)“ aus.

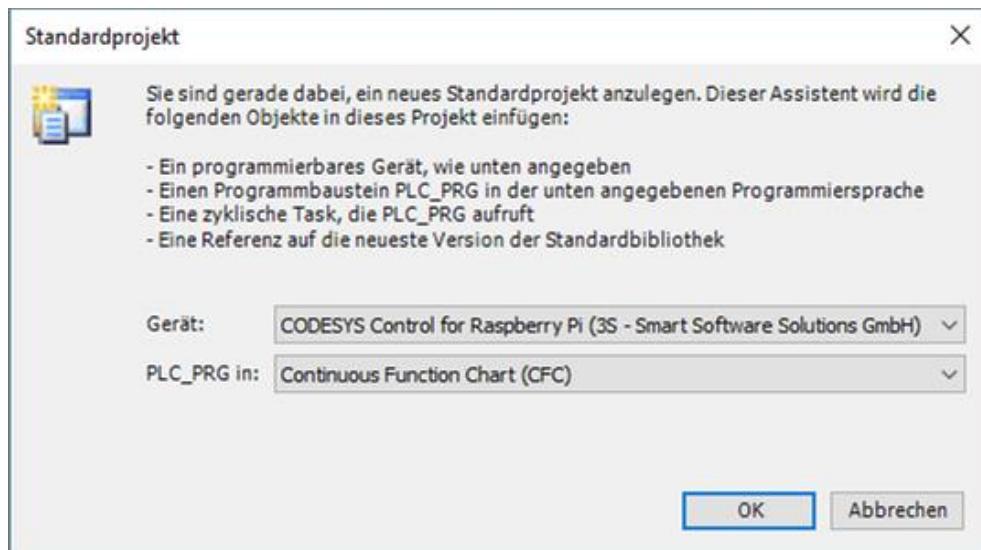


Abbildung 34: CODESYS - DAC - Gerät auswählen

Nachdem CODESYS das Standard Projekt angelegt hat, erhalten Sie ein Projekt mit folgender Struktur:

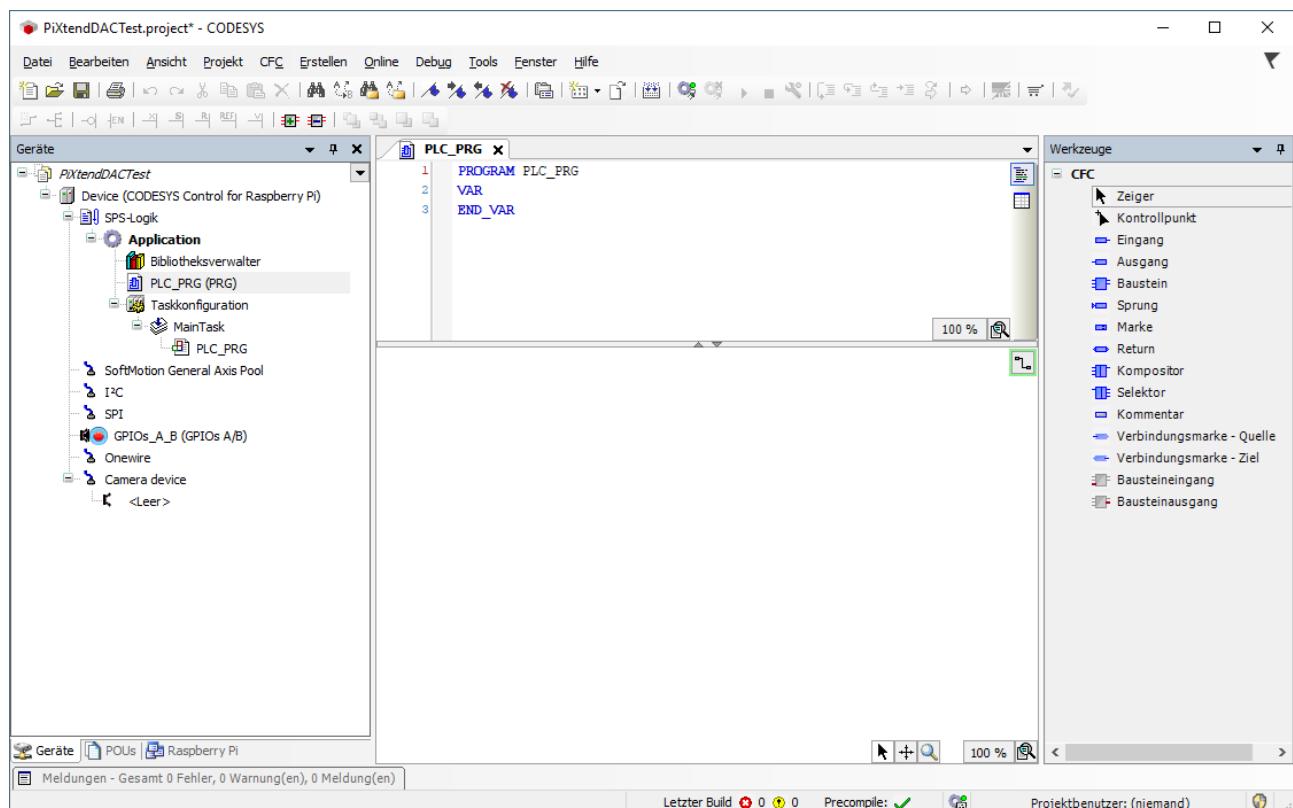


Abbildung 35: CODESYS - DAC - Projektübersicht

## 7.6.2. SPI Gerät anhängen

Im Projektbaum führen Sie einen Rechtsklick auf den Eintrag „SPI“ aus und wählen „Gerät anhängen“:

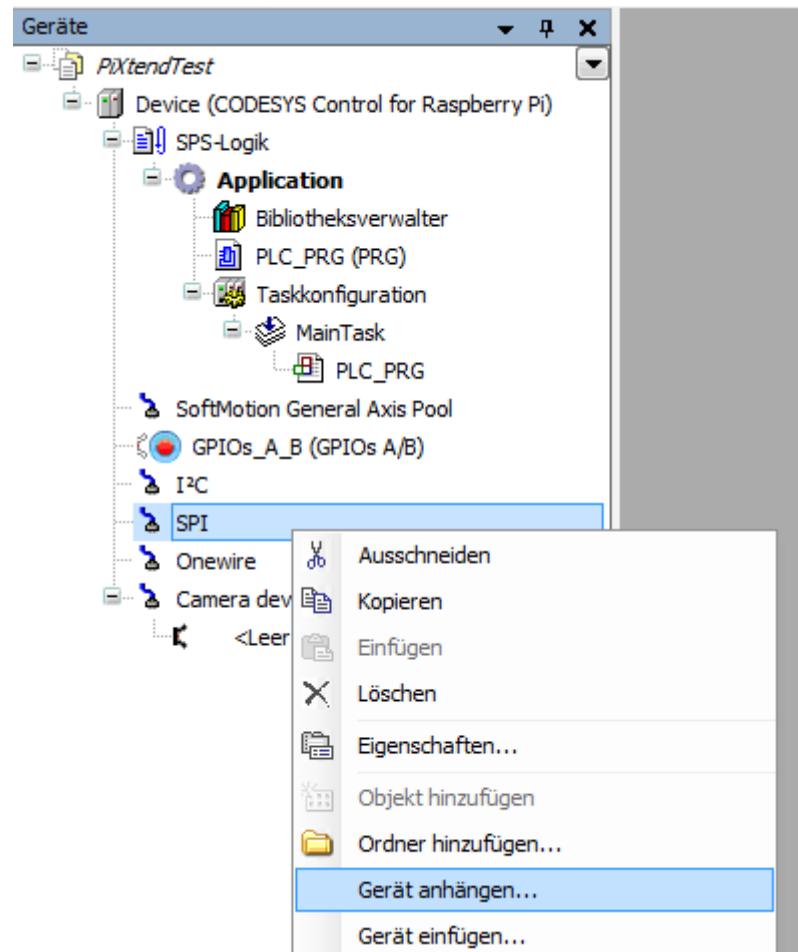


Abbildung 36: CODESYS - DAC - SPI Gerät anhängen

Wählen Sie „SPI master“ als Gerät aus und klicken Sie auf den Button „Gerät anhängen“.

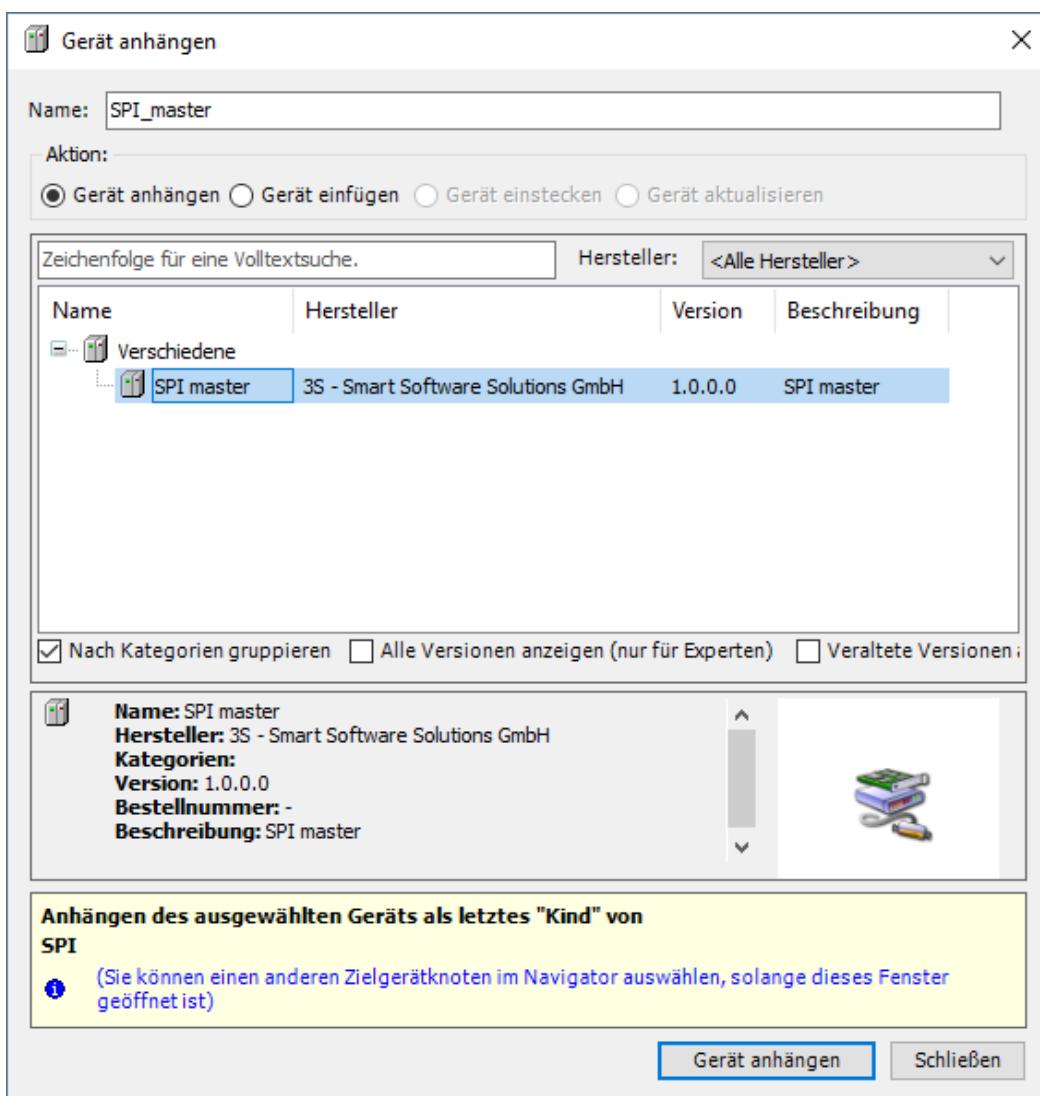


Abbildung 37: CODESYS - DAC - SPI master anhängen

Im Projektbaum unter SPI existiert nun ein neuer Eintrag „SPI\_master (SPI Master)“.

Da der PiXtend V2 DAC am Raspberry Pi über den SPI port 0.1 angeschlossen ist, müssen Sie die Parameter des SPI masters anpassen. Führen Sie einen Doppelklick auf den Eintrag „SPI\_master“ aus und ändern Sie den SPI port auf „/dev/spidev0.1“: ab.

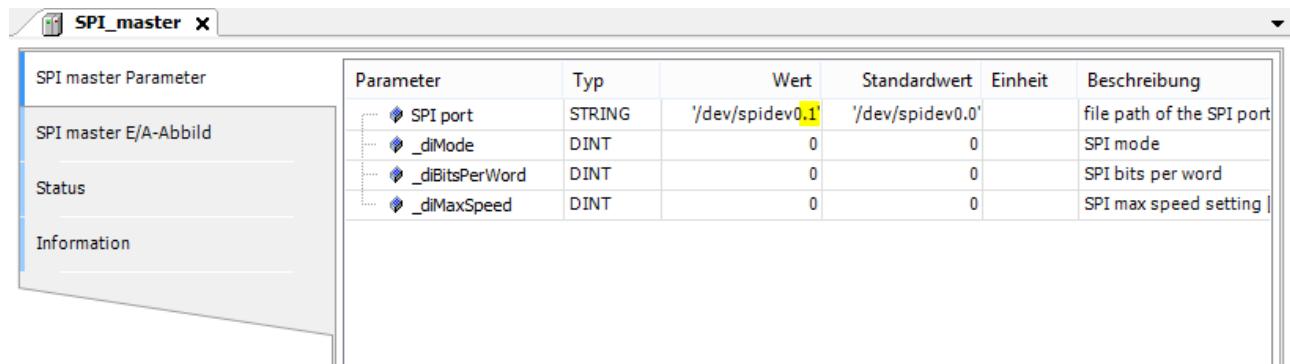


Abbildung 38: CODESYS - DAC - SPI master konfigurieren

### NOTICE

Der „SPI port“ muss auf den Gerätepfad

/dev/spidev0.1 eingestellt sein, sonst erfolgt keine Kommunikation mit dem DAC auf dem PiXtend V2 Board.

### 7.6.3. Globale Variablen Liste erzeugen

Klicken sie auf den Eintrag „Application“, rechts im Projektbaum und fügen Sie mit „Objekt hinzufügen“ -> „Globale Variablen Liste“ eine neue Globale Variablen Liste mit der Bezeichnung „GVL“ hinzu.

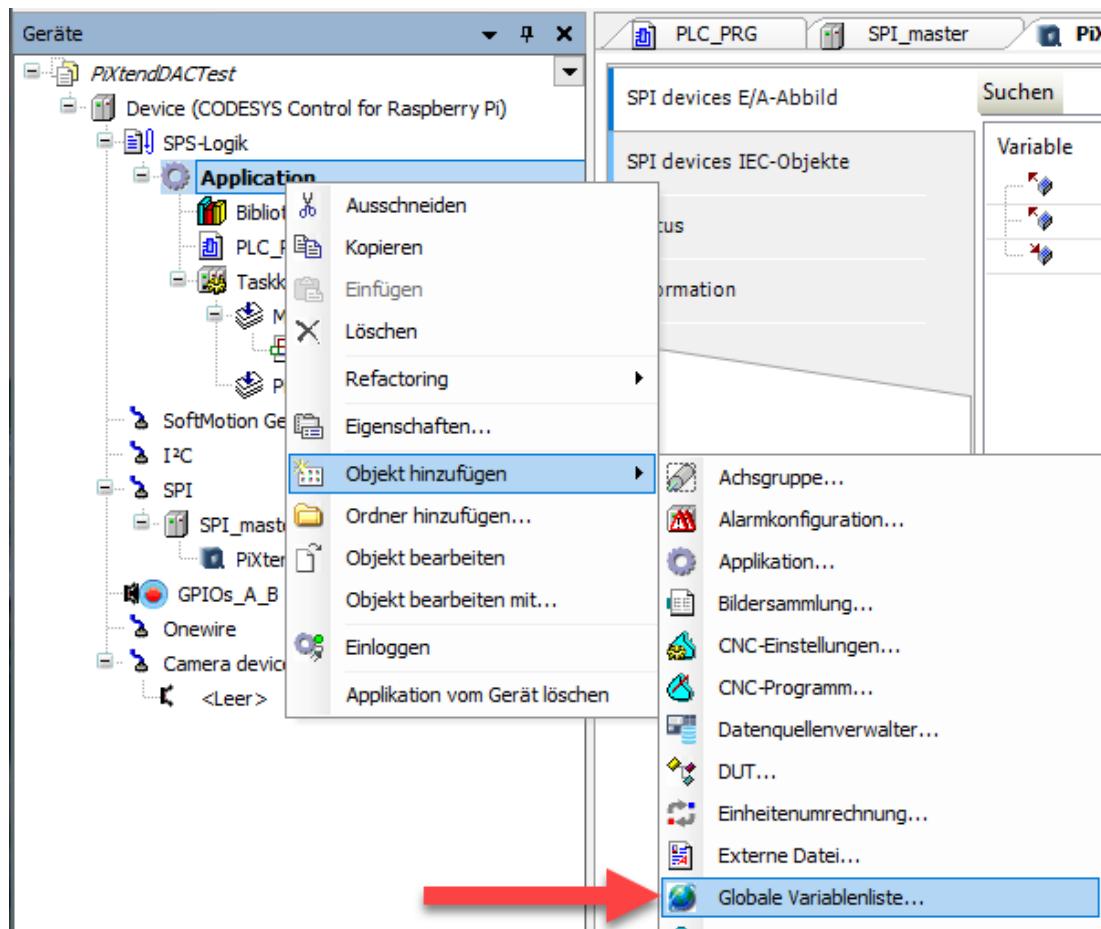


Abbildung 41: CODESYS - DAC - Globale Variablen Liste hinzufügen

Öffnen Sie die GVL und fügen Sie folgende Einträge hinzu:

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

The screenshot shows the 'GVL' editor window. It contains a single variable declaration:

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

Abbildung 42: CODESYS - DAC - GVL Variablen

## 7.6.4. Mapping der Variablen

Haben Sie die benötigten Variablen in der GVL angelegt, müssen diese entsprechen „gemapped“, sprich den Ausgängen des PiXtend V2 -S- DAC (hier in unserem Beispiel für ein PiXtend V2 -S- Board) zugewiesen werden.

Öffnen Sie dazu das Ihnen schon bekannte „SPI devices E/A-Abbild“ indem Sie auf das „PiXtend\_V2\_S\_DAC“ Gerät doppelklicken. Weisen Sie die beiden Variablen den Ausgängen zu

Variable	Mapping	Kanal	Adresse	Typ	Einheit	Beschreibung
Application.GVL.rAOut0		AnalogOut0	%Q0@0	REAL	Volts [V]	Analog Output 0 - This v...
Application.GVL.rAOut1		AnalogOut1	%Q0@1	REAL	Volts [V]	Analog Output 1 - This v...
		BusCycleError	%IX0.0	BIT		The cycle time (interval)

Abbildung 43: CODESYS - DAC - Variablenmapping

indem Sie auf die noch leere linke Spalte einen Doppelklick ausführen. Es erscheinen 3 Punkte (...), mit einem Klick darauf öffnet sich die Eingabehilfe, wählen Sie die gewünschte Variable aus.

Wählen Sie die Variable Application.GVL.rAOut0 für AnalogOut0

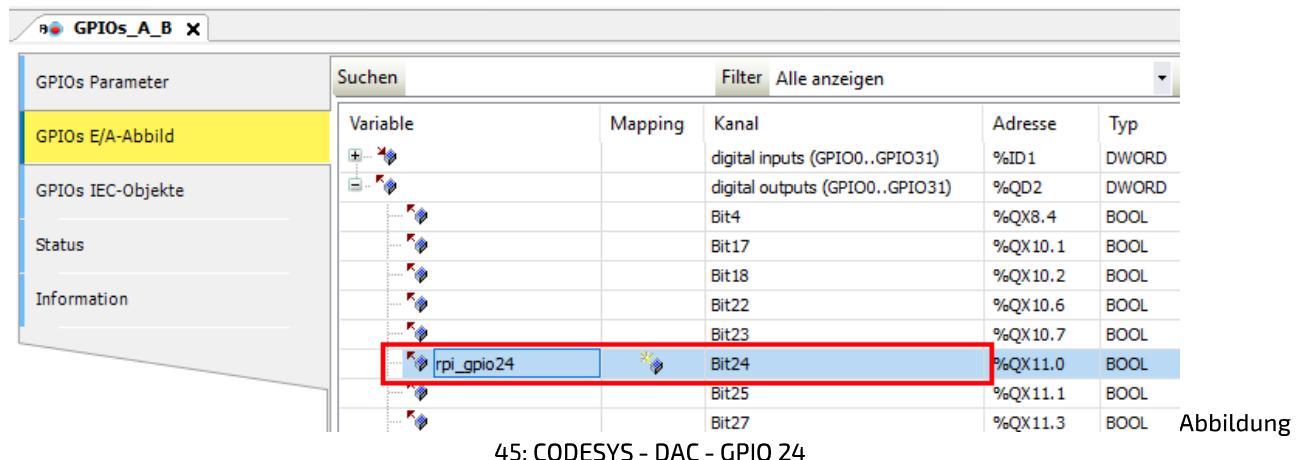
Wählen Sie die Variable Application.GVL.rAOut1 für AnalogOut1

Zuletzt sollte das „GPIO Bit 24“ des Raspberry Pi als Ausgang konfiguriert werden. Öffnen Sie dazu die Raspberry Pi GPIO Konfiguration indem Sie auf „GPIOs\_A\_B“ im Projektbaum doppelklicken. Wählen Sie „Output“ für „GPIO24“.

Parameter	Typ	Wert	Standardwert	Einheit
GPIO4	Enumeration of BYTE	not used	not used	
GPIO17	Enumeration of BYTE	not used	not used	
GPIO18	Enumeration of BYTE	not used	not used	
GPIO22	Enumeration of BYTE	not used	not used	
GPIO23	Enumeration of BYTE	not used	not used	
GPIO24	Enumeration of BYTE	Output	not used	
GPIO25	Enumeration of BYTE	not used	not used	
GPIO27	Enumeration of BYTE	not used	not used	
GPIO28	Enumeration of BYTE	not used	not used	

Abbildung 44: CODESYS - DAC - GPIO 24 als Ausgang verwenden

Wechseln Sie anschließend auf den Reiter „GPIO E/A -Abbild“ und klicken Sie auf „digital outputs“. Als Variable für „Bit 24“ geben Sie „rpi\_gpio24“ ein.



GPIOs Parameter		Suchen	Filter	Alle anzeigen	
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	

45: CODESYS - DAC - GPIO 24

### NOTICE

Der „GPIO 24“ muss im Programm immer auf „TRUE“ stehen damit eine Kommunikation mit dem PiXtend V2 DAC überhaupt möglich ist. Setzen Sie zuerst die Variable „rpi\_gpio24“ in Ihrem Programm auf „TRUE“ damit dies später nicht vergessen wird.

## 7.6.5. PiXtend V2 DAC Gerät anhängen

Markieren Sie den „SPI\_master“ Eintrag im Projektbaum und hängen Sie ein weiteres Gerät an (Rechtsklick>Gerät anhängen).

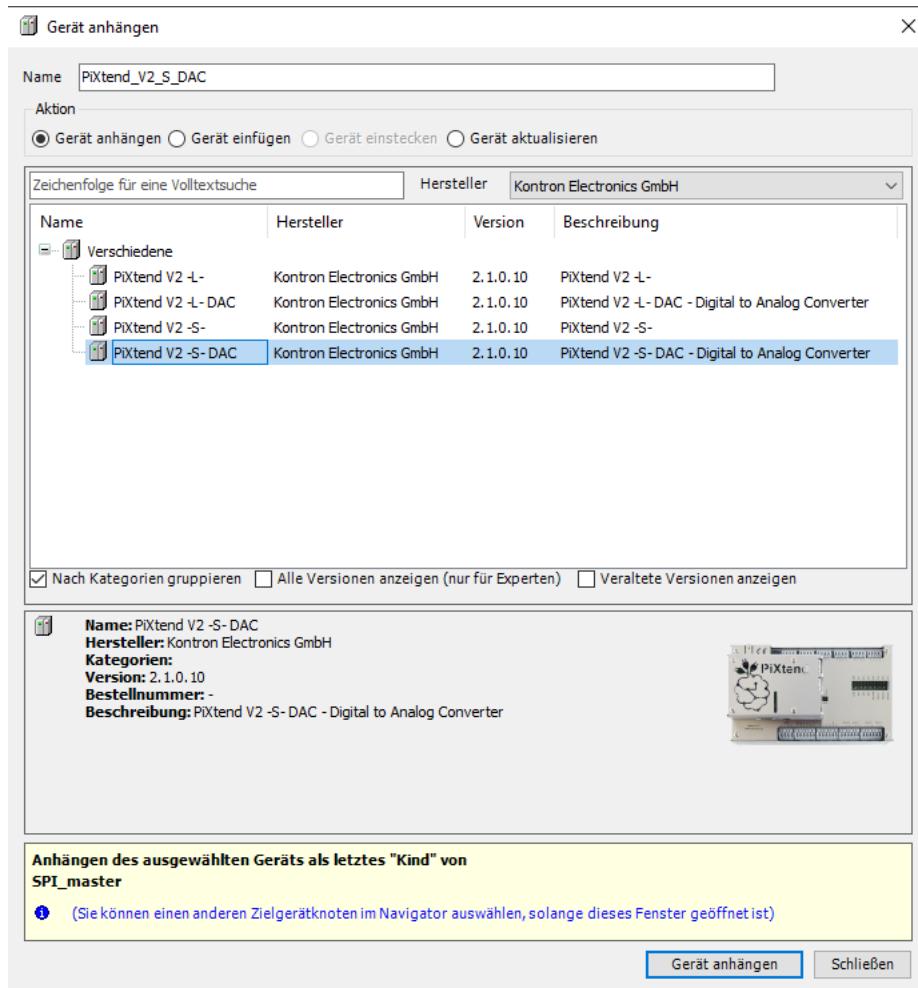


Abbildung 39: CODESYS - DAC - PiXtend V2 DAC anhängen

Wählen Sie im Geräte Hersteller DropDownList-Menü den Eintrag „Kontron Electronics GmbH“ aus. Anschließend wählen Sie als Gerät „PiXtend V2 -S- DAC“, für ein PiXtend V2 -S- Board oder „PiXtend V2 -L- DAC“ für ein PiXtend V2 -L- Board (nicht „PiXtend V2 -S-“ oder „PiXtend V2 -L-“), klicken Sie rechts unten auf den Button „Gerät anhängen“ und schließen Sie das Fenster.

Nun erscheint der PiXtend V2 -S- DAC bzw. PiXtend V2 -L- DAC als Gerät unter „SPI\_master (SPI Master)“.

Ein Doppelklick auf das „PiXtend\_V2\_S\_DAC“ (in unserem Beispiel, bei PiXtend V2 -L- DAC kann der Name abweichen) Ein Klick auf Gerät im Projektbaum öffnet die Konfigurationsseite für das Gerät. Im Reiter „SPI devices E/A-Abbild“ sehen Sie die beiden analogen Ausgänge die der PiXtend V2 DAC für die Verwendung in CODESYS bereitstellt.

Damit dieses Prozessabbild zyklisch ausgetauscht wird, sind noch zwei Änderungen notwendig. Wählen Sie für „Variablen aktualisieren“ bitte den Eintrag „Aktiviert 1 (Buszyklus-Task verwenden, wenn in keinem Task verwendet wird)“ sowie für Buszyklus-Task den „PiXtend\_TASK“ aus, sollte dies noch nicht der Fall sein.

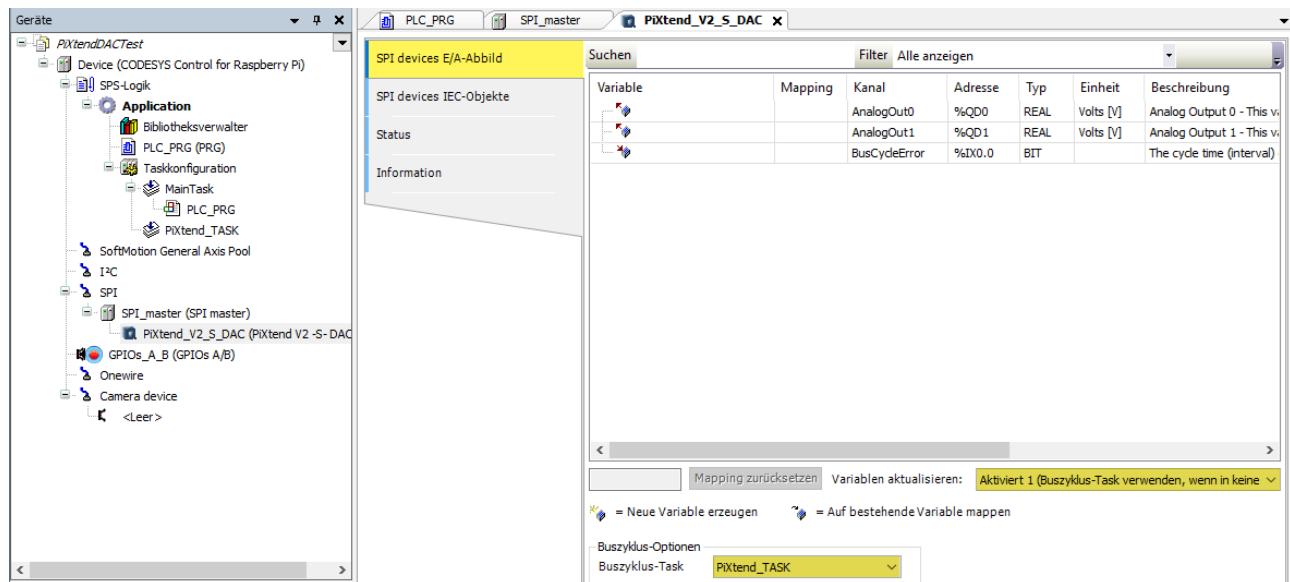


Abbildung 40: CODESYS - DAC - E/A Abbild Einstellungen

Später im laufenden Betrieb („Online zur Steuerung“) können Sie in diesem Fenster den Ausgängen direkt Werte zuweisen (mit F7 Werte „forcen“ - deutsch: erzwingen).

Da wir aus der Visualisierung direkt auf die Ausgänge zugreifen können, erzeugen wir eine Globale Variablen Liste (GVL) um den Ausgängen Variablen zuzuweisen.

## 7.6.6. Task Konfiguration

Falls gewünscht, können Sie die Zykluszeit des PiXtend\_TASK anpassen. Klicken Sie dazu auf die Taskkonfiguration → PiXtend\_TASK. Prinzipiell kann die Zykluszeit bis auf t#1ms reduziert werden, für dieses Test Projekt ist die Voreinstellung von 30 ms ausreichend. Möchten Sie später das PiXtend V2 Board selbst ansteuern, sollte dieser Wert unverändert bleiben.

## 7.6.7. Erstellung des Hauptprogramms

Führen Sie einen Doppelklick auf PLC\_PRG aus, um den Editor zu öffnen und mit der Erstellung des Programm Codes zu beginnen.

In CFC wird grafisch programmiert. Ziehen Sie zunächst einen „Eingang“ Block per Drag&Drop aus der Werkzeug Box (rechts) und platzieren Sie diesen im Arbeitsbereich.

Fügen Sie anschließend einen „Ausgang“ Block hinzu und verbinden Sie beide miteinander.

Klicken Sie in die Mitte eines Blockes um Variablen oder Konstanten wie in der Abbildung zuzuweisen:

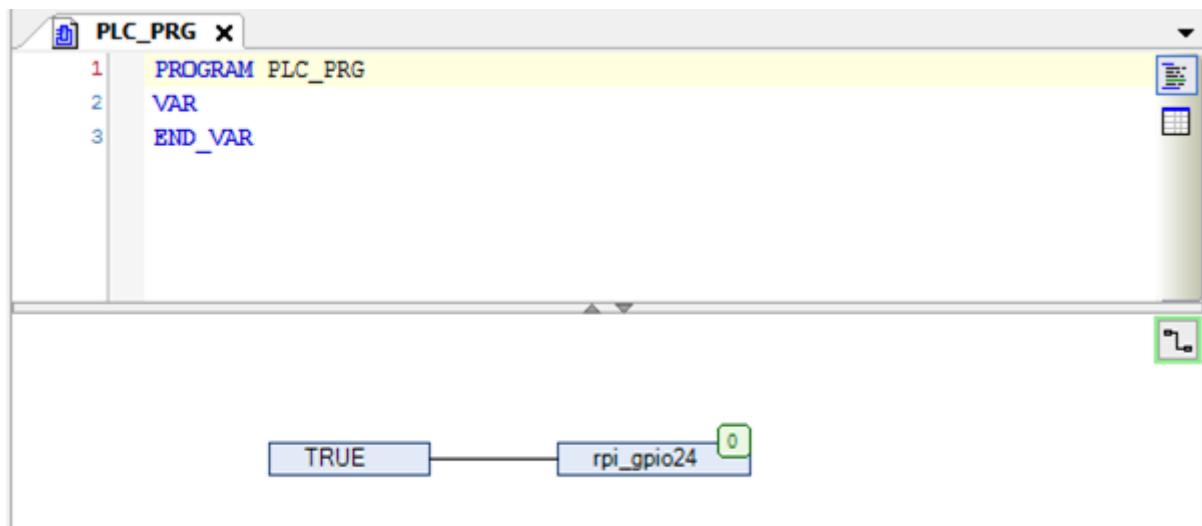


Abbildung 46: CODESYS - DAC - Variable rpi\_gpio24 setzen

Klicken Sie in der Menüleiste auf Erstellen → Übersetzen, um das Programm zu kompilieren.

## 7.6.8. Verbindung mit PiXtend V2 und Programm Download

Ist das Programm ohne Fehler kompiliert, doppelklicken Sie im Projektbaum auf „Device (CODESYS Control for Raspberry Pi)“ und anschließend auf den Button „Netzwerk durchsuchen“.



Abbildung 47: CODESYS - DAC - Netzwerk durchsuchen

CODESYS sucht nun in Ihrem lokalen Netzwerk nach einem Raspberry Pi der mit der CODESYS Runtime Erweiterung ausgestattet ist. Wird kein Device gefunden überprüfen Sie bitte folgende Punkte:

- Ist die korrekte SD-Karte mit dem Image für die CODESYS Laufzeit Erweiterung eingelegt? Ein vorgefertigtes Image finden Sie im Download Bereich ([www.pixtend.de/downloads](http://www.pixtend.de/downloads)). Zur Erstellung eines eigenen CODESYS Images für den Raspberry Pi lesen Sie bitte das entsprechende Kapitel in der CODESYS Online Hilfe durch. Abschnitt „Add-ons“.
- Ist der Raspberry Pi eingeschaltet, besitzt er eine gültige IP-Adresse, die Sie von Ihrem PC aus pingen können? (via ping Befehl aus der Windows Kommandozeile)
- Die IP-Adresse Ihres Raspberry Pi erfahren Sie ebenfalls mit dem Befehl „ifconfig“ direkt in der Raspberry Pi Kommandozeile, per Tastatur und Bildschirm.
- Sofern Sie hier eine gültige IP sehen und der Ping ist nicht erfolgreich, überprüfen Sie die Netzwerkverbindung zu Ihrem Raspberry Pi
- War der Raspberry Pi länger als 2 Stunden eingeschaltet, beendet sich die CODESYS Runtime Erweiterung automatisch und der Raspberry Pi muss neu gestartet werden. (sudo reboot)

Wurde der Raspberry Pi gefunden und ausgewählt, klicken Sie im Hauptmenü auf „Online → Einloggen“ und führen Sie einen Download auf den Raspberry Pi durch.

### 7.6.9. Erstellung der CODESYS Webvisu

Wir wollen dem Projekt nun eine einfache Visualisierung hinzufügen.

Führen Sie im Projektbaum einen Rechtsklick auf: „Application → Objekt hinzufügen → Visualisierung“ aus.

CODESYS erzeugt automatisch den „Visualisierungsmanager“ und eine leere Visualisierung namens „Visualization“.

Im Manager können Parameter für die Webvisu, wie der Name der Visualisierung und die bevorzugte Auflösung eingestellt werden.

Ändern Sie die Skalierungsoptionen auf den Wert „Isotropisch“:

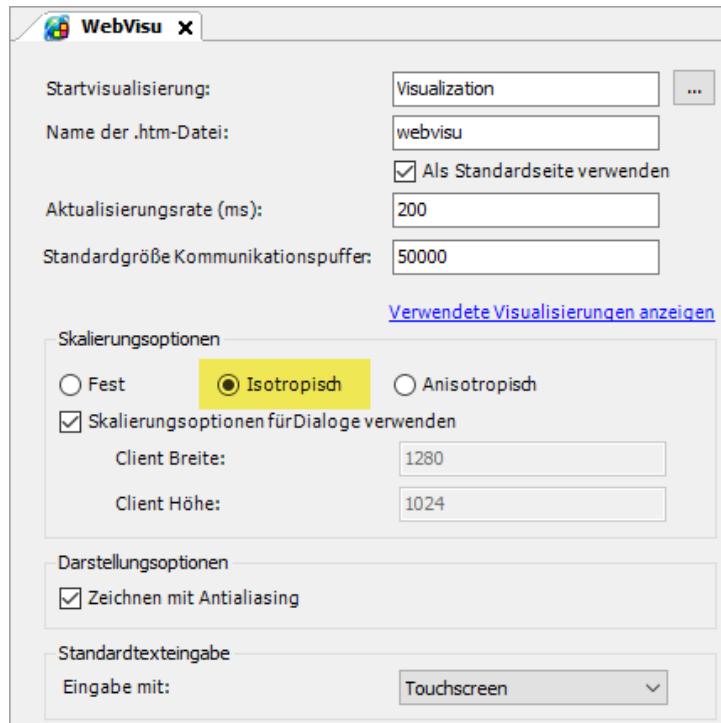


Abbildung 48: CODESYS - DAC - WebVisu Einstellungen

Ein Doppelklick auf „Visualization“ öffnet den Editor für die Visualisierung. CODESYS verfügt bereits über eine Reihe vorgefertigter Steuerelemente.  
Öffnen Sie in der Werkzeugleiste die Gruppe „Allgemeine Steuerelemente“ und platzieren Sie zwei „Schieberegler“ im Arbeitsbereich:

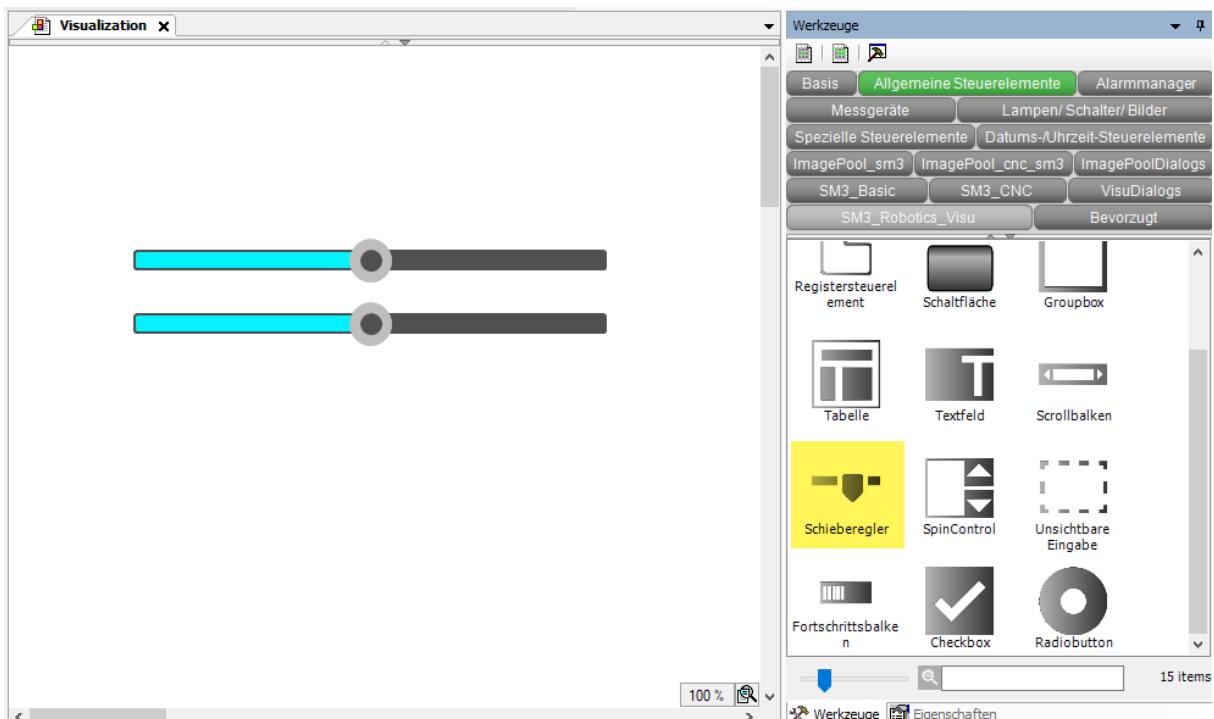


Abbildung 49: CODESYS - DAC - WebVisu - Schieberegler

Fügen Sie zwei Beschriftungen hinzu und weisen Sie den Schieberegler unter Eigenschaften „Variable“ die jeweiligen Variablen zu (GVL.rAOut0, GVLrAOut1).  
Ändern Sie das Skalenende beider Schieberegler auf „10“:

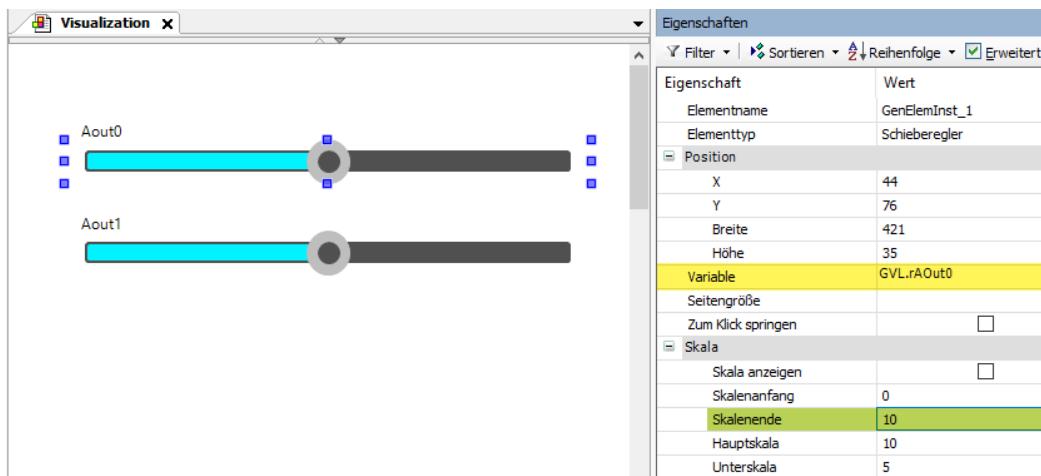


Abbildung 50: CODESYS - DAC - Schieberegler - Skalenende festlegen

Fügen Sie nun zwei Textfelder (Rechtecke) hinzu und geben Sie „%2.1f“ für die „Text“ Eigenschaft ein. Dies erzeugt einen Platzhalter für eine REAL Variable mit zwei Vor- und einer Nachkommastelle.

Weisen Sie der Eigenschaft „Textvariable“ den Wert „GVL.rAOut0“ bzw. „GVL.rAOut1“ zu.

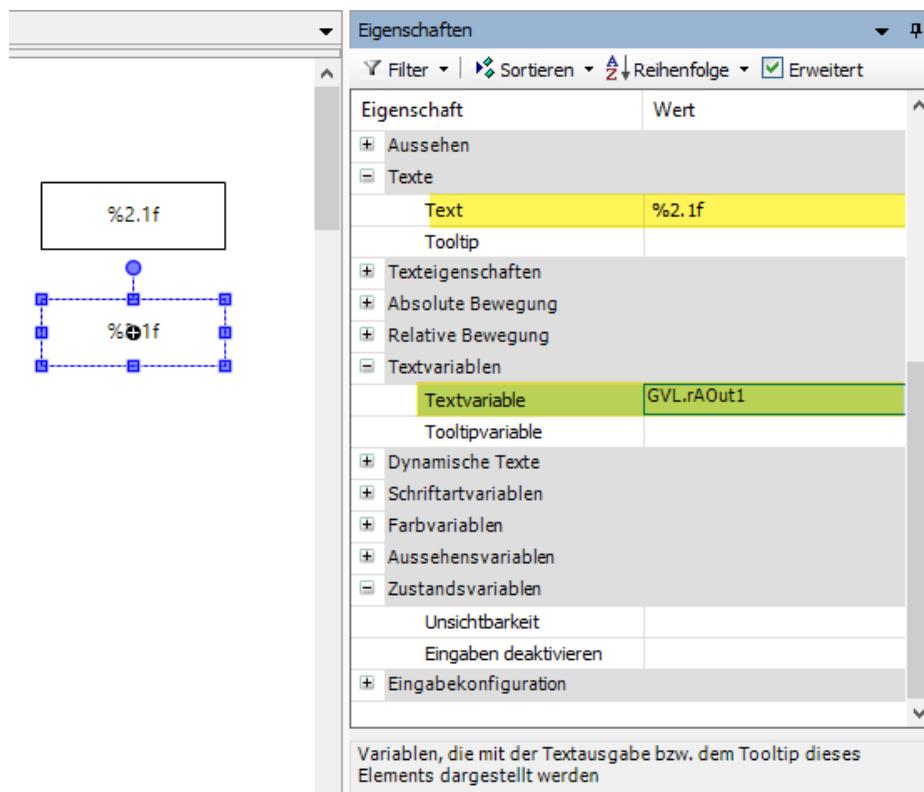


Abbildung 51: CODESYS - DAC - WebVisu - Textfelder einstellen

Abschließend fügen wir eine Überschrift und ein animiertes CODESYS Logo (Spezielle Steuerelemente - Wartesymbol Würfel) hinzu um es optisch aufzuwerten.

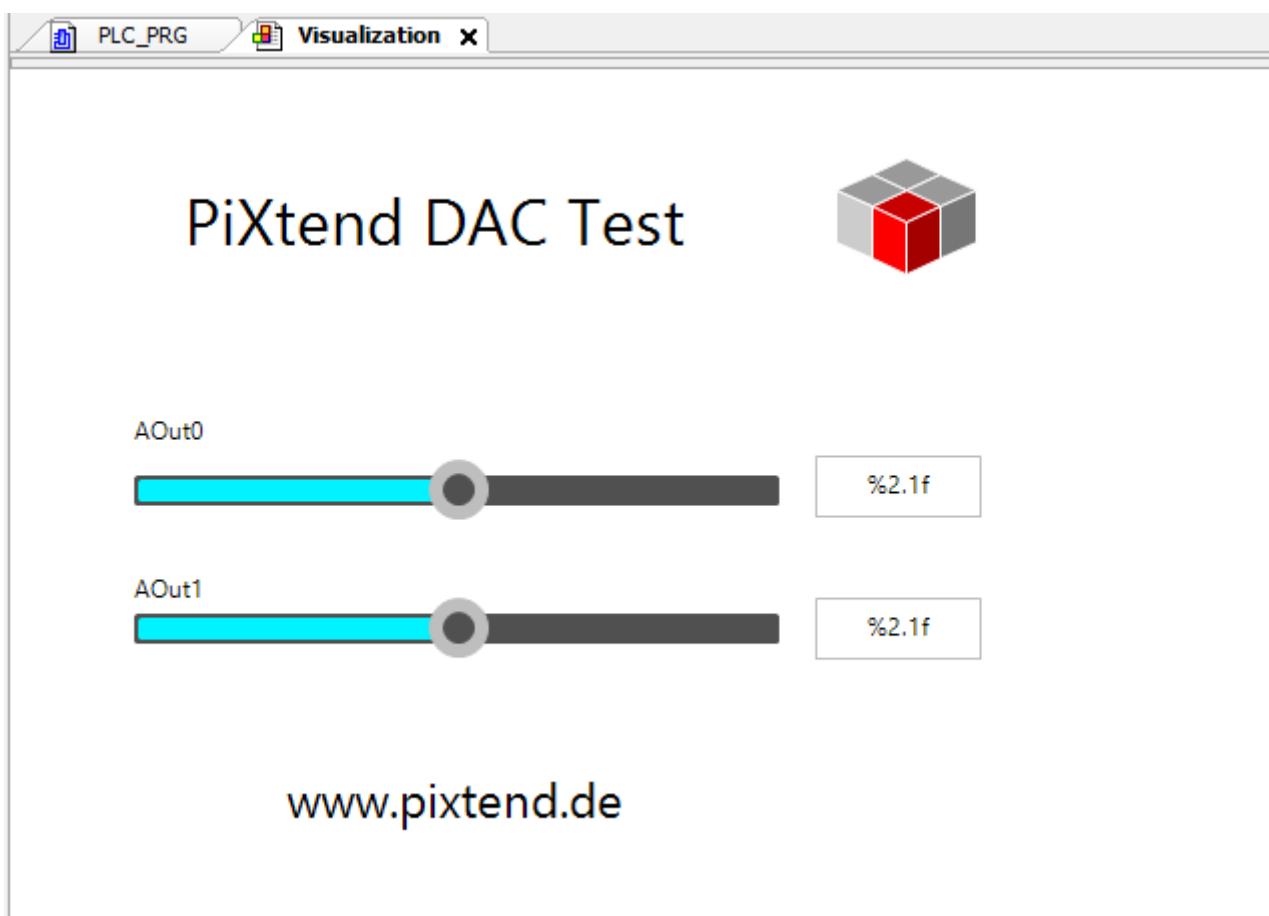


Abbildung 52: CODESYS - DAC - WebVisu - Fertig

Kompilieren Sie das Projekt erneut mittels „Erstellen→Übersetzen“ und führen Sie einen kompletten Download durch.

Öffnen Sie einen Browser Ihrer Wahl auf Ihrem PC oder Smartphone/Tablet und geben Sie die IP Adresse Ihres Raspberry Pi ein, gefolgt von „:8080/webvisu.htm“, z.B. <http://192.168.137.99:8080/webvisu.htm>

Bewegen Sie nun die Schieberegler, um die Spannung der beiden analogen Ausgänge zu verändern.

**NOTICE**

Wird parallel zum PiXtend V2 DAC das PiXtend V2 Board angesprochen, achten Sie darauf, dass Sie das PiXtend V2 Gerät an einen eigenen (zusätzlichen) SPI Master anhängen und nicht an den SPI Master, an dem sich bereits der DAC befindet.

Beispiel für PiXtend V2 -S- (alle Angaben gelten analog auch für PiXtend V2 -L-):

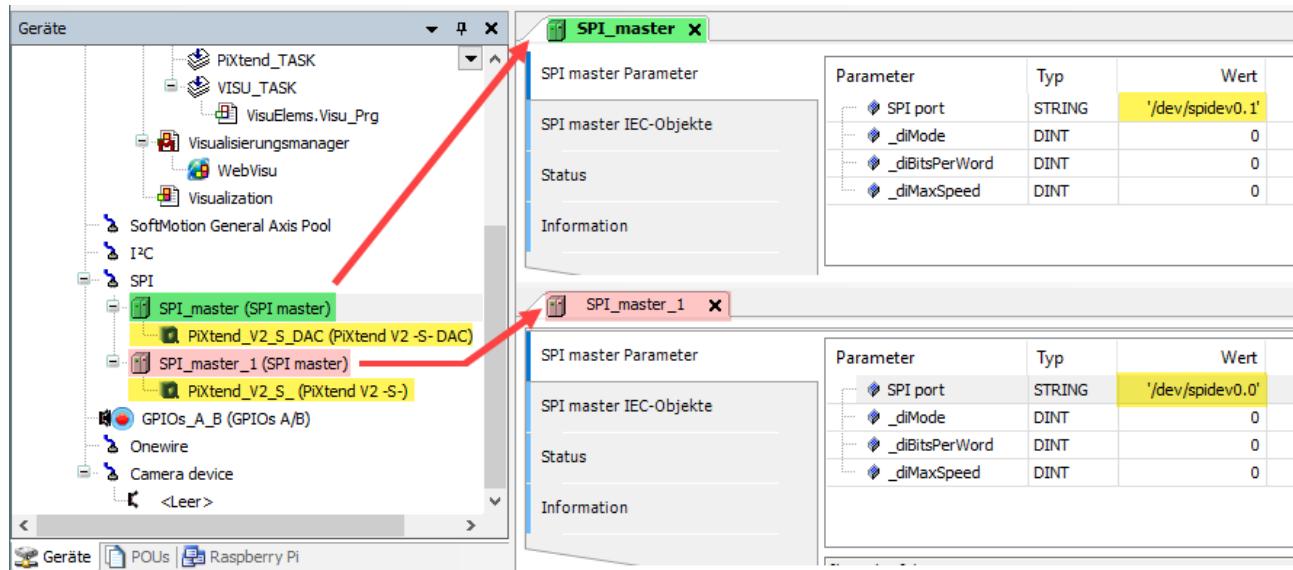


Abbildung 53: CODESYS - Hinweis zur Verwendung beider PiXtend V2 SPI Geräte

Am ersten SPI Master (SPI\_Master, grün hervorgehoben) hängt der PiXtend V2 -S- DAC, in unserem Beispiel haben wir zunächst diesem Gerät den SPI Bus angehängt. Das PiXtend V2 -S- DAC Gerät ist gelb hervorgehoben. Bitte beachten Sie die SPI port Einstellung für den DAC. Die Kommunikation läuft über den zweiten Chip Select des Raspberry Pi, daher muss das Gerät „/dev/spidev0.1“ (gelb hervorgehoben auf der rechten Seite) angegeben werden.

Das PiXtend V2 -S- Board wird über einen weiteren, zweiten SPI Master angesteuert. Im oberen Bild rot hervorgehoben mit dem Namen „SPI\_master\_1“. An diesen zweiten SPI Master kann das PiXtend V2 -S- angehängt (ebenfalls gelb hervorgehoben) und betrieben werden. Die Kommunikation läuft über den ersten Chip Select des Raspberry Pi, daher muss das Gerät „/dev/spidev0.0“ (gelb hervorgehoben auf der rechten Seite, untere Hälfte) angegeben werden.

Sie können selbstverständlich zuerst das PiXtend V2 -S- an den ersten und den DAC an den zweiten SPI Master anhängen, vergessen Sie nicht die SPI port Einstellung für das jeweilige Gerät entsprechend nachzuziehen.

## 7.7. CODESYS Serielle Kommunikation

Dieses Kapitel beschreibt alle notwendigen Schritte um eine Kommunikation zwischen dem Raspberry Pi (CODESYS) und einem Windows PC (Terminal), über die RS232 Schnittstelle von PiXtend V2, einzurichten.

### 7.7.1. Voraussetzungen

Die Schritte in diesem Kapitel setzen voraus, dass Sie die vorhergehenden Kapitel, insbesondere Kapitel 7.2 Voraussetzungen und 7.3 Installation der benötigten Software Komponenten, gelesen und entsprechend das CODESYS Development System mit den 3 Komponenten CODESYS, CODESYS Control for Raspberry Pi sowie PiXtend V2 Professional for CODESYS erfolgreich installiert haben.

### 7.7.2. RS232-Schnittstelle - Anschluss des Kabels

Um die RS232-Schnittstelle zu nutzen, müssen die Datenleitungen überkreuz angeschlossen werden, jeweils RX an TX.

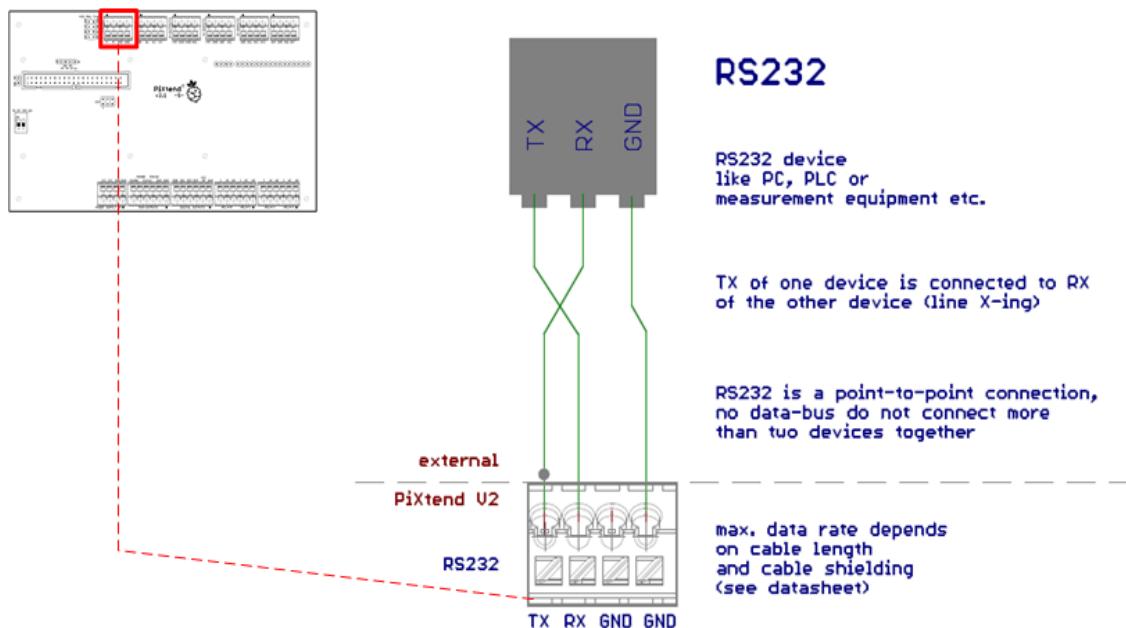


Abbildung 54: CODESYS - Schaltplan RS232

Um eine Verbindung mit einem PC herzustellen wird ein USB-zu-Seriell-Adapter benötigt.

Weitere Informationen zu RS232 können Sie im Datenblatt (PiXtend V2 -S- bzw. PiXtend V2 -L-technisches Datenblatt im Hardwarehandbuch) nachlesen.

### 7.7.3. RS485-Schnittstelle - Anschluss des Kabels (nur PiXtend V2 -L-)

Zur Nutzung der RS485-Schnittstelle müssen die Datenleitungen 1:1 angeschlossen werden, jeweils A (A) an A (A) und B (B) an B (B).

Um die RS485-Schnittstelle zu nutzen, sollte der GPIO 18 des Raspberry Pi aktiviert werden. Dadurch wird die serielle Schnittstelle des Raspberry Pi mit dem RS485 Hardware Chip verbunden. Eine Nutzung der RS232-Schnittstelle ist nach der Umschaltung nicht mehr möglich.

#### NOTICE

Die RS485-Schnittstelle steht nur auf dem PiXtend V2 -L- zu Verfügung!

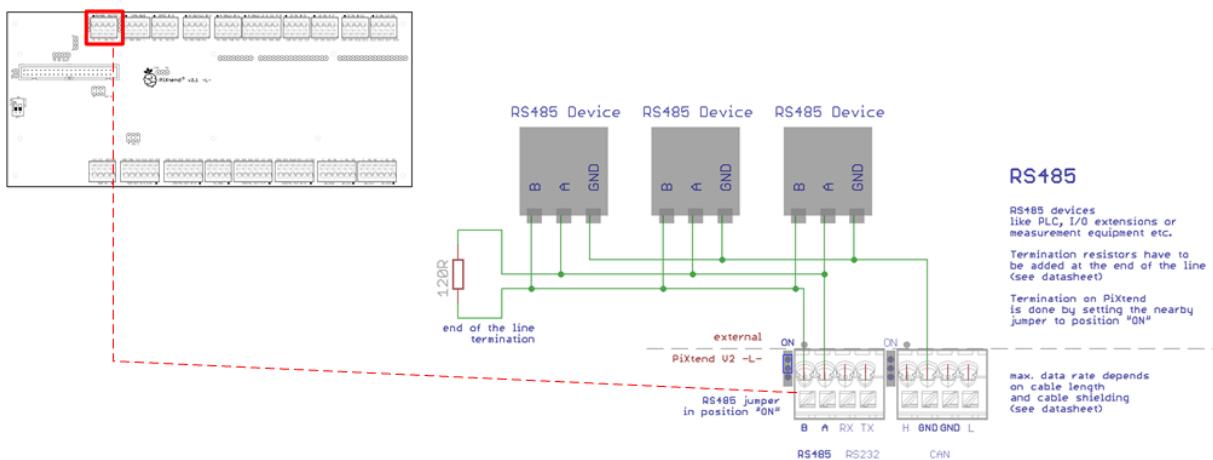


Abbildung 55: CODESYS - Schaltplan / Anschluss RS485

## 7.7.4. Vorbereitung Linux

### 7.7.4.1 Anpassung der Schnittstelleneinstellungen

Zunächst werden die Einstellungen der Raspberry Pi Schnittstellen angepasst, um die serielle Schnittstelle mit CODESYS zu verwenden. Das heißt, die serielle Schnittstelle wird mit den nachstehenden Befehlen für Linux deaktiviert und sendet keine Daten mehr (der Zugriff auf die Linux-Konsole über die serielle Schnittstelle wird deaktiviert).

Die serielle Schnittstelle des Raspberry Pi lässt sich über das Programm „raspi-config“ aktivieren oder umstellen:

```
sudo raspi-config
```

Im Programm wechseln Sie zu:

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

Im Anschluss findet sich in der Datei /boot/config.txt folgender Eintrag:

```
enable_uart=1
```

Bluetooth wird automatisch deaktiviert und UART aktiviert. Das Programm „raspi-config“ übernimmt diese Aufgabe.

Verwenden Sie unser CODESYS SD-Karten Image, dann starten Sie den Raspberry Pi neu und können anschließend mit Kapitel 7.7.5 Terminal Programm fortfahren

### 7.7.4.2 Die Schnittstelle in der CODESYS-Konfiguration aktivieren

Im nächsten Schritt tragen wir in der CODESYS-Konfiguration die Schnittstelle ein, somit kann CODESYS darauf zugreifen.

Ab CODESYS V3.5 SP11 und später (V3.5.1x.x):

Ab SP11 müssen die beiden genannten Zeilen „auskommentiert“ werden. Dies wird mit einem Strickpunkt „;“ vor den Zeilen erreicht. Beachten Sie bitte, dass die Änderungen nicht in der CODESYSControl.cfg durchgeführt werden, sondern in der CODESYSControl\_User.cfg.

Verwenden Sie folgenden Befehl:

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

Fügen Sie am Ende der Datei diese Zeilen an:

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

Auf dem SD-Image für PiXtend V2 „CODESYS V2.X.X“ mit SP13-Runtime sind die Zeilen bereits eingetragen und auskommentiert. Es müssen, wenn Sie mit diesem Image arbeiten, keine Anpassung an der CODESYSControl\_User.cfg vorgenommen werden

Möchten Sie eine serielle USB-Schnittstelle verwenden, zum Beispiel ein weiterer RS232/RS485 Dongle, dann müssen die Strichpunkte entfernt und der Name des Linux-Gerätes (Device) angepasst werden:

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

Achten Sie darauf, dass hinter ttyUSB keine Zahl steht, sonst funktioniert der Zugriff aus CODESYS nicht.

Sie können das Gerät bzw. den Gerätenamen mit folgendem Befehl herausfinden:

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

#### 7.7.4.3 Raspberry Pi neu starten

Starten Sie den Raspberry Pi neu damit die Einstellungen übernommen werden.

```
sudo reboot
```

## 7.7.5. Terminal Programm

Soll die Verbindung mit einem PC aufgebaut werden, benötigen Sie zur Kommunikation ein Terminal-Programm.

### 7.7.5.1 Terminal Programm installieren

Ist noch kein Terminal Programm vorhanden, sollte im nächsten Schritt ein Programm installiert werden. In unserem Beispiel verwenden wir HTerm (<http://www.derhammer.info/terminal/>).

### 7.7.5.2 Terminal öffnen und einstellen

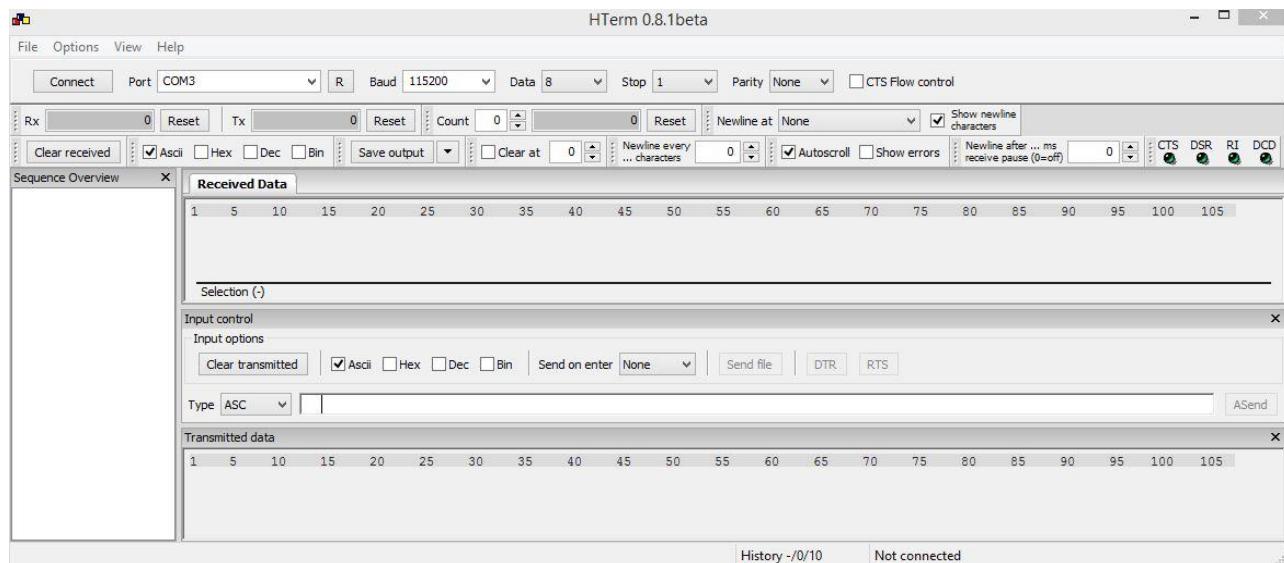


Abbildung 56: CODESYS - HTerm 0.8.1beta - RS232 Testprogramm

- Einstellungen:**
- Port: Im Geräte-Manager unter Anschlüsse (COM & LPT)
  - Baudrate: wie in CODESYS, z.B. 9600 Baud
  - Data: 8
  - Stop: 1
  - Parity: none

Sobald Sie mit der Taste „Connect“ die Verbindung hergestellt haben, können Daten empfangen und gesendet werden.

## 7.7.6. CODESYS

### 7.7.6.1 CODESYS Projekt „PiXtendSerialTest“ starten

Zunächst muss das Projekt PiXtendSerialTest geöffnet und das Programm auf den Raspberry Pi übertragen und gestartet werden. Das Testprojekt kann unter <https://www.pixtend.de/downloads> heruntergeladen werden.

#### NOTICE

Beachten Sie, dass dieses Projekt auf PiXtend V1.2/1.3 ausgelegt ist, auf dem PiXtend V2 -S- steht Ihnen keine RS485 Schnittstelle zur Verfügung. Ignorieren Sie daher den Schalter „Enable RS485“. Der RS232 Teil des Projekts ist vollständig mit PiXtend V2 -S- kompatibel. Das PiXtend V2 -L- verfügt über eine RS485 Schnittstelle und der RS485 Test kann durchgeführt werden.

### 7.7.6.2 Visualisierung



Abbildung 57: CODESYS - Visualisierung - Testprogramm - Serielle Kommunikation

**Kurzanleitung:**

1. Baudrate einstellen (muss mit Kommunikationspartner übereinstimmen)
2. „CONNECT“-Button betätigen  
Daten werden jetzt automatisch empfangen und angezeigt. Um eine Nachricht zu versenden:
3. Nachricht in das „Send“-Feld eingeben
4. „SEND“-Button betätigen

**Funktion der Buttons, Schalter und Auswahlfelder:**

„CONNECT / DISCONNECT“

Verbindung zum PC-Terminal, oder sonstigem Gerät, herstellen bzw. trennen.

„CLEAR ALL“

Die empfangenen Nachrichten, empfangene (RX) und gesendete (TX) Bytes und Fehler werden gelöscht. Außerdem wird das automatische Versenden von Nachrichten deaktiviert.

„SEND“

versendet die in das Eingabefeld eingegebene Nachricht.

„Baudrate“

Hier kann die gewünschte Baudrate (Bits/Sekunde) eingestellt werden.

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

Mit diesem Wechselschalter kann zwischen der RS232 und der RS485 Schnittstelle umgeschaltet werden. Dazu muss allerdings zuerst ein Disconnect durchgeführt werden, da beide Schnittstellen nicht gleichzeitig betrieben werden können.

„auto scroll“

Wenn diese Checkbox aktiv ist (Häkchen), scrollt die Anzeige der empfangenen Nachrichten automatisch zur aktuellsten Nachricht.

„enable“ Auto send

aktiviert das automatische Senden der eingegebenen Nachricht im eingestellten Intervall.

**Ein-/Ausgabefelder:**

„Receive“

Hier werden die empfangenen Daten mit Zeitstempel tabellarisch angezeigt.

„Send“

Hier werden die Daten, die gesendet werden sollen, eingegeben.

„Auto send“

Daten, die automatisch gesendet werden sollen.

„interval“

Ein Intervall, in dem die Daten automatisch gesendet werden sollen, Minimum ist 1 Sekunde.

**Statusanzeigen:**

„Error Code“

Tritt während der Bedienung ein Fehler beim Öffnen, Schließen, Lesen oder Schreiben auf, wird dieser über die Fehler LEDs angezeigt. Zusätzlich wird im Feld „Error Code“ der entsprechende Fehler Code angezeigt.

„Status“

Hier wird der Status angezeigt, in dem sich das Programm im Moment befindet.

## 7.8. CAN-Kommunikation

### 7.8.1. Einführung

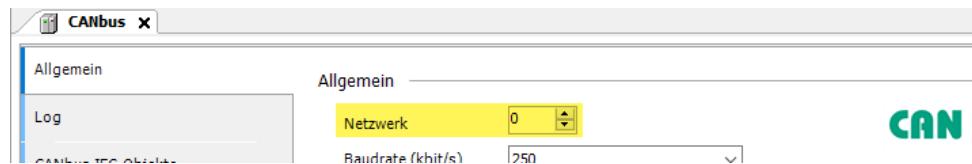
Der robuste und zuverlässige Bus wird in vielen Bereichen, zum Beispiel in der industriellen Automation, eingesetzt. In der Automatisierungstechnik wird oftmals das CANopen Protokoll (OSI-Schicht 7 – Anwendungsschicht) verwendet, welches von CODESYS unterstützt und für den Raspberry Pi verfügbar ist.

In diesem Kapitel beschreiben wir die notwendigen Schritte um mit PiXtend V2 -L- eine CANOpen-Kommunikation unter CODESYS einzurichten. Hierfür benötigen Sie eine SD-Karte mit dem PiXtend V2 SD-Karten Image „CODESYS Control“, anschließend wird die CANopen-Konfiguration in CODESYS erklärt.

#### NOTICE

Die CAN-Bus Schnittstelle steht nur auf dem PiXtend V2 -L- zur Verfügung!

Unter CODESYS muss für das CANBus Geräte immer die Netzwerknummer „0“ eingestellt werden um den CAN-Bus des PiXtend V2 -L- nutzen zu können.



Viele weitere Informationen, Tipps und Tricks finden Sie in unserem Support-Forum unter:  
<https://www.pixtend.de/forum/>

Sollten Sie darüberhinaus Fragen haben, senden Sie uns diese bitte per E-Mail ([support@pixtend.de](mailto:support@pixtend.de)), Sie erhalten schnellst möglich eine Antwort und weitere Informationen.

Die jeweils neuesten Versionen aller Dokumente und Software-Komponenten finden Sie im Download-Bereich unserer Homepage: <https://www.pixtend.de/pixtend/downloads/>

### 7.8.1.1 Voraussetzungen

Zur Aktivierung der CAN-Bus Funktionalität sind Grundlagen im Umgang mit dem Raspbian Betriebssystem (Linux) sowie mit der Bedienung von CODESYS notwendig.

Um die CAN-Kommunikation unter Linux zu testen, muss das PiXtend V2 -L- an einen CAN-Bus mit mindestens einem aktiven CAN-Gerät angeschlossen sein. Es kann mit einem oder mehreren anderen PiXtends (PiXtend V1.3 sowie PiXtend V2 -L-) verbunden werden.

Für die CODESYS CANopen Master/Slave Kommunikation werden in diesem Kapitel zwei PiXtend V2 -L- Boards verwendet.

### 7.8.1.2 Einschränkungen

Der PiXtend V2 -L- CAN Controller kann nicht gleichzeitig mit der PiXtend V2 -L- DAC verwendet werden, beide Chips „teilen“ sich einen SPI Chip-Select Kanal.

## 7.8.2. Hardware-Verbindung zum CAN-Bus

Bitte entnehmen Sie die korrekte Anschlussbelegung des PiXtend V2 -L- CAN-Ports dem PiXtend V2 -L- Hardware Handbuch.

#### ▲ CAUTION

Um die CAN-Bus Hardware auf PiXtend V2 -L- verwenden zu können, muss der Jumper „CAN“ / „AO“ auf Stellung „CAN“ umgesteckt werden, der CAN-Bus lässt sich sonst nicht verwenden. Der DAC wird durch diese Aktion deaktiviert und kann nicht länger genutzt werden.

Beachten Sie bitte den Jumper für die „Terminierung“ des CAN-Buses.

### 7.8.3. Vorbereitung und Test unter Linux

Um den CAN Controller in CODESYS zu verwenden, sind einige Anpassungen am Raspbian (Linux) Betriebssystem notwendig.

Haben Sie bereits eigene Anpassungen an Ihrem Image vorgenommen, so empfehlen wir für erste CAN-Tests eine neue SD-Karte zu erstellen. Beispielsweise unser „CODESYS Control“ SD-Karten Image, das wir in der aktuellsten Version in unserem Download-Bereich zur Verfügung stellen.

#### **NOTICE**

Alle hier beschriebenen Schritte beziehen sich auf das Raspbian Image „Stretch“ und werden als User „pi“ durchgeführt. Die Unterstützung des auf dem PiXtend V2 -L- verwendeten CAN-Controllers MCP2515 ist erst ab der Kernel Version 4.0.8 gewährleistet. Verwenden Sie ein „Wheezy“ Image, muss zuerst auf eine Kernel Version 4.0.8 (oder neuer) upgedated werden.

#### **CAUTION**

Verwenden Sie bereits eine SD-Karte mit „CODESYS Control“ Image, dann stoppen Sie die CODESYS Laufzeitkomponente, um den CAN-Bus in Betrieb zu nehmen. Verwenden Sie folgenden Befehl: `sudo service codesyscontrol stop`

Nach einem Neustart des Raspberry Pi steht Ihnen CODESYS wie gewohnt zur Verfügung.

#### Vorbereitung des SD-Karten-Image

Laden Sie das Raspbian „Stretch“ Image von der offiziellen Download-Seite herunter:

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

Übertragen Sie das heruntergeladene Image mit einem Tool ihrer Wahl auf eine SD-Karte (empfohlen 8 GB). Unter Windows kann für diese Aufgabe das kostenlose Programm Win32DiskImager verwendet werden.

Booten Sie das neue Image und führen Sie die üblichen Anpassungen mittels `raspi-config` durch:

- Erweiterung des Filesystems
- Einstellung Ihrer Zeitzone
- Ländereinstellungen
- Spracheinstellungen
- Keyboard Layout
- SSH Zugriff aktivieren

Starten Sie den Raspberry Pi mit diesem Befehl neu

`sudo reboot`

und loggen Sie sich anschließend über SSH in der Kommandozeile als User pi ein (beispielsweise mit Putty) oder arbeiten Sie direkt mit Bildschirm und Tastatur.

#### Tipp:

Verwenden Sie unser PiXtend Image „CODESYS“, können Sie die Abschnitte 7.8.3.3 und 7.8.3.4 überspringen, da wir bereits alles für Sie vorbereitet haben. Lediglich eine kleine Anpassung ist notwendig, wie in 7.8.3.2 beschrieben.

### 7.8.3.1 Konfiguration des Device Tree Overlay

Um den MCP2515 CAN-Chip des PiXtend-Boards von Kernel zu bedienen, sind folgende Anpassungen an der Datei /boot/config.txt notwendig. Öffnen Sie den Editor mit nachstehendem Befehl

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

und fügen Sie folgende Einträge am Ende hinzu:

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

Speichern Sie die Änderungen mit Strg+O und verlassen Sie den Editor mit Strg+X.

Starten Sie den Raspberry Pi neu:

```
sudo reboot
```

### 7.8.3.2 Blacklist bearbeiten

Damit das Kernel Modul „mcp251x“ beim Booten nicht automatisch geladen wird, fügen Sie es der Blacklist hinzu.

Öffnen Sie dazu die Datei /etc/modprobe.d/raspi-blacklist.conf

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

und fügen Sie folgende Zeile ein:

```
blacklist mcp251x
```

Die Datei ist zunächst komplett leer, kann aber dennoch verwendet werden.

Speichern Sie die Änderungen mit Strg+O und verlassen Sie den Editor mit Strg+X.

Starten Sie das Raspberry Pi neu:

```
sudo reboot
```

### 7.8.3.3 CAN Script anlegen

Um den PiXtend V2 -L- CAN-Controller zu aktivieren müssen folgende Aufgaben durchgeführt werden:

- GPIO 24 als Ausgang konfigurieren und auf TRUE setzen (SPI Enable)
- GPIO 27 als Ausgang konfigurieren und auf TRUE setzen (SPI CS1 wird dadurch an den CAN Controller weitergereicht, der DAC wird deaktiviert)
- Das Kernel Modul „mcp251x“ laden
- Das Interface can0 konfigurieren und aktivieren

Da hierzu immer die gleichen Schritte notwendig sind, legen wir ein neues Skript im Home Verzeichnis des Users pi an:

```
nano activateCAN.sh
```

Fügen Sie folgende Zeilen ein:

```
#!/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
```

Speichern Sie die Änderungen mit Strg-O und verlassen Sie den Editor mit Strg-X.

Führen Sie das Skript aus, indem Sie folgenden Befehl eingeben:

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

### 7.8.3.4 CAN aktivieren

Wenn Sie nun das Skript mittels

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

aufrufen, wird der CAN Controller aktiviert.

Mit einem Aufruf von

```
ifconfig
```

lässt sich kontrolliert, ob ein Eintrag für ein can0 Interface vorhanden ist.

### 7.8.3.5 CAN-Utilities installieren und verwenden

Installieren Sie die „can-utils“, eine Sammlung nützlicher User-Space Programme, um mit SocketCAN unter Linux direkt auf die CAN-Schnittstelle zugreifen zu können:

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

Mit dem Befehl

```
cansend can0 123#DEADBEEF
```

wird eine CAN Message mit ID 123 und dem Inhalt 0xDEADBEEF verschickt.

Mit dem Befehl

```
candump can0
```

kann der gesamte Datenverkehr auf can0 mitgelauscht werden:

```
pi@raspberrypi ~ $ candump can0
can0 123 [1] DE
can0 123 [2] DE AD
can0 123 [3] DE AD BE
can0 123 [4] DE AD BE EF
```

Abbildung 58: CODESYS - CAN-Bus Test unter Linux

Mit dem Program „cangen“ können laufend CAN-Messages zu Testzwecken generiert werden.

Weitere nützliche Funktionen und Parameter der can-utils entnehmen Sie bitte den jeweiligen Hilfe Seiten.

Mit diesen Tools können Sie bereits an einer CAN Bus Kommunikation unter Linux teilnehmen.

Bitte fahren Sie mit den weiteren Schritten (Vorbereitung für CODESYS) erst dann fort, wenn Sie mit den „can-utils“ erfolgreich Daten empfangen und versendet haben.

## 7.8.4. Vorbereitungen für CODESYS

Für die Schritte in diesem Kapitel benötigen Sie eine SD-Karte mit installierter CODESYS Runtime. Unabhängig davon ob Sie mit dem CAN-Bus beginnen oder bereits ein eigenes Image und System aufgebaut haben, empfehlen wir Ihnen mit unserem SD-Karten Image „CODESYS Control“, zu beginnen.

### 7.8.4.1 Start Skript anlegen

Um das Kernel Modul für den PiXtend V2 -L- CAN-Controller korrekt zu laden, benötigen wir ein Start Skript das folgende Aktionen in exakt dieser Reihenfolge ausführt:

- CODESYS Control stoppen
- CAN aktivieren mit Hilfe des vorher erstellen Skripts „activateCAN.sh“  
(Siehe Punkt 7.8.3.4)
- CODESYS Control starten

Öffnen Sie eine Konsole auf dem Raspberry Pi (via SSH mit dem Tool Putty) und legen Sie die Datei „startPLCCAN.sh“ im home-Verzeichnis an:

```
nano startPLCCAN.sh
```

Fügen Sie folgende Zeilen ein:

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

Speichern Sie die Änderungen mit Strg-O und verlassen Sie den Editor mit Strg-X.

Führen Sie das Skript aus, indem Sie folgenden Befehl eingeben:

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

### 7.8.4.2 Baudrate Skript anlegen

Legen Sie folgende Baudrate Skript an:

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

und fügen Sie folgende Zeilen ein:

```
#!/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
```

Speichern Sie die Änderungen mit Ctrl-O und verlassen Sie den Editor mit Ctrl-X.

Führen Sie das Skript aus, indem Sie folgenden Befehl eingeben:

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

### 7.8.4.3 CODESYS Control mit PiXtend V2 -L- CAN-Unterstützung starten

Starten Sie den Raspberry Pi neu:

```
sudo reboot
```

Nach dem Bootvorgang wird die CODESYS Control Runtime automatisch gestartet.

Um CODESYS mit CAN Unterstützung zu verwenden, muss nach jedem Start zusätzlich das Skript „startPLCCAN.sh“ aufgerufen werden:

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

Durch den Aufruf des Skripts wird zunächst die CODESYS-Runtime beendet, die GPIOs entsprechend konfiguriert, das mcp251x Kernel Modul geladen, die can0 Schnittstelle konfiguriert, und CODESYS Control im Anschluß wieder gestartet.

Möchten Sie die CAN Unterstützung dauerhaft aktivieren, lassen Sie das Start Skript bei jedem Neustart automatisch ausführen, in dem Sie die Datei

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

bearbeiten und folgende Zeilen vor „exit 0“ hinzufügen:

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

Testen Sie die Konfiguration indem Sie das Raspberry Pi neu starten:

```
sudo reboot
```

Mit einem Aufruf von „lsmod“ können Sie prüfen ob das „mcp251x“ Modul nach dem Neustart geladen wurde. Mit einem Aufruf von „top“ ob das CODESYS Control arbeitet.

## 7.8.5. CODESYS Projekt mit CANopen

Waren die Vorbereitungen und Tests unter Linux erfolgreich, kann ein CANopen Test Projekt in CODESYS erstellt werden. Wir verwenden zwei PiXtend V2 -L- Geräte und lassen diese miteinander über CANopen kommunizieren.

Nun erstellen wir ein leeres Projekt zu dem zwei PiXtend V2 -L- Geräte hinzugefügt werden. Ein Gerät ist der CANopen-Master und das andere der CANopen-Slave. Sie können an Stelle eines zweiten PiXtend V2 -L- ein beliebiges anderes CAN-Bus Gerät als Slave verwenden. Zur Konfiguration ziehen Sie einfach das Handbuch Ihres CAN-Slaves zu rate.

In unserem Beispiel verwenden wir zunächst nur 1 Byte, das vom Master an den Slave übertragen wird. Das Beispiel lässt sich anschließend beliebig erweitern.

Erzeugen Sie zunächst ein leeres Projekt (Datei -> Neues Projekt -> Leeres Projekt) und nennen Sie es „PiXtendCAN\_MasterSlave“. Legen Sie den Speicherort Ihres Projektes fest, sofern Sie mit der Vorgabe nicht einverstanden sind.

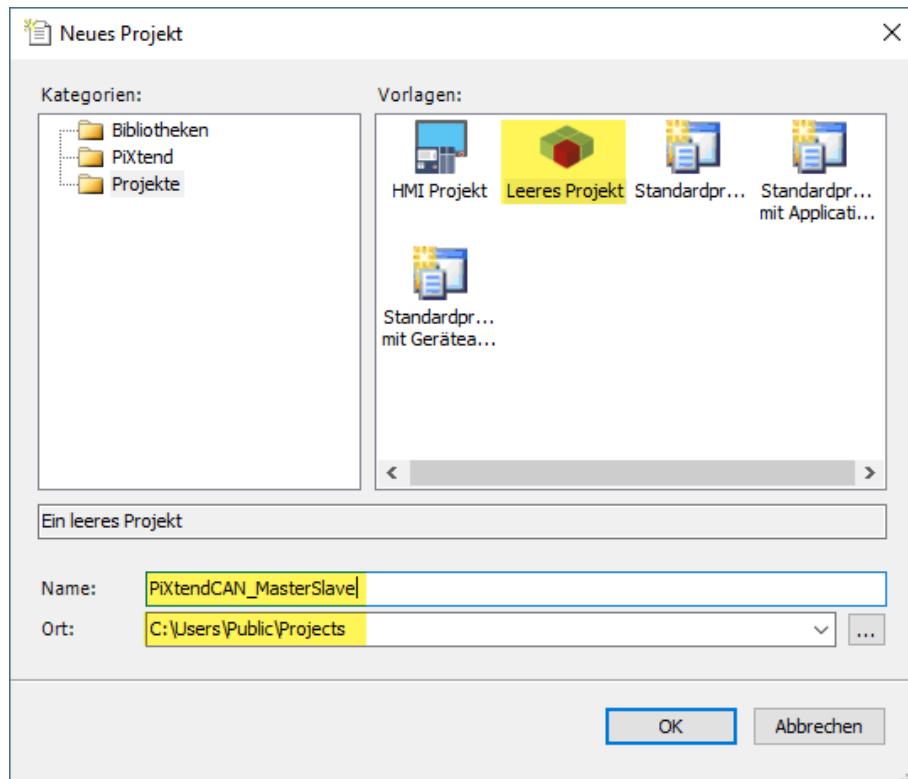


Abbildung 59: CODESYS - CAN-Bus - Neues Projekt erstellen

### 7.8.5.1 PiXtend V2 -L- als CAN-Slave

Fügen Sie dem Projekt ein neues Gerät hinzu, indem Sie im Projektbaum auf das Projekt rechts-klicken und „Gerät hinzufügen“ auswählen.

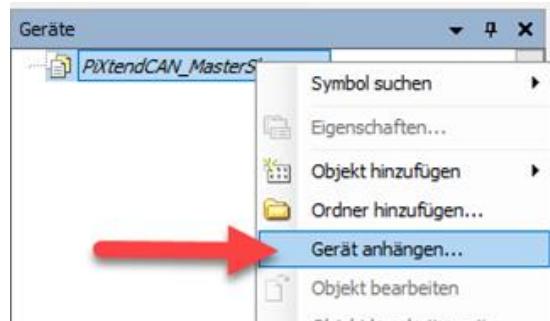


Abbildung 60: CODESYS - CAN-Bus Slave - SPS Gerät anhängen

Stellen Sie zuerst den Hersteller auf „3S – Smart Software Solutions GmbH“ um, wählen Sie die „CODESYS Control for Raspberry Pi“ als Gerät aus, geben Sie ihm den Namen „PiXtendCAN\_Slave“. Klicken Sie zum Abschluss auf „Gerät anhängen“ und „Schließen“.

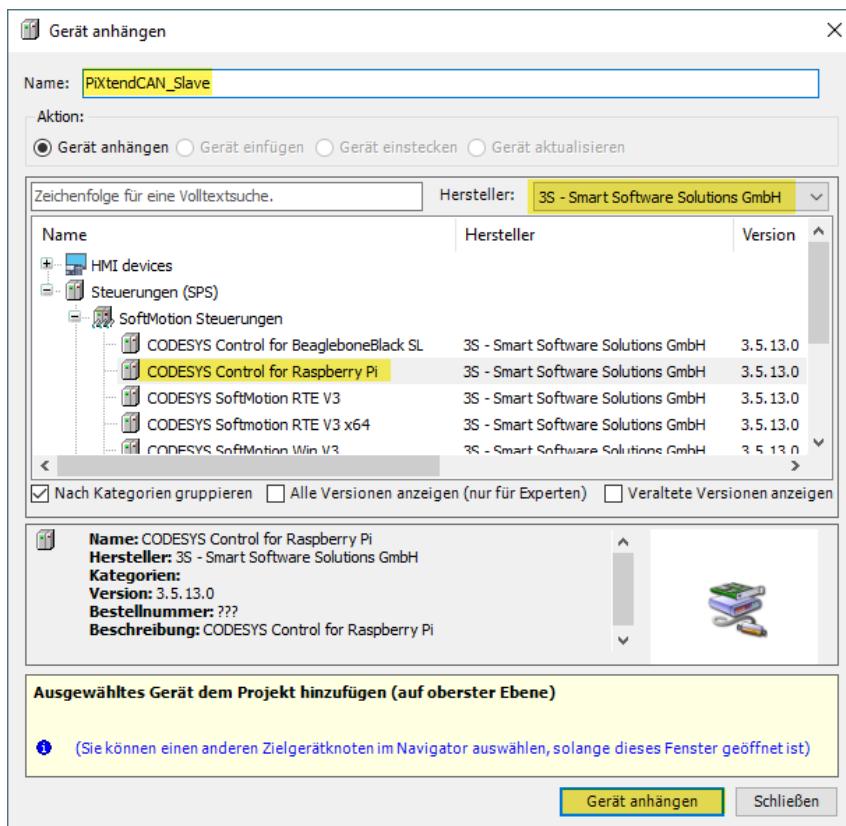


Abbildung 61: CODESYS - CAN-Bus Slave- RPi anhängen

Führen Sie im Projektbaum einen Rechtsklick auf das soeben hinzugefügte „PiXtendCAN\_Slave“ Gerät aus:

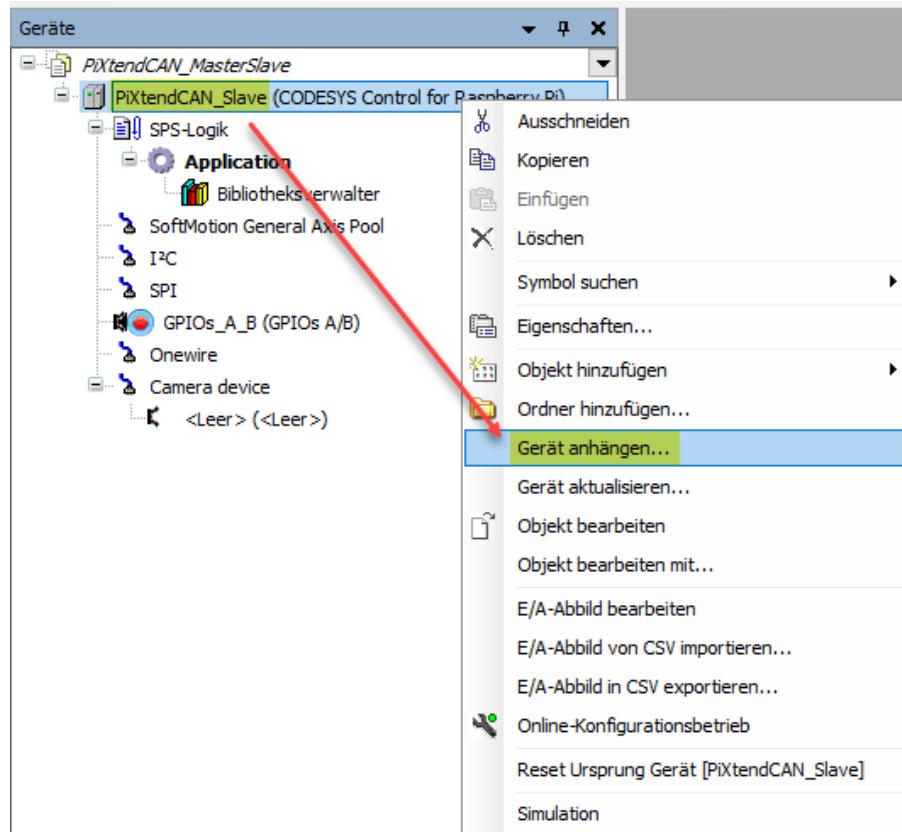


Abbildung 62: CODESYS - CAN-Bus Slave - Gerät an SPS anhängen

Im Dialog wählen Sie bitte „CANbus“ aus und hängen das Gerät an. Die Auswahl erfolgt schneller, wenn Sie unter „Hersteller“ die Firma „3S – Smart Software Solutions GmbH“ auswählen.

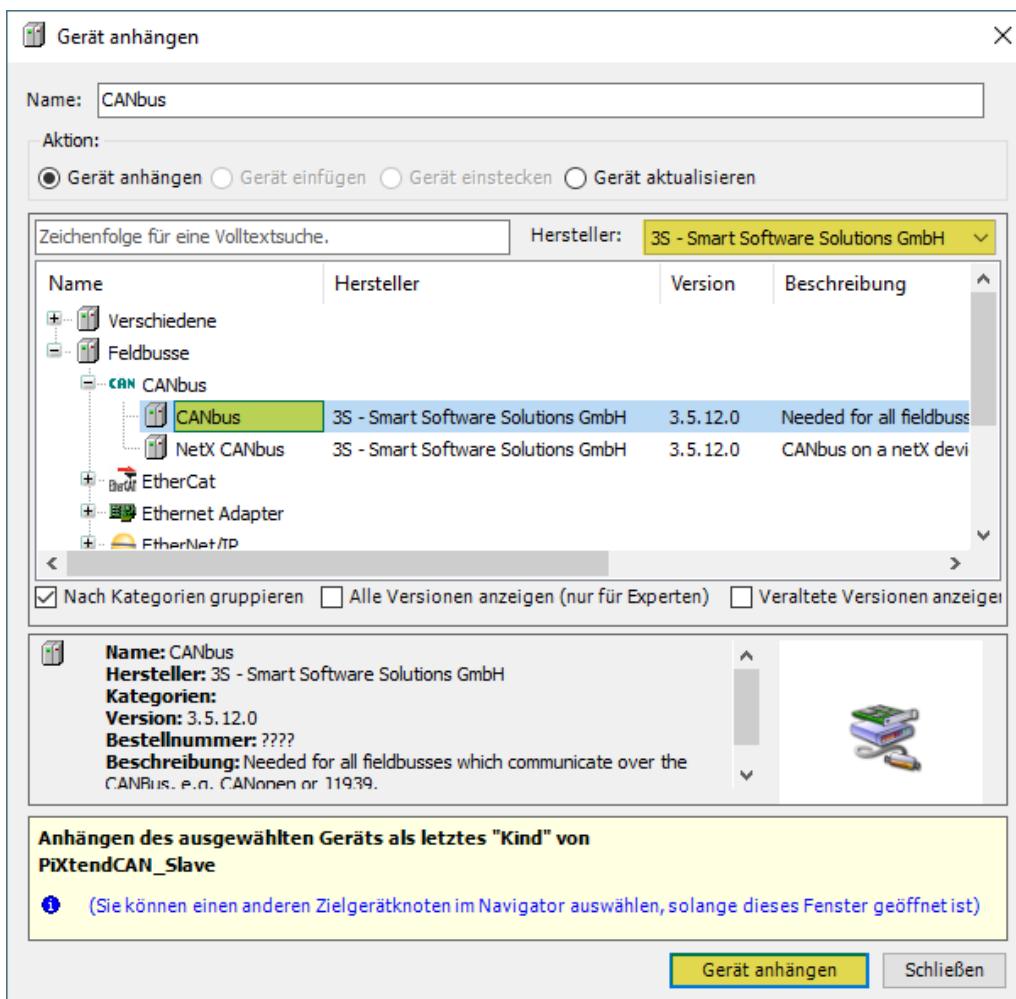


Abbildung 63: CODESYS - CAN-Bus Slave - CAN-Bus-Gerät anhängen

Lassen Sie das Fenster geöffnet, wir benötigen es noch einmal auf der nächsten Seite.

Führen Sie einen Linksklick auf das soeben hinzugefügte „CANbus“ Gerät im Gerätebaum aus, während das Fenster „Geräte anhängen“ noch offen ist. Nach dem Klick ändert sich der Inhalt des „Geräte anhängen“ Fensters und Sie können ein „CANopen Device“ unterhalb des CANbus Geräts anhängen. Die Auswahl wird einfacher, wenn Sie unter „Hersteller“ die Firma „3S – Smart Software Solutions GmbH“ auswählen

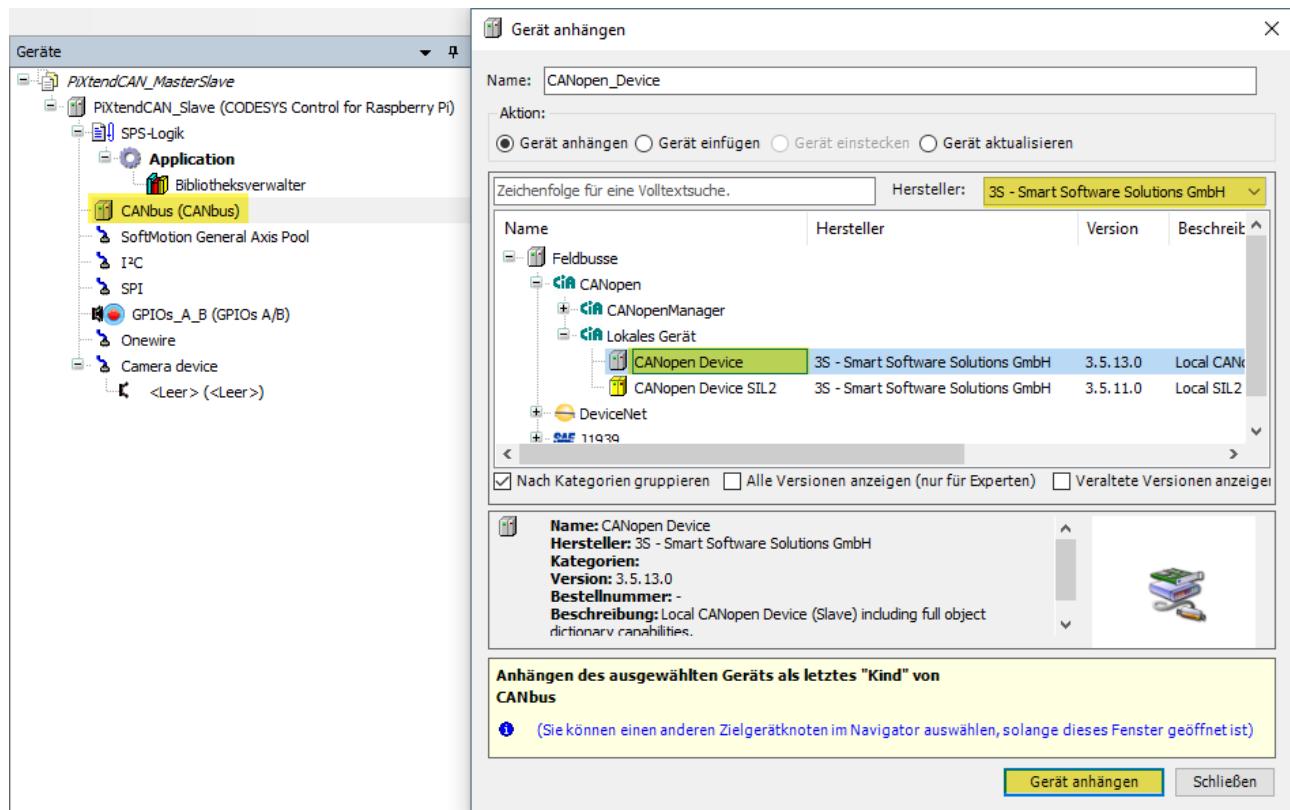


Abbildung 64: CODESYS – CAN-Bus Slave – CANopen Device Gerät anhängen

Schließen Sie nach dem Anhängen das Fenster „Geräte anhängen“.

Doppelklicken Sie nun im Gerätebaum auf das hinzugefügte CANopen-Gerät und klicken Sie anschließend auf die Schaltfläche „E/A-Bereich bearbeiten“

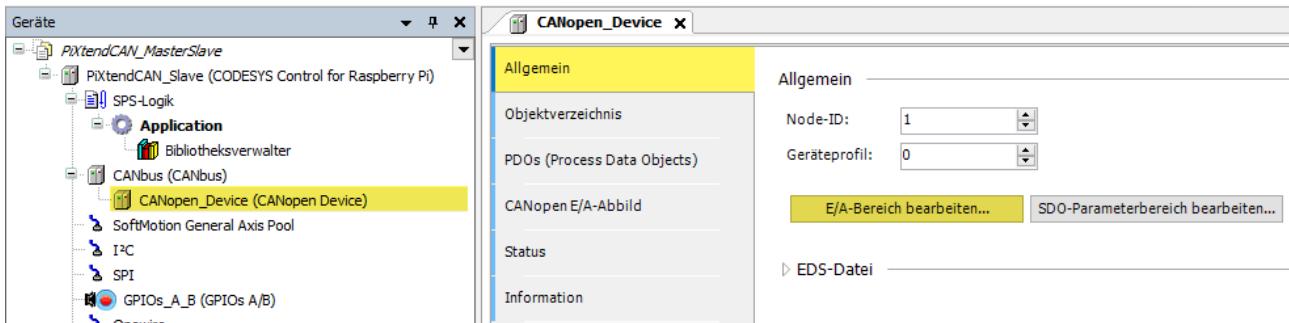


Abbildung 65: CODESYS - CAN-Bus Slave - E/A-Bereich konfigurieren

Fügen Sie einen neuen Bereich hinzu und wählen Sie „Empfangen“. Die restlichen Felder können unverändert bleiben da der Master zunächst nur ein einzelnes USINT (BYTE) an den Slave schicken soll.

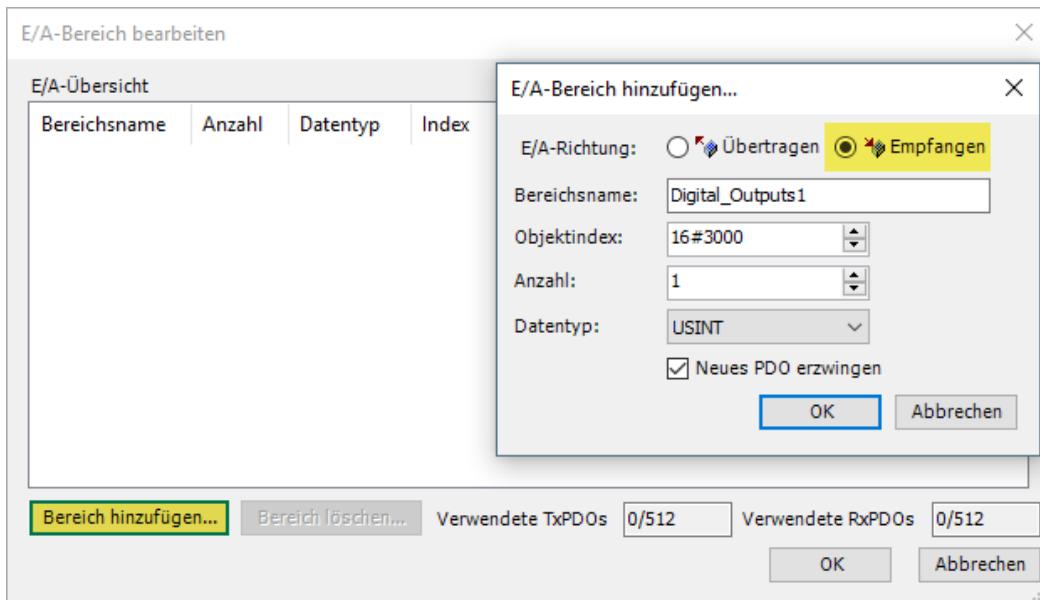


Abbildung 66: CODESYS - CAN-Bus Slave - PDO festlegen

Bestätigen Sie Ihre Eingaben und schließen Sie den Dialog.

Klappen Sie nun den Bereich „EDS-Datei“ auf und installieren Sie das soeben erzeugte Gerät im Geräterepository über die entsprechende Schaltfläche. Um das eigene Gerät besser zu finden ändern Sie Herstellername und/oder den Produktnamen ab.

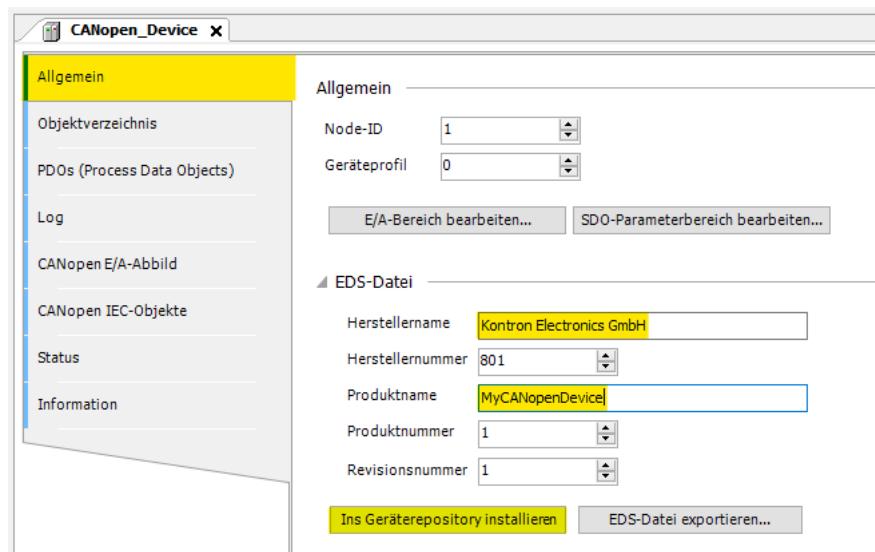


Abbildung 67: CODESYS - CAN-Bus Slave - Eigenes CAN-Gerät installieren

Erhalten Sie einen Hinweis, dass es bereits ein Gerät mit diesem Namen oder dieser Herstellernummer gibt, dann können Sie die Herstellernummer ändern. Sind Sie sicher, dass es sich bei dem anderen Gerät nur um ein Versuchsgerät handelt, überschreiben Sie es.

Alle Informationen über unser selbst erstelltes CANopen Gerät sind nun in der CODESYS Gerätedatenbank gespeichert und liegen zur Verwendung bereit, zum Beispiel durch den CANopen Master.

Fügen Sie der Applikation eine Taskkonfiguration hinzu sowie eine POU namens „POU\_Main“.

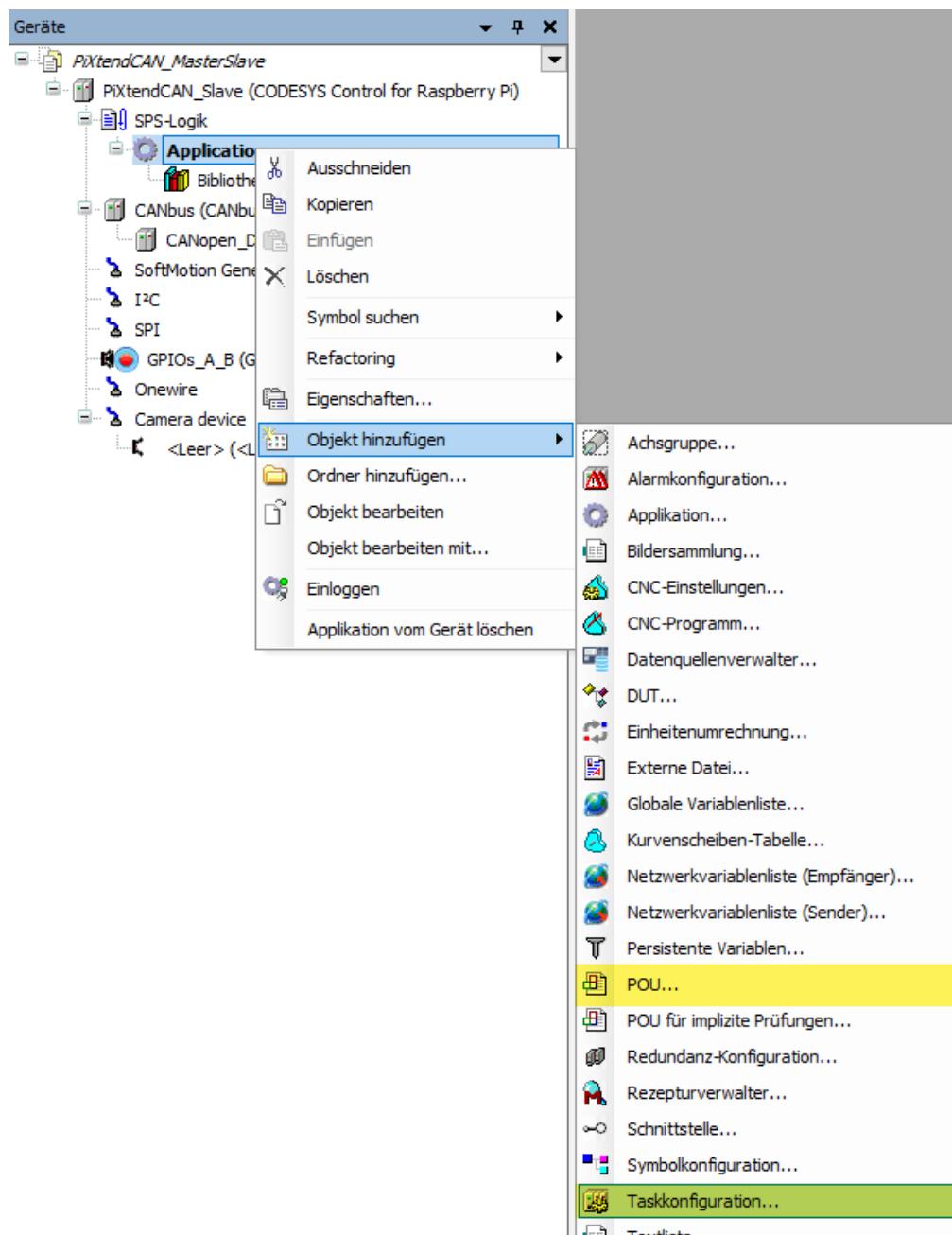


Abbildung 68: CODESYS - CAN-Bus Slave - POU und Taskkonfiguration

Öffnen Sie die Taskkonfiguration und fügen Sie dem Task einen Aufruf des Hauptprogrammes „POU\_Main“ hinzu:

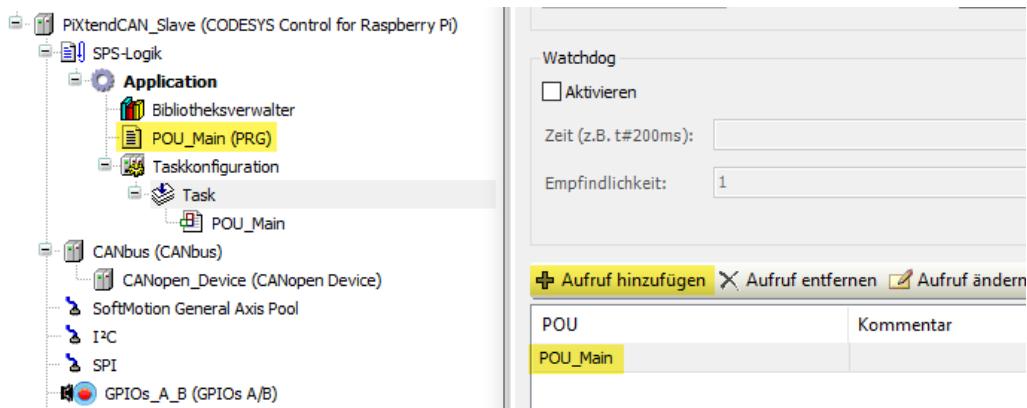


Abbildung 69: CODESYS - CAN-Bus Slave - Taskkonfiguration

Wechseln Sie zum „CANopen\_Device“ und ändern Sie dort die Einstellungen für die Aktualisierung der Variablen unter dem Reiter „CANopen E/A-Abbild“:

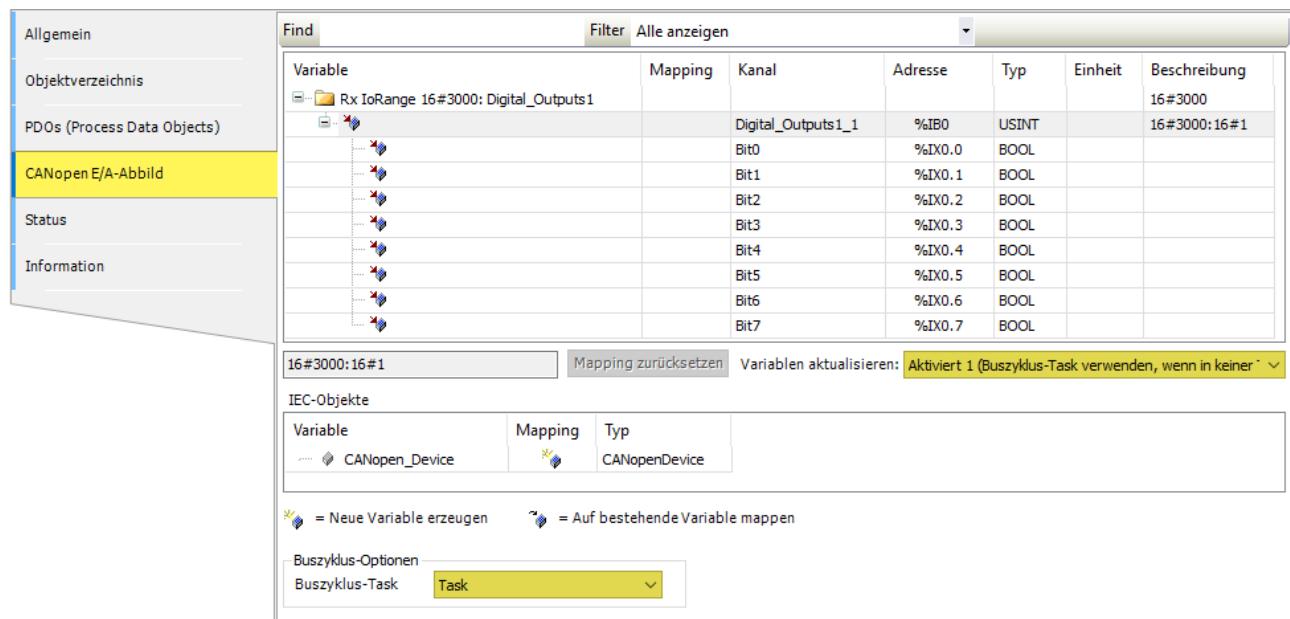


Abbildung 70: CODESYS - CAN-Bus Slave - CANopen E/A-Abbild

Wählen Sie die Option „Aktiviert 1 – Buszyklustask verwenden, wenn in keinem Task verwendet“ sowie „Task“ als Buszyklus-Task.

Abschließend sollte die Baudrate für den CAN Bus auf „125 kBit/s“ reduziert werden:

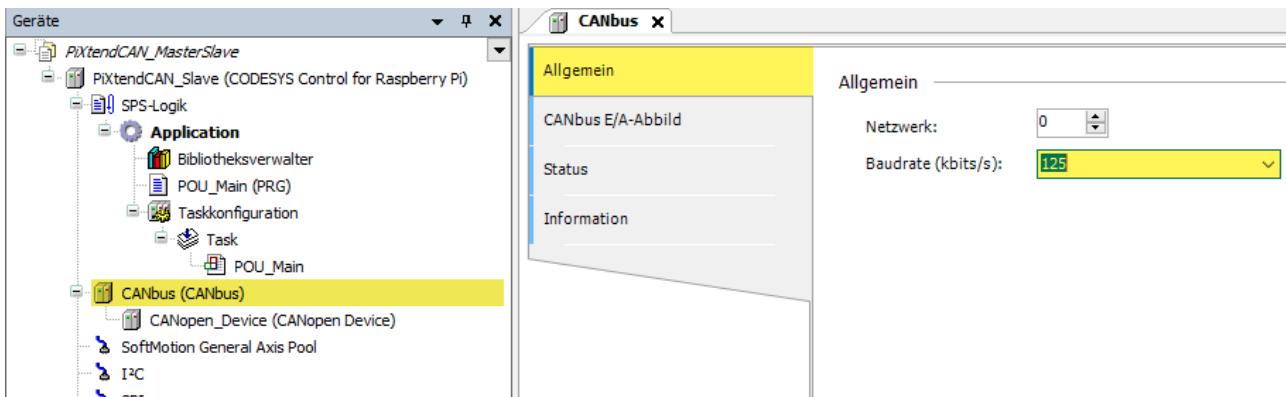


Abbildung 71: CODESYS - CAN-Bus Slave - Baudrate

Jetzt ist die Konfiguration des Slaves abgeschlossen.

### 7.8.5.2 PiXtend V2 -L- als CANopen-Master

Wir fügen dem Projekt ein weiteres PiXtend V2 -L- Gerät hinzu, das die Rolle des CANopen-Masters übernimmt.

Führen Sie einen Rechtsklick auf das Projekt „PiXtendCAN\_MasterSlave“ im Projektbaum aus und fügen Sie ein weiteres „CODESYS Control for Raspberry Pi“ Gerät mit dem Namen „PiXtendCAN\_Master“ hinzu:

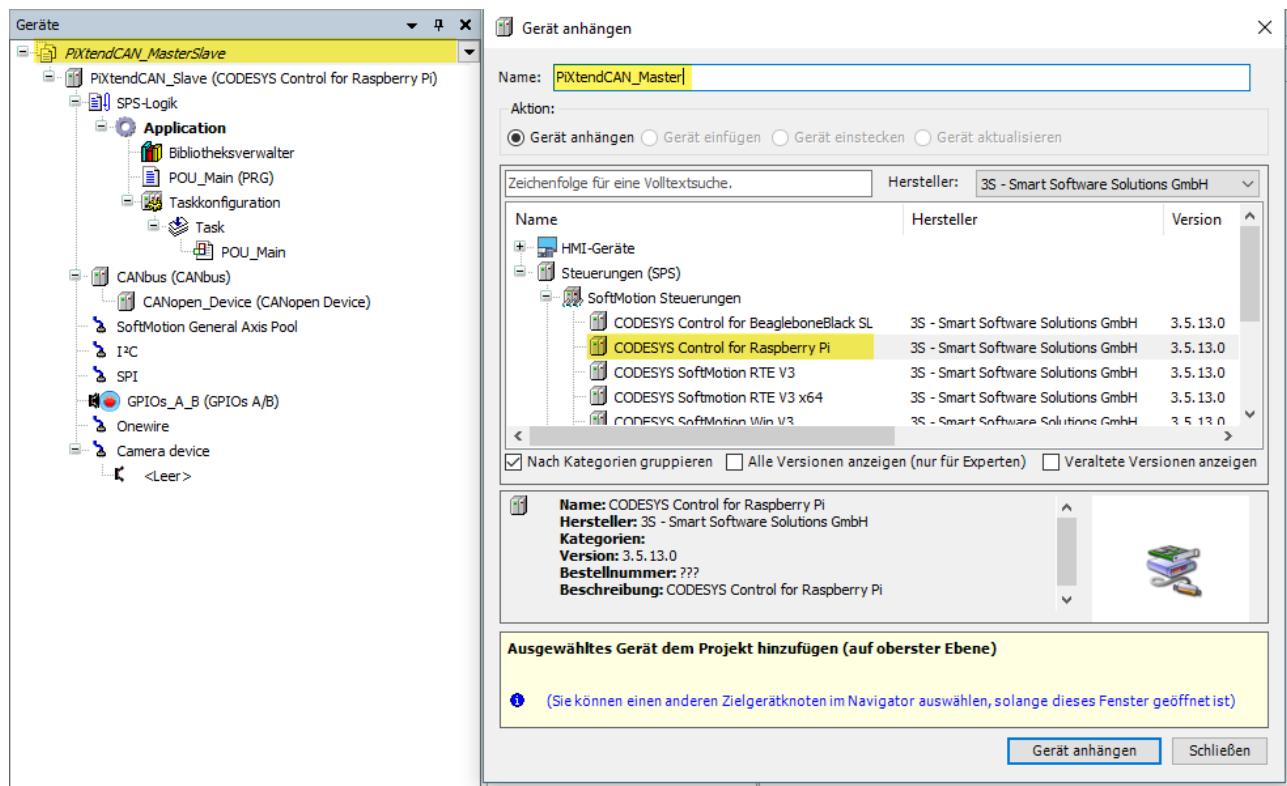


Abbildung 72: CODESYS - CAN-Bus Master - Raspberry Pi anhängen

Hängen Sie an den „PiXtendCAN\_Master“ ebenfalls ein „CANbus“ Gerät an:

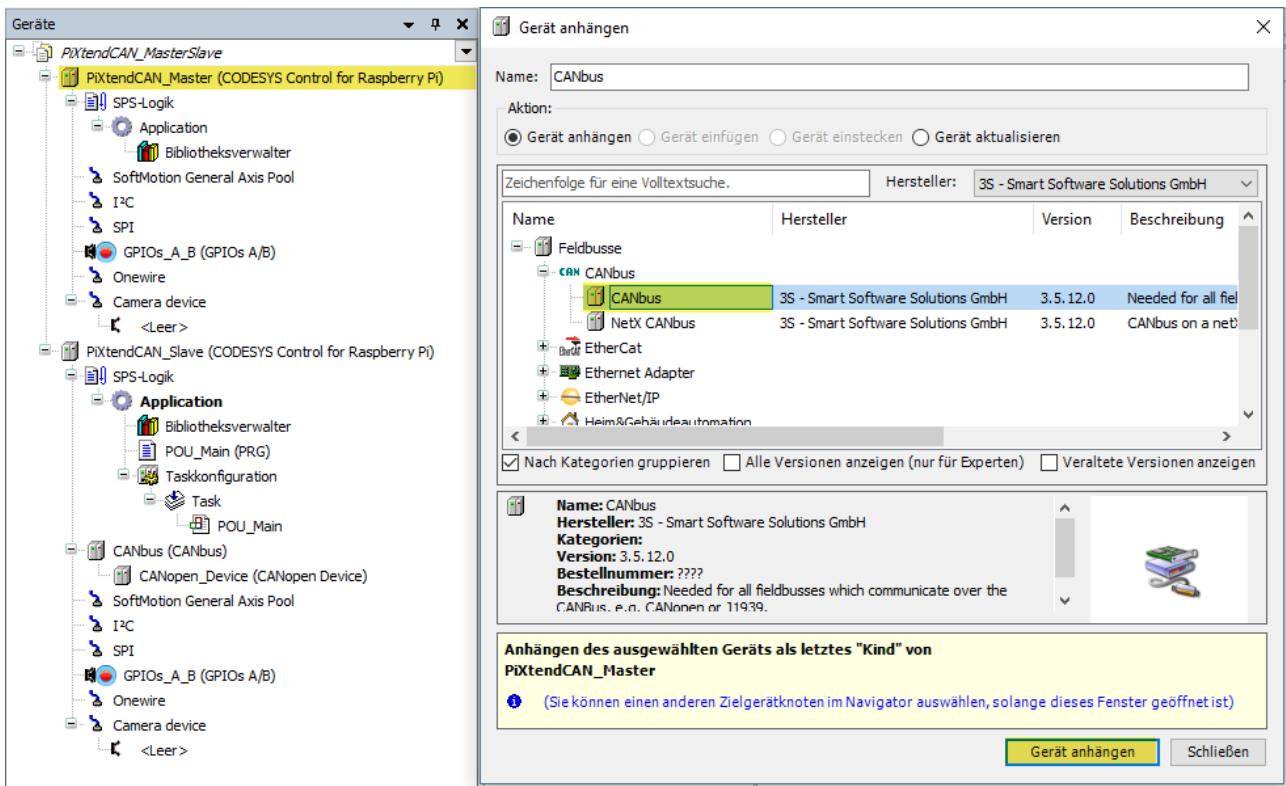


Abbildung 73: CODESYS - CAN-Bus Master - CANbus anhängen

Reduzieren bzw. setzen Sie die Baudrate des CANbuses auf „125 kBit/s“, oder auf die gleiche Einstellung, die beim Slave gewählt wurde.

Machen Sie nun einen Rechtsklick auf den soeben hinzugefügten „CANbus“ Eintrag und hängen Sie diesem einen „CANopen\_Manager“ an:

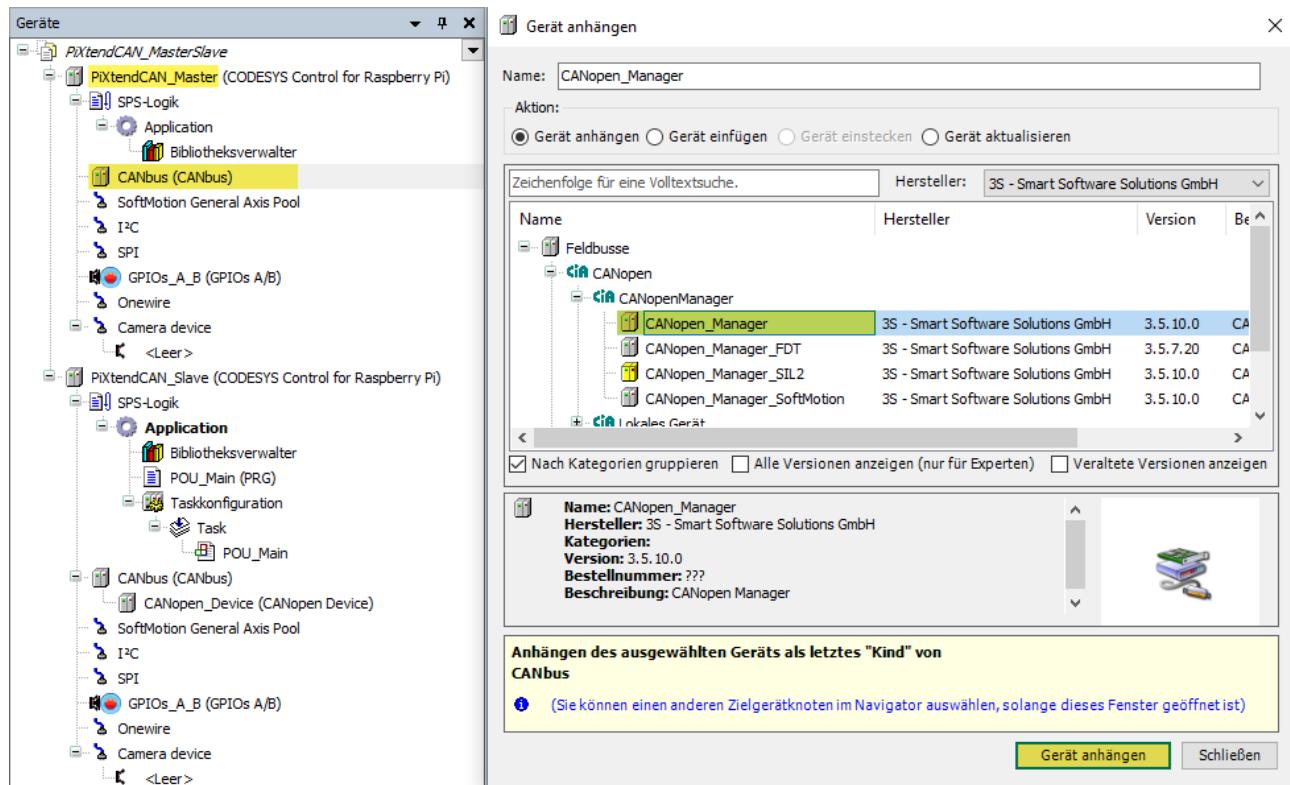


Abbildung 74: CODESYS - CAN-Bus Master - CANopen Manager anhängen

Lassen Sie das Fenster geöffnet, wir benötigen es gleich noch einmal auf der nächsten Seite.

Wählen Sie nun den soeben hinzugefügten „CANopen\_Manager“ aus und hängen Sie das von Ihnen erstellte „MyCANopenDevice“ an. Haben Sie bei der Erstellung der CANopen Slaves Geräts einen eigenen Herstellernamen eingetragen, so können Sie unter „Hersteller“ diesen Namen auswählen

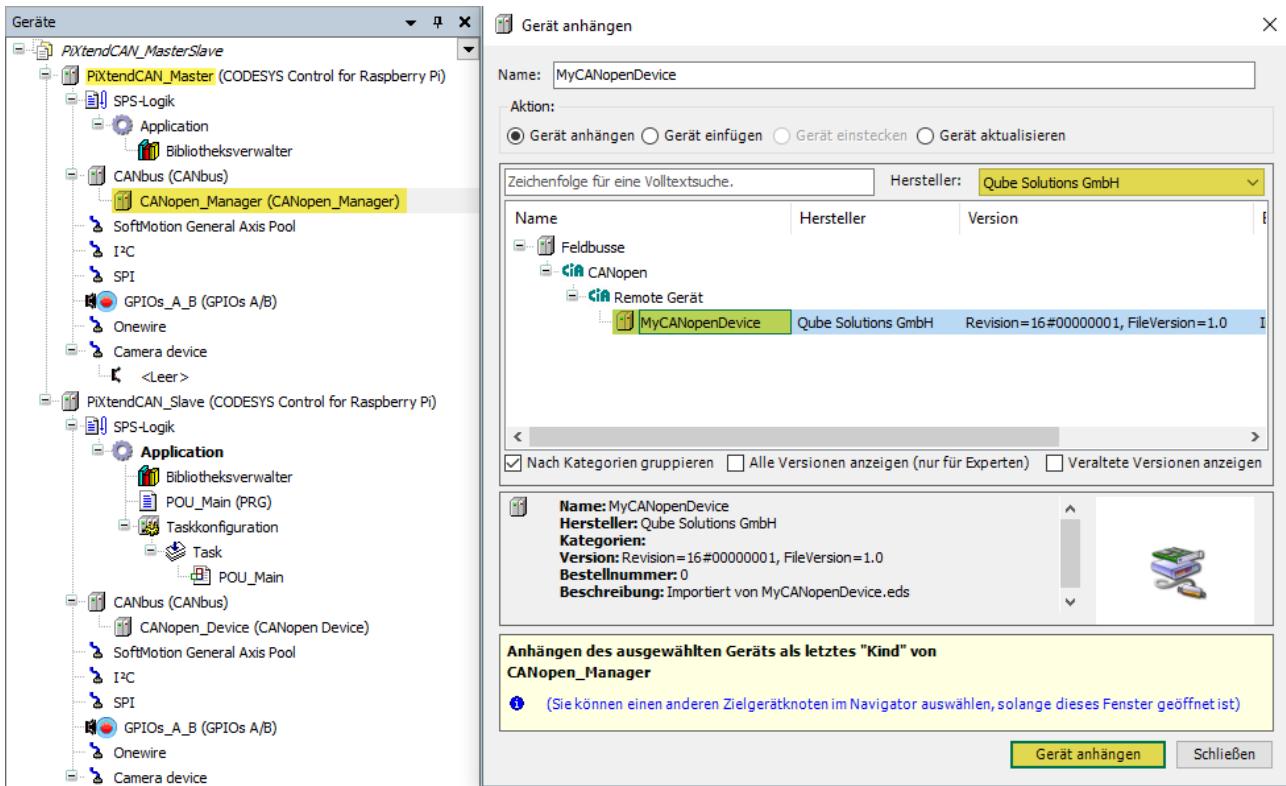


Abbildung 75: CODESYS - CAN-Bus Master - CAN-Bus Slave anhängen

Fügen Sie dem „PiXtendCAN\_Master“ Gerät eine „Taskkonfiguration“ und eine „POU\_Main“ als Hauptprogramm für die Applikation hinzu. Erstellen Sie beim Eintrag „Task“ in der „Taskkonfiguration“ einen Aufruf für den Baustein „POU\_Main“, ähnlich wie Sie es bei der Erstellung des CANopen Slave Geräts getan haben.

Öffnen Sie nun die E/A Konfiguration (CANopen E/A-Abbild) des Gerätes „MyCANopenDevice“ und wählen Sie den Eintrag „Aktiviert 1 – Buszyklustask verwenden, wenn in keinem Task verwendet“

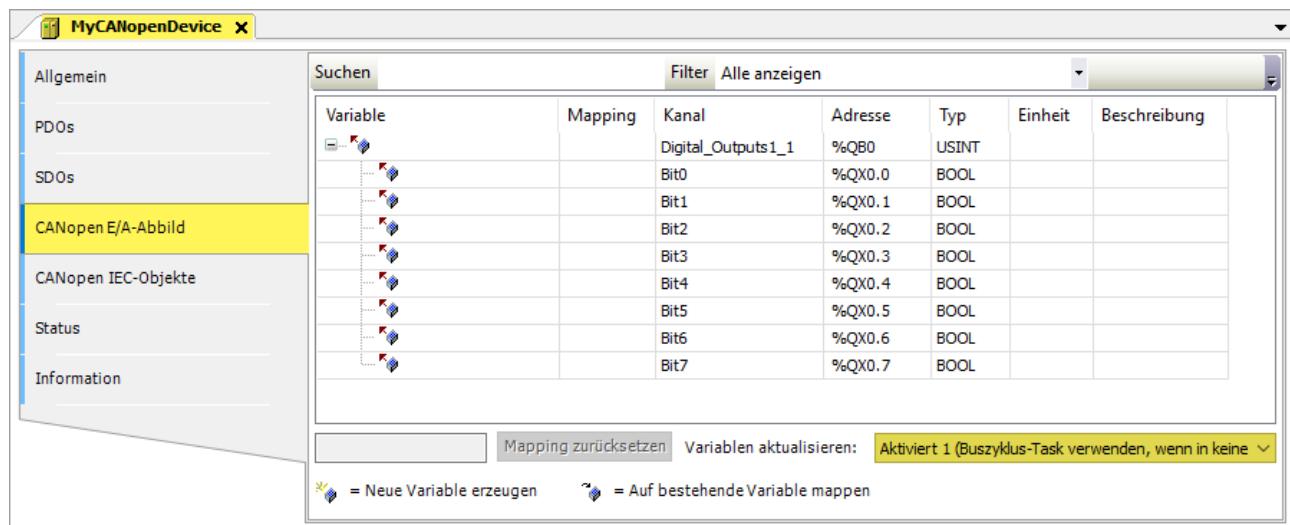


Abbildung 76: CODESYS - CAN-Bus Master - CAN-Slave konfigurieren

### 7.8.5.3 Programm Download und Test

Öffnen Sie nacheinander die Kommunikationseinstellungen für die beiden Geräte „PiXtendCAN\_Master“ und „PiXtendCAN\_Slave“. Wählen Sie die entsprechenden Raspberry Pi Geräte aus, auf die der Download erfolgen soll.

Anschließend verwenden Sie im Hauptmenü „Online“ die Funktion „Mehrfacher Download“, um die Applikationen gleichzeitig auf die entsprechenden Controller zu laden.

Hinweis: Sobald sich mehrere Applikationen im Projektbaum befinden kann mit einem Rechtsklick auf die Applikation -> „Aktive Applikation setzen“ die gewünschte Applikation ausgewählt und aktiv gesetzt werden.

Gehen Sie nacheinander „Online“ zu den Applikationen und starten Sie diese.

Wurde alles richtig konfiguriert und beide Geräte sind korrekt miteinander verbunden, dann werden die CAN-Einträge im Projektbaum grün markiert dargestellt.

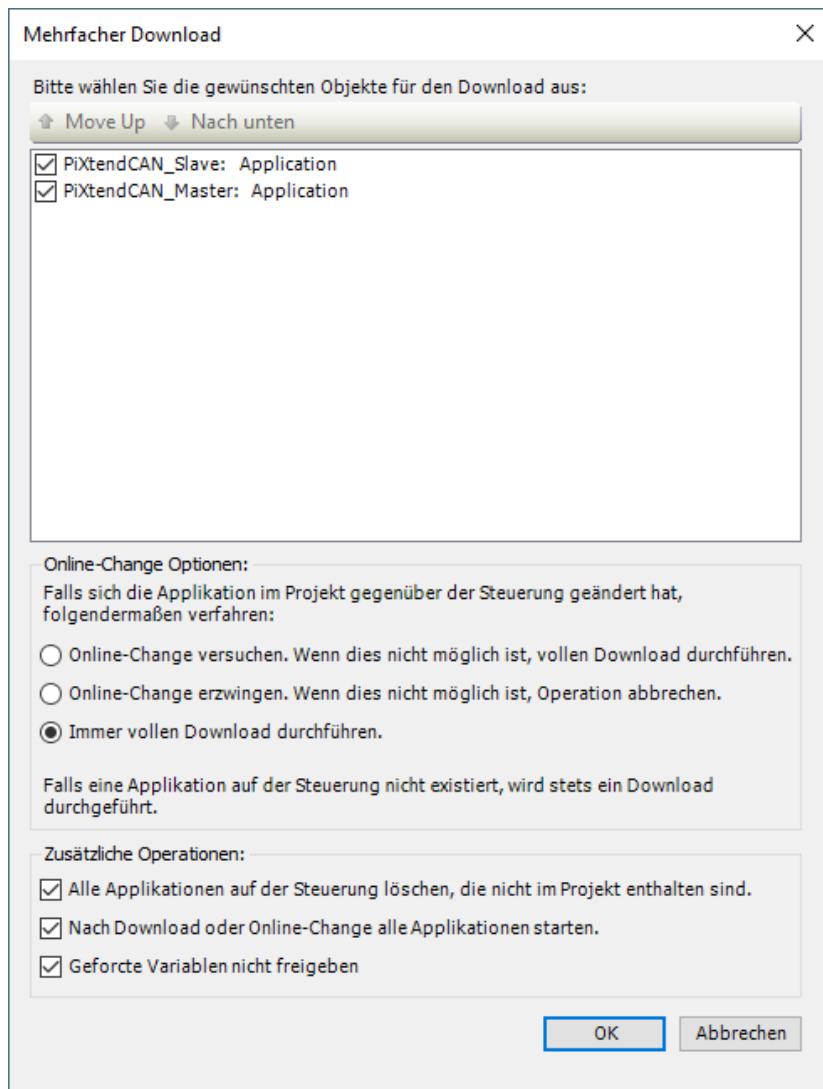


Abbildung 77: CODESYS - CAN-Bus Test - Mehrfacher Download

Öffnen Sie nun den E/A Bereich des „CANopen Masters“, Sie verändern und schreiben (Strg+F7) die Werte, dann werden diese Werte automatisch vom Master an den Slave übertragen:

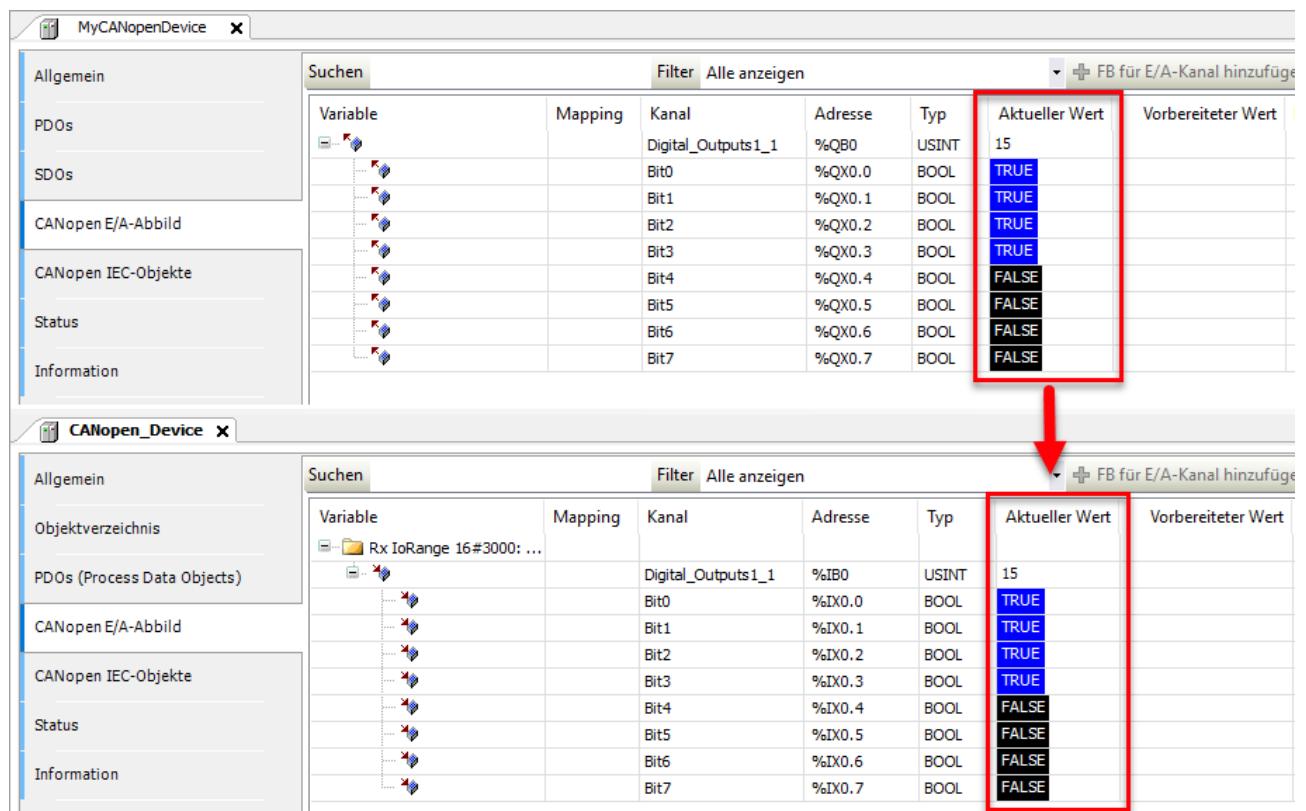


Abbildung 78: CODESYS - CAN-Bus Test - Master und Slave tauschen Daten aus

Natürlich könnte die Variable gemapped und beispielsweise direkt im Hauptprogramm verwendet werden.

## 7.9. CODESYS – PiXtend Retain Speicher

Das PiXtend V2 -S- verfügt über einen 32 Byte großen Flashspeicher (bei PiXtend V2 -L- sind es 64 Bytes), den ein CODESYS Anwender mit beliebigen Werten beschreiben kann. Dieses Kapitel veranschaulicht die Verwendung des Retain Speichers und zeigt auf, welche Punkte zu beachten sind. Wir verwenden ein PiXtend V2 -S-, Sie können gerne ein PiXtend V2 -L- nehmen.

### 7.9.1. Vorbereitung

Als Vorbereitung für dieses Beispiel benötigen Sie ein neues leeres CODESYS Standard-Projekt und der SPI-Bus ist mit einem SPI-Master und einem PiXtend V2 -S- konfiguriert.

In den Raspberry Pi GPIOs sollte der Pin 24 als Ausgang definiert sein und im E/A-Abbild den Variablenamen „SPI\_ENABLE“ erhalten.

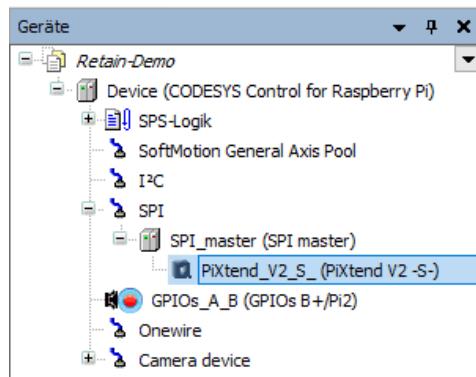


Abbildung 79: CODESYS - Retain-Demo – Gerätebaum

Im E/A-Abbild des PiXtend V2 -S- werden mehrere Variablen benötigt, diese sind nachfolgend tabellarisch aufgeführt.

Ordner	Kanal	Variable	Beschreibung
Control	RetainDataEnable	xRetainDataEnable	Ein- / Ausschalten der Retain Funktion
State	Firmware	byFirmware	Firmware Version des Mikrocontrollers
State	Hardware	byHardware	Hardware Revision des PiXtend V2 -S-
State	ModelIn	byModel	Modellnummer des PiXtend V2 -S-
State	RetainCRCError	xRetainCRCError	Retain CRC Fehlerbit
State	RetainVoltageError	xRetainVoltageError	Retain Spannungsversorgung unter 19 Volt
State	Run	xRun	Mikrocontroller aktiv Signal
State	CRCHeaderInError	xCRCHeaderInError	CRC Fehler in den SPI-Kopfdaten festgestellt
State	CRCDataInError	xCRCDataInError	CRC Fehler in den SPI-Nutzdaten gefunden
Retain Data	RetainDataOut	arRetainDataOut	Retain-Daten zum Mikrocontroller
Retain Data	RetainDataIn	arRetainDataIn	Retain-Daten vom Mikrocontroller

Das CODESYS Projekt ist jetzt vorbereitet, alle benötigten Einstellungen wurden vorgenommen.

## 7.9.2. Programm erstellen

Im Programm-Baustein „PLC\_PRG“ brauchen wir 4 neue Variablen, um unser Testprogramm zu steuern. Im Deklarationsabschnitt folgende Variablen anlegen:

- xInit: BOOL
- xRetainInit: BOOL
- iRetainStep: INT
- xSetnewValue: BOOL

```

1 PROGRAM PLC_PRG
2
3 VAR
4   xInit: BOOL;
5   xRetainInit: BOOL;
6   iRetainStep: INT;
7   xSetnewValue: BOOL;
8 END_VAR

```

Abbildung 80: CODESYS - Retain - Variablendeclaration

Im Programmteil fügen wir folgendes Programm ein:

```

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

```

Die Zahlen (4, 21 und 83) für die Vergleiche von byFirmware, byHardware und byModel müssen durch die tatsächlichen Werte des vorliegenden PiXtend V2 Boards ersetzt werden. Hier handelt es sich lediglich um Beispielwerte!

```

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

```

Abbildung 81: CODESYS - Retain - Beispielprogramm

#### Erläuterung zum Programmablauf:

In ersten Abschnitt mit der Zeile „If xInit = False then“ wird eine sehr kurze Initialisierung durchgeführt, es wird lediglich die SPI-Kommunikation mit „SPI\_ENABLE :=True“ eingeschaltet.

Im nächsten Abschnitt werden per Schritt kette bestehende Retain-Daten in CODESYS eingelesen, sofern die Prüfung verschiedener Variablen, siehe Vorbereitung, erfolgreich verläuft. Im Schritt 1 wird das erste Byte der RetainDataIn Array-Variable auf das erste Byte der RetainDataOut Array-Variable geschrieben und im 3 Schritt wird die Retain-Funktion des PiXtend V2 -S- eingeschaltet.

Wir empfehlen die Retain-Funktion immer zuletzt zu aktivieren, nachdem die „alten“ (bisherigen) Daten wiederhergestellt wurden.

Den letzten Abschnitt verwenden wir dazu, das erste Byte der RetainDataOut Array-Variable jeweils, um einen Zähler zu erhöhen, indem die Variable xSetnewValue auf True gesetzt wird. Wir trennen die Versorgung des PiXtend V2-S-. Nach einem Neustart sollten wir in der RetainDataOut Array-Variable den Wert vorfinden, der vor dem Neustart dort abzulesen war.

**Beispielablauf:**

1. Nach dem Programmstart haben arRetainDataOut[0] und arRetainDataIn[0] den gleichen Wert, hier im Beispiel ist es die Zahl 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;

```

Abbildung 82: CODESYS - Retain – Startwert 13

2. Nach dem Setzen der Variable xSetNewValue auf True, erhöht sich der Wert in arRetainDataOut[0] um einen Zähler auf 14.

```

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;

```

Abbildung 83: CODESYS - Retain – arRetainDataOut[0] wurde um 1 auf 14 erhöht

Die Variable arRetainDataIn[0] zeigt uns immer noch den „alten“ Wert an.

3. Wir ziehen den Stecker und trennen die Versorgung des PiXtend V2 -S-, warten kurz und stellen die Versorgung wieder her (Power-Cycle).

4. Nach dem Neustart wurde der alte Wert aus dem

Flashspeicher des Mikrocontrollers ist wiederhergestellt und sofort in das erste Byte der RetainDataOut Array-Variable geschrieben. Ab jetzt können wir mit diesem Wert weiterarbeiten.

```

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;

```

Abbildung 84: CODESYS - Retain – Nach Power-Cycle

## 7.9.3. Weitere Informationen

### 7.9.3.1 Datensicherheit erhöhen

Die Retain-Daten werden im Mikrocontroller Flash mit einem CRC abgespeichert und beim Start gelesen und geprüft. Es ist sinnvoll diese Prüfung ebenfalls in CODESYS durchzuführen, um die Korrektheit der persistenten Daten sicherzustellen.

Zu diesem Zweck kann eine Prüfsumme (CRC), mit 16 Bits erstellt werden. In CODESYS gibt es diese Funktionen in den Bibliotheken CAA Memory und CmpChecksum oder in der CODESYS Open Source Bibliothek, kurz OSCAT.

Eine 16 Bit Prüfsumme kann in zwei Bytes aufgeteilt und in die letzten beiden Bytes des Retain Speichers geschrieben werden. Nach einem Neustart besteht in CODESYS die Möglichkeit, die wiederhergestellten Werte anhand dieser Prüfsumme zu überprüfen.

### 7.9.3.2 Arbeiten mit Strukturen

Die aus- und eingehenden Retain-Daten werden als Array mit 32 Bytes bereitgestellt. Jedes Byte in diesem Array lässt sich einzeln ansprechen, das ermöglicht eine einfache Verarbeitung in einer Schleife und erlaubt einen direkten Zugriff.

Werden 16 Bit, 32 Bit oder 64 Bit Werte im Retain Speicher abgelegt, so ist zu beachten, dass da man jedes Byte einzeln zuweisen muss.

Um diese Situationen zu vereinfachen bietet CODESYS zwei praktische Ansätze. Mit Strukturen (Struct) lassen sich Daten in CODESYS organisieren und zusammenfassen. Kombiniert man eine Struktur mit einem Union-Datentyp, spricht man von einer Überlagerung und erhält die Möglichkeit die Retain-Daten über eine Struktur anzusprechen. Diese Vorgehensweise hat den Vorteil, dass man sich nicht um die Aufteilung der einzelnen Bytes im Array kümmern müssen.

#### Beispiel:

Eine DWORD Variable ist 4 Bytes groß. Kombinieren wir diese Variable in einem Union-Datentyp mit einem Array der Länge 4 Bytes, wird jeder in die DWORD Variable geschriebene Wert automatisch auf die 4 Bytes des Arrays aufgeteilt. Da die DWORD Variable und das 4 Byte große Array denselben Speicherplatz im CODESYS Arbeitsspeicher belegen (Überlagerung) ist das möglich.

```
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 – Raspberry Pi herunterfahren

CODESYS bietet keine eigene Funktion um den Raspberry Pi „ordentlich“ herunterzufahren, auszuschalten und von der Stromversorgung zu trennen.

Das Herunterfahren des Raspberry Pi bietet Vorteile in Bezug auf die Datensicherheit, zusätzlich kann die CODESYS eigene Retain-Funktion verwendet werden. Die Werte der CODESYS Retain-Funktion werden nur beim Herunterfahren auf die SD-Karte geschrieben und beim nächsten Start wieder eingelesen. Das bietet sich an, wenn man mehr als 32 oder 64 Bytes an Retain-Speicher benötigt.

Das nachfolgende Beispiel zeigt das Herunterfahren aus CODESYS

Fügen Sie Ihrem Projekt 2 die beiden neuen Bibliotheken „SysProcess“ und „SysTypes2 Interfaces“. hinzu. Nutzen Sie beim Hinzufügen die Suchleiste im Bibliotheksverwalter und geben Sie die Namen der Bibliotheken direkt ein. Um die Systembibliotheken sichtbar zu machen klicken Sie auf das Plus-Symbol bei Bibliothek hinzufügen.

In der „SysProcess“-Bibliothek befindet sich die Funktion „SysProcessExecuteCommand“ die wir für das Herunterfahren benötigen.

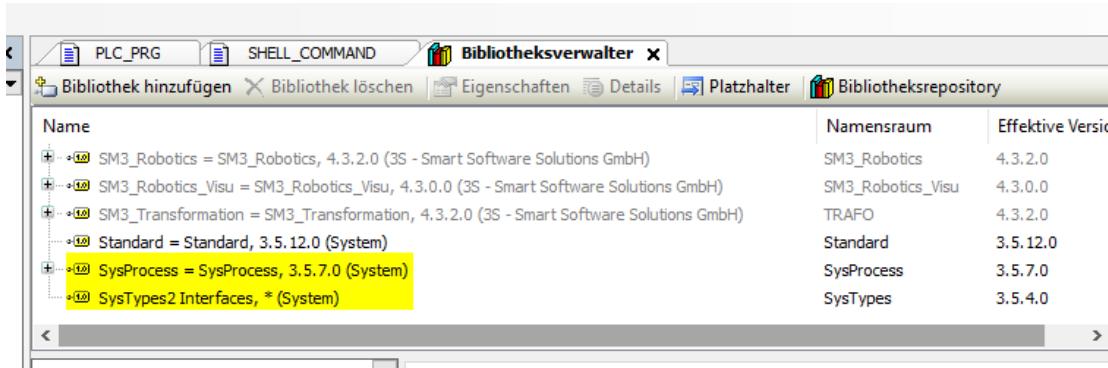


Abbildung 85: CODESYS – Herunterfahren - Bibliotheksverwalter

Beim Hinzufügen von Bibliotheken ist die Suchleiste (2) am oberen Rand des Dialogs sehr hilfreich, aktivieren Sie die erweiterte Ansicht (1). Klicken Sie auf den fettgedruckten Namen der Bibliothek (3) um diese auszuwählen.

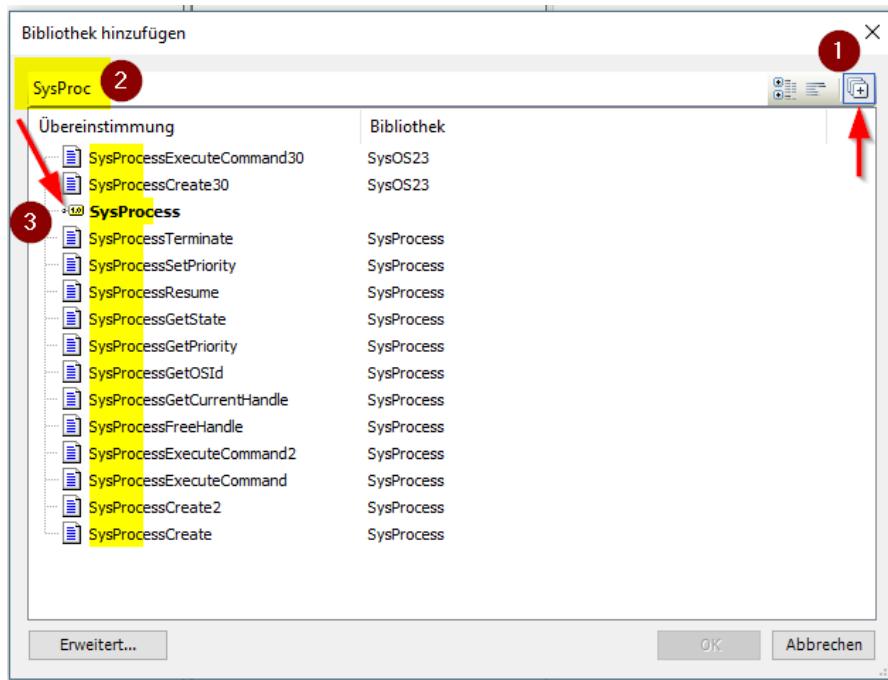


Abbildung 86: CODESYS - Herunterfahren - Bibliothek hinzufügen

Wurden beide Bibliotheken dem Projekt hinzugefügt, legen Sie einen neuen Programmstein an und zusätzlich 4 neue Variablen.

Es werden 4 Variablen mit folgenden Typen benötigt:

- 1 x BOOL
- 1 x DINT
- 1 x SysTypes.RTS\_IEC\_RESULT
- 1 x STRING

```

1 PROGRAM SHELL_COMMAND
2 VAR
3     _xDoExecuteShell : BOOL;
4     _RetVal: DINT;
5     _pResult: SysTypes.RTS_IEC_RESULT;
6     _sCommand: STRING;
7 END_VAR
8
9

```

Abbildung 87: CODESYS - Herunterfahren – Variablen

Mit einer IF Abfrage im neuen Programmbaustein lässt sich die Ausführung der Funktion „SysProcessExecuteCommand“ kapseln. Es ist wichtig, dass wir den Befehl zum Herunterfahren nur einmal ausführen. Geschieht die Ausführung wiederholt, kann es bei CODESYS zu Problemen führen.

Befehl für das Betriebssystem lautet:

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

Beim „&“-Zeichen und dem Wort „disown“ handelt es sich um keinen Druckfehler, diese Teile des Befehls dürfen nicht fehlen.

Ist die boolsche Variable TRUE, gehen wir in die IF Abfrage rein, setzen den Bool wieder zurück, setzen unser gewünschtes Kommando in die Stringvariable und übergeben alles an die Funktion „SysProcessExecuteCommand“. Das Resultat der Funktion übergeben wir als Pointer mit dem Operator ADR.

Auf diesem Weg lässt sich der Raspberry Pi einfach herunterfahren. Nach dem Aufruf des Befehls sollte man kurz warten, ähnlich wie bei Windows, bis alle Dienste und CODESYS beendet wurden. Dann ist das Betriebssystem in einen sicheren Zustand und die Versorgung gefahrlos trennen.

```

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

```

Abbildung 88: CODESYS - Herunterfahren - Beispielprogramm

## 7.11. PiXtend V2 -S- SPI Device

### 7.11.1. SPI Device Parameter

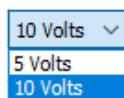
Das PiXtend V2 -S- SPI Gerät in CODESYS verfügt über einige Einstellungen, die vor dem Übersetzen im Reiter SPI devices Parameter eingestellt werden können. Diese Parameter lassen sich während der Laufzeit nicht mehr ändern.

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	

Abbildung 89: CODESYS PiXtend V2 -S- - SPI devices Parameter Reiter

### 7.11.1.1 5 Volt/10 Volt Jumper Setting

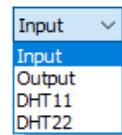
Dieser Parameter dient dazu dem CODESYS Treiber für das PiXtend mitzuteilen, ob die analogen Signale an AnalogIn0 und AnalogIn1 im Bereich von 5 Volt (Jumper gesetzt) oder 10 Volt (Jumper nicht gesetzt, Werkseinstellung) liegen. Der Treiber verwendet je nach Einstellung einen anderen Umrechnungsfaktor zur Berechnung der Spannungswerte. Das PiXtend erkennt nicht ob ein Jumper gesetzt ist. Der Anwender muss die Einstellung in CODESYS mit den tatsächlichen (physikalisch) vorhanden Jumfern auf PiXtend abgleichen und eine entsprechende Einstellung bei diesem Parameter vornehmen.



### 7.11.1.2 GPIO Configuration

In der GPIO Konfiguration wird festgelegt, welche der 4 nachstehenden Einstellungen jeder einzelne GPIO haben soll.

- Input (Eingang) – Standardeinstellung
- Output (Ausgang)
- DHT11 - (Eingang für Temperatur- und Luftfeuchtemessung)
- DHT22 - (Eingang für Temperatur- und Luftfeuchtemessung)



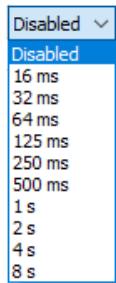
Ferner bietet die GPIO Konfiguration die Möglichkeit die GPIO PullUps zu aktivieren. Steht diese Einstellung auf On (An), so lassen sich die GPIO PullUps über das Setzen des jeweiligen GPIO Ausgangs aktivieren, während ein GPIO als Input (Eingang) konfiguriert ist.

### 7.11.1.3 Mikrocontroller Settings

Der Mikrocontroller des PiXtend V2 -S- bietet zwei Einstellungen, die der Anwender verändern kann.

#### 1. WatchdogEnable

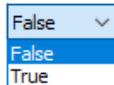
Mit dieser Einstellung kann der Watchdog des Mikrocontrollers ein- und ausgeschaltet werden. Die Stellung Disabled bedeutet der Watchdog ist aus (inaktiv), die Auswahl einer entsprechenden Wartezeit aktiviert den Watchdog. Es stehen Zeiten von 16 Millisekunden bis hin zu 8 Sekunden zur Auswahl.



#### 2. StateLEDDisable

Das PiXtend V2 -S- verfügt über eine Status LED, die beim Auftreten eines Problems im Mikrocontroller, einen Fehler signalisiert. Sollte es erforderlich sein, kann diese LED deaktiviert werden.

Die Einstellung False bedeutet die LED ist aktiv (ein), das Setzen auf True hingegen deaktiviert die LED.



## 7.11.2. I/O Übersicht

In diesem Kapitel finden Sie eine Aufstellung aller I/Os die in CODESYS für das PiXtend V2 -S- zur Verfügung stehen, zusammen mit dem Datentyp und einer kurzen Beschreibung.

Bezeichnung	Typ	Datentyp	Beschreibung
<b>Control</b>			
PWM0Ctrl1	Ausgang	WORD	Die Auswirkung des PWM0Ctrl1 Wertes hängt vom gewählten Mode in PWM0Ctrl0 ab. Dieser Wert gilt für die Kanäle A & B von PWM 0.
PWM1Ctrl1	Ausgang	BYTE	Die Auswirkung des PWM1Ctrl1 Wertes hängt vom gewählten Mode in PWM1Ctrl0 ab. Dieser Wert gilt für die Kanäle A & B von PWM 1.
PWM0Ctrl0	Ausgang	BYTE	PWM 0 Control 0 – Ermöglicht das Einstellen des PWM Mode, Kanal Enable und Presaclers. PWM 0 verwendet 16 Bit Werte.
PWM1Ctrl0	Ausgang	BYTE	PWM 1 Control 0 – Ermöglicht das Einstellen des PWM Mode, Kanal Enable und Presaclers. PWM 1 verwendet 8 Bit Werte.
GPIODebounce01	Ausgang	BYTE	Debounce der GPIO Eingänge 0 and 1, Wert * Bus-Zyklus Intervall = Debounce Zeit.
GPIODebounce23	Ausgang	BYTE	Debounce der GPIO Eingänge 2 and 3, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce01	Ausgang	BYTE	Debounce der Digital Eingänge 0 und 1, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce23	Ausgang	BYTE	Debounce der Digital Eingänge 2 und 3, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce45	Ausgang	BYTE	Debounce der Digital Eingänge 4 und 5, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce67	Ausgang	BYTE	Debounce der Digital Eingänge 6 und 7, Wert * Bus-Zyklus Intervall = Debounce Zeit.
SafeState	Ausgang	BIT	Versetzt den Mikrocontroller in den sicheren Zustand, sollte die Steuerung einen Neustart oder Shutdown benötigen. True = Sicherer Zustand, False = Bleibe EIN.
RetainDataEnable	Ausgang	BIT	Retain-Daten Speicherung im Mikrocontroller einschalten. TRUE = On, FALSE = Off.
RetainCopy	Ausgang	BIT	Wenn TRUE, dann werden die aktuellen RetainDataOut Bytes im Mikrocontroller nach RetainDataIn kopiert. Bei FALSE werden die aktuell im Flash gespeicherten Daten in RetainDataIn ausgegeben.
<b>State</b>			
Firmware	Eingang	BYTE	Firmware-Version des Mikrocontrollers auf PiXtend V2 -S- Board.
Hardware	Eingang	BYTE	PiXtend V2 -S- Boardrevision, 20 = 2.0, 21 = 2.1, usw.
ModellIn	Eingang	BYTE	Modellnummer ausgelesen vom PiXtend V2 -S-.
Error	Eingang	BYTE	Fehlerbyte vom Mikrocontroller. Siehe Status-Bytes.
RetainCRCError	Eingang	BIT	Der CRC des Retain-Speichers im Mikrocontroller ist falsch, die gespeicherten Daten können fehlerhaft sein.
RetainVoltageError	Eingang	BIT	Wenn TRUE, dann kann die Retain-Funktion nicht genutzt werden, die Versorgungsspannung liegt unter 19 Volt.

Run	Eingang	BIT	Die Kommunikation mit dem Mikrocontroller ist in Ordnung.
BusCycleError	Eingang	BIT	Die Zykluszeit (Intervall) des Buszyklustasks ist zu schnell.
CRCHeaderInError	Eingang	BIT	Ein CRC Fehler wurde im SPI Header festgestellt, es liegt ein Kommunikationsfehler vor.
CRCDataInError	Eingang	BIT	Ein CRC Fehler in den PiXtend V2 -S- SPI Daten wurde festgestellt, es gibt keine brauchbaren Daten.
ModellInError	Eingang	BIT	Modellfehler festgestellt, das eingestellte PiXtend V2 -S- Model in CODESYS V3 und die tatsächliche Hardware stimmen nicht überein.
Sensor0Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 0 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
Sensor1Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 1 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
Sensor2Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 2 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
Sensor3Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 3 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
<hr/>			
<b>Analog Inputs</b>			
AnalogIn0	Eingang	REAL	Analog Eingang 0 gemessen in Volt [V], Bereich ist 0 Volt bis +10 Volt.
AnalogIn0Raw	Eingang	WORD	Analog Eingang 0 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
AnalogIn1	Eingang	REAL	Analog Eingang 1 gemessen in Volt [V], Bereich ist 0 Volt bis +10 Volt.
AnalogIn1Raw	Eingang	WORD	Analog Eingang 1 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
<hr/>			
<b>Digital Inputs</b>			
DigitalInputs	Eingang	BYTE	Digital Eingänge 0-7, Bit-Zugriff ist möglich.
GPIOInputs	Eingang	BYTE	GPIO Eingänge 0-3 als Byte, Bit-Zugriff ist möglich. Diese Funktion ist abhängig von den Einstellungen im Reiter Parameter.
<hr/>			
<b>Digital Outputs</b>			
DigitalOutputs	Ausgang	BYTE	Digital Ausgänge 0-3 als Byte, Bit-Zugriff ist möglich.
GPIOOutputs	Ausgang	BYTE	GPIO Ausgänge 0-3 als Byte, Bit-Zugriff ist möglich. Diese Funktion ist abhängig von den Einstellungen im Reiter Parameter.
RelayOutputs	Ausgang	BYTE	Relais-Ausgänge 0-3 als Byte, Bit-Zugriff ist möglich.
<hr/>			
<b>PWM Outputs</b>			

PWM0A	Ausgang	WORD	PWM 0, Kanal A Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM0Ctrl0 ab.
PWM0B	Ausgang	WORD	PWM 0, Kanal B Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM0Ctrl0 ab.
PWM1A	Ausgang	BYTE	PWM 1, Kanal A Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM1Ctrl0 ab.
PWM1B	Ausgang	BYTE	PWM 1, Kanal B Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM1Ctrl0 ab.
<b>Humidity Inputs</b>	Luftfeuchtwert eines Sensors der an GPIO 0-3 angeschlossen ist. Die Sensoroption muss im Reiter Parameter aktiviert sein. Die Einheit ist %RH.		
Humid0	Eingang	REAL	
Humid1	Eingang	REAL	
Humid2	Eingang	REAL	
Humid3	Eingang	REAL	
<b>Temperature Inputs</b>	Temperaturwert eines Sensors der an GPIO 0-3 angeschlossen ist. Die Sensoroption muss im Reiter Parameter aktiviert sein. Die Einheit ist Grad Celsius (°C).		
Temp0	Eingang	REAL	
Temp1	Eingang	REAL	
Temp2	Eingang	REAL	
Temp3	Eingang	REAL	
<b>Retain Data</b>	Die Retain-Daten können dazu verwendet werden, um in Fall eines Stromausfalls bis zu 32 Bytes im Speicher der Mikrocontrollers zu sichern. Nach einem Neustart lassen sich diese Daten wiederherstellen bzw. zurücklesen.		
RetainDataOut	Ausgang	ARRAY[0..31] OF BYTE	
RetainDataIn	Eingang	ARRAY[0..31] OF BYTE	

Tabelle 3: CODESYS E/A Übersicht für PiXtend V2 -S-

## 7.12. PiXtend V2 -L- SPI Device

### 7.12.1. SPI Device Parameter

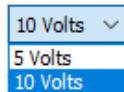
Das PiXtend V2 -L- SPI Gerät in CODESYS verfügt über einige Einstellungen, die vor dem Übersetzen im Reiter SPI devices Parameter eingestellt werden können. Diese Parameter können später zur Laufzeit nicht mehr geändert werden.

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	

Abbildung 90: CODESYS PiXtend V2 -L- - SPI devices Parameter Reiter

#### 7.12.1.1 5 Volt/10 Volt Jumper Setting

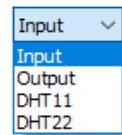
Dieser Parameter dient dazu dem CODESYS Treiber für das PiXtend mitzuteilen, ob die analogen Signale an AnalogIn0 bis AnalogIn3 im Bereich von 5 Volt (Jumper gesetzt) oder 10 Volt (Jumper nicht gesetzt, Werkseinstellung) liegen. Der Treiber verwendet je nach Einstellung einen anderen Umrechnungsfaktor zur Berechnung der Spannungswerte. Das PiXtend erkennt nicht ob ein Jumper gesetzt ist. Der Anwender muss die Einstellung in CODESYS mit den tatsächlichen (physikalisch) vorhanden Jumpern auf PiXtend abgleichen und eine entsprechende Einstellung bei diesem Parameter vornehmen.



### 7.12.1.2 GPIO Configuration

Die GPIO Konfiguration ermöglicht die Festlegung, welche der vier Einstellung jeder einzelne GPIO haben soll.:

- Input (Eingang) – Standardeinstellung
- Output (Ausgang)
- DHT11 - (Eingang für Temperatur- und Luftfeuchtemessung)
- DHT22 - (Eingang für Temperatur- und Luftfeuchtemessung)



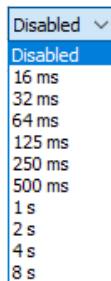
Ferner bietet die GPIO Konfiguration die Möglichkeit die GPIO PullUps zu aktivieren. Steht diese Einstellung auf On (An), so lassen sich die GPIO PullUps über das Setzen des jeweiligen GPIO Ausgangs aktivieren, während ein GPIO als Input (Eingang) konfiguriert ist.

### 7.12.1.3 Mikrocontroller Settings

Der Mikrocontroller des PiXtend V2 -L- bietet zwei Einstellungen, die der Anwender verändern kann.

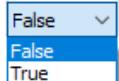
#### 1. WatchdogEnable

Mit dieser Einstellung kann der Watchdog des Mikrocontrollers ein- und ausgeschaltet werden. Die Stellung Disabled bedeutet der Watchdog ist aus (inaktiv), die Auswahl einer entsprechenden Wartezeit aktiviert den Watchdog. Es stehen Zeiten von 16 Millisekunden bis hin zu 8 Sekunden zur Auswahl.



#### 2. StateLEDDisable

Das PiXtend V2 -L- verfügt über eine Status LED, die beim Auftreten eines Problems im Mikrocontroller, einen Fehler signalisiert. Sollte es erforderlich sein, kann diese LED deaktiviert werden. Die Einstellung False bedeutet die LED ist aktiv (ein), das Setzen auf True hingegen deaktiviert die LED.



## 7.12.2. I/O Übersicht

In diesem Kapitel finden Sie eine Aufstellung aller I/Os die in CODESYS für das PiXtend V2 -L- zur Verfügung stehen, zusammen mit dem Datentyp und einer kurzen Beschreibung.

Bezeichnung	Typ	Datentyp	Beschreibung
<b>Control</b>			
PWM0Ctrl1	Ausgang	WORD	Die Auswirkung des PWM0Ctrl1 Wertes hängt vom gewählten Mode in PWM0Ctrl0 ab. Dieser Wert gilt für die Kanäle A & B von PWM 0.
PWM1Ctrl1	Ausgang	WORD	Die Auswirkung des PWM1Ctrl1 Wertes hängt vom gewählten Mode in PWM1Ctrl0 ab. Dieser Wert gilt für die Kanäle A & B von PWM 1.
PWM2Ctrl1	Ausgang	WORD	Die Auswirkung des PWM2Ctrl1 Wertes hängt vom gewählten Mode in PWM2Ctrl0 ab. Dieser Wert gilt für die Kanäle A & B von PWM 2.
PWM0Ctrl0	Ausgang	BYTE	PWM 0 Control 0 – Ermöglicht das Einstellen des PWM Mode, Kanal Enable und Presaclers. PWM 0 verwendet 16 Bit Werte.
PWM1Ctrl0	Ausgang	BYTE	PWM 1 Control 0 – Ermöglicht das Einstellen des PWM Mode, Kanal Enable und Presaclers. PWM 1 verwendet 16 Bit Werte.
PWM2Ctrl0	Ausgang	BYTE	PWM 2 Control 0 – Ermöglicht das Einstellen des PWM Mode, Kanal Enable und Presaclers. PWM 2 verwendet 16 Bit Werte.
GPIODebounce01	Ausgang	BYTE	Debounce der GPIO Eingänge 0 and 1, Wert * Bus-Zyklus Intervall = Debounce Zeit.
GPIODebounce23	Ausgang	BYTE	Debounce der GPIO Eingänge 2 and 3, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce01	Ausgang	BYTE	Debounce der Digital Eingänge 0 und 1, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce23	Ausgang	BYTE	Debounce der Digital Eingänge 2 und 3, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce45	Ausgang	BYTE	Debounce der Digital Eingänge 4 und 5, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce67	Ausgang	BYTE	Debounce der Digital Eingänge 6 und 7, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce89	Ausgang	BYTE	Debounce der Digital Eingänge 8 und 9, Wert * Bus-Zyklus Interval = Debounce Zeit.
DigitalInDebounce1011	Ausgang	BYTE	Debounce der Digital Eingänge 10 und 11, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce1213	Ausgang	BYTE	Debounce der Digital Eingänge 12 und 13, Wert * Bus-Zyklus Intervall = Debounce Zeit.
DigitalInDebounce1415	Ausgang	BYTE	Debounce der Digital Eingänge 14 und 15, Wert * Bus-Zyklus Intervall = Debounce Zeit.
SafeState	Ausgang	BIT	Versetzt den Mikrocontroller in den sicheren Zustand, sollte die Steuerung einen Neustart oder Shutdown benötigen. True = Sicherer Zustand, False = Bleibe EIN.
RetainDataEnable	Ausgang	BIT	Retain-Daten Speicherung im Mikrocontroller einschalten. TRUE = On, FALSE = Off.
RetainCopy	Ausgang	BIT	Wenn TRUE, dann werden die aktuellen RetainDataOut Bytes im Mikrocontroller nach RetainDataIn kopiert. Bei FALSE werden die aktuell im Flash gespeicherten Daten in RetainDataIn ausgegeben.
<b>State</b>			

Firmware	Eingang	BYTE	Firmware-Version des Mikrocontrollers auf PiXtend V2 -L- Board.
Hardware	Eingang	BYTE	PiXtend V2 -L- Boardrevision, 20 = 2.0, 21 = 2.1, usw.
Modelln	Eingang	BYTE	Modellnummer ausgelesen vom PiXtend V2 -L-.
Error	Eingang	BYTE	Fehlerbyte vom Mikrocontroller. Siehe Status-Bytes.
RetainCRCError	Eingang	BIT	Der CRC des Retain-Speichers im Mikrocontroller ist falsch, die gespeicherten Daten können fehlerhaft sein.
RetainVoltageError	Eingang	BIT	Wenn TRUE, dann kann die Retain-Funktion nicht genutzt werden, die Versorgungsspannung liegt unter 19 Volt.
Run	Eingang	BIT	Die Kommunikation mit dem Mikrocontroller ist in Ordnung.
BusCycleError	Eingang	BIT	Die Zykluszeit (Intervall) des Buszyklustasks ist zu schnell.
CRCHeaderInError	Eingang	BIT	Ein CRC Fehler wurde im SPI Header festgestellt, es liegt ein Kommunikationsfehler vor.
CRCDataInError	Eingang	BIT	Ein CRC Fehler in den PiXtend V2 -L- SPI Daten wurde festgestellt, es gibt keine verwendbaren Daten.
ModellnError	Eingang	BIT	Modellfehler festgestellt, das eingestellte PiXtend V2 -L- Model in CODESYS V3 und die tatsächliche Hardware stimmen nicht überein.
I2CError	Eingang	BIT	Ein Fehler auf dem I <sup>2</sup> C Bus wurde festgestellt oder der Mikrocontroller konnte keinen Slave MC finden.
Sensor0Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 0 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
Sensor1Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 1 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
Sensor2Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 2 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
Sensor3Error	Eingang	BIT	Wenn TRUE, dann konnte der Mikrocontroller keine Daten vom Sensor am GPIO 3 lesen. Die Temperatur- und Luftfeuchtewerte sind ungültig.
<hr/>			
<b>Analog Inputs</b>			
AnalogIn0	Eingang	REAL	Analog Eingang 0 gemessen in Volt [V], Bereich ist 0 Volt bis +10 Volt.
AnalogIn0Raw	Eingang	WORD	Analog Eingang 0 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
AnalogIn1	Eingang	REAL	Analog Eingang 1 gemessen in Volt [V], Bereich ist 0 Volt bis +10 Volt.
AnalogIn1Raw	Eingang	WORD	Analog Eingang 1 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
AnalogIn2	Eingang	REAL	Analog Eingang 2 gemessen in Volt [V], Bereich ist 0 Volt bis +10 Volt.
AnalogIn2Raw	Eingang	WORD	Analog Eingang 2 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
AnalogIn3	Eingang	REAL	Analog Eingang 3 gemessen in Volt [V], Bereich ist 0 Volt bis +10 Volt.

AnalogIn3Raw	Eingang	WORD	Analog Eingang 3 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
AnalogIn4	Eingang	REAL	Analog Eingang 4 gemessen in Milliampere [mA], Bereich ist 0 mA bis +20 mA.
AnalogIn4Raw	Eingang	WORD	Analog Eingang 4 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
AnalogIn5	Eingang	REAL	Analog Eingang 5 gemessen in Milliampere [mA], Bereich ist 0 mA bis +20 mA.
AnalogIn5Raw	Eingang	WORD	Analog Eingang 5 Rohwert, tatsächlicher 16 Bit-Wert vom ADC.
<hr/>			
<b>Digital Inputs</b>			
DigitalInputs0	Eingang	BYTE	Digital Eingänge 0-7, Bit-Zugriff ist möglich.
DigitalInputs1	Eingang	BYTE	Digital Eingänge 8-15, Bit-Zugriff ist möglich.
GPIOInputs	Eingang	BYTE	GPIO Eingänge 0-3 als Byte, Bit-Zugriff ist möglich. Diese Funktion ist abhängig von den Einstellungen im Reiter Parameter.
<hr/>			
<b>Digital Outputs</b>			
DigitalOutputs0	Ausgang	BYTE	Digital Ausgänge 0-7 als Byte, Bit-Zugriff ist möglich.
DigitalOutputs1	Ausgang	BYTE	Digital Ausgänge 8-11 als Byte, Bit-Zugriff ist möglich.
GPIOOutputs	Ausgang	BYTE	GPIO Ausgänge 0-3 als Byte, Bit-Zugriff ist möglich. Diese Funktion ist abhängig von den Einstellungen im Reiter Parameter.
RelayOutputs	Ausgang	BYTE	Relais-Ausgänge 0-3 als Byte, Bit-Zugriff ist möglich.
<hr/>			
<b>PWM Outputs</b>			
PWM0A	Ausgang	WORD	PWM 0, Kanal A Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM0Ctrl0 ab.
PWM0B	Ausgang	WORD	PWM 0, Kanal B Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM0Ctrl0 ab.
PWM1A	Ausgang	WORD	PWM 1, Kanal A Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM1Ctrl0 ab.
PWM1B	Ausgang	WORD	PWM 1, Kanal B Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM1Ctrl0 ab.
PWM2A	Ausgang	WORD	PWM 2, Kanal A Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM2Ctrl0 ab.
PWM2B	Ausgang	WORD	PWM 2, Kanal B Ausgabewert. Das tatsächliche PWM-Verhalten hängt von der Mode-Einstellung im Byte PWM2Ctrl0 ab.
<hr/>			
<b>Humidity Inputs</b>	Luftfeuchtwert eines Sensors der an GPIO 0-3 angeschlossen ist. Die Sensoroption muss im Reiter Parameter aktiviert sein. Die Einheit ist %RH.		

Humid0	Eingang	REAL	
Humid1	Eingang	REAL	
Humid2	Eingang	REAL	
Humid3	Eingang	REAL	
<b>Temperature Inputs</b>	Temperaturwert eines Sensors der an GPIO 0-3 angeschlossen ist. Die Sensoroption muss im Reiter Parameter aktiviert sein. Die Einheit ist Grad Celsius (°C).		
Temp0	Eingang	REAL	
Temp1	Eingang	REAL	
Temp2	Eingang	REAL	
Temp3	Eingang	REAL	
<b>Retain Data</b>	Die Retain-Daten können dazu verwendet werden, um in Fall eines Stromausfalls bis zu 64 Bytes im Speicher der Mikrocontrollers zu sichern. Nach einem Neustart lassen sich diese Daten wiederherstellen bzw. zurücklesen.		
RetainDataOut	Ausgang	ARRAY[0..63] OF BYTE	
RetainDataIn	Eingang	ARRAY[0..63] OF BYTE	

Tabelle 4: CODESYS E/A Übersicht für PiXtend V2 -L-

## 7.13. FAQ – Häufig gestellte Fragen und Fehlerbehandlung

### 7.13.1. CODESYS Raspberry Pi Runtime und PiXtend V2

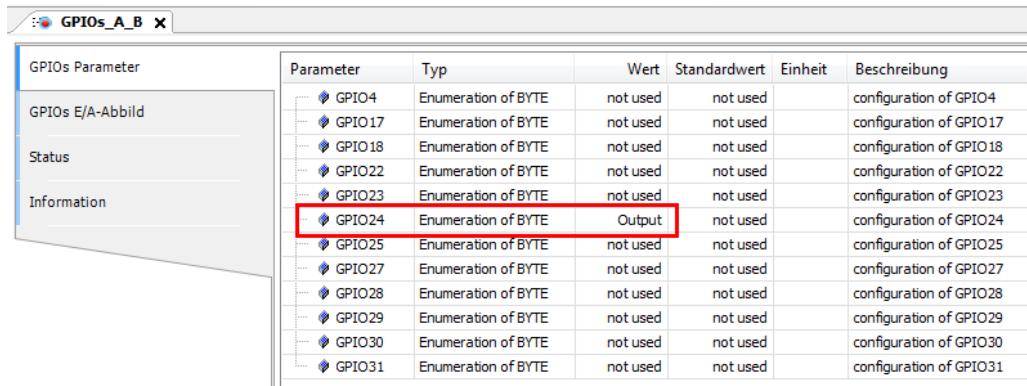
#### Problem

Die grüne LED „+5V“ leuchtet und der Raspberry Pi fährt normal hoch. Der Datenaustausch zwischen Raspberry Pi und PiXtend V2 scheint nicht zu funktionieren (Relais oder Ausgänge lassen sich nicht setzen).

#### Fehlersuche

Überprüfen Sie, ob der Schalter „SPI\_EN“ auf der Position „ON“ steht. Dieser ermöglicht die Kommunikation zwischen Raspberry Pi und PiXtend V2.

Wenn Sie unter CODESYS arbeiten und ein eigenes Projekt angelegt haben, so ist darauf zu achten, dass der GPIO24 (des Raspberry Pi) als Ausgang konfiguriert und auf „TRUE“ gesetzt wurde (siehe Abb. 91). Der GPIO24 ist das Signal, welches über den „SPI\_EN“ Schalter übertragen wird und die Datenübertragung aktiviert.



The screenshot shows the 'GPIOs\_A\_B' configuration window in CODESYS. On the left, there's a sidebar with tabs for 'GPIOs Parameter', 'GPIOs E/A-Abbildung', 'Status', and 'Information'. The main area is a table with columns: Parameter, Typ, Wert, Standardwert, Einheit, and Beschreibung. The table lists GPIO pins from 4 to 31. The row for GPIO24 is highlighted with a red box. The 'Typ' column for GPIO24 shows 'Enumeration of BYTE' and the 'Wert' column shows 'Output'.

Parameter	Typ	Wert	Standardwert	Einheit	Beschreibung
GPIO4	Enumeration of BYTE	not used	not used		configuration of GPIO4
GPIO17	Enumeration of BYTE	not used	not used		configuration of GPIO17
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
GPIO29	Enumeration of BYTE	not used	not used		configuration of GPIO29
GPIO30	Enumeration of BYTE	not used	not used		configuration of GPIO30
GPIO31	Enumeration of BYTE	not used	not used		configuration of GPIO31

Abbildung 91: CODESYS - GPIO24 als Ausgang konfiguriert

## 7.13.2. Serielle Kommunikation

### 7.13.2.1 Es werden nicht alle Zeichen übertragen oder kommen nacheinander an

Werden zu viele Zeichen gesendet ist es möglich, dass nicht alle Zeichen auf einmal übertragen werden. Ein Zyklus reicht zum Senden aller Zeichen nicht aus. Abhilfe kann eine Erhöhung der Baudrate schaffen.

Führen Sie folgende Rechnung durch um zu überschlagen, wie viele Zeichen übertragen werden können:

Anzahl der möglichen Zeichen = (Zykluszeit \* Baudrate)/zu übertragende Bits

- ▶ die Zykluszeit wird in Sekunden angegeben (10 ms = 0,010 s)
- ▶ die zu übertragenden Bits enthalten je nach Protokoll ein Start- oder Stopbit
- ▶ ein Zeichen besteht aus 8 Bits (= 1 Byte)

### 7.13.2.2 Wozu werden im Programm die GPIOs 18 und 22 umgeschaltet?

Gilt nur für PiXtend V1.2/V1.3 und PiXtend V2 -L-!

GPIO_18:TRUE	=>	RS485
	FALSE	=> RS232
GPIO_22:	TRUE	=> RS485 Sende-Modus
	FALSE	=> RS485 Empfangs-Modus

Die Umschaltung von Senden und Empfangen bei RS485 erfolgt manuell, da der Raspberry Pi nicht die dafür benötigten Pins (RTS/CTS) zur Verfügung stellt.

#### NOTICE

Das PiXtend V2 -L- verfügt zusätzlich über eine automatische Sende-/Empfangsmodusumschaltung, die ab Werk voreingestellt ist. Der GPIO 22 wird hier nicht benötigt.

### 7.13.2.3 Das Scrollen der Tabelle funktioniert nicht korrekt

Ist der Mauszeiger über der Tabelle kann es sein, dass das Scrolling nicht korrekt funktioniert. Einfach den Mauszeiger aus der Tabelle bewegen.

### 7.13.2.4 Die Seite/Oberfläche wird nicht richtig angezeigt

Bei der Seite handelt es sich um ein Canvas-Element, das nicht richtig angezeigt wird. Abhilfe schafft ein Neuladen der Seite (F5) und die Verwendung eines Browsers, der HTML5 beherrscht.

### 7.13.2.5 Die erste Nachricht wird nicht richtig übertragen

Sind zwei PiXtend V2 miteinander verbunden, so wird die erste gesendete Nachricht nicht korrekt übertragen, sondern nur das erste Zeichen. Dieses Verhalten kann auftreten, wenn beide PiXtend V2 und Raspberry Pi's gleichzeitig gestartet werden. Die Ursache hierfür ist die Kernel-Meldung „Uncompressing Linux...“ welche immer beim Start auf der seriellen Schnittstelle ausgegeben wird, auch wenn diese deaktiviert ist. Dabei handelt es sich um ein Problem der Raspbian Jessie Version vom 21.11.2015, welches mit der nächsten Version behoben sein sollte.

### 7.13.2.6 Warum gibt es bei RS485 die „Auto send“-Funktion nicht?

Gilt nur für PiXtend V1.2/V1.3 und PiXtend V2 -L-

Diese Funktion ist im Programm vorbereitet, wird aber nicht verwendet. RS485 ist ein Busprotokoll und es darf immer nur ein Teilnehmer zur gleichen Zeit senden. Wird die „Auto send“-Funktion genutzt und mehr als ein Teilnehmer sendet zur gleichen Zeit, kann es zur Kollision kommen und die Daten werden nicht korrekt übertragen.

### 7.13.2.7 Es werden nicht alle Baudraten angezeigt

In diesem Beispiel sind nur einige Baudraten im Bereich von 9600 bis 115200 voreingestellt. Werden andere Baudraten benötigt, müssen diese ebenfalls in das Programm eingepflegt werden (in der Textliste „Baudrates“ sowie im Hauptprogramm in „state 5“).

Wird die Auswahlliste nicht richtig angezeigt, kann eine Änderung des Seitenverhältnisses des Fensters Abhilfe schaffen.

## 8. pxdev – Linux Tools & Library

Diese Anleitung beschreibt alle notwendigen Schritte um pxdev, die Linux Entwicklungs-Tools für das PiXtend V2 System ([www.pixtend.de](http://www.pixtend.de)), zu installieren und in Betrieb zu nehmen. Für eine manuelle Installation siehe Kapitel: 8.4 Manuelle Installation von pxdev.

Alternativ können Sie mit unserem vorbereiteten SD-Karten-Images starten. Diese Möglichkeit empfehlen wir Einsteigern und für die erste Inbetriebnahme von PiXtend V2 – Kapitel: 8.3 Verwendung des PiXtend V2 Basis Images.

**pxdev beinhaltet folgende Komponenten:**

**pixtend** – Die PiXtend/PiXtend V2 C-Library ermöglicht den Zugriff auf PiXtend V2 I/O-Hardware. Der Datenaustausch zwischen Mikrocontroller und Raspberry Pi findet per SPI (Serial Peripheral Interface) statt. Dazu wird die wiringPi-Library verwendet. (<https://projects.drogon.net/raspberry-pi/wiringpi/>)

**pixtendtool2s** – Ein Kommandozeilen-Tool, mit dem auf die PiXtend V2 -S- I/O-Hardware und Konfigurationsbytes mittels einfacher Konsolenbefehle zugegriffen werden kann.

**pxauto2s** – Eine einfache Konsolen-Anwendung mit grafischem User-Interface. PiXtend V2 -S- kann damit kontinuierlich überwacht werden. Die Anwendung eignet sich für die schnelle Inbetriebnahme und I/O-Tests.

**pixtendtool2l** – Ein Kommandozeilen-Tool, mit dem auf die PiXtend V2 -L- I/O-Hardware und Konfigurationsbytes mittels einfacher Konsolenbefehle zugegriffen werden kann.

**pxauto2l** – Eine einfache Konsolen-Anwendung mit grafischem User-Interface. PiXtend V2 -L- lässt sich damit kontinuierlich überwachen. Die Anwendung eignet sich für die schnelle Inbetriebnahme und I/O-Tests.

## 8.1. Hinweise

### 8.1.1. Einsatzbereiche pixtendtool2(s/l) – Einzelbefehle für PiXtend V2

Das pixtendtool2(s/l) bietet direkten Zugriff auf die PiXtend V2 Hardware aus der Linux Kommandozeile. Es eignet sich daher u.a. für folgende Einsatzzwecke:

- Absetzen von Einzelbefehlen über die Kommandozeile: lokal oder via SSH-Fernzugriff über das Netzwerk, um Ausgänge zu bedienen und Eingänge abzufragen
- Aufruf von pixtendtool2(s/l)-Befehlen aus eigenen Shell-Skripten
- Zyklischer Aufruf von pixtendtool2(s/l)-Befehle um Daten periodisch zu loggen und für andere Dienste bereitzustellen, zum Beispiel in einer MySQL-Datenbank.
- Dient als Referenz für die Verwendung der PiXtend/PiXtend V2 C-Library im „manuellen Modus“ (Einzelbefehle)
- Kann an Ihre eigenen Bedürfnisse angepasst werden

### 8.1.2. Einsatzbereiche pxauto2(s/l) – GUI für PiXtend V2

Das pxauto2(s/l)-Tool verfügt über eine grafische Oberfläche und eignet sich für die schnelle Inbetriebnahme Ihrer PiXtend V2 Hardware. Die Zustände der I/Os können zyklisch überwacht werden.

- pxauto2(s/l) kann sowohl lokal als auch „remote“, mit einem SSH-Terminal, verwendet werden
- I/O-Tests der angeschlossen Hardware, Sensoren und Aktoren
- Testen des Funktionsumfangs von PiXtend V2
- Grundlage für eigene Projekte mit GUI (Graphical User Interface)
- Dient als Referenz für die Verwendung der PiXtend/PiXtend V2 C-Library im „Auto-Modus“ (zyklisches Prozessabbild in C verwendbar)

### 8.1.3. Einsatzbereiche PiXtend/PiXtend V2 C-Library

Die PiXtend/PiXtend V2 C-Library wird von den bisher genannten Programmen verwendet und bildet die Schnittstelle zwischen Raspberry Pi und dem auf PiXtend V2 verwendeten Mikrocontroller. Änderungen an der Library sind normalerweise nicht notwendig.

## 8.2. Voraussetzungen

### Benötigte Software

- PiXtend V2 SD-Image „Basis“ (C / Python / Node-RED) (**empfohlen!**)  
Download unter: <https://www.pixtend.de/downloads/>

oder:

- Aktuelle Raspbian Linux-Distribution auf einer SD-Karte
- wiringPi (<https://projects.drogon.net/raspberry-pi/wiringpi/>)
- ncurses Libraries libncurses5-dev libncursesw5-dev
- Optional SSH-Client (z.B. putty.exe – [www.putty.org](http://www.putty.org))

### Benötigte Hardware

PiXtend V2 Board ([www.pixtend.de](http://www.pixtend.de))

Raspberry Pi      Model B+ / 2 B / 3 B / 3 B+ / 4 B

### 8.3. Verwenden des PiXtend V2 Basis Images

Der einfachste Weg zu ersten Tests und weiteren Arbeiten mit pxdev ist der Einsatz unserer vorinstallierten SD-Karte und des SD-Karten-Image. Das jeweils aktuellste Image kann jederzeit kostenlos auf unserer Homepage heruntergeladen werden: <https://www.pixtend.de/downloads/>

Wie Sie das Image auf eine SD-Karte aufspielen können, erfahren Sie im Kapitel 6.5 SD-Karte für Ihren Raspberry Pi vorbereiten

Nach dem Booten wechseln Sie mit folgendem Befehl in den pxdev-Ordner:

```
cd pxdev
```

bzw. in den Ordner des „pixtendtools2s“ Programms:

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

Für PiXtend V2 -L-, in den Ordner des „pixtendtools2l“ Programms:

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

Überspringen Sie das Kapitel 8.4 Manuelle Installation von pxdev und fahren Sie direkt mit der Verwendung der Tools oder der Library in Kapitel 8.5 Verwendung des pictendtools 2(S/l) fort.

## 8.4. Manuelle Installation von pxdev

### 8.4.1. Vorbereitung und Installation

Die aktuellste Version des pxdev-Package kann auf unserer Homepage im Download Bereich als .zip-Archiv heruntergeladen oder mit GIT aus dem pxdev-Repository aus-gecheckt werden. GIT wird zur Versionsverwaltung eingesetzt und ermöglicht, mit einfachen Befehlen komplette Repositorien aus dem Internet herunterzuladen. Wir empfehlen folgende Vorgehensweise:

Geben Sie zunächst folgenden Befehl:

```
sudo apt-get update
```

in der Kommandozeile des Raspberry Pi ein, um die Datenbank des Paket-Managers zu aktualisieren.

Pxdev verwendet intern die wiringPi-Bibliotheken (<https://projects.drogon.net/raspberry-pi/wiringpi/>) als Grundlage.

Installieren Sie die wiringPi Libraries auf Raspbian Buster oder später durch ein Installationspaket wie folgt:

```
cd /tmp
wget https://www.pixtend.de/downloads/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
gpio -v
```

Die wiringPi Bibliothek sollte sich mit Version 2.52 melden.

pxauto2(s/l) verwendet die ncurses Library.

Installieren Sie die benötigten Pakete:

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

Installieren Sie das pxdev-Paket in das Home-Verzeichnis Ihres Raspberry Pi.

```
cd ~
```

Klonen Sie das pxdev-Repository wie folgt:

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

mit dem Befehl ls stellen Sie die Inhalte Ihres Home-Verzeichnisses dar, wechseln Sie in den Ordner pxdev:

```
cd pxdev
```

mit ls -la können Sie die Inhalte des pxdev-Paketes einsehen:

Darin ist ein einfaches build-Skript enthalten, machen Sie das Skript ausführbar

```
chmod +x build
```

Sie können das build-Skript nun starten:

```
./build
```

Nun werden der Reihe nach die PiXtend/PiXtend V2 Library, das pixtendtool2s, pxauto2s-Tool, pixtendtool2l und pxauto2l-Tool erstellt. Das Shell-Skript erstellt zusätzlich die Programme für PiXtend V1.x, lassen Sie diese außer Acht.

## 8.4.2. SPI-Bus aktivieren

Die Kommunikation zwischen dem Raspberry Pi und dem PiXtend V2 Mikrocontroller verwendet SPI. Stellen Sie deshalb sicher, dass das SPI-Modul auf dem Raspberry Pi aktiviert ist. Im Folgenden zeigen wir Ihnen wie der SPI Bus aktiviert wird.

Öffnen Sie raspi-config:

```
sudo raspi-config
```

und aktivieren Sie den SPI-Bus unter „5 Interfacing Options“ → P4 SPI → „Yes“ → „Ok“

Führen Sie im Anschluss, sollte raspi-config dies nicht anbieten, manuell einen Neustart des Raspberry Pi durch.

```
sudo reboot
```

## 8.5. Verwendung des pixtendtool2(s/l)

Die nachfolgenden Anweisungen und Beispiele führen wir anhand des PiXtend V2 -S- durch, sie finden ebenso Anwendung für das PiXtend V2 -L-, sofern das entsprechende Programm verwendet wird. Anstatt „pixtendtool2s“ verwenden Sie dann „pixtendtool2l“.

Wechseln Sie in das Verzeichnis das pixtendtool2s beinhaltet:

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

Geben Sie folgenden Befehle ein:

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

Dieser Befehl gibt alle verfügbaren Aufrufe und Optionen/Parameter aus:

```
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]
-srw1 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]
-pwm1 CHANNEL VALUE	Set PWM 1 CHANNEL[0-1] to VALUE[0-255]
-pwm0c VALUE0 VALUE1	Set PWM 0 control VALUE0[0-255] and VALUE1[0-65535]
-pwm1c VALUE0 VALUE1	Set PWM 1 control VALUE0[0-255] and VALUE1[0-255]
-swc SERIALDEVICE BAUDRATE CHAR	Write a CHAR on SERIALDEVICE
-sws SERIALDEVICE BAUDRATE STRING	Write a STRING on SERIALDEVICE (max 255)
-sr SERIALDEVICE BAUDRATE	Read data from SERIALDEVICE until Ctrl^C
-ucc0 VALUE	Set the microcontroller control 0 to VALUE[0-255]
-ucc1 VALUE	Set the microcontroller control 1 to VALUE[0-255]
-ucs	Get microcontroller status register
-ucr	Reset the microcontroller
-ucv	Get microcontroller version

```
-ucw                                     Get microcontroller warnings

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

Die Ausgabe kann abweichen, wenn es inzwischen eine neue Version des pixtendtool2s gibt. Die hier dargestellte Ausgabe bezieht sich auf die Version 0.5.5.

### 8.5.1. Beispiele

Der Befehl

```
sudo ./pixtendtool2s -di
```

gibt Auskunft über den Zustand des digitalen Eingangs-Bytes zum Zeitpunkt des Aufrufs:

```
sudo pi@raspberrypi:~/pxdev/PiXtend_V2/pixtendtool2s $ sudo ./pixtendtool2s -di
Digital input byte is [0]
```

Abbildung 92: Linux-Tools - pixtendtool2s - Schalter '-di'

```
./pixtendtool2s -do 15
```

Setzen Sie die digitalen Ausgänge auf dez. 15, das repräsentiert binär 00001111 und setzt die Ausgänge 0, 1, 2 und 3.

Um analoge Eingänge abzufragen, verwenden Sie:

```
sudo ./pixtendtool2s -ai 0
```

```
pi@raspberrypi:~/pxdev/PiXtend_V2/pixtendtool2s $ sudo ./pixtendtool2s -ai 0
Analog input 0 value is [516]
```

Abbildung 93: Linux-Tools - pixtendtool2s - Schalter '-ai'

Analog-Eingang 0 liefert uns den Wert 516. Bei einer Referenzspannung von 10V und einer Auflösung von 10Bit ( $2^{10} = 1024$ ) entspricht der Wert also  $10V / 1024 * 516 = 5,04$  Volt.

Serielle Übertragungen per RS232 können mit dem pixtendtool2s sehr einfach aufgerufen werden.

Eine Zeichenfolge wird mit folgendem Befehl über die UART-Schnittstelle des Raspberry Pi versendet:

```
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
```

Abbildung 94: Linux-Tools - pixtendtool2s - Schalter '-sws'

Beim Raspberry Pi 3 B (3 B+) hat die UART-Schnittstelle den Namen „/dev/ttys0“. Hier wird der String (Zeichenfolge) „test123“ mit einer Baudrate von 115.200 baud übertragen.

PiXtend V2 -S- verfügt über eine UART-Schnittstelle, die zu einem RS232-Wandler führt.<sup>1</sup>

---

<sup>1</sup>Das PiXtend V2 -L- verfügt zusätzlich über einen RS485 Wandler, der per GPIO 18 aktiviert werden kann. Die RS232-Schnittstelle steht dann auf dem PiXtend V2 -L- nicht mehr zu Verfügung.

Auf seriellen Schnittstellen kann empfangen / gelesen werden:

```
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-!
```

Abbildung 95: Linux-Tools - pixtendtool2s - Schalter '-sr'

Zusätzliche Hinweise zur UART-Schnittstelle des Raspberry Pi:

Standardmäßig ist die serielle Schnittstelle des Raspberry Pi als Remote-Konsole nutzbar. Soll die Schnittstelle exklusiv für eigene Datenübertragungen eingesetzt werden, muss das Programm raspi-config verwendet werden:

```
sudo raspi-config
```

Wählen Sie:

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

Im Anschluss ist ein Reboot notwendig:

```
sudo reboot
```

### 8.5.2. Weitere Schritte

Experimentieren Sie mit den Optionen und Parametern.

Nachdem Sie mit der Verwendung und den Parametern des pixtendtool2(s/l) vertraut sind, können Sie Ihre eigenen Shell-Skripte schreiben in denen Sie die pixtendtool2(s/l)-Aufrufe verwenden.

Sie können alle Ausgaben von pixtendtool2(s/l) z.B. direkt in eine Text Datei umleiten:

PiXtend V2 -S-:

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

PiXtend V2 -L-:

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

oder Daten an eine bestehende Datei anhängen:

PiXtend V2 -S-:

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

PiXtend V2 -L-:

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

Richten Sie einen cronjob<sup>2</sup> ein, der ein Skript zyklisch aufruft und die Zustände aller analogen Eingänge in einer Datei ablegt.

- Oder ein Skript das die analogen Werte in eine MySQL-Datenbank einträgt.

- Oder ein Skript das täglich zu einer gewissen Uhrzeit ein Relais einschaltet, und zu einer anderen Uhrzeit wieder aus.

Vielleicht fragen Sie sich, wie die Control-Register bedient oder bestimmte Werte verarbeitet werden müssen. Dazu gibt es weitere Informationen im Anhang (Kapitel 14 Anhang mit den entsprechenden Unterkapiteln).

In diesen Kapiteln finden Sie Bit-genau alle Informationen und Möglichkeiten die PiXtend V2 -S- und PiXtend V2 -L- für Sie bereithalten.

---

<sup>2</sup><https://de.wikipedia.org/wiki/Cron>

## 8.6. Verwendung von pxauto2(s/l)

Die nachfolgenden Anweisungen und Beispiele werden anhand des PiXtend V2 -S- durchgeführt. Sie lassen sich ebenso für das PiXtend V2 -L-anwenden, sofern das entsprechende Programm genutzt wird. Anstatt „pxauto2s“ verwenden Sie „pxauto2l“.

Wechseln Sie in das Verzeichnis das pxauto2s beinhaltet mit:

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

Geben Sie den Befehl

```
sudo ./pxauto2s
```

ein, um pxauto2s zu starten.

### NOTICE

Hier ist die Verwendung des „sudo“ Befehls sehr wichtig. Wird pxauto2(s/l) ohne „sudo“ gestartet, kann es zu unerwartetem Verhalten kommen, da eventuell nicht auf die Hardware zugegriffen werden kann. Außerdem könnte die Konsolen-Ausgabe gestört werden.

pxauto2s verfügt momentan über folgende Menüpunkte:

- DIn – Digitale Eingänge
- AIn – Analoge Eingänge/Temperatur/Luftfeuchte – DHT11/22 Einstellung
- GPIO – General Purpose Input/Output
- DOut – Digitale Ausgänge
- AOut – Analoge Ausgänge
- PWM – PWM Ausgänge
- Ctrl – Control-, Debounce und Jumper-Bytes
- Stat – Status Bytes
- RetIn – Retain Input (siehe Kapitel 6.4 Retain-Speicher (dt. Remanenz))
- RetOut – Retain Output (siehe Kapitel 6.4 Retain-Speicher (dt. Remanenz))

Abbildung 96: Linux-Tools - pxauto2s - Hauptmenü

### 8.6.1. Navigation

Nach dem Start von pxauto2(s/l) befinden Sie sich zunächst im Menü (MENU-MODE).

- Mit den Pfeiltasten HOCH und RUNTER können Sie einen anderen Menü-Punkt anwählen.
- Mit RETURN wechseln Sie in das gerade ausgewählte Fenster (EDIT-MODE).
- Farblich hervorgehobene Felder (gelb) können editiert werden indem Sie einen Wert eintippen.
- Um Werte zu löschen nutzen Sie bitte die Keyboard-Tasten DEL/BACKSPACE.
- Boolscche Werte (z.B. digitale Ausgänge) und Auswahlmöglichkeiten können mit den Pfeiltasten LINKS / RECHTS gewählt werden.
- Um Änderungen zu übernehmen verlassen Sie das Feld mit den HOCH/RUNTER Tasten.
- Um wieder ins Menü zu gelangen, drücken Sie erneut RETURN.
- Bei diesem Vorgang wird der Wert im aktuellen Feld automatisch übernommen.

Mit einem Tastendruck auf „q“ im Hauptmenü beenden Sie das Programm.

### 8.6.2. Felder editieren

Die farblich hervorgehobenen Felder (gelb) können verändert werden. Navigieren Sie mit den Pfeil-Tasten zu dem gewünschten Feld und verwenden Sie LINKS/RECHTS, um zwischen den verfügbaren Werten zu wählen oder tippen Sie einen neuen Wert direkt ein.

(Hinweis: Nummernblock wird in der Regel nicht unterstützt bei Zugriff via SSH-Terminal)

Abbildung 97: Linux-Tools - pxauto2s - Menü DOut

## 9. C-Library „pxdev“

Die PiXtend/PiXtend V2 C-Library („pxdev“) ermöglicht es Ihnen eigene Linux-Programme für Ihr PiXtend V2 Board zu programmieren und die Funktionalitäten in Ihre eigenen Programme zu integrieren.

Sie erhalten Zugriff auf alle Eingänge und Ausgänge, die Ihnen mit PiXtend V2 zur Verfügung stehen.

Dieses Kapitel erläutert die dafür notwendigen Schritte. Des Weiteren erhalten Sie eine Übersicht über die Möglichkeiten der C-Library. Am Ende dieses Kapitels finden Sie zwei dokumentierte Beispielprogramme, die Sie testen oder als Grundlage für eigene Entwicklungen verwenden können.

Vielelleicht haben Sie bereits die Programme pxauto2(s/l) und pixtendtool2(s/l) ausprobiert. Diese Programme wurden von Kontron Electronics GmbH mit Hilfe der C-Library erstellt.

Abbildung 98: Linux-Tools - pxauto2s

Abbildung 99: Linux-Tools - pixtendtool2s

## 9.1. Voraussetzungen

Dieses Kapitel setzt voraus, dass Sie die C-Library pxdev, bereits installiert haben. Ist das nicht der Fall, finden Sie im Kapitel 8 pxdev – Linux Tools & Library pxdev – Linux Tools und Library weitere Informationen dazu.

Die einfachste Möglichkeit ist der Einsatz unseres SD-Karten „Basis Image“. Sie können es kostenlos auf unserer Homepage herunterladen. In diesem Image sind pxdev und die benötigten Tools bereits installiert.

Zunächst sollten Sie sich mit dem Kapitel 14 (Anhang) vertraut machen. Wählen Sie die Kapitel entsprechend Ihrem PiXtend V2 aus, Grundkenntnisse in der C-Programmierung setzen wir voraus. Der Umgang mit unterschiedlichen Zahlensystemen (dezimal, binär, hexadezimal) und die Umrechnung zwischen diesen sollte bekannt sein.

Unter <https://sourceforge.net/projects/pixtend/> finden Sie immer die neusten Quelltexte der PiXtend/PiXtend V2 Bibliothek. Programmieren Sie unter Linux, so kann es hilfreich sein an einem PC die Quelltexte darzustellen oder diese auszudrucken. Somit vermeiden Sie ein ständiges Wechseln zwischen Ihrem Programmcode und den Quelltexten von pxdev.

## 9.2. Vorbereitung

Der Datenaustausch zwischen PiXtend V2 und Raspberry Pi erfolgt zyklisch über den SPI-Bus. Dieser Vorgang wird Auto Mode genannt, in Anlehnung an die Funktionen von PiXtend V1.3. Damit ist es möglich alle Ein- und Ausgangswerte auf einmal auszutauschen. Informationen zu SPI und Zykluszeiten finden Sie im Kapitel 6.2 SPI Kommunikation, Datenübertragung und Zykluszeit.

Der Auto Mode bietet folgende Vorteile, einen optionalen Watchdog-Timer und eine Überprüfung auf Übertragungsfehler (CRC-Prüfsumme). Unsere Beispielprogramme pxauto2(s/l) verwenden den Auto Mode.

Wir betrachten die Vorgehensweise des Auto Mode und legen Ordner und Dateien an. Im Kapitel 9.6 Kompilieren und Ausführen des Programms erstellen wir aus unserem Code ein ausführbares Programm.

### Hinweis:

Alle Anweisungen und Beispiele führen wir exemplarisch am PiXtend V2 -S- durch. Es lässt sich ebenso mit einem PiXtend V2 -L- umsetzen. Ersetzen Sie am Ende einer Anweisung oder eines Befehls einfach das „S“ durch ein „L“, um ein PiXtend V2 -L- anzusteuern.

Wir arbeiten immer im Home-Verzeichnis des Users „pi“. Probleme mit Lese- und Schreibberechtigungen, die in anderen Ordnern auftreten können, werden somit ausgeschlossen.

Nach dem Einloggen per SSH oder dem Booten mit der am Raspberry Pi angeschlossenen Tastatur und Maus, landen Sie automatisch in diesem Verzeichnis. Haben Sie das Verzeichnis gewechselt, kehren Sie mit folgendem Befehl zurück:

```
cd /home/pi
```

### 9.2.1. Erstellen eines neuen Ordners und eines C-Files

```
mkdir pixCExample
cd pixCExample
```

Nun erstellen wir ein neues C-File pixCExample und editieren dieses mit dem Editor „nano“:

```
touch pixCExample.c
nano pixCExample.c
```

Der Editor nano startet und Sie sehen, dass die Datei pixCExample.c noch leer ist.

## 9.2.2. Einbinden der Bibliotheken

Nun binden wir die benötigte PiXtend/PiXtend V2 Bibliothek (pxdev) in unser C-File ein:

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

Außerdem sollten Sie folgende Standard-C-Bibliotheken einbinden:

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

Jetzt kann mit dem eigentlichen Programmieren begonnen werden.

## 9.3. Programmierung

In diesem Abschnitt werden die verschiedenen C-Funktionen, die pxdev bietet, vorgestellt und genauer erläutert. Die Namensgebung der in dieser Dokumentation verwendeten Übergabeparameter unterscheiden sich zum Teil vom Header-File der PiXtend/PiXtend V2 Bibliothek. Dies dient ausschließlich zur besseren Übersichtlichkeit und Verständlichkeit.

Die SPI Auto Mode Funktion erlaubt die zyklische und schnelle Abarbeitung der verschiedenen PiXtend/PiXtend V2 C-Funktionen. Mit Hilfe der zyklischen Redundanzprüfung (cyclinc redundancy check - CRC) wird die Übertragung per SPI auf Fehler überprüft. Des Weiteren bietet diese Funktion den Vorteil, dass der Watchdog des PiXtend V2 Mikrocontrollers verwendet werden kann. Bei einem Softwarefehler verhindert dieser einen undefinierten Zustand des Systems. Die Software teilt dem Watchdog in regelmäßigen Abständen mit, dass sie ordnungsgemäß läuft. Meldet sich das laufende Programm nach einer gewissen Zeit nicht zurück, löst der Watchdog einen Reset des Mikrocontrollers aus und das Programm stoppt (siehe Kapitel 6.1 Definition „sicherer Zustand“).

Um die Zykluszeit einzustellen und die dauerhafte Übertragung zwischen Raspberry Pi und PiXtend V2 zu gewährleisten, muss beim Auto Mode eine Endlosschleife („infinity loop“) implementiert werden. Die Schleife wird abhängig von einer Bedingung oder nach einer bestimmten Zeit erneut aufgerufen. Um die Zykluszeit einzustellen, können verschiedene Wartezeiten-Werte verwendet werden. Eine Übersicht zu möglichen Werten finden Sie im Kapitel 6.2 SPI-Kommunikation, Datenübertragung und Zykluszeit.

## Vewendete Funktionen:

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

## Voraussetzungen:

Die Funktion Spi\_SetupV2(0) muss für die Kommunikation mit dem Mikrocontroller aufgerufen werden.

Die Funktion Spi\_SetupV2(1) muss für die Kommunikation mit dem DAC (für analoge Ausgänge) aufgerufen werden.

Das Programm des Mikrocontrollers startet automatisch.

Soll zusätzlich der Watchdog-Timer verwendet werden, muss die Funktion wie folgt aufgerufen werden:

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

Mit der Funktion Spi\_AutoModeV2S (struct OutputData, struct InputData) wird ein kompletter Datenaustausch zwischen PiXtend V2 -S- und Raspberry Pi initiiert. Um die Funktion zu nutzen, muss jeweils eine neue Struktur vom Typ pixtInV2S und pixtOutV2S angelegt werden.

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

### NOTICE

Bei den Ausgangsdaten, hier im Beispiel OutputData, muss immer die Variable OutputData.byModelOut mit dem Wert des zutreffenden PiXtend V2 Modells beschrieben werden. Hierbei handelt es sich um die Dezimalzahlentsprechung der Buchstaben „S“ und „L“ aus der ASCII-Tabelle.

- „S“ entspricht dem Wert „83“ in der ASCII-Tabelle, daher gilt bei PiXtend V2 -S-: **OutputData.byModelOut = 83**
- „L“ entspricht dem Wert „76“ in der ASCII-Tabelle, daher gilt bei PiXtend V2 -L-: **OutputData.byModelOut = 76**

Folgende Variablen stehen in der pixtOutV2S Struktur für das PiXtend V2 -S- zur Verfügung. Für weiter Infos zu den einzelnen Bits und Bytes verweisen wir an dieser Stelle auf die entsprechenden Kapitel im Anhang.

```

OutputData.byModelOut;           //Ein Byte Model als Handshake
OutputData.byUCMode;            //Reserviert, keine Verwendung
OutputData.byUCCtrl0;            //Ein Byte für uC Control 0
OutputData.byUCCtrl1;            //Ein Byte für uC Control 1
OutputData.byDigitalInDebounce01; //Ein Byte für Digital Inputs 0 und 1 debounce
OutputData.byDigitalInDebounce23; //Ein Byte für Digital Inputs 2 und 3 debounce
OutputData.byDigitalInDebounce45; //Ein Byte für Digital Inputs 4 und 5 debounce
OutputData.byDigitalInDebounce67; //Ein Byte für Digital Inputs 6 und 7 debounce
OutputData.byDigitalOut;          //Ein Byte für Setzen der digitalen Ausgänge
OutputData.byRelayOut;            //Ein Byte für Setzen der Relais
OutputData.byGPIOCtrl;           //Ein Byte für GPIO Control
OutputData.byGPIOOut;             //Ein Byte für Setzen der GPIO Outputs
OutputData.byGPIODebounce01;     //Ein Byte für GPIO Inputs 0 und 1 debounce
OutputData.byGPIODebounce23;     //Ein Byte für GPIO Inputs 2 und 3 debounce
OutputData.byPWM0Ctrl0;           //Ein Byte für PWM 0 Control 0
OutputData.wPWM0Ctrl1;            //Zwei Bytes für PWM 0 Control 1
OutputData.wPWM0A;                //Zwei Bytes für PWM 0 Kanal A Value
OutputData.wPWM0B;                //Zwei Bytes für PWM 0 Kanal B Value
OutputData.byPWM1Ctrl0;           //Ein Byte für PWM 1 Control 0
OutputData.byPWM1Ctrl1;           //Ein Byte für PWM 1 Control 1
OutputData.byPWM1A;                //Ein Byte für PWM 1 Kanal A Value
OutputData.byPWM1B;                //Ein Byte für PWM 1 Kanal B Value
OutputData.byJumper10V;            //Ein Byte für Jumperstellung Analog In 0,1
OutputData.byGPIO0Dht11;           //Ein Byte DHT11 / 22 Auswahl für GPIO 0
OutputData.byGPIO1Dht11;           //Ein Byte DHT11 / 22 Auswahl für GPIO 1
OutputData.byGPIO2Dht11;           //Ein Byte DHT11 / 22 Auswahl für GPIO 2
OutputData.byGPIO3Dht11;           //Ein Byte DHT11 / 22 Auswahl für GPIO 3
OutputData.abyRetainDataOut[32];    //Ein Array mit 32 Bytes Retain Data Output

```

Folgende Variablen stehen in der pixtOutV2L Struktur für das PiXtend V2 -L- zur Verfügung. Für weitere Infos zu den einzelnen Bits und Bytes verweisen wir an dieser Stelle auf die entsprechenden Kapitel im Anhang.

```

OutputData.byModelOut;           //Ein Byte Model als Handshake
OutputData.byUCMode;            //Reserviert, keine Verwendung
OutputData.byUCCtrl0;            //Ein Byte für uC Control 0
OutputData.byUCCtrl1;            //Ein Byte für uC Control 1
OutputData.byDigitalInDebounce01; //Ein Byte für Digital Inputs 0 und 1 debounce
OutputData.byDigitalInDebounce23; //Ein Byte für Digital Inputs 2 und 3 debounce
OutputData.byDigitalInDebounce45; //Ein Byte für Digital Inputs 4 und 5 debounce
OutputData.byDigitalInDebounce67; //Ein Byte für Digital Inputs 6 und 7 debounce
OutputData.byDigitalInDebounce89; //Ein Byte für Digital Inputs 8 und 9 debounce
OutputData.byDigitalInDebounce1011; //Ein Byte für Digital Inputs 10 und 11 debounce
OutputData.byDigitalInDebounce1213; //Ein Byte für Digital Inputs 12 und 13 debounce
OutputData.byDigitalInDebounce1415; //Ein Byte für Digital Inputs 14 und 14 debounce
OutputData.byDigitalOut0;         //Ein Byte für Setzen der digitalen Ausgänge 0 - 7
OutputData.byDigitalOut1;         //Ein Byte für Setzen der digitalen Ausgänge 8 - 11
OutputData.byRelayOut;           //Ein Byte für Setzen der Relais
OutputData.byGPIOCtrl;           //Ein Byte für GPIO Control
OutputData.byGPIOOut;            //Ein Byte für Setzen der GPIO Outputs
OutputData.byGPIODebounce01;     //Ein Byte für GPIO Inputs 0 und 1 debounce
OutputData.byGPIODebounce23;     //Ein Byte für GPIO Inputs 2 und 3 debounce
OutputData.byPWM0Ctrl0;          //Ein Byte für PWM 0 Control 0
OutputData.wPWM0Ctrl1;           //Zwei Bytes für PWM 0 Control 1
OutputData.wPWM0A;               //Zwei Bytes für PWM 0 Kanal A Value
OutputData.wPWM0B;               //Zwei Bytes für PWM 0 Kanal B Value
OutputData.byPWM1Ctrl0;          //Ein Byte für PWM 1 Control 0
OutputData.wPWM1Ctrl1;           //Zwei Bytes für PWM 1 Control 1
OutputData.wPWM1A;               //Zwei Bytes für PWM 1 Kanal A Value
OutputData.wPWM1B;               //Zwei Bytes für PWM 1 Kanal B Value
OutputData.byPWM2Ctrl0;          //Ein Byte für PWM 2 Control 0
OutputData.wPWM2Ctrl1;           //Zwei Bytes für PWM 2 Control 1
OutputData.wPWM2A;               //Zwei Bytes für PWM 2 Kanal A Value
OutputData.wPWM2B;               //Zwei Bytes für PWM 2 Kanal B Value
OutputData.byJumper10V;           //Ein Byte für Jumperstellung Analog In 0,1
OutputData.byGPIO0Dht11;          //Ein Byte DHT11 / 22 Auswahl für GPIO 0
OutputData.byGPIO1Dht11;          //Ein Byte DHT11 / 22 Auswahl für GPIO 1
OutputData.byGPIO2Dht11;          //Ein Byte DHT11 / 22 Auswahl für GPIO 2
OutputData.byGPIO3Dht11;          //Ein Byte DHT11 / 22 Auswahl für GPIO 3
OutputData.abyRetainDataOut[64];   //Ein Array mit 64 Bytes Retain Data Output

```

Mit `OutputData.byJumper10V` kann die Jumperstellung der analogen Eingänge AI0 und AI1 auf 5V oder 10V gesetzt werden. Bei kleineren Spannungen als 5V ist es sinnvoll die 5V Jumperstellung zu verwenden, da mit einer höheren Auflösung gemessen werden kann<sup>3</sup>.

Möchte man den AI0 auf 10V setzen, muss die Variable wie folgt befüllt werden:

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

Soll AI1 auf eine Jumperstellung von 10V gesetzt werden, verwendet man:

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

Sollen beide analogen Eingänge auf 10V gesetzt werden:

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

Für die Jumperstellung 5V, müssen die Bits nicht gesetzt oder mit dem Wert „0“ beschrieben werden.

Weitere Informationen finden Sie im Anhang für Ihr jeweiliges PiXtend V2 Gerät.

---

<sup>3</sup>Bitte denken Sie daran vor der Umstellung von 10 V auf 5 V auch tatsächlich den physikalischen Jumper auf dem PiXtend Board zu setzen, sonst erhalten Sie später im Programm falsche Werte.

Die Eingangsdaten von PiXtend V2 -S- können folgenden Variablen der Struktur von pixtlnV2S entnommen werden:

```
InputData.byFirmware; //Ein Byte für die Mikrocontroller Firmwareversion
InputData.byHardware; //Ein Byte für den Hardwarestand
InputData.byModelIn; //Ein Byte Model für Handshake
InputData.byUCState; //Ein Byte für den Mikrocontrollerzustand
InputData.byUCWarnings; //Ein Byte für Warnungen des Mikrocontrollers
InputData.byDigitalIn; //Ein Byte für die digitalen Eingänge
InputData.byGPIOIn; //Ein Byte für die GPIO Eingänge
InputData.wAnalogIn0; //Zwei Byte für analog Eingang 0
InputData.wAnalogIn1; //Zwei Byte für analog Eingang 1
InputData.wTemp0 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO0
InputData.wTemp1 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO1
InputData.wTemp2 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO2
InputData.wTemp3 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO3
InputData.wHumid0 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO0
InputData.wHumid1 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO1
InputData.wHumid2 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO2
InputData.wHumid3 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO3
InputData.rAnalogIn0; //float Variable für Spannung analog Eingang 0
InputData.rAnalogIn1; //float Variable für Spannung analog Eingang 1
InputData.rTemp0; //float Variable für Temperatur an GPIO0
InputData.rTemp1; //float Variable für Temperatur an GPIO1
InputData.rTemp2; //float Variable für Temperatur an GPIO2
InputData.rTemp3; //float Variable für Temperatur an GPIO3
InputData.rHumid0; //float Variable für Luftfeuchte an GPIO0
InputData.rHumid1; //float Variable für Luftfeuchte an GPIO1
InputData.rHumid2; //float Variable für Luftfeuchte an GPIO2
InputData.rHumid3; //float Variable für Luftfeuchte an GPIO3
InputData.abyRetainDataIn[32]; //Array mit 32 Bytes Retain Data Input
```

Die Eingangsdaten von PiXtend V2 -L- können folgenden Variablen der Struktur von pixtlnV2L entnommen werden:

```
InputData.byFirmware; //Ein Byte für die Mikrocontroller Firmwareversion
InputData.byHardware; //Ein Byte für den Hardwarestand
InputData.byModelIn; //Ein Byte Model für Handshake
InputData.byUCState; //Ein Byte für den Mikrocontrollerzustand
InputData.byUCWarnings; //Ein Byte für Warnungen des Mikrocontrollers
InputData.byDigitalIn0; //Ein Byte für die digitalen Eingänge 0 - 7
InputData.byDigitalIn1; //Ein Byte für die digitalen Eingänge 8 - 15
InputData.byGPIOIn; //Ein Byte für die GPIO Eingänge
InputData.wAnalogIn0; //Zwei Byte für analog Eingang 0
InputData.wAnalogIn1; //Zwei Byte für analog Eingang 1
InputData.wAnalogIn2; //Zwei Byte für analog Eingang 2
InputData.wAnalogIn3; //Zwei Byte für analog Eingang 3
InputData.wAnalogIn4; //Zwei Byte für analog Eingang 4
InputData.wAnalogIn5; //Zwei Byte für analog Eingang 5
InputData.wTemp0 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO0
InputData.wTemp1 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO1
InputData.wTemp2 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO2
InputData.wTemp3 //Zwei Bytes für Temperatur Wert DHT11/22 an GPIO3
InputData.wHumid0 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO0
InputData.wHumid1 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO1
InputData.wHumid2 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO2
InputData.wHumid3 //Zwei Bytes für Luftfeuchte Wert DHT11/22 an GPIO3
InputData.rAnalogIn0; //float Variable für Spannung analog Eingang 0
InputData.rAnalogIn1; //float Variable für Spannung analog Eingang 1
InputData.rAnalogIn2; //float Variable für Spannung analog Eingang 2
InputData.rAnalogIn3; //float Variable für Spannung analog Eingang 3
InputData.rAnalogIn4; //float Variable für Strom analog Eingang 4
InputData.rAnalogIn5; //float Variable für Strom analog Eingang 5
InputData.rTemp0; //float Variable für Temperatur an GPIO0
InputData.rTemp1; //float Variable für Temperatur an GPIO1
InputData.rTemp2; //float Variable für Temperatur an GPIO2
InputData.rTemp3; //float Variable für Temperatur an GPIO3
InputData.rHumid0; //float Variable für Luftfeuchte an GPIO0
InputData.rHumid1; //float Variable für Luftfeuchte an GPIO1
InputData.rHumid2; //float Variable für Luftfeuchte an GPIO2
InputData.rHumid3; //float Variable für Luftfeuchte an GPIO3
InputData.abyRetainDataIn[64]; //Array mit 64 Bytes Retain Data Input
```

Um die Input-und Output Strukturen der Funktion zu übergeben, wird folgende Zeile verwendet:

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

Die Struktur OutputData enthält die Werte für die Control-Bytes und Prozessdaten (Ausgangswerte aus Sicht des Raspberry Pi hin zu PiXtend V2 -S-).

In die Struktur InputData werden die Werte der Eingänge während des Programmablaufs geschrieben (vom PiXtend V2 -S- Mikrocontroller).

#### Ein kleines Beispiel:

Mit folgendem Code kann die Temperatur über GPIO0 von einem DHT Sensor ausgelesen und auf der Konsole ausgegeben werden:

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

Die Funktion Spi\_AutoModeDAC(struct OutputDataDAC) wird ausschließlich für die Kommunikation mit dem Digital-Analog-Converter (DAC) verwendet. Die Funktion verlangt als Übergabeparameter eine Struktur vom Typ pixtOutDAC. Diese kann folgendermaßen erstellt werden:

```
struct pixtOutDAC OutputDataDAC
```

Die Struktur muss die Werte für die beiden analogen Ausgänge beinhalten.

```
OutputDataDAC.wAOut0 = 1023; //1023 entspricht dem Maximalwert, 0 Minimalwert
OutputDataDAC.wAOut1 = 1023;
```

Hier werden die analogen Ausgänge auf 100% gesetzt.

Mit dem folgenden Aufruf übergeben wir die Struktur OutputDataDAC der Funktion.

```
Spi_AutoModeDAC (&OutputDataDAC);
```

## 9.4. Beispielprogramm für PiXtend V2 -S-

```

//Include the librarys
#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, PiXtend V2 -S- = 83
    OutputData.byGPIOCtrl = 32; //GPIO1 is used with DHT22 sensor
    OutputDataDAC.wAOut0 = 1023; //pre-load analog outputs
    OutputDataDAC.wAOut1 = 1023;

    //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. Beispielprogramm für PiXtend V2 -L-

```

//Include the librarys
#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;

    //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. Kompilieren und Ausführen des Programms

Um das Programm auszuführen, muss es zunächst kompiliert werden, geben Sie folgende Zeile ein:

```
sudo gcc -Wall -o "pixCExample" "pixCExample.c" -lpixtend -lwiringPi
```

Nun wird pixCExample.c kompiliert und in pixCExample gespeichert.

Mit dem Befehl:

```
sudo ./pixCExample
```

wird das Programm ausgeführt.

Wer sudo nicht verwenden möchte, kann der kompilierten Datei pixCExample die benötigten Rechte geben. Verwenden Sie dazu folgende Zeile:

```
sudo chmod -R 0777 pixCExample
```

Nun können Sie sudo<sup>4</sup> beim Ausführen des C-Programms weglassen.

Zum Beenden eines Programms verwenden Sie diese Tastenkombination STRG + C (Ctrl + C).

---

<sup>4</sup>sudo wird hier überhaupt erst notwendig, weil die SPI-Schnittstelle über die wiringPi-Library (<http://wiringpi.com/>) angesprochen wird. Dieser Zugriff erfordert je nach Version immer noch Superuser-Rechte.

## 9.7. Frequently Asked Questions (FAQ)

**Was muss besonders beachtet werden, wenn die PiXtend / PiXtend V2 C-Library verwendet wird?**

Zuerst muss die SPI-Schnittstelle, mit der Funktion Spi\_SetupV2(int device), konfiguriert werden. Die Spi\_SetupV2 Funktion darf maximal zweimal (einmal für device 0, einmal für device 1) aufgerufen werden. Lagern Sie die Aufrufe in eine Initialisierungsroutine aus, die nur einmal ausgeführt wird (direkt nach dem Start des Programms).

Über das Output Byte byModelOut muss PiXtend V2 der Handshake mitgeteilt werden, dieser ist für das PiXtend V2 -S- immer die Dezimalzahl 83 und für das PiXtend V2 -L- die Dezimalzahl 76.

**Es treten immer wieder Übertragungsproblemen zwischen PiXtend und Raspberry Pi auf. Was ist zu tun?**

Überprüfen Sie, ob Sie die Funktion Spi\_SetupV2(int device) mehrfach oder zyklisch in ihrem Programm aufrufen wird. Wenn ja, dann ist das der Grund für die Übertragungsfehler. Findet keine Übertragung statt, überprüfen Sie ob der Schalter SPI\_EN auf dem PiXtend V2 auf ON steht. Werden DHT-Sensoren verwendet, muss die Zykluszeit mindestens 30 ms sein. Die Zykluszeit sollte nicht weniger als 2,5 ms (PiXtend V2 -S-) bzw. 5 ms (PiXtend V2 -L-) betragen.

**Wo wird die Zykluszeit eingestellt?**

Bei C-Programmen, unter Verwendung der PiXtend Library, muss sich der Anwender um die Zykluszeit kümmern. Im nachfolgenden Beispiel wird der PiXtend V2 -L- AutoMode in einer Endlosschleife alle 100 ms aufgerufen und am Ende eines Durchlaufs wird die Programmausführung für 100 ms angehalten, erst dann geht es weiter.

```
//Auto Mode in infinity loop
while(1)
{
    //Do the data transfer!
    Spi_AutoModeV2L(&OutputData, &InputData);
    Spi_AutoModeDAC(&OutputDataDAC);

    //Take a 100 ms break between the cycles
    delay(100);
}
return 0;
```

Abbildung 100: C-Programm - 100ms Zykluszeit

## 10. FHEM

Bei FHEM handelt es sich um ein Perl-basiertes Serverprogramm für die Hausautomatisierung. Das ermöglicht dem Anwender Aktoren und Sensoren, beispielsweise Schalter, Lampen, Heizungen oder Temperatur- und Luftfeuchtigkeitssensoren in ein gemeinsames System zu integrieren und Vorgänge zu automatisieren.

Durch ein FHEM-Modul erhalten Sie innerhalb der gewohnten FHEM-Umgebung vollen Zugriff auf die Funktionen von PiXtend V2.

Bei dem von Kontron Electronics GmbH entwickelten FHEM-Modul für den PiXtend V2 handelt es sich um ein eigenständiges Modul. Es wurde für den Raspberry Pi entwickelt und besitzt keine Softwareabhängigkeiten zu anderen Modulen oder Hilfsprogrammen.

### 10.1. Voraussetzungen

Für die Installation von FHEM auf der PiXtend V2-Steuerung werden grundlegende Kenntnisse im Umgang mit dem Raspberry Pi oder seinem Linux-Betriebssystem vorausgesetzt. Zudem werden neben einem PiXtend V2 mit Raspberry Pi folgende Punkte benötigt:

- SD-Karte mit einem aktuellen Raspbian Image. Vgl. Kapitel 6.5 SD-Karte für Ihren Raspberry Pi SD-Karte für Ihren Raspberry Pi vorbereiten zur Erstellung einer solchen SD-Karte.
- SSH-Client-Software (z.B. putty.exe – [www.putty.org](http://www.putty.org))
- FTP-Client-Software (z.B. WinSCP – [www.winscp.net](http://www.winscp.net))

Für die anschließende Anwendung ist es sinnvoll, sich einen Überblick über die zur Verfügung stehenden Funktionen und Möglichkeiten von PiXtend V2 zu machen.

**NOTICE**

Beachten Sie die Hinweise zur Kompatibilität von Software-Komponenten in Abschnitt 6.10 Kompatibilität der Software-Komponenten.

## 10.2. Installation

Für die Installation von FHEM und dem FHEM-Modul für PiXtend V2 müssen folgende Schritte durchlaufen werden. Sollte FHEM bereits auf dem PiXtend V2-Board installiert sein, kann der erste Schritt übersprungen und direkt mit der Integration des Moduls begonnen werden.

### 10.2.1. Installation von FHEM

Die jeweils aktuelle Anleitung zur Installation von FHEM auf dem Raspberry Pi ist unter folgendem Link zu finden: <https://debian.fhem.de/>.

Es wird empfohlen, die Installation nach „The easy way: use apt-get“ (im linken Menü aufgeführt) auszuführen. Für den Stand (21.09.2017 – Rev 5.8.15110) sieht die Installation wie folgt aus.

Der Repository-Key für FHEM wird durch folgendes Kommando aktualisiert:

```
wget -qO - http://debian.fhem.de/archive.key | sudo apt-key add -
```

Nun muss die Adresse des Repository in die Datei `sources.list` eingetragen werden. Öffnen Sie die Datei mit folgendem Befehl:

```
sudo nano /etc/apt/sources.list
```

und fügen Sie die nächste Zeile ans Ende ein. Speichern Sie die Datei (Strg+x und mit y und anschließend Enter bestätigen). Der Eintrag wird bei der Installation von FHEM automatisch wieder gelöscht.

```
deb http://debian.fhem.de/nightly/ /
```

Danach müssen die installierten Pakete mit dem neuen Repository aktualisiert werden:

```
sudo apt-get update
```

Im Anschluss kann FHEM installiert werden (die Nachfrage ob FHEM installiert werden soll mit „y“ bestätigen):

```
sudo apt-get install fhem
```

### 10.2.2. Integration des Moduls

Um das FHEM-Modul für PiXtend V2 in die FHEM-Umgebung zu integrieren muss das Modul in das FHEM-Verzeichnis kopiert werden. Das Modul finden Sie als ZIP-Datei im Download-Bereich unserer Website. Da das FHEM-Verzeichnis dem Benutzer „fhem“ gehört, können ohne Rechte keine Dateien abgelegt werden. Aus diesem Grund müssen den Benutzern der Gruppe vorübergehend die nötigen Rechte erteilt werden:

```
sudo chmod g+w /opt/fhem/FHEM
```

Im Anschluss kann das Modul „97\_PiXtendV2.pm“ mit Hilfe eines FTP-Clients in das Verzeichnis „/opt/fhem/FHEM“ auf den Raspberry Pi kopiert werden. Am Ende sollten die Rechte der Gruppe wieder zurückgesetzt werden:

```
sudo chmod g-w /opt/fhem/FHEM
```

### 10.2.3. Anpassung des Systems

Der Daten-Austausch zwischen dem Raspberry Pi und dem PiXtend V2-Board erfolgt über die SPI-Schnittstelle. Darum muss die Schnittstelle in den Einstellungen des Raspberry Pi mit diesem Befehl aktiviert werden:

```
sudo raspi-config
```

Sie öffnen das Menü und aktivieren die Schnittstelle unter „5 Interfacing Options“ mit Punkt „P4 SPI“. Verlassen Sie das Menü anschließend wieder.

Der Benutzer „fhem“ hat standardmäßig keine Rechte für die SPI-Schnittstelle und die GPIOs. Benötigt er diese werden die Rechte der Gruppen hinzugefügt:

```
sudo adduser fhem gpio; sudo adduser fhem spi; sudo reboot
```

Durch diese Zeile wird ebenso ein benötigter Neustart durchgeführt.

Nachdem Neustart des Raspberry Pi, kann im Browser über folgende Adresse die Seite der FHEM-Oberfläche des Raspberry Pi aufrufen werden:

```
http://<RaspberryIP>:8083/fhem
```

ersetzen Sie <RaspberryIP> durch die von Ihnen vorgegebene IP-Adresse. Die aktuell eingestellte IP-Adresse können Sie mit folgendem Befehl auslesen:

```
ifconfig
```

Die Installation von FHEM und die Integration des Moduls sind nun abgeschlossen und Sie können mit Ihrer Hausautomation beginnen.

### 10.3. Modulbeschreibung

Das PiXtend V2-Modul für FHEM verfügt über eine Vielzahl an Funktionen. Nachstehend wird beschrieben, wie ein Modul angelegt und die Funktionen genutzt werden können. Die Funktionen lassen sich über das FHEM-Web-Interface des Moduls oder über die Eingabe des entsprechenden Befehls in die FHEM-Befehlszeile ausführen.

#### 10.3.1. Anlegen eines neuen PiXtend V2-Geräts

Ein neues PiXtend V2 -S--Modul wird durch die Eingabe des Befehls

```
define <name> PiXtendV2
```

Ein neues PiXtend V2 -L--Modul wird durch die Eingabe des Befehls

```
define <name> PiXtendV2 L
```

in der Befehlszeile von FHEM angelegt. Ersetzen Sie <name> durch einen beliebigen Namen.

Wir raten davon ab, mehrere PiXtend V2-Module in FHEM für ein tatsächlich physisch existierendes Gerät anzulegen. Es macht keinen Sinn und kann zu Problemen führen, da von mehreren Stellen auf die gleichen Hardware-Funktionen zugegriffen wird.

#### 10.3.2. Internals

Die vom Modul veränderbaren Internals sind zum einen der Wert „RetainSize“, der die Größe des Retain-Speichers in Bytes angibt und zum anderen das Feld „STATE“. Dieses Feld kann die Werte „defined“, „active“ oder „error“ annehmen. Der Wert „defined“ wird nur kurz nach dem Anlegen des Moduls angezeigt und geht dann sofort in „active“ über. Tritt ein Fehler auf, wechselt der Status in „error“ und es wird zusätzlich im Feld „ErrorMsg“ die Information zur Fehlerursache angezeigt. Meistens liegt es an einer fehlerhaften Kommunikation zwischen PiXtend V2 und dem Raspberry Pi, wenn sich PiXtend V2 beispielsweise im sicheren Zustand befindet.

#### 10.3.3. Set-Kommandos

Das PiXtend V2-Modul verfügt über mehrere Set-Befehle, mit denen sich die digitalen Ausgänge oder die Relays steuern lassen. Kommandos, um die Basiskonfiguration für den PiXtend durchzuführen, beginnen mit einem Unterstrich (\_) und können im Attribut „PiXtend\_Parameter“ hinterlegt werden.

Unterstützt ein Kommando mehrere Kanäle, wird das Raute-Zeichen (#) durch die Kanal-Nummer ersetzt. Alle Set-Kommandos können unabhängig von der Groß- und Kleinschreibung verwendet werden.

**\_GPIO#Ctrl [input, output, DHT11, DHT22]**

Mit dieser Einstellung wird die Funktion des GPIO eingestellt. [input], [output] oder [DHT11] und [DHT22] wenn ein DHT-Sensor an den GPIO angeschlossen ist. Ist ein DHT-Sensor angeschlossen und wird verwendet, lässt sich die normale Funktion des GPIO nicht gleichzeitig als Eingang/Ausgang nutzen.

**\_GPIOPullupsEnable [yes, no]**

Diese Einstellung aktiviert [yes] oder deaktiviert [no] für alle GPIOs die Möglichkeit, die internen PullUp-Widerstände durch „GPIOOut“ zu setzen.

**\_JumperSettingAI# [5V, 10V]**

Diese Einstellung beeinflusst die Berechnung der Spannung an den analogen Eingängen und bezieht sich dabei auf die tatsächliche Position des Jumpers auf dem PiXtend V2-Board [5V,10V]. Wird kein Jumper verwendet, entspricht das der Standardeinstellung von [10V].

**\_StateLEDDisable [yes, no]**

Diese Einstellung deaktiviert [yes] oder aktiviert [no] die Status-LED auf dem PiXtend V2-Board. Ist die LED deaktiviert, leuchtet sie im Fehlerfall nicht auf.

**WatchdogEnable [disable,125ms,1s,8s]**

Diese Einstellung ermöglicht die Konfiguration des Watchdog-Timers. Ist der Watchdog konfiguriert, geht PiXtend V2 in den sicheren Zustand über, sofern innerhalb der eingestellten Zeit keine gültige Übertragung zwischen PiXtend V2 und Raspberry Pi stattgefunden hat. Eine erneute Kommunikation ist erst wieder möglich, nachdem PiXtend V2 zurückgesetzt wurde („Reset“).

**AnalogOut# []**

Stellt am analogen Ausgang eine Spannung ein. Der übergebene Wert kann eine Spannung zwischen 0V und 10V oder ein Rohwert zwischen 0 und 1023 sein. Um den Wert als Spannung zu übergeben, muss der Wert ein ":" enthalten, auch wenn der Wert ganzzahlig ist.

**DigitalDebounce# [0-255]**

Ermöglicht das Entprellen der digitalen Eingänge. Die Einstellung beeinflusst immer zwei Kanäle, DigitalDebounce01 beeinflusst DigitalIn0 und DigitalIn1. Die resultierende Verzögerung berechnet sich durch (eingestellten Wert)\*(100 ms). Der übergebene Wert kann eine beliebige Zahl zwischen 0 und 255 sein. Entprellen ist sinnvoll, wenn an den Eingängen Schalter oder Taster angeschlossen sind.

**DigitalOut# [on, off, toggle]**

Setzt den digitalen Ausgang auf HIGH [on] oder LOW [off] oder [toggle]t ihn.

**GPIODebounce# [0-255]**

Ermöglicht das Entprellen der GPIO Eingänge. Die Einstellung beeinflusst immer zwei Kanäle, GPIODebounce01 beeinflusst somit GPIOIn0 und GPIOIn1. Die resultierende Verzögerung berechnet sich durch (eingestellten Wert)\*(100 ms). Der übergebene Wert kann eine beliebige Zahl zwischen 0 und 255 sein. Entprellen kann sinnvoll sein, falls an den Eingängen Schalter oder Taster angeschlossen sind.

**GPIOOut# [on, off, toggle]**

Setzt den GPIO auf HIGH [on] oder LOW [off] oder [toggle]t ihn, falls er als Ausgang konfiguriert ist. Ist der GPIO als Eingang konfiguriert, kann mit diesem Kommando der interne PullUp-Widerstand aktiviert [on], deaktiviert [off] oder ge[toggle]t werden. Dazu muss die Möglichkeit allerdings global durch „\_GPIOPullupsEnable“ aktiviert werden.

**PWM**

PiXtendV2 unterstützt mehrere PWM-Modi, die mit den Kommandos, beginnend mit „PWM“ konfiguriert werden können. Die genaue Bedeutung der Werte und wie die PWM-Modi eingestellt werden kann in Kapitel 14 nachgelesen werden.

**RelayOut# [on, off, toggle]**

Setzt das Relay auf HIGH [on] oder LOW [off] oder [toggle]t es.

**Reset**

Setzt den PiXtend V2-Controller zurück, zum Beispiel wenn er sich im sicheren Zustand befindet, um ihn erneut zu konfigurieren.

**RetainCopy [on, off]**

Wenn RetainCopy aktiviert [on] ist, werden die geschriebenen Daten RetainDataOut von PiXtend V2 in RetainDataIn zurückgegeben. Die Aktivierung ist sinnvoll, um zu überprüfen, welche Daten an PiXtend V2 geschickt wurden. Ist die Funktion deaktiviert [off] werden die zuletzt gespeicherten Daten in RetainDataIn zurückgegeben.

**RetainDataOut [0-(RetainSize-1)] [0-255]**

PiXtend V2 unterstützt die Speicherung remanenter/persistenter Daten – auch Retain genannt. Die Retain-Daten sind in Bytes organisiert, jedes Byte kann individuell mit einem Wert zwischen 0 und 255 beschrieben werden. Als ersten Parameter erwartet das Kommando den Index des Bytes, der zwischen 0 und (RetainSize-1) liegt. RetainSize finden Sie in den "Internals". Als zweiter Parameter wird der zu speichernde Wert erwartet.

**RetainEnable [on, off]**

Die Funktion um Retain-Daten auf PiXtend V2 zu speichern muss aktiviert [on] werden, sonst [off] ist keine Speicherung der Daten möglich. Bitte beachten Sie, für den Retain-Speicherbereich werden 10.000 Schreibzyklen unterstützt. Aktivieren Sie diese Funktion nur, wenn sie tatsächlich benötigt wird.

**SafeState**

Mit dieser Einstellung kann PiXtend V2 in den sicheren Zustand versetzt werden. Ist die Retain-Speicherung aktiviert, werden die Daten gesichert. Im sicheren Zustand kommuniziert PiXtend V2 nicht mit dem Raspberry Pi und somit nicht mit FHEM. Für einen PiXtend V2 Neustart muss ein Reset durchgeführt werden.

### 10.3.4. Get-Kommandos

Durch Get-Kommandos können die Readings und einige Systemzustände abgefragt werden. Das Format des Rückgabewertes kann durch ein Attribut („PiXtend\_GetFormat“) auf einen Text oder den „reinen“ Wert eingestellt werden. Ist die Ausgabe als Text aktiv, befindet sich der Wert in eckigen Klammern. Unterstützt ein Kommando mehrere Kanäle, wird das Raute-Zeichen (#) durch die Kanal-Nummer ersetzt. Alle Get-Kommandos können unabhängig von der Groß- und Kleinschreibung verwendet werden.

**AnalogIn#**

Gibt den Wert des ausgewählten analogen Eingangs zurück. Der Wert hängt von der Einstellung \_JumperSettingAI# und der tatsächlichen Jumper-Position auf dem Board ab.

**DigitalIn#**

Gibt den Status on (HIGH) oder off (LOW) des digitalen Eingangs zurück.

**GPIOIn#**

Gibt den Status on (HIGH) oder off (LOW) des GPIOs zurück, unabhängig von der Konfiguration (input, output).

**RetainDataIn [0-(RetainSize-1)]**

Gibt den Wert des ausgewählten RetainDataIn-Bytes zurück.

**Sensor# [temperature, humidity]**

Ist ein DHT-Sensor an den entsprechenden GPIO angeschlossen und \_GPIO#Ctrl auf DHT11 oder DHT22 eingestellt, wird die Temperatur und Luftfeuchtigkeit gemessen und kann ausgelesen werden.

**SysState**

Gibt den Systemstatus [defined, active, error] des FHEM-Moduls zurück.

**UCState**

Gibt den Status von PiXtend V2 zurück. Ist der Status 1, ist alles in Ordnung. Ist der Status allerdings größer als 1 ist ein Fehler aufgetreten oder steht noch an. In diesem Fall kann PiXtend V2 nicht konfiguriert werden. Die genaue Bedeutung des Wertes kann in Kapitel 14.1.2.4 und 14.2.2.4 nachgelesen werden.

**UCWarnings**

Der zurückgegebene Wert repräsentiert die Warnungen von PiXtend V2. Die genaue Bedeutung des Wertes kann in Kapitel 14.1.2.4 und 14.2.2.4 nachgelesen werden.

**Version**

Gibt die Version des FHEM-Moduls sowie die PiXtend-Version [Model-Hardware-Firmware] zurück.

### 10.3.5. Readings

Bei Readings handelt es sich um von der Steuerung zur Verfügung gestellte Werte, wie Sensormesswerte oder Zustände der Eingänge. Die Bedeutung der Readings entspricht denen der Get-Kommandos. Die meisten Readings lösen ein Ereignis (Event) aus. Ändern Sie sich, kann in FHEM eine Reaktion ausgeführt werden.

**AnalogIn#**

Zeigt das Ergebnis der Messungen der analogen Eingänge in Volt an.

**DigitalIn#**

Zeigt den Status on (HIGH) oder off (LOW) der digitalen Eingänge an.

**Firmware**

Zeigt die Firmware-Version an.

**GPIOIn#**

Zeigt den Status on (HIGH) oder off (LOW) der GPIOs, unabhängig von deren Konfiguration (input, output).

**Hardware**

Zeigt die Hardware-Version an.

**Model**

Zeigt das Model an.

**RetainDataIn**

Zeigt die Werte der empfangenen RetainDataIn-Daten an. Die Werte von RetainDataIn sind in einer Zeile zusammengefasst. Der Wert, am weitesten links steht, entspricht Byte0/RetainDataIn0. Die Werte, durch ein Leerzeichen voneinander getrennt, können bei Bedarf durch folgende Funktion in Perl ausgewertet werden:

```
my ($str) = ReadingsVal(pix, "RetainDataIn", "?")
if($str ne "?") {
    my @val = split(/ /, $str);      => $val[0] enthält nun Byte0,
                                         $val[1] Byte1, usw
    ...
}
```

**Sensor#T/H**

Zeigt die Temperatur (T) in °C und die Luftfeuchtigkeit (H) in % des Sensors an, der an den entsprechenden GPIO angeschlossen ist.

**UCState**

Zeigt den Status von PiXtend V2 an. Ist der Status 1, ist alles in Ordnung. Ist der Status allerdings größer als 1 ist ein Fehler aufgetreten oder steht noch an. In diesem Fall kann PiXtend V2 nicht konfiguriert werden. Die genaue Bedeutung des Wertes kann in Kapitel 14.1.2.4 und 14.2.2.4 nachgelesen werden.

**UCWarnings**

Der angezeigte Wert repräsentiert die Warnungen von PiXtend V2.

### 10.3.6. Attribute

Für den Attribut-Namen muss die Groß- und Kleinschreibung beachtet werden.

#### PiXtend\_GetFormat [text,value]

Ändert die Darstellung, wie die Werte durch die Get-Kommandos zurückgegeben werden. Die Werte können entweder in einer Nachricht [text] oder als rohe Werte [value] zurückgegeben werden. Standard ist die Ausgabe als Text.

#### PiXtend\_Parameter

Dieses Attribut lässt sich verwenden, um die Einstellungen zur Basiskonfiguration (Set-Kommandos beginnend mit "\_") als Attribut zu speichern. Attribute werden im Gegensatz zu Set-Kommandos in der Config-Datei von FHEM gespeichert und beim Start automatisch geladen. Der Wert eines jeden Kommandos wird nach einem Doppelpunkt übergeben und mehrere Kommandos werden durch ein Leerzeichen getrennt, zum Beispiel:

```
attr pix PiXtend_Parameter _gpio0ctrl:dht11 _gpio3ctrl:dht22
```

### 10.4. Beispiele

Als Ausgangspunkt für die folgenden Beispiele dient immer ein angelegtes PiXtend V2 -S- oder PiXtend V2 -L-- Gerät in FHEM, mit dem Namen „pix“. Das Gerät wird über folgendes Kommando angelegt:

```
define pix PiXtendV2
```

oder

```
define pix PiXtendV2 L
```

### 10.4.1. Darstellen von Sensorwerten

In FHEM werden Werte in einem Diagramm (Plot) dargestellt, an PiXtend V2 angeschlossene Sensoren können somit grafisch aufbereitet werden. In diesem Beispiel wird an den GPIO0 ein DHT11 und an GPIO3 ein DHT22 Temperatur- und Luftfeuchtigkeitssensor angeschlossen. Die Spannungsversorgung erfolgt über eine an den Klemmen bereitgestellte 5 V Leitung.

Um die Sensorfunktion zu aktivieren, wird das Gerät „pix“ in FHEM geöffnet. Anschließend wird im Set-Bereich das Feld „\_GPIO0Ctrl“ und im zweiten Feld „DHT11“ ausgewählt und durch Anklicken des set-Knopfes bestätigt. Alternativ kann das folgende Kommando direkt in die Kommandozeile eingegeben werden (vgl. Abbildung 101):

```
set pix _gpio0ctrl dht11
```

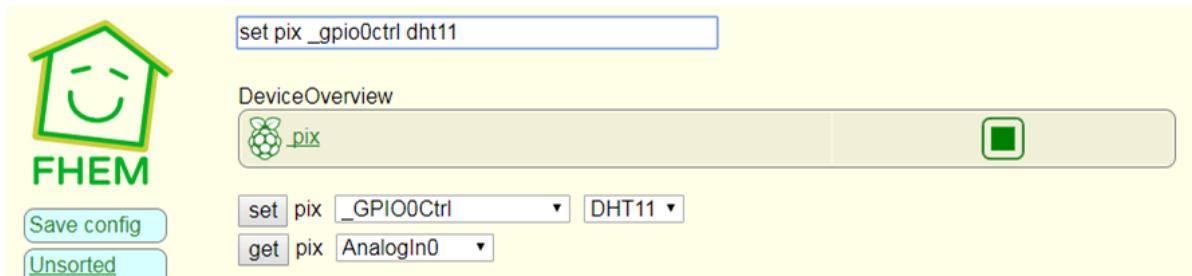


Abbildung 101: FHEM - Aktivieren der Sensorfunktion

Dieser Schritt wird mit dem DHT22-Sensor an GPIO3 wiederholt.

Zusätzlich besteht die Möglichkeit, die Konfiguration der GPIOs (und weiterer Funktionen) direkt beim Start automatisch von FHEM laden zu lassen. Hierzu sollten die Kommandos im Attribut „PiXtend\_Parameter“ hinterlegt werden. In der Befehlszeile werden die Kommandos wie folgt hinterlegt:

```
attr pix PiXtend_Parameter _gpio0ctrl:dht11 _gpio3ctrl:dht22
```

Der Wert eines Kommandos wird nach einem Doppelpunkt (:) übergeben und mehrere Kommandos werden durch ein Leerzeichen getrennt. Mit dieser Methode können alle Set-Befehle, die mit einem Unterstrich (\_) beginnen im FHEM-Start hinterlegt werden.

Nach der Konfiguration werden nun im Bereich „Readings“ des Geräts „pix“ die gemessenen Werte für Sensor0 und Sensor3 angezeigt.

Die Darstellung der gemessenen Werte erfolgt größtenteils nach diesem Tutorial:

[https://wiki.fhem.de/wiki/Buderus\\_Web\\_Gateway#Mit\\_FileLog](https://wiki.fhem.de/wiki/Buderus_Web_Gateway#Mit_FileLog)

Im ersten Schritt wird einLogFile erzeugt, es werden nur die benötigten gemessenen Werte aus den Readings gespeichert:

```
define pixlog FileLog ./log/pix-%Y-%m.log
pix:Sensor0H:.*|pix:Sensor0T:.*|pix:Sensor3H:.*|pix:Sensor3T:.*
```

Im zweiten Schritt wird in diesemLogFile ein SVG Plot erstellt, indem auf die entsprechende Schaltfläche geklickt wird (vgl. Abbildung 102).

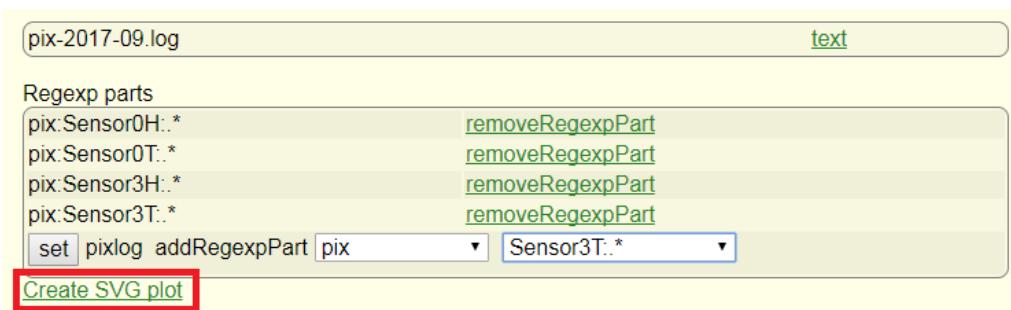


Abbildung 102: FHEM - Plot durchLogFile erstellen

Das Diagramm kann wie in Abbildung 103 dargestellt, eingerichtet werden. Wurden alle Einstellungen getätigt, muss im Anschluss der Button „write .gplot file“ angeklickt werden.

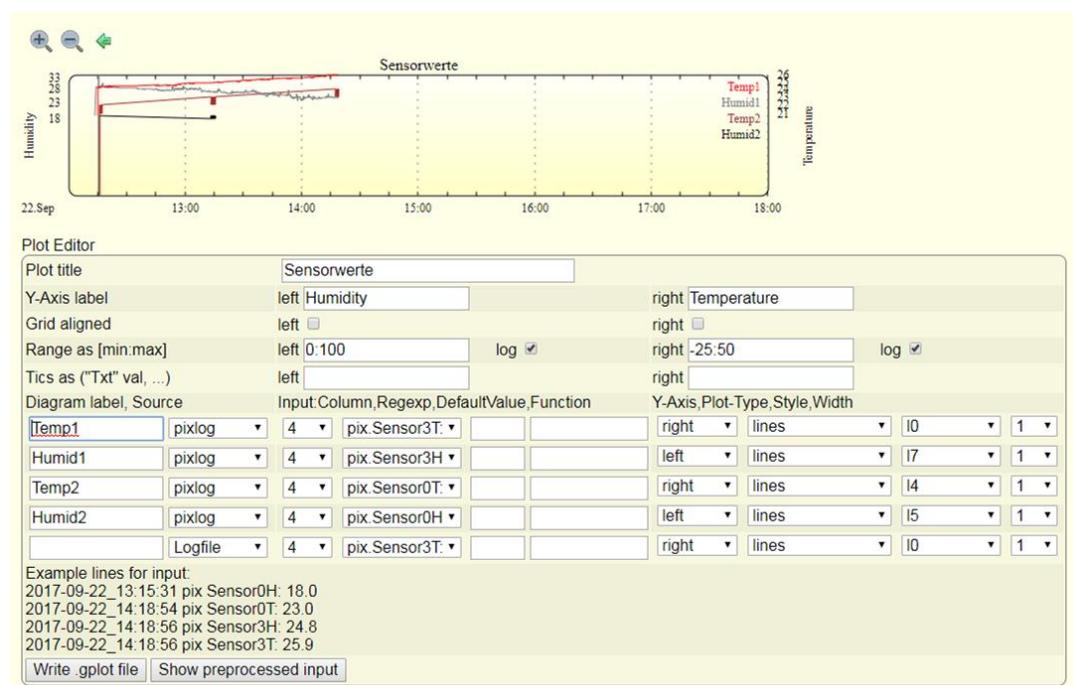


Abbildung 103: FHEM - Diagramm der Sensormesswerte

### NOTICE

Der DHT11-Sensor (Temp2 & Humid2) weist nur eine Genauigkeit von 1°C bzw. 1% auf. Es entsteht nur ein neuer Messpunkt, wenn sich die Readings, das heißt der Messwert ändert.

## 10.4.2. Ein- und Ausgänge entprellen und verknüpfen

Im ersten Schritt soll ein digitaler Eingang (DIO) entprellt und anschließend ein Relay (Relay0) geschaltet werden. Zum Entprellen des Eingangs wird das nachstehende Kommando ausgeführt. Die digitalen Eingänge DIO und DI1 müssen über eine Zeit von  $10 \times 100 \text{ ms} = 1 \text{ s}$  einen konstanten Wert aufweisen, das heißt sie werden entprellt.

```
set pix digitaldebounce01 10
```

Um ein Relay zu schalten lässt sich ein „notify“ mit folgendem Kommando erstellen:

```
define n_relay notify pix:DigitalIn0:.* set pix RelayOut0 $EVTPART1
```

Das Relay wird entsprechend dem Zustand (\$EVTPART1) von DigitalIn0 auf „on“ oder „off“ gesetzt, der digitale Eingang reagiert im Idealfall um 1 s verzögert.

Im zweiten Schritt wird unter Verwendung des digitalen Eingang DI2 und dem Relay1 eine Ausschaltverzögerung realisiert. Zunächst wird der digitale Eingang entprellt, dieses Mal nur 100 ms.

```
set pix digitaldebounce23 1
```

Nun wird ein DOIF-Modul mit dem Namen „TimedSW“ erstellt:

```
define TimedSW DOIF ([pix:DigitalIn2:"(on)"]) (set pix relayout1 on, set pix digitaldebounce23 50)
DOELSE (set pix relayout1 off, set pix digitaldebounce23 1)
```

Ist DigitalIn2 „on“, wird im zweiten Schritt das Relay1 auf „on“ gesetzt und zusätzlich der Entprellwert auf 50 (= 5 s) eingestellt. Ist DigitalIn2 „off“, wird Relay1 auf „off“ gesetzt und der Entprellwert wieder zurückgesetzt (0,1 s). Damit erreichen wir, dass das Relay nach einem kurzen Entprellen sofort eingeschaltet jedoch um 5 s verzögert ausgeschaltet wird.

Im dritten Schritt werden zwei Bedingungen miteinander verknüpft. Relay2 wird nur geschaltet, wenn am analogen Eingang AIO eine Spannung größer als 5 V gemessen wird und gleichzeitig der digitale Eingang DI4 aktiv ist. Wir erstellen ein DOIF-Modul mit folgendem Kommando:

```
define Condi DOIF ([pix:DigitalIn4:"(on)"] and [pix:AnalogIn0] > 5.0) (set pix relayout2 on) DOELSE
(set pix relayout2 off)
```

## 10.4.3. Servo-Motor ansteuern

Dieses Beispiel verdeutlicht die Ansteuerung eines Servo-Motors in FHEM. Standardmäßig ist der Servo-Mode in den PWM-Einstellungen aktiv und muss nicht konfiguriert werden. Die Ausgänge der Servo-Kanäle müssen aktiviert werden, nur dann wird ein Signal ausgegeben. Entsprechend der Beschreibung in Kapitel 14 zur Konfiguration der PWM-Kanäle, muss der Wert „24“ in PWMxCtrl0 geschrieben werden. Zum Beispiel

```
set pix pwm0ctrl0 24
```

um die beiden PWM-Kanäle PWM0A und PWM0B zu aktivieren. Über die Felder PWM0A und PWM0B lässt sich der Servo-Motor ansteuern. Bitte beachten Sie, dass der Wert 0 dem Minimalausschlag und der Wert 16000 dem Maximalausschlag entspricht, beispielsweise:

```
set pix pwm0a 9000
```

Standardmäßig entspricht der Urzustand der Servo-Mode dem Wert 0 für PWMxA/B. Sie möchten mit einem anderen Wert beginnen? Dann stellen Sie zuerst diesen Wert ein und aktivieren anschließend den zugehörigen Kanal.

## 10.5. Frequently Asked Questions (FAQ)

**Warum zeigen die Readings für die Sensorwerte „Sensor not connected“ oder „Function not enabled“ an?**

Die Funktion Sensorwerte auslesen wurde nicht für den entsprechenden GPIO-Kanal aktiviert („\_GPIOxCtrl“) oder die Einstellung wurde ausgewählt aber kein Sensor angeschlossen oder falsch angeschlossen. Aktivieren Sie die Funktion oder überprüfen Sie die Verkabelung des Sensors.

**Während dem Anlegen (define) des Geräts erscheint der Fehler „ioctl.ph is not available but needed“.**

Das PiXtend V2-Modul für FHEM benötigt ioctl.ph um auf die SPI-Schnittstelle zuzugreifen. Die Datei wurde nicht gefunden, wahrscheinlich liegt ein Fehler in der Raspbean-Installation vor. Aus diesem Grund empfehlen wir Ihnen die von uns getesteten Images zu verwenden.

**Während dem Anlegen (define) des Geräts erscheint der Fehler „Error! Device not xxx. Please change access rights for fhem user ...“.**

Der Benutzer „fhem“ benötigt Rechte für die SPI-Schnittstelle und die GPIOs des Raspberry Pi. Der Benutzer erhält die Rechte, indem Sie ihn in die entsprechenden Gruppen aufnehmen:

```
sudo adduser fhem gpio; sudo adduser fhem spi; sudo reboot
```

**Als „STATE“ wird nur „error“ angezeigt, worin liegt das Problem?**

Es findet keine Kommunikation zwischen PiXtend V2 und Raspberry Pi statt, überprüfen Sie die Verbindung der beiden Geräte. Ist die Schalterstellungen der beiden Schalter auf dem PiXtend V2-Board korrekt? Befindet sich PiXtend V2 nicht im sicheren Zustand (Reset durchführen) und ist die SPI-Schnittstelle im Raspberry Pi aktiviert?

## 11. PiXtend Python Library V2 (PPLV2)

Die Programmiersprache Python ist eine universelle und interpretierbare Sprache, Ihre Form ist knapp und lesbar und erleichtert somit das Programmieren. Eines der Phyton Hauptmerkmal ist die Strukturierung des Programmcodes, sie erfolgt durch die Einrückung des Codes<sup>5</sup>. Ein weiterer Vorteil ist die große Bibliothek, für fast jeden Anwendungsfall gibt es eine Dokumentation.



Abbildung 104: Python - Bildquelle: <https://www.python.org/>,  
The Python Brochure

Python ist bereits im Raspbian Image integriert<sup>6</sup>. Nach dem ersten Start kann mit der Python Programmierung auf dem Raspberry Pi sofort begonnen werden. Der Zugriff auf die on-board GPIOs oder den SPI-Bus funktionieren „Out of the Box“.

Die PPLV2 ist Open Source und kann von jedem verwendet, geändert und nach Belieben erweitert werden.

Viele weitere Informationen, Tipps und Tricks finden Sie in unserem Support-Forum unter:  
<https://www.pixtend.de/forum/>

Die jeweils neusten Versionen aller Dokumente und Software-Komponenten finden Sie im Download-Bereich unserer Homepage: <https://www.pixtend.de/downloads/>

### 11.1. Voraussetzungen

Die Treiber-Unterstützung von der PPLV2 bezieht sich auf das PiXtend V2 für Python 2.7.9 und Python 3. Wir empfehlen eines dieser Raspberry Pi Modelle: B+, 2 B, 3 B, 3 B+, 4 B.

Laden Sie das **PiXtend Basis 2.x.x.x SD-Karten Image** aus unserem Download-Bereich herunter und verwenden Sie es als Ausgangspunkt für Ihre Projekte. Alternativ können Sie ein original Raspbian Buster Image verwenden.

Auf den Raspberry Pi können Sie mit direkt angeschlossener Tastatur und Monitor zugreifen oder von jedem PC per SSH Zugriff (TeraTerm / Putty). Verwenden Sie ein Original Raspbian Image, dann benötigt der Raspberry Pi eine aktive Internetverbindung, um die Schritte in diesem Kapitel durchzuführen.

Wir empfehlen die PiXtend V2 Kapitel Prozessdaten und Control & Status -Bytes im Anhang. Diese enthalten Informationen zur Konfiguration der PWM-Ausgänge und der analogen Eingänge, Wissenswertes zu den GPIOs, den digitalen Ein- und Ausgängen und den Relais auf PiXtend V2.

Für Informationen zu SSH-Clients, dem Programm Putty, WinSCP oder zur Auswahl eines geeigneten Programmiertools, laden Sie die PiXtend V1.3 App-Note für Python, APP-PX-401 PiXtend Python Library, herunter. Viele Abschnitte dieser App-Note bieten nützliche Hinweise zur Arbeit mit der PPLV2.

<sup>5</sup>Quelle Wikipedia April 2017: [https://de.wikipedia.org/wiki/Python\\_\(Programmiersprache\)](https://de.wikipedia.org/wiki/Python_(Programmiersprache))

<sup>6</sup><https://www.raspberrypi.org/documentation/usage/python/> und <https://www.raspberrypi.org/downloads/raspbian/>

## 11.2. Installation mit PiXtend V2 Image

Im PiXtend Basis 2.x.x.x SD-Karten Image ist die PiXtend Python Library V2 bereits vorinstalliert und kann sofort verwendet werden. Weitere Informationen zur Erstellung einer SD-Karte für den Raspberry Pi entnehmen Sie dem Kapitel 6.5 SD-Karte für Ihren Raspberry Pi vorbereiten.

Auf dem Raspberry Pi finden Sie alle Dateien im Ordner /home/pi/pplv2/.

## 11.3. Installation auf dem Original-Raspbian

### 11.3.1. Vorbereitung

Wir starten mit einem originalen Raspbian Buster Image (Release: 26.09.19). Bei diesem Image startet nach dem ersten Booten automatisch die PIXEL-Oberfläche. Da wir diese hier nicht benötigen, deaktivieren wir sie. Für die nachfolgenden Schritte werden Bildschirm, Tastatur und Maus benötigt. Ein Anmelden über Ethernet per SSH ist nicht möglich, da standardmäßig der SSH Server deaktiviert ist.

In der Menüleiste der PIXEL-Oberfläche auf das Terminal-Symbol klicken und ein Konsolenfenster öffnen. Das Fenster maximieren.



Abbildung 105: Python - Raspbian PIXEL- Terminal Icon in der Taskleiste

Die Konfiguration auf dem Raspberry Pi öffnen:

```
sudo raspi-config
```

Das Raspberry Pi Konfigurationsprogramm wird ausgeführt. Im Menüpunkt „3 Boot Options“ „B1 Desktop / CLI“ auswählen und im Untermenü den Punkt „B2 Console Autologin“. Die Auswahl sieht wie folgt aus:

„3 Boot Options“ → „B1 Desktop / CLI“ → „B2 Console Autologin“

Aktivieren Sie im Konfigurationsprogramm raspi-config den SPI-Bus:

„5 Interfacing Options“ → „P4 SPI“ → <Yes> → <Ok>

Um über das Netzwerk auf den Raspberry Pi per SSH zugreifen können, zur Ausführung eigener Python Programme, muss der SSH Server aktiviert werden:

„5 Interfacing Options“ → „P2 SSH“ → <Yes> → <Ok>

Um das Konfigurationsprogramm zu verlassen, zwei Mal die Tabulatortaste drücken bis das Wort <Finish> rot hervorgehoben wird, anschließend die Eingabetaste drücken.

Nach diesen Änderungen muss ein Reboot durchgeführt werden. Sollte raspi-config beim Verlassen des Programms nicht automatisch ein Reboot anbieten, folgenden Befehl eingeben:

```
sudo reboot
```

Als nächsten Schritte laden und installieren wir die benötigten Komponenten, dafür benötigt Raspberry Pi eine aktive Internetverbindung.

### 11.3.2. PiXtend Python Library V2 installieren

Für die Installation der PPLV2 legen wir im ersten Schritt ein Verzeichnis für das PPLV2 Package an. Wir laden das Package herunter und entpacken es in das erstellte Verzeichnis, anschließend wird das PPLV2 Package installiert. Jetzt kann die PiXtend Python Library V2 global in allen Python 2.7.9 und Python 3 Programmen verwendet werden.

Die Installation eines Python Packages kann viele Ausgaben auf der Konsole hervorrufen, dies ist völlig normal.

Folgende Schritte der Reihe nach ausführen:

```
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
sudo python3 setup.py install
```

Wurde der letzte Befehl erfolgreich ausgeführt, ist die PiXtend Python Library V2 installiert und kann in eigenen Python Programmen verwendet werden. Die Installation des PPLV2 Packages lässt sich mit pip freeze bzw. pip3 freeze überprüfen. In der Liste der installierten Packages befindet sich jetzt der Eintrag pixtendlibv2==0.1.3.

## 11.4. Programmierung

Unser Beispielprogramm befindet sich bereits auf dem Raspberry Pi und ist Teil der PiXtend Python Library V2, es ist im Ordner /home/pi/pplv2/examples.

Arbeiten Sie direkt am Raspberry Pi, dann können direkt starten. Verbinden Sie sich per Netzwerk mit dem Raspberry Pi dann sollten Sie zuerst ein SSH-Client Programm „Putty“ herunterladen.

### 11.4.1. Beispiel-Verzeichnis

Nach dem Login sind wir im pi-Benutzer Home-Verzeichnis (/home/pi). Hier befindet sich der eingangs erstellte Ordner pplv2, der auf unserer SD-Karte bereits vorhanden ist und in dem sich der Quellcode der PiXtend Python Library V2 und der Unterordner examples mit den Beispielen befindet.

Wir wechseln in das Verzeichnis mit den Beispielen:

```
cd ~/pplv2/examples/
```

```
pi@raspberrypi:~ $ cd ~/pplv2/examples/
pi@raspberrypi:~/pplv2/examples $
```

Abbildung 106: Python - Konsole - PPLV2 Verzeichnis

### 11.4.2. Beispiel ausführen

Wir verwenden exemplarisch ein PiXtend V2 -S- und starten das Python PiXtend V2 Demo Programm. Es werden alle digitalen Eingänge gelesen und alle digitalen Ausgänge abwechselnd geschaltet. Das Klacken der Relais ist gut hörbar und die LEDs zeigen zusätzlich den Zustand an.

Mit folgendem Befehl starten wir:

```
sudo python pixtendv2s_demo.py
```

Haben Sie ein PiXtend V2 -L- probieren Sie folgendes:

```
sudo python pixtendv2l_demo.py
```

## Nach dem Start des

Abbildung 107: Python - PiXtend Python Library V2 Demo-Programm

Programms ändern die digitalen Ausgänge und die Relais in etwa jede Sekunde ihre Zustände. In der Konsole wird der aktuelle Zustand in Textform angezeigt, „False“ bedeutet Aus und „True“ bedeutet Ein. Es gibt einen Zykluszähler, der sich jede Sekunde um eins erhöht.

Das Python Programm kann mit den Tasten Strg + C (Ctrl + C) beendet werden.

### 11.4.3. Eigenes Programm erstellen

Öffnen Sie einen Editor oder ein Programm, mit dem Sie Python Programme schreiben bzw. bearbeiten können. Legen Sie eine neue leere Python-Datei an. Speichern Sie diese Datei zum Beispiel unter dem Namen „programm1.py“. Wir verwenden exemplarisch ein PiXtend V2 -S-, die gleichen Funktionen stehen beim PiXtend V2 -L- zur Verfügung. Tauschen Sie das „S“ durch ein „L“ aus.

In unserem ersten Programm programmieren wir, dass das Relais 0 zyklisch jede Sekunde an und wieder ausgeht.

Das Programm könnte folgendermaßen aussehen:

```
#!/usr/bin/env python
```

```
from pixtendv2s 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)
```

Abbildung 108: Python - Einfaches Demo-Programm

Programm Erläuterung:

- In der ersten Zeile schreiben wir, dass es sich um ein Python Programm handelt.
- Danach importieren wir die PiXtendV2S Klasse um Zugriff auf Eigenschaften und Funktionen von PiXtend V2 - S- zu erhalten. Die Kommunikation mit dem Mikrocontroller und DAC wird möglich.
- Zum Warten am Ende des Programms importieren wir die time Klasse.
- Mit p = PiXtendV2S erstellen wir ein Objekt, mit dem wir arbeiten können, die Kommunikation mit dem Mikrocontroller wird automatisch im Hintergrund ausgeführt.
- In einer While Schleife fragen wir das Relais 0 ab ob es an oder aus ist und reagieren entsprechend.
- Die Konstanten ON und OFF können durch True und False ersetzt werden. Das Arbeiten mit Namen verdeutlicht das eigentliche Vorgehen.
- Am Ende warten wir eine Sekunde, um das Relais 0 nicht zu überfordern.

Übertragen Sie das Python Programm auf den Raspberry Pi und führen Sie es mit folgendem Aufruf aus:

```
sudo python programm1.py
```

Nach dem Start beginnt das Relais 0 sich ein- und auszuschalten, das erfolgt nun jede Sekunde.

Um das Programm zu beenden wird die Tastenkombination Strg+C (Ctrl+C) zwei Mal nacheinander angeklickt. Beim ersten Mal verlassen wir die While Schleife, beim zweiten Mal beenden wir die im Hintergrund laufende Kommunikation mit dem PiXtend V2 -S- Board. In Ihren eigenen Programmen können Sie die „close“ Funktion der PiXtendV2S Klasse vor dem Beenden des Programms aufrufen. Die Kommunikation mit dem Mikrocontroller wird beendet, der SPI-Treiber geschlossen und die asynchrone Kommunikation eingestellt.

**NOTICE**

Beachten Sie, dass es auf einem Raspberry Pi immer nur eine einzige Instanz der PiXtendV2S oder PiXtendV2L Klasse geben darf. Es spielt keine Rolle wie viele Python Programme erstellt werden. Diese Einschränkung hängt mit dem SPI-Bus des Raspberry Pi zusammen, dieser darf nur von einem Programm zu einer Zeit verwendet werden.

#### 11.4.4. Kurzübersicht

Die PPLV2 besteht aus mehreren Python Klassen: PiXtendV2Core, PiXtendV2S und PiXtendV2L. Die PiXtendV2Core Klasse stellt die Basisklasse dar, mit vielen grundlegenden Funktionen und Eigenschaften, ist jedoch nicht lauffähig. Die PiXtendV2S/PiXtendV2L Klassen erben diese Eigenschaften und Funktionen und bilden alle verbleibenden Eigenschaften des jeweiligen PiXtend V2 aus.

Nach der Instanziierung der PiXtendV2S/PiXtendV2L Klasse, wird die Kommunikation mit dem Mikrocontroller auf dem PiXtend V2 Board automatisch im Hintergrund ausgeführt. Der Anwender kann alle digitalen und analogen Ein- und Ausgänge verwenden, ohne sich um die Kommunikation zu kümmern. Ein Warten nach jedem Zyklus ist nicht erforderlich.

In der Zip-Datei (pplv2\_v0.1.x.zip) bzw. im Raspberry Pi Ordner pplv2, befinden sich im Unterordner „doc“, HTML-Dateien, die eine API-Dokumentation zu den genannten Python-Klassen enthalten. Der Hilfe-Datei für die PiXtendV2S/PiXtendV2L Klassen können Sie alle verwendbaren Funktionen und Eigenschaften entnehmen. Möchten Sie die GPIOs, die Temperatur- und Luftfeuchte Eingänge oder die PWM-Ausgänge verwenden, dann schauen Sie dort nach.

Die folgende Tabelle zeigt eine kurze Übersicht der wichtigsten Funktionen und Eigenschaften der Python-Klasse PiXtendV2S<sup>7</sup>:

Name	Typ	Beschreibung
		<p>Instanziierung der PiXtendV2S Klasse und Start der Kommunikation mit dem PiXtend V2 -S- Mikrocontroller.          Beispiel:  <code>p = PiXtendV2S()</code></p> <p>Instanziierung der PiXtendV2L Klasse und Start der Kommunikation mit dem PiXtend V2 -L- Mikrocontroller.          Beispiel:  <code>p = PiXtendV2L()</code></p>
close	Funktion	<p>Das Python Programm soll beendet werden, dann rufen Sie die close Funktion auf und erst im Anschluss die PiXtendV2S Instanz zum Löschen. Die close Funktion setzt alle internen Variablen und Objekte zurück, schließt den SPI Treiber und beendet die Hintergrund-Kommunikation.</p> <p>Beispiel:  <code>p.close()</code>  <code>p = None</code></p>
digital_in0 .. 7	Eigenschaft (r)	Mit diesen 8 nur lesen Eigenschaften (digital_in0 bis digital_in7) können die Zustände der digitalen Eingänge vom PiXtend V2 -S- gelesen werden. Die Eigenschaften liefern entweder den Wert False für aus (OFF) oder den Wert True für an (ON).
digital_out0 .. 3	Eigenschaft (rw)	Die 4 digitalen Ausgänge können über die Eigenschaften digital_out0 bis digital_out3 sowohl gelesen als auch geschrieben werden. Wird der Eigenschaft der Wert False zugewiesen, so geht der entsprechende digitale Ausgang aus (OFF) oder bei der Zuweisung des Wertes True geht der Ausgang an (ON).
relay0 .. 3	Eigenschaft (rw)	Die 4 Relais können über die Eigenschaften relay0 bis relay3 sowohl gelesen als auch geschrieben werden. Wird der Eigenschaft der Wert False zugewiesen, so geht das entsprechende Relais aus (OFF) oder bei der Zuweisung des Wertes True geht es an (ON).
analog_in0 .. 1	Eigenschaft (r)	Über diese beiden Eigenschaften können die analogen Eingänge vom PiXtend V2 -S- eingelesen werden.
hardware	Eigenschaft (r)	Über diese nur lesen Eigenschaft kann die Hardware-Version ausgelesen werden. Der Wert 21 steht beispielsweise für Board Version 2.1.
firmware	Eigenschaft (r)	Über diese nur lesen Eigenschaft kann die Firmware-Version des Mikrocontrollers ermittelt werden.
ON	Konstante	Entspricht dem Wert True.
OFF	Konstante	Entspricht dem Wert False.

#### 11.4.5. Python Programm automatisch starten

Nach einem Reboot / Power-Up des Raspberry Pi startet ein Python-Programm nicht automatisch. Das Python-Programm kann mit folgendem Befehl gestartet werden, das myprogram.py bitte durch den eigenen Programmnamen ersetzen:

```
sudo python myprogram.py
```

Dieser Befehl lässt sich automatisch beim Hochfahren des Linux-Systems ausführen.

Die Änderung ist in der Linux-Console einfach umgesetzt:

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

---

<sup>7</sup>Bedeutung (r) und (rw) hinter einer Eigenschaft: r = read only / nur lesen und rw = read and write / lesen und schreiben.  
 Die Funktionen und Eigenschaften für das PiXtend V2 -L- bitte in der API-Dokumentation im doc Ordner nachschlagen.

Es öffnet sich die Datei „rc.local“ mit etwas Inhalt. Wir tragen vor der Zeile „exit 0“ zwei neue Zeilen ein mit der Annahme, dass das zu startende Programm liegt im Home-Verzeichnis:

```
cd /home/pi/  
sudo python myprogram.py &
```

Das „&“ ist kein Tippfehler, damit startet das Python-Programm als Prozess im Hintergrund.

Abspeichern und mit STRG + X verlassen → Rebooten

**⚠ CAUTION**

Beinhaltet das Programm einen Fehler, die digitalen Ausgänge werden unkontrolliert oder sehr schnell angesteuert, kann dies die angeschlossene Peripherie beschädigen. Da das fehlerhafte Programm als "Autostartprogramm" eingerichtet ist, hilft kein Neustart, um das Problem zu beheben. Erstellen Sie immer eine Sicherungskopie Ihrer Arbeit.

## 11.4.6. Verwendung der seriellen Schnittstelle

Die Verwendung der seriellen Schnittstelle ist nicht im PPLV2 Package enthalten, greifen Sie auf pySerial zurück, es betrifft den Raspberry Pi und Python. Bei der in dieser Dokumentation erwähnten Raspbian Version läuft die serielle Schnittstelle beispielsweise über das Linux Gerät /dev/ttyS0.

Bitte beachten Sie, es werden die Ausgaben der Linux Konsole ausgegeben. Vor Verwendung sollte die /boot/cmdline.txt bearbeiten und der Teil console=serial0,115200 aus der ersten Zeile entfernen werden oder Sie verwenden das Programm raspi-config um die Konsolen-Ausgabe abzuschalten.

## 11.4.7. Frequently Asked Questions (FAQ)

### Das Programm lässt sich nicht starten, Python gibt eine Fehlermeldung über eine ungültige Identifikation aus?

Das Problem liegt vermutlich bei der Einrückung der verschiedenen Code-Zeilen im Programm. Überprüfen Sie nochmals ob bei jeder Zeile die Einrückung stimmt. Wir empfehlen stets 4 Leerzeichen pro Einrückungsebene, fehlt ein Leerzeichen oder Leerzeichen und Tabulatoren wurden vermischt, gibt Python eine Fehlermeldung aus. In Notepad++ die „nicht druckbaren“ Zeichen anzeigen lassen, so lässt sich der Fehler schnell erkennen. Außerdem teilt Python mit in welcher Zeile es den Fehler festgestellt hat.

### Kann mein Programm ohne sudo arbeiten?

Das ist mit Python in dem hier verwendeten Release vom Raspbian tatsächlich möglich, wir empfehlen dennoch, sudo zu verwenden, damit Python Systemfunktionen ausführen kann. Es kann sonst zu Problemen kommen, deren Ursache nicht einfach zu erklären ist.

### Warum schalten sich die Ausgänge und Relais von selbst aus oder blinken, wenn sie im Programm eingeschaltet werden?

Auf einem Raspberry Pi darf die PiXtendV2S oder PiXtendV2L Klasse, je nach PiXtend Board, nur einmal verwendet werden. Eine Mehrfachnutzung ist auf Grund des SPI-Busses nicht möglich, unabhängig davon, wie viele Python Programme man erstellen möchte. Ein Master-/ Hauptprogramm, mit nur einem PiXtend V2 Objekt, darf mit dem PiXtend V2 Board kommunizieren, nicht mehrere.

### Wo und wie lassen sich beim PPLv2 die Zykluszeit einstellen?

Die Zykluszeit, wie bereits in Kapitel 6.2 SPI-Kommunikation, Datenübertragung und Zykluszeit aufgeführt, kann vom Anwender geändert werden. Die Zykluszeit sagt aus in welchen Abständen oder Intervallen die PPLv2 mit dem PiXtend V2 Board Daten austauscht.

Um bei jedem Python-Projekt die optimale Zykluszeit einzustellen, verfügt die Basis Klasse der PPLv2 über den Parameter „com\_interval“. Man kann in Abhängigkeit des vorhandenen PiXtend V2 Boards die Zykluszeit bequem bei der Instanziierung eines PiXtend V2 Board Objekts anpassen. Standardmäßig arbeitet die PPLv2 mit einer Zykluszeit von 30 ms (0.03 Sekunden), diese Angaben entnehmen Sie der API-Dokumentation im „doc“ Unterordner des „pplv2“ Ordners.

### Beispiel für PiXtend V2 -S-, 100 ms Zykluszeit:

```
#!/usr/bin/env python
from pixtendv2s import PiXtendV2S
import time
p = PiXtendV2S(com_interval=0.1)
while True:
    if p.relay0 == p.OFF:
        p.relay0 = p.ON
    else:
        p.relay0 = p.OFF
    time.sleep(1)
```

### Beispiel für PiXtend V2 -L-, 50 ms Zykluszeit:

```
#!/usr/bin/env python
from pixtendv2l import PiXtendV2L
import time
p = PiXtendV2L(com_interval=0.05)
while True:
    if p.relay0 == p.OFF:
        p.relay0 = p.ON
    else:
        p.relay0 = p.OFF
    time.sleep(1)
```

## 12. Anhang

Dieses Kapitel zeigt auf, welche Prozessdaten zwischen PiXtend-Mikrocontroller und dem Raspberry Pi (RPi) ausgetauscht werden. Wir sprechen von Prozessdaten, da es sich um die Daten der Ein- und Ausgänge des PiXtend-Boards handelt.

Bei den PiXtend-Beispielprogrammen für CODESYS werden die Daten korrekt übertragen, ausgewertet und gegebenenfalls aufbereitet.

Bei den Linux-Tools und der Verwendung der PiXtend-Library (pxdev) sowie der PiXtend Python Library ist es wichtig zu wissen wie die Daten beeinflusst und ausgewertet werden können.

Die Prozessdaten sind im PiXtend-Mikrocontroller festgelegt und daher unabhängig von der verwendeten Programmiersprache bzw. Linux-Software.

Die jeweils neusten Versionen aller Dokumente und Software-Komponenten finden Sie im Download-Bereich unserer Homepage: <https://www.pixtend.de/downloads/>

## 12.1. PiXtend V2 -S-

### 12.1.1. Prozessdaten

#### 12.1.1.1 Prozessdaten im Überblick

Es gibt zwei Arten von Prozessdaten:

- Prozessdaten die vom Raspberry Pi an PiXtend übertragen werden
- Prozessdaten die von PiXtend an den Raspberry Pi übertragen werden

#### 12.1.1.1.1 Prozessdaten die vom RPi an PiXtend übertragen werden

Daten für digitale Ausgänge, Relais und GPIOs (als Ausgänge) PWM-Daten, Retain-Daten

- DigitalOut
- RelayOut
- GPIOOut
- PWM0X (L/H)
- PWM1X (L/H)
- RetainDataOutX

#### 12.1.1.1.2 Prozessdaten die vom RPi an den PiXtend-DAC übertragen werden

Daten für analoge Ausgänge

- AnalogOutX (L/H) -

#### 12.1.1.1.3 Prozessdaten die von PiXtend an den RPi übertragen werden

Daten der digitalen und analogen Eingänge, GPIOs (als Eingänge)

Temperatur und Luftfeuchtigkeit von DHT11/22- oder AM2302-Sensoren, Retain-Daten

- DigitalIn
- AnalogInX (L/H)
- GPIOIn
- TempX (L/H)
- HumidX (L/H)
- RetainDataInX

## 12.1.1.2 SPI-Bus Protokoll Übersicht

INPUT			OUTPUT			
	Name	used Bits	Byte Idx	Name	used Bits	
HeaderIn	Firmware	8	0	HeaderOut	ModelOut	8
	Hardware	8	1		UCMode	8
	Modelln	8	2		UCCtrl0	8
	UCState	8	3		UCCtrl1	8
	UCWarnings	8	4		Reserved	0
	Reserved	0	5		Reserved	0
	Reserved	0	6		Reserved	0
	CRCHeaderInL	8	7		CRCHandlerOutL	8
	CRCHandlerInH	8	8		CRCHandlerOutH	8
DataIn	DigitalIn	8	9	DataOut	DigitalInDebounce01	8
	AnalogIn0L	8	10		DigitalInDebounce23	8
	AnalogIn0H	2	11		DigitalInDebounce45	8
	AnalogIn1L	8	12		DigitalInDebounce67	8
	AnalogIn1H	2	13		DigitalOut	4
	GPIOIn	4	14		RelayOut	4
	Temp0L	8	15		GPIOCtrl	8
	Temp0H	8	16		GPIOOut	4
	Humid0L	8	17		GPIODebounce01	8
	Humid0H	8	18		GPIODebounce23	8
	Temp1L	8	19		PWM0Ctrl0	8
	Temp1H	8	20		PWM0Ctrl1L	8
	Humid1L	8	21		PWM0Ctrl1H	8
	Humid1H	8	22		PWM0AL	8
	Temp2L	8	23		PWM0AH	8
	Temp2H	8	24		PWM0BL	8
	Humid2L	8	25		PWM0BH	8
	Humid2H	8	26		PWM1Ctrl0	8
	Temp3L	8	27		PWM1Ctrl1L	8
	Temp3H	8	28		PWM1Ctrl1H	8
	Humid3L	8	29		PWM1AL	8
	Humid3H	8	30		PWM1AH	8
	Reserved	0	31		PWM1BL	8
	Reserved	0	32		PWM1BH	8
	RetainDataIn0 .. 31	all	33		RetainDataOut0 .. 31	all
	CRCDataInL	8	65		CRCDataOutL	8
	CRCDataInH	8	66		CRCDataOutH	8

Tabelle 5: PiXtend V2 -S- SPI-Protokoll Übersicht

In diesem Kapitel finden Sie eine Beschreibung zu jedem Byte des SPI-Protokolls, zum Teil mit Anwendungshinweisen und Berechnungsbeispielen, siehe PWMs. Das SPI-Protokoll für das PiXtend V2 -S- umfasst insgesamt 67 Bytes, die Tabelle 7 zeigt eine verkürzte Form, hier wurden die 32 Bytes des Retain-Speichers zusammengefasst.

Handelt es sich bei einem Eintrag um ein einzelnes Byte, so steht im SPI-Protokoll nur der Name oder die Bezeichnung des Bytes. Bei 16 Bit Werten (2 Bytes), zum Beispiel Temperatur, wird der Wert auf 2 einzelne Bytes aufgeteilt. Dies ist an den Zusätzen „L“ für Low-Byte und „H“ für High-Byte zu erkennen. Sie möchten das SPI-Protokoll implementieren, dann achten Sie darauf, dass Sie die Bytes für High und Low entsprechend zuordnen.

Die CRC-Berechnung für den Header und die Daten sind nicht explizit aufgeführt, bitte entnehmen Sie weitere Details dem Sourcecode für unsere Linux Tools (pxdev), zu finden auf SourceForge, unserer Homepage im Downloadbereich oder in unseren Python Modulen. Wir haben eine entsprechende Implementierung, die Sie als Vorlage für Ihre Programmiersprache verwenden können, bereitgestellt.

### 12.1.1.3 SPI-Bus Konfiguration

Um PiXtend V2 SPI Devices aus eigenen Applikationen anzusprechen sollten folgende SPI-Bus Einstellungen beachtet werden:

- SPI-Mode 0 (CPOL = 0, CPHA = 0)
- SPI Geschwindigkeit: 700 kHz
- SPI ChipSelect 0 (CS0): PiXtend V2 Mikrocontroller
- SPI ChipSelect 1 (CS1): PiXtend DAC bzw. CAN-Interface (bei V2 -L-)
- GPIO24, wiringPi BCM Nummerierung (wiringPi's eigene Nummerierung GPIO 5, Name: GPIO. 5) des Raspberry Pi muss auf "high" / logisch "1" gesetzt werden → SPI Enable.

Zykluszeitbeschränkung:

- Minimale Zykluszeit V2 -S-: 2,5 ms
- Minimale Zykluszeit V2 -L-: 5 ms

Bei dieser Angabe handelt es sich um den zeitlichen Mindestabstand zwischen zwei Kommunikationsvorgängen mit dem Mikrocontroller. Längere Zykluszeiten sind besser.

Wir empfehlen den Sourcecode der PiXtend Bibliothek und/oder der Python Module zu studieren um ein besseres Gefühl zur Nutzung des SPI-Busses zu bekommen.

## 12.1.1.4 Ausgangsdaten

Als Ausgangsdaten werden die Daten bezeichnet, die der Raspberry Pi an den PiXtend-Mikrocontroller überträgt, beispielsweise digitale Ausgänge oder Relais.

### 12.1.1.4.1 DigitalOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	D03	D02	D01	D00
Startwert	0	0	0	0	0	0	0	0

Tabelle 6: Bits des DigitalOut Bytes

#### Bit 0...3

Vier digitale Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von D00. Bit 7 (MSB), Bit 6, Bit 5 und Bit 4 sind nicht belegt. Der Startwert nach dem Power-Up oder Reset des Mikrocontrollers ist für das gesamte DigitalOut-Byte „0“. Die Ausgänge sind während des Start-/SafeState deaktiviert.

Durch das Schreiben einer „1“ in eines der Bits (0...3) wird der entsprechende Ausgang aktiviert. Die zugehörige LED leuchtet auf dem PiXtend-Board auf.

Die digitalen Ausgänge auf PiXtend V2 verfügen über eine separate Einspeisung. Sie erkennen an der Leuchtdiode „VCC DO“ ob die Einspeisung angeschlossen und mit Spannung versorgt ist.

Die LEDs der digitalen Ausgänge leuchten auch dann, wenn die VCC-DO-Versorgung nicht angeschlossen wurde, an den DO-Pins wird jedoch keine Spannung „sichtbar“.

Um die Funktion der Ausgänge zu testen, kann für das DigitalOut-Byte der Wert 255 (dezimal) oder 0xFF (hex) geschrieben werden. Es werden alle vier Ausgänge aktiviert, dass dabei die Bits 4 bis 7 gesetzt werden, hat keine Auswirkung.

### 12.1.1.4.2 RelayOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	RELAY3	RELAY2	RELAY1	RELAY0
Startwert	0	0	0	0	0	0	0	0

Tabelle 7: Bits des RelayOut Bytes

#### Bit 0...3

Die vier Relais-Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von RELAY0. Bit 7 (MSB) bis Bit 4 sind nicht belegt. Als Startwert nach dem Power-Up oder Reset des Mikrocontrollers ist das gesamte RelayOut-Byte „0“. Die Relais sind im Start-/SafeState deaktiviert.

Durch das Schreiben einer „1“ in eines der Bits (0...3) wird das entsprechende Relais aktiviert. Die zugehörige LED auf dem PiXtend-Board leuchtet auf und das Relais schaltet hörbar.

Um die Funktion der Relais zu testen, kann für das RelayOut-Byte der Wert 255 (dezimal) oder 0xFF (hex) geschrieben werden. Es werden alle vier Relais aktiviert, dass dabei die Bits 4...7 gesetzt werden, hat keine Auswirkung.

### 12.1.1.4.3 GPIOOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startwert	0	0	0	0	0	0	0	0

Tabelle 8: Bits des GPIOOut Bytes

#### Bit 0...3

Die vier GPIO-Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von GPIO0. Bit 7 (MSB) bis Bit 4 sind nicht belegt. Als Startwert nach dem Power-Up bzw. Reset des Mikrocontrollers ist das gesamte GPIOOut-Byte „0“. Die GPIOs sind im Start-/StafeState deaktiviert.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl. Weitere Informationen entnehmen Sie dem Kapitel: Control- und Status-Bytes.

Durch das Schreiben einer „1“ in eines der Bits (0...3) wird der entsprechende GPIO aktiviert (sofern als Ausgang konfiguriert). Sind der oder die GPIOs als Eingang oder für die genannten Sensoren konfiguriert, so hat das Verändern der Werte in GPIOOut keine Auswirkung\*.

Um die Funktion der GPIOs zu testen, können diese als Ausgänge konfiguriert werden. Anschließend kann für das GPIOOut-Byte der Wert 255 (dezimal) bzw. 0xFF (hex) geschrieben werden. Alle vier GPIO-Ausgänge werden aktiviert, Dass dabei die Bits 4...7 gesetzt werden, hat keine Auswirkung.

\* Ist ein GPIO als Eingang mit PullUps konfiguriert und das zugehörige Bit wird in GPIOOut auf „1“ gesetzt, dann wird ein PullUp-Widerstand an diesem GPIO aktiviert. Damit bekommt der GPIO-Eingang einen hochohmigen (~35 kOhm) Bezug zu 5V und floatet nicht mehr. Ohne externe Beschaltung verharrt der Eingang nun auf High-Pegel/Level.

Standardmäßig handelt es sich bei den GPIO-Eingängen um floating-Inputs ohne definierten Pegel (ohne externe Beschaltung). Die PullUp-Widerstände sollten generell über das Bit „GPIOPullUpEnable“ aktiviert werden.

### 12.1.1.4.4 PWM0X (L/H) – 16 Bit Auflösung

Die PWM-Einheiten von PiXtend können in vier verschiedenen Modi betrieben werden. In jedem Modus haben die PWM-Prozessdaten eine andere Auswirkung. Aus diesem Grund gehen wir in diesem Abschnitt separat auf die vier Modi „Servo-Mode“, „Duty-Cycle-Mode“, „Universal-Mode“ und „Frequency-Mode“ ein.

### 12.1.1.4.5 PWM0Ctrl – für 16 Bit PWMs.

Bei den PWM0X-Bytes handelt es sich um die Prozessdaten für die beiden 16 Bit PWM-Kanäle, die auf der Baugruppe die volle Bezeichnung PWM0A und PWM0B tragen.

### 12.1.1.4.5.1. Frequency-Mode

Im Frequency-Mode enthalten die Bytes PWM0XL und PWM0XH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort wird für die Einstellung der Frequenz des jeweiligen Kanals verwendet. Der Duty-Cycle ist für Kanal A und B immer 50% und kann nicht verändert werden.

Der Modus eignet sich optimal, wenn viele unterschiedliche Frequenzen benötigt werden, der Duty-Cycle aber keine Rolle spielt. Dies ist beispielweise der Fall, wenn mit den PWM-Kanälen Schrittmotortreiber angesteuert werden, die lediglich auf Signalflanken reagieren und keinen variablen Duty-Cycle benötigen. So kann mit PiXtend die Geschwindigkeit von bis zu vier Schrittmotoren gesteuert werden (2x 16 Bit PWM0X, 2x 8 Bit PWM1X).

PWMOAL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 15: Bits des PWMOAL Bytes

PWMOAH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 16: Bits des PWMOAH Bytes

Die ungefähre Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0...2) in PWM0Ctrl0, die Feineinstellung per PWM0XL/H. Die maximale Frequenz in diesem Modus liegt bei 20 kHz. Bei der Einstellung von höheren Frequenzen wird das ausgegebene Signal auf 20 kHz begrenzt.

Die Berechnung der Frequenz erfolgt folgendermaßen:  
**Frequenz Kanal X = 16 MHz / 2 / Prescaler / PWM0X**

#### Ein Beispiel

Auf Kanal A soll eine Frequenz von 500 Hz, auf Kanal B 250 Hz ausgegeben werden.

- Kanal A und B aktivieren, Frequency-Mode auswählen, Prescaler auf 64 konfigurieren  
→ PWM0Ctrl0 = 123 (dezimal) bzw. 01111011b (binär)

- PWM0A (16 Bit Datenwort) = 250 (dezimal)  
→ Frequenz Kanal A = 16 MHz / 2 / 64 / 250 = 500 Hz
- PWM0B (16 Bit Datenwort) = 500 (dezimal)  
→ Frequenz Kanal B = 16 MHz / 2 / 64 / 500 = 250 Hz

Sollen die PWM-Signale nicht durchgehend an den Ausgängen anstehen, lassen sich bei Bedarf in jedem Zyklus der oder die Kanäle aktiviert/deaktiviert.

#### NOTICE

Eine Kombination aus Frequency-Mode und DHT-Sensoren ist nicht möglich. Wurden ein oder mehrere DHT-Sensoren an den PiXtend-GPIOs aktiviert, so lässt sich der Frequency-Mode nicht aktivieren. Der Versuch den Frequency-Mode trotzdem zu aktivieren, obwohl DHT-Sensoren verwendet werden, führt zur Deaktivierung der PWM-Ausgänge.

### 12.1.1.4.5.2. Servo-Mode

Der Servo-Modus ist speziell auf die Anforderungen von Modellbauservos ausgelegt. Dabei ist die Periodendauer fest auf 50 Hz (20 ms) eingestellt. Jeweils zu Beginn der Periode ist das Signal mindestens 1 ms (Minimalausschlag) oder maximal 2 ms (Maximalausschlag) auf High-Pegel. Die verbleibende Zeit ist das Signal auf Low-Pegel.

Die PWM0X-Bytes L und H enthalten ein zusammengehöriges 16 Bit Datenwort. Dabei ist PWM0XL das Low-Byte und PWM0XH das High-Byte. Das „X“ wird durch den Buchstaben des jeweiligen PWM-Kanals ersetzt („A“ oder „B“).

PWM0XL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 9: PWM0XL Byte

PWM0XH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	

Tabelle 10: PWM0XH Byte

#### Einstellbarer Wertebereich

kleinster Wert: 0 (dezimal) → 1 ms, Minimalausschlag

größter Wert: 16000 (dezimal) → 2 ms, Maximalausschlag

Werte größer 16000 werden vom Controller begrenzt und wirken wie 16000. Im Bereich zwischen 0 und 16000 ist die Bewegung des Servos linear. Der Startwert des PWM0X-Bytes, nach einem Power-Up/Reset, ist „0“.

### 12.1.1.4.5.3. Duty-Cycle-Mode

Im Duty-Cycle-Mode enthalten die Bytes PWM0XL und PWM0XH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort wird für die Einstellung des Tastverhältnisses (englisch: duty cycle) verwendet. Das „X“ wird durch den Buchstaben des jeweiligen PWM-Kanals ersetzt („A“ oder „B“). Jedem Kanal kann ein separater Duty-Cycle zugewiesen werden. Die Frequenz und Periodendauer werden gemeinsam für beide Kanäle eingestellt.

Der Modus eignet sich optimal um beispielsweise zwei Antriebe oder Gebläse, unabhängig in deren Geschwindigkeit, von 0% bis 100% zu steuern.

PWM0XL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 11: PWM0XL Byte

PWM0XH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 12: PWM0XH Byte

Welcher Duty-Cycle sich durch einen bestimmten Wert ergibt, hängt nicht ausschließlich von den PWM0XL/H-Werten ab, sondern zusätzlich von den Control-Bytes PWM0Ctrl1L und PWM0Ctrl1H. Die ungefähre Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0...2) in PWM0Ctrl0, die Feineinstellung per PWM0Ctrl1L/H.

Weitere Informationen zur Konfiguration der PWM0-Kanäle entnehmen Sie dem Abschnitt: 14.1.2.3.7 PWM0Ctrl.-für 16 Bit PWMs

#### Berechnungsformeln – PWM0 – 16 Bit

Frequenz:  $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWM0Ctrl1}$   
Duty-Cycle:  $\%_{\text{on}} = 100 \% * \text{PWM0X} / \text{PWM0Ctrl1}$

#### Ein Beispiel

Duty-Cycle Mode sowie die Kanäle A und B aktivieren, Prescaler auf 1024:  
PWM0Ctrl0 = 249 (dezimal) bzw. 11111001b (binär)

In PWM0Ctrl1L/H wird folgender 16 Bit Wert geschrieben: 5000 (dezimal).

In PWM0XL/H wird der 16 Bit Wert 2500 (dezimal) geschrieben. Der Duty-Cycle beträgt hiermit 50%.

Wird in PWM0XL/H ein größerer Wert als in PWM0Ctrl1L/H geschrieben, so verbleibt der PWM-Kanal durchgehend auf logisch „1“. Durchgehend logisch „0“ wird durch den Wert „0“ in PWM0XL/H erreicht.

In unserem Beispiel ergibt sich eine Frequenz von 1,56 Hz.

Wir empfehlen die Implementierung der PWM-Funktionen in Ihren eigenen Programmen zunächst ohne angeschlossene Geräte/Aktoren zu testen und wenn möglich mit einem Oszilloskop zu überprüfen.

#### Setzen Sie die PWM-Werte mit Bedacht!

##### ⚠ CAUTION

Manche Aktoren können durch falsche Frequenzen oder Duty-Cycles beschädigt oder zerstört werden. Überprüfen Sie Ihre Programme immer mit einem Oszilloskop, bevor Sie Geräte an die PWM-Kanäle anschließen.

#### 12.1.1.4.5.4. Universal-Mode

Im Universal-Mode enthalten die Bytes PWM0AL und PWM0AH ein zusammengehöriges 16 Bit Datenwort. Das Datenwort wird für die Einstellung des Tastverhältnisses bzw. Duty-Cycle von Kanal A verwendet.

Der B-Kanal ist in diesem Modus nicht einstellbar und gibt eine PWM mit 50% Duty-Cycle und der halben Frequenz des A-Kanals aus.

Der Modus ist für Anwendungen mit unterschiedlichen Frequenzen geeignet und wenn ein Kanal des Duty-Cycles konfiguriert werden soll.

**PWM0AL**

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 13: PWM0AL Byte

**PWM0AH**

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 14: PWM0AH Byte

Die Konfiguration des A-Kanals wird wie beim Duty-Cycle-Mode gehandhabt.

Eine Änderung der Bytes PWM0BL und PWM0BH hat keine Auswirkung. Der B-Kanal hängt von der Frequenz-Konfiguration des A-Kanals ab.

#### 12.1.1.4.6 PWM1X (L/H) – 8 Bit Auflösung

Die PWM-Einheiten von PiXtend können in vier verschiedenen Modi betrieben werden. In jedem Modus haben die PWM-Prozessdaten eine andere Auswirkung. Aus diesem Grund gehen wir in diesem Abschnitt auf die vier Modi „Servo-Mode“, „Duty-Cycle-Mode“, „Universal-Mode“ und „Frequency-Mode“ separat ein.  
Weitere Informationen zur Konfiguration und Modi der PWM1-Ausgänge finden Sie im Abschnitt: 14.1.2.3.8 PWM1Ctrl.-für 8 BIT PWMs.

Bei den PWM1X-Bytes handelt es sich um die Prozessdaten der beiden 8 Bit PWM-Kanäle, die auf der Baugruppe die volle Bezeichnung PWM1A und PWM1B tragen.

### 12.1.1.4.6.1.Servo-Mode

Der Servo-Modus ist speziell auf die Anforderungen von Modellbauservos ausgelegt. Dabei ist die Periodendauer fest auf 50 Hz (20 ms) eingestellt. Jeweils zu Beginn der Periode ist das Signal mindestens 1 ms (Minimalausschlag) oder maximal 2 ms (Maximalausschlag) auf High-Pegel. Die verbleibende Zeit ist das Signal auf Low-Pegel.  
Bei den PWM1-Kanälen existieren zwei PWM1X-Bytes (L & H) verwendet wird nur das Low-Byte (L). CODESYS bietet nur das verwendete Low-Byte an. Das „X“ wird durch den Buchstaben des jeweiligen PWM-Kanals ersetzt („A“ oder „B“).

#### PWM1XL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 17: Bits des PWM1XL Bytes

#### PWM1XH – wird nicht verwendet

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 18: Bits des PWM1XH Byte

#### Einstellbarer Wertebereich

kleinster Wert: 0 (dezimal) → 1 ms, Minimalausschlag

größter Wert: 125 (dezimal) → 2 ms, Maximalausschlag

Werte größer 125 führen zum Maximalausschlag (zwei Millisekunden). Im Bereich zwischen 0 und 125 ist die Bewegung des Servos linear.

Die Startwert des PWM1XL-Bytes, nach einem Power-Up/Reset, ist „0“.

### 12.1.1.4.6.2. Duty-Cycle-Mode

Bei den PWM1-Kanälen existieren zwei PWM1X-Bytes (L & H), nur das Low-Byte (L) wird verwendet. CODESYS bietet nur das zu verwendende Low-Byte an. Das PWM1XL-Byte wird für die Einstellung des Tastverhältnisses (englisch: duty cycle) verwendet. Das „X“ wird durch den Buchstaben des jeweiligen PWM-Kanals ersetzt („A“ oder „B“). Jedem Kanal lässt sich ein separater Duty-Cycle zuweisen.

Die Frequenz und Periodendauer werden für beide Kanäle gemeinsam eingestellt. Der Modus eignet sich optimal um zwei Antriebe oder Gebläse, unabhängig von deren Geschwindigkeit, von 0% bis 100% zu steuern.

PWM1XL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 19: Bits des PWM1XL Bytes

PWM1XH – wird nicht verwendet

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 20: Bits des PWM1XH Bytes

Welcher Duty-Cycle sich durch einen bestimmten Wert ergibt, hängt ausschließlich von den PWM1XL-Werten ab. Die Frequenz kann bei den PWM1-Kanälen nicht so genau eingestellt werden wie bei den 16 Bit PWM0-Kanälen.

Zwei Faktoren beeinflussen die Frequenz:

1. Prescaler in PWM1Ctrl0 festlegen
2. Wird in PWM1Ctrl1 eine „1“ geschrieben, verdoppelt sich die per Prescaler eingestellte Frequenz

**Berechnung der Frequenz:  $f = 16 \text{ MHz} / 2 / \text{Prescaler} / 255$**

Das Teilen durch 2 entfällt, wenn in PWM1Ctrl1 eine „1“ geschrieben wird.

Weitere Informationen zur Konfiguration finden Sie im Abschnitt: 14.1.2.3.8 PWM1Ctrl.-für 8 Bit PWMs PWM1Ctrl-für 8 Bit PWMs

Wir empfehlen die Implementierung der PWM-Funktionen in Ihren eigenen Programmen zunächst ohne angeschlossene Geräte/Aktoren zu testen und wenn möglich mit einem Oszilloskop zu überprüfen.

**Setzen Sie die PWM-Werte mit Bedacht!**



Manche Aktoren können durch falsche Frequenzen oder Duty-Cycles beschädigt oder zerstört werden. Überprüfen Sie Ihre Programme immer mit einem Oszilloskop, bevor Sie Geräte an die PWM-Kanäle anschließen.

### 12.1.1.4.6.3. Universal-Mode

Bei den PWM1-Kanälen existieren zwei PWM1X-Bytes (L und H), verwendet wird nur das Low-Byte (L). CODESYS verwendet nur das angebotene Low-Byte. Das PWM1AL-Byte wird für die Einstellung des Tastverhältnisses (englisch: duty cycle) von Kanal A verwendet. Der B-Kanal ist in diesem Modus nicht einstellbar, er gibt ein PWM-Signal mit 50% Duty-Cycle und der halben Frequenz des A-Kanals aus.

Der Modus ist für Anwendungen mit unterschiedlichen Frequenzen geeignet und wenn ein Kanal des Duty-Cycles konfiguriert werden soll.

PWM1AL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 21: Bits des PWM1AL Bytes

PWM1AH – wird nicht verwendet

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 22: Bits des PWM1AH Bytes

#### Berechnungsformeln

Frequenz:  $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWM1Ctrl1}$

Duty-Cycle:  $\%_{\text{on}} = 100 \% * \text{PWM1AL} / \text{PWM1Ctrl1}$

Eine Änderung des Bytes PWM1BL hat keine Auswirkung. Der B-Kanal hängt lediglich von der Frequenz-Konfiguration des A-Kanals ab.

#### 12.1.1.4.6.4. Frequency-Mode

Bei den PWM1-Kanälen existieren zwei PWM1X-Bytes (L und H) jedoch nur das Low-Byte (L) wird verwendet. CODESYS bietet nur das zu verwendende Low-Byte an. Das PWM1AL-Byte wird für die Einstellung der Frequenz des jeweiligen Kanals verwendet. Der Duty-Cycle ist für Kanal A und B immer 50% und kann nicht verändert werden.

Der Modus eignet sich optimal, wenn viele unterschiedliche Frequenzen benötigt werden, der Duty-Cycle aber keine Rolle spielt. Dies ist der Fall, wenn mit den PWM-Kanälen Schrittmotortreiber angesteuert werden, die lediglich auf Signalflanken reagieren und keinen variablen Duty-Cycle benötigen. So können mit PiXtend bis zu vier Schrittmotoren in dieser Geschwindigkeit gesteuert werden (2x 16 Bit PWMO<sub>X</sub>, 2x 8 Bit PWM1X).

PWM1AL

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 23: Bits des PWM1AL Bytes

PWM1AH – wird nicht verwendet

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 24: Bits des PWM1AH Bytes

Die ungefähre Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0...2) in PWM1Ctrl0, die Feineinstellung per PWM1XL. Die maximale Frequenz in diesem Modus beträgt 20 kHz. Bei Einstellung von höheren Frequenzen wird das ausgegebene Signal auf 20 kHz begrenzt.

Die Berechnung der Frequenz erfolgt folgendermaßen:

Berechnung der Frequenz:  $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWM1XL}$

#### Ein Beispiel

Auf Kanal A soll eine Frequenz von 500 Hz, auf Kanal B 250 Hz ausgegeben werden.

- Kanal A und B aktivieren, Frequency-Mode auswählen, Prescaler auf 256 konfigurieren
- PWM1Ctrl0 = 219 (dezimal) / 11011011 (binär)
- PWM1AL = 63 (dezimal)
- Frequenz Kanal A =  $16 \text{ MHz} / 2 / 256 / 63 = 496 \text{ Hz}$
- PWM1BL = 250 (dezimal)
- Frequenz Kanal B =  $16 \text{ MHz} / 2 / 256 / 250 = 125 \text{ Hz}$

Bei Bedarf kann in jedem Zyklus der oder die Kanäle aktiviert/deaktiviert werden, dann stehen die PWM-Signale nicht durchgehend an den Ausgängen an. Im Beispiel wird die Frequenz von 500 Hz am A-Kanal nicht perfekt eingestellt. Es müsste 62,5 eingestellt werden, das ist bei einem Integer-Datentyp nicht möglich. Prüfen Sie für die gewünschte Frequenz die unterschiedlichen Prescaler-Einstellungen. Mit verschiedenen Prescaler-Einstellungen lassen sich gleiche oder ähnliche Frequenzen einzustellen.

#### NOTICE

Die Kombination aus Frequency-Mode und DHT-Sensoren ist nicht möglich. Werden ein oder mehrere DHT-Sensoren an den PiXtend-GPIOs aktiviert, so lässt sich der Frequency-Mode nicht aktivieren. Der Versuch den Frequency-Mode zu aktivieren, obwohl DHT-Sensoren verwendet werden, führt zur Deaktivierung der PWM-Ausgänge.

### 12.1.1.4.7 RetainDataOutX

Die Retain-Daten bestehen aus 32 Bytes je Richtung (RetainDataOut und RetainDataIn). Bei RetainDataOut handelt es sich um die Daten, die zur eigentlichen Sicherung vom Raspberry Pi an den PiXtend-Mikrocontroller übertragen werden. Der Speicherort der Retain-Daten ist der Mikrocontroller.

Für die 32 Bytes gibt es für das Datenformat keine Vorgabe. In der Anwendungssoftware können Sie entscheiden, wie diese Daten beschrieben und genutzt werden sollen. Unter CODESYS werden die 32 Bytes als Byte-Array angeboten.

Die RetainDataOutX werden in jedem Zyklus übertragen. Ist der Retain-Speicher aktiviert, so werden die Daten im Falle eines Stromausfalls oder einer Spannungsunterbrechung im Mikrocontroller abgespeichert und stehen beim nächsten Start in den RetainDataIn-Bytes wieder zur Verfügung.

### 12.1.1.5 Ausgangsdaten – DAC

Auf PiXtend V2 -S- befindet sich ein Digital to Analog Converter (DAC), der direkt vom Raspberry Pi angesteuert wird und für die beiden analogen Ausgänge zuständig ist. Der DAC ist nicht Teil des Mikrocontrollers und die Daten des DAC befinden sich daher nicht im Prozessabbild, das zwischen Raspberry Pi und Mikrocontroller ausgetauscht wird.

Der DAC hat ein vom Hersteller des Chips definiertes Datenformat. Für weitergehende Informationen verweisen wir Sie auf das entsprechende Datenblatt des Chips<sup>8</sup>. Auf PiXtend ist die 10 Bit Variante des Chips verbaut.

#### 12.1.1.5.1 AnalogOutX (L/H)

Der DAC nimmt ein 16 Bit Datenwort pro Kanal in Empfang. Eine „Antwort“ liefert der DAC nicht – es gibt nur eine Datenrichtung. Das X in AnalogOutX steht für den Kanal A beziehungsweise B des Chips.

Der Chip ist per SPI-Bus mit dem Raspberry Pi verbunden (ChipSelect – CS1). Zuerst wird das High-Byte (AnalogOutXH) dann das Low-Byte (AnalogOutXL) übertragen.

Auf PiXtend V2 -S- ist der Kanal A für AO0 und der Kanal B für AO1 zuständig.

##### AnalogOutXL

Bit	7	6	5	4	3	2	1	0
Name	D5	D4	D3	D2	D1	D0	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 25: Bits des AnalogOutL Bytes

Der 10 Bit Wert für den Ausgabewert steckt sowohl in AnalogOutXL und in AnalogOutXL – D0 (LSB) bis D9 (MSB). Im Low-Byte AnalogOutXL befinden sich die unteren 6 Bits. Die Bits „0“ und „1“ von AnalogOutXL werden nicht ausgewertet.

##### AnalogOutXH

Bit	7	6	5	4	3	2	1	0
Name	/A-B	-	/GA	/SHDN	D9	D8	D7	D6
Startwert	0	0	0	0	0	0	0	0

Tabelle 26: Bits des AnalogOutH Bytes

Die oberen 4 Bits des 10 Bit Ausgangswerts befinden sich in AnalogOutXH. Außerdem befinden sich drei Konfigurationsbits im AnalogOutXH-Byte:

<sup>8</sup>Microchip MCP4812

**Bit 4 – /SHDN**

Die Abkürzung SHDN bedeutet „Output Shutdown Control“. Der per Bit 7 (/A-B) eingestellte Kanal kann über das /SHDN Bit aktiviert oder deaktiviert werden. Wird eine „1“ in /SHDN geschrieben, ist der Kanal aktiv. Normalerweise sowie nach dem PowerUp ist der Kanal auf dem Wert „0“ und somit deaktiviert (Startwert). Im deaktivierten Zustand wird an den analogen Ausgängen auf PiXtend V2 -S- eine Spannung kleiner 50 mV ausgegeben. Unabhängig auf welchen Wert die Ausgänge eingestellt sind.

**Bit 5 – /GA**

Einstellung der „Output Gain Selection“. Für den vorgesehene Betriebsfall von PiXtend V2 -S- (0...10 V Ausgangsspannung) ist das /GA Bit immer auf „0“ zu setzen (Startwert).

**Bit 7 – /A-B**

Einstellung des Kanals – A oder B. Enthält das Bit eine „1“, so wird das 16 Bit Datenwort, bestehend aus AnalogOutXL und AnalogOutXH, für den Kanal B verwendet. Ist das Bit 7 auf „0“ eingestellt (Startwert), so wird es für Kanal A angesprochen.

Alle weiteren Informationen finden im Datenblatt des DAC Chips – Microchip MCP4812.

## 12.1.1.6 Eingangsdaten

Als Eingangsdaten werden die Daten bezeichnet, die der PiXtend-Mikrocontroller an den Raspberry Pi überträgt, beispielsweise die digitalen und analogen Eingänge.

### 12.1.1.6.1 Digitalln

Bit	7	6	5	4	3	2	1	0
Name	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Startwert	0	0	0	0	0	0	0	0

Tabelle 27: Bits des Digitalln Bytes

Die acht digitalen Eingänge (Digitalln) sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von DI0, das höchstwertige (MSB) den Zustand von DI7.

Die Eingänge haben im Ruhezustand immer den Wert „0“. Im aktiven Zustand (Spannung angelegt), wechselt der Wert auf „1“. Welcher Pegel zu einer „1“ oder „0“ führt, ist dem Hardware-Handbuch für PiXtend V2 -S- zu entnehmen.

Es gibt die Möglichkeit die digitalen Eingänge zu entprellen, was bei vielen Anwendungen zweckmäßig ist. Weitere Informationen finden Sie im Abschnitt 14.1.2.3.4 DigitallnDebounce.

## 12.1.1.6.2 AnalogInX (L/H)

### AnalogInXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 28: AnalogInXL

### AnalogInXH

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	MSB	
Startwert	0	0	0	0	0	0	0	0

Tabelle 29: Bits des AnalogInXH Bytes

Der im PiXtend-Controller integrierte Analog-/Digital-Wandler liefert 10 Bit Werte. Die Werte sind in den Prozessdaten in einem 16 Bit Datenwort, bestehend aus zwei Bytes, organisiert. Die höchstwertigen beiden Bits, des 10 Bit Werts, stehen im Byte AnalogInXH.

Das „X“ wird durch die Nummer des jeweiligen Analog-Kanals ersetzt („0“ oder „1“).

Die analogen Eingänge werden bereits im Mikrocontroller digital gefiltert. Jeweils 10 Werte werden aufgenommen, der Mittelwert gebildet und das Ergebnis in AnalogInX gespeichert.

Der Wertebereich liegt zwischen „0“ und „1023“. Die analogen Eingänge sind auf der Baugruppe und auf dem Edelstahlgehäuse mit AI0 und AI1 gekennzeichnet.

Um aus diesen Werten Spannungen zu berechnen, wird folgende Berechnungsformel verwendet:

Analogwert in Spannung umrechnen

$$\text{AnalogInX} * 10 / 1024 = \text{Spannung an Kanal X [V]}$$

(Spannungen werden an AI0 und AI1 gemessen)

Bei den Spannungseingängen ist der Jumper zu beachten. Die „10“ in der obigen Berechnungsformel wird verwendet, wenn kein Jumper gesetzt ist (0...10V Bereich).

Wird der Jumper für einen Kanal eingesteckt (0...5V Bereich), so ist in der Formel eine „5“ statt einer „10“ zu verwenden.

### 12.1.1.6.3 GPIOIn

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startwert	0	0	0	0	0/1	0/1	0/1	0/1

Tabelle 30: Bits des GPIOIn Bytes

Die vier GPIO-Eingänge (GPIO\_IN) sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von GPIO0. Bit 7 (MSB) bis Bit 4 sind nicht belegt.

Die GPIO-Eingänge haben keinen definierten Ruhezustand\* (floating input). Erst durch das externe Anlegen einer Spannung ergibt sich ein stabiler Zustand:

0V → Wert „0“  
5V → Wert „1“

Da die Eingänge schwimmend (floating) sind, können diese ohne externen Anschluss bei „0“ oder „1“ liegen und zwischen den Werten hin und her pendeln.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl. Weitere Informationen finden Sie im Kapitel Control und Status Bytes.

Nach einem Reset oder Power-Up von PxiTend sind die GPIOs als Eingänge konfiguriert.

\* Es gibt die Möglichkeit die GPIOs als Eingänge zu konfigurieren und per GPIOOut sogenannte PullUp-Widerstände zu aktivieren. Dadurch wird im Ruhezustand eine „1“ am Eingang erzeugt (ohne handelt es sich um Floating-Inputs). Weitere Informationen finden Sie im Abschnitt: GPIOOut

## 12.1.1.6.4 TempX (L/H)

### TempXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 31: TempXL Byte

### TempXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 32: TempXH Byte

Im 1-Wire/DHT-Mode der GPIOs enthalten die Bytes TempXL und TempXH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort enthält die Temperatur-Information eines Sensors (sofern ein Sensor angeschlossen und konfiguriert ist). Das „X“ steht für die Nummer des GPIOs, sprich „0“ bis „3“.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl. Der enthaltende Wert lässt sich einfach in einen Temperaturwert (Gleitkommawert) umwandeln, es muss jedoch zwischen DHT11 und DHT22 unterschieden werden (TempX stellt den 16 Bit Wert dar):

**DHT11: TempX / 256 = Gleitkommawert [°C]**

Beispiel: 5632 / 256 = 22,0 °C – DHT11 Sensoren liefern keine Nachkommastelle!

**DHT22: TempX / 10 = Gleitkommawert [°C]**

Beispiel: 224 / 10 = 22,4 °C

Je nach verwendeter Programmiersprache und Datentyp muss ein „typecast“ durchgeführt werden, sonst ergibt sich nach dem Teilen durch 10 (DHT22) kein Gleitkommawert, sondern der ganzzahlige Wert (im Beispiel 22 statt 22,4). Das Teilen durch 256 bei den DHT11-Sensoren kann durch einen „right shift“ um 8 Stellen erreicht werden. Bei DHT22 Sensoren können auch negative Temperaturen erfasst werden. Um dies zu erkennen muss das MSB in TempXH ausgewertet werden. Ist das MSB eine „1“, dann handelt es sich um eine negative Temperatur.

### 12.1.1.6.5 HumidX (L/H)

HumidXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 33: HumidXL Byte

HumidXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 34: HumidXH Byte

Im 1-Wire/DHT-Mode der GPIOs enthalten die Bytes HumidXL und HumidXH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort enthält die Luftfeuchtigkeits-Information eines Sensors (sofern ein Sensor angeschlossen und konfiguriert ist). Das „X“ steht für die Nummer des GPIOs, sprich „0“ bis „3“.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl.

Der enthaltene Wert kann sehr einfach in einen Luftfeuchtigkeitswert (Gleitkommawert) umgewandelt werden. Es muss jedoch zwischen DHT11 und DHT22 unterschieden werden (HumidX stellt den 16 Bit Wert dar):

**DHT11: HumidX / 256 = Gleitkommawert [%] - relative Luftfeuchtigkeit**

Beispiel:  $10496 / 256 = 41,0\%$  – DHT11 Sensoren liefern keine Nachkommastelle!

**DHT22: HumidX / 10 = Gleitkommawert [%] - relative Luftfeuchtigkeit**

Beispiel:  $417 / 10 = 41,7\%$

Je nach verwendeter Programmiersprache und Datentyp muss ein „typecast“ durchgeführt werden. Wenn nicht, dann ergibt sich nach dem Teilen durch 10 kein Gleitkommawert, sondern der ganzzahlige Wert (im Beispiel 41 statt 41,7). Das Teilen durch 256 bei den DHT11-Sensoren kann durch einen „right shift“ um 8 Stellen erreicht werden.

### 12.1.1.6.6 RetainDataInX

Die Retain-Daten bestehen aus 32 Bytes je Richtung (RetainDataOut & RetainDataIn). Bei RetainDataIn handelt es sich um die zuvor gesicherten Daten, die nach einem Stromausfall/Spannungsunterbrechung wieder vom PiXtend-Mikrocontroller an den Raspberry Pi übergeben werden. Die Daten stehen nach dem erneuten Hochfahren des PiXtend-Systems in RetainDataIn bereit, sofern zuvor der Retain-Speicher aktiviert wurde.

Die Daten bleiben so lange erhalten bis neue Retain-Daten abgespeichert werden. Die Daten können über beliebig viele Power-Cycles des Systems erhalten bleiben, wenn die Retain-Funktionalität nicht erneut aktiviert wird.

Wird der Retain-Speicher erneut aktiviert und es kommt zur Spannungsunterbrechung, werden die Daten durch die aktuellen Daten in RetainDataOut überschrieben.

## 12.1.2. Control- & Status-Bytes

PiXtend oder der PiXtend-Mikrocontroller kann über Control-Bytes konfiguriert werden. Darüber hinaus informiert der Mikrocontroller den Raspberry Pi über Status-Bytes, über den Zustand und über Fehler und Warnungen.

Auf den folgenden Seiten wird ein Überblick über die Control- und Status-Bytes vermittelt, anschließend wird jedes Byte genau beschrieben. Wir zeigen unterschiedliche Möglichkeiten auf und führen beispielhafte Berechnungen durch.

Sollten dennoch Fragen offenbleiben, so wenden Sie sich bitte per E-Mail ([support@pixtend.de](mailto:support@pixtend.de)) an uns und Sie erhalten schnellstmöglich eine Antwort und weitere Informationen.

### 12.1.2.1 Control-Bytes im Überblick

Control-Bytes werden vom Raspberry Pi an den PiXtend-Mikrocontroller übertragen. Sie lässt sich das Verhalten des Mikrocontrollers steuern.

Es gibt folgende Control-Bytes:

- **ModelOut**  
gibt an, welche Hardware-Version (Modell) die RPi-Software (z.B. CODESYS oder pxdev) erwartet
- **UCMode**  
gibt an, welcher Übertragungsmodus verwendet werden soll
- **UCCtrl**  
grundlegende Einstellungen des Mikrocontrollers (Retain, Watchdog, SafeState...)
- **DigitalInDebounce**  
gibt an, ob und wie die digitalen Eingänge zu entprellen sind
- **GPIOCtrl**  
konfiguriert die PiXtend-GPIOs (Eingang, Ausgang, DHT-Mode)
- **GPIODebounce**  
gibt an, ob und wie die GPIO-Eingänge entprellt werden sollen
- **PWM0Ctrl & PWM1Ctrl**  
konfiguriert die PWM-Kanäle (Modus, Freischaltung, Frequenzen)

### 12.1.2.2 Status-Bytes im Überblick

Status-Bytes werden vom PiXtend-Mikrocontroller an den Raspberry Pi übertragen. Sie enthalten wichtige Informationen über den Zustand von PiXtend und des Mikrocontrollers.

Es gibt folgende Status-Bytes:

- **Firmware**  
informiert über die Mikrocontroller Firmware-Version
- **Hardware**  
informiert über die PiXtend Hardware-Version
- **ModelIn**  
informiert über das PiXtend Modell
- **UCState**  
enthält den Betriebszustand & Fehlercodes des Mikrocontrollers
- **UCWarnings**  
enthält Warnungen des Mikrocontrollers

## 12.1.2.3 Beschreibung der Control-Bytes

### 12.1.2.3.1 ModelOut

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	x	x	x	x	x	x	x	x

Tabelle 35: ModelOut Byte

Durch das Byte ModelOut informiert die Anwendungssoftware, die auf dem Raspberry Pi ausgeführt wird, den Mikrocontroller welches PiXtend-Modell sie erwartet. Der Mikrocontroller entscheidet ob es sich um das erwartete Modell handelt und wenn nicht, kann er einen Fehler an den Raspberry Pi zurückgeben.  
Der Inhalt von ModelOut ist ein ASCII-Zeichen. Im Fall von PiXtend V2 -S- enthält das Byte folgenden Wert: ASCII-Zeichen S (Großbuchstabe S – 83 dezimal / 53 hexadezimal).

Für den Anwender gibt es in der Regel keinen Grund an diesem Wert etwas zu verändern. Als Beispiel werden mit dem CODESYS-Package „PiXtend V2 Professional for CODESYS“ bereits Treiber mitgeliefert, bei denen das Modell fest einprogrammiert ist und nicht verändert werden kann.

### 12.1.2.3.2 UCMode

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 36: Bits des UCMode Bytes

Aktuell (Stand Oktober 2017) gibt es nur einen Übertragungsmodus, den sogenannten „Auto-Mode“. Das Byte ist für eine zukünftige Nutzung reserviert.

Für den Betrieb von PiXtend ist es nicht notwendig hier einen Wert einzutragen. Wird ein Wert eingetragen, so hat dies keine Auswirkung.

### 12.1.2.3.3 UCCtrl

#### 12.1.2.3.3.1. UCCtrl0

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	WDE3	WDE3	WDE1	WDE0
Startwert	0	0	0	0	0	0	0	0

Tabelle 37: Bits des UCCtrl0 Bytes

Bit 0...3 – WDE0...3 (WatchdogEnable0...3)

Mit den WDE0...3 Bits lässt sich der Watchdog-Timer des PiXtend-Mikrocontrollers aktivieren und einstellen. In der Standard-Einstellung sind alle Bits „0“. Die Aktivierung und Einstellung des Watchdogs wird in der folgenden Tabelle erläutert:

WDE3	WDE2	WDE1	WDE0	Status Watchdog	konfigurierte Zeit
0	0	0	0	deaktiviert	deaktiviert
0	0	0	1	aktiv	16 ms
0	0	1	0	aktiv	32 ms
0	0	1	1	aktiv	64 ms
0	1	0	0	aktiv	0,125 s
0	1	0	1	aktiv	0,25 s
0	1	1	0	aktiv	0,5 s
0	1	1	1	aktiv	1 s
1	0	0	0	aktiv	2 s
1	0	0	1	aktiv	4 s
1	0	1	0	aktiv	8 s

Tabelle 38: Watchdog Einstellung

Ist der Watchdog aktiviert, wird die Kommunikation zwischen Raspberry Pi und PiXtend überwacht. Gibt es zwischen zwei gültigen Zyklen eine Pause, die größer als die eingestellte Zeit ist, wird der Watchdog aktiv und versetzt den Mikrocontrollers in den sicheren Zustand\*. Ein ungültiger Zyklus (zum Beispiel durch CRC-Fehler) wird vom Watchdog gewertet, als wäre kein Zyklus durchgeführt worden.

Die Verwendung des Watchdog ist sinnvoll, wenn im Fehlerfall die PiXtend-Steuerung in einen definierten Zustand gebracht werden soll. Ein Fehlerfall kann vorliegen, wenn das Anwendungsprogramm auf dem Raspberry Pi nicht reagiert und keine Daten an den Mikrocontroller übertragen werden.

#### Empfehlung

Aktivieren Sie den Watchdog nicht während der Entwicklung Ihrer Anwender-Software. Wird per CODESYS ein neues Programm übertragen oder die Anwendersoftware ge-debugged, dann findet für eine bestimmte Zeit keine Übertragung statt. Der Watchdog würde aktiv werden und ein Neustart des Systems wäre notwendig. Dies führt zu einer unnötigen Verzögerung Ihres Entwicklungsprozesses.

\* Definition „sicherer Zustand“:

- Alle digitalen Ausgänge und Relais werden abgeschaltet/in den Ruhezustand gebracht
- PWM-Ausgänge werden hochohmig (Tri-State) geschaltet
- Retain-Daten werden abgespeichert, wenn Retain aktiviert ist
- Die Status-LED „L1“ blinkt, gemäß der Fehlerursache (Fehler-LED „L1“ - Signalisierung), wenn die LED aktiviert ist
- Der Mikrocontroller beziehungsweise das PiXtend-System muss im Anschluss neu gestartet werden

Wenn das Ausschalten der digitalen Ausgänge für Ihre Anwendung nicht dem sicheren Zustand entspricht, so sind externe Maßnahmen vorzusehen, um dies abzufangen.

### 12.1.2.3.3.2. UCCtrl1

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	GPUE	SLED	RE	RC	SAFE
Startwert	0	0	0	0	0	0	0	0

Tabelle 39: Bits des UCCtrl1 Bytes

#### Bit 0 – SAFE (SafeState)

Das SAFE-Bit kann, wenn es durch den Wert „1“ aktiviert wird, den Mikrocontroller unverzüglich in den sicheren Zustand\* versetzen. Das Bit hat standardmäßig den Wert „0“

\* Definition „sicherer Zustand“:

- Alle digitalen Ausgänge und Relais werden abgeschaltet/in den Ruhezustand gebracht
- PWM-Ausgänge werden hochohmig (Tri-State) geschaltet
- Retain-Daten werden abgespeichert, wenn Retain aktiviert ist
- Die Status-LED „L1“ blinkt, gemäß der Fehlerursache, wenn die LED aktiviert ist
- Der Mikrocontroller beziehungsweise das PiXtend-System muss im Anschluss neu gestartet werden

Wenn das Ausschalten der digitalen Ausgänge für Ihre Anwendung nicht dem sicheren Zustand entspricht, so sind externe Maßnahmen vorzusehen, um dies abzufangen.

#### Bit 1 – RC (RetainCopy)

Mit dem RC-Bit lässt sich konfigurieren, welche Daten im Retain Input-Bereich (RetainDataIn0...31) sichtbar werden.

Beim Startwert „0“ werden die zuletzt gespeicherten Daten vom Mikrocontroller an den Raspberry Pi übergeben (normaler Retain Betrieb).

Wird der Wert „1“ für das RC-Bit gesetzt, so werden die zuletzt vom Raspberry Pi gesendeten Daten wieder zurückgegeben. Im Retain-Bereich RetainDataIn0...31 wird (RetainDataOut0...31) gespiegelt, jedoch mit einer Zyklus Verzögerung (Retain Copy Betrieb). Der Inhalt der Retain-Daten geht dadurch nicht verloren, er wird nur nicht mehr angezeigt, solange das RC-Bit „1“ ist.

#### Bit 2 – RE (RetainEnable)

Mit dem RE-Bit kann die Retain-Funktion des PiXtend aktiviert werden. Dazu wird eine „1“ in dieses Bit geschrieben. Standardmäßig ist RE „0“ (nach einem Reset oder Power-Up) und somit ist Retain deaktiviert.

#### Empfehlung

Aktivieren Sie Retain nur dann, wenn die Funktionalität wirklich benötigt wird. Die Anzahl der Retain-Speicherzyklen kann nur bis 10.000 Zyklen garantiert werden.

Weitere Informationen zur Retain-Funktion von PiXtend V2 -S- finden Sie im Kapitel: 6 Basis Wissen.

#### Bit 3 – SLED (StatusLED)

Die Status-LED „L1“ kann per SLED-Bit deaktiviert werden, sofern sie nicht benötigt wird. Standardmäßig ist die Status-LED aktiv und lässt sich durch den Wert „1“ in SLED deaktivieren.

#### Bit 4 – GPUE (GPIOPullUpEnable)

Mit dem GPUE-Bit lassen sich die PullUp-Widerstände der PiXtend-GPIOs freischalten (diese werden durch diesen Vorgang noch nicht aktiv). Die Freischaltung hat eine Auswirkung, wenn die GPIOs als Eingänge konfiguriert sind und im Byte GPIOOut eine „1“ für den jeweiligen GPIO geschrieben wird.  
Eine „1“ in GPUE aktiviert diese Funktionalität.

### 12.1.2.3.4 DigitalInDebounce

DigitalInDebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 40: Bits des DigitalInDebounce01 Bytes

DigitalInDebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 41: Bits des DigitalInDebounce23 Bytes

DigitalInDebounce45

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 42: Bits des DigitalInDebounce45 Bytes

DigitalInDebounce67

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 43: Bits des DigitalInDebounce67 Bytes

Mit Hilfe der DigitalInDebounce-Bytes lässt sich das Entprellen der acht digitalen Eingänge von PiXtend konfigurieren. Standardmäßig ist die Entprellung der Eingänge nicht aktiv. Das Entprellen wird jeweils gemeinsam für zwei Kanäle aktiviert/deaktiviert. Es entstehen keine Wechselwirkungen zwischen den beiden Kanälen, sie erhalten lediglich die gleiche Entprellungs-Einstellung.

Die Zahl in einem DigitalInDebounce-Byte entspricht der Zyklen-Anzahl, in der ein am digitalen Eingang anliegendes elektrisches Signal konstant bleiben muss, um die Pegeländerung an der Anwendungs-Software (Raspberry Pi) weiterzureichen. Das gilt für einen Wechsel des Signalpegels von „0“ (low) zu „1“ (high) sowie umgekehrt. Werden die Bytes nicht verändert oder wird eine „0“ als Wert geschrieben, so findet kein Entprellen statt und die Anwendungs-Software erhält in jedem Zyklus einen neuen Wert.

### Ein Beispiel

Die ersten beiden digitalen Eingänge (DIO und DI1) sollen entprellt werden. Nur Änderungen an den digitalen Eingängen, die länger als 100 ms konstant anliegen, sollen sichtbar werden. Die Zykluszeit (Datenaustausch zwischen PiXtend und Raspberry Pi) beträgt 10 ms.

- Das Byte DigitalInDebounce01 wird verwendet
- Berechnung des Wertes für DigitalInDebounce01:  $100 \text{ ms} / 10 \text{ ms} = 10$

Das Byte wird mit einer 10 (dezimal) beschrieben, um den gewünschten Effekt zu erhalten. Wird die Zykluszeit während der Software-Entwicklung verändert, muss der Wert für die Entprellung angepasst werden.

## 12.1.2.3.5 GPIOCtrl

Bit	7	6	5	4	3	2	1	0
Name	SENS3	SENS2	SENS1	SENS0	IO3	IO2	IO1	IO0
Startwert	0	0	0	0	0	0	0	0

Tabelle 44: Bits des GPIOCtrl Bytes

### Bits 0...3 – IO0...3

Mit den IO-Bits können die vier PiXtend-GPIOs als digitaler Eingang oder Ausgang konfiguriert werden. Mit dem Startwert „0“ sind alle GPIOs als Eingänge konfiguriert. Wird das Bit IO3 auf „1“ gesetzt, dann wird GPIO3 zum Ausgang. Der Wert wird über das Datenwort GPIOOut gesetzt.

### Bits 4...7 – SENS0...3 (Sensor0...3)

Die GPIOs können als Eingang/Ausgang sowei für 1-Wire Sensoren (DHT11, DHT22, AM2302) genutzt werden. Dafür stehen die oberen vier Bits des GPIOCtrl-Datenworts bereit. Wird hier das Bit SENS3 auf „1“ gesetzt, so kann an GPIO3 ein Sensor angeschlossen und ausgewertet werden. Die Einstellung des IO-Bit spielt dann keine Rolle mehr. Die SENSX-Bits setzen sich gegenüber den IOX-Bits durch.

Die Ergebnisse/Messwerte der Sensoren stehen in TempXL/TempXH und HumidXL/HumidXH. Das „X“ entspricht der Nummer des GPIOs, an dem ein Sensor angeschlossen ist.

#### NOTICE

Für die Kommunikation mit den Sensoren wird Zeit benötigt (ca. 25 ms), unter Verwendung mindestens eines Sensors ergibt sich eine minimale Zykluszeit von 30 ms. Es spielt dabei keine Rolle ob ein oder vier Sensoren abgefragt werden.

### 12.1.2.3.6 GPIODebounce

GPIODebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 45: Bits des GPIODebounce01 Bytes

GPIODebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 46: Bits des GPIODebounce23 Bytes

Das Entprellen der GPIOs ist nur möglich, wenn diese als Eingänge konfiguriert sind. Die GPIODebounce-Bytes werden wie die DigitalInDebounce-Bytes verwendet. Weitere Informationen & Beispiele befinden sich im Abschnitt: DigitalInDebounce

### 12.1.2.3.7 PWM0Ctrl – für 16 Bit PWMs

Die nachfolgenden Beschreibungen beziehen sich auf die PWM0-Kanäle (PWM0A und PWM0B) mit 16 Bit Auflösung. Die Informationen zu den 8 Bit PWM-Kanälen finden Sie im Kapitel PWM1Ctrl – für 8 Bit PWMs.

### 12.1.2.3.7.1. PWM0Ctrl0

Bit	7	6	5	4	3	2	1	0
Name	PS2	PS1	PS0	EnableB	EnableA	-	MODE1	MODE0
Startwert	0	0	0	0	0	0	0	0

Tabelle 47: Bits des PWM0Ctrl0 Bytes

#### Bit 0...1 – MODE0...1

Die PWM-Ausgänge werden in vier verschiedenen Modi betrieben. Der Modus wird über die MODE-Bits konfiguriert. Standardmäßig startet PiXtend im Servo-Mode (MODE0..1 = 00).

MODE1	MODE0	Modus	Beschreibung
0	0	Servo-Mode	Beide Kanäle (A/B) werden zur Ansteuerung von Modellbau-Servos verwendet – spezielle Signalform
0	1	Duty-Cycle-Mode	Beide Kanäle (A/B) haben die gleiche Frequenz, aber unabhängig einstellbaren Duty Cycle
1	0	Universal-Mode	Frei einstellbare Frequenz & Duty Cycle für Kanal A, halbe Frequenz für Kanal B und 50% Duty Cycle
1	1	Frequency-Mode	Frei einstellbare Frequenzen für beide Kanäle (A/B), Duty Cycle ist immer 50%

Tabelle 48: Mode Bits im PWM0Ctrl0 Byte

Servo-Mode und Duty-Cycle-Mode sind ausreichend, Universal- und Frequency-Mode können für spezielle Anwendungen sinnvoll und hilfreich sein, sind jedoch aufwändiger zu konfigurieren und zu verwenden.

#### Bit 3...4 – EnableA, EnableB

Mit den Bits EnableA und EnableB wird der jeweilige Kanal aktiviert. Standardmäßig sind die PWM-Kanäle deaktiviert. Durch das Schreiben einer „1“ in das jeweilige Bit wird der Kanal aktiviert und die PWM wird am Ausgang sichtbar. Die PWM kann zunächst für die gewünschte Anforderung konfiguriert (Modus, Frequenz/Wert...) und erst dann aktiviert werden. Die Aktivierung ist im gleichen Zyklus möglich.

### Bit 5...7 – PS0...PS2 (Prescaler0...3)

Die Prescaler-Bits (PS-Bits) ermöglichen die ungefähre Einstellung der PWM-Frequenz. Je nach PWM-Modus (siehe MODE-Bits) haben die PS-Bits verschiedene Auswirkungen.

Folgende Tabelle gibt Aufschluss über die Auswirkungen der PS-Bits:

PS2	PS1	PS0	Beschreibung	Grundfrequenz
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

Tabelle 49: Prescaler-Bits im PWM0Ctrl0 Byte

Wie bereits erwähnt haben die Prescaler-Einstellungen, abhängig vom eingestellten Modus, unterschiedliche Auswirkungen:

Modus	Beschreibung
Servo-Mode	Keine Auswirkung
Duty-Cycle-Mode	Einstellung des Prescalers bzw. der Grundfrequenz
Universal-Mode	Einstellung des Prescalers bzw. der Grundfrequenz
Frequency-Mode	Einstellung des Prescalers bzw. der Grundfrequenz

Tabelle 50: PWM Modus

Die ungefähre Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0...2) in PWM0Ctrl0, die Feineinstellung per PWM0Ctrl1L/H.

Wie die Frequenz/Periodendauer genau eingestellt werden kann, erfahren Sie im Abschnitt 16 Bit Auflösung.

#### NOTICE

Soll der Frequency-Mode verwendet werden, dürfen keine DHT-Sensoren angeschlossen und im GPIOCtrl-Byte aktiviert werden. Die Kombination aus Frequency-Mode und DHT-Sensoren ist nicht möglich. Wurden ein oder mehrere DHT-Sensoren an den PiXtend-GPIOs aktiviert, so lässt sich der Frequency-Mode nicht aktivieren. Der Versuch den Frequency-Mode trotzdem zu aktivieren, obwohl DHT-Sensoren verwendet werden, führt zur Deaktivierung der PWM-Ausgänge.

### 12.1.2.3.7.2. PWM0Ctrl1 (L/H)

Die PWM0Ctrl1-Bytes L und H enthalten ein zusammengehöriges 16 Bit Datenwort. Dabei ist PWM0Ctrl1L das Low-Byte und PWM0Ctrl1H das High-Byte.

Mit den beiden Bytes kann die PWM-Frequenz/Periodendauer eingestellt werden (im PWM-Modus „Duty-Cycle“ und „Universal“). Die Frequenz hat für beide Kanäle Gültigkeit. Im Duty-Cycle-Mode kann das Tastverhältnis unabhängig für jeden Kanal konfiguriert werden. Im Universal-Mode kann das Tastverhältnis nur für Kanal A eingestellt werden.

Für den Frequency-Mode haben die PWM1Ctrl1-Bytes keine Auswirkung.

PWM0Ctrl1L

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 51: PWM0Ctrl1L Byte

PWM0Ctrl1H

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 52: PWM0Ctrl1H Byte

Beispiel für die PWM-Periodendauer im „Duty-Cycle-Mode“:

Der Prescaler (PS0...2) wird auf den Wert 64 und damit die Grundfrequenz auf 250 kHz eingestellt. Außerdem wird der Duty-Cycle-Mode und der Kanal A aktiviert:

PWM0Ctrl0: 01101001b

Das 16 bit Datenwort PWM0Ctrl1 wird auf den Wert 1000 eingestellt:

PWM0Ctrl1L: 11101000b

PWM0Ctrl1H: 00000011b

$$\text{PWM-Periodendauer} = \text{Mikrocontroller-Basistakt} / 2 / \text{Prescaler} / \text{PWM0Ctrl1}$$

$$\text{PWM-Periodendauer} = 16 \text{ MHz} / 2 / 64 / 1000 = 125 \text{ Hz}$$

### 12.1.2.3.8 PWM1Ctrl – für 8 Bit PWMs

Die nachfolgenden Beschreibungen beziehen sich auf die PWM1-Kanäle (PWM1A & PWM1B) mit 8 Bit Auflösung. Die Informationen zu den 16 Bit PWM-Kanälen finden Sie im vorherigen Kapitel 12.1.2.3.7 PWMOCtrl-für 16 Bit PWMS.

#### 12.1.2.3.8.1. PWM1Ctrl0

Bit	7	6	5	4	3	2	1	0
Name	PS2	PS1	PS0	EnableB	EnableA	-	MODE1	MODE0
Startwert	0	0	0	0	0	0	0	0

##### Bit 0...1 – MODE0...1

Die PWM-Ausgänge werden in vier verschiedenen Modi betrieben. Der Modus wird über die MODE-Bits konfiguriert, standardmäßig startet PiXtend im Servo Modus (MODE0..1 = 00).

MODE1	MODE0	Modus	Beschreibung
0	0	Servo-Mode	Beide Kanäle (A/B) werden zur Ansteuerung von Modellbau-Servos verwendet – spezielle Signalform
0	1	Duty-Cycle-Mode	Beide Kanäle (A/B) haben die gleiche Frequenz, aber unabhängig einstellbaren Duty Cycle
1	0	Universal-Mode	Frei einstellbare Frequenz & Duty Cycle für Kanal A, halbe Frequenz für Kanal B und 50% Duty Cycle
1	1	Frequency-Mode	Frei einstellbare Frequenzen für beide Kanäle (A/B), Duty Cycle ist immer 50%

Servo-Mode und Duty-Cycle-Mode sind ausreichend, Universal- und Frequency-Mode können für spezielle Anwendungen sinnvoll und hilfreich sein, sind jedoch aufwändiger zu konfigurieren und zu verwenden.

##### Bit 3...4 – EnableA, Enable B

Mit den Bits EnableA und EnableB wird der jeweilige Kanal aktiviert. Standardmäßig sind die PWM-Kanäle deaktiviert. Durch das Schreiben einer „1“ in das jeweilige Bit wird der Kanal aktiviert und die PWM wird am Ausgang sichtbar. Die PWM kann zunächst für die gewünschte Anforderung konfiguriert (Modus, Frequenz/Wert...) und im Anschluss aktiviert werden. Die Aktivierung ist im gleichen Zyklus möglich.

### Bit 5...7 – PS0...PS2 (Prescaler0...3)

Die Prescaler-Bits (PS-Bits) ermöglichen die ungefähre Einstellung der PWM-Frequenz. Je nach PWM-Modus (siehe MODE-Bits) haben die PS-Bits verschiedene Auswirkungen.

Folgende Tabelle gibt Aufschluss über die Auswirkungen der PS-Bits:

CS2	CS1	CS0	Beschreibung	Grundfrequenz
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): 32	0,5 MHz
1	0	0	Vorteiler (Prescaler): 64	250 kHz
1	0	1	Vorteiler (Prescaler): 128	125 kHz
1	1	0	Vorteiler (Prescaler): 256	62,5 kHz
1	1	1	Vorteiler (Prescaler): 1024	15,625 kHz

Die Prescaler-Einstellungen haben abhängig vom eingestellten Modus, unterschiedliche Auswirkungen:

Modus	Beschreibung
Servo-Mode	Keine Auswirkung
Duty-Cycle-Mode	Einstellung des Prescalers bzw. der Grundfrequenz
Universal-Mode	Einstellung des Prescalers bzw. der Grundfrequenz
Frequency-Mode	Einstellung des Prescalers bzw. der Grundfrequenz

Die ungefähre Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0...2) in PWM1Ctrl0, die Feineinstellung per PWM1Ctrl1L/H.

Wie die Frequenz/Periodendauer genau eingestellt werden kann, erfahren Sie im Abschnitt: 12.1.1.4.5 PWM1X (L/H) – 8 Bit Auflösung.

#### NOTICE

Eine Kombination aus Frequency-Mode und DHT-Sensoren ist nicht möglich. Wurden ein oder mehrere DHT-Sensoren an den PiXtend-GPIOs aktiviert, dann lässt sich der Frequency-Mode nicht aktivieren. Der Versuch den Frequency-Mode trotzdem zu aktivieren, obwohl DHT-Sensoren verwendet werden, führt zur Deaktivierung der PWM-Ausgänge.

### 12.1.2.3.8.2. PWM1Ctrl1 (L/H)

Bei den PWM1-Kanälen (8 Bit PWM) existieren zwei PWM1Ctrl-Bytes (L & H) es wird jedoch nur das Low-Byte (L) verwendet.

Mit PWM1Ctrl1L kann die PWM-Frequenz/Periodendauer eingestellt werden (im PWM-Modus „Duty-Cycle“ und „Universal“). Die Frequenz hat für beide PWM1-Kanäle (A und B) Gültigkeit.

Im Duty-Cycle-Mode lässt sich das Tastverhältnis für jeden Kanal unabhängig konfigurieren. Im Universal-Mode kann das Tastverhältnis nur für Kanal A eingestellt werden. Für den Frequency-Mode haben die PWM1Ctrl1-Bytes keine Auswirkung.

PWM1Ctrl1L

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 53: Bits des PWM1Ctrl1L Bytes

PWM1Ctrl1H – wird nicht verwendet

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 54: Bits des PWM1Ctrl1H Bytes

Beispiel für die PWM-Periodendauer im „Universal-Mode“:

Der Prescaler (PS0..2) wird auf den Wert 64 und damit die Grundfrequenz auf 250 kHz eingestellt, außerdem wird der Universal-Mode und der Kanal A aktiviert:

PWM1Ctrl0: 10001010b

Das 8 bit Datenbyte PWM1Ctrl1L wird auf den Wert 150 eingestellt:

PWM1Ctrl1L: 10010110b

PWM1Ctrl1H: wird nicht verwendet

PWM-Periodendauer = Mikrocontroller-Quarz / 2 / Prescaler / PWM0Ctrl1

PWM-Periodendauer = 16 MHz / 2 / 64 / 150 = 0,833 kHz

Beispiel für die PWM-Periodendauer im „Duty-Cycle-Mode“:

Der Prescaler (PS0...2) wird auf den Wert 64 und damit auf 250 kHz eingestellt, außerdem wird der Duty-Cycle-Mode und der Kanal A aktiviert:

PWM1Ctrl0: 10001001b

Hier ist eine Besonderheit der 8 Bit PWMs im Duty-Cycle-Mode zu beachten:  
es gibt zwei verschiedene Einstellungsmöglichkeiten für PWM1Ctrl1L („0“ oder „1“)

PWM1Ctrl1L: 00000000b

PWM1Ctrl1H: wird nicht verwendet

PWM-Periodendauer = Mikrocontroller-Basistakt / Prescaler / 255 / 2

PWM-Periodendauer = 16 MHz / 64 / 255 / 2 = 490 Hz

PWM1Ctrl1L: 00000001b

PWM1Ctrl1H: wird nicht verwendet

PWM-Periodendauer = Mikrocontroller-Basistakt / Prescaler / 255

PWM-Periodendauer = 16 MHz / 64 / 255 = 980 Hz

## 12.1.2.4 Beschreibung der Status-Bytes

Die Status-Bytes informieren den Anwender und das Anwenderprogramm, das auf dem Raspberry Pi läuft, über den Status des Mikrocontrollers.

Die Status-Bytes können per CODESYS, PiXtend-C-Library oder PiXtend-Python-Library (PPL) abgefragt werden.

### 12.1.2.4.1 Firmware

Im Firmware-Byte befindet sich eine Zahl, die für die Firmware-Version steht. Die Zahl kann zwischen 0 und 255 liegen. Falls Sie Support für Ihr PiXtend V2 Gerät anfragen, so teilen Sie uns die Firmware-Version bitte immer mit.

### 12.1.2.4.2 Hardware

Das Hardware-Byte enthält die Versionsnummer Ihres PiXtend-Boards. Mit der Versionsnummer kann die Anwender-Software prüfen, ob Software und Treiber zur vorhandenen Hardware passen.

Beispiel:

Hardware-Byte enthält die Zahl 21  
→ PiXtend Hardware Version 2.1

### 12.1.2.4.3 ModelIn

Im ModelIn-Byte informiert der Mikrocontroller die Anwender-Software über das PiXtend V2 Modell. Mit der Modell-Kennung kann die Anwendersoftware prüfen, ob Software und Treiber zur vorhandenen Hardware passen.

Der Inhalt von ModelIn ist ein ASCII-Zeichen. Im Fall von PiXtend V2 -S- enthält das Byte folgenden Wert: ASCII-Zeichen S (Großbuchstabe S – 83 dezimal / 53 hexadezimal).

## 12.1.2.4.4 UCState

Bit	7	6	5	4	3	2	1	0
Name	ERR3	ERR2	ERR1	ERR0	-	-	-	RUN
Startwert	0	0	0	0	0	0	0	0

Tabelle 55: Bits des UCState Bytes

### Bit 0 – RUN

Durch Abfrage dieses Bits kann der Anwender prüfen, ob PiXtend betriebsbereit („1“) ist beziehungsweise ob eine erfolgreiche Datenübertragung durchgeführt wurde. Das kann nach dem Start des PiXtend-Systems sinnvoll sein. Wird der Controller in den sicheren Zustand gebracht (zum Beispiel durch den Watchdog), so enthält das Bit eine „0“. Das lässt sich nicht mehr abfragen, da keine Kommunikation möglich ist, bis zu einem Neustart.

### Bit 4...7 – ERR0...3

Die ERR-Bits informieren über Probleme, die der Mikrocontroller erkennt und an das Anwenderprogramm weiterreicht.

Die nachfolgende Tabelle beschreibt die Fehlercodes und deren Ursache:

ERR3	ERR2	ERR1	ERR0	Fehlerbeschreibung
0	0	0	0	Kein Fehler
0	0	0	1	-
0	0	1	0	CRC-Fehler - Data die vom Raspberry Pi erhaltenen Nutzdaten sind fehlerhaft
0	0	1	1	Datenblock zu kurz Treiber fehlerhaft/falsches Modell eingestellt
0	1	0	0	PiXtend Modell stimmt nicht überein das erwartete und das tatsächliche Modell stimmen nicht überein (ModelIn und ModelOut sind ungleich)
0	1	0	1	CRC-Fehler - Header die vom Raspberry Pi erhaltenen Headerdaten sind fehlerhaft
0	1	1	0	SPI-Frequenz zu hoch die eingestellte SPI-Frequenz ist zu hoch, es kommt zu Übertragungsfehlern

Tabelle 56: Aufschlüsselung Error-Bits im UCState Byte

## 12.1.2.4.5 UCWarnings

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	W1	W0	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 57: Bits des UCWarnings Bytes

### Bit 1...2 – W0...1 (RetainCRCError, RetainVoltageError)

Die WX-Bits informieren den Anwender über Warnungen in Bezug auf die Retain-Funktionalität von PiXtend V2. Im Anwenderprogramm kann auf diese Bits abgeprüft werden. So lässt sich erkennen, ob der Retain-Speicher korrekt arbeitet oder nicht.

#### W0 – RetainCRCError

Ist „1“, wenn bei der Überprüfung des Retain-Speichers ein Fehler erkannt wurde.

#### W1 – RetainVoltageError

Ist „1“, wenn die aktuelle Versorgungsspannung von PiXtend zu gering ist (kleiner 19V). Die Retain-Speicherung kann nicht aktiviert werden.

## 12.2. PiXtend V2 -L-

### 12.2.1. Prozessdaten

#### 12.2.1.1 Prozessdaten im Überblick

Es wird zwischen zwei Arten von Prozessdaten unterschieden:

- Prozessdaten die vom Raspberry Pi an PiXtend übertragen werden
- Prozessdaten die von PiXtend an den Raspberry Pi übertragen werden

#### 12.2.1.1.1 Prozessdaten die vom RPi an PiXtend übertragen werden

Daten für digitale Ausgänge, Relais und GPIOs (als Ausgänge) PWM-Daten, Retain-Daten

- DigitalOutX
- RelayOut
- GPIOOut
- PWMYX (L/H)
- RetainDataOutX

#### 12.2.1.1.2 Prozessdaten die vom RPi an den PiXtend-DAC übertragen werden

Daten für analoge Ausgänge

- AnalogOutX (L/H)

#### 12.2.1.1.3 Prozessdaten die von PiXtend an den RPi übertragen werden

Daten der digitalen und analogen Eingänge, GPIOs (als Eingänge)

Temperatur und Luftfeuchtigkeit von DHT11/22- bzw. AM2302-Sensoren, Retain-Daten

- DigitalInX
- AnalogInX (L/H)
- GPIOIn
- TempX (L/H)
- HumidX (L/H)
- RetainDataInX

## 12.2.1.2 SPI-Bus Protokoll Übersicht

INPUT			OUTPUT			
	MC (Out) --> RasPi (In)			RasPi (Out) --> MC (In)		
	Name	used Bits	Byte Idx	Name	used Bits	
HeaderIn	Firmware	8	0	HeaderOut	ModelOut	8
	Hardware	8	1		UCMode	8
	Modelln	8	2		UCCtrl0	8
	UCState	8	3		UCCtrl1	8
	UCWarnings	8	4		Reserved	0
	Reserved	0	5		Reserved	0
	Reserved	0	6		Reserved	0
	CRCHeaderInL	8	7		CRCHandlerOutL	8
	CRCHeaderInH	8	8		CRCHandlerOutH	8
DataIn	DigitalIn0	8	9	DataOut	DigitalInDebounce01	8
	DigitalIn1	8	10		DigitalInDebounce23	8
	AnalogIn0L	8	11		DigitalInDebounce45	8
	AnalogIn0H	2	12		DigitalInDebounce67	8
	AnalogIn1L	8	13		DigitalInDebounce89	8
	AnalogIn1H	2	14		DigitalInDebounce1011	8
	AnalogIn2L	8	15		DigitalInDebounce1213	8
	AnalogIn2H	2	16		DigitalInDebounce1415	8
	AnalogIn3L	8	17		DigitalOut0	8
	AnalogIn3H	2	18		DigitalOut1	4
	AnalogIn4L	8	19		RelayOut	4
	AnalogIn4H	2	20		GPIOCtrl	8
	AnalogIn5L	8	21		GPIOOut	4
	AnalogIn5H	2	22		GPIODebounce01	8
	GPIOIn	4	23		GPIODebounce23	8
	Temp0L	8	24		PWM0Ctrl0	8
	Temp0H	8	25		PWM0Ctrl1L	8
	Humid0L	8	26		PWM0Ctrl1H	8
	Humid0H	8	27		PWM0AL	8
	Temp1L	8	28		PWM0AH	8
	Temp1H	8	29		PWM0BL	8
	Humid1L	8	30		PWM0BH	8
	Humid1H	8	31		PWM1Ctrl0	8
	Temp2L	8	32		PWM1Ctrl1L	8
	Temp2H	8	33		PWM1Ctrl1H	8

Humid2L	8	34		PWM1AL	8
Humid2H	8	35		PWM1AH	8
Temp3L	8	36		PWM1BL	8
Temp3H	8	37		PWM1BH	8
Humid3L	8	38		PWM2Ctrl0	8
Humid3H	8	39		PWM2Ctrl1L	8
Reserved	0	40		PWM2Ctrl1H	8
Reserved	0	41		PWM2AL	8
Reserved	0	42		PWM2AH	8
Reserved	0	43		PWM2BL	8
Reserved	0	44		PWM2BH	8
RetainDataIn0 ... 63	all	45		RetainDataOut0 ... 63	all
CRCDataInL	8	109		CRCDataOutL	8
CRCDataInH	8	110		CRCDataOutH	8

Tabelle 58: PiXtend V2 -L- SPI-Protokoll Übersicht

Jedes Byte vom SPI-Protokoll wird in diesem Kapitel beschrieben, zum Teil mit Anwendungshinweisen und Berechnungsbeispielen, siehe beispielsweise bei den PWMs.

Das SPI-Protokoll für das PiXtend V2 -L- umfasst insgesamt 111 Bytes, die Tabelle 60 zeigt eine verkürzte Form, hier wurden die 64 Bytes des Retain-Speichers zusammengefasst. Handelt es sich bei einem Eintrag um ein einzelnes Byte, so steht im SPI-Protokoll nur der Name oder die Bezeichnung des Bytes.

Bei 16 Bit Werten (2 Bytes), beispielsweise Temperatur, wird der Wert auf 2 einzelne Bytes aufgeteilt. Dies ist an den Zusätzen „L“ für Low-Byte und „H“ für High-Byte zu erkennen. Implementieren Sie das SPI-Protokoll, achten Sie darauf, dass die Bytes für High und Low entsprechend zugeordnet sind.

Die CRC-Berechnung für den Header und die Daten werden nicht explizit aufgeführt. Bitte entnehmen Sie diese Information dem Sourcecode für unsere Linux Tools (pxdev), zu finden auf SourceForge, unserer Homepage im Downloadbereich oder unseren Python Modulen. Sie finden eine entsprechende Implementierung die Sie als Vorlage für Ihre Programmiersprache verwenden können.

### 12.2.1.3 SPI-Bus Konfiguration

Um die PiXtend V2 SPI Devices aus eigenen Applikationen ansprechen zu können müssen folgende SPI-Bus Einstellungen beachtet werden:

- SPI-Mode 0 (CPOL = 0, CPHA = 0)
- SPI Geschwindigkeit: 700 kHz
- SPI ChipSelect 0 (CS0): PiXtend V2 Mikrocontroller
- SPI ChipSelect 1(CS1): PiXtend DAC bzw. CAN-Interface (bei V2 -L-)
- GPIO24, wiringPi BCM Nummerierung (wiringPi's eigene Nummerierung GPIO 5, Name: GPIO. 5) des Raspberry Pi muss auf "high" / logisch "1" gesetzt werden → SPI Enable.

#### Zykluszeitbeschränkung:

- Minimale Zykluszeit V2 -S-: 2,5 ms
- Minimale Zykluszeit V2 -L-: 5 ms

Bei dieser Angabe handelt es sich um den zeitlichen Mindestabstand zwischen zwei Kommunikationsvorgängen mit dem Mikrocontroller. Längere Zykluszeiten sind besser.

Wir empfehlen den Sourcecode der PiXtend Bibliothek und/oder der Python Module zu studieren um ein besseres Gefühl für die Nutzung des SPI-Busses zu erhalten.

## 12.2.1.4 Ausgangsdaten

Als Ausgangsdaten bezeichnen wir die Daten, die der Raspberry Pi an den PiXtend-Mikrocontroller überträgt, zum Beispiel digitale Ausgänge oder Relais.

### 12.2.1.4.1 DigitalOutX

Die digitalen Ausgänge sind in zwei Bytes organisiert. Das Byte DigitalOut0 enthält die ersten acht Ausgänge (D00 – D07). Das Byte DigitalOut1 die verbleibenden vier Ausgänge (D08 – D011).

#### DigitalOut0

Bit	7	6	5	4	3	2	1	0
Name	D07	D06	D05	D04	D03	D02	D01	D00
Startwert	0	0	0	0	0	0	0	0

Tabelle 59: Bits des DigitalOut0 Bytes

#### DigitalOut0 - Bit 0...7

Das niedwertigste Bit (LSB) enthält den Zustand von D00. Das höchstwertige Bit (MSB) enthält den Zustand von D07.

#### DigitalOut1

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	D011	D010	D09	D08
Startwert	0	0	0	0	0	0	0	0

Tabelle 60: Bits des DigitalOut1 Bytes

#### DigitalOut1 - Bit 0...3

Das niedwertigste Bit (LSB) enthält den Zustand von D08. Bit 7 (MSB), Bit 6, Bit 5 und Bit 4 sind nicht belegt.

Der Startwert nach dem Power-Up oder Reset des Mikrocontrollers ist für die DigitalOutX-Bytes „0“. Die Ausgänge sind im Start-/SafeState deaktiviert. Durch das Schreiben einer „1“ in eines der DO-Bits wird der entsprechende Ausgang aktiviert. Die zugehörige LED auf dem PiXtend-Board leuchtet auf.

Die digitalen Ausgänge auf PiXtend V2 verfügen über jeweils eine separate Einspeisung für vier Ausgänge:

D00 – D03: VCC-D00-3

D04 – D07: VCC-D04-7

D08 – D011: VCC-D08-11

Die Versorgungsanschlüsse befinden sich auf dem Klemmenblock, auf dem auch die vier Ausgänge untergebracht sind. An der Leuchtdiode „VCC-DOx-x“ lässt sich erkennen, ob eine Versorgungsspannung an die Einspeisung angeschlossen ist.

Die LEDs der digitalen Ausgänge leuchten auch dann, wenn die VCC-DO-Versorgung nicht angeschlossen wurde. Eine Spannung an den DO-Pins „wird nicht sichtbar“. Weitere Infos finden Sie im Hardware-Handbuch von PiXtend V2 -L-.

Um die Funktion der Ausgänge zu testen, kann in die DigitalOutX-Bytes der Wert 255 (dezimal) bzw. 0xFF (hex) geschrieben werden. Alle zwölf Ausgänge werden aktiviert, dass die Bits 4 bis 7 von DigitalOut1 gesetzt werden, hat keine Auswirkung.

#### 12.2.1.4.2 RelayOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	RELAY3	RELAY2	RELAY1	RELAY0
Startwert	0	0	0	0	0	0	0	0

Tabelle 61: Bits des RelayOut Bytes

##### Bit 0...3

Die vier Relais-Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von RELAY0. Bit 7 (MSB) bis Bit 4 sind nicht belegt. Als Startwert nach dem Power-Up oder Reset des Mikrocontrollers ist das gesamte RelayOut-Byte „0“. Die Relais sind im Start-/SafeState deaktiviert.

Durch das Schreiben einer „1“ in eines der Bits (0...3) wird das entsprechende Relais aktiviert. Die zugehörige LED auf dem PiXtend-Board leuchtet auf und das Relais schaltet hörbar.

Um die Funktion der Relais zu testen, kann für das RelayOut-Byte der Wert 255 (dezimal) bzw. 0xFF (hex) geschrieben werden. Alle vier Relais werden aktiviert. Dass dabei die Bits 4...7 gesetzt werden, hat keine Auswirkung.

### 12.2.1.4.3 GPIOOut

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startwert	0	0	0	0	0	0	0	0

Tabelle 62: Bits des GPIOOut Bytes

#### Bit 0...3

Die vier GPIO-Ausgänge sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von GPIO0. Bit 7 (MSB) bis Bit 4 sind nicht belegt. Als Startwert nach dem Power-Up oder Reset des Mikrocontrollers ist das gesamte GPIOOut-Byte „0“. Die GPIOs sind im Start-/SafeState deaktiviert.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl. Weitere Informationen finden Sie im Kapitel: 12.2.2 Control- und Status-Bytes.

Durch das Schreiben einer „1“ in eines der Bits (0...3) wird der entsprechende GPIO aktiviert (sofern als Ausgang konfiguriert). Sind der oder die GPIOs als Eingang oder für die genannten Sensoren konfiguriert, so hat das Verändern der Werte in GPIOOut keine Auswirkung\*.

Um die Funktion der GPIOs zu testen, können diese als Ausgänge konfiguriert werden. Anschließend kann für das GPIOOut-Byte der Wert 255 (dezimal) oder 0xFF (hex) geschrieben werden. Alle vier GPIO-Ausgänge werden aktiviert, Dass dabei die Bits 4...7 gesetzt werden, hat keine Auswirkung.

\* Ist ein GPIO als Eingang mit PullUps konfiguriert und das zugehörige Bit wird in GPIOOut auf „1“ gesetzt, wird ein PullUp-Widerstand an diesem GPIO aktiviert. Damit bekommt der GPIO-Eingang einen hochohmigen (~35 kOhm) Bezug zu 5V und floatet nicht mehr. Ohne externe Beschaltung verharret der Eingang auf High-Pegel/Level. Standardmäßig handelt es sich bei den GPIO-Eingängen um floating-Inputs ohne definierten Pegel (ohne externe Beschaltung). Die PullUp-Widerstände sind generell über das Bit „GPIOPullUpEnable“ zu aktivieren.

### 12.2.1.4.4 PWMYX – 16 Bit Auflösung

Die PWM-Einheiten von PiXtend können in vier verschiedenen Modi betrieben werden. In jedem Modus haben die PWM-Prozessdaten eine andere Auswirkung. Aus diesem Grund gehen wir in diesem Abschnitt auf die vier Modi „Servo-Mode“, „Duty-Cycle-Mode“, „Universal-Mode“ und „Frequency-Mode“ separat ein.

Weitere Informationen zur Konfiguration und Modi der PWM-Ausgänge finden Sie im Abschnitt: 12.2.2.3.7 PWMYCtrl – für 16 Bit PWMs.

Bei den nachstehend beschriebenen PWM-Bytes handelt es sich um die Prozessdaten der sechs 16 Bit PWM-Kanäle, die auf der Baugruppe die volle Bezeichnung PWM0A (P0A), PWM0B (P0B), PWM1A (P1A), PWM1B (P1B), PWM2A (P2A) und PWM2B (P2B) tragen.

Wir sprechen bei zwei Kanälen mit der gleichen Nummer von einer Gruppe, so bilden beispielsweise PWM0A und PWM0B die erste PWM-Gruppe. Wird eine Gruppe in einen Modus versetzt, sind beide Kanäle A und B, in diesem Modus. Ein einzelner Kanal einer Gruppe kann nicht in einen anderen Modus versetzt werden, immer nur die gesamte Gruppe.

### 12.2.1.4.4.1. Servo-Mode

Der Servo-Modus ist speziell auf die Anforderungen von Modellbauservos ausgelegt. Die Periodendauer ist fest auf 50 Hz (20 ms) eingestellt. Zu Beginn der Periode ist das Signal mindestens 1 ms (Minimalausschlag) oder maximal 2 ms (Maximalausschlag) auf High-Pegel. Die verbleibende Zeit ist das Signal immer auf Low-Pegel.

Die PW<sub>Y</sub>X-Bytes L und H enthalten ein zusammengehöriges 16 Bit Datenwort. Dabei ist PW<sub>Y</sub>X<sub>L</sub> das Low-Byte und PW<sub>Y</sub>X<sub>H</sub> das High-Byte. Das „Y“ wird durch die Nummer der Gruppe ersetzt („0“, „1“ oder „2“). Das „X“ wird durch den Buchstaben des jeweiligen PWM-Kanals ersetzt („A“ oder „B“). So lassen sich alle sechs PWM-Ausgänge nach dem gleichen Prinzip abbilden.

PW<sub>Y</sub>X<sub>L</sub>

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 63: PW<sub>Y</sub>X<sub>L</sub> BytePW<sub>Y</sub>X<sub>H</sub>

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 64: PW<sub>Y</sub>X<sub>H</sub> Byte

#### Einstellbarer Wertebereich

kleinster Wert: 0 (dezimal) → 1 ms, Minimalausschlag

größter Wert: 16000 (dezimal) → 2 ms, Maximalausschlag

Werte größer 16000 werden vom Controller begrenzt und wirken wie 16000. Im Bereich zwischen 0 und 16000 ist die Bewegung des Servos linear. Der Startwert der PW<sub>Y</sub>X-Bytes, nach einem Power-Up oder Reset, ist „0“.

### 12.2.1.4.4.2. Duty-Cycle-Mode

Im Duty-Cycle-Mode enthalten die Bytes PWM0XL und PWM0XH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort wird für die Einstellung des Tastverhältnisses (englisch: duty cycle) verwendet.

Das „Y“ wird durch die Nummer der Gruppe ersetzt („0“, „1“ oder „2“). Das „X“ wird durch den Buchstaben des jeweiligen PWM-Kanals ersetzt („A“ oder „B“). Jedem Kanal kann ein separater Duty-Cycle zugewiesen werden. Die Frequenz und die Periodendauer werden gemeinsam für beide Kanäle einer Gruppe eingestellt.

Der Modus eignet sich optimal um mehrere Antriebe oder Gebläse, unabhängig von deren Geschwindigkeit, von 0% bis 100% zu steuern – mit gleicher Frequenz.

PWMYXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 65: PWMYXL Byte

PWMYXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 66: PWMYXH Byte

Welcher Duty-Cycle sich durch einen bestimmten Wert ergibt, hängt von den PWMYXL/H-Werten ab und von den Control-Bytes PWMYCtrl1L und PWMYCtrl1H. Die ungefähre Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0...2) in PWMYCtrl0, die Feineinstellung per PWMYCtrl1L/H.

Weitere Informationen zur Konfiguration der PWMYX-Kanäle erfahren Sie im Abschnitt: 12.2.2.3.7 PWMYCtrl – für 16 Bit PWMs.

## Berechnungsformeln – PWM – 16 Bit

Frequenz:  $f = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWMYCtrl1}$

Duty-Cycle:  $\%_{\text{on}} = 100 \% * \text{PWMYX} / \text{PWMYCtrl1}$

### Ein Beispiel

Duty-Cycle Mode und die Kanäle 0A und 0B aktivieren, Prescaler auf 1024:

PWM0Ctrl0 = 249 (dezimal) bzw. 11111001b (binär)

In PWM0Ctrl1L/H wird folgender 16 Bit Wert geschrieben: 5000 (dezimal).

In PWM0XL/H wird der 16 Bit Wert 2500 (dezimal) geschrieben. Der Duty-Cycle beträgt hiermit 50%.

Wird in PWM0XL/H ein größerer Wert als in PWM0Ctrl1L/H geschrieben, so verbleibt der PWM-Kanal durchgehend auf logisch „1“. Durchgehend logisch „0“ wird durch den Wert „0“ in PWM0XL/H erreicht.

In diesem Beispiel ergibt sich eine Frequenz von 1,56 Hz.

Wir empfehlen die Implementierung der PWM-Funktionen in Ihren Programmen zunächst ohne angeschlossene Geräte und Aktoren zu testen und mit einem Oszilloskop zu überprüfen.

### Setzen Sie die PWM-Werte mit Bedacht!

#### ⚠ CAUTION

Aktoren können durch falsche Frequenzen oder Duty-Cycles beschädigt oder zerstört werden. Überprüfen Sie Ihre Programme zunächst mit einem Oszilloskop, bevor reale Geräte an die PWM-Kanäle angeschlossen werden.

### 12.2.1.4.4.3. Universal-Mode

Im Universal-Mode enthalten die Bytes PWMYAL und PWMYAH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort wird für die Einstellung des Tastverhältnisses beziehungsweise Duty-Cycle von Kanal A verwendet.

Der B-Kanal ist in diesem Modus nicht einstellbar. Er gibt eine PWM mit 50% Duty-Cycle und der halben Frequenz des A-Kanals aus. Das „Y“ wird durch die Nummer der Gruppe ersetzt („0“, „1“ oder „2“).

Der Modus ist für Anwendungen mit unterschiedlichen Frequenzen geeignet und wenn ein Kanal des Duty-Cycles konfiguriert werden soll.

PWMYAL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 67: PWMYAL Byte

PWMYAH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 69: PWMYAH Byte

Die Konfiguration des A-Kanals läuft wie beim Duty-Cycle-Mode ab.

Eine Änderung der Bytes PWMYBL und PWMYBH hat keine Auswirkung. Der B-Kanal hängt von der Frequenz-Konfiguration des A-Kanals ab.

#### 12.2.1.4.4.4. Frequency-Mode

Im Frequency-Mode enthalten die Bytes PWM<sub>Y</sub>X<sub>L</sub> und PWM<sub>Y</sub>X<sub>H</sub> ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort wird für die Einstellung der Frequenz des jeweiligen Kanals verwendet. Das „Y“ wird durch die Nummer der Gruppe ersetzt („0“, „1“ oder „2“). Das „X“ wird durch den Buchstaben des jeweiligen PWM-Kanals ersetzt („A“ oder „B“).

Auf allen Kanälen, die auf Frequency-Mode konfiguriert sind, ist der Duty-Cycle auf 50% festgesetzt und kann nicht verändert werden.

Der Modus eignet sich optimal, wenn viele unterschiedliche Frequenzen benötigt werden, der Duty-Cycle aber keine Rolle spielt. Dies ist beispielsweise der Fall, wenn mit den PWM-Kanälen Schrittmotortreiber angesteuert werden, die lediglich auf Signalflanken reagieren und keinen variablen Duty-Cycle benötigen. So kann mit PiXtend V2 -L- die Geschwindigkeit von bis zu sechs Schrittmotoren gesteuert werden.

#### PWM<sub>Y</sub>X<sub>L</sub>

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 70: Bits des PWM<sub>Y</sub>X<sub>L</sub> Bytes

#### PWM<sub>Y</sub>X<sub>H</sub>

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 70: Bits des PWM<sub>Y</sub>X<sub>H</sub> Bytes

Die ungefähre Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0..2) in PWM<sub>Y</sub>Ctrl0, die Feineinstellung per PWM<sub>Y</sub>X<sub>L/H</sub>.

Die maximale Frequenz in diesem Modus liegt bei 20 kHz. Bei der Einstellung von höheren Frequenzen wird das ausgegebene Signal auf 20 kHz begrenzt.

Die Frequenz wird wie folgt berechnet:

$$\text{Frequenz Kanal X} = 16 \text{ MHz} / 2 / \text{Prescaler} / \text{PWMYX}$$

### Ein Beispiel

Es soll auf Kanal 0A eine Frequenz von 500 Hz, auf Kanal 0B 250 Hz ausgegeben werden (bei der PWM-Gruppe 0).

- Kanal A und B aktivieren, Frequency-Mode auswählen, Prescaler auf 64 konfigurieren
- PWM0Ctrl0 = 123 (dezimal) bzw. 01111011b (binär)
- PWM0A (16 Bit Datenwort) = 250 (dezimal)
- Frequenz Kanal A =  $16 \text{ MHz} / 2 / 64 / 250 = 500 \text{ Hz}$
- PWM0B (16 Bit Datenwort) = 500 (dezimal)
- Frequenz Kanal B =  $16 \text{ MHz} / 2 / 64 / 500 = 250 \text{ Hz}$

Bei Bedarf können in jedem Zyklus der oder die Kanäle aktiviert oder deaktiviert werden, wenn die PWM-Signale nicht durchgehend an den Ausgängen anstehen sollen,

#### **NOTICE**

Eine Kombination aus Frequency-Mode und DHT-Sensoren ist nicht möglich. Wurden ein oder mehrere DHT-Sensoren an den PiXtend-GPIOs aktiviert, so lässt sich der Frequency-Mode nicht aktivieren. Der Versuch den Frequency-Mode trotzdem zu aktivieren, obwohl DHT-Sensoren verwendet werden, führt zur Deaktivierung der PWM-Ausgänge.

### 12.2.1.4.5 RetainDataOutX

Die Retain-Daten bestehen aus 64 Bytes je Richtung (RetainDataOut und RetainDataIn). Bei RetainDataOut handelt es sich um die Daten, die zur eigentlichen Sicherung vom Raspberry Pi an den PiXtend-Mikrocontroller übertragen werden. Der Speicherort der Retain-Daten ist der Mikrocontroller.

Es gibt keine Vorgabe für das Datenformat der 64 Bytes. In der Anwendungs-Software entscheiden Sie, wie die Daten beschrieben und genutzt werden. Unter CODESYS werden die 64 Bytes als Byte-Array angeboten.

Die RetainDataOutX werden in jedem Zyklus übertragen. Ist der Retain-Speicher aktiviert, so werden die Daten bei einem Stromausfall oder einer Spannungsunterbrechung im Mikrocontroller abgespeichert. Sie stehen beim nächsten Start in den RetainDataIn-Bytes wieder zur Verfügung.

## 12.2.1.5 Ausgangsdaten – DAC

Auf PiXtend V2 -L- befindet sich ein Digital to Analog Converter (DAC), der vom Raspberry Pi direkt angesteuert wird und für die beiden analogen Ausgänge zuständig ist. Der DAC ist nicht Teil des Mikrocontrollers und die Daten des DAC befinden sich nicht im Prozessabbild, das zwischen Raspberry Pi und Mikrocontroller ausgetauscht wird.

Der DAC hat ein vom Hersteller des Chips definiertes Datenformat, das nachstehend erläutert wird. Für weitergehende Informationen verweisen wir Sie auf das Datenblatt des Chips<sup>9</sup>. Auf PiXtend ist die 10 Bit Variante des Chips verbaut.

### 12.2.1.5.1 AnalogOutX (L/H)

Der DAC nimmt ein 16 Bit Datenwort pro Kanal in Empfang. Eine „Antwort“ liefert der DAC nicht – es gibt nur eine Datenrichtung. Das X in AnalogOutX steht für den Kanal A beziehungsweise B des Chips.

Der Chip ist per SPI-Bus mit dem Raspberry Pi verbunden (ChipSelect – CS1). Zuerst wird das High-Byte (AnalogOutXH) dann das Low-Byte (AnalogOutXL) übertragen.

Auf PiXtend V2 -L- ist der Kanal A für A00 und der Kanal B für A01 zuständig.

AnalogOutXL

Bit	7	6	5	4	3	2	1	0
Name	D5	D4	D3	D2	D1	D0	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 71: Bits des AnalogOutXL Bytes

Der 10 Bit Wert für den Ausgabewert findet sich in AnalogOutXL und in AnalogOutXL – D0 (LSB) bis D9 (MSB). Im Low-Byte AnalogOutXL befinden sich die unteren 6 Bits. Die Bits „0“ und „1“ von AnalogOutXL werden nicht ausgewertet.

---

<sup>9</sup>Microchip MCP4812

## AnalogOutXH

Bit	7	6	5	4	3	2	1	0
Name	/A-B	-	/GA	/SHDN	D9	D8	D7	D6
Startwert	0	0	0	0	0	0	0	0

Tabelle 72: Bits des AnalogOutXH Bytes

Die oberen 4 Bits des 10 Bit Ausgangswerts befinden sich in AnalogOutXH.

Die drei Konfigurationsbits sind im AnalogOutXH-Byte:

**Bit 4 – /SHDN**

Die Abkürzung SHDN bedeutet „Output Shutdown Control“. Der per Bit 7 (/A-B) eingestellte Kanal kann über das /SHDN Bit aktiviert oder deaktiviert werden. Wird eine „1“ in /SHDN geschrieben, ist der Kanal aktiv.

Nach dem PowerUp ist der Kanal auf dem Wert „0“ und somit deaktiviert (Startwert). Im deaktivierten Zustand wird an den analogen Ausgängen auf PiXtend V2 -L- eine Spannung kleiner 50 mV ausgegeben, unabhängig auf welchen Wert die Ausgänge eingestellt sind.

**Bit 5 – /GA**

Einstellung der „Output Gain Selection“. Für den vorgesehene Betriebsfall von PiXtend V2 -L- (0...10V Ausgangsspannung) ist das /GA Bit immer auf „0“ zu setzen (Startwert).

**Bit 7 – /A-B**

Einstellung des Kanals – A oder B. Enthält das Bit eine „1“, so wird das 16 Bit Datenwort, bestehend aus AnalogOutXL und AnalogOutXH, für den Kanal B verwendet. Ist das Bit 7 auf „0“ eingestellt (Startwert), wird es für Kanal A angesprochen.

Alle weiteren Informationen finden Sie im Datenblatt des DAC Chips – Microchip MCP4812.

## 12.2.1.6 Eingangsdaten

Als Eingangsdaten werden die Daten bezeichnet, die der PiXtend-Mikrocontroller an den Raspberry Pi überträgt, im Beispiel sind das digitale und analoge Eingänge.

### 12.2.1.6.1 DigitalInX

#### DigitalIn0

Bit	7	6	5	4	3	2	1	0
Name	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0
Startwert	0	0	0	0	0	0	0	0

Tabelle 73: Bits des DigitalIn0 Bytes

Die ersten acht digitalen Eingänge (DigitalIn0) sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von DI0, das höchstwertige (MSB) den Zustand von DI7.

#### DigitalIn1

Bit	7	6	5	4	3	2	1	0
Name	DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8
Startwert	0	0	0	0	0	0	0	0

Tabelle 74: Bits des DigitalIn1 Bytes

Die weiteren acht digitalen Eingänge (DigitalIn1) sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von DI8, das höchstwertige (MSB) den Zustand von DI15.

Die Eingänge haben im Ruhezustand den Wert „0“. Im aktiven Zustand (Spannung angelegt), wechselt der Wert auf „1“. Welcher Pegel zu einer „1“ oder „0“ führt, ist dem Hardware-Handbuch für PiXtend V2 -L- zu entnehmen.

Es gibt die Möglichkeit die digitalen Eingänge zu entprellen, das ist bei vielen Anwendungen sinnvoll. Weitere Informationen finden Sie im Abschnitt: DigitalInBebounce.

### 12.2.1.6.2 AnalogInX (L/H)

#### AnalogInXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 75: AnalogInXL

#### AnalogInXH

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	MSB	
Startwert	0	0	0	0	0	0	0	0

Tabelle 76: Bits des AnalogInXH Bytes

Der im PiXtend-Controller integrierte Analog-/Digital-Wandler liefert 10 Bit Werte. Die Werte sind in den Prozessdaten in einem 16 Bit Datenwort, bestehend aus zwei Bytes, organisiert. Die höchstwertigen beiden Bits, des 10 Bit Werts, stehen im Byte AnalogInXH.

Das „X“ wird durch die Nummer des jeweiligen Analog-Kanals ersetzt („0“ oder „1“). Die analogen Eingänge werden bereits im Mikrocontroller digital gefiltert. Es werden jeweils 10 Werte aufgenommen, der Mittelwert gebildet und das Ergebnis in AnalogInX gespeichert.

Der Wertebereich liegt zwischen „0“ und „1023“. Die analogen Eingänge sind auf der Baugruppe und auf dem Edelstahlgehäuse mit AI0, AI1, AI2, AI3, AI4 und AI5 gekennzeichnet. Dabei sind AI0 – AI3 Spannungseingänge (0...5V oder 0...10V), die Eingänge AI4 und AI5 sind Stromeingänge (0...20 mA / 4...20 mA).

Um aus den Werten von AI0 - AI3 Spannungen zu berechnen, wird folgende Berechnungsformel verwendet:

Analogwert in Spannung umrechnen:

$$\text{AnalogInX} * 10 / 1024 = \text{Spannung an Kanal X [V]}$$

(Spannungen werden an AI0 - AI3 gemessen)

Bitte beachten Sie bei den Spannungseingängen den Jumper. Die „10“ in der obigen Berechnungsformel wird verwendet, wenn kein Jumper gesetzt ist (0...10V Bereich). Wird der Jumper für einen Kanal eingesteckt (0...5V Bereich), so ist in der Formel eine „5“ statt einer „10“ zu verwenden.

Um aus den Werten von AI4 - AI5 Ströme zu berechnen, wird folgende Berechnungsformel verwendet:

Analogwert in Strom umrechnen

$$\text{AnalogInX} * 0,020158400229358 = \text{Strom in Kanal X [mA]}$$

(Ströme werden an AI4 - AI5 gemessen)

### 12.2.1.6.3 GPIOIn

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	GPIO3	GPIO2	GPIO1	GPIO0
Startwert	0	0	0	0	0/1	0/1	0/1	0/1

Tabelle 77: Bits des GPIOIn Bytes

Die vier GPIO-Eingänge (GPIO\_IN) sind in einem Byte organisiert. Das niederwertigste Bit (LSB) enthält den Zustand von GPIO0. Bit 7 (MSB) bis Bit 4 sind nicht belegt.

Die GPIO-Eingänge haben keinen definierten Ruhezustand\* (floating input). Erst durch das externe Anlegen einer Spannung ergibt sich ein stabiler Zustand:

0V → Wert „0“

5V → Wert „1“

Da die Eingänge schwimmend (floating) sind, können diese ohne externen Anschluss bei „0“ oder „1“ liegen oder zwischen den Werten hin und her pendeln.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl. Weitere Informationen finden Sie im Kapitel Control- und Status-Bytes.

Nach einem Reset oder Power-Up von PxiTend sind die GPIOs als Eingänge konfiguriert.

\* Es gibt die Möglichkeit die GPIOs als Eingänge zu konfigurieren und per GPIOOut sogenannte PullUp-Widerstände zu aktivieren. Dadurch wird im Ruhezustand eine „1“ am Eingang erzeugt (ohne handelt es sich um Floating-Inputs). Weitere Informationen finden Sie im Abschnitt: GPIOOut.

## 12.2.1.6.4 TempX (L/H)

### TempXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 78: TempXL Byte

### TempXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 79: TempXH Byte

Im 1-Wire/DHT-Mode der GPIOs enthalten die Bytes TempXL und TempXH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort enthält die Temperatur-Information eines Sensors (sofern ein Sensor angeschlossen und konfiguriert ist). Das „X“ steht für die Nummer des GPIOs, also „0“ bis „3“.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl. Weitere Informationen finden Sie im Kapitel Control- und Status-Bytes.

Der enthaltende Wert kann in einen Temperaturwert (Gleitkommawert) umgewandelt werden. Es muss jedoch zwischen DHT11 und DHT22 unterschieden werden (TempX stellt den 16 Bit Wert dar):

DHT11: TempX / 256 = Gleitkommawert [°C]

Beispiel: 5632 / 256 = 22,0 °C – DHT11 Sensoren liefern keine Nachkommastelle!

DHT22: TempX / 10 = Gleitkommawert [°C]

Beispiel: 224 / 10 = 22,4 °C

Je nach verwendeter Programmiersprache und Datentyp sollte ein „typecast“ durchgeführt werden. Ist das nicht der Fall, ergibt sich nach dem Teilen durch 10 (DHT22) kein Gleitkommawert, sondern der ganzzahlige Wert (im Beispiel 22 statt 22,4).

Das Teilen durch 256 bei den DHT11-Sensoren lässt sich durch einen „right shift“ um 8 Stellen erreichen. Bei DHT22 Sensoren können auch negative Temperaturen erfasst werden, um dies zu erkennen muss das MSB in TempXH ausgewertet werden. Ist das MSB eine „1“, so handelt es sich um eine negative Temperatur.

## 12.2.1.6.5 HumidX (L/H)

HumidXL

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 80: HumidXL Byte

HumidXH

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 81: HumidXH Byte

Im 1-Wire/DHT-Mode der GPIOs enthalten die Bytes HumidXL und HumidXH ein zusammengehöriges 16 Bit Datenwort. Dieses Datenwort enthält die Luftfeuchtigkeits-Information eines Sensors (falls ein Sensor angeschlossen und konfiguriert ist). Das „X“ steht für die Nummer des GPIOs, sprich „0“ bis „3“.

Die GPIOs können drei unterschiedliche Aufgaben übernehmen: Eingang, Ausgang oder Temperatur-/Luftfeuchtesensoren DHT11/22, AM2302 ansteuern. Die Konfiguration erfolgt über das Control-Byte GPIOCtrl. Weitere Informationen finden Sie im Kapitel Control- und Status-Bytes.

Der enthaltene Wert kann in einen Luftfeuchtigkeitswert (Gleitkommawert) umgewandelt werden. Es muss jedoch zwischen DHT11 und DHT22 unterschieden werden (HumidX stellt den 16 Bit Wert dar):

DHT11: HumidX / 256 = Gleitkommawert [%] - relative Luftfeuchtigkeit

Beispiel:  $10496 / 256 = 41,0\%$  – DHT11 Sensoren liefern keine Nachkommastelle!

DHT22: HumidX / 10 = Gleitkommawert [%] - relative Luftfeuchtigkeit

Beispiel:  $417 / 10 = 41,7\%$

Je nach verwendeter Programmiersprache und Datentyp muss ein „typecast“ durchgeführt werden. Ist das nicht der Fall, ergibt sich nach dem Teilen durch 10 kein Gleitkommawert, sondern der ganzzahlige Wert (im Beispiel 41 statt 41,7). Das Teilen durch 256 bei den DHT11-Sensoren lässt sich durch einen „right shift“ um 8 Stellen erreichen.

### 12.2.1.6.6 RetainDataInX

Die Retain-Daten bestehen aus 64 Bytes je Richtung (RetainDataOut und RetainDataIn). Bei RetainDataIn handelt es sich um die gesicherten Daten, die nach einem Stromausfall oder einer Spannungsunterbrechung vom PiXtend-Mikrocontroller an den Raspberry Pi übergeben werden. Die Daten stehen nach dem erneuten Hochfahren des PiXtend-Systems in RetainDataIn bereit, wenn zuvor der Retain-Speicher aktiviert wurde.

Die Daten bleiben so lange erhalten bis neue Retain-Daten abgespeichert werden. Die Daten können über beliebig viele Power-Cycles des Systems erhalten bleiben, wird die Retain-Funktionalität nicht erneut aktiviert. Wird der Retain-Speicher erneut aktiviert und es kommt zur Spannungsunterbrechung, werden die Daten durch die aktuellen Daten in RetainDataOutX überschrieben.

### 12.2.2. Control- & Status-Bytes

PiXtend oder der PiXtend-Mikrocontroller kann über Control-Bytes konfiguriert werden. Des Weiteren informiert der Mikrocontroller den Raspberry Pi über Status-Bytes, Zustand sowie über Fehler und Warnungen.

Auf den folgenden Seiten vermitteln wir einen Überblick über die Control- und Status-Bytes, anschließend wird jedes Byte genau beschrieben. Es werden unterschiedliche Möglichkeiten aufgezeigt und beispielhafte Berechnungen durchgeführt.

Sollten Fragen offenbleiben, so bitten wir Sie uns über unsere Forum oder per E-Mail ([support@pixtend.de](mailto:support@pixtend.de)) zu informieren, Sie erhalten schnellstmöglich eine Antwort und weitere Informationen.

### 12.2.2.1 Control-Bytes im Überblick

Control-Bytes werden vom Raspberry Pi an den PiXtend-Mikrocontroller übertragen und können das Verhalten des Mikrocontrollers steuern.

Es gibt folgende Control-Bytes:

- **ModelOut**  
gibt an, welche Hardware-Version (Modell) die RPi-Software (z.B. CODESYS oder pxdev) erwartet
- **UCMode**  
gibt an, welcher Übertragungsmodus verwendet werden soll
- **UCCtrl**  
grundlegende Einstellungen des Mikrocontrollers (Retain, Watchdog, SafeState...)
- **DigitalInDebounce**  
gibt an, ob und wie die digitalen Eingänge entprellt werden können
- **GPIOCtrl**  
konfiguriert die PiXtend-GPIOs (Eingang, Ausgang, DHT-Mode)
- **GPIODebounce**  
gibt an, ob und wie die GPIO-Eingänge entprellt werden können
- **PWMYCtrl**  
konfiguriert die PWM-Kanäle (Modus, Freischaltung, Frequenzen)

### 12.2.2.2 Status-Bytes im Überblick

Status-Bytes werden vom PiXtend-Mikrocontroller an den Raspberry Pi übertragen und enthalten wichtige Informationen über den Zustand von PiXtend beziehungsweise des Mikrocontrollers.

Es gibt folgende Status-Bytes:

- **Firmware**  
informiert über die Mikrocontroller Firmware-Version
- **Hardware**  
informiert über die PiXtend Hardware-Version
- **ModelIn**  
informiert über das PiXtend Modell
- **UCState**  
enthält den Betriebszustand und Fehlercodes des Mikrocontrollers
- **UCWarnings**  
enthält Warnungen des Mikrocontrollers

## 12.2.2.3 Beschreibung der Control-Bytes

### 12.2.2.3.1 ModelOut

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	x	x	x	x	x	x	x	x

Tabelle 82: ModelOut Byte

Durch das Byte ModelOut informiert die Anwendungs-Software, die auf dem Raspberry Pi ausgeführt wird, den Mikrocontroller darüber, welches PiXtend-Modell sie erwartet. Der Mikrocontroller entscheidet ob es sich um das erwartete Modell handelt und wenn nicht, kann er einen Fehler an den Raspberry Pi zurückgeben.  
Der Inhalt von ModelOut ist ein ASCII-Zeichen. Im Fall von PiXtend V2 -L- enthält das Byte folgenden Wert: ASCII-Zeichen L (Großbuchstabe L – 76 dezimal / 4C hexadezimal).

Für den Anwender gibt es keinen Grund an diesem Wert etwas zu verändern. Als Beispiel werden mit dem CODESYS-Package „PiXtend V2 Professional for CODESYS“ bereits Treiber mitgeliefert, bei denen das Modell fest einprogrammiert ist und nicht verändert werden kann.

### 12.2.2.3.2 UCMODE

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 83: Bits des UCMODE Bytes

Aktuell (Stand Juni 2018) gibt es nur einen Übertragungsmodus, den sogenannten „Auto-Mode“. Das Byte ist für eine zukünftige Nutzung reserviert.

Für den Betrieb von PiXtend ist es nicht notwendig einen Wert einzutragen. Wird ein Wert eingetragen, so hat dies keine Auswirkung.

### 12.2.2.3.2.1. UCCtrl

### 12.2.2.3.2.2. UCCtrl0

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	WDE3	WDE3	WDE1	WDE0
Startwert	0	0	0	0	0	0	0	0

Tabelle 84: Bits des UCCtrl0 Bytes

#### Bit 0...3 – WDE0..3 (WatchdogEnable0...3)

Mit den WDE0..3 Bits lässt sich der Watchdog-Timer des PiXtend-Mikrocontrollers aktivieren und einstellen. In der Standard-Einstellung sind alle Bits „0“. Die Aktivierung und Einstellung des Watchdogs erläutern wir in der folgenden Tabelle:

WDE3	WDE2	WDE1	WDE0	Status Watchdog	konfigurierte Zeit
0	0	0	0	deaktiviert	deaktiviert
0	0	0	1	aktiv	16 ms
0	0	1	0	aktiv	32 ms
0	0	1	1	aktiv	64 ms
0	1	0	0	aktiv	0,125 s
0	1	0	1	aktiv	0,25 s
0	1	1	0	aktiv	0,5 s
0	1	1	1	aktiv	1 s
1	0	0	0	aktiv	2 s
1	0	0	1	aktiv	4 s
1	0	1	0	aktiv	8 s

Tabelle 85: Watchdog Einstellung

Ist der Watchdog aktiviert, wird die Kommunikation zwischen Raspberry Pi und PiXtend überwacht. Gibt es zwischen zwei gültigen Zyklen eine Pause, größer als die eingestellte Zeit, wird der Watchdog aktiv und bringt den Mikrocontroller in den sicheren Zustand\*. Ein ungültiger Zyklus (zum Beispiel durch CRC-Fehler) wird vom Watchdog gewertet, als wäre kein Zyklus durchgeführt worden.

Die Verwendung des Watchdog ist sinnvoll, wenn die PiXtend-Steuerung im Fehlerfall in einen definierten Zustand gebracht werden soll. Ein Fehlerfall liegt vor, wenn das Anwendungs-Programm auf dem Raspberry Pi nicht reagiert und keine Daten an den Mikrocontroller übertragen werden.

## Empfehlung

Aktivieren Sie den Watchdog nicht während der Entwicklung Ihrer Anwender-Software. Wird per CODESYS ein neues Programm übertragen oder die Anwender-Software ge-debugged, findet für eine bestimmte Zeit keine Übertragung statt. Der Watchdog wird immer wieder aktiv und ein Neustart des Systems wird notwendig. Dies führt zu einer unnötigen Verzögerung Ihres Entwicklungsprozesses.

### \* Definition „sicherer Zustand“:

- Alle digitalen Ausgänge und Relais werden abgeschaltet/in den Ruhezustand gebracht
- PWM-Ausgänge werden hochohmig (Tri-State) geschaltet
- Retain-Daten werden abgespeichert, wenn Retain aktiviert ist
- Die Status-LED „L1“ blinkt je nach Fehlerursache (Fehler-LED „L1“ – Signalisierung), wenn die LED aktiviert ist
- Der Mikrocontroller beziehungsweise das PiXtend-System muss im Anschluss neu gestartet werden

Wenn das Ausschalten der digitalen Ausgänge für Ihre Anwendung nicht dem sicheren Zustand entspricht, so sind externe Maßnahmen vorzusehen, um dies abzufangen.

### 12.2.2.3.2.3. UCCtrl1

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	GPUE	SLED	RE	RC	SAFE
Startwert	0	0	0	0	0	0	0	0

Tabelle 86: Bits des UCCtrl1 Bytes

#### Bit 0 – SAFE (SafeState)

Das SAFE-Bit kann, wenn es durch den Wert „1“ aktiviert wird, den Mikrocontroller unverzüglich in den sicheren Zustand\* versetzen. Das Bit hat standardmäßig den Wert „0“

\* Definition „sicherer Zustand“:

- Alle digitalen Ausgänge und Relais werden abgeschaltet/in den Ruhezustand gebracht
- PWM-Ausgänge werden hochohmig (Tri-State) geschaltet
- Retain-Daten werden abgespeichert, wenn Retain aktiviert ist
- Die Status-LED „L1“ blinkt je nach Fehlerursache (Fehler-LED „1“ – Signalisierung), wenn die LED aktiviert ist
- Der Mikrocontroller beziehungsweise das PiXtend-System muss im Anschluss neu gestartet werden

Wenn das Ausschalten der digitalen Ausgänge für Ihre Anwendung nicht dem sicheren Zustand entspricht, so sind externe Maßnahmen vorzusehen, um dies abzufangen.

#### Bit 1 – RC (RetainCopy)

Mit dem RC-Bit lassen sich die Daten konfigurieren, die im Retain Input-Bereich (RetainDataIn0...63) sichtbar werden sollen. Beim Startwert „0“ werden die zuletzt vom Mikrocontroller gespeicherten Daten an den Raspberry Pi übergeben, normaler Retain Betrieb. Wird der Wert „1“ für das RC-Bit gesetzt, werden die zuletzt vom Raspberry Pi gesendeten Daten wieder zurückgegeben. Im Retain-Bereich RetainDataIn0...63 wird (RetainDataOut0...63) gespiegelt, jedoch mit einem Zyklus Verzögerung (Retain Copy Betrieb). Der Inhalt der Retain-Daten geht nicht verloren, er wird nicht mehr angezeigt, solange das RC-Bit „1“ ist.

Weitere Informationen zur Retain-Funktion von PiXtend finden Sie 6.4 Retain-Speicher (dt. Remanenz).

#### Bit 2 – RE (RetainEnable)

Mit dem RE-Bit kann die Retain-Funktion von PiXtend aktiviert werden. Dazu wird eine „1“ in dieses Bit geschrieben. Standardmäßig ist RE „0“, nach einem Reset oder Power-Up und somit ist Retain deaktiviert.

#### Empfehlung

Aktivieren Sie Retain nur dann, wenn Sie die Funktionalität wirklich benötigen. Die Anzahl der Retain-Speicherzyklen kann nur bis 10.000 Zyklen garantiert werden. Weitere Informationen zur Retain-Funktion von PiXtend V2 finden Sie im Kapitel: 6 Basis Wissen.

#### Bit 3 – SLED (StatusLED)

Die Status-LED „L1“ kann per SLED-Bit deaktiviert werden, wenn sie nicht benötigt wird. Standardmäßig ist die Status-LED aktiv und lässt sich durch den Wert „1“ in SLED deaktivieren.

#### Bit 4 – GPUE (GPIOPullUpEnable)

Mit dem GPUE-Bit lassen sich die PullUp-Widerstände der PiXtend-GPIOs freischalten, sie sind jedoch noch nicht aktiv. Die Freischaltung hat eine Auswirkung, wenn die GPIOs als Eingänge konfiguriert sind und im Byte GPIOOut eine „1“ für den jeweiligen GPIO geschrieben wird. Eine „1“ in GPUE aktiviert diese Funktionalität.

Weitere Informationen zu den PiXtend-GPIOs finden Sie im Abschnitt: GPIOOut.

### 12.2.2.3.3DigitalInDebounce

#### DigitalInDebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 87: Bits des DigitalInDebounce01 Bytes

#### DigitalInDebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 88: Bits des DigitalInDebounce23 Bytes

#### DigitalInDebounce45

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 89: Bits des DigitalInDebounce45 Bytes

#### DigitalInDebounce67

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 90: Bits des DigitalInDebounce67 Bytes

#### DigitalInDebounce89

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 91: Bits des DigitalInDebounce89 Bytes

#### DigitalInDebounce1011

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 92: Bits des DigitalInDebounce1011 Bytes

**DigitalInDebounce1213**

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 93: Bits des DigitalInDebounce1213 Bytes

**DigitalInDebounce1415**

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 94: Bits des DigitalInDebounce1415 Bytes

Mit Hilfe der DigitalInDebounce-Bytes lässt sich das Entprellen der 16 digitalen Eingänge von PiXtend V2 -L-konfigurieren. Standardmäßig ist die Entprellung der Eingänge nicht aktiv. Das Entprellen wird jeweils gemeinsam für zwei Kanäle aktiviert/deaktiviert. Es entstehen dadurch keine Wechselwirkungen zwischen den beiden Kanälen, diese erhalten lediglich die gleiche Entprellungs-Einstellung.

Die Zahl in einem DigitalInDebounce-Byte entspricht der Anzahl der Zyklen, in der ein am digitalen Eingang anliegendes elektrisches Signal konstant bleiben muss, damit die Pegeländerung an der Anwendungssoftware (auf dem Raspberry Pi) weitergereicht wird. Das gilt sowohl für einen Wechsel des Signalpegels von „0“ (low) zu „1“ (high), als auch umgekehrt. Werden die Bytes nicht verändert oder wird eine „0“ als Wert geschrieben, so findet kein Entprellen statt und die Anwendungssoftware erhält in jedem Zyklus einen neuen Wert.

**Ein Beispiel:**

Die ersten beiden digitalen Eingänge (DIO und DI1) werden entprellt. Die Änderung soll nur an den digitalen Eingängen, die länger als 100 ms konstant anliegen, sichtbar werden. Die Zykluszeit (Datenaustausch zwischen PiXtend und Raspberry Pi) beträgt 10 ms.

- Das Byte DigitalInDebounce01 wird verwendet
- Berechnung des Wertes für DigitalInDebounce01:  $100 \text{ ms} / 10 \text{ ms} = 10$

Das Byte wird mit einer 10 (dezimal) beschrieben, um den gewünschten Effekt zu erhalten. Wird die Zykluszeit während der Software-Entwicklung verändert, achten Sie bitte darauf, den Wert für die Entprellung anzupassen.

## 12.2.2.3.4GPIOCtrl

Bit	7	6	5	4	3	2	1	0
Name	SENS3	SENS2	SENS1	SENS0	IO3	IO2	IO1	IO0
Startwert	0	0	0	0	0	0	0	0

Tabelle 95: Bits des GPIOCtrl Bytes

### Bits 0...3 – IO0...3

Mit den IO-Bits lassen sich die vier PiXtend-GPIOs als digitaler Eingang oder Ausgang konfigurieren. Mit dem Startwert „0“ sind alle GPIOs als Eingänge konfiguriert. Setzen Sie das Bit IO3 auf „1“, so wird GPIO3 zum Ausgang. Der Wert wird über das Datenwort GPIOOut gesetzt.

### Bits 4...7 – SENS0...3 (Sensor0...3)

Die GPIOs können zusätzlich zum Eingang/Ausgang für 1-Wire Sensoren (DHT11, DHT22, AM2302) genutzt werden, dafür stehen die oberen vier Bits des GPIOCtrl-Datenworts bereit. Wird das Bit SENS3 auf „1“ gesetzt, so lässt sich an GPIO3 ein Sensor anschliessen und auswerten. Die Einstellung des IO-Bit spielt in diesem Fall keine Rolle, die SENSX-Bits setzen sich gegenüber den IOX-Bits durch.

Die Ergebnisse/Messwerte der Sensoren können Sie TempXL/TempXH und HumidXL/HumidXH entnehmen. Das „X“ entspricht der Nummer des GPIOs, an dem ein Sensor angeschlossen ist.

#### NOTICE

Für die Kommunikation mit den Sensoren wird Zeit benötigt (ca. 25 ms), unter Verwendung mindestens eines Sensors ergibt sich eine minimale Zykluszeit von 30 ms. Es spielt dabei keine Rolle ob ein oder vier Sensoren abgefragt werden.

### 12.2.2.3.5 GPIODebounce

GPIODebounce01

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 96: Bits des GPIODebounce01 Bytes

GPIODebounce23

Bit	7	6	5	4	3	2	1	0
Name	MSB							LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 97: Bits des GPIODebounce23 Bytes

Das Entprellen der GPIOs ist möglich, wenn diese als Eingänge konfiguriert sind. Die GPIODebounce-Bytes werden genau gleich verwendet, wie die DigitalInDebounce-Bytes. Weitere Informationen und Beispiele befinden sich im Abschnitt: DigitalInDebounce.

## 12.2.2.3.6PWMYCtrl – für 16 Bit PWMs

Die nachfolgenden Beschreibungen beziehen sich auf alle 16 Bit PWM<sup>YX</sup>-Kanäle (PWM0A, PWM0B, PWM1A, PWM1B, PWM2A & PWM2B).

Das „Y“ wird durch die Nummer der Gruppe ersetzt („0“, „1“ oder „2“) und das „X“ durch den Buchstaben des jeweiligen PWM-Kanals („A“ oder „B“).

### 12.2.2.3.6.1. PWMYCtrl0

Bit	7	6	5	4	3	2	1	0
Name	PS2	PS1	PS0	EnableB	EnableA	-	MODE1	MODE0
Startwert	0	0	0	0	0	0	0	0

Tabelle 98: Bits des PWMYCtrl0 Bytes

#### Bit 0...1 – MODE0...1

Die PWM-Ausgänge können in vier verschiedenen Modi betrieben werden. Der Modus wird über die MODE-Bits konfiguriert. Standardmäßig startet PiXtend im Servo-Mode (MODE0..1 = 00).

MODE1	MODE0	Modus	Beschreibung
0	0	Servo-Mode	Beide Kanäle (A/B) werden zur Ansteuerung von Modellbau-Servos verwendet – spezielle Signalform
0	1	Duty-Cycle-Mode	Beide Kanäle (A/B) haben die gleiche Frequenz, unabhängig vom einstellbaren Duty Cycle
1	0	Universal-Mode	Frei einstellbare Frequenz und Duty Cycle für Kanal A, halbe Frequenz für Kanal B und 50% Duty Cycle
1	1	Frequency-Mode	Frei einstellbare Frequenzen für beide Kanäle (A/B), Duty Cycle ist immer 50%

Tabelle 99: Mode Bits im PWMYCtrl0 Byte

Servo-Mode und Duty-Cycle-Mode sind ausreichend, Universal- und Frequency-Mode können für spezielle Anwendungen sinnvoll und hilfreich sein, sind jedoch aufwändiger zu konfigurieren und zu verwenden.

#### Bit 3..4 – EnableA, EnableB

Mit den Bits EnableA und EnableB kann der jeweilige Kanal aktiviert werden. Standardmäßig sind die PWM-Kanäle deaktiviert. Durch das Schreiben einer „1“ in das jeweilige Bit wird der Kanal aktiviert und die PWM am Ausgang sichtbar.

Die PWM lässt sich zunächst für die gewünschte Anforderung konfigurieren (Modus, Frequenz/Wert...) und erst im zweiten Schritt aktivieren. Die Aktivierung kann im gleichen Zyklus erfolgen.

### Bit 5...7 – PS0...PS2 (Prescaler0...3)

Die Prescaler-Bits (PS-Bits) ermöglichen die grobe Einstellung der PWM-Frequenz. Je nach PWM-Modus (siehe MODE-Bits) haben die PS-Bits verschiedene Auswirkungen.

Folgende Tabelle gibt Aufschluss über die Auswirkungen der PS-Bits:

PS2	PS1	PS0	Beschreibung	Grundfrequenz
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

Tabelle 100: Prescaler-Bits im PWMMYCtrl0 Byte

Die Prescaler-Einstellungen, abhängig vom eingestellten Modus, haben unterschiedliche Auswirkungen:

Modus	Beschreibung
Servo-Mode	Keine Auswirkung
Duty-Cycle-Mode	Einstellung des Prescalers bzw. der Grundfrequenz
Universal-Mode	Einstellung des Prescalers bzw. der Grundfrequenz
Frequency-Mode	Einstellung des Prescalers bzw. der Grundfrequenz

Tabelle 101: PWM Modus

Die grobe Einstellung der Frequenz erfolgt über die Prescaler-Bits (PS0...2) in PWMMYCtrl0, die Feineinstellung per PWMMYCtrl1L/H.

Wie die Frequenz/Periodendauer genau eingestellt werden kann, erfahren Sie im Abschnitt: PWMMYX – 16 Bit Auflösung.

#### NOTICE

Wird der der Frequency-Mode verwendet dürfen keine DHT-Sensoren angeschlossen und im GPIOCtrl-Byte aktiviert werden. Eine Kombination aus Frequency-Mode und DHT-Sensoren ist nicht möglich. Wurden ein oder mehrere DHT-Sensoren an den PiXtend-GPIOs aktiviert, so lässt sich der Frequency-Mode nicht aktivieren. Der Versuch den Frequency-Mode trotzdem zu aktivieren, obwohl DHT-Sensoren verwendet werden, führt zur Deaktivierung der PWM-Ausgänge

### 12.2.2.3.6.2. PWMYCtrl1 (L/H)

Die PWMYCtrl1-Bytes L und H enthalten ein zusammengehöriges 16 Bit Datenwort, PWMYCtrl1L ist das Low-Byte und PWMYCtrl1H ist das High-Byte.

Mit den beiden Bytes kann die PWM-Frequenz/Periodendauer eingestellt werden (im PWM-Modus „Duty-Cycle“ und „Universal“). Die Frequenz hat für beide Kanäle Gültigkeit.

Im Duty-Cycle-Mode kann das Tastverhältnis für jeden Kanal unabhängig konfiguriert werden, im Universal-Mode lässt sich das Tastverhältnis nur für Kanal A einstellen.

Für den Frequency-Mode haben die PWMYCtrl1-Bytes keine Auswirkung.

PWMYCtrl1L

Bit	7	6	5	4	3	2	1	0
Name								LSB
Startwert	0	0	0	0	0	0	0	0

Tabelle 102: PWMYCtrl1L Byte

PWMYCtrl1H

Bit	7	6	5	4	3	2	1	0
Name	MSB							
Startwert	0	0	0	0	0	0	0	0

Tabelle 103: PWMYCtrl1H Byte

Beispiel für die PWM-Periodendauer im „Duty-Cycle-Mode“:

Der Prescaler (PS0...2) wird auf den Wert 64 und somit die Grundfrequenz auf 250 kHz eingestellt, der Duty-Cycle-Mode und der Kanal PWM0A wird aktiviert:

PWM0Ctrl0: 01101001b

Das 16 Bit Datenwort PWM0Ctrl1 wird auf den Wert 1000 eingestellt:

PWM0Ctrl1L: 11101000b

PWM0Ctrl1H: 00000011b

PWM-Periodendauer = Mikrocontroller-Basistakt / 2 / Prescaler / PWMYCtrl1

PWM-Periodendauer = 16 MHz / 2 / 64 / 1000 = 125 Hz

## 12.2.2.4 Beschreibung der Status-Bytes

Die Status-Bytes informieren den Anwender und das Anwenderprogramm, das auf dem Raspberry Pi läuft, über den Status des Mikrocontrollers. Die Status-Bytes können per CODESYS, PiXtend-C-Library oder PiXtend-Python-Library (PPL) und so weiter abgefragt werden.

### 12.2.2.4.1 Firmware

Im Firmware-Byte befindet sich eine Zahl, die für die Firmware-Version steht, die Zahl kann zwischen 0 und 255 liegen. Wenn Sie Support für Ihr PiXtend V2 Gerät anfragen, geben Sie bitte immer die Firmware-Version an.

### 12.2.2.4.2 Hardware

Das Hardware-Byte enthält die Versionsnummer Ihres PiXtend-Boards. Mit der Versionsnummer kann die Anwendersoftware prüfen, ob die Software/Treiber zur vorhandenen Hardware passen.

**Beispiel:**

Hardware-Byte enthält die Zahl 21  
→ PiXtend Hardware Version 2.1

### 12.2.2.4.3 ModellIn

Im ModellIn-Byte informiert der Mikrocontroller die Anwender-Software über das PiXtend V2 Modell. Mit der Modell-Kennung kann die Anwendersoftware prüfen, ob die Software/Treiber zur vorhandenen Hardware passen. Der Inhalt von ModellIn ist ein ASCII-Zeichen. Im Fall von PiXtend V2 -L- enthält das Byte folgenden Wert: ASCII-Zeichen L (Großbuchstabe L – 76 dezimal / 4C hexadezimal).

## 12.2.2.4.4 UCState

Bit	7	6	5	4	3	2	1	0
Name	ERR3	ERR2	ERR1	ERR0	-	-	-	RUN
Startwert	0	0	0	0	0	0	0	0

Tabelle 104: Bits des UCState Bytes

### Bit 0 – RUN

Der Anwender kann durch die Abfrage dieses Bits prüfen, ob PiXtend betriebsbereit („1“) ist und ob eine erfolgreiche Datenübertragung durchgeführt wurde. Das kann nach dem Start des PiXtend-Systems sinnvoll sein.

Wird der Controller in den SafeState gebracht, beispielsweise durch den Watchdog, so enthält das Bit eine „0“. Da keine Kommunikation möglich ist, bis zu einem Neustart, kann dieses Bit nicht abgefragt werden.

### Bit 4...7 – ERR0...3

Die ERR-Bits informieren über Probleme, die der Mikrocontroller erkennt und an das Anwenderprogramm weiterreicht. Die nachfolgende Tabelle beschreibt die Fehlercodes und deren Ursache:

ERR3	ERR2	ERR1	ERR0	Fehlerbeschreibung
0	0	0	0	Kein Fehler
0	0	0	1	-
0	0	1	0	CRC-Fehler - Data die vom Raspberry Pi erhaltenen Nutzdaten sind fehlerhaft
0	0	1	1	Datenblock zu kurz Treiber fehlerhaft/falsches Modell eingestellt
0	1	0	0	PiXtend Modell stimmt nicht überein das erwartete und das tatsächliche Modell stimmen nicht überein (ModellIn und ModelOut sind ungleich)
0	1	0	1	CRC-Fehler - Header die vom Raspberry Pi erhaltenen Headerdaten sind fehlerhaft
0	1	1	0	SPI-Frequenz zu hoch die eingestellte SPI-Frequenz ist zu hoch, es kommt zu Übertragungsfehlern

Tabelle 105: Aufschlüsselung Error-Bits im UCState Byte

## 12.2.2.4.5UCWarnings

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	W2	W1	W0	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 106: Bits des UCWarnings Bytes

### Bit 1...2 – W0...1 (RetainCRCError, RetainVoltageError)

Die W0...1-Bits informieren den Anwender über Warnungen in Bezug auf die Retain-Funktionalität von PiXtend V2. Im Anwenderprogramm kann auf diese Bits abgeprüft und erkannt werden, ob der Retain-Speicher korrekt arbeitet.

#### W0 – RetainCRCError

Ist „1“, wenn bei der Überprüfung des Retain-Speichers ein Fehler erkannt wurde.

#### W1 – RetainVoltageError

Ist „1“, wenn die aktuelle Versorgungsspannung von PiXtend zu gering ist (kleiner 19V). Die Retain-Speicherung lässt sich nicht aktivieren.

### Bit 3 – W2 (I2CError)

Der PiXtend-Mikrocontroller und der PWM-Controller sind per I<sup>2</sup>C-Bus verbunden.

W2 ist „1“, wenn bei der Überprüfung (I<sup>2</sup>C/TWI) der PWM-Daten für PWM1A, PWM1B, PWM2A und PWM2B ein Fehler aufgetreten ist.

### 12.3. Fehler-LED „L1“ - Signalisierung

Es gibt Fehlerzustände, die im Mikrocontroller auftreten können, die sich gegebenenfalls nicht mehr per SPI-Bus an den Raspberry Pi zurückmelden lassen. Zum Beispiel bei einer fehlerhaften Kommunikation oder wenn der Watchdog des Mikrocontrollers ausgelöst hat. Für diese Situation wurde auf dem PiXtend V2 Board eine LED (L1) vorgesehen. Diese LED kann zur Problemermittlung herangezogen werden, die Bedeutungen der verschiedenen Zustände sind nachfolgend aufgeführt.

Zustand	Bedeutung	Maßnahmen
Aus	Betriebsbereit, keine Störung	-
An/Blinken sehr schnell	Übertragungsfehler (SPI)	Schalter „SPI_EN“ prüfen, verwendete Softwarekomponenten prüfen, es darf nur ein Programm mit dem Mikrocontroller kommunizieren. In C-Programmen müssen die Input- und Output-Strukturvariablen global deklariert sein oder lokal mit dem Schlüsselwort „static“.
Blinken schnell (0,2s)	Watchdog hat ausgelöst	RPi herunterfahren und Versorgung trennen. Der Mikrocontroller muss neu gestartet werden damit dieser wieder betriebsbereit ist.
Ausblenden (3s)	AC/Retain Fehler	Die Versorgungsspannung ist nicht ausreichend, obwohl die Retain-Funktion aktiviert wurde. Die Retain-Daten können nicht gespeichert werden und sind außer Funktion. Die Versorgungsspannung muss 24Volt betragen.
Pulsieren (5s)	Runterfahren (sicherer Zustand)	Der Mikrocontroller wird in den „sicheren Zustand“ versetzt. Bit 0 im Control-Byte UCCtrl1 wurde gesetzt.

Tabelle 107: LED L1 Signalisierung

## 13. Abbildungsverzeichnis

Abbildung 1: Software - Win32DiskImager.....	22
Abbildung 2: Basis Wissen - RPi IP-Adresse in CODESYS ermitteln.....	25
Abbildung 3: Basis Wissen - Alle im Netzwerk gefundenen Raspberry Pi's.....	25
Abbildung 4: RPi - Bluetooth ausschalten.....	30
Abbildung 5: CODESYSControl_User.cfg - Serieller Anschluss ttyAMA.....	31
Abbildung 6: CODESYS Development System (Quelle: 3S) .....	37
Abbildung 7: CODESYS - Visualisierung für das PiXtend V2 Demo Projekt.....	38
Abbildung 8: CODESYS - Neues Projekt.....	42
Abbildung 9: CODESYS - Neues Projekt - Auswahl Gerät.....	43
Abbildung 10: CODESYS - PiXtendTest Projekt.....	43
Abbildung 11: CODESYS - Gerät anhängen.....	44
Abbildung 12: CODESYS - Gerät anhängen - SPI master .....	45
Abbildung 13: CODESYS - Gerät anhängen - PiXtend V2 -S- .....	46
Abbildung 14: CODESYS - Globale Variablenliste einfügen.....	47
Abbildung 15: CODESYS - GVL - Deklarationen.....	48
Abbildung 16: CODESYS - PiXtend V2 -S- E/A-Abbild.....	49
Abbildung 17: CODESYS - Eingabehilfe.....	50
Abbildung 18: CODESYS - PiXtend V2 -S- E/A-Abbild - State .....	51
Abbildung 19: CODESYS - GPIO Parameter - GPIO24.....	51
Abbildung 20: CODESYS - GPIO E/A-Abbild - rpi_gpio24 Variable.....	52
Abbildung 21: CODESYS - Taskkonfiguration - MainTask.....	53
Abbildung 22: CODESYS - PLC_PRG (CFC) - Eingang.....	54
Abbildung 23: CODESYS - CFC - 3 Eingänge verbunden mit 3 Ausgängen.....	55
Abbildung 24: CODESYS - CFC - Parametriert.....	55
Abbildung 25: CODESYS - Kommunikation - Netzwerk durchsuchen.....	56
Abbildung 26: CODESYS - PiXtend V2 -S- E/A-Abbild online.....	57
Abbildung 27: CODESYS - PLC_PRG (CFC) - Demo-Programm .....	58
Abbildung 28: CODESYS - Web-Visualisierung - Konfiguration.....	59
Abbildung 29: CODESYS - Visualisierung.....	60
Abbildung 30: CODESYS - Visualisierung - Elementeigenschaften .....	66
Abbildung 31: CODESYS - Visualisierung - Konfiguration eines Rechtecks.....	61
Abbildung 32: CODESYS - Visualisierung - Fertiggestellt.....	62
Abbildung 33: CODESYS - DAC - Neues Projekt erstellen .....	63
Abbildung 34: CODESYS - DAC - Gerät auswählen.....	64
Abbildung 35: CODESYS - DAC - Projektübersicht .....	64
Abbildung 36: CODESYS - DAC - SPI Gerät anhängen.....	65
Abbildung 37: CODESYS - DAC - SPI master anhängen.....	66
Abbildung 38: CODESYS - DAC - SPI master konfigurieren.....	67
Abbildung 39: CODESYS - DAC - PiXtend V2 DAC anhängen .....	71
Abbildung 40: CODESYS - DAC - E/A Abbild Einstellungen.....	72
Abbildung 41: CODESYS - DAC - Globale Variablen Liste hinzufügen.....	68
Abbildung 42: CODESYS - DAC - GVL Variablen .....	68
Abbildung 43: CODESYS - DAC - Variablenmapping .....	69
Abbildung 44: CODESYS - DAC - GPIO 24 als Ausgang verwenden.....	69
Abbildung 45: CODESYS - DAC - GPIO 24 .....	70
Abbildung 46: CODESYS - DAC - Variable rpi_gpio24 setzen .....	73
Abbildung 47: CODESYS - DAC - Netzwerk durchsuchen .....	74
Abbildung 48: CODESYS - DAC - WebVisu Einstellungen.....	75
Abbildung 49: CODESYS - DAC - WebVisu - Schieberegl器 .....	76
Abbildung 50: CODESYS - DAC - Schieberegl器 - Skalenende festlegen .....	76
Abbildung 51: CODESYS - DAC - WebVisu - Textfelder einstellen .....	77
Abbildung 52: CODESYS - DAC - WebVisu - Fertig .....	78
Abbildung 53: CODESYS - Hinweis zur Verwendung beider PiXtend V2 SPI Geräte .....	79
Abbildung 54: CODESYS - Schaltplan RS232 .....	80
Abbildung 55: CODESYS - Schaltplan / Anschluss RS485 .....	81
Abbildung 56: CODESYS - HTerm 0.8.1beta - RS232 Testprogramm .....	84
Abbildung 57: CODESYS - Visualisierung - Testprogramm - Serielle Kommunikation .....	85
Abbildung 58: CODESYS - CAN-Bus Test unter Linux .....	92
Abbildung 59: CODESYS - CAN-Bus - Neues Projekt erstellen .....	95
Abbildung 60: CODESYS - CAN-Bus Slave - SPS Gerät anhängen .....	96
Abbildung 61: CODESYS - CAN-Bus Slave- RPi anhängen .....	96
Abbildung 62: CODESYS - CAN-Bus Slave - Gerät an SPS anhängen .....	97
Abbildung 63: CODESYS - CAN-Bus Slave - CAN-Bus-Gerät anhängen .....	98
Abbildung 64: CODESYS - CAN-Bus Slave - CANopen Device Gerät anhängen .....	99

Abbildung 65: CODESYS - CAN-Bus Slave - E/A-Bereich konfigurieren.....	100
Abbildung 66: CODESYS - CAN-Bus Slave - PDO festlegen.....	100
Abbildung 67: CODESYS - CAN-Bus Slave - Eigenes CAN-Gerät installieren .....	101
Abbildung 68: CODESYS - CAN-Bus Slave - POU und Taskkonfiguration .....	102
Abbildung 69: CODESYS - CAN-Bus Slave - Taskkonfiguration.....	103
Abbildung 70: CODESYS - CAN-Bus Slave - CANopen E/A-Abbild .....	103
Abbildung 71: CODESYS - CAN-Bus Slave - Baudrate.....	104
Abbildung 72: CODESYS - CAN-Bus Master - Raspberry Pi anhängen.....	105
Abbildung 73: CODESYS - CAN-Bus Master - CANbus anhängen.....	106
Abbildung 74: CODESYS - CAN-Bus Master - CANopen Manager anhängen .....	107
Abbildung 75: CODESYS - CAN-Bus Master - CAN-Bus Slave anhängen.....	108
Abbildung 76: CODESYS - CAN-Bus Master - CAN-Slave konfigurieren .....	109
Abbildung 77: CODESYS - CAN-Bus Test - Mehrfacher Download.....	110
Abbildung 78: CODESYS - CAN-Bus Test - Master und Slave tauschen Daten aus.....	111
Abbildung 79: CODESYS - Retain-Demo - Gerätebaum.....	112
Abbildung 80: CODESYS - Retain - Variablen Deklaration .....	113
Abbildung 81: CODESYS - Retain - Beispielprogramm.....	114
Abbildung 82: CODESYS - Retain – Startwert 13.....	115
Abbildung 83: CODESYS - Retain - arRetainDataOut[0] wurde um 1 auf 14 erhöht.....	115
Abbildung 84: CODESYS - Retain - Nach Power-Cycle .....	115
Abbildung 85: CODESYS - Herunterfahren - Bibliotheksverwalter .....	117
Abbildung 86: CODESYS - Herunterfahren - Bibliothek hinzufügen .....	118
Abbildung 87: CODESYS - Herunterfahren - Variablen .....	119
Abbildung 88: CODESYS - Herunterfahren - Beispielprogramm.....	119
Abbildung 89: CODESYS PiXtend V2 -S- - SPI devices Parameter Reiter.....	120
Abbildung 90: CODESYS PiXtend V2 -L- - SPI devices Parameter Reiter .....	125
Abbildung 91: CODESYS - GPIO24 als Ausgang konfiguriert.....	131
Abbildung 92: Linux-Tools - pixtendtool2s - Schalter '-di' .....	141
Abbildung 93: Linux-Tools - pixtendtool2s - Schalter '-ai' .....	141
Abbildung 94: Linux-Tools - pixtendtool2s - Schalter '-sws' .....	141
Abbildung 95: Linux-Tools - pixtendtool2s - Schalter '-sr' .....	142
Abbildung 96: Linux-Tools - pxauto2s - Hauptmenü.....	144
Abbildung 97: Linux-Tools - pxauto2s - Menü DOut .....	146
Abbildung 98: Linux-Tools - pxauto2s .....	147
Abbildung 99: Linux-Tools - pixtendtool2s .....	147
Abbildung 100: C-Programm - 100ms Zykluszeit.....	162
Abbildung 101: FHEM - Aktivieren der Sensorfunktion.....	171
Abbildung 102: FHEM - Plot durchLogFile erstellen .....	172
Abbildung 103: FHEM - Diagramm der Sensormesswerte .....	172
Abbildung 104: Python - Bildquelle: <a href="https://www.python.org/">https://www.python.org/</a> , The Python Brochure .....	175
Abbildung 105: Python - Raspbian PIXEL- Terminal Icon in der Taskleiste.....	176
Abbildung 106: Python - Konsole - PPLV2 Verzeichnis .....	177
Abbildung 107: Python - PiXtend Python Library V2 Demo-Programm .....	178
Abbildung 108: Python - Einfaches Demo-Programm.....	179

## 14. Tabellenverzeichnis

Tabelle 1: Raspberry Pi GPIO24 Bezeichnungen.....	20
Tabelle 2: Technische Daten – Retain System .....	21
Tabelle 3: CODESYS E/A Übersicht für PiXtend V2 -S- .....	124
Tabelle 4: CODESYS E/A Übersicht für PiXtend V2 -L- .....	130
Tabelle 5: PiXtend V2 -S- SPI-Protokoll Übersicht.....	187
Tabelle 6: Bits des DigitalOut Bytes .....	189
Tabelle 7: Bits des RelayOut Bytes .....	189
Tabelle 8: Bits des GPIOOut Bytes .....	190
Tabelle 9: PWM0XL Byte.....	192
Tabelle 10: PWM0XH Byte .....	192
Tabelle 11: PWM0XL Byte.....	193
Tabelle 12: PWM0XH Byte.....	193
Tabelle 13: PWM0AL Byte .....	194
Tabelle 14: PWM0AH Byte .....	194
Tabelle 15: Bits des PWM0AL Bytes .....	194
Tabelle 16: Bits des PWM0AH Bytes.....	191
Tabelle 17: Bits des PWM1XL Bytes .....	195
Tabelle 18: Bits des PWM1XH Byte .....	195
Tabelle 19: Bits des PWM1XL Bytes .....	196
Tabelle 20: Bits des PWM1XH Bytes.....	196
Tabelle 21: Bits des PWM1AL Bytes .....	197
Tabelle 22: Bits des PWM1AH Bytes.....	197
Tabelle 23: Bits des PWM1AL Bytes .....	198
Tabelle 24: Bits des PWM1AH Bytes.....	198
Tabelle 25: Bits des AnalogOutL Bytes .....	199
Tabelle 26: Bits des AnalogOutH Bytes.....	199
Tabelle 27: Bits des DigitalIn Bytes .....	201
Tabelle 28: AnalogInXL.....	202
Tabelle 29: Bits des AnalogInXH Bytes.....	202
Tabelle 30: Bits des GPIOIn Bytes .....	203
Tabelle 31: TempXL Byte .....	204
Tabelle 32: TempXH Byte .....	204
Tabelle 33: HumidXL Byte .....	205
Tabelle 34: HumidXH Byte .....	205
Tabelle 35: ModelOut Byte .....	208
Tabelle 36: Bits des UCMode Bytes .....	208
Tabelle 37: Bits des UCCtrl0 Bytes.....	209
Tabelle 38: Watchdog Einstellung.....	209
Tabelle 39: Bits des UCCtrl1 Bytes .....	211
Tabelle 40: Bits des DigitalInDebounce01 Bytes.....	212
Tabelle 41: Bits des DigitalInDebounce23 Bytes .....	212
Tabelle 42: Bits des DigitalInDebounce45 Bytes .....	212
Tabelle 43: Bits des DigitalInDebounce67 Bytes.....	212
Tabelle 44: Bits des GPIOCtrl Bytes.....	214
Tabelle 45: Bits des GPIODebounce01 Bytes .....	215
Tabelle 46: Bits des GPIODebounce23 Bytes .....	215
Tabelle 47: Bits des PWM0Ctrl0 Bytes .....	216
Tabelle 48: Mode Bits im PWM0Ctrl0 Byte.....	216
Tabelle 49: Prescaler-Bits im PWM0Ctrl0 Byte .....	217
Tabelle 50: PWM Modus .....	217
Tabelle 51: PWM0Ctrl1L Byte .....	218
Tabelle 52: PWM0Ctrl1H Byte .....	218
Tabelle 53: Bits des PWM1Ctrl1L Bytes .....	221
Tabelle 54: Bits des PWM1Ctrl1H Bytes .....	221
Tabelle 55: Bits des UCState Bytes .....	224
Tabelle 56: Aufschlüsselung Error-Bits im UCState Byte .....	224
Tabelle 57: Bits des UCWarnings Bytes .....	225
Tabelle 58: PiXtend V2 -L- SPI-Protokoll Übersicht.....	228
Tabelle 59: Bits des DigitalOut0 Bytes .....	230
Tabelle 60: Bits des DigitalOut1 Bytes .....	230
Tabelle 61: Bits des RelayOut Bytes .....	231
Tabelle 62: Bits des GPIOOut Bytes .....	232
Tabelle 63: PWMYXL Byte.....	233
Tabelle 64: PWMYXH Byte.....	233

Tabelle 65: PWMYXL Byte.....	234
Tabelle 66: PWMYXH Byte.....	234
Tabelle 67: PWMYAL Byte.....	236
Tabelle 68: PWMYAH Byte.....	236
Tabelle 69: Bits des PWMYXL Bytes.....	237
Tabelle 70: Bits des PWMYXH Bytes.....	237
Tabelle 71: Bits des AnalogOutXL Bytes.....	239
Tabelle 72: Bits des AnalogOutXH Bytes .....	240
Tabelle 73: Bits des DigitalIn0 Bytes.....	241
Tabelle 74: Bits des DigitalIn1 Bytes.....	241
Tabelle 75: AnalogInXL.....	242
Tabelle 76: Bits des AnalogInXH Bytes.....	242
Tabelle 77: Bits des GPIOIn Bytes.....	243
Tabelle 78: TempXL Byte.....	244
Tabelle 79: TempXH Byte.....	244
Tabelle 80: HumidXL Byte.....	245
Tabelle 81: HumidXH Byte.....	245
Tabelle 82: ModelOut Byte.....	248
Tabelle 83: Bits des UCMode Bytes.....	248
Tabelle 84: Bits des UCCtrl0 Bytes .....	249
Tabelle 85: Watchdog Einstellung.....	249
Tabelle 86: Bits des UCCtrl1 Bytes.....	251
Tabelle 87: Bits des DigitalInDebounce01 Bytes .....	252
Tabelle 88: Bits des DigitalInDebounce23 Bytes .....	252
Tabelle 89: Bits des DigitalInDebounce45 Bytes .....	252
Tabelle 90: Bits des DigitalInDebounce67 Bytes .....	252
Tabelle 91: Bits des DigitalInDebounce89 Bytes .....	252
Tabelle 92: Bits des DigitalInDebounce1011 Bytes .....	252
Tabelle 93: Bits des DigitalInDebounce1213 Bytes .....	253
Tabelle 94: Bits des DigitalInDebounce1415 Bytes .....	253
Tabelle 95: Bits des GPIOCtrl Bytes .....	254
Tabelle 96: Bits des GPIODebounce01 Bytes .....	255
Tabelle 97: Bits des GPIODebounce23 Bytes .....	255
Tabelle 98: Bits des PWMYCtrl0 Bytes .....	256
Tabelle 99: Mode Bits im PWMYCtrl0 Byte .....	256
Tabelle 100: Prescaler-Bits im PWMYCtrl0 Byte .....	257
Tabelle 101: PWM Modus .....	257
Tabelle 102: PWMYCtrl1L Byte .....	258
Tabelle 103: PWMYCtrl1H Byte .....	258
Tabelle 104: Bits des UCState Bytes .....	260
Tabelle 105: Aufschlüsselung Error-Bits im UCState Byte .....	260
Tabelle 106: Bits des UCWarnings Bytes .....	261
Tabelle 107: LED L1 Signalisierung .....	262