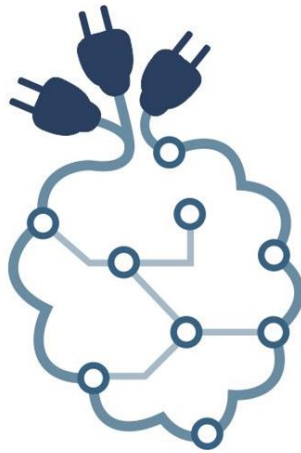




PiXtend[®]

PiXtend eIO

Versionshistorie

Version	Datum	Beschreibung	Bearbeiter
1.00	23.10.2019	Dokument erstellt, erste Version	RT

Inhaltsverzeichnis

1.	Hinweise zu dieser Dokumentation	5
1.1.	Gültigkeitsbereich.....	5
1.2.	Urheberschutz.....	5
1.3.	Wortmarken und Bildmarken.....	6
1.4.	Symbole.....	7
1.4.1.	Allgemeine Warnsymbole.....	7
1.4.2.	Spezielle Warnsymbole.....	8
2.	Wichtige Erläuterungen	9
2.1.	Änderungsvorbehalt	9
2.2.	Bestimmungsgemäße Verwendung	9
2.3.	Technischer Zustand	11
2.4.	Zulassungen.....	12
2.5.	Sicherheitshinweise	13
2.6.	Haftungsausschluss.....	14
2.7.	Kontaktinformationen	14
2.8.	Hilfestellung erhalten.....	14
3.	Überblick	15
4.	Voraussetzungen	16
5.	Lizenzen	17
6.	Basis-Wissen.....	18
6.1.	Namensdefinition, Abkürzungen und Bezeichnungen.....	18
6.2.	Definition „sicherer Zustand“	21
6.3.	Modbus - Kurzeinführung und Überblick.....	21
6.4.	Modbus RTU und PiXtend eIO Protokoll-Definitionen.....	22
6.5.	PiXtend eIO Protokoll-Kurzeinführung und Überblick	22
6.6.	RS-485/EIA-485 - Überblick zur seriellen Kommunikation.....	23
6.7.	Modbus RTU Geschwindigkeit.....	24
6.8.	Serielle Schnittstelle vorbereiten für eigenes SD-Karten Abbild	25
6.9.	PiXtend V1.3 / V2 -S- mit USB-zu-RS485 Dongle erweitern.....	26
6.10.	PiXtend V2 -L- – Serielle Schnittstelle prüfen.....	27
6.11.	PiXtend eIO Watchdog-Timer.....	29
6.11.1.	Watchdog-Timer Verhalten.....	29
6.11.2.	Watchdog-Timer Modi.....	29
6.11.3.	Watchdog-Timer Ausgang.....	30
6.11.4.	Watchdog-Timer - Ausgänge abschalten	30
6.12.	Zähler Funktion im PiXtend eIO Digital One	30
6.12.1.	Standardfunktion der Zähler	30
6.12.2.	Funktion der 2 Kanal Zähler	31
6.13.	Hyper-Logic IO's.....	32
6.14.	Gerätekonfiguration per Modbus RTU in CODESYS V3.5	33
6.15.	Anzahl unterstützter Geräte.....	34
6.16.	Mehrzweckausgänge-Digitalausgang 0, 4 und 7	34
6.17.	Strom und Spannung-Umrechnungsfaktoren.....	34
6.18.	Zwei Bytes zu einem 16 Bit Wert kombinieren.....	35
6.18.1.	CODESYS V3.5.....	35
6.18.2.	Python	35
6.18.3.	C.....	35
6.19.	Einen 16 Bit Wert in zwei Bytes zerlegen.....	35
6.19.1.	CODESYS V3.5	35
6.19.2.	Python	35
6.19.3.	C.....	35
6.20.	Fehlersignalisierung bei PiXtend eIO Geräten	36
7.	Geräte-LEDs – Signalisierung	37
7.1.	Signalisierung: Versorgungsspannung LED „+5V“	37
7.2.	Signalisierung: Kommunikation LED „COM“	37
7.3.	Signalisierung: Fehler LED „ERR“	38
8.	PiXtend eIO - Gerätekonfiguration	39
8.1.	Überblick	39
8.2.	Geräte-Adresse.....	40
8.3.	Serielle Konfiguration	47
8.4.	Protokollauswahl	49
8.5.	Busterminierung.....	50
9.	Modbus RTU - Protokoll.....	51
9.1.	Einführung.....	51
9.2.	Übersicht.....	51

9.3.	Unterstützte Funktion Codes	52
9.3.1.	PiXtend eIO Digital One - Modbus Adressen und Funktionen.....	52
9.3.2.	PiXtend eIO Analog One - Modbus Adressen und Funktionen.....	69
9.4.	Modbus RTU Fehlernummern.....	77
9.5.	Beispiel CODESYS V3.5	78
9.5.1.	PiXtend eIO Digital One	78
9.5.2.	PiXtend eIO Analog One.....	91
9.6.	Beispiel Python.....	104
9.6.1.	PiXtend eIO Digital One	104
9.6.2.	PiXtend eIO Analog One.....	107
9.7.	Beispiel C	110
9.7.1.	PiXtend eIO Digital One	110
9.7.2.	PiXtend eIO Analog One.....	110
10.	PiXtend eIO Protokoll.....	111
10.1.	Einführung.....	111
10.2.	Übersicht.....	111
10.3.	Ablauf der Kommunikation.....	112
10.4.	Protokollaufbau	114
10.5.	PiXtend eIO Digital One.....	116
10.5.1.	Übersicht unterstützter Befehle - Basic.....	116
10.5.2.	Übersicht der Bits im Error Register.....	117
10.5.3.	Beispiel - Eingänge lesen.....	120
10.5.4.	Beispiel - Ausgänge setzen	121
10.5.5.	Beispiel - Error Register auslesen.....	122
10.6.	PiXtend eIO Analog One.....	123
10.6.1.	Übersicht unterstützter Befehle - Basic.....	123
10.6.2.	Übersicht der Bits im Error Register.....	123
10.6.3.	Beispiel - Eingänge lesen.....	126
10.6.4.	Beispiel - Ausgänge setzen	126
10.6.5.	Beispiel - Error Register auslesen.....	127
10.7.	PiXtend eIO Protokoll - Advanced	127
10.7.1.	PiXtend eIO Digital One	127
10.7.2.	Beispiel - Zähler verwenden.....	142
10.7.3.	Beispiel - Hyper-Logic Prozessor verwenden.....	145
10.7.4.	Beispiel - Watchdog-Timer aktivieren.....	146
10.7.5.	Beispiel - Status auslesen	147
10.7.6.	PiXtend eIO Analog One.....	148
10.7.7.	Beispiel - Watchdog-Timer aktivieren.....	151
10.7.8.	Beispiel - Status auslesen	152
10.8.	PiXtend eIO Protokoll - Fehlernummern.....	153
11.	Abbildungsverzeichnis.....	154
12.	Tabellenverzeichnis	155

1. Hinweise zu dieser Dokumentation

Bewahren Sie diese Dokumentation auf!

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

1.1. Gültigkeitsbereich

Die vorliegende Dokumentation gilt ausschließlich für PiXtend eIO Geräte in den folgenden Ausführungen:

- PiXtend eIO Digital One Basic (Artikelnummer: 50199 007)
- PiXtend eIO Digital One Pro (Artikelnummer: 50199 008)
- PiXtend eIO Analog One Basic (Artikelnummer: 50199 009)
- PiXtend eIO Analog One Pro (Artikelnummer: 50199 010)

Alle weiteren Dokumente zu diesen Produkten finden Sie auf unserer Homepage unter <https://www.pixtend.de/downloads/>

Hinweis:

Wird im Dokumentationstext nur der Name PiXtend eIO verwendet, so gelten die genannten Informationen für beide Produkte (PiXtend eIO Digital One und PiXtend eIO Analog One) gleichermaßen.

1.2. Urheberrecht

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

Copyright 2019 © Kontron Electronics GmbH

1.3. Wortmarken und Bildmarken

- „Raspberry Pi“ und das zugehörige Logo sind geschützte Markenzeichen der Raspberry Pi Foundation – www.raspberrypi.org
- „CODESYS“ und das zugehörige Logo sind geschützte Markenzeichen der Firma 3S-Smart Software GmbH – www.codesys.com
- „PiXtend“, „ePLC“ und das zugehörige Logo sind geschützte Markenzeichen der Firma Kontron Electronics GmbH – www.kontron-electronics.de
- „AVR“, „ATmega“ und das zugehörige Logo sind geschützte Markenzeichen der Atmel Corporation – www.atmel.com bzw. Microchip Technology Corporation www.microchip.com
- „Debian“ und „Raspbian“ sind geschützte Markenzeichen des Debian Project – www.debian.org
- „I2C“ bzw. „I²C“ sind geschützte Markenzeichen von NXP Semiconductors – www.nxp.com
- „Arduino“ ist ein geschütztes Markenzeichen der Arduino AG – www.arduino.cc
- „Modbus“ & das zugehörige Logo sind geschützte Markenzeichen der Modbus Organization, Inc. – www.modbus.org
- „Node-RED“ & das zugehörige Logo sind geschützte Markenzeichen der JS Foundation – js.foundation – www.nodered.org
- „Java“ & das zugehörige Logo sind geschützte Markenzeichen der Oracle Corporation – www.oracle.com
- Das Logo „C++“ ist ein geschütztes Markenzeichen der Standard C++ Foundation – www.isocpp.org
- Python und das dazugehörige Logo sind geschützte Markenzeichen der Python Software Foundation – <https://www.python.org/psf/>
- Netbeans und das dazugehörige Logo sind geschützte Markenzeichen der The Apache Software Foundation – www.netbeans.org
- C# wurde von Microsoft entwickelt und herausgegeben, das dazugehörige Logo wurde von von Chris McKee unter der CC BY-SA 4.0 Lizenz veröffentlicht.
<https://www.microsoft.com> – <https://creativecommons.org/licenses/by-sa/4.0/>

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

1.4. Symbole

1.4.1. Allgemeine Warnsymbole

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.

1.4.2. Spezielle Warnsymbole



Heiße Oberfläche!

NICHT berühren! Vor der Wartung abkühlen lassen.



Stromschlag!

Dieses Symbol warnt vor Gefahren durch Stromschläge (> 60V) beim Berühren von Produkten oder Teilen von Produkten. Die Nichtbeachtung der gesetzlich vorgeschriebenen Vorsichtsmaßnahmen kann es Ihr Leben/Gesundheit gefährden und/oder zu Schäden an Ihrer Anlage führen.



ESD-empfindliches Gerät!

Dieses Symbol weist darauf hin, dass die elektronischen Leiterplatten und ihre Komponenten empfindlich auf statische Elektrizität reagieren. Daher ist bei allen Handhabungsvorgängen und Inspektionen dieses Produkts Vorsicht geboten, um die Produktintegrität jederzeit zu gewährleisten.

2. Wichtige Erläuterungen

Dieses Kapitel enthält Informationen zu rechtlichen Grundlagen, die bestimmungsgemäße Verwendung des hier beschriebenen Produkts, den technischen Zustand bei der Lieferung und wichtige Sicherheitshinweise.

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

Die PiXtend eIO Geräte erfüllen die Funktion von Ein- und Ausgabeeinheiten für elektrische Signale. Es können Sensorsignale erfasst, ausgewertet und Aktoren angesteuert werden. Die Spezifikationen der Ein- und Ausgänge entsprechen der internationalen Norm für speicherprogrammierbare Steuerungen, kurz: SPS – IEC/EN 61131-2. Außerdem entsprechen die Geräte den EMV-Anforderungen der Zone B (allgemeine Industrieumgebung) aus der genannten Norm.

Die Kommunikation zwischen PiXtend eIO und anderen Geräten erfolgt über eine serielle Schnittstelle (RS485/EIA485). Dabei handelt es sich um ein sogenanntes Bussystem, in dem bis zu 32 Teilnehmer miteinander kommunizieren können. PiXtend eIO erfüllt in einem solchen Bussystem (auch Netzwerk genannt) die Funktion eines „Slave“. Das bedeutet, dass den Geräten Informationen gesendet oder solche abgefragt werden können. Ein Slave wird am Bus immer nur dann aktiv, wenn er von einem „Master“ angesprochen, also zur Kommunikation aufgefordert wird. Eine direkte Datenübertragung zwischen den Slaves findet nicht statt. Mögliche Master für die PiXtend eIO Geräte sind Personal Computer (PC), industrielle Steuerungen (SPS, IPC), Raspberry Pi Computer, Arduino, Embedded-Geräte, speziell ausgestattete Smartphones & Tablets und viele weitere Computersysteme, die über eine RS485-Schnittstelle verfügen oder nachgerüstet werden können.

Im Besonderen ist die Kombination von PiXtend eIO Geräten mit der Steuerung PiXtend V2 -L- zu nennen. Dies ist der von Kontron Electronics empfohlene Referenz-Bus-Master. Die PiXtend eIO Geräte beherrschen zwei Protokolle – „Modbus RTU“ und „PiXtend eIO ASCII Protokoll“. Grundlegende Einstellungen für den Betrieb am Bus (Baudrate, Parität, Stoppbits, Adresse, Protokoll und Abschlusswiderstand) werden vor der Verwendung mit mechanischen Schaltern am Gerät eingestellt. Die Programmierung der Eingänge, Ausgänge und Sonderfunktionen kann unter anderem mit der Profi-Software „CODESYS V3“ der Firma 3S Smart Software Solutions GmbH geschehen.

Kontron Electronics bietet noch weitere, aus den Bereichen Automation, IT und Home Automation stammende, Programmiersprachen und Systeme an, die der Kunde zur Ansteuerung der PiXtend eIO Geräte einsetzen kann (C, C++, C#, Python, Java, FHEM, Node-RED). Dafür existieren jeweils Informationen und Beispiele von Kontron Electronics. Beide Protokolle sind offengelegt und können vom Kunden – auf eigene Verantwortung – in anderen Programmiersprachen und -Systemen implementiert werden.

Im Gegensatz zu früheren Produkten der Firma Kontron Electronics werden die PiXtend eIO Geräte ausschließlich als Fertiggeräte geliefert, die sofort einsatzbereit sind und keine Endmontage durch den Kunden erfordern.

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

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

Vorgesehene Einsatzbereiche sind:

- Maschinen-, Anlagen- und Gerätebau
- Prüfgeräte
- Labor- und Fertigungsautomation
- Home-Automation / Smart-Home
- Prototyping in der Industrie (Entwicklungsabteilungen)
- Einsatz im Bildungsbereich (technische Schulen, Hochschulen und Universitäten)
- IoT (Internet of Things), IIoT (Industrial Internet of Things) Konzepte und Lösungen
- Industrie 4.0 Konzepte und Lösungen
- Mobile Anwendungen – Hilfsfunktionen im Automotive-Sektor

Alle PiXtend eIO Geräte sind für Personen ab 14 Jahren, die das Sicherheitsdatenblatt und die Handbücher gelesen und verstanden haben, geeignet.

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 eIO eingebaut oder verwendet werden soll.

2.3. Technischer Zustand

Jedes PiXtend eIO Gerät wird, unabhängig von dessen Variante, mit einer definierten Konfiguration ausgeliefert:

PiXtend eIO Digital One

- Alle digitalen Eingänge sind für den 24V Bereich konfiguriert (kein Jumper gesteckt)
- Adresse: 1, Baudrate: 19200 baud, Paritäts-Bit: Even, Stopp-Bits: 1 (8E1)
- Terminierung inaktiv, Modus/Protokoll: Modbus RTU
- Die Mikrocontroller-Firmware entspricht bei Auslieferung immer der neusten, von Kontron Electronics veröffentlichten Version. Die aktuelle Version kann auf unserer Homepage eingesehen werden

PiXtend eIO Analog One

- Alle analoge Spannungseingänge sind für den 0..10 V Bereich konfiguriert (kein Jumper gesteckt)
- Adresse: 3, Baudrate: 19200 baud, Paritäts-Bit: Even, Stopp-Bits: 1 (8E1)
- Terminierung inaktiv, Modus/Protokoll: Modbus RTU
- Die Mikrocontroller-Firmware entspricht bei Auslieferung immer der neusten, von Kontron Electronics GmbH veröffentlichten Version. Die aktuelle Version kann auf unserer Homepage eingesehen werden

Jedes Gerät kann in zwei verschiedenen Ausführungen / Varianten bezogen werden:

Basic

- Basisausführung ohne Gehäuse

Pro

- Profi-Ausführung mit Edelstahlhaube & Hutschienegehäuse

Wenn Sie eine andere Ausführung oder eine andere Hardware- und Software-Kombination benötigen, richten Sie Ihre Anfrage bitte direkt an uns (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 führt 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

Lesen Sie das komplette Dokument und die Sicherheits- und Anschlusshinweise, bevor Sie mit dem Anschluss oder dem Betrieb von PiXtend eIO Geräten beginnen. Bewahren Sie dieses Dokument auf, auch nachdem Sie alle Komponenten in Betrieb genommen haben.

⚠ CAUTION

Für Schäden jeglicher Art, welche durch die Nichtbeachtung der Datenblätter und Bedienungsanleitungen entstehen, übernimmt die Firma Kontron Electronics GmbH keinerlei Haftung. Der Garantie- bzw. Gewährleistungsanspruch und die Zulassung erlischt.

- PiXtend eIO darf nur mit der dafür vorgeschriebenen Spannung (24V DC $\pm 20\%$) und einem mit VDE- und CE-Prüfzeichen (für Europa) versehenen Netzgerät mit stabilisierter Ausgangsspannung betrieben werden. Das Netzteil muss den gesetzlichen Vorschriften des jeweiligen Landes entsprechen, in dem PiXtend eIO zum Einsatz kommt.
- Das Gerät ist nur für den Gebrauch in trockenen und sauberen Räumen ausgelegt und nicht für den Betrieb im Freien oder in Feuchträumen geeignet!
- Die zulässige Betriebstemperatur liegt zwischen 0°C und 60°C.
- PiXtend eIO Geräte sowie alle damit verbundenen Kabel, Steckverbinder und Netzteile sind von Flüssigkeiten fernzuhalten.
- PiXtend eIO Geräte dürfen nicht in der Umgebung von brennbaren Flüssigkeiten, Gasen oder Stäuben verwendet werden.
- Bei einer Reparatur dürfen nur Original- bzw. empfohlenen Ersatzteile verwendet werden. Fragen Sie im Zweifel immer erst bei Kontron Electronics an! – support@pixtend.de
- PiXtend eIO darf an keiner Stelle Netzspannung mit 230V, 115V AC oder eine andere gefährliche Spannung größer 50V angelegt werden. **Achtung: Lebensgefahr!**
- Weder PiXtend eIO noch die Zubehörteile gehören in Kinderhände (unter 14 Jahren).
- In Schulen, Hobbywerkstätten und Ausbildungseinrichtungen ist der Betrieb durch geschultes Personal zu überwachen.

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 gänzlich 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
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 entsprechenden Kapiteln der Handbücher und in den FAQ's auf unserer Homepage nachzusehen.

Wenn die Frage dort nicht geklärt wird, können Sie uns per E-Mail (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 eIO (economic Input and Output) ist das I/O-Erweiterungssystem für alle, die auf der Suche nach kostengünstigen Ein- und Ausgängen sind, digital und analog. Die Datenanbindung erfolgt mit dem robusten RS485 Bus. Falls nicht vorhanden, kann dieser an jedem Computersystem nachgerüstet werden.

Die eIO-Geräte und deren Ein- und Ausgänge entsprechen der Industrienorm IEC 61131-2 und sind damit für den rauen Arbeitseinsatz in Laboren, Industrie- und Gewerbebetrieben vorgesehen. Die Einhaltung des weltweiten Industriestandards hat große Vorteile in Bezug auf Stabilität und Störunempfindlichkeit. Alle gängigen Sensoren und Aktoren finden an PiXtend eIO leicht Anschluss.

Innovative Features, wie die besonders schnelle digitale „Hyper Logic“, bringen zusätzlichen Mehrwert für Ihr Projekt. Einfache logische Verknüpfungen zwischen Ein- und Ausgängen können von den Geräten selbst ausgeführt werden, ohne dass eine Datenübertragung über den Bus notwendig ist. Digitale Eingänge lassen sich zusätzlich zum Zählen von Ereignissen nutzen. Ein Watchdog-Timer überwacht die per CRC-Prüfsumme gesicherte Datenübertragung und versetzt die Ausgänge bei Bedarf in einen einstellbaren, sicheren Zustand.

Der Name „PiXtend“ ist bereits von unseren Raspberry Pi basierten SPS- bzw. IoT-Geräten bekannt. Das eIO-System fügt sich optisch, haptisch und funktional perfekt an diese Steuerungen an. Es lässt sich darüber hinaus an fast jedes andere Computersystem und Embedded Gerät anschließen.

In Bezug auf die Kommunikation und die Protokolle, ist PiXtend eIO einfach und anwenderfreundlich. Durch die Modbus RTU Unterstützung fügen sich eIO-Geräte ebenso in vorhandene Systeme ein. Das Protokoll ist sicher und bereits über viele Jahre industrieeerprobt. Zusätzlich lassen sich die eIO-Geräte mit dem besonders einfachen und schlanken „PiXtend eIO Protokoll“ ansprechen, welches die Daten in Klartext übermittelt (ASCII). Wir bieten Open Source Beispielprogramme für C, C#, Python, CODESYS (61131-3) und Java an. Die Einbindung in eigene Applikationen ist ein Kinderspiel.

Die Grundeinstellungen lassen sich direkt und unkompliziert am Gerät durchführen, ohne Software und ohne Vorkenntnisse.

4. Voraussetzungen

Nachfolgend finden Sie Informationen zu den Voraussetzungen, die mindestens erfüllt sein müssen, um ein PiXtend eIO Gerät in Betrieb zu nehmen beziehungsweise einzusetzen.

- Systemanforderungen für Entwicklungsrechner (PC)
 - Microsoft Windows 7/8/10 (32/64 Bit), Linux, macOS
 - Ausreichend Festplattenplatz zur Installation neuer Programme
 - Entsprechende Rechte (z.B. Administratorrechte) die eine Installation erlauben
 - Serieller RS485 Anschluss oder USB-zu-Serial RS485 Dongle, sofern das Gerät direkt vom Computer aus angesteuert werden soll. Alternativ wird eine SPS oder ein embedded Gerät mit seriellem RS485 Anschluss benötigt.
 - Terminal- oder Konsolenprogramm wie „Putty“ mit Zugriffserlaubnis auf die seriellen Schnittstellen. Das ist dann hilfreich, wenn das PiXtend eIO Protokoll zur direkten Eingabe von Befehlen verwendet werden soll.
 - Für Modbus RTU ist eine Modbus RTU Master Software am PC erforderlich.
- Systemanforderungen für SPS
 - Serieller RS485 Anschluss
 - Ist keine RS485 Schnittstelle vorhanden, kann ein USB-zu-Serial Dongle verwendet werden um eine serielle Schnittstelle nachzurüsten oder ein Busumsetzer, welcher der SPS eine RS485 Schnittstelle zur Verfügung stellt.
 - Modbus RTU Master Baustein/Funktion für Modbus RTU Kommunikation oder serielle Kommunikation mittels des einfachen und kostenfreien PiXtend eIO ASCII Protokolls.
- Benötigte Hardware
 - PiXtend eIO Gerät (www.pixtend.de)
 - Modbus RTU – RS485 taugliches Kabel
 - 24V Spannungsversorgung

Um PiXtend eIO Geräte effizient einsetzen zu können, sollten Sie über Grundwissen in einer oder mehreren der folgenden Programmiersprachen oder Programmierwerkzeuge verfügen. Die nachfolgende Liste stellt einen kleinen Auszug dar. Es kann jede Programmiersprache verwendet werden, die Zugriff auf eine serielle Schnittstelle auf dem gewünschten Zielsystem bietet.

- C
- C++
- C#
- Python
- Java
- Node-RED
- CODESYS

Für die diese Sprachen und Programmiersysteme gibt es verschiedene Beispiele und Demo-Projekte im Download-Bereich auf unserer Homepage. Wer zudem am PC mit Visual Studio 2017 oder später arbeitet, kann auf ein Windows Beispiel mit graphischer Oberfläche zurückgreifen. Ohne tiefer in die Programmierung des PiXtend eIO Protokolls einzusteigen, kann man PiXtend eIO Analog One und/oder PiXtend eIO Digital One einfache Befehle schicken. Es lässt sich der Zustand der digitalen und analogen Ausgänge verändern oder der Zustand der jeweiligen Eingänge abfragen.

5. Lizenzen

Alle Beispiele und Programme zur Verwendung des PiXtend eIO Systems wurden von Kontron Electronics GmbH unter der GNU GPL v3 Lizenz veröffentlicht und können frei genutzt und modifiziert werden. Der Einsatz in eigenen Projekten sowie zur kommerziellen Nutzung, in Verbindung mit PiXtend eIO Produkten, ist erlaubt.



Es gibt derzeit für das PiXtend eIO Protokoll Beispielprojekte für die Programmiersprachen „C#“, Python, Node-RED, Java (Netbeans 11), CODESYS V3.5 und „C“ für den Raspberry Pi Computer.

Für Modbus RTU sind Beispiele für die Programmiersprachen Python, C für den Raspberry Pi Computer und CODESYS V3.5 verfügbar.

Die Dateien können frei verteilt und in beliebiger Anzahl verwendet werden, der Lizenzhinweis und die Angabe des Firmennamens, Kontron Electronics GmbH, müssen erhalten bleiben und dürfen nicht entfernt werden.

Verwenden Sie entsprechende Zusatzbibliotheken oder Packages wie pySerial, pyModbus oder libModbus, beachten Sie bitte die jeweiligen Lizenzbestimmungen.

6. Basis-Wissen

Das Kapitel Basis-Wissen enthält wichtige Informationen rund um das PiXtend eIO System und sollte aufmerksam und vollständig gelesen werden, bevor man mit der Inbetriebnahme von PiXtend eIO Geräten beginnt.

6.1. Namensdefinition, Abkürzungen und Bezeichnungen

Die nachfolgende Tabelle zeigt eine Übersicht verschiedener Begriffe, Namen, Abkürzungen und Bezeichnungen, die in dieser Dokumentation verwendet werden. Sie dient als Leitfaden und zum besseren Verständnis der Dokumentation.

Begriff	Beschreibung
A	Ein großes A steht für das Wort Ausgang. Diese Abkürzung kann im Zusammenhang mit analogen und/oder digitalen Ausgängen auftreten.
E	Ein großes E steht für das Wort Eingang. Diese Abkürzung kann im Zusammenhang mit analogen und/oder digitalen Eingängen auftreten.
I	Ein großen I steht für das englische Wort Input, zu Deutsch: Eingang. Siehe Begriff E.
O	Ein großen O steht für das englische Wort Output, zu Deutsch: Ausgang. Siehe Begriff A.
AI	Der Begriff AI steht für den englischen Ausdruck Analog Input, zu Deutsch: Analogeingang.
AO	Der Begriff AO steht für den englischen Ausdruck Analog Output, zu Deutsch: Analogausgang.
DI	Der Begriff DI steht für den englischen Ausdruck Digital Input, zu Deutsch: Digitaleingang.
DO	Der Begriff DO steht für den englischen Ausdruck Digital Output, zu Deutsch: Digitalausgang.
HIGH	Der Begriff HIGH, zu finden als HIGH-Pegel oder HIGH Signal, beschreibt in dieser Dokumentation den Zustand, wenn an einem Digitaleingang ein entsprechendes elektrisches Signal anliegt, welches in den Bereich einer logischen 1, ebenso bezeichnet als An oder True (Wahr), fällt. Die genauen elektrischen Bestimmungen stehen im PiXtend eIO Hardware Handbuch.
LOW	Der Begriff LOW, zu finden als LOW-Pegel oder LOW Signal, beschreibt in dieser Dokumentation den Zustand, wenn an einem Digitaleingang ein entsprechendes elektrisches Signal anliegt, welches in den Bereich einer logischen 0, auch bezeichnet als Aus oder False (Falsch), fällt. Die genauen elektrischen Bestimmungen stehen im PiXtend eIO Hardware Handbuch.
An	Beschreibt den Zustand, dass eine PiXtend eIO Gerätefunktion eingeschaltet bzw. aktiv ist. Andere Bezeichnungen mit der gleichen Bedeutung sind die Zahl 1, der logische Zustand 1 oder der boolesche Ausdruck True (Wahr).
Aus	Beschreibt den Zustand, dass eine PiXtend eIO Gerätefunktion ausgeschaltet oder inaktiv ist. Andere Bezeichnungen mit der gleichen Bedeutung sind die Zahl 0, der logische Zustand 0 oder der boolesche Ausdruck False (Falsch).
1	Siehe Begriff An.
0	Siehe Begriff Aus.
True	Siehe Begriff An. Siehe aus Wikipedia Begriff: Aussagenlogik
False	Siehe Begriff Aus. Siehe aus Wikipedia Begriff: Aussagenlogik
MC	Als Mikrocontroller (μ Controller, μ C, MCU) bezeichnet man Halbleiterchips, die einen Prozessor und zugleich Peripheriefunktionen enthalten. In vielen Fällen befindet sich der Arbeits- und Programmspeicher teilweise oder komplett auf demselben Chip. Ein Mikrocontroller ist ein Ein-Chip-Computersystem. Für manche Mikrocontroller wird der Begriff System-on-a-Chip oder SoC verwendet. Auf modernen Mikrocontrollern finden sich häufig komplexe Peripheriefunktionen wie zum Beispiel CAN- (Controller Area Network), LIN- (Local Interconnect Network), USB- (Universal Serial Bus), I ² C- (Inter-Integrated Circuit), SPI- (Serial Peripheral Interface), serielle oder Ethernet-Schnittstellen, PWM-Ausgänge, LCD-Controller und -Treiber sowie Analog-Digital-Umsetzer. Einige Mikrocontroller verfügen über programmierbare digitale und/oder analoge bzw. hybride Funktionsblöcke. (Wikipedia, https://de.wikipedia.org/wiki/Mikrocontroller , Stand Oktober 2019)

Begriff	Beschreibung
Power Cycle	Wird ein PiXtend eIO Gerät von der Versorgung getrennt und nach kurzer Wartezeit wieder mit Strom versorgt. Durch diesen Vorgang werden alle Zustände im PiXtend eIO Gerät zurückgesetzt und alle Konfigurationseinstellungen beim Neustart übernommen.
Watchdog	Der Begriff Watchdog (englisch für Wachhund; auch Watchdog-Timer genannt) bezeichnet eine Funktion zur Ausfallerkennung eines digitalen Systems, vorwiegend in Steuerungsanwendungen. Wird eine mögliche Fehlfunktion erkannt, wird dies gemäß Systemvereinbarung an andere Komponenten signalisiert (z. B. Umschalten auf ein redundantes System), eine geeignete Sprunganweisung bzw. ein Reset zur selbsttätigen Behebung des Ausfalls eingeleitet oder ein sicheres Abschalten veranlasst. (Wikipedia, https://de.wikipedia.org/wiki/Watchdog , Stand Oktober 2019)
Watchdog-Timer	Siehe Begriff Watchdog.
Nachricht	In der Netzwerktechnik, auf die sich diese Dokumentation stützt, versteht man unter Nachricht ein Datenpaket. Dieses Datenpaket kann beispielsweise einen Befehl oder eine Abfrage für ein Endgerät an einem Datenbus sein. Erhält das Endgerät eine Nachricht, kann es diese auswerten und darauf antworten. Ausführlichere Informationen zu diesem Thema stehen auf Wikipedia unter https://de.wikipedia.org/wiki/Datenpaket (Stand Oktober 2019).
Telegramm	Siehe Begriff Nachricht.
RS485	Auch EIA-485 genannt, ist die Bezeichnung für eine physische Schnittstelle zur seriellen Datenübertragung. Auf Basis dieser Schnittstelle lassen sich verschiedenste Geräte miteinander verbinden und ermöglichen zusätzlich eine Kommunikation unter den Geräten. Weitere Informationen zur RS485 siehe Wikipedia unter https://de.wikipedia.org/wiki/EIA-485 (Stand Oktober 2019). Informationen zu Bussystemen: https://de.wikipedia.org/wiki/Bus_(Datenverarbeitung)#Bussysteme
RS232	Bezeichnet eine serielle Schnittstelle zur Datenübertragung zwischen zwei Geräten oder Teilnehmern. Mehr Informationen finden Sie auf Wikipedia unter https://de.wikipedia.org/wiki/Serielle_Schnittstelle (Stand Oktober 2019).
USB-zu-RS485	Bezeichnet einen speziellen Adapter mit dem ein Computer unter Verwendung einer USB-Schnittstelle um eine RS485 Schnittstelle erweitert werden kann. Siehe dazu Begriff RS485 und Wikipedia unter https://de.wikipedia.org/wiki/Universal_Serial_Bus für weitere Informationen zu USB.
Baudrate	Die Baudrate bezeichnet die Geschwindigkeit der Übertragung eines Symbols pro Sekunde in einem physischen Medium. Die Baudrate muss beim Sender und Empfänger immer gleich sein. Eine sehr ausführliche Beschreibung liefert Wikipedia unter https://de.wikipedia.org/wiki/Baud (Stand Oktober 2019).
Parität	Das Paritätsbit einer Folge von Bits dient als Ergänzungsbit, um die Anzahl der mit 1 belegten Bits (inklusive Paritätsbit) der Folge als gerade oder ungerade zu ergänzen. Die „Parität“ der Bitfolge heißt gerade (englisch even), wenn die Anzahl der mit 1 belegten Bits in der Folge gerade ist, andernfalls ungerade (englisch odd). Die ausführliche Beschreibung steht auf Wikipedia unter https://de.wikipedia.org/wiki/Paritätsbit (Stand Oktober 2019).
Even	Englisches Wort für „gerade“, siehe dazu Parität.
Odd	Englisches Wort für „ungerade“, siehe dazu Parität.
Stoppbit	Ein Startbit und bis zu zwei Stoppbits werden bei der asynchronen seriellen Datenübertragung verwendet, um dem Empfänger eine Synchronisation auf jedes übertragene Zeichen zu ermöglichen. Das Start- und die Stoppbits haben dabei immer die gleichen, aber voneinander unterschiedliche logische Pegel. Ein Startbit wird vor dem zu übertragenen Datenwort gesendet, das oder die Stoppbits danach. Die vollständige Beschreibung steht auf Wikipedia unter https://de.wikipedia.org/wiki/Stoppbit (Stand Oktober 2019), siehe auch Begriffe Parität

Begriff	Beschreibung
	und Baudrate.
Overflow	Bezeichnet als Overflow-Bit (Überlauf), gibt bei Zählern an, dass sie das Ende ihres Zahlenraums erreicht bzw. darüber hinaus gezählt haben. Dies gilt für die positive und negative Zählrichtung. Je nach System und Gerät kann ein Overflow Ereignis einen Fehler verursachen oder wie es oft in der Automatisierungstechnik vorkommt, die Zähler zählen einfach weiter. Nicht jedes System verfügt und stellt ein Overflow-Bit zur Verfügung. Eine Erklärung ist auch auf Wikipedia zu finden unter https://de.wikipedia.org/wiki/Arithmetischer_Überlauf (Stand Oktober 2019).
Überlauf	Siehe Begriff Overflow.
Steigende Flanke	Ist der Wechsel eines Signals von einer logischen 0 zu einer logischen 1.
Fallende Flanke	Ist der Wechsel eines Signals von einer logischen 1 zu einer logischen 0.
SF	Abkürzung für Steigende Flanke.
FF	Abkürzung für Fallende Flanke.
Rising Edge	Englischer Begriff für Steigende Flanke.
Falling Edge	Englischer Begriff für Fallende Flanke.
RE	Englische Abkürzung für Rising Edge, zu Deutsch: Steigende Flanke
FE	Englische Abkürzung für Falling Edge, zu Deutsch: Fallende Flanke

Tabelle 1: Namensdefinition, Abkürzungen und Bezeichnungen

6.2. Definition „sicherer Zustand“

Das PiXtend eIO System verfügt über einen vom Benutzer aktivierbaren Watchdog-Timer. Ist dieser aktiviert und ein Kommunikationsausfall tritt auf, löst der Watchdog-Timer nach der vom Benutzer eingestellten Wartezeit aus. Läuft die Überwachungszeit ab, löst der Watchdog-Timer aus und der Mikrocontroller geht in einen „sicheren Zustand“. Dieser Zustand hat nachfolgende Eigenschaften:

- Alle digitalen und analogen Ausgänge verbleiben in ihrem aktuellen Zustand (Voreinstellung). Dies kann vom Anwender geändert werden, so dass alle Ausgänge abgeschaltet bzw. auf 0V und 0mA gesetzt werden.
- Die ERR-LED blinkt 3-mal kurz und 3-mal lang, dann erfolgt 1 Sekunde Pause, danach wiederholt sich die Abfolge.
- Der Mikrocontroller bzw. das PiXtend eIO-Gerät muss im Anschluss neu gestartet werden (Power Cycle).
- Es ist keine Kommunikation mit dem Gerät möglich, das heißt es kann kein Remote-Reset (Zurücksetzen) aus der Ferne erfolgen. Das Gerät muss Vorort von der Versorgung getrennt werden.

6.3. Modbus - Kurzeinführung und Überblick¹

Das Modbus-Protokoll, ein Kommunikationsprotokoll, basiert auf einer Master/Slave- bzw. Client/Server-Architektur. In der Industrie hat sich der Modbus zu einem De-facto-Standard entwickelt, es handelt sich um ein offenes Protokoll. Seit 2007 ist die Version Modbus TCP Teil der Norm IEC 61158.

Mittels Modbus können ein Master (ein PC) und mehrere Slaves (Mess- und Regelsysteme) verbunden werden. Es gibt zwei Versionen: Eine für die serielle Schnittstelle (EIA-232 und EIA-485) und eine für Ethernet.

Bei der Datenübertragung unterscheiden wir zwei Betriebsarten:

- Modbus RTU
- Modbus TCP

Jeder Busteilnehmer muss eine eindeutige Adresse besitzen. Die Adresse 0 ist dabei für einen Broadcast reserviert. Jeder Teilnehmer darf Nachrichten über den Bus senden. Die Kommunikation wird immer durch den Master initiiert und ein adressierter Slave antwortet.

¹Text von Wikipedia, Stand August 2019, <https://de.wikipedia.org/wiki/Modbus>

Lese- und Schreibzugriffe sind auf folgende Objekttypen möglich:

Objekttyp	Zugriff	Größe
Einzelner Ein-/Ausgang „Coil“	Lesen und Schreiben	1-Bit
Einzelner Eingang „Discrete Input“	nur Lesen	1-Bit
(analoge) Eingänge „Input Register“	nur Lesen	16-Bits
(analoge) Ein-/Ausgänge „Holding Register“	Lesen und Schreiben	16-Bits

Tabelle 2: Modbus RTU Objekttypen

Modbus RTU (RTU: Remote Terminal Unit, entfernte Terminaleinheit) überträgt die Daten in binärer Form, dies sorgt für einen guten Datendurchsatz. Die Daten lassen sich nicht direkt vom Menschen auswerten, sie müssen zuvor in ein lesbares Format umgesetzt werden. Die Datenübertragung wird zudem mit einer zyklische Redundanzprüfung (englisch cyclic redundancy check, oft zu sehen als „CRC“) abgesichert. Somit kann sowohl der Master als auch ein Slave die Richtigkeit der empfangenen Daten prüfen. Wird ein Fehler festgestellt, werden die Daten verworfen.

6.4. Modbus RTU und PiXtend eIO Protokoll-Definitionen

Beide Kommunikationsprotokolle basieren auf einer Master/Slave Architektur. Der Master muss immer eine Anfrage (Nachricht oder Telegramm) an einen Slave senden, nur dieser eine Slave darf antworten. Slaves können und dürfen nicht untereinander kommunizieren, der Master steuert die gesamte Kommunikation und verteilt die Informationen.

Beim Modbus RTU Protokoll für Master und Slave gibt es weitere Bezeichnungen die zu Verwirrung führen können.

- Modbus Master → Modbus Client
- Modbus Slave → Modbus Server

Beachten Sie diese unterschiedlichen Bezeichnungen, wenn Sie andere Literatur, Hinweise oder Informationen bezüglich Modbus RTU lesen.

6.5. PiXtend eIO Protokoll-Kurzeinführung und Überblick

Das PiXtend eIO Protokoll ist ein einfaches Klartextprotokoll der Firma Kontron Electronics GmbH, es ist darauf ausgelegt, über einen seriellen Bus übertragen zu werden.

Hinter dem PiXtend eIO Protokoll steht die Grundidee, dass die PiXtend eIO Geräte direkt von einem PC, über einen USB-zu-RS485 Dongle, angesprochen werden. Der Anwender kann ein serielles Programm oder ein Terminal Programm, mit Zugriff auf eine serielle Schnittstelle verwenden, um einen Befehl zu erteilen. Die Befehle können im Klartext per Tastatur eingegeben werden.

Verschiedene Befehle sind verfügbar, um die digitalen und analogen Ein- und Ausgänge einzulesen und zu setzen.

Ist ein RS485 Bussystem vorhanden, es wird nicht per Modbus RTU kommuniziert, dann ist das PiXtend eIO Protokoll perfekt geeignet. Es erweitert das bestehende System um digitale und analoge Ein- und Ausgänge. Zudem lässt sich das PiXtend eIO Protokoll einfach in eigene Programme integrieren, eine Vielzahl an Beispielen und Demoprojekten unterstützt den Anwender bei der schnellen Umsetzung.

6.6. RS-485/EIA-485 - Überblick zur seriellen Kommunikation

EIA-485², als RS-485 bezeichnet, ist ein Industriestandard für eine physische Schnittstelle für die asynchrone serielle Datenübertragung. Die symmetrische Leitung erhöht die elektromagnetische Verträglichkeit.

EIA-485 nutzt ein Leitungspaar, um den invertierten und einen nicht-invertierten Pegel eines 1-Bit-Datensignals zu übertragen. Am Empfänger wird aus der Differenz der beiden Spannungspegel das ursprüngliche Datensignal rekonstruiert. Das hat den Vorteil, dass sich Gleichtaktstörungen nicht auf die Übertragung auswirken und somit die Störsicherheit vergrößert wird.

Im Gegensatz zu anderen Bussen sind bei EIA-485 nur die elektrischen Schnittstellenbedingungen definiert. Das Protokoll kann anwendungsspezifisch gewählt werden. Wie zum Beispiel bei den PiXtend eIO Geräten, diese können mit dem Modbus RTU Protokoll angesprochen werden oder alternative mit dem Kontron Electronics eigenen PiXtend eIO Protokoll. Da die Übertragung seriell und je nach gewählter Geschwindigkeit erfolgt, muss beim Ansprechen der Geräte Zeit für die Kommunikation eingeplant werden.

Die Angaben der Tabelle basieren auf einer Nachricht an ein Digital One und ein Analog One Gerät mit dem PiXtend eIO Protokoll. Die Antwort des Slaves wird nicht berücksichtigt.

Baudrate	Zeit pro Bit	Zeit pro Byte	D-One Nachricht (16 Bytes)	A-One Nachricht (17 Bytes)
2400	416,7 µs	3,336 ms	53,34 ms	56,67 ms
9600	104,2 µs	0,836 ms	13,34 ms	14,17 ms
57600	17,4 µs	0,139 ms	2,23 ms	2,37 ms
115200	8,68 µs	0,069 ms	1,11 ms	1,18 ms

Tabelle 3: Auswahl an Baudraten und deren berechneter Übertragungsdauer

Beim PiXtend eIO Protokoll erhält der Master nach zehn Millisekunden eine Antwort vom Slave (zuzüglich der Übertragungszeit aus Tabelle 3), dieser Umstand erlaubt die Verwendung von PiXtend eIO mit PiXtend V1.x Boards.

NOTICE

Zwischen zwei Nachrichten bzw. Telegrammen, einem vollständigen Kommunikationszyklus, muss eine ausreichend lange Pause erfolgen. Der Master schickt eine Anfrage/Befehl an den Slave und wartet auf die Antwort. Hat der Master die Antwort vom Slave erhalten, muss eine Pause erfolgen.

Folgende Zeiten werden empfohlen:

- Slave Antwort-Wartezeit: 1000 ms
- Pausenzeit für Baudraten unter 9600: ≥ 4 ms
- Pausenzeit für Baudraten ab 9600: ≥ 2 ms

Diese Empfehlung gilt für das PiXtend eIO Protokoll und ist unabhängig von der Anzahl der Busteilnehmer.

Für Modbus RTU siehe Modbus Spezifikation 1.02, Kapitel Modbus Message RTU Framing.

²Informationen aus Wikipedia Stand August 2019, siehe <https://de.wikipedia.org/wiki/EIA-485>

6.7. Modbus RTU Geschwindigkeit

Bei Verwendung des Modbus RTU Protokolls fallen je nach Funktion Code wenig Daten an. Durch die binäre Kodierung ist der Ablauf effizienter als beim PiXtend eIO Protokoll für die gleiche Aufgabe (siehe vorherige Seite). Die nachfolgende Tabelle zeigt auf, wie schnell mit jedem Gerät kommuniziert sowie wie schnell und wie oft jedes Gerät pro Sekunde abgefragt werden kann.

Gerät	FC	Anzahl I/O's	Updates	Bemerkung
Digital One	02/15	8 I oder 8 O	166/s	8 Eingänge lesen oder 8 Ausgänge schreiben
Digital One	02/15	8 I und 8 O	100/s	Lesen und schreiben aller I/O's
Analog One	04/06	1 I oder O	100/s	Ein Register enthält 16 Bits (2 Bytes). Einen Eingang lesen oder einen Ausgang schreiben
Analog One	04/16	8 I und 6 O	41/s	Lesen und schreiben aller I/O's

Tabelle 4: Modbus RTU - Kommunikationsgeschwindigkeit

Die genannten Updates und Zykluszeiten pro Sekunde beziehen sich immer auf nur ein einzelnes Gerät, das mit einer Baudrate von 115200 angesprochen wird, mit acht Datenbits, keiner Parität und einem Stoppbit.

Die Zeiten wurden unter CODESYS V3.5 SP14 auf einem Raspberry Pi ermittelt ohne eine nebenläufige Anwendung oder Visualisierung (HMI). Bei Verwendung von erweiterten Funktionen oder größerer Programme können die genannten Zeiten abweichen, dies gilt insbesondere, wenn mehrere Busteilnehmer vorhanden sind.

Die Aktualisierungshäufigkeit unter CODESYS V3.5 hängt im Wesentlichen von drei Faktoren ab. Zum einen entscheidet der Buszyklustask des Modbus Masters darüber, wie schnell das Modbus Protokoll abgearbeitet wird und die damit verbundenen Daten verarbeitet werden. Mit Daten ist die Anfrage an den Slave und dessen Antwort gemeint.

Des Weiteren spielt die eingestellte Zykluszeit des jeweiligen Modbus RTU Kanals eine Rolle, diese Zeiteinstellung gibt vor in welchem Intervall mit einem Slave kommuniziert wird. Standardmäßig schlägt hier CODESYS V3.5 immer 100 ms vor.

Der letzte Punkt ist die Einstellung „Zeit zwischen den Frames [ms]“ im CODESYS V3.5 ModBus Master. Diese Einstellung steht üblicherweise auf zehn Millisekunden. Der Modbus Master schickt immer nur alle zehn Millisekunden eine Nachricht an einen Slave, egal was in den Modbus Slave-Kanälen als Trigger konfiguriert wurde.

Zum besseren Verständnis der Timings unter Modbus RTU finden Sie Informationen in der Modbus Spezifikation 1.02, Kapitel Modbus Message RTU Framing.

6.8. Serielle Schnittstelle vorbereiten für eigenes SD-Karten Abbild

Sie möchten keines unserer vorbereiteten CODESYS SD-Karten Abbilder einsetzen, in Python oder C programmieren und setzen auf unser Basis Image? Die nachfolgende Beschreibung hilft Ihnen, die serielle Schnittstelle des Raspberry Pi Computers entsprechend einzustellen. Dieser Abschnitt ist ebenso hilfreich, wenn Sie ein eigenes SD-Karten Abbild mit dem Raspbian Betriebssystem verwenden.

Damit die serielle Schnittstelle des Raspberry Pi Computers für eigene Zwecke genutzt werden kann, muss diese zuerst aktiviert und konfiguriert werden. Sie haben die serielle Schnittstelle bisher nicht genutzt? Mit folgenden Befehlen wird sie aktiviert und angepasst, sie sendet eigenständig keine Daten. Der Zugriff über die serielle Schnittstelle auf die Linux-Konsole wird durch diesen Vorgang ebenfalls deaktiviert. Diese Umstellung ist wichtig, um eine korrekte und konsistente Datenübertragung zu erhalten.

Die serielle Schnittstelle des Raspberry Pi wird über das Programm raspi-config aktiviert bzw. umgestellt:

```
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, die serielle Schnittstelle wird aktiviert und die sogenannte Login Shell wird deaktiviert. Das Programm raspi-config übernimmt diese Arbeit.

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

Einen Neustart führen Sie durch folgenden Befehl aus:

```
sudo reboot
```

NOTICE

Die genannten Schritte richten sich an Anwender eines PiXtend V2 -L- Boards, die die serielle RS485 Schnittstelle verwenden um mit den PiXtend eIO Geräten zu kommunizieren. PiXtend V2 -S- und PiXtend V1.x benötigen einen USB-zu-RS485 Dongle um mit den PiXtend eIO Geräten zu sprechen. Die serielle Schnittstelle des PiXtend V2 -S- und PiXtend V1.x sind nicht Modbus RTU fähig.

Sofern sie das PiXtend eIO Protokoll verwenden, können Sie die RS485 Schnittstelle auf dem PiXtend V1.x nutzen. Die PiXtend eIO Geräte machen eine zehn Millisekunden Pause, bevor sie die Antwort an den Master schicken. Somit bleibt dem Anwender Zeit den Transceiver von Senden auf Empfangen umzustellen.

6.9. PiXtend V1.3 / V2 -S- mit USB-zu-RS485 Dongle erweitern

Das PiXtend V2 -L- verfügt über eine RS485 Schnittstelle, die über das Setzen des GPIO 18 vom Raspberry Pi aktiviert werden kann. Das Modbus RTU Protokoll lässt sich somit sehr einfach verwenden.

PiXtend V1.3 und V2 -S- benötigt einen zusätzlichen USB-zu-RS485 Dongle, der die passende Hardware bereitstellt, um Modbus RTU zu nutzen. PiXtend V1.3 verfügt über eine RS485 Schnittstelle, diese ist nicht Modbus RTU fähig. PiXtend V2 -S- verfügt lediglich über eine RS232 Schnittstelle. Einen USB-zu-RS485 Dongle können Sie über unserem Shop beziehen.

Um unter CODESYS V3.5 Zugriff auf die zusätzliche serielle Schnittstelle zu erhalten und Modbus RTU nutzen zu können, ist folgendes zu beachten. Die CODESYS SD-Karten Abbilder sind standardmäßig eingestellt auf: Raspberry Pi serielle Schnittstelle mit der Gerätebezeichnung ttyAMA0 (auch bekannt als PL011 oder serial0). Möchten Sie die durch den USB-Dongle bereitgestellte serielle Schnittstelle verwenden, lautet der Gerätenamen üblicherweise ttyUSB0. Die Null kann eine andere Zahl sein, wenn bereits ein anderer Dongle am Raspberry Pi eingesteckt ist.

Wir empfehlen folgende Schritte durchzuführen, um den RS485 USB-Dongle unter CODESYS zu nutzen.

- PiXtend einschalten und CODESYS starten
- Per Konsolen-Anwendung (z.B. Putty) oder direkt per Bildschirm, Maus und Tastatur am Raspberry Pi anmelden
- Den RS485 USB-Dongle jetzt am Raspberry Pi einstecken
- In der Konsole folgende Anweisung eingeben: `dmesg`
- Es werden viele Informationen angezeigt, schauen Sie an das Textende und suchen Sie nach einer Textzeile, die in etwa wie folgt lautet:
 - `FTDI USB Serial Device converter now attached to ttyUSB0`
- Der RS485 USB-Dongle aus unserem Shop meldet sich als FTDI USB Serial Device an. Bei einem anderen Dongle kann der Name anderes lauten. Am Ende der Zeile steht, mit welchem Gerätenamen der RS485 USB-Dongle am Raspberry Pi angemeldet wurde. Im Beispiel lautet der Name `ttyUSB0`, anstatt der Null kann eine andere Zahl stehen. Notieren Sie sich diesen Gerätenamen.
- HINWEIS: Die Nummer hinter dem Gerätenamen `ttyUSB` muss zwischen „0“ und „98“ liegen, sonst kann der RS485 USB-Dongle in CODESYS nicht verwendet werden.
- Im nächsten Schritt muss die CODESYS Benutzerkonfigurationsdatei angepasst werden, hier teilen wir CODESYS mit, welche serielle Schnittstelle wir verwenden:
 - 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/ttyUSB
portnum := COM.SysCom.SYS_COMPORT1
```
 - entfernen Sie die beiden Semikolons, ändern Sie den Gerätenamen von `ttyS` auf `ttyUSB`, 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`

Greifen Sie nun unter CODESYS V3.5 auf die neue serielle Schnittstelle zu. Beachten Sie, dass CODESYS beim COM-Port mit der Nummerierung bei eins beginnt und nicht bei null. Ist Ihr RS485 USB-Dongle mit dem Gerätenamen `ttyUSB0` im Raspbian Betriebssystem angemeldet, verwenden Sie in CODESYS den COM-Port eins. Ist der Dongle mit dem Gerätenamen `ttyUSB3` registriert, würden Sie unter CODESYS den COM-Port vier einstellen usw.

Im PiXtend V2 Software Handbuch, Kapitel 7, Unterkapitel Vorbereitung Linux, Unterkapitel Anpassung der Schnittstelleneinstellungen, finden Sie hierzu weitere Informationen.

6.10. PiXtend V2 -L – Serielle Schnittstelle prüfen

Um mit den PiXtend eIO Geräte uneingeschränkt zu arbeiten, empfehlen wir die Verwendung der vollwertigen seriellen Schnittstelle des Raspberry Pi, auch PL011 oder serial0 genannt, mit dem 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 2.1.1.1L für das PiXtend V2 -L- CODESYS Image, verwenden die ttyAMA0 Schnittstelle zur Kommunikation.

Verwenden Sie ein älteres Image bzw. eine ältere Raspbian Version oder Sie sind sich nicht sicher, ob Sie schon ein neues Image nutzen, dann schauen Sie am besten in die config.txt Datei in der Raspbian Bootpartition.

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

Verwenden Sie folgenden Befehl, um es am Raspberry Pi direkt zu prüfen:

→ `sudo nano /boot/config.txt`

Am Ende der Datei finden Sie den Eintrag `dtoverlay=pi3-disable-bt`. Ist er vorhanden, dann verwenden Sie bereits das Gerät ttyAMA0 als serielle Schnittstelle und müssen nichts weiter tun.

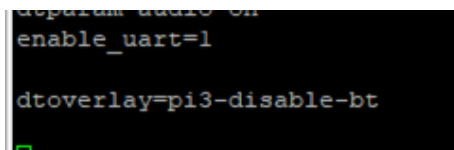


Abbildung 1: RPi - Bluetooth ausschalten

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

- `sudo nano /boot/config.txt`
- Gehen Sie ans Dateiende
- 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`

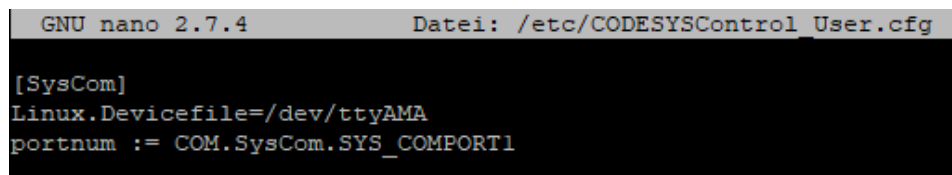
Um mit CODESYS V3.5 die PiXtend eIO Geräte anzusteuern, ist nach der vorherigen Änderung noch eine weitere Anpassung notwendig, um die richtige serielle Schnittstelle zu verwenden.

- Folgenden Befehl ausführen: `sudo nano /etc/CODESYSControl_User.cfg`
- In der Datei CODESYSControl_User.cfg den Abschnitt [SysCom] auswählen
- 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
```
- Entfernen Sie die beiden Semikolons, 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 2: CODESYSControl_User.cfg - Serieller Anschluss ttyAMA

Denken Sie bitte daran auf der serial0 bzw. ttyAMA0 Schnittstelle die sog. Login-Shell zu deaktivieren. Siehe Kapitel 6.8 Serielle Schnittstelle vorbereiten für eigenes SD Karten Abbild.

6.11. PiXtend eIO Watchdog-Timer

6.11.1. Watchdog-Timer Verhalten

Wird der Watchdog-Timer auf einem Gerät aktiviert und läuft ab, so geht das Gerät in den sicheren Zustand. In der Voreinstellung behalten alle Ausgänge ihren aktuellen Zustand bei, es wird nicht eingegriffen. Zusätzlich zeigt die ERR-LED einen Blinkcode an.

Der Anwender kann folgende Funktionen des Watchdog-Timers beeinflussen:

- Watchdog-Timer Timeoutzeit (16 ms bis 8 Sek.)
 - Modbus RTU Protokoll Voreinstellung: Holding Register, abhängig siehe Tabelle 18: Watchdog-Timer Timeoutzeit Übersicht
 - PiXtend eIO Protokoll Voreinstellung: 4 Sekunden Timeout
- Aktives Abschalten aller Ausgänge und setzen auf 0V/0mA
- Nur Digital One Gerät:

Setzen des Digitalausgangs 7, wenn der Watchdog-Timer abläuft

 - Voreinstellung: Digitalausgang 7 wird von LOW auf HIGH gesetzt, im Normalzustand ist der Ausgang immer LOW.
 - Alternativ: Digitalausgang 7 wird von HIGH auf LOW gesetzt, also invertiert verwendet, im Normalzustand ist der Ausgang immer HIGH.
 - Hat der Anwender diese Option des Watchdog-Timers aktiviert, dann steht der Digitalausgang 7 nicht mehr für eigene Anwendungen zur Verfügung, sondern wird exklusiv vom Watchdog-Timer verwendet.

6.11.2. Watchdog-Timer Modi

Jedes PiXtend eIO Gerät verfügt über einen Watchdog-Timer, der dazu genutzt werden kann die Kommunikation am Bus und dem Gerät selbst zu überwachen.

Der Watchdog-Timer hat grundsätzlich 2 Modi.

Im ersten Modus überwacht der Watchdog-Timer „nur“ die Kommunikation am Bus. Das heißt, werden Daten vom Master gesendet, antwortet ein anderes Gerät (Slave) am Bus und findet ein Datenaustausch statt. Nimmt der Anwender keine Änderung an der Watchdog-Timer Konfiguration vor, ist dieser Modus voreingestellt.

Im zweiten Modus überwacht der Watchdog-Timer die tatsächliche Kommunikation mit dem PiXtend eIO Gerät. Der Modbus RTU Master kommuniziert innerhalb einer vorgegebenen Zeit mindestens einmal mit dem Gerät und verschickt zum Beispiel eine Nachricht, die die Input Register liest oder Ausgänge setzt. Nur wenn die Nachricht direkt für das Gerät bestimmt ist, wird der Watchdog-Timer zurückgesetzt und die Wartezeit des Watchdog-Timers beginnt von neuem. Nun sendet der Master innerhalb der Wartezeit die nächste Nachricht an das Gerät, nach jeder empfangenen Nachricht fängt der Kreislauf von vorne an.

Der Modus des Watchdog-Timers ist hilfreich, um festzustellen, ob der Modbus RTU Master schnell genug ist, um in der vorgegebenen Zeitspanne fortlaufend mit dem PiXtend eIO Gerät zu kommunizieren. Bei großen Modbus Netzwerken sind bis zu 32 Geräte an einem Modbus RTU Bus erlaubt. Der Master spricht viele Geräte an und es könnte passieren, dass das PiXtend eIO Gerät nicht mehr oder nur sehr selten angesprochen wird. Die maximal mögliche Wartezeit für den Watchdog-Timer beträgt 8 Sekunden.

6.11.3. Watchdog-Timer Ausgang

Hinweis: Dieser Abschnitt gilt nur für das PiXtend eIO Digital One Gerät

Der Watchdog-Timer kann entsprechend konfiguriert werden, so dass er den Digitalausgang 7 setzt, sobald dieser abläuft. Dies geschieht unabhängig davon ob der Anwender möchte, dass der Watchdog-Timer alle Ausgänge abschaltet. Der Ausgang 7 würde dennoch aktiv bleiben, auch wenn die restlichen 7 Ausgänge inaktiv werden. Mit dieser Funktion lässt sich beispielsweise eine Signallampe oder Hupe einschalten.

Zusätzlich kann dieses „Watchdog-Timer ist abgelaufen Signal“ invertiert werden. Im Normalbetrieb ist der Digitalausgang 7 immer HIGH. Läuft der Watchdog-Timer ab und löst aus, so wechselt das Signal auf LOW. Denkbar wäre hier, dass man diesen Umstand nutzt um ein anderes Gerät, im Normalbetrieb, ein Bereitsignal zu geben. Im Problemfall geht dieses Signal automatisch aus.

NOTICE

Bei Aktivierung dieser Option, steht der digitale Ausgang 7 dem Anwender nicht mehr als steuerbarer Ausgang zur Verfügung. Dieser Ausgang wird jetzt vom Watchdog-Timer kontrolliert.

CAUTION

Wird die Versorgung getrennt, so kann der digitale Ausgang 7 kein Signal schalten. Der Anschluss eines Aktors ist denkbar. Achten Sie darauf, dass dieser keine sicherheitsrelevante Einrichtung darstellt oder direkt erreicht werden kann

6.11.4. Watchdog-Timer - Ausgänge abschalten

Die Voreinstellung des Watchdog-Timers ist, wenn er abläuft, dass alle Ausgänge ihren aktuellen Zustand beibehalten, es wird nicht eingegriffen. Dies gilt für jedes Gerät, für alle analogen und digitalen Ausgänge. Der Anwender hat die Möglichkeit über die Watchdog-Timer Konfiguration, den Watchdog-Timer anzuweisen alle Ausgänge aktiv abzuschalten. Das heißt beim PiXtend eIO Digital One Gerät werden alle digitalen Ausgänge auf LOW gesetzt bzw. ausgeschaltet und beim PiXtend eIO Analog One werden alle Ausgänge auf 0V und 0mA eingestellt.

6.12. Zähler Funktion im PiXtend eIO Digital One

Das Digital One Gerät verfügt insgesamt über 8 Zählereinheiten, jeder Digitaleingang stellt eine Zählereinheit dar. Jeder der 8 Zähler kann unterschiedlich eingestellt werden. Es handelt sich hierbei um sogenannte 16 Bit Zähler, die einen Zahlenbereich von 0 bis 65535 umfassen. Beim Erreichen des höchsten oder niedrigsten Wertes, abhängig von der Zählrichtung (hoch oder runter), geschieht automatisch ein Überlauf (engl.: Overflow), der betreffende Zähler zählt weiter. Die Zähler haben eine Einstellungsmöglichkeit, über die der Anwender festlegen kann welche Flanken eines Signals gezählt werden.

6.12.1. Standardfunktion der Zähler

Jeder der 8 Zähler im PiXtend eIO Digital One bietet verschiedene Einstellungen, die als Standardfunktion bezeichnet werden.

Zu diesen Standardfunktionen gehört die Einstellung welche Flanke bzw. Flanken eines digitalen Eingangssignals gezählt werden sollen. Die Zähler können steigende und fallende Flanken zählen oder beides. Werden beide Flanken gezählt, ist der Zählwert später doppelt so groß als wenn nur eine Flanke gezählt wird. Die Zähler zählen jede Flanke einzeln.

Zusätzlich lässt sich für jeden Zähler festgelegt, ob dieser hoch zählt, der aktuelle Wert wird um 1 erhöht, oder ob der Zähler runter zählt, der aktuelle Wert wird um 1 verringert. Die Einstellung der Zählrichtung bietet noch eine dritte Option, den 2 Kanal Zähler. Diese spezielle Funktion wird im nächsten Kapitel beschrieben.

Jeder Zähler kann einzeln oder alle Zähler können gleichzeitig zurückgesetzt werden. Jeder Zähler kann mit einem Wert vorgeladen bzw. vorbelegt werden, wenn eine Zählung beispielsweise nicht bei null beginnen soll.

6.12.2. Funktion der 2 Kanal Zähler

Das PiXtend eIO Digital One Gerät verfügt über insgesamt 8 Zählereinheiten, 4 Zähler können als sogenannte 2 Kanal Zähler (2 Sensorbetrieb genannt) genutzt werden. In diesem speziellen Modus werden für die Zähler 0, 2, 4 und 6 die jeweils dazu passenden digitalen Eingänge zusammengefasst.

Zählereinheit	Erster Eingang	Zweiter Eingang	Bemerkung
Zähler 0	DI0	DI1	Nur SF/FF ³ , Zähler 1 deaktiviert
Zähler 2	DI2	DI3	Nur SF/FF, Zähler 3 deaktiviert
Zähler 4	DI4	DI5	Nur SF/FF, Zähler 5 deaktiviert
Zähler 6	DI6	DI7	Nur SF/FF, Zähler 7 deaktiviert

Tabelle 5: Übersicht Digital One – 2 Kanal Zähler Funktion, Eingänge

In diesem Modus können Sie für eine Zählereinheit 2 Sensoren, Schalter oder Taster anschließen und somit durch die Reihenfolge der Signale bestimmen ob der Zähler hoch oder runter zählt. Beim PiXtend eIO Digital One Gerät wird hochgezählt. Wird ein Signal am ersten Eingang, vor dem Signal am zweiten Eingang erkannt, ist die Situation umgedreht, es wird runtergezählt.

NOTICE

Bitte beachten Sie, dass beide digitalen Eingänge nicht gleichzeitig ihren Pegel ändern, die Pegeländerung treten nacheinander auf. Der Zeitabstand zwischen beiden Pegeländerungen muss mind. 2,5 ms betragen bzw. um diese versetzt sein.

Die Übersicht verdeutlicht, welche Signalreihenfolge der digitalen Eingänge welche Auswirkung hat.

- | | | |
|-------------------|-------------------|-----------------------|
| - Erster Eingang | → Zweiter Eingang | → Zähler zählt hoch |
| - Zweiter Eingang | → Erste Eingang | → Zähler zählt runter |

Die 2 Kanal Zähler sind von Ihrer Beschaffenheit so ausgelegt, dass man prinzipiell einen Quadratur Encoder anschließen kann, der über eine A und B Spur verfügt. Achten Sie darauf, dass vor einer Drehrichtungsänderung immer eine vollständige Umdrehung erfolgt, sonst besteht die Möglichkeit, dass eine Umdrehung bzw. Zählung verloren geht.

NOTICE

Die Zählereingänge sind auf eine maximale Frequenz von 400 Hz ausgelegt. Sind die Signale schneller, werden sie nicht mehr sicher erfasst und gezählt.

Beachten Sie, dass im 2 Kanal Zähler Modus die Zählereinheit des jeweils zweiten Digitaleingangs nicht mehr separat genutzt werden kann. Dies betrifft die Zähler 1, 3, 5 und 7, sofern die anderen 4 Zählereinheiten als 2 Kanal Zähler konfiguriert sind. Der Anwender kann unabhängig von der eingestellten Zählerkonfiguration jederzeit den aktuellen Zustand der 8 digitalen Eingänge auslesen.

³SF = Steigende Flanke, FF = Fallende Flanke

6.13. Hyper-Logic IO's

Die digitale Hyper-Logic ist ein innovatives Feature der PiXtend eIO Digital One Geräte. Mit einfachen logischen Verknüpfungen zwischen Ein- und Ausgängen, die der Anwender im Gerät festlegen kann, können die Geräte Digitalausgänge schalten, ohne dass eine Datenübertragung über den Bus notwendig ist.

Jedes Gerät verfügt über zwei Hyper-Logic Prozessoren (logische Verarbeitungseinheiten, abgekürzt HL) die jeweils einen Digitalausgang ansteuern können. Die Steuerung von jedem Ausgang basiert auf einer logischen Verknüpfung von bis zu vier Digitaleingängen. Der Anwender hat eine Auswahl an verschiedenen UND/ODER Verknüpfungen zwischen den verschiedenen Digitaleingängen, um festzulegen bei welchem Zustand der Digitalausgang geschaltet werden soll. Zusätzlich lässt sich für jeden Digitaleingang festlegen, ob dieser negiert (invertiert) werden soll. Dadurch ist es Möglichkeit negativ schaltende Logiken aufzubauen und zu verwenden.

Dem Hyper-Logic Prozessor 0, die erste Hyper-Logic Einheit, ist der Digitalausgang 0 sowie die Digitaleingänge 0 bis 3 zugeordnet.

Dem Hyper-Logic Prozessor 1, die zweite Hyper-Logic Einheit, ist der Digitalausgang 4 sowie die Digitaleingänge 4 bis 7 zugeordnet.

Durch die Aktivierung der Hyper-Logic Funktion kann der Anwender den jeweiligen Digitalausgang, der einer Hyper-Logic Einheit zugeordnet ist, nicht mehr selbst steuern. Die Zustände der Digitaleingänge lassen sich ohne Einschränkung über den Bus abfragen, unabhängig von der Einstellung des Hyper-Logic Prozessors.

Wurde eine Hyper-Logic Einheit aktiviert, läuft diese auf dem PiXtend eIO Gerät autark weiter. Eine fortlaufende Kommunikation ist nicht erforderlich damit die Hyper-Logic ihre Arbeit verrichtet.

6.14. Gerätekonfiguration per Modbus RTU in CODESYS V3.5

Die PiXtend eIO Geräte bieten verschiedene Einstellungs- und Konfigurationsmöglichkeiten. Für die meisten Einstellungen ist es völlig ausreichend, wenn diese einmalig beim Start des Hauptprogramms an das PiXtend eIO Gerät übertragen werden. Ein zyklisches Übertragen der Konfigurationswerte ist nicht notwendig und man kann auf diese Weise im Regelbetrieb die Buslast verringern, da weniger Kommunikation mit den Geräten erforderlich ist.

Unter CODESYS V3.5 bietet es sich deshalb an, die Übertragung der einzelnen Holding Register, die Einstellungsinformationen enthalten, nur bei einer steigenden Flanke einer Triggervariable zu übertragen.

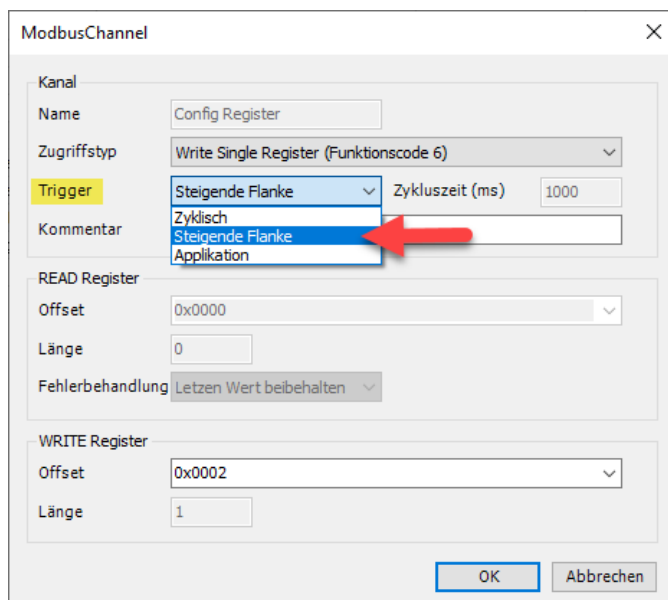


Abbildung 3: CODESYS Modbus, Übertragung per Triggervariable (steigende Flanke) ausführen

Wird im laufenden Betrieb nachträglich eine Einstellung geändert, kann diese in CODESYS durchgeführt werden. Durch das erneute Setzen der Triggervariablen für das gewünschte Holding Register, wird der Inhalt neu übertragen. Die Übertragung erfolgt nur bei Bedarf und nicht zyklisch.

Ob eine Hauptfunktion, wie beispielsweise der Watchdog-Timer aktiv ist, lässt sich über das Auslesen des Status Registers anzeigen, das immer zyklisch ausgelesen werden sollte.

6.15. Anzahl unterstützter Geräte

Das Modbus RTU Protokoll und das PiXtend eIO Protokoll sind für die Verwendung von bis zu 32 Geräten ausgelegt.

Es gilt zu beachten, dass bei Modbus RTU die Geräte-Adresse im Bereich von 1 bis 247 liegen muss und beim PiXtend eIO Protokoll hingegen zwischen 0 und 255 liegen kann.

NOTICE

Jede Geräte-Adresse darf an einem seriellen RS485 Bussystem nur einmal vorkommen. Bei einer doppelten Vergabe kommt es zu Fehlfunktionen und Datenverlust.

6.16. Mehrzweckausgänge-Digitalausgang 0, 4 und 7

Hinweis: Dieser Abschnitt betrifft nur das Digital One Gerät.

Das Digital One Gerät verfügt über verschiedene Zusatzfunktionen, die bei ihrer Aktivierung die Kontrolle bzw. Hoheit über einen der genannten Digitalausgänge übernehmen. Der Anwender hat dann keinen Einfluss mehr auf den entsprechenden Ausgang, d.h. per Software (Modbus RTU oder PiXtend eIO Protokoll) ist dieser nicht mehr steuerbar.

Bitte entnehmen Sie der nachstehenden Auflistung die Information, welche Funktion bei ihrer Aktivierung welchen Digitalausgang steuert.

- Hyper-Logic Prozessor 0 → Digitalausgang 0
- Hyper-Logic Prozessor 1 → Digitalausgang 4
- Watchdog-Timer, Watchdog-Timer Ausgang Funktion → Digitalausgang 7

6.17. Strom und Spannung-Umrechnungsfaktoren

Beim Analog One Gerät erhält der Anwender die Rohwerte des Gerätes. Diese Werte müssen in Strom- bzw. Spannungswerte umgerechnet werden. Nachfolgend finden Sie die entsprechenden Formeln.

- Stromeingänge: $\text{Rohwert}_{\text{in}} * 0,020158400229358 = \text{mA}_{\text{in}}$
- Spannungseingänge 10 Volt-Bereich: $(\text{Rohwert}_{\text{in}} * 10)/1023 = \text{Volt}_{\text{in}}$
- Spannungseingänge 5 Volt-Bereich: $(\text{Rohwert}_{\text{in}} * 5)/1023 = \text{Volt}_{\text{in}}$

Um die analogen Ausgänge zu setzen ist ebenfalls eine Umrechnung notwendig, dieses Mal in anderer Richtung. Die gewünschten Strom- und Spannungswerte müssen in Rohwerte überführt werden. Nachfolgend sind die passenden Formeln dazu aufgeführt.

- Stromausgänge: $\text{mA}_{\text{out}} * (4095/20) = \text{Rohwert}_{\text{out}}$
- Spannungsausgänge: $\text{V}_{\text{out}} * (4095/10) = \text{Rohwert}_{\text{out}}$

6.18. Zwei Bytes zu einem 16 Bit Wert kombinieren

Arbeitet man mit dem PiXtend eIO Protokoll, lassen sich 16 Bit Werte, Zahlen von 0 bis 65535, nicht auf einmal übertragen, sie müssen über 2 separate Bytes abgerufen werden. Diese zwei Bytes muss der Anwender nach Erhalt wieder zu einem 16 Bit Wert kombinieren. Nachfolgend sind Code-Beispiele aufgeführt.

6.18.1. CODESYS V3.5

- `result_16bit_value := SHL(BYTE_TO_WORD(data_high_byte),8) OR
BYTE_TO_WORD(data_low_byte);`

6.18.2. Python

- `result_16bit_value = (data_high_byte << 8) + data_low_byte`

6.18.3. C

- `result_16bit_value = (uint16_t)(data_high_byte<<8) | (data_low_byte);`

6.19. Einen 16 Bit Wert in zwei Bytes zerlegen

Arbeitet man mit dem PiXtend eIO Protokoll, lassen sich 16 Bit Werte, Zahlen von 0 bis 65535, nicht auf einmal übertragen, sie müssen über 2 separate Bytes übertragen werden. Diese zwei Bytes muss der Anwender aus dem 16 Bit Wert extrahieren. Nachfolgend sind Code-Beispiele aufgeführt.

6.19.1. CODESYS V3.5

- `result_8bit_low_byte := WORD_TO_BYTE(data_16bit_value AND 16#FF);`
- `result_8bit_high_byte := WORD_TO_BYTE(SHR(data_16bit_value, 8) AND
16#FF);`

6.19.2. Python

- `result_8bit_low_byte = data_16bit_value & 0xFF`
- `result_8bit_high_byte = (data_16bit_value >> 8) & 0xFF`

6.19.3. C

- `result_8bit_low_byte = (uint8_t)(data_16bit_value & 0xFF);`
- `result_8bit_high_byte = (uint8_t)((data_16bit_value>>8) & 0xFF);`

6.20. Fehlersignalisierung bei PiXtend eIO Geräten

Alle PiXtend eIO Geräte sind für einen zyklischen Betrieb ausgelegt und können über sporadische Einzelbefehle genutzt und gesteuert werden.

Wird eine Nachricht an ein Gerät geschickt, werden die Informationen ausgewertet und das Gerät gibt eine Rückantwort. Wird ein Fehler oder ein Problem festgestellt, die Kommunikation arbeitet fehlerfrei und die ERR-LED liefert eine Signalisierung, so erhält man einen Fehlercode.

Die Rückmeldung der ERR-LED bezieht sich immer auf eine Nachricht. Wird bei der nächsten Nachricht kein Fehler festgestellt und im Gerät selbst ist alles in Ordnung, so geht die ERR-LED wieder aus. Liegt kein fortlaufender Fehler vor, wird die Anzeige des letzten Fehlers automatisch quittiert.

Dieses Verhalten kann im zyklischen Betrieb dazu führen, dass die ERR-LED eingeschaltet und gleich wieder ausgeschaltet wird, es stellt sich ein Blinken der ERR-LED ein. Dieses Verhalten kann daran liegen, dass dem Gerät unterschiedliche Nachrichten geschickt werden und unter diesen Nachrichten ist eine oder sind mehrere Nachrichten, die auf einen Fehler hinweisen.

Wenn Sie feststellen, dass die ERR-LED immer wieder an und ausgeht, das Blinken kann unregelmäßig sein, dann stoppen Sie die Kommunikation mit dem Gerät. Prüfen Sie die DIP-Schalter, ob diese in der richtigen Position stehen. Führen Sie am Gerät einen Power Cycle aus und testen Sie im Anschluss alle Modbus RTU Funktion Codes oder PiXtend eIO Protokoll Befehle einzeln durch, um festzustellen welcher fehlerhaft ist.

Lesen Sie in einer solchen Situation das Fehlerregister (Error Register) aus, um festzustellen, welche Fehler aufgetreten sind. Es werden alle aufgetretenen Fehler festgehalten, bis der Anwender diese quittiert oder ein Power Cycle durchführt.

7. Geräte-LEDs – Signalisierung



Abbildung 4: Status LEDs "ERR", "COM" & "+5V" auf PiXtend eIO Geräten

7.1. Signalisierung: Versorgungsspannung LED „+5V“

Die grüne LED mit der Beschriftung „+5V“ signalisiert das Vorhandensein der internen Versorgungsspannung von 5V DC.

Zustand	Benennung	Bedeutung
Aus/schwaches Leuchten	nicht betriebsbereit	Versorgungsspannung nicht vorhanden und unzureichend
An	Betriebsbereit	Versorgungsspannung vorhanden

Sollte diese LED nicht leuchten oder nur leicht glimmen, dann ist die Versorgung des Geräts nicht vorhanden oder unzureichend. Prüfen Sie in diesem Fall die Verdrahtung, die Spannung zwischen den Klemmen „VCC“ und „GND“ und kontrollieren Sie die Anforderungen der Spannungsversorgung. Genauere Informationen finden Sie im Hardware Handbuch.

7.2. Signalisierung: Kommunikation LED „COM“

Die orangene LED mit der Beschriftung „COM“ signalisiert Datenübertragungen auf dem RS485-Bus. Es ist unerheblich, ob die übertragenen Daten für das jeweilige Gerät bestimmt sind oder nicht. Jede Kommunikation auf dem Bus führt zu einem Blinken. Wenn keine Datenübertragung stattfindet, bleibt diese LED dunkel.

Zustand	Benennung	Bedeutung
Aus	Ruhezustand	
Blinken	Kommunikation	Auf dem RS485-Bus findet Datenübertragung/ Kommunikation statt

An der Blink-Frequenz kann eine Tendenz abgelesen werden (je höher die Frequenz, umso mehr Daten bzw. umso schneller werden Daten auf dem Bus übertragen). Die genaue Baudrate oder Datenmenge kann durch diese LED nicht abgelesen werden.

7.3. Signalisierung: Fehler LED „ERR“

Manche Fehlerzustände die bei PiXtend eIO auftreten, lassen sich unter Umständen nicht per RS485 an den Bus-Master zurückmelden, zum Beispiel bei fehlerhafter Kommunikation oder wenn der Watchdog-Timer des PiXtend eIO abgelaufen ist. Für diese Situationen wurde auf den PiXtend eIO Geräten eine rote LED „ERR“ vorgesehen.

Diese LED kann zur Problemerkennung herangezogen werden, die Bedeutung der verschiedenen Zustände sind nachfolgend aufgeführt:

Zustand	Benennung	Bedeutung
Aus	Betriebsbereit, kein Fehler	-
An (dauerhaft)	Telegrammfehler	Modbus RTU Protokoll: Falscher Funktion Code (FC) Anzahl der Register, Colis oder Discrets ist falsch
	Nachrichtenfehler	PiXtend eIO ASCII Protokoll: Frame Error, Parity Error, Overrun Error, falsche Baudrate eingestellt
Blinken schnell (0,05 s)	Kommunikationsfehler Konfigurationsfehler	Bei Modbus RTU & PiXtend eIO ASCII Protokoll: Fehler bei der Datenübertragung Konfiguration per DIP-Schalter unzulässig oder während des Betriebs verändert
Blinken mittel (0,1 s)	Kommando-Fehler	Nur bei PiXtend eIO ASCII Protokoll: das gesendete Kommando ist unbekannt oder wird nicht unterstützt
Blinken langsam (0,2 s)	I/O-Fehler	Nur bei PiXtend eIO ASCII Protokoll: der gewünschte Ein- oder Ausgang ist nicht vorhanden (z.B. beim Versuch Ausgang 9 zu setzen, es gibt aber nur 8 Ausgänge)
Blinken 3x schnell, 3x langsam, 1 s Pause	Watchdog-Timer	Bei Modbus RTU & PiXtend eIO ASCII Protokoll: Watchdog-Timer des Geräts hat ausgelöst

Tabelle 6: LED „ERR“ - Signalisierung von Fehlerzuständen

Das einmalige rote Blinken beim Start der Geräte ist normal und weist nicht auf einen Fehler hin. Ebenso blinkt die orange LED „COM“ beim Start kurz auf.

8. PiXtend eIO - Gerätekonfiguration

8.1. Überblick

Alle PiXtend eIO Geräte verfügen über 2 DIP⁴ Schalter Blöcke mit jeweils 8 Schaltern. Der erste DIP Block ist mit ADDRESS gekennzeichnet, über diese Schalter wird die Geräte-Adresse eingestellt. Über die 8 Schalter können Bus-Adressen im Bereich von 1 - 247 (Modbus RTU) bzw. 0 - 255 (PiXtend eIO Protokoll) eingestellt werden.

Jede Geräte-Adresse darf am Bus nur einmal verwendet werden!

Die Angabe der Geräte-Adresse geschieht wie erwähnt über 8 Schalter, wobei die Anordnung der Schalter von links nach rechts geht, Zahlen 1 bis 8 (siehe Bild).

Ist ein Schalter (weiß) am oberen Rand eines DIP Blocks, so befindet sich dieser in der Stellung ON und ist somit aktiv. Eine vollständige Auflistung aller möglichen Geräte-Adressen finden Sie in Kapitel 8.2 Geräte-Adressen.

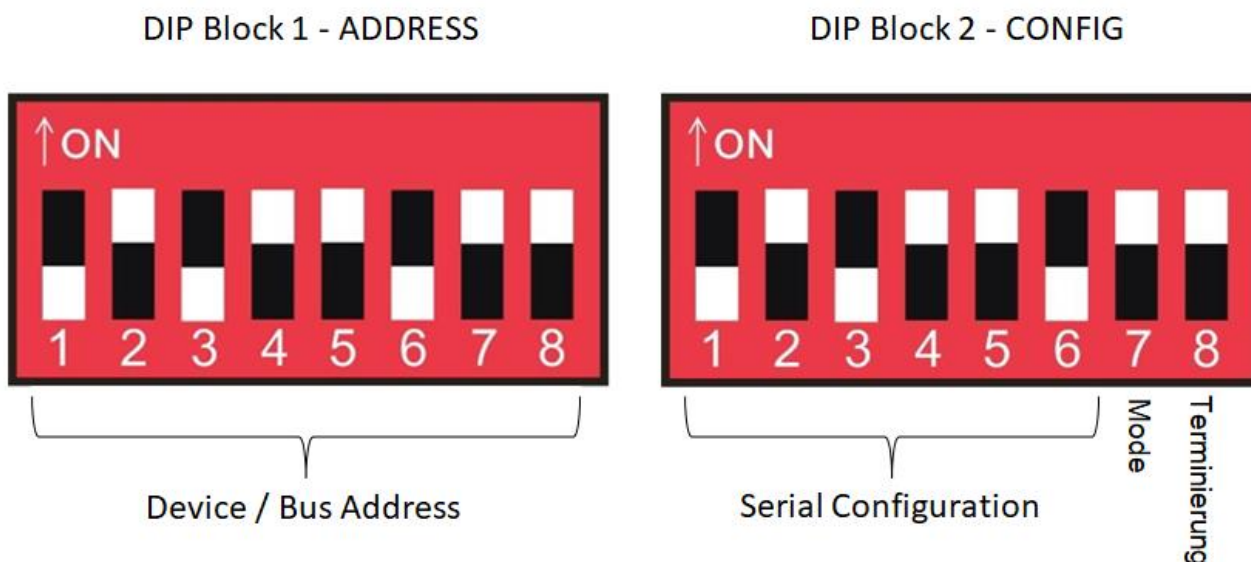


Abbildung 5: Gerätekonfiguration - Übersicht DIP Schalter - Schema

Der zweite DIP Block ist mit CONFIG gekennzeichnet und dient dazu, die serielle Konfiguration einzustellen, das Kommunikationsprotokoll (Mode) auszuwählen und die Bustermiierung ein- oder auszuschalten.

Die Konfiguration der seriellen Schnittstelle erfolgt über die ersten 6 Schalter (Zahlen 1 bis 6). Über den Schalter 7 erfolgt die Auswahl des Kommunikationsprotokolls. In der Stellung OFF, wenn der Schalter unten steht, ist das Modbus RTU Protokoll ausgewählt bzw. aktiv und wird vom Gerät zur Kommunikation verwendet. In der Stellung ON verwendet das Gerät das Kontron Electronics GmbH eigene PiXtend eIO Protokoll zum Datenaustausch mit anderen Geräten.

Über den Schalter 8 kann die Bustermiierung eingeschaltet werden, dazu muss der Schalter in die Stellung ON nach oben geschoben werden. Befindet er sich unten ist die Bustermiierung ausgeschaltet.

Für weitere Informationen zur Kommunikationseinstellung siehe Kapitel 8.3 Serielle Konfiguration, 8.4 Protokollauswahl und 8.5 Bustermiierung.

⁴DIP ist die Abkürzung für Dual In-line Package

8.2. Geräte-Adresse

Die Geräte-Adresse ermöglicht dem Bus-Master immer genau ein Gerät am Bus anzusprechen. Jede Nachricht, die der Bus-Master versendet enthält eine eindeutige Geräte-Adresse. Somit ist immer sichergestellt, welches Gerät angesprochen ist und dass nie mehrere Geräte gleichzeitig antworten.

DIP Block 1 - ADDRESS

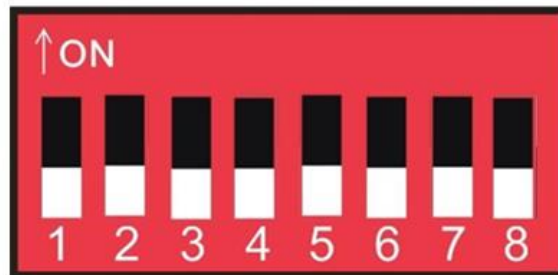


Abbildung 6: DIP Block 1 - ADDRESS - Geräte-Adresse

In der nachfolgenden Tabelle sind alle 255 möglichen Geräte-Adressen aufgeführt, zusammen mit der jeweils dazugehörigen Schalterstellung am DIP Block 1 für alle 8 Schalter. Eine 0 bedeutet Stellung OFF, der Schalter steht in der unteren Position und eine 1 bedeutet Stellung ON, der Schalter muss in der oberen Position stehen. Die Reihenfolge der Schalter ist von links nach rechts. Beachten Sie die Nummerierung, diese ist von 1 = Schalter 1 bis 8 = Schalter 8.

Geräte-Adresse	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0
5	1	0	1	0	0	0	0	0
6	0	1	1	0	0	0	0	0
7	1	1	1	0	0	0	0	0
8	0	0	0	1	0	0	0	0
9	1	0	0	1	0	0	0	0
10	0	1	0	1	0	0	0	0
11	1	1	0	1	0	0	0	0
12	0	0	1	1	0	0	0	0
13	1	0	1	1	0	0	0	0
14	0	1	1	1	0	0	0	0
15	1	1	1	1	0	0	0	0
16	0	0	0	0	1	0	0	0
17	1	0	0	0	1	0	0	0
18	0	1	0	0	1	0	0	0
19	1	1	0	0	1	0	0	0

Geräte-Adresse	1	2	3	4	5	6	7	8
20	0	0	1	0	1	0	0	0
21	1	0	1	0	1	0	0	0
22	0	1	1	0	1	0	0	0
23	1	1	1	0	1	0	0	0
24	0	0	0	1	1	0	0	0
25	1	0	0	1	1	0	0	0
26	0	1	0	1	1	0	0	0
27	1	1	0	1	1	0	0	0
28	0	0	1	1	1	0	0	0
29	1	0	1	1	1	0	0	0
30	0	1	1	1	1	0	0	0
31	1	1	1	1	1	0	0	0
32	0	0	0	0	0	1	0	0
33	1	0	0	0	0	1	0	0
34	0	1	0	0	0	1	0	0
35	1	1	0	0	0	1	0	0
36	0	0	1	0	0	1	0	0
37	1	0	1	0	0	1	0	0
38	0	1	1	0	0	1	0	0
39	1	1	1	0	0	1	0	0
40	0	0	0	1	0	1	0	0
41	1	0	0	1	0	1	0	0
42	0	1	0	1	0	1	0	0
43	1	1	0	1	0	1	0	0
44	0	0	1	1	0	1	0	0
45	1	0	1	1	0	1	0	0
46	0	1	1	1	0	1	0	0
47	1	1	1	1	0	1	0	0
48	0	0	0	0	1	1	0	0
49	1	0	0	0	1	1	0	0
50	0	1	0	0	1	1	0	0
51	1	1	0	0	1	1	0	0
52	0	0	1	0	1	1	0	0
53	1	0	1	0	1	1	0	0
54	0	1	1	0	1	1	0	0
55	1	1	1	0	1	1	0	0
56	0	0	0	1	1	1	0	0
57	1	0	0	1	1	1	0	0
58	0	1	0	1	1	1	0	0
59	1	1	0	1	1	1	0	0
60	0	0	1	1	1	1	0	0
61	1	0	1	1	1	1	0	0

Geräte-Adresse	1	2	3	4	5	6	7	8
62	0	1	1	1	1	1	0	0
63	1	1	1	1	1	1	0	0
64	0	0	0	0	0	0	1	0
65	1	0	0	0	0	0	1	0
66	0	1	0	0	0	0	1	0
67	1	1	0	0	0	0	1	0
68	0	0	1	0	0	0	1	0
69	1	0	1	0	0	0	1	0
70	0	1	1	0	0	0	1	0
71	1	1	1	0	0	0	1	0
72	0	0	0	1	0	0	1	0
73	1	0	0	1	0	0	1	0
74	0	1	0	1	0	0	1	0
75	1	1	0	1	0	0	1	0
76	0	0	1	1	0	0	1	0
77	1	0	1	1	0	0	1	0
78	0	1	1	1	0	0	1	0
79	1	1	1	1	0	0	1	0
80	0	0	0	0	1	0	1	0
81	1	0	0	0	1	0	1	0
82	0	1	0	0	1	0	1	0
83	1	1	0	0	1	0	1	0
84	0	0	1	0	1	0	1	0
85	1	0	1	0	1	0	1	0
86	0	1	1	0	1	0	1	0
87	1	1	1	0	1	0	1	0
88	0	0	0	1	1	0	1	0
89	1	0	0	1	1	0	1	0
90	0	1	0	1	1	0	1	0
91	1	1	0	1	1	0	1	0
92	0	0	1	1	1	0	1	0
93	1	0	1	1	1	0	1	0
94	0	1	1	1	1	0	1	0
95	1	1	1	1	1	0	1	0
96	0	0	0	0	0	1	1	0
97	1	0	0	0	0	1	1	0
98	0	1	0	0	0	1	1	0
99	1	1	0	0	0	1	1	0
100	0	0	1	0	0	1	1	0
101	1	0	1	0	0	1	1	0
102	0	1	1	0	0	1	1	0
103	1	1	1	0	0	1	1	0

Geräte-Adresse	1	2	3	4	5	6	7	8
104	0	0	0	1	0	1	1	0
105	1	0	0	1	0	1	1	0
106	0	1	0	1	0	1	1	0
107	1	1	0	1	0	1	1	0
108	0	0	1	1	0	1	1	0
109	1	0	1	1	0	1	1	0
110	0	1	1	1	0	1	1	0
111	1	1	1	1	0	1	1	0
112	0	0	0	0	1	1	1	0
113	1	0	0	0	1	1	1	0
114	0	1	0	0	1	1	1	0
115	1	1	0	0	1	1	1	0
116	0	0	1	0	1	1	1	0
117	1	0	1	0	1	1	1	0
118	0	1	1	0	1	1	1	0
119	1	1	1	0	1	1	1	0
120	0	0	0	1	1	1	1	0
121	1	0	0	1	1	1	1	0
122	0	1	0	1	1	1	1	0
123	1	1	0	1	1	1	1	0
124	0	0	1	1	1	1	1	0
125	1	0	1	1	1	1	1	0
126	0	1	1	1	1	1	1	0
127	1	1	1	1	1	1	1	0
128	0	0	0	0	0	0	0	1
129	1	0	0	0	0	0	0	1
130	0	1	0	0	0	0	0	1
131	1	1	0	0	0	0	0	1
132	0	0	1	0	0	0	0	1
133	1	0	1	0	0	0	0	1
134	0	1	1	0	0	0	0	1
135	1	1	1	0	0	0	0	1
136	0	0	0	1	0	0	0	1
137	1	0	0	1	0	0	0	1
138	0	1	0	1	0	0	0	1
139	1	1	0	1	0	0	0	1
140	0	0	1	1	0	0	0	1
141	1	0	1	1	0	0	0	1
142	0	1	1	1	0	0	0	1
143	1	1	1	1	0	0	0	1
144	0	0	0	0	1	0	0	1
145	1	0	0	0	1	0	0	1

Geräte-Adresse	1	2	3	4	5	6	7	8
146	0	1	0	0	1	0	0	1
147	1	1	0	0	1	0	0	1
148	0	0	1	0	1	0	0	1
149	1	0	1	0	1	0	0	1
150	0	1	1	0	1	0	0	1
151	1	1	1	0	1	0	0	1
152	0	0	0	1	1	0	0	1
153	1	0	0	1	1	0	0	1
154	0	1	0	1	1	0	0	1
155	1	1	0	1	1	0	0	1
156	0	0	1	1	1	0	0	1
157	1	0	1	1	1	0	0	1
158	0	1	1	1	1	0	0	1
159	1	1	1	1	1	0	0	1
160	0	0	0	0	0	1	0	1
161	1	0	0	0	0	1	0	1
162	0	1	0	0	0	1	0	1
163	1	1	0	0	0	1	0	1
164	0	0	1	0	0	1	0	1
165	1	0	1	0	0	1	0	1
166	0	1	1	0	0	1	0	1
167	1	1	1	0	0	1	0	1
168	0	0	0	1	0	1	0	1
169	1	0	0	1	0	1	0	1
170	0	1	0	1	0	1	0	1
171	1	1	0	1	0	1	0	1
172	0	0	1	1	0	1	0	1
173	1	0	1	1	0	1	0	1
174	0	1	1	1	0	1	0	1
175	1	1	1	1	0	1	0	1
176	0	0	0	0	1	1	0	1
177	1	0	0	0	1	1	0	1
178	0	1	0	0	1	1	0	1
179	1	1	0	0	1	1	0	1
180	0	0	1	0	1	1	0	1
181	1	0	1	0	1	1	0	1
182	0	1	1	0	1	1	0	1
183	1	1	1	0	1	1	0	1
184	0	0	0	1	1	1	0	1
185	1	0	0	1	1	1	0	1
186	0	1	0	1	1	1	0	1
187	1	1	0	1	1	1	0	1

Geräte-Adresse	1	2	3	4	5	6	7	8
188	0	0	1	1	1	1	0	1
189	1	0	1	1	1	1	0	1
190	0	1	1	1	1	1	0	1
191	1	1	1	1	1	1	0	1
192	0	0	0	0	0	0	1	1
193	1	0	0	0	0	0	1	1
194	0	1	0	0	0	0	1	1
195	1	1	0	0	0	0	1	1
196	0	0	1	0	0	0	1	1
197	1	0	1	0	0	0	1	1
198	0	1	1	0	0	0	1	1
199	1	1	1	0	0	0	1	1
200	0	0	0	1	0	0	1	1
201	1	0	0	1	0	0	1	1
202	0	1	0	1	0	0	1	1
203	1	1	0	1	0	0	1	1
204	0	0	1	1	0	0	1	1
205	1	0	1	1	0	0	1	1
206	0	1	1	1	0	0	1	1
207	1	1	1	1	0	0	1	1
208	0	0	0	0	1	0	1	1
209	1	0	0	0	1	0	1	1
210	0	1	0	0	1	0	1	1
211	1	1	0	0	1	0	1	1
212	0	0	1	0	1	0	1	1
213	1	0	1	0	1	0	1	1
214	0	1	1	0	1	0	1	1
215	1	1	1	0	1	0	1	1
216	0	0	0	1	1	0	1	1
217	1	0	0	1	1	0	1	1
218	0	1	0	1	1	0	1	1
219	1	1	0	1	1	0	1	1
220	0	0	1	1	1	0	1	1
221	1	0	1	1	1	0	1	1
222	0	1	1	1	1	0	1	1
223	1	1	1	1	1	0	1	1
224	0	0	0	0	0	1	1	1
225	1	0	0	0	0	1	1	1
226	0	1	0	0	0	1	1	1
227	1	1	0	0	0	1	1	1
228	0	0	1	0	0	1	1	1
229	1	0	1	0	0	1	1	1

Geräte-Adresse	1	2	3	4	5	6	7	8
230	0	1	1	0	0	1	1	1
231	1	1	1	0	0	1	1	1
232	0	0	0	1	0	1	1	1
233	1	0	0	1	0	1	1	1
234	0	1	0	1	0	1	1	1
235	1	1	0	1	0	1	1	1
236	0	0	1	1	0	1	1	1
237	1	0	1	1	0	1	1	1
238	0	1	1	1	0	1	1	1
239	1	1	1	1	0	1	1	1
240	0	0	0	0	1	1	1	1
241	1	0	0	0	1	1	1	1
242	0	1	0	0	1	1	1	1
243	1	1	0	0	1	1	1	1
244	0	0	1	0	1	1	1	1
245	1	0	1	0	1	1	1	1
246	0	1	1	0	1	1	1	1
247	1	1	1	0	1	1	1	1
248	0	0	0	1	1	1	1	1
249	1	0	0	1	1	1	1	1
250	0	1	0	1	1	1	1	1
251	1	1	0	1	1	1	1	1
252	0	0	1	1	1	1	1	1
253	1	0	1	1	1	1	1	1
254	0	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1

Tabelle 7: PiXtend eIO: Auswahltable - Geräte-Adresse

8.3. Serielle Konfiguration

Damit das Gerät in möglichst viele RS485 Netze integriert werden kann, gibt es auf dem DIP Block 2 insgesamt 6 Schalter (Zahlen 1 - 6) über die die serielle Schnittstelle des Gerätes konfiguriert werden kann. Zur Auswahl stehen verschiedene Kombinationen aus Baudrate, Parität und Stoppbit. Insgesamt gibt es 45 Einstellmöglichkeiten, wobei jeweils immer nur eine über die 6 Schalter eingestellt werden kann.

Eine 0 bedeutet Stellung OFF, der Schalter steht in der unteren Position und eine 1 bedeutet Stellung ON, der Schalter muss in der oberen Position stehen. Die Reihenfolge der Schalter geht von links nach rechts, Zahlen 1 - 6.

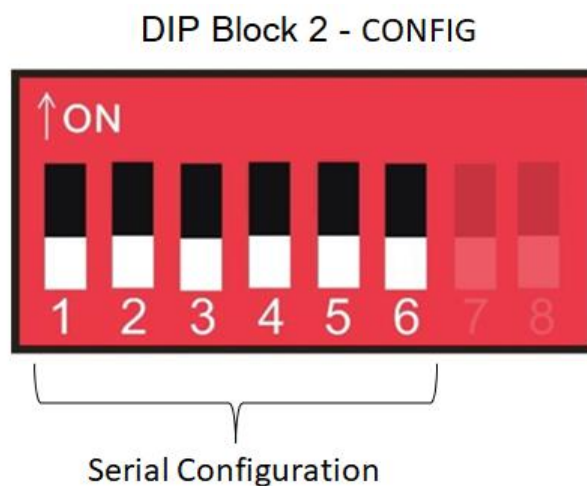


Abbildung 7: DIP Block 2 - CONFIG - Serielle Konfiguration

Wählen Sie aus der nachfolgenden Tabelle die gewünschte Kombination aus Baudrate, Parität und Stoppbit(s) und übertragen Sie dann die entsprechende Schalterreihenfolge der Tabelle auf die 6 ersten Schalter von DIP Block 2 von links nach rechts entsprechend der Zuordnung 1 = Schalter 1 bis 6 = Schalter 6. Die Schalter 7 und 8 können hier außer Acht gelassen werden, diese werden in den folgenden Kapiteln behandelt.

Nummer	Baudrate	Parität	Stoppbits	1	2	3	4	5	6
0	19200	Even	1	0	0	0	0	0	0
1	2400	Keine	1	1	0	0	0	0	0
2	4800	Keine	1	0	1	0	0	0	0
3	9600	Keine	1	1	1	0	0	0	0
4	14400	Keine	1	0	0	1	0	0	0
5	19200	Keine	1	1	0	1	0	0	0
6	28800	Keine	1	0	1	1	0	0	0
7	38400	Keine	1	1	1	1	0	0	0
8	57600	Keine	1	0	0	0	1	0	0
9	76800	Keine	1	1	0	0	1	0	0
10	115200	Keine	1	0	1	0	1	0	0
11	230400	Keine	1	1	1	0	1	0	0
12	2400	Keine	2	0	0	1	1	0	0
13	4800	Keine	2	1	0	1	1	0	0
14	9600	Keine	2	0	1	1	1	0	0

Nummer	Baudrate	Parität	Stoppbits	1	2	3	4	5	6
15	14400	Keine	2	1	1	1	1	0	0
16	19200	Keine	2	0	0	0	0	1	0
17	28800	Keine	2	1	0	0	0	1	0
18	38400	Keine	2	0	1	0	0	1	0
19	57600	Keine	2	1	1	0	0	1	0
20	76800	Keine	2	0	0	1	0	1	0
21	115200	Keine	2	1	0	1	0	1	0
22	230400	Keine	2	0	1	1	0	1	0
23	2400	Even	1	1	1	1	0	1	0
24	4800	Even	1	0	0	0	1	1	0
25	9600	Even	1	1	0	0	1	1	0
26	14400	Even	1	0	1	0	1	1	0
27	19200	Even	1	1	1	0	1	1	0
28	28800	Even	1	0	0	1	1	1	0
29	38400	Even	1	1	0	1	1	1	0
30	57600	Even	1	0	1	1	1	1	0
31	76800	Even	1	1	1	1	1	1	0
32	115200	Even	1	0	0	0	0	0	1
33	230400	Even	1	1	0	0	0	0	1
34	2400	Odd	1	0	1	0	0	0	1
35	4800	Odd	1	1	1	0	0	0	1
36	9600	Odd	1	0	0	1	0	0	1
37	14400	Odd	1	1	0	1	0	0	1
38	19200	Odd	1	0	1	1	0	0	1
39	28800	Odd	1	1	1	1	0	0	1
40	38400	Odd	1	0	0	0	1	0	1
41	57600	Odd	1	1	0	0	1	0	1
42	76800	Odd	1	0	1	0	1	0	1
43	115200	Odd	1	1	1	0	1	0	1
44	230400	Odd	1	0	0	1	1	0	1

Tabelle 8: PiXtend eIO: Auswahltable - Serielle Gerätekonfiguration

8.4. Protokollauswahl

Mit Hilfe des Schalters 7 im DIP Block 2 - CONFIG kann das Kommunikationsprotokoll für das PiXtend eIO Gerät ausgewählt werden. Zur Auswahl stehen das bekannte Modbus RTU Protokoll und das Kontron Electronics GmbH eigene PiXtend eIO Protokoll.

In der Schalterstellung OFF, wenn sich der Schalter in der unteren Position befindet, ist das Modbus RTU Protokoll (Werkseinstellung) ausgewählt und aktiv.

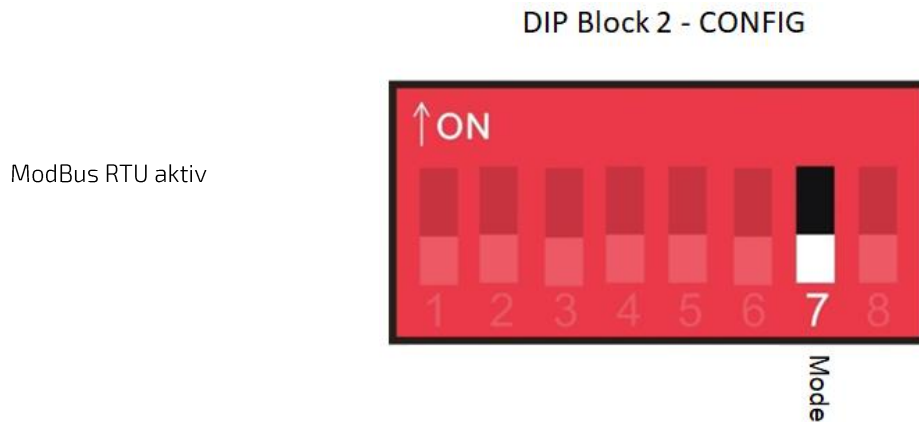


Abbildung 8: DIP Block 2 - CONFIG - Mode - Modbus RTU

In der Schalterstellung ON, wenn sich der Schalter in der oberen Position befindet, ist das PiXtend eIO Protokoll ausgewählt und wird anstelle des Modbus RTU Protokolls für die Kommunikation verwendet.

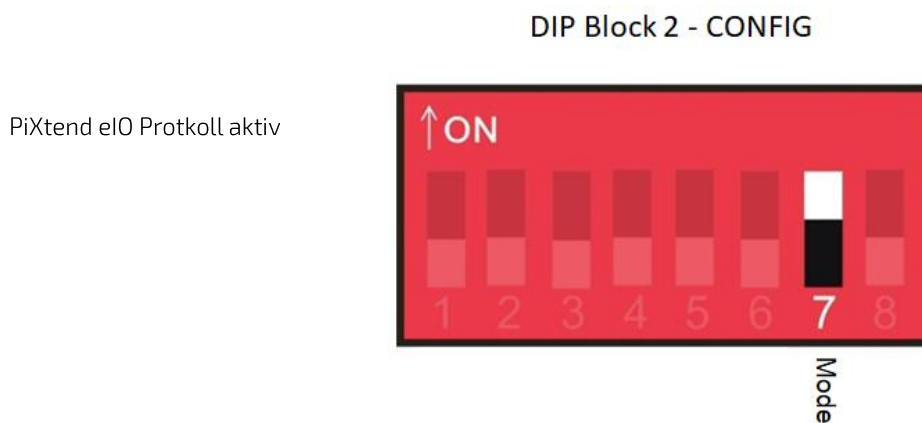


Abbildung 9: DIP Block 2 - CONFIG - Mode - PiXtend eIO Protokoll

8.5. Busterminierung

Mit Hilfe des Schalters 8 im DIP Block 2 - CONFIG kann die Busterminierung für das PiXtend eIO Gerät ein- bzw. ausgeschaltet werden. Die Auswahl hängt davon ab, ob das Gerät das erste oder letzte Gerät am Bus ist. In der Schalterstellung OFF, wenn sich der Schalter in der unteren Position befindet, ist die Busterminierung aus (Werkseinstellung).

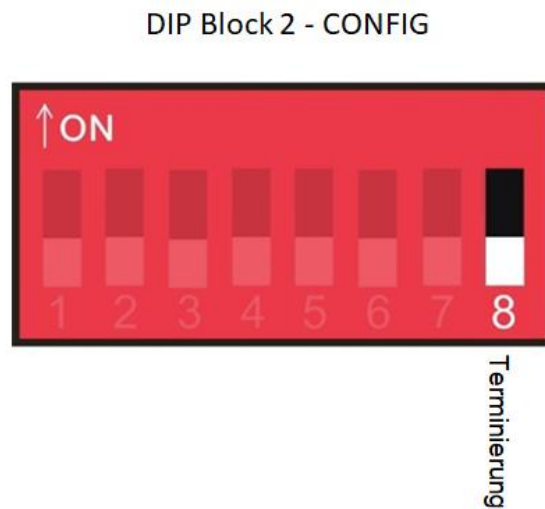


Abbildung 10: DIP Block 2 - CONFIG - Busterminierung aus

In der Schalterstellung ON, wenn sich der Schalter in der oberen Position befindet, ist die Busterminierung aktiv bzw. eingeschaltet. Schalten Sie die Terminierung nur für das erste und das letzte Gerät am Bus ein.

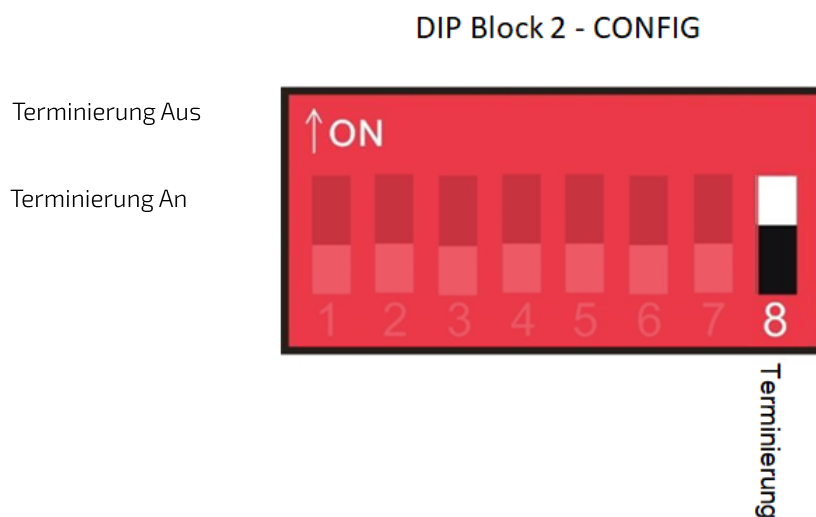


Abbildung 11: DIP Block 2 - CONFIG - Busterminierung an

9. Modbus RTU - Protokoll

9.1. Einführung

In diesem Kapitel finden Sie Informationen zu den Modbus Adressen, Registern und Funktionen die von dem jeweiligen PiXtend eIO Gerät unterstützt werden. Durch diese Angaben kann der Anwender digitale und analoge Ausgänge schalten und digitale und analoge Eingänge einlesen. Zudem können über bestimmte Register und Konfigurationseinstellungen weitere Funktionen im Gerät aktiviert werden.

9.2. Übersicht

Allgemeines - für alle eIO-Geräte

- Unterstützte Funktion Codes
- Modbus RTU Fehlernummern

PiXtend eIO Digital One

- Modbus Adressen und Funktionen
- Beispiel CODESYS
- Beispiel Python
- Beispiel C

PiXtend eIO Analog One

- Modbus Adressen und Funktionen
- Beispiel CODESYS
- Beispiel Python
- Beispiel C

9.3. Unterstützte Funktion Codes

Die nachfolgenden Funktion Codes werden von den PiXtend eIO Geräten unterstützt.

- FC02 – Lesen Discrete Inputs
- FC01|FC05|FC15 – Lesen/Schreiben der Coils
 - Nur Digital One Gerät
- FC04 – Lesen der Input Register
- FC03|FC06|FC16 – Lesen/Schreiben der Holding Register

9.3.1. PiXtend eIO Digital One – Modbus Adressen und Funktionen

9.3.1.1 Discrete Inputs-Einlesen der digitalen Eingänge

Die digitalen Eingänge können einzeln oder alle gelesen werden. In CODESYS wird jeder Eingang als Bit in einem Byte zur Verfügung gestellt, andere Softwareplattformen und Modbus RTU Implementierungen können davon abweichen. Bitte lesen Sie dies in der jeweiligen Dokumentation zu Ihrem System nach.

Verwenden Sie hier den Modbus RTU Funktion Code 2 zum Einlesen der Werte.

Bezeichnung	Adresse	Datentyp	Bemerkung
Digital Input 0...7	0...7	Bit	Jeder digitale Eingang wird als einzelnes Bit mit eigener Discrete Input Adresse bereitgestellt

Tabelle 9: Digital One: Digitale Eingänge als Discrete Inputs einlesen

9.3.1.2 Coils (Outputs)-Setzen der digitalen Ausgänge

Die digitalen Ausgänge können einzeln oder alle gleichzeitig gesetzt bzw. geschrieben werden. In CODESYS wird jeder Ausgang als Bit in einem Byte zur Verfügung gestellt, andere Softwareplattformen und Modbus RTU Implementierungen können davon abweichen. Bitte lesen Sie dies in der jeweiligen Dokumentation zu Ihrem System nach.

Verwenden Sie hier die Modbus RTU Funktion Codes 1, 5 und 15 zum Lesen bzw. Schreiben der Werte.

Bezeichnung	Adresse	Datentyp	Bemerkung
Digital Output 0...7	0...7	Bit	Jeder digitale Ausgang wird als einzelnes Bit mit eigener Coil Adresse bereitgestellt

Tabelle 10: Digital One: Digitale Ausgänge als Coils lesen/schreiben

9.3.1.3 Input Register-Einlesen verschiedener Informationen und Zustände

Die Input Register ermöglichen das Einlesen verschiedener Informationen vom PiXtend eIO Gerät. Alle verfügbaren Modbus Input Register sind in der nachfolgenden Tabelle aufgeführt. Verweise zu Aufschlüsselungen der verschiedenen Bits, die in einem Register vorhanden sein können, steht jeweils in der Spalte Bemerkung. Verwenden Sie den Funktion Code 4 um die Input Register des Gerätes auszulesen.

Die Adressen der Input Register beginnen bei der Zählung bei Adresse 0. Insgesamt hat das Gerät 12 Adressen bzw. 12 Input Register. Der Datentyp jedes Registers ist 16 Bit vom Typ WORD, 2 Bytes lang und kann die Zahlenwerte von 0 bis 65535 enthalten.

Bezeichnung	Adresse	Bemerkung
Digital Inputs	0	Dies ist eine Alternative zum Einlesen der digitalen Eingänge, wenn man keine Discrete Inputs verwenden möchte. Nur die ersten (unteren) 8 Bits im WORD werden vom Gerät verwendet.
Status Register	1	Dieses Register enthält Status Informationen vom Gerät als Rückmeldung an den Anwender, damit geprüft werden kann ob eine Funktion eingeschaltet ist oder nicht. So kann festgestellt werden, ob der Watchdog-Timer aktiv ist. Die Tabelle im Kapitel 9.3.1.5 Bits im Status Register enthält eine Aufschlüsselung der Statusbits.
Fehler Register	2	Enthält ein Abbild der im Gerät festgestellten Fehler, die nachträglich nicht direkt per ERR-LED abgelesen werden können, da der fehlerhafte Zustand nicht mehr vorhanden ist. Die Tabelle im Kapitel 9.3.1.6 Bits im Fehler Register enthält eine Aufschlüsselung der Fehlerbits.
Zähler 0 Register	3	Aktueller Wert des Zählers 0, Wertebereich 0... 65535
Zähler 1 Register	4	Aktueller Wert des Zählers 1, Wertebereich 0... 65535
Zähler 2 Register	5	Aktueller Wert des Zählers 2, Wertebereich 0... 65535
Zähler 3 Register	6	Aktueller Wert des Zählers 3, Wertebereich 0... 65535
Zähler 4 Register	7	Aktueller Wert des Zählers 4, Wertebereich 0... 65535
Zähler 5 Register	8	Aktueller Wert des Zählers 5, Wertebereich 0... 65535
Zähler 6 Register	9	Aktueller Wert des Zählers 6, Wertebereich 0... 65535
Zähler 7 Register	10	Aktueller Wert des Zählers 7, Wertebereich 0... 65535
Version Register	11	Version der Firmware in Zahlen, 101 = 1.01, 102 = 1.02 etc...

Tabelle 11: Digital One: Input Register

9.3.1.4 Holding Register - Setzen der Ausgänge und Konfigurationsbits

Die Holding Register ermöglichen das Setzen der einzelnen Ausgänge und die Einstellung verschiedener Konfigurationen, wie das Aktivieren der Zähler oder des Watchdog-Timers. Alle verfügbaren Modbus Holding Register sind in der nachfolgenden Tabelle aufgeführt. Verweise zu Aufschlüsselungen der verschiedenen Bits, die in einem Register vorhanden sein können, stehen jeweils in der Spalte Bemerkung. Verwenden Sie die Funktion Codes 6 und 16 um die Holding Register des Gerätes zu beschreiben.

Die Adressen der Holding Register beginnen bei der Zählung bei Adresse 0. Insgesamt hat das Gerät 7 Adressen bzw. 7 Holding Register. Der Datentyp jedes Registers ist 16 Bit vom Typ WORD, 2 Bytes lang und kann die Zahlenwerte von 0 bis 65535 enthalten.

Bezeichnung	Adresse	Bemerkung
Digital Outputs	0	Wurde im Config Register das entsprechende Bit gesetzt, können die Ausgänge des Digital One Geräts mittels der ersten 8 Bits in diesem Register gesetzt werden. Dies ist eine Alternative gegenüber der Verwendung und der Steuerung über Coils
Watchdog-Timer Register	1	Das Watchdog-Timer Register erlaubt das Einstellen und Einschalten des Gerätes Watchdog-Timer zur Überwachung der Bus-Kommunikation. Die Tabelle im Kapitel 9.3.1.7 Bits im Watchdog-Timer Register enthält eine Aufschlüsselung der Konfigurationsbits.
Config Register	2	Im Config Register können verschiedene Einstellungen im Gerät vorgenommen werden der interne Fehlerspeicher kann gelöscht werden, die Hyper-Logic lässt sich aktivieren, das Zurücksetzen der Zähler wird durchgeführt und es lässt sich festlegen ob man die Ausgänge anstatt per Coils über ein Register setzen will. Die Tabelle im Kapitel 9.3.1.8 Bits im Config Register enthält eine Aufschlüsselung der Konfigurationsbits.
Counter Config Register 0	3	Dieses Register ermöglicht das Einschalten eines Zählers (0 - 7) und die Festlegung der Flankenerkennung. Es können steigenden Flanken, fallende Flanken oder beide Flanken gezählt werden. Die Tabelle im Kapitel 9.3.1.9 Bits im Counter Config Register enthält eine

Bezeichnung	Adresse	Bemerkung
		Aufschlüsselung der Konfigurationsbits.
Counter Config Register 1	4	In diesem Register können für jeden Zähler eine Zählfunktion bzw. ein Typ festgelegt werden. Es gibt die Optionen hoch- und runterzählen, mit jeweils 2 Eingängen. Der Hochzähler (Up-Counter) ist voreingestellt. Die Tabelle im Kapitel 9.3.1.10 Bits im Counter Config Register enthält eine Aufschlüsselung der Konfigurationsbits.
Counter Pre-Set Value Register	5	Zähler 0 - 7 mit einem Wert vorladen, z.B. zum Runterzählen auf 0 muss in Verbindung mit den Reset-Bits im Config Register geschehen. Siehe dazu auch Kapitel 9.3.1.8 Bits im Config Register.
Hyper-Logic Config Register	6	Über dieses Register kann die Hyper-Logic im Digital One Gerät konfiguriert werden. Die Tabelle im Kapitel 9.3.1.11 Bits im Hyper-Logic Config Register enthält eine Aufschlüsselung der Konfigurationsbits.

Tabelle 12: Digital One: Holding Register

NOTICE

Werden Zähler 0, 2, 4, 6 im 2 Sensorbetrieb betrieben, können die anderen Zähler nicht verwendet werden. Das Gerät verhindert dies intern und schaltet die betroffenen Zähler automatisch ab.

NOTICE

Die Zähler 1, 3, 5, 7 lassen sich nicht in den „2 Sensorbetrieb“ schalten. Das Gerät ignoriert diese Konfiguration und deaktiviert die betroffenen Zähler automatisch.

9.3.1.5 Bits im Status Register

Das Status Register zeigt an, welche Funktion im Gerät aktiv ist und welche nicht. Ein zyklisches Abfragen wird empfohlen, zusätzlich kann auf diesem Weg geprüft werden ob eine Funktion aktiviert wurde oder nicht.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	CNT1	CNT0	HL1	HL0	WDT OBE	WDT OUT	WDT RST	WDT ON
Startwert	0	0	0	0	0	0	0	0

Tabelle 13: Bits im Status Register - Unteres Byte (Low Byte)

Bit 0 - WDT ON → Watchdog-Timer aktiv

Aus: Der Watchdog-Timer ist aus.

An: Der Watchdog-Timer ist aktiv und wird je nach Einstellung zurückgesetzt bzw. löst nach Ablauf der vorgegebenen Zeit aus.

Bit 1 - WDT RST → Watchdog-Timer Reset-Typ Einstellung

Aus: Der Watchdog-Timer wird immer bei Bus-Aktivität zurückgesetzt. Findet keine Kommunikation auf dem Bus statt, da der Master nicht mehr sendet, dann löst der Watchdog-Timer nach Ablauf des Timeout's aus.

An: Der Watchdog-Timer wird zurückgesetzt, wenn das Gerät innerhalb der eingestellten Timeoutzeit eine vollständige Modbus RTU Nachricht vom Master erhält. Dies ist sinnvoll, wenn viele Modbus RTU Geräte mit einem Bus in Verbindung stehen und man feststellen möchte, ob der Modbus RTU Master das Gerät innerhalb der Timeoutzeit anspricht. Das Timeout kann bis maximal 8 Sekunden eingestellt werden.

Bit 2 - WDT OUT → Watchdog-Timer Ausgang aktiv (Digitalausgang 7)

Aus: Wenn der Watchdog-Timer abläuft wird der Digitalausgang 7 nicht gesetzt, er wird als normaler digitaler Ausgang behandelt.

An: Der Digitalausgang 7 wird vom Watchdog-Timer verwendet und steht dem Anwender nicht mehr zur Verfügung. In der Voreinstellung setzt der Watchdog-Timer, wenn er abläuft, den Digitalausgang 7 auf HIGH. Dies kann vom Anwender geändert werden. Siehe Bit 3 - WDT OBE.

Bit 3 - WDT OBE → Watchdog-Timer Ausgang Verhalten

Aus: Der Digitalausgang 7 wird von LOW → HIGH geschaltet, wenn der Watchdog-Timer abläuft. Der Normalzustand für diese Einstellung ist LOW. Diese Einstellung entspricht dem Standardverhalten (Voreinstellung). Diese Einstellung ist hilfreich, wenn beispielsweise eine Lampe oder Sirene geschaltet werden soll. Läuft der Watchdog-Timer ab so lässt sich schneller ein Fehlzustand der Anlage erkennen.

An: Der Digitalausgang 7 wird von HIGH → LOW geschaltet, wenn der Watchdog-Timer abläuft. Der Normalzustand für den Digitalausgang 7 ist HIGH. Diese Einstellung ist praktisch, wenn im Betrieb ein Relais angezogen sein muss, damit die Anlage betriebsbereit ist. Läuft der Watchdog-Timer ab, wird der Digitalausgang 7 auf LOW geschaltet und das Relais geht aus.

Bit 4 - HLO → Hyper-Logic Prozessor 0

Aus: Es findet keine Hyper-Logic Verarbeitung für Hyper-Logic Prozessor 0 statt.

An: Die Hyper-Logic Verarbeitung ist eingeschaltet und der Zustand des Digitalausgangs 0 kann über bis zu vier Digitaleingänge gesteuert werden. Es stehen verschiedene Kombinationsmöglichkeiten der Digitaleingänge 0 bis 3 zur Verfügung, siehe weitere Informationen im Abschnitt 9.3.1.8 Bits im Config Register

Bit 5 - HL1 → Hyper-Logic Prozessor 1

Aus: Es findet keine Hyper-Logic Verarbeitung für Hyper-Logic Prozessor 1 statt.

An: Die Hyper-Logic Verarbeitung ist eingeschaltet und der Zustand des Digitalausgangs 4 kann über bis zu vier Digitaleingänge gesteuert werden. Es stehen verschiedene Kombinationsmöglichkeiten der Digitaleingänge 4 bis 7 zur Verfügung, siehe weitere Informationen im Abschnitt 9.3.1.8 Bits im Config Register

Bit 6 - CNT0 → Zähler 0 aktiv (Counter 0)

Aus: Der Zähler 0 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 0 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 0 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register 0 und 9.3.1.10 Bits im Counter Config Register 1

Bit 7 - CNT1 → Zähler 1 aktiv (Counter 1)

Aus: Der Zähler 1 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 1 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 1 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register 0 und 9.3.1.10 Bits im Counter Config Register 1

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	DOREG ON	CNT7	CNT6	CNT5	CNT4	CNT3	CNT2
Startwert	0	0	0	0	0	0	0	0

Tabelle 14: Bits im Status Register - Oberes Byte (High Byte)

Bit 8 - CNT2 → Zähler 2 aktiv (Counter 2)

Aus: Der Zähler 2 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 2 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 2 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register und 9.3.1.10 Bits im Counter Config Register 1.

Bit 9 - CNT3 → Zähler 3 aktiv (Counter 3)

Aus: Der Zähler 3 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 3 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 3 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register 0 und 9.3.1.10 Bits im Counter Config Register 1.

Bit 10 - CNT4 → Zähler 4 aktiv (Counter 4)

Aus: Der Zähler 4 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 4 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 4 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register 0 und 9.3.1.10 Bits im Counter Config Register 1..

Bit 11 - CNT5 → Zähler 5 aktiv (Counter 5)

Aus: Der Zähler 5 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 5 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 5 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register 0 und 9.3.1.10 Bits im Counter Config Register 1.

Bit 12 - CNT6 → Zähler 6 aktiv (Counter 6)

Aus: Der Zähler 6 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 6 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 6 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register 0 und 9.3.1.10 Bits im Counter Config Register 1.

Bit 13 - CNT7 → Zähler 7 aktiv (Counter 7)

Aus: Der Zähler 7 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 7 ist eingeschaltet. Der 16 Bit Wert des Zählers steht im Input Register Zähler 7 Register. Das Verhalten des Zählers kann über die Holding Register Counter Config Register 0 und Counter Config Register 1 vom Anwender beeinflusst werden. . Weitere Informationen dazu stehen im jeweiligen Abschnitt 9.3.1.9 Bits im Counter Config Register 0 und 9.3.1.10 Bits im Counter Config Register 1.

Bit 14 - DOREG ON → Digitale Ausgänge per Register setzen

Aus: Die Digitalausgänge 0 bis 7 können nur über Coils (FC01, FC05, FC15) gesteuert werden.

An: Die Digitalausgänge 0 bis 7 können nur über das Holding Register Digital Outputs gesteuert werden. Ein Setzen bzw. Verändern der Digitalausgänge über Coils ist bei dieser Einstellung nicht möglich.

9.3.1.6 Bits im Fehler-Register

Das Fehler-Register enthält ein Abbild der im Gerät aufgetretenen Fehler. An Hand der Bits im Fehler-Register lassen sich die aufgetretenen Fehler feststellen. Im laufenden Betrieb bleiben die Bits solange gesetzt, bis sie über das Bit QUIT ERR im Config Register abgelöscht werden oder ein Power-Cycle durchgeführt wird. Ausgenommen von dieser Regel ist das Watchdog-Timer Fehlerbit, weitere Informationen stehen im Erklärungstext zu Bit 4.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	COM ERR	COF ERR	TEL ERR	WDT ERR	HLD OCL ERR	CNTCL ERR	WDTDO7 ERR	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 15: Bits im Fehler Register - Unteres Byte (Low Byte)

Bit 1 - WDTDO7 ERR → Watchdog-Timer - Digitalausgang 7 Konflikt

Wurde der Watchdog-Timer so konfiguriert, dass der Digitalausgang 7 als Signalausgang verwendet wird, dann steht er dem Anwender nicht zur Verfügung. Versucht der Anwender dennoch den Digitalausgang 7 zu setzen, wird dieses Bit auf 1 gesetzt und zeigt eine Fehlbedienung an.

Bit 2 - CNTCL ERR → Zählerkonfiguration Konflikt

Die Digitaleingänge 1, 3, 5 und 7 können nicht als 2 Kanal Zähler verwendet bzw. in den 2 Sensorbetrieb geschaltet werden. Wird dies dennoch versucht, wird dieses Bit gesetzt und man erhält eine Rückmeldung über den unzulässigen Vorgang.

Bit 3 - HLD OCL ERR → Hyper-Logic Ausgang Konflikt

Die Hyper-Logic Prozessoren 0 und 1 verwenden die Digitalausgänge 0 und 4. Die entsprechenden Digitalausgänge stehen dem Anwender nicht zur Verfügung, wenn ein Hyper-Logic Prozessor eingeschaltet wurde. Versucht man dennoch die Digitalausgänge zu setzen, wird dieses Bit als Fehlerrückmeldung gesetzt und zeigt an, dass es einen Konflikt zwischen der Einstellung des Geräts und den verwendeten Digitalausgängen gibt.

Bit 4 - WDT ERR → Der Watchdog-Timer ist abgelaufen

Dieses Bit zeigt an ob vor dem letzten Einschalten der Watchdog-Timer abgelaufen war oder nicht. Da das Gerät bei Ablauf des Watchdog-Timers in den sicheren Zustand geht, kann dieses Bit immer nur nach einem darauffolgenden Power-Cycle abgefragt werden. Das Bit kann über das QUIT ERR Bit im Config Register im laufenden Betrieb abgelöscht werden oder es wird ein weiterer Power-Cycle durchgeführt.

Bit 5 - TEL ERR → Function Code- oder Adressfehler

Wurde dem Gerät ein Telegramm bzw. Nachricht mit einem nicht unterstützten Funktion Code geschickt oder die empfangene Modbus RTU Nachricht enthält eine nicht vorhandene Adresse, dann wird dieses Fehlerbit gesetzt. Mittels dieses Fehlerbits kann im Steuerungsprogramm festgestellt werden, ob dieser Fehler seit dem letzten Einschalten aufgetreten ist.

Bit 6 - COF ERR → Konfigurationsfehler

Dieses Bit wird gesetzt, wenn im Gerät ein Konfigurationsfehler festgestellt wird. Im normalen Betrieb taucht dieser Fehler nicht auf, außer es werden die DIP Schalter während des Betriebs geändert. Um diesen Zustand zu beheben, müssen entweder die DIP-Schalter wieder in ihre ursprüngliche Position geschoben werden oder es erfolgt ein Power Cycle, dadurch wird die neue Konfiguration vom Gerät übernommen.

NOTICE

Die Durchführung eines Power Cycle mit veränderten DIP Schaltern führt zu einer geänderten Geräteeinstellung. Findet die Anpassung nicht im Steuerungsprogramm statt, lässt sich das Gerät weder ansprechen noch verwenden.

Bit 7 - COM ERR → Kommunikationsfehler

Dieses Bit wird gesetzt, wenn das Gerät einen Kommunikationsfehler feststellt. In der Regel handelt es sich hier um drei Fehlerzustände. Das Gerät stellt einen Parity-, Frame- oder Overrun-Fehler fest.

Ein Parity-Fehler tritt meistens dann auf, wenn die Einstellungen am Steuerungsgerät nicht korrekt sind. Gleiches gilt für den Frame-Fehler, hier stimmt in den meisten Fällen die Baudrate nicht und das Gerät empfängt keine gültigen Daten. Der Overrun-Fehler hingegen bedeutet, dass zu schnell zu viele Daten über den Bus geschickt werden und das Gerät nicht alles aufnehmen kann.

Prüfen Sie die Einstellungen am Steuergerät, stimmen diese mit der Konfiguration des Geräts überein, wurde die Verkabelung korrekt durchgeführt und liegt keine Störungen am Bus vor.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 16: Bits im Fehler Register - Oberes Byte (High Byte)

Das obere Byte (High Byte) des Fehler Registers enthält keine Informationen bzw. Fehlerbits und kann ignoriert werden.

9.3.1.7 Bits im Watchdog-Timer Register

Über das Watchdog-Timer Register kann der Anwender Einfluss auf das Verhalten des Geräte Watchdog-Timers nehmen. Die Einstellungsmöglichkeiten reichen von der Timeoutzeit bis dahin, dass der Watchdog-Timer einen Ausgang setzt, wenn er abläuft.

Wird der Watchdog-Timer ohne weitere Konfiguration eingeschaltet, so ist ein Timeout von 16 Millisekunden eingestellt. Das Rücksetzen des Watchdog-Timers erfolgt sobald Aktivität am Bus festgestellt wird, unabhängig davon ob die Daten für das Gerät gedacht sind oder nicht.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	WDT0	WDTRST	WDTEN	WB	WT3	WT2	WT1	WT0
Startwert	0	0	0	0	0	0	0	0

Tabelle 17: Bits im Watchdog-Timer Register - Unteres Byte (Low Byte)

Bit 3...0 - WT3...WT0 → Watchdog-Timer Zeit (Watchdog-Timer Timeout)

Die Watchdog-Timer Zeit legt fest, innerhalb welcher Zeitspanne der Watchdog-Timer auslöst.

Die nachfolgende Tabelle zeigt auf, welches Bitmuster eingestellt werden muss, damit der Watchdog-Timer bei seiner Aktivierung die Timeoutzeit übernimmt und damit arbeitet. Werden keine Bits gesetzt, verwendet der Watchdog-Timer die kürzeste Zeit von 16 ms. Diese Zeitspanne ist für eine Modbus RTU Kommunikation viel zu kurz. Wählen Sie in jedem Fall einen größeren Wert, z.B. 4 Sekunden.

Watchdog-Timer Timeoutzeit:

WT3	WT2	WT1	WT0	Timeoutzeit
0	0	0	0	16 ms
0	0	0	1	32 ms
0	0	1	0	64 ms
0	0	1	1	0.125 s
0	1	0	0	0.25 s
0	1	0	1	0.5 s
0	1	1	0	1.0 s
0	1	1	1	2.0 s
1	0	0	0	4.0 s
1	0	0	1	8.0 s

Tabelle 18: Watchdog-Timer Timeoutzeit Übersicht

Vorgehen zum Setzen/Ändern der Watchdog Zeit im laufenden Betrieb:

- Watchdog ist deaktiviert (ausgeschaltet)
- Gewünschte Watchdog-Timer Timeoutzeit setzen
- Watchdog einschalten

Bit 4 - WB → Watchdog-Timer Verhalten (Watchdog-Timer Behaviour)

Aus: Der Zustand der digitalen Ausgänge bleibt wie er ist, wenn der Watchdog-Timer abläuft.

An: Alle digitalen Ausgänge werden aktiv ausgeschaltet, wenn der Watchdog-Timer abläuft.

Bit 5 - WDTEN → Watchdog-Timer einschalten (Watchdog-Timer enable)

Aus: Der Watchdog-Timer ist aus.

An: Der Watchdog-Timer ist aktiviert bzw. eingeschaltet.

Bit 6 - WDTRST → Watchdog-Timer Reset Verhalten (Watchdog-Timer Reset Type)

Aus: Der Watchdog-Timer Reset wird bei Bus-Aktivität ausgeführt, egal um welche Daten es sich handelt und ob diese für das Gerät bestimmt sind oder nicht. Mit dieser Einstellung kann festgestellt werden, ob der Master grundsätzlich kommuniziert oder ob dieser ausgefallen ist. Erfolgt innerhalb der gewählten Watchdog-Timer Zeit keine Kommunikation am Bus, dann läuft der Watchdog-Timer ab und löst aus.

An: Der Watchdog-Timer Reset wird nur dann ausgeführt, wenn das Gerät eine gültige Modbus RTU Nachricht vom Master erhält. Wird das Gerät vom Master nicht innerhalb der Watchdog-Timer Zeit mit einer gültigen Modbus RTU Nachricht angesprochen, läuft der Watchdog-Timer ab. Diese Einstellung ist hilfreich um festzustellen, dass in einem großen Modbus RTU Verbund jedes Gerät innerhalb der gewählten Watchdog-Timer Zeit angesprochen wird.

Bit 7 - WDTO → Watchdog-Timer Ausgang (Watchdog-Timer Output)

Aus: Der Digitalausgang 7 wird nicht vom Watchdog-Timer verwendet und steht somit dem Anwender zur Verfügung.

An: Der Digitalausgang 7 wird vom Watchdog-Timer als Signalausgang verwendet. In der Voreinstellung setzt der Watchdog-Timer diesen Ausgang auf HIGH, wenn er abläuft.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	WDTOB
Startwert	0	0	0	0	0	0	0	0

Tabelle 19: Bits im Watchdog-Timer Register - Oberes Byte (High Byte)

Bit 8 - WDTOB → Watchdog-Timer Ausgang Verhalten (Watchdog-Timer Output Behaviour)

Aus: Der Digitalausgang 7 wird von LOW auf HIGH gesetzt, wenn der Watchdog-Timer abläuft. Der Normalzustand im Regelbetrieb ist LOW.

An: Der Digitalausgang 7 wird von HIGH auf LOW gesetzt, wenn der Watchdog-Timer abläuft. Der Normalzustand im Regelbetrieb ist HIGH.

Für weitere Informationen zum Watchdog-Timer siehe Kapitel 6.11 PiXtend eIO Watchdog Timer, Abschnitt Basiswissen.

9.3.1.8 Bits im Config Register

Das Config Register ermöglicht das Einstellen und Aktivieren verschiedener Gerätefunktionen. Über die Bits lässt sich der interne Fehlerspeicher im laufenden Betrieb zurücksetzen, die Hyper-Logic Prozessoren können eingestellt und aktiviert werden. Das Zurücksetzen der Zähler ist möglich, sowie die Umschaltung der digitalen Ausgänge von Coils auf Holding Register 0.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	HL1CFG0	HLOCFG3	HLOCFG2	HLOCFG1	HLOCFG0	HL1ACT	HLOACT	QUIT ERR
Startwert	0	0	0	0	0	0	0	0

Tabelle 20: Bits im Config Register - Unteres Byte (Low Byte)

Bit 0 – QUIT ERR → Fehlerspeicher löschen (Quit Errors)

Aus: Das Gerät setzt die Fehlerbits im Fehlerspeicher.

An: Wenn dieses Bit auf 1 gesetzt wird, dann wird der interne Fehlerspeicher des Geräts gelöscht. Das Fehler Register nach dem Setzen dieses Bits prüfen, ob alle Fehler gelöscht wurden.

Bit 1 – HLOACT → Hyper-Logic Prozessor 0 aktiv

Aus: Der Hyper-Logic Prozessor 0 ist ausgeschaltet und arbeitet nicht.

An: Der Hyper-Logic Prozessor 0 ist aktiv und steuert den Digitalausgang 0 entsprechend der gewählten Hyper-Logic Verknüpfung. Siehe Bits 6 bis 3. Der Digitalausgang 0 steht dem Anwender nicht mehr zur Verfügung, er wird exklusiv vom Hyper-Logic Prozessor 0 verwendet. Ob der Hyper-Logic Prozessor 0 aktiv ist, kann im Status Register geprüft werden.

Bit 2 – HL1ACT → Hyper-Logic Prozessor 1 aktiv

Aus: Der Hyper-Logic Prozessor 1 ist ausgeschaltet und arbeitet nicht.

An: Der Hyper-Logic Prozessor 1 ist aktiv und steuert den Digitalausgang 4 entsprechend der gewählten Hyper-Logic Verknüpfung. Siehe Bits 10 bis 7. Der Digitalausgang 4 steht dem Anwender nicht mehr zur Verfügung, er wird exklusiv vom Hyper-Logic Prozessor 1 verwendet. Ob der Hyper-Logic Prozessor 1 aktiv ist, kann im Status Register geprüft werden.

Bits 6 ... 3 - HLOCFG3 ... 0 → Hyper-Logic Prozessor 0 Verknüpfung

Die folgende Tabelle zeigt welche Verknüpfungen der Digitaleingänge 0 bis 3 vom Hyper-Logic Prozessor 0 unterstützt werden. Die Bindungsstärke der logischen Verknüpfungen wird durch Klammern symbolisiert. Ein doppeltes kaufmännisches UND (&&) stellt eine UND-Verknüpfung dar und eine ODER-Verknüpfung wird durch zwei senkrechte Striche (||) ausgedrückt. Die Angabe der Spalten HLOCFG3 .. 0 kann direkt auf diese 4 Bits übertragen werden, so wie die Bits im Register definiert sind. Der Digitalausgang vom Hyper-Logic Prozessor 0 ist immer der Digitalausgang 0.

Num	HLOCFG3	HLOCFG2	HLOCFG1	HLOCFG0	Verknüpfung
0	0	0	0	0	D10
1	0	0	0	1	D10 && D11
2	0	0	1	0	D10 D11
3	0	0	1	1	D10 && D11 && D12
4	0	1	0	0	(D10 && D11) D12
5	0	1	0	1	(D10 D11) && D12
6	0	1	1	0	D10 D11 D12
7	0	1	1	1	D10 && D11 && D12 && D13
8	1	0	0	0	(D10 && D11 && D12) D13
9	1	0	0	1	D10 && (D11 D12) && D13
10	1	0	1	0	D10 D11 && D12 && D13
11	1	0	1	1	(D10 && D11) D12 D13
12	1	1	0	0	(D10 D11) && (D12 D13)
13	1	1	0	1	(D10 D11 D12) && D13
14	1	1	1	0	D10 D11 D12 D13

Tabelle 21: Hyper-Logic Prozessor 0 - Verknüpfungsübersicht

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	DOREG	CRST3	CRST2	CRST1	CRST0	HL1CFG3	HL1CFG2	HL1CFG1
Startwert	0	0	0	0	0	0	0	0

Tabelle 22: Bits im Config Register - Oberes Byte (High Byte)

Bits 10 ... 7 - HL1CFG3 ... 0 → Hyper-Logic Prozessor 1 Verknüpfung

Die folgende Tabelle zeigt welche Verknüpfungen der Digitaleingänge 4 bis 7 vom Hyper-Logic Prozessor 1 unterstützt werden. Die Bindungsstärke der logischen Verknüpfungen wird durch Klammern symbolisiert. Ein doppeltes kaufmännisches UND (&&) stellt eine UND-Verknüpfung dar und eine ODER-Verknüpfung wird durch zwei senkrechte Striche (||) ausgedrückt. Die Angabe der Spalten HL1CFG3 ... 0 kann direkt auf diese 4 Bits übertragen werden, so wie die Bits im Register definiert sind. Der Digitalausgang vom Hyper-Logic Prozessor 1 ist immer der Digitalausgang 4.

Num	HL1CFG3	HL1CFG2	HL1CFG1	HL1CFG0	Verknüpfung
0	0	0	0	0	DI4
1	0	0	0	1	DI4 && DI5
2	0	0	1	0	DI4 DI5
3	0	0	1	1	DI4 && DI5 && DI6
4	0	1	0	0	(DI4 && DI5) DI6
5	0	1	0	1	(DI4 DI5) && DI6
6	0	1	1	0	DI4 DI5 DI6
7	0	1	1	1	DI4 && DI5 && DI6 && DI7
8	1	0	0	0	(DI4 && DI5 && DI6) DI7
9	1	0	0	1	DI4 && (DI5 DI6) && DI7
10	1	0	1	0	DI4 DI5 && DI6 && DI7
11	1	0	1	1	(DI4 && DI5) DI6 DI7
12	1	1	0	0	(DI4 DI5) && (DI6 DI7)
13	1	1	0	1	(DI4 DI5 DI6) && DI7
14	1	1	1	0	DI4 DI5 DI6 DI7

Tabelle 23: Hyper-Logic Prozessor 1 – Verknüpfungsübersicht

Bits 14 ... 11 - CRST3 ... 0 → Zähler zurücksetzen

Über die 4 Zähler-Reset Bits kann jeder Zähler einzeln oder alle Zähler auf einmal zurückgesetzt werden. In der nachfolgenden Tabelle steht sowohl die Dezimalzahl als auch der Binärcode zum Zurücksetzen jeden Zählers. Die Angabe der Spalten CRST3 ... 0 kann direkt auf diese 4 Bits übertragen werden, wobei rechts das LSB steht und links das MSB, entsprechend der Anordnung der Bits im Register.

Dezimal	CRST3	CRST2	CRST1	CRST0	Zähler
0	0	0	0	0	Aus, keine Aktion
1	0	0	0	1	Reset Zähler 0
2	0	0	1	0	Reset Zähler 1
3	0	0	1	1	Reset Zähler 2
4	0	1	0	0	Reset Zähler 3
5	0	1	0	1	Reset Zähler 4
6	0	1	1	0	Reset Zähler 5
7	0	1	1	1	Reset Zähler 6
8	1	0	0	0	Reset Zähler 7
9	1	0	0	1	Reset aller Zähler

Tabelle 24: Config Register - Bits: Zähler zurücksetzen

NOTICE

Beim Reset eines Zählers, wird der Wert des Registers Counter Pre-Set Value Register in das Zählerregister geladen. Ohne Anwendereingriff ist dieser Wert 0.

Der Anwender kann jeden beliebigen Wert zwischen 0 und 65535 in einen Zähler vorladen. Ein Zähler muss nicht bei 0 zu zählen beginnen, nach einem Reset kann er einen anderen Wert annehmen. Dieses Verhalten ist bei einem Down Counter von Vorteil, wenn man immer 100 Einheiten abzählen möchte. Der Zähler kann auf 100 voreingestellt werden, bei 0 angekommen, setzt man mit einem Reset den Zähler zurück auf 100.

Bit 15 - DOREG → Setzen der Digitalausgänge über Holding Register 0

Aus: Die Digitalausgänge können nur über Coils gesetzt werden.

An: Die Digitalausgänge können nur über das Holding Register 0 gesetzt werden, anstelle der Verwendung von Coils.

9.3.1.9 Bits im Counter Config Register 0

Das Counter Config Register 0 erlaubt die Einstellung der Flankenerkennung der Zähler, das heißt auf welche Flanke soll der Zähler reagieren, den Zählwert erhöhen bzw. herabsetzen. Das Setzen der Flankeneinstellung aktiviert zeitgleich den Zähler. Für jeden Zähler können im Counter Config Register 0 zwei Bits über die Flankeneinstellung ausgewählt werden. Jeder Zähler hat vier Zustände. Gemäß der Aufschlüsselung der Bits, wird die Einstellung für einen Zähler gezeigt, der analog für alle Zähler gilt.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	CED31	CED30	CED21	CED20	CED11	CED10	CED01	CED00
Startwert	0	0	0	0	0	0	0	0

Tabelle 25: Bits im Counter Config Register 0 - Unteres Byte (Low Byte)

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	CED71	CED70	CED61	CED60	CED51	CED50	CED41	CED40
Startwert	0	0	0	0	0	0	0	0

Tabelle 26: Bits im Counter Config Register 0 - Oberes Byte (High Byte)

Dezimal	CEDX1	CEDX0	Flankeneinstellung
0	0	0	Aus, Zähler nicht aktiv
1	0	1	Steigende Flanke (Rising Edge), 0 → 1
2	1	0	Fallende Flanke (Falling Edge), 1 → 0
3	1	1	Steigende + Fallende Flanke (RE + FE), 0 → 1 + 1 → 0

Tabelle 27: Counter Config Register 0 - Flankeneinstellung - Aufschlüsselung

Das „x“ im Kürzel CEDX1 bzw. CEDX0 steht stellvertretend für einen der acht Zähler (0 bis 7).

NOTICE

Die Flankeneinstellung Nummer 3 (Binär 11 - Steigende und Fallende Flanke) kann nicht für den 2 Sensorbetrieb (2 Kanal Zähler) verwendet werden. Wird diese Einstellung dennoch gewählt, wertet das Gerät nur die steigenden Flanken aus, die fallenden Flanken werden ignoriert. Das Gerät verhält sich wie bei Flankeneinstellung Nummer 1 (Binär 01 - Steigende Flanke).

9.3.1.10 Bits im Counter Config Register 1

Über das Counter Config Register 1 kann für jeden Zähler ein Zählertyp festgelegt werden. Als Voreinstellung zählt jeder Zähler hoch (Up Counter). Für jeden Zähler gibt es 3 Zählertypen. Gemäß der Aufschlüsselung der Bits, wird die Einstellung für einen Zähler gezeigt, der analog für alle Zähler gilt.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	-	CFT30	CFT21	CFT20	-	CFT10	CFT01	CFT00
Startwert	0	0	0	0	0	0	0	0

Tabelle 28: Bits im Counter Config Register 1 - Unteres Byte (Low Byte)

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	CFT70	CFT61	CFT60	-	CFT50	CFT41	CFT40
Startwert	0	0	0	0	0	0	0	0

Tabelle 29: Bits im Counter Config Register 1 - Oberes Byte (High Byte)

Dezimal	CFTX1	CFTX0	Zählertyp
0	0	0	Zähler zählt hoch, Up Counter (Default)
1	0	1	Zähler zählt runter, Down Counter
2	1	0	Zähler zählt hoch/runter, 2 Sensorbetrieb (2 Kanal Zähler)
3	1	1	Reserviert

Tabelle 30: Counter Config Register 1 - Zählertyp - Aufschlüsselung

Das „x“ im Kürzel CFTX1 bzw. CFTX0 steht stellvertretend für einen der acht Zähler (0 bis 7).

Weitere Informationen und Hinweise zu den Zählern stehen im Kapitel 6.12 Zähler Funktion im PiXtend eIO Digital One, Abschnitt Basis-Wissen.

9.3.1.11 Bits im Hyper-Logic Config Register

Im Hyper-Logic Config Register kann der Anwender für jeden Digitaleingang festzulegen, ob dieser positiv (Voreinstellung) oder negativ (invertiert) betrachtet werden soll. Diese Einstellung ist hilfreich, wenn man einen Sensor verwendet der LOW aktiv ist und beim Erkennen eines Gegenstandes oder Werkstücks von HIGH auf LOW schaltet.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	HLIN7	HLIN6	HLIN5	HLIN4	HLIN3	HLIN2	HLIN1	HLIN0
Startwert	0	0	0	0	0	0	0	0

Tabelle 31: Bits im Hyper-Logic Config Register - Unteres Byte (Low Byte)

Für jeden der acht Digitaleingänge steht ein Bit (HLINx, x = 0 bis 7) zur Verfügung.

Bit 7 ... 0 - HLIN7 ... 0 → Hyper-Logic Prozessor 0 & 1 - Einganginvertierung

Aus: Alle digitalen Eingänge werden positiv betrachtet.

An: Das Signal am gewählten Eingang wird invertiert bzw. negativ gewertet.
Bei dieser Einstellung wird davon ausgegangen, dass ein angeschlossener Sensor oder Schalter im Ruhezustand immer ein HIGH-Pegel liefert.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	HLSM1	HLSM0
Startwert	0	0	0	0	0	0	0	0

Tabelle 32: Bits im Hyper-Logic Config Register - Oberes Byte (High Byte)

Der Hyper-Logic Prozessor 0 und 1 arbeiten üblicherweise mit einer Geschwindigkeit schneller als die digitalen Eingänge reagieren können um eine möglichst schnelle Reaktion zu erreichen. Für manche Anwendungen kann es jedoch sinnvoll sein, die Eingangssignale zu entprellen um somit ein sichereres und stabileres Ausgangssignal für jeden Hyper-Logic Prozessor zu erhalten.

Bit 8 - HLSM0 → Hyper-Logic Prozessor 0 - Signal-Smoothing (Debounce)

Aus: Es wird kein Smoothing (Debounce) durchgeführt.

An: Die Eingangssignale für den Hyper-Logic Prozessor 0 werden entprellt, d.h. bei den Digitaleingängen 0 bis 3, wenn diese in der vom Anwender gewählten Hyper-Logic Verknüpfung vorkommen, muss jedes Signal länger als 2 ms anliegen bevor es als gültig betrachtet wird und von der Hyper-Logic verarbeitet werden kann.

Bit 9 - HLSM1 → Hyper-Logic Prozessor 1 - Signal-Smoothing (Debounce)

Aus: Es wird kein Smoothing (Debounce) durchgeführt.

An: Die Eingangssignale für den Hyper-Logic Prozessor 1 werden entprellt, d.h. bei den Digitaleingängen 4 bis 7, wenn diese in der vom Anwender gewählten Hyper-Logic Verknüpfung vorkommen, muss jedes Signal länger als 2 ms anliegen bevor es als gültig betrachtet wird und von der Hyper-Logic verarbeitet werden kann.

9.3.2. PiXtend eIO Analog One - Modbus Adressen und Funktionen

9.3.2.1 Discrete Inputs - Einlesen der Overload Feedback Signale

Das Analog One Gerät verfügt über zwei Discrete Inputs, darüber können die Overload Feedback Signale der beiden Stromausgänge erfasst werden, beispielsweise zur Erkennung von Kabelbruch. In CODESYS wird jeder Eingang als Bit in einem Byte zur Verfügung gestellt, andere Softwareplattformen und Modbus RTU Implementierungen können davon abweichen. Bitte lesen Sie dies in der jeweiligen Dokumentation zu Ihrem System nach.

Verwenden Sie hier den Modbus RTU Function Code 2 zum Einlesen der Werte.

Bezeichnung	Adresse	Datentyp	Bemerkung
Analog Output 4 Overload Feedback	0	Bit	Overload Feedback Signal des analogen Stromausgangs 4
Analog Output 5 Overload Feedback	1	Bit	Overload Feedback Signal des analogen Stromausgangs 5

Tabelle 33: Analog One: Overload Feedback Signale als Discrete Inputs

Ist alles in Ordnung und am analogen Stromausgang eine Last angeschlossen, dann liefern die Overload Feedback Bits eine 0. Unter CODESYS V3.5 entspricht dies dem Wert FALSE.

Liegt hingegen ein Kabelbruch vor oder es wurde keine Last angeschlossen, dann liefern die Overload Feedback Bits eine 1, unter CODESYS V3.5 entspricht dies dem Wert TRUE, es liegt eine Überlastsituation vor. Prüfen Sie in solchen Fällen die Verkabelung auf Fehler.

9.3.2.2 Input Register - Einlesen der Analogeingänge und weitere Informationen

Die Input Register ermöglichen das Einlesen verschiedener Informationen vom PiXtend eIO Gerät. Alle verfügbaren Modbus Input Register sind in der nachfolgenden Tabelle aufgeführt. Verweise zu Aufschlüsselungen der verschiedenen Bits stehen in der Spalte Bemerkung.

Verwenden Sie den Funktion Code 4 um die Input Register des Gerätes auszulesen.

Die Adressen der Input Register beginnen mit der Zählung bei Adresse 0. Insgesamt hat das Gerät 11 Adressen bzw. 11 Input Register. Der Datentyp jedes Registers ist 16 Bit vom Typ WORD, d.h. 2 Bytes lang und kann die Zahlenwerte von 0 bis 65535 enthalten.

Bezeichnung	Adresse	Bemerkung
Analog Input 0	0	Analoger Spannungseingang 0, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0V - 10V
Analog Input 1	1	Analoger Spannungseingang 1, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0V - 10 V
Analog Input 2	2	Analoger Spannungseingang 2, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0V - 10V
Analog Input 3	3	Analoger Spannungseingang 3, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0V - 10V
Analog Input 4	4	Analoger Stromeingang 4, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0mA - 20mA
Analog Input 5	5	Analoger Stromeingang 5, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0mA - 20mA
Analog Input 6	6	Analoger Stromeingang 6, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0mA - 20mA
Analog Input 7	7	Analoger Stromeingang 7, es wird ein Wertebereich von 0 - 1023 unterstützt, dies entspricht dem Bereich: 0mA - 20mA
Status Register	8	Dieses Register enthält Status Informationen vom Gerät als Rückmeldung an den Anwender. Es kann geprüft werden ob eine Funktion eingeschaltet ist oder nicht. So wird festgestellt ob der Watchdog-Timer aktiv ist. Die Tabelle im Kapitel 9.3.2.4 Bits im Status Register enthält eine Aufschlüsselung der Statusbits.
Fehler Register	9	Enthält ein Abbild der im Gerät festgestellten Fehler. Die Tabelle im Kapitel 9.3.2.5 Bits im Fehler Register enthält eine Aufschlüsselung der Fehlerbits.
Version Register	10	Version der Firmware in Zahlen, 101 = 1.01, 102 = 1.02 etc...

Tabelle 34: Analog One: Input Register

9.3.2.3 Holding Register - Setzen der Analogausgänge und Einstellungen

Die Holding Register ermöglichen das Einstellen der einzelnen Ausgänge und die Änderung verschiedener Konfigurationen, zum Beispiel das Aktivieren des Watchdog-Timers. Alle verfügbaren Modbus Holding Register sind in der nachfolgenden Tabelle aufgeführt. Verweise zu Aufschlüsselungen der verschiedenen Bits, stehen in der Spalte Bemerkung.

Verwenden Sie die Funktion Codes 6 und 16, um die Holding Register des Gerätes zu beschreiben.

Die Adressen der Holding Register beginnen mit der Zählung bei Adresse 0. Insgesamt hat das Gerät 8 Adressen bzw. 8 Holding Register. Der Datentyp jedes Registers ist 16 Bit vom Typ WORD, 2 Bytes lang und kann die Zahlenwerte von 0 bis 65535 enthalten.

Bezeichnung	Adresse	Bemerkung
Analog Output 0	0	Analoger Spannungsausgang 0, es wird ein Wertebereich von 0 - 4095 unterstützt, dies entspricht dem Bereich: 0V - 10V
Analog Output 1	1	Analoger Spannungsausgang 1, es wird ein Wertebereich von 0 - 4095 unterstützt, dies entspricht dem Bereich: 0V - 10V
Analog Output 2	2	Analoger Spannungsausgang 2, es wird ein Wertebereich von 0 - 4095 unterstützt, dies entspricht dem Bereich: 0V - 10V
Analog Output 3	3	Analoger Spannungsausgang 3, es wird ein Wertebereich von 0 - 4095 unterstützt, dies entspricht dem Bereich: 0V - 10V
Analog Output 4	4	Analoger Stromausgang 4, es wird ein Wertebereich von 0 - 4095 unterstützt, dies entspricht dem Bereich: 0mA - 20mA
Analog Output 5	5	Analoger Stromausgang 5, es wird ein Wertebereich von 0 - 4095 unterstützt, dies entspricht dem Bereich: 0mA - 20mA
Watchdog-Timer Register	6	Das Watchdog-Timer Register erlaubt das Einstellen und Einschalten des Geräte Watchdog-Timers zur Überwachung der Bus-Kommunikation. Die Tabelle im Kapitel 9.3.2.6 Bits im Watchdog-Timer Register enthält eine Aufschlüsselung der Konfigurationsbits.
Config Register	7	Im Config Register können verschiedene Einstellungen im Gerät verändert werden. Die Tabelle im Kapitel 9.3.2.7 Bits im Config Register enthält eine Aufschlüsselung der Konfigurationsbits.

Tabelle 35: Analog One: Holding Register

9.3.2.4 Bits im Status Register

Das Status Register zeigt an, welche Funktion im Gerät aktiv ist und welche nicht. Ein zyklisches Abfragen wird empfohlen und es kann geprüft werden, ob eine Funktion aktiviert wurde oder nicht.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	WDT ON
Startwert	0	0	0	0	0	0	0	0

Tabelle 36: Bits im Status Register - Unteres Byte (Low Byte)

Bit 0 - WDT ON → Watchdog-Timer aktiv

Aus: Der Watchdog-Timer ist aus.

An: Der Watchdog-Timer ist aktiv und wird je nach Einstellung zurückgesetzt bzw. löst nach Ablauf der vorgegebenen Timeoutzeit aus.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 37: Bits im Status Register - Oberes Byte (High Byte)

Das obere Byte (High Byte) des Status Registers enthält keine Informationen bzw. Statusbits und kann ignoriert werden.

9.3.2.5 Bits im Fehler Register

Das Fehler Register enthält ein Abbild der im Gerät aufgetretenen Fehler. Im laufenden Betrieb bleiben die Bits solange gesetzt, bis sie über das Bit QUIT ERR im Config Register abgelöscht werden oder ein Power-Cycle durchgeführt wird. Ausgenommen von dieser Regel ist das Watchdog-Timer Fehlerbit, weitere Informationen stehen im Erklärungstext zu Bit 4.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	COM ERR	COF ERR	TEL ERR	WDT ERR	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 38: Bits im Fehler Register - Unteres Byte (Low Byte)

Bit 4 - WDT ERR → Der Watchdog-Timer ist abgelaufen

Dieses Bit zeigt an ob vor dem letzten Einschalten der Watchdog-Timer abgelaufen war oder nicht. Da das Gerät bei Ablauf des Watchdog-Timers in den sicheren Zustand geht, kann dieses Bit immer nur nach einem darauffolgenden Power-Cycle abgefragt werden. Das Bit kann über das QUIT ERR Bit im Config Register im laufenden Betrieb abgelöscht werden oder es wird ein weiterer Power-Cycle durchgeführt.

Bit 5 - TEL ERR → Function Code- oder Adressfehler

Wurde dem Gerät ein Telegramm bzw. eine Nachricht mit einem nicht unterstützen Funktion Code geschickt oder die empfangene Modbus RTU Nachricht enthält eine nicht vorhandene Adresse, dann wird dieses Fehlerbit gesetzt. Im Steuerungsprogramm lässt sich nachträglich feststellen, ob einer dieser Fehler seit dem letzten Einschalten aufgetreten ist, auch wenn die ERR-LED nicht leuchtet.

Bit 6 - COF ERR → Konfigurationsfehler

Dieses Bit wird gesetzt, wenn im Gerät ein Konfigurationsfehler festgestellt wird. Im normalen Betrieb taucht dieser Fehler nicht auf, außer es werden die DIP Schalter während des Betriebs geändert. Um diesen Zustand zu beheben, müssen die DIP-Schalter in ihre ursprüngliche Position geschoben werden oder es erfolgt ein Power Cycle, dann wird die neue Geräte-Konfiguration übernommen.

Bit 7 - COM ERR → Kommunikationsfehler

Dieses Bit wird gesetzt, wenn das Gerät einen Kommunikationsfehler feststellt. In der Regel handelt es sich hier um drei Fehlerzustände. Das Gerät kann einen Parity-, Frame- oder Overrun-Fehler feststellen.

Ein Parity-Fehler tritt meistens auf, wenn die Einstellungen am Steuerungsgerät nicht korrekt sind. Gleiches gilt für den Frame-Fehler, hier stimmt in den meisten Fällen die Baudrate nicht und das Gerät empfängt keine gültigen Daten. Der Overrun-Fehler bedeutet, dass zu schnell zu viele Daten über den Bus geschickt werden und das Gerät nicht alles aufnehmen kann.

Prüfen Sie die Einstellungen am Steuergerät, stimmen diese mit der Konfiguration des Geräts überein, wurde die Verkabelung korrekt durchgeführt und liegt keine Störungen am Bus vor.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 39: Bits im Fehler Register - Oberes Byte (High Byte)

Das obere Byte (High Byte) des Fehler Registers enthält keine Informationen bzw. Fehlerbits und kann ignoriert werden.

9.3.2.6 Bits im Watchdog-Timer Register

Das Watchdog-Timer Register ermöglicht dem Anwender Einfluss auf das Verhalten des Geräte Watchdog-Timers zu nehmen. Die Einstellungsmöglichkeiten reichen von der Timeoutzeit bis zum Verhalten des Watchdog-Timers.

Wird der Watchdog-Timer ohne weitere Konfiguration aktiviert, ist eine Timeoutzeit von 16 Millisekunden voreingestellt. Das Rücksetzen des Watchdog-Timers erfolgt, wenn Aktivität am Bus festgestellt wird, unabhängig davon ob die Daten für das Gerät bestimmt sind oder nicht.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	-	WDTRST	WDTEN	WB	WT3	WT2	WT1	WT0
Startwert	0	0	0	0	0	0	0	0

Tabelle 40: Bits im Watchdog-Timer Register - Unteres Byte (Low Byte)

Bit 3...0 - WT3...WT0 → Watchdog-Timer Timeoutzeit (Watchdog-Timer Timeout)

Die Watchdog-Timer Zeit definiert, wie lange der Watchdog-Timer aktiv ist bevor er auslöst.

Die Tabelle zeigt, welches Bitmuster eingestellt werden muss, damit der Watchdog-Timer bei seiner Aktivierung die Timeoutzeit übernimmt und damit arbeitet. Bitte beachten Sie, werden keine Bits gesetzt, dann verwendet der Watchdog-Timer die kürzeste Zeit, 16 ms. Diese Zeitspanne ist für eine Modbus RTU Kommunikation viel zu kurz. Wählen Sie in jedem Fall einen größeren Wert, zum Beispiel 4 Sekunden.

Watchdog-Timer Timeoutzeit:

WT3	WT2	WT1	WT0	Timeoutzeit
0	0	0	0	16 ms
0	0	0	1	32 ms
0	0	1	0	64 ms
0	0	1	1	0.125 s
0	1	0	0	0.25 s
0	1	0	1	0.5 s
0	1	1	0	1.0 s
0	1	1	1	2.0 s
1	0	0	0	4.0 s
1	0	0	1	8.0 s

Tabelle 41: Watchdog-Timer Timeoutzeit Übersicht

Vorgehen zum Setzen/Ändern der Watchdog Zeit im laufenden Betrieb:

- Watchdog ist deaktiviert
- Gewünschte Watchdog-Timer Timeoutzeit setzen
- Watchdog einschalten

Bit 4 - WB → Watchdog-Timer Verhalten (Watchdog-Timer Behaviour)

Aus: Die analogen Ausgänge verbleiben in ihrem aktuellen Zustand, wenn der Watchdog-Timer abläuft.

An: Alle analogen Ausgänge werden aktiv auf den Wert 0 gesetzt, wenn der Watchdog-Timer abläuft. Dies entspricht 0V bzw. 0mA.

Bit 5 - WDTEN → Watchdog-Timer einschalten (Watchdog-Timer enable)

Aus: Der Watchdog-Timer ist aus.

An: Der Watchdog-Timer ist aktiviert bzw. eingeschaltet.

Bit 6 - WDTRST → Watchdog-Timer Reset Verhalten (Watchdog-Timer Reset Type)

Aus: Der Watchdog-Timer Reset wird bei Bus-Aktivität ausgeführt, unabhängig davon um welche Daten es sich handelt. Anhand dieser Einstellung wird definiert, ob der Master grundsätzlich kommuniziert oder ob er ausgefallen ist. Erfolgt innerhalb der gewählten Watchdog-Timer Zeit keine Kommunikation am Bus, läuft der Watchdog-Timer ab und löst aus.

An: Der Watchdog-Timer Reset wird nur dann ausgeführt, wenn das Gerät eine gültige Modbus RTU Nachricht vom Master erhält. Wird das Gerät vom Master nicht innerhalb der Watchdog-Timer Zeit mit einer gültigen Modbus RTU Nachricht angesprochen, läuft der Watchdog-Timer ab. Diese Einstellung ist hilfreich um festzustellen, ob. in einem großen Modbus RTU Verbund jedes Gerät innerhalb der gewählten Watchdog-Timer Zeit angesprochen wird.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 42: Bits im Watchdog Register - Oberes Byte (High Byte)

Das obere Byte (High Byte) des Watchdog Registers enthält keine Informationen bzw. Konfigurationsbits und kann ignoriert werden.

9.3.2.7 Bits im Config Register

Über das Bit in diesem Register kann der interne Fehlerspeicher im Betrieb zurückgesetzt werden.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	-	-	QUIT ERR
Startwert	0	0	0	0	0	0	0	0

Tabelle 43: Bits im Config Register - Unteres Byte (Low Byte)

Bit 0 - QUIT ERR → Fehlerspeicher löschen (Quit Errors)

Aus: Das Gerät setzt die Fehlerbits im Fehlerspeicher.

An: Wird dieses Bit auf 1 gesetzt, erfolgt die Löschung des internen Geräte-Fehlerspeichers. Nach dem Setzen des Bits überprüfen, ob alle Fehler im Fehlerregister gelöscht wurden.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 44: Bits im Config Register - Oberes Byte (High Byte)

Das obere Byte (High Byte) des Config Registers enthält keine Informationen bzw. Konfigurationsbits und kann ignoriert werden.

9.4. Modbus RTU Fehlernummern

Das Modbus RTU Protokoll kennt verschiedene Fehlernummern (Exception Codes) die ein Master als Antwort von einem Slave zurück erhalten kann.

Die PiXtend eIO Geräte verfügen über zwei Fehlernummern, die gemeldet werden. Die nachfolgende Tabelle gibt einen Überblick über diese Fehler.

Fehlernummern:

Bezeichnung	Nummer	Bemerkung
Illegal Function	1	Der verwendete Funktion Code wird nicht unterstützt.
Illegal Data Value	3	Die übermittelten Daten enthalten einen Fehler oder des angegebenen Coils / Discrete Inputs / Register Adresse ist falsch bzw. existiert nicht.

Tabelle 45: Modbus RTU Fehlernummern (Exception Codes)

Ausdruck	Datentyp	Wert
 Device.Application.Modbus_Slave_COM_Port	IoDrvModbus.ModbusSlaveComPort	
 xTrigger	BOOL	FALSE
 xReset	BOOL	FALSE
 xAcknowledge	BOOL	FALSE
 xDoInit	BOOL	TRUE
 xInitDone	BOOL	TRUE
 xBusy	BOOL	FALSE
 xDone	BOOL	FALSE
 xError	BOOL	TRUE
 byModbusError	MB_ERRORCODES	ILLEGAL_DATA_VALUE
 iChannelIndex	INT	-1

Abbildung 12: CODESYS V3.5 - Modbus RTU - Gerät meldet Fehler

9.5. Beispiel CODESYS V3.5

Dieses Kapitel zeigt für jedes PiXtend eIO Gerät ein einfaches Beispiel mit dem SPS Programmierwerkzeug CODESYS V3.5.

In den Beispielen gehen wir von der Annahme aus, dass die Geräte auf Werkseinstellung stehen. Das Digital One Gerät hat die Geräte-Adresse 1 und das Analog One Gerät die Geräte-Adresse 3. Die Schalter 1 - 6 des DIP Blocks stehen auf 2 - CONFIG in der unteren Position, sie sind ausgeschaltet. Der COM-Port 1 in CODESYS V3.5 wird mit einer Baudrate von 19200 eingestellt, mit Parität „Even“, einem Stoppbit und 8 Datenbits (8E1). Das jeweilige Modul ist verkabelt und mit dem RS485 Port am PiXtend V2 -L- verbunden. PiXtend V2 -S- und PiXtend V1.x benötigen einen entsprechenden USB-zu-RS485 Dongle.

9.5.1. PiXtend eIO Digital One

Wir zeigen, wie man mit dem CODESYS V3.5 Modbus RTU Master am Digital One Gerät verschiedene digitale Ausgänge setzen und diese über die digitalen Eingänge wieder einlesen kann. Die gewünschte Bitfolge wird auf die Digital One Coils geschrieben und wir lesen den Zustand über die Discrete Inputs wieder zurück. Wir wenden die Funktion Codes 02 und 15 an.

Zunächst muss der Digitalausgang 0 mit dem Digitaleingang 0 über ein Kabel verbunden werden. Weitere Details zum Anschluss einer Stromversorgung und Verkabelung eines Digital One Gerätes, entnehmen Sie dem PiXtend eIO Hardware Handbuch.

9.5.1.1 Neues Projekt anlegen und SPS auswählen

Öffnen Sie in

CODESYS das Datei-Menü und wählen Sie den Menüeintrag

Neues Projekt...

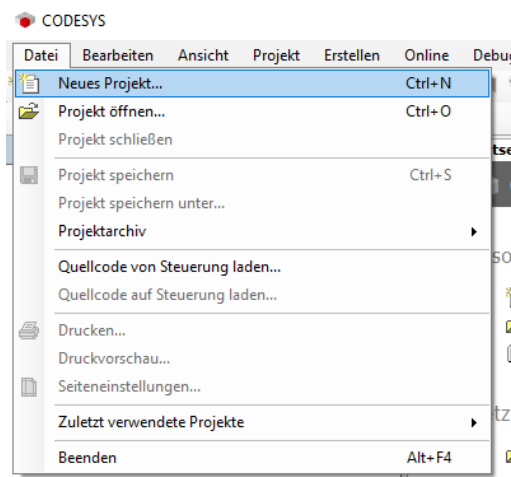


Abbildung 13: CODESYS - Datei Menü

Im Menü Neues Projekt wählen Sie die Kategorie Projekte aus und selektieren unter Vorlagen (rechts) den Eintrag Standardprojekt. Vergeben Sie unter Namen einen Projektnamen, zum Beispiel. Modbus - Digital One Beispiel. Wählen Sie nun unter Ort den Speicherplatz des Projektes aus.

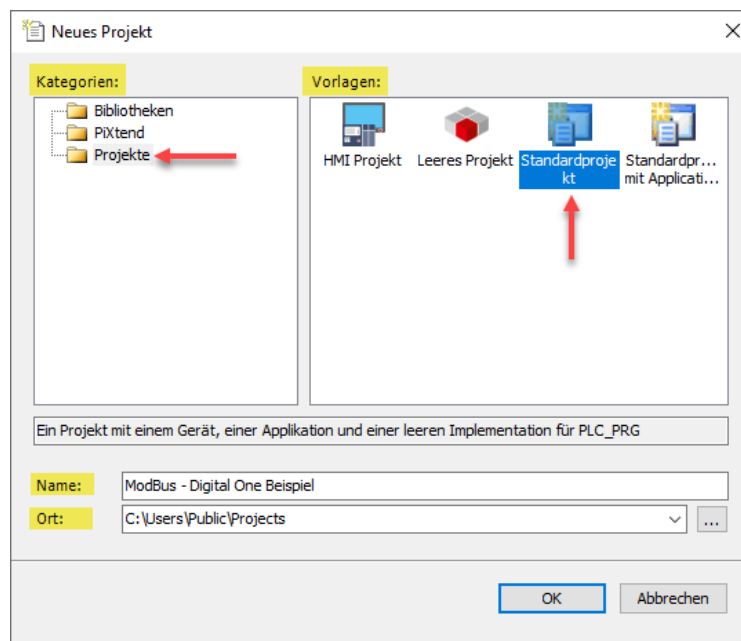


Abbildung 14: CODESYS - Neues Projekt

Im Fenster Standardprojekt wählen Sie Ihre SPS aus, alle PiXtend V2 Anwender wählen hier den Raspberry Pi als Gerät aus.

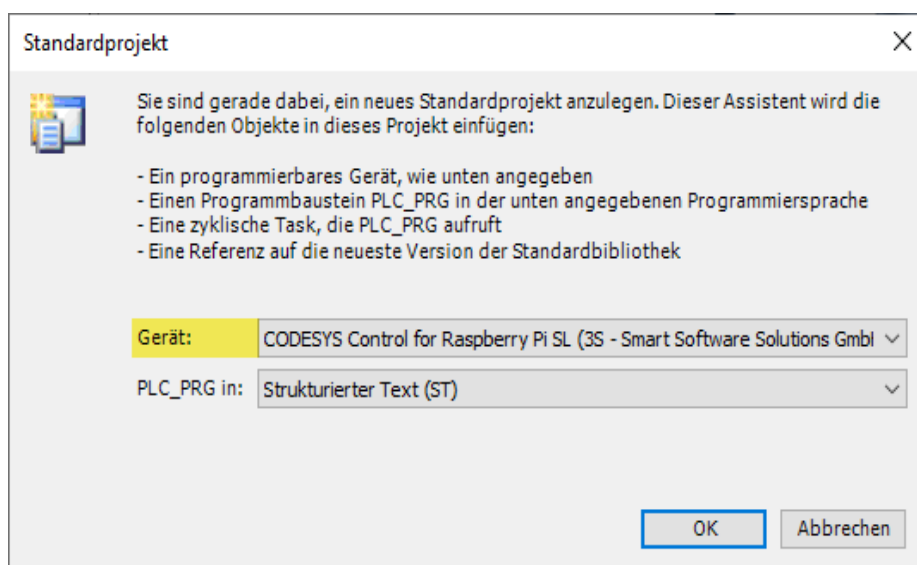


Abbildung 15: CODESYS - Standardprojekt - Geräteauswahl

Klicken sie auf OK und das neue Projekt wird erstellt.

9.5.1.2 Modbus COM Port einfügen

Ihr Projekt wurde angelegt und nun klicken Sie im Geräte Fenster auf den Eintrag Device (1). Führen Sie einen Rechtsklick auf dem Eintrag aus und wählen Sie den Eintrag Gerät anhängen... (2) aus.

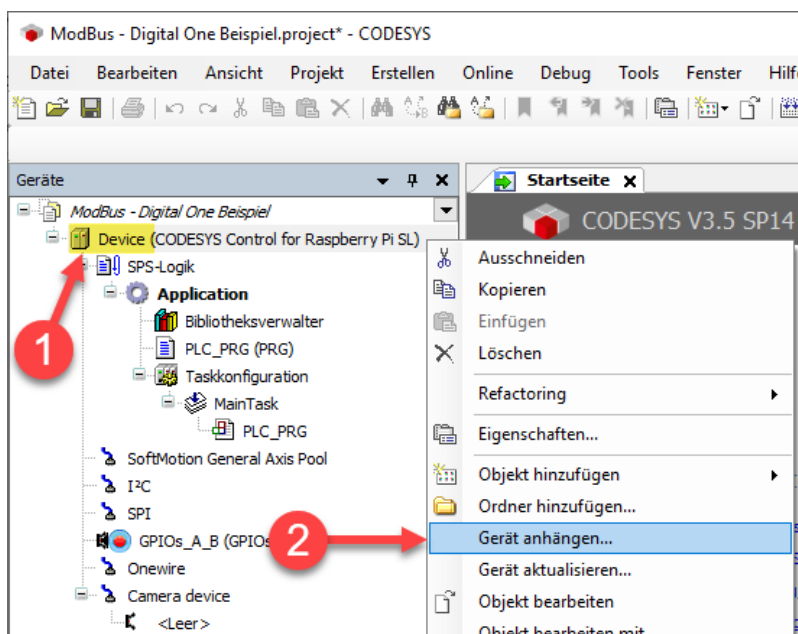


Abbildung 16: CODESYS - Gerät anhängen Menü

Öffnen Sie im Fenster Gerät anhängen, über das Plus-Zeichen, den Eintrag Feldbusse. Öffnen Sie den Eintrag Modbus und wählen Sie das Gerät Modbus COM Port aus.

Wählen Sie diesen Eintrag aus und klicken Sie rechts unten im Fenster auf die Schaltfläche Gerät anhängen.

Schließen Sie das Fenster und wählen Sie im Geräte Fenster den eben hinzugefügten Modbus_COM_Port Eintrag aus.

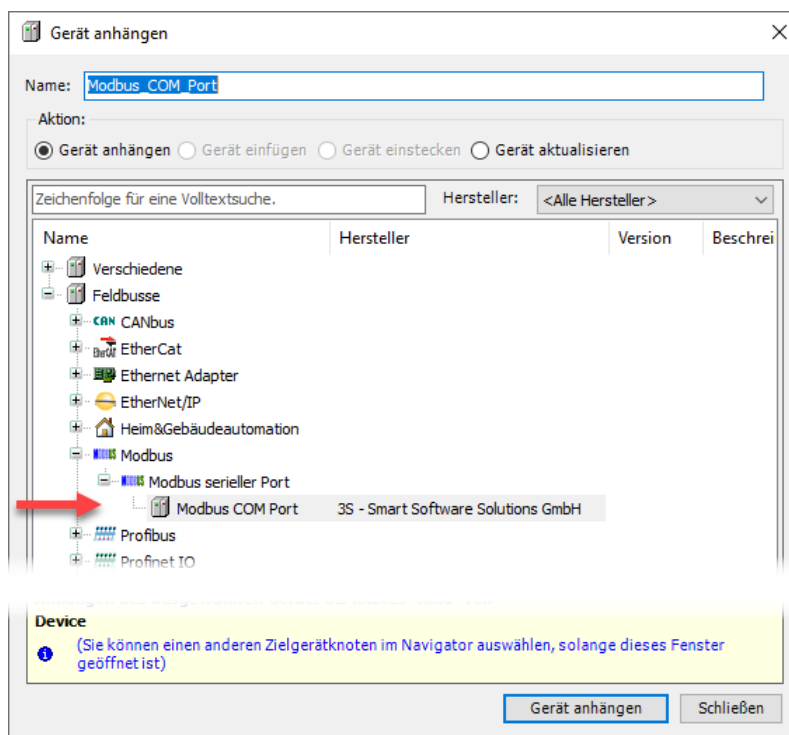


Abbildung 17: CODESYS - Gerät anhängen Fenster

Führen Sie auf dem Eintrag einen Rechtsklick aus und wählen Sie im Menü den Eintrag Gerät anhängen. Im Fenster Gerät anhängen suchen Sie nach dem Eintrag Modbus Master, COM Port. Fügen Sie dieses Gerät an den vorhandenen Modbus COM Port an.

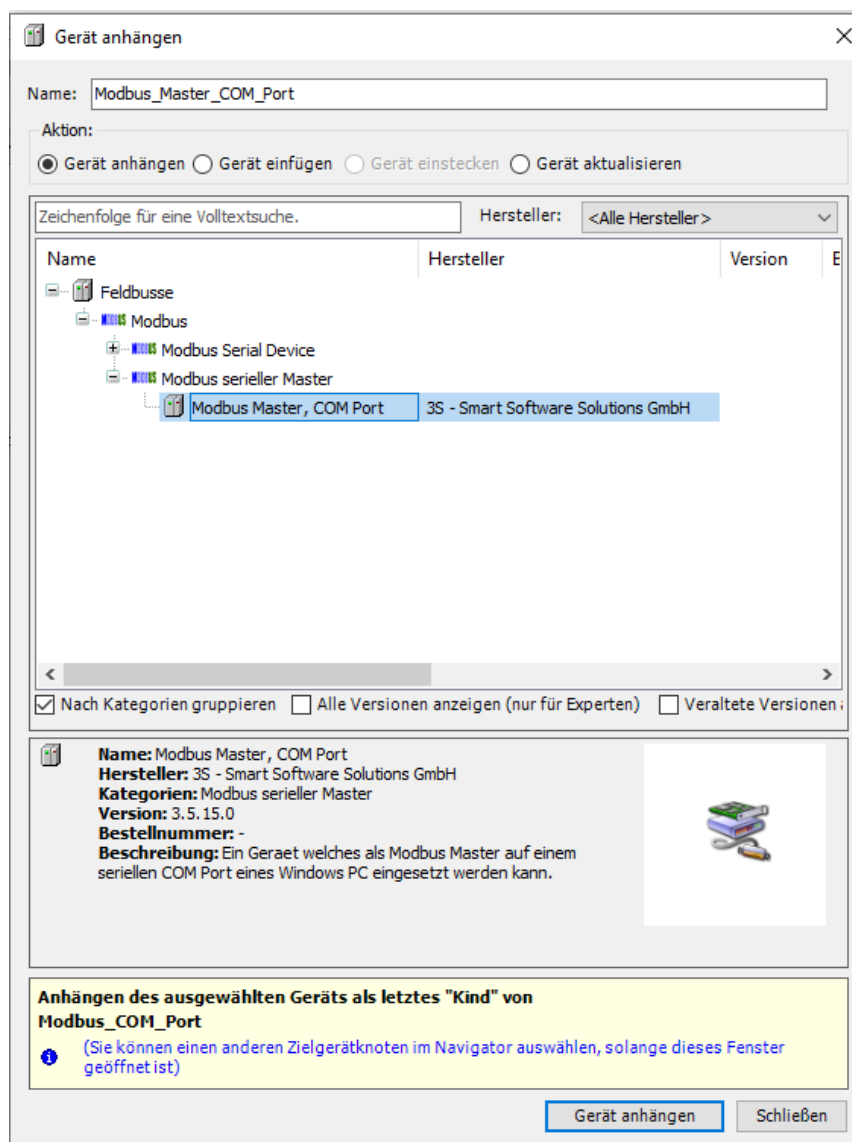


Abbildung 18: CODESYS - Gerät anhängen - Modbus Master

Schließen Sie das Fenster, wählen Sie im Geräte Fenster den Eintrag Modbus_Master_COM_Port aus. Führen Sie auf dem Eintrag einen Rechtsklick aus und wählen Sie im Menü den Eintrag Gerät anhängen. Im Fenster Gerät anhängen suchen Sie nach dem Eintrag Modbus Slave, COM Port und fügen Sie dieses Gerät an den vorhandenen Modbus_Master_COM_Port an.

Schließen Sie das Fenster. Alle Geräte sind eingefügt und vorhanden. Jetzt fehlt nur noch die Konfiguration.

Im CODESYS Geräte Fenster sollten Sie eine Konfiguration wie in Abbildung 19 vorfinden, unabhängig von Ihrer SPS.

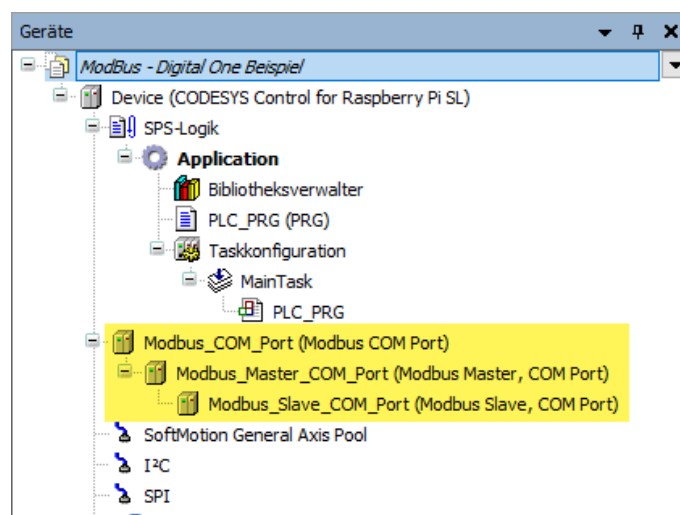


Abbildung 19: CODESYS - Alle Geräte angehängt

9.5.1.3 Modbus COM Port konfigurieren

Führen Sie einen Doppelklick auf den Geräte-Eintrag Modbus_COM_Port aus, um die Einstellungsseite für den seriellen Port zu öffnen. Wählen Sie den Reiter Allgemein aus und stellen Sie folgende Konfiguration ein:

COM-Port: 1

Baurate: 19200, Parität: Even, Daten-Bits: 8 und Stopbits: 1

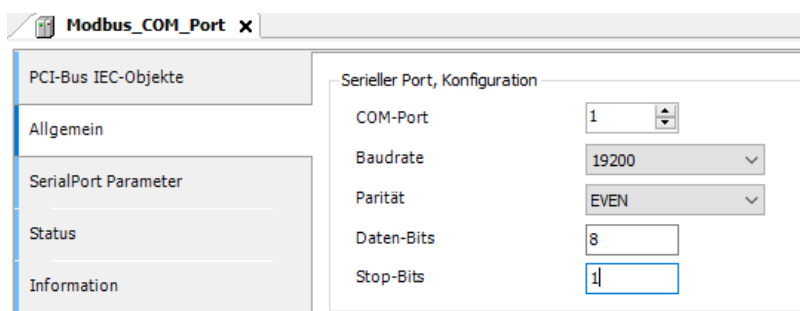


Abbildung 20: CODESYS - serielle Einstellungen

Wenn Sie kein PiXtend V2 -S-/-L- oder V1.x als SPS verwenden, schlagen Sie im Handbuch Ihrer SPS nach welcher COM Port RS485 geeignet ist und stellen Sie diese Nummer unter COM-Port, anstelle der Nummer 1, ein.

9.5.1.4 Modbus Master COM Port konfigurieren

Im Modbus Master COM Port muss eine Einstellung im Reiter Allgemein vorgenommen werden. Aktivieren Sie die Option automatischer Neustart Kommunikation.

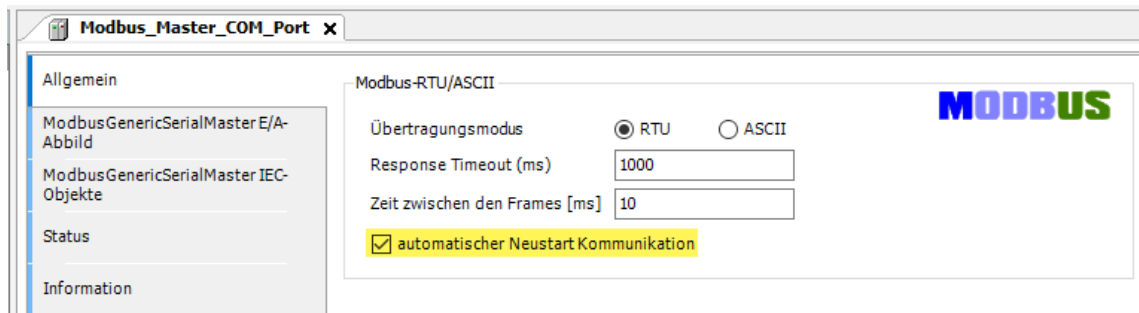


Abbildung 21: CODESYS - Modbus Master Einstellungen

9.5.1.5 Modbus Slave COM Port konfigurieren

Beim Modbus Slave erfolgt im Reiter Allgemein keine Anpassung. Das Digital One Gerät besitzt ab Werk die Geräte-Adresse 1 und das Timeout bleibt unverändert.

Die Einstellung in CODESYS entsprechen dem folgenden Bild:

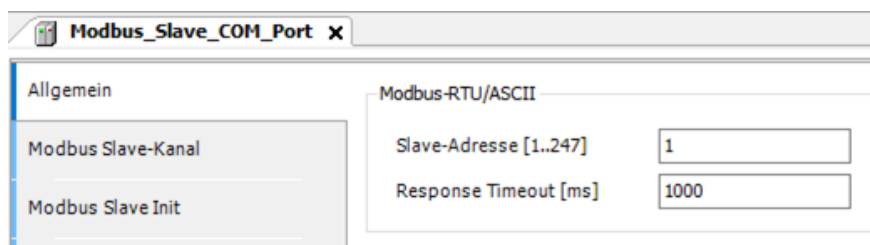


Abbildung 22: CODESYS - Modbus Slave Einstellungen

9.5.1.6 Modbus Slave-Kanal (Funktion Codes) einstellen

Wechseln Sie zum Reiter Modbus Slave-Kanal und klicken Sie rechts unten auf die Schaltfläche Kanal hinzufügen. Vergeben Sie einen Namen, beispielsweise Digital Outputs. Bitte beachten, dieser kann später nicht mehr geändert werden. Es ist wichtig, dass Sie bei Zugriffstyp den Funktion Code 15 - Write Multiple Coils einstellen und im Abschnitt WRITE Register einen Offset von 0 und eine Länge von 8 eintragen. Alle weiteren Einstellungen bleiben unverändert.

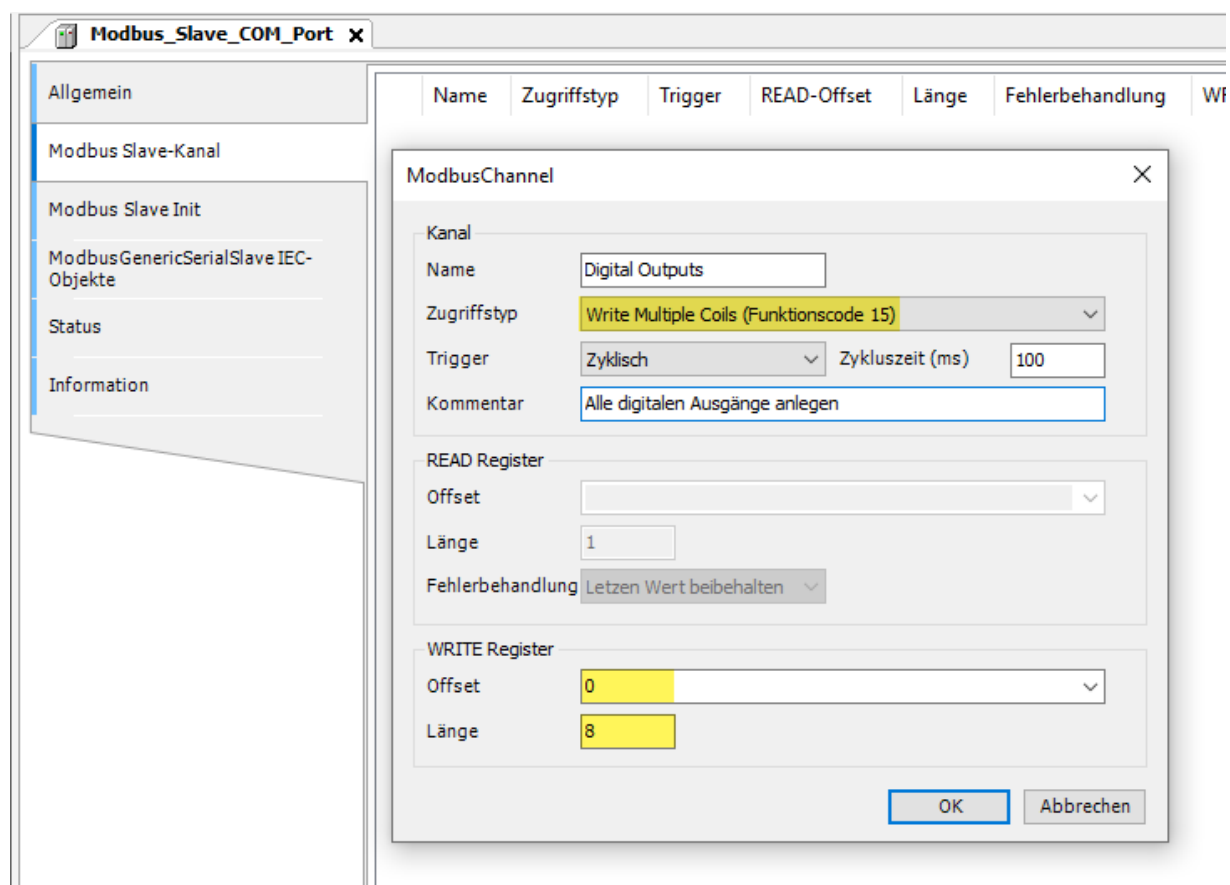


Abbildung 23: CODESYS - Modbus Slave Kommunikationskanal anlegen

Mit einem Klick auf die OK Schaltfläche fügen Sie in CODESYS einen neuen Modbus RTU Kommunikationskanal hinzu. Klicken Sie im Anschluss sofort auf die Schaltfläche Kanal hinzufügen... um einen weiteren Modbus RTU Kommunikationskanal einzufügen.

Über diesen zweiten Kanal werden die digitalen Eingänge vom Digital One Gerät gelesen.

Für diesen zweiten Kanal stellen Sie unter Zugriffstyp den Function Code 2 - Read Discrete Inputs ein. Im Abschnitt READ Register tragen Sie unter Offset eine 0 und bei Länge eine 8 ein. Somit gibt es für jeden digitalen Eingang ein Bit. Die Einstellung Fehlerbehandlung können Sie ändern und Auf ZERO setzen. Ist das Digital One nicht erreichbar, werden alle Digitaleingänge in CODESYS auf FALSE gesetzt. Die anderen Einstellungen bleiben unverändert.

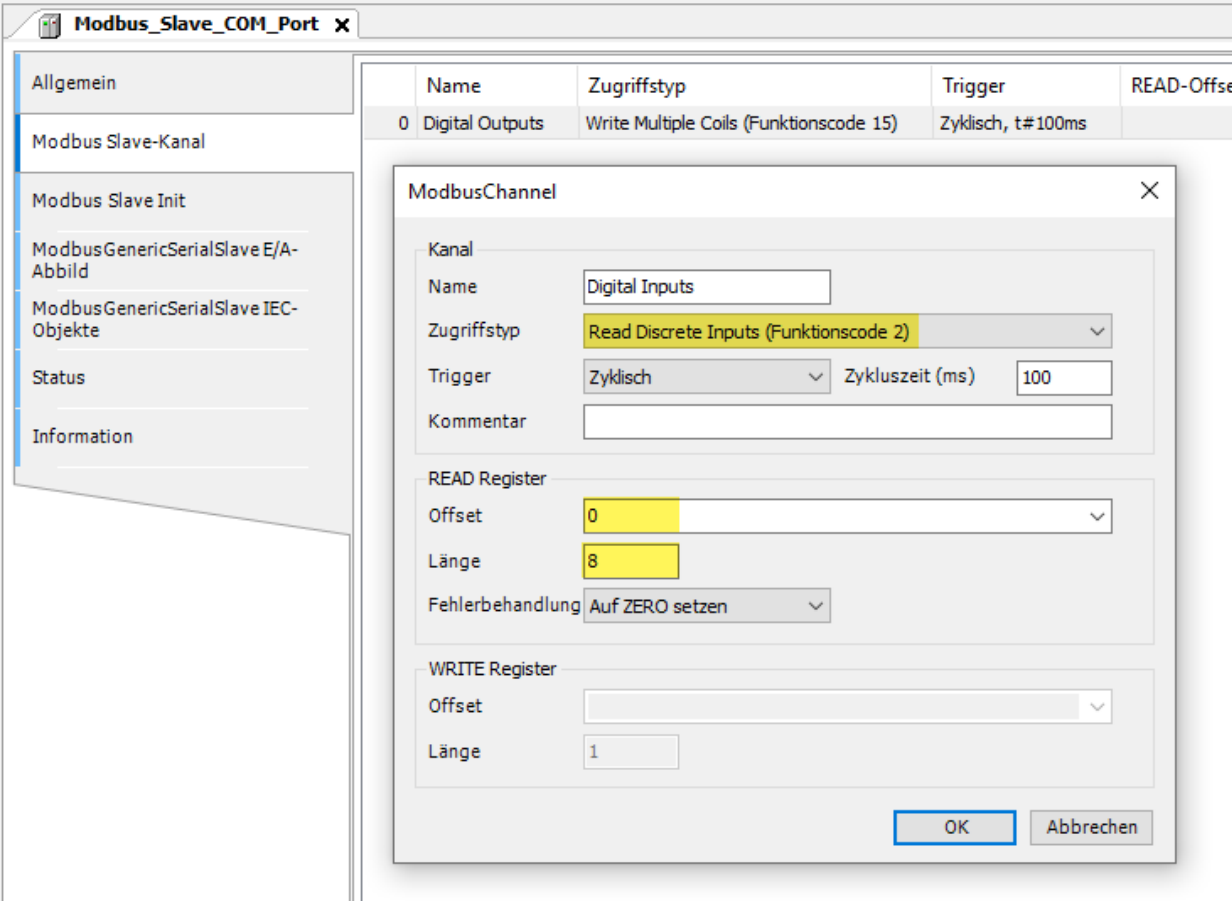


Abbildung 24: CODESYS - Modbus Slave Kommunikationskanal anlegen (2)

Mit einem Klick auf die OK Schaltfläche fügen Sie in CODESYS einen neuen Modbus RTU Kommunikationskanal hinzu. Die Einstellungen in CODESYS entsprechen folgender Abbildung:

Name	Zugriffstyp	Trigger	READ-Offset	Länge	Fehlerbehandlung	WRITE Offset	Länge
0 Digital Outputs	Write Multiple Coils (Funktionscode 15)	Zyklisch, t#100ms				16#0000	8
1 Digital Inputs	Read Discrete Inputs (Funktionscode 02)	Zyklisch, t#100ms	16#0000	8	Auf ZERO setzen		

Abbildung 25: CODESYS - Modbus Slave Kommunikationskanalübersicht

Nun muss die Variablen Aktualisierung auf die Einstellung Aktiviert 2 umgestellt werden. Gehen Sie dazu auf den Reiter ModbusGenericSerialSlave E/A-Abbild im Modbus Slave und ändern Sie die Einstellung rechts unten auf den Eintrag Aktiviert 2 ab.

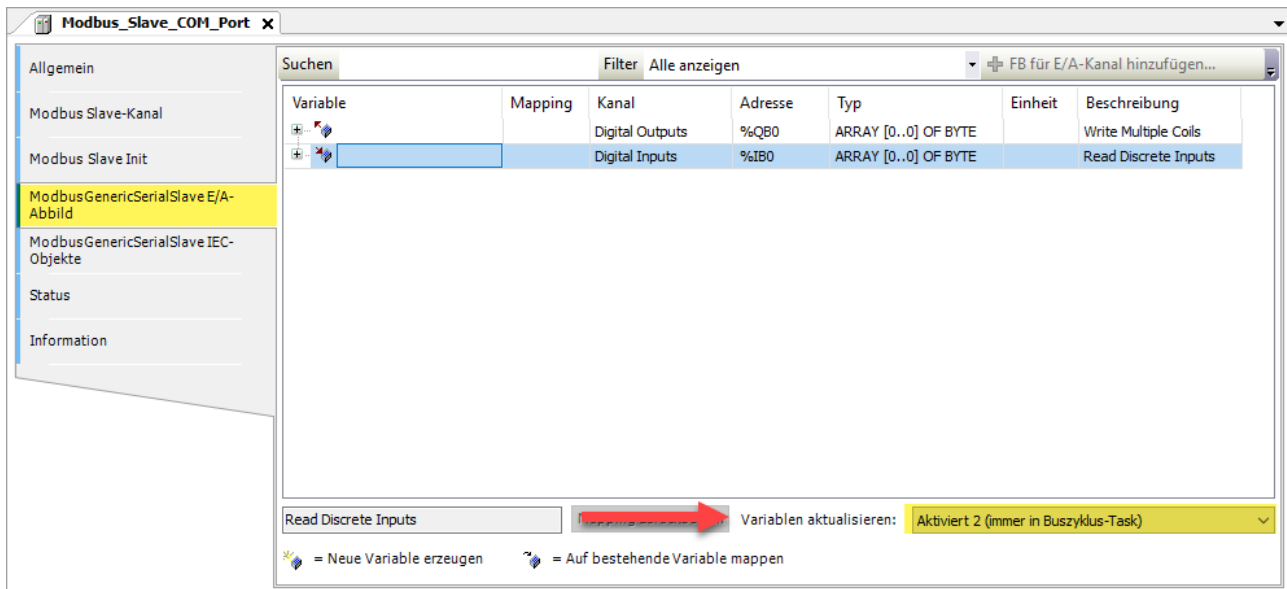
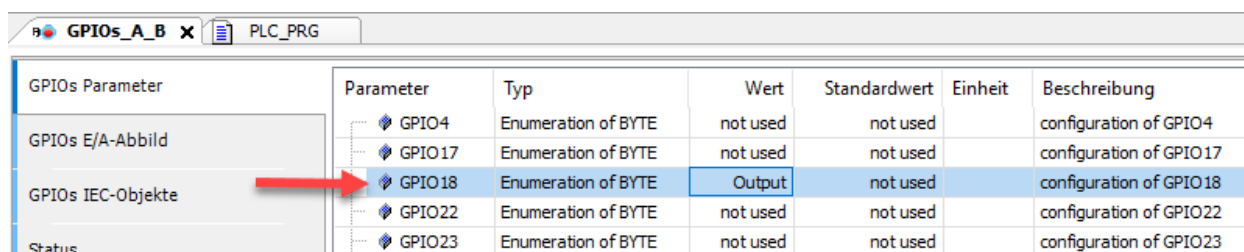


Abbildung 26: CODESYS - Variablen aktualisieren

Nun muss noch ein kleines Programm geschrieben werden, das auf PiXtend V2-L- den seriellen Anschluss von RS232 auf RS485 umschaltet. Verwenden Sie ein PiXtend V2 -S- oder V1.x mit einem RS485 Dongle oder eine andere SPS, dann können Sie den nächsten Schritt einfach auslassen.

9.5.1.7 PiXtend V2 -L- Umschaltung serieller Port RS232 zu RS485

Das PiXtend V2 -L- verfügt über eine integrierte RS485 Schnittstelle, diese muss von CODESYS aktiviert werden, da die Standardeinstellung auf der RS232 Schnittstelle liegt. Über den GPIO18 auf dem Raspberry Pi kann zwischen beiden seriellen Schnittstellentypen ausgewählt werden. Richten Sie den GPIO18 des Raspberry Pi als Ausgang (Output) ein und vergeben Sie im GPIO E/A-Abbild den Variablennamen RPI_GPIO_18.



GPIOs Parameter	Parameter	Typ	Wert	Standardwert	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	Output	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

Abbildung 27: CODESYS - GPIO18 als Ausgang einstellen

GPIOs_A_B x

PLC_PRG

GPIOs Parameter

GPIOs E/A-Abbild

GPIOs IEC-Objekte

Status

Information

Suchen

Filter

Alle anzeigen

Variable	Mapping	Kanal	Adresse	Typ
 		digital inputs (GPIO0..GPIO31)	%ID1	DWORD
 		digital outputs (GPIO0..GPIO31)	%QD1	DWORD
 		Bit4	%QX4.4	BOOL
 		Bit17	%QX6.1	BOOL
 		Bit18	%QX6.2	BOOL
 		Bit22	%QX6.6	BOOL
 		Bit23	%QX6.7	BOOL
 		Bit24	%QX7.0	BOOL
 		Bit25	%QX7.1	BOOL

Abbildung 28: CODESYS - GPIO18 als Variable anlegen

Alles was jetzt noch fehlt ist ein ganz kurzes Programm das uns die RPI_GPIO_18 Variable beim Start auf TRUE setzt. Eine Möglichkeit wie dies gehen kann sehen Sie im nachfolgenden Bild:

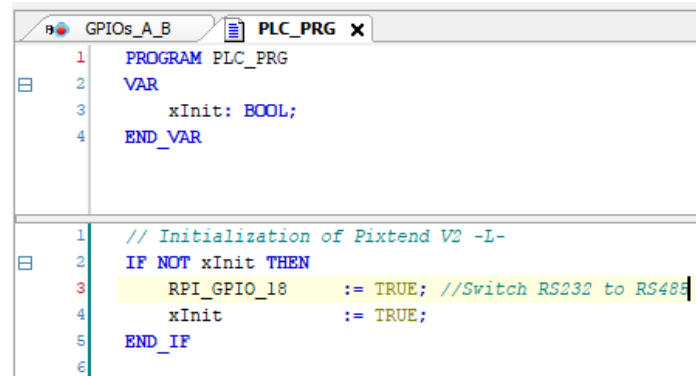


Abbildung 29: CODESYS - RS485 aktivieren

Alternativ schreiben Sie in die erste Zeile des PLC_PRG Bausteins folgende Anweisung:

```
RPI_GPIO_18 := TRUE; //Switch RS232 to RS485
```

Fertig, das Programm auf die Steuerung übertragen, startet und Sie können beginnen!

Weitere Informationen zur seriellen Schnittstelle auf dem Raspberry Pi und PiXtend V2 und V1.x Geräten finden Sie in den Kapiteln 6.8 Serielle Schnittstelle vorbereiten für eigenes SD Karten Abbild, 6.9 PiXtend V1.3/V2-S- mit USB zu RS485 Dongle erweitern und 6.10 PiXtend V2-L- Serielle Schnittstelle prüfen.

9.5.1.8 Programm übertragen und starten

Verbinden Sie sich mit der Steuerung und übertragen Sie das Programm. Hat alles geklappt, dann starten Sie das Programm über die Start-Taste oder mit F5.

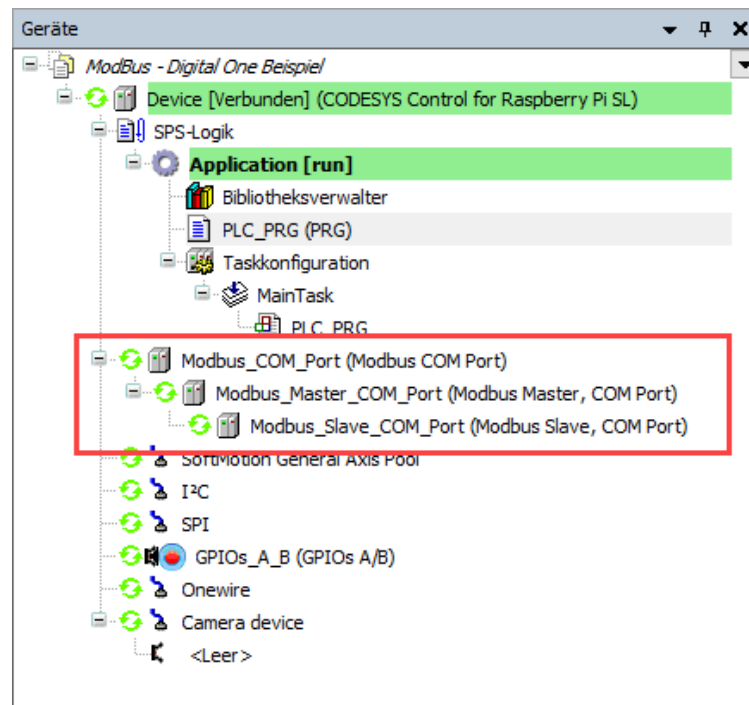


Abbildung 30: CODESYS - Modbus Bus läuft

Es hat funktioniert, wenn Sie vor den drei Modbus Geräten im CODESYS Geräte Fenster grüne Kreise aus Pfeilen sehen. Ist die Farbe der Kreise orange, ist keine CODESYS V3.5 Runtime Lizenz auf dem Raspberry Pi installiert, es liegt kein Fehler des Modbus Masters vor.

Es gibt eine Einschränkung, die CODESYS Runtime läuft 2 Stunden und stoppt anschließend. Der Modbus RTU Master hingegen kann nur „30 Minuten“ getestet werden. Er stellt dann seine Arbeit ein und der Raspberry Pi muss neu gestartet werden.

Verwenden Sie eine andere Steuerung, dann konsultieren Sie Ihren Hersteller, sollten nicht alle Kreise grün sein. Gegebenenfalls benötigen Sie eine Modbus RTU Zusatzlizenz um Modbus RTU unter CODESYS auf Ihrer SPS nutzen können.

9.5.1.9 Ausgang schalten und Eingang lesen

Das Programm und der Modbus Bus arbeiten. Jetzt schalten wir den Digitalausgang 0 auf TRUE, der Digitaleingang 0 sollte ebenfalls auf TRUE schalten. Dafür muss der Digitalausgang 0 mit dem Digitaleingang 0 verbunden sein. Kehren Sie zum ModbusGenericSerialSlave E/A-Abbild Reiter im Modbus_Slave_COM_Port Gerät zurück. Klappen Sie beide Kanäle auf bis alle Ein- und Ausgangsbits des Slaves ersichtlich werden

Variable	Mapping	Kanal	Adresse	Typ	Aktueller Wert	Vorbereiteter Wert	Einheit	Beschreibung
		Digital Outputs	%QB0	ARRAY [0..0] OF BYTE				Write Multiple Coils
		Digital Outputs[0]	%QB0	BYTE	0			Write Multiple Coils
		Bit0	%QX0.0	BOOL	FALSE			0x0000
		Bit1	%QX0.1	BOOL	FALSE			0x0001
		Bit2	%QX0.2	BOOL	FALSE			0x0002
		Bit3	%QX0.3	BOOL	FALSE			0x0003
		Bit4	%QX0.4	BOOL	FALSE			0x0004
		Bit5	%QX0.5	BOOL	FALSE			0x0005
		Bit6	%QX0.6	BOOL	FALSE			0x0006
		Bit7	%QX0.7	BOOL	FALSE			0x0007
		Digital Inputs	%IB0	ARRAY [0..0] OF BYTE				Read Discrete Inputs
		Digital Inputs[0]	%IB0	BYTE	0			Read Discrete Inputs
		Bit0	%IX0.0	BOOL	FALSE			0x0000
		Bit1	%IX0.1	BOOL	FALSE			0x0001
		Bit2	%IX0.2	BOOL	FALSE			0x0002
		Bit3	%IX0.3	BOOL	FALSE			0x0003
		Bit4	%IX0.4	BOOL	FALSE			0x0004
		Bit5	%IX0.5	BOOL	FALSE			0x0005
		Bit6	%IX0.6	BOOL	FALSE			0x0006
		Bit7	%IX0.7	BOOL	FALSE			0x0007

Abbildung 31: CODESYS - Modbus Slave Coils & Discrete Inputs Übersicht

Setzen Sie über die Spalte Vorbereiteter Wert das Bit0, das ist der Digitalausgang 0 auf dem Digital One Gerät. Nach kurzer Zeit sollte das Bit0 der Digitaleingänge TRUE werden, das bedeutet, dass am Digitaleingang 0 ein HIGH-Pegel anliegt.

Variable	Mapping	Kanal	Adresse	Typ	Aktueller Wert
		Digital Outputs	%QB0	ARRAY [0..0] OF BYTE	
		Digital Outputs[0]	%QB0	BYTE	1
		Bit0	%QX0.0	BOOL	TRUE
		Bit1	%QX0.1	BOOL	FALSE
		Digital Inputs	%IB0	ARRAY [0..0] OF BYTE	
		Digital Inputs[0]	%IB0	BYTE	1
		Bit0	%IX0.0	BOOL	TRUE
		Bit1	%IX0.1	BOOL	FALSE

Abbildung 32: CODESYS - Digital Ausgang und Eingang aktiv

9.5.2. PiXtend eIO Analog One

Im Beispiel zeigen wir, wie man mit dem CODESYS V3.5 Modbus RTU Master am Analog One Gerät einen analogen Ausgang setzt und diesen über einen analogen Eingang wieder einliest. Hierzu benötigen wir die Funktion Codes 04 und 06. Der analoge Ausgang 0 mit dem analogen Eingang 0 muss über ein Kabel verbunden sein. Weitere Informationen zum Anschluss einer Stromversorgung und Verkabelung eines Analog One Gerätes, entnehmen Sie dem PiXtend eIO Hardware Handbuch.

9.5.2.1 Neues Projekt anlegen und SPS auswählen

Öffnen Sie in CODESYS das Datei-Menü und wählen Sie den Menüeintrag Neues Projekt...

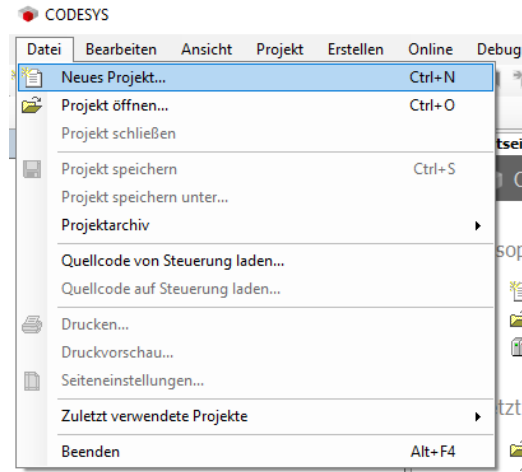


Abbildung 33: CODESYS - Datei Menü

Wählen Sie die Kategorie Projekte im Fenster Neues Projekt aus und selektieren Sie unter Vorlagen (rechts) den Eintrag Standardprojekt. Geben Sie unter Name den Projektnamen an, zum Beispiel Modbus - Analog One Beispiel. Im Anschluss wählen Sie unter Ort den Speicherplatz des Projektes aus.

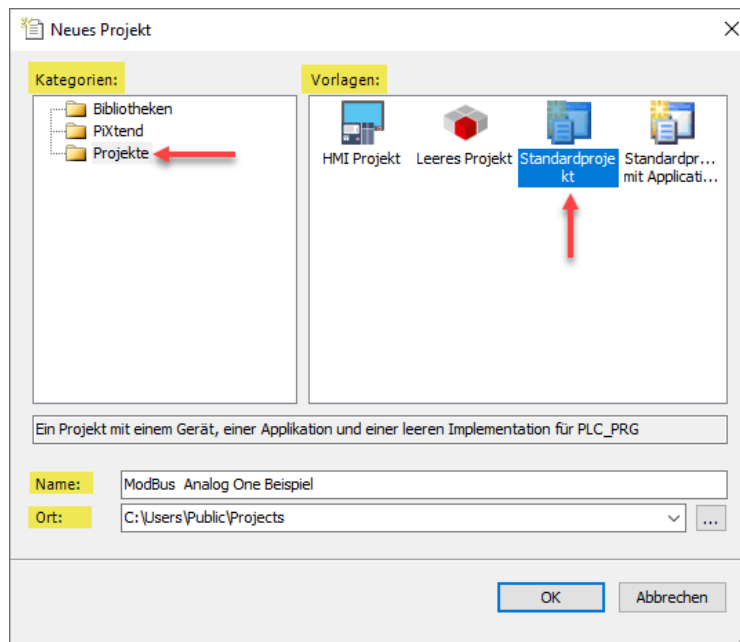


Abbildung 34: CODESYS - Neues Projekt

Im Fenster Standardprojekt wählen Sie Ihre SPS aus, PiXtend V2 Anwender wählen Raspberry Pi als Gerät aus.

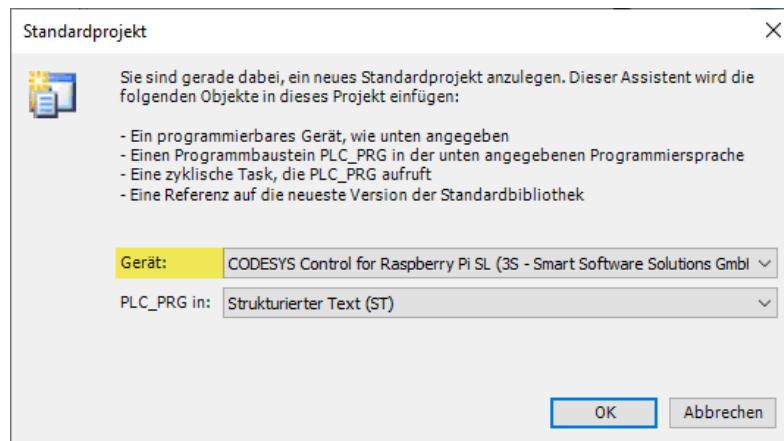


Abbildung 35: CODESYS - Standardprojekt - Geräteauswahl

Klick Sie auf OK und das neue Projekt wird erstellt.

9.5.2.2 Modbus COM Port einfügen

Nachdem das Projekt angelegt wurde, im Geräte Fenster den Eintrag Device (1) anklicken, um ihn auszuwählen. Einen Rechtsklick auf dem Eintrag ausführen und im erscheinenden Menü den Eintrag Gerät anhängen... (2) auswählen.

Im Fenster Gerät anhängen den Eintrag Feldbusse über das Plus-Zeichen öffnen. Den Eintrag Modbus öffnen und das Gerät Modbus COM Port suchen.

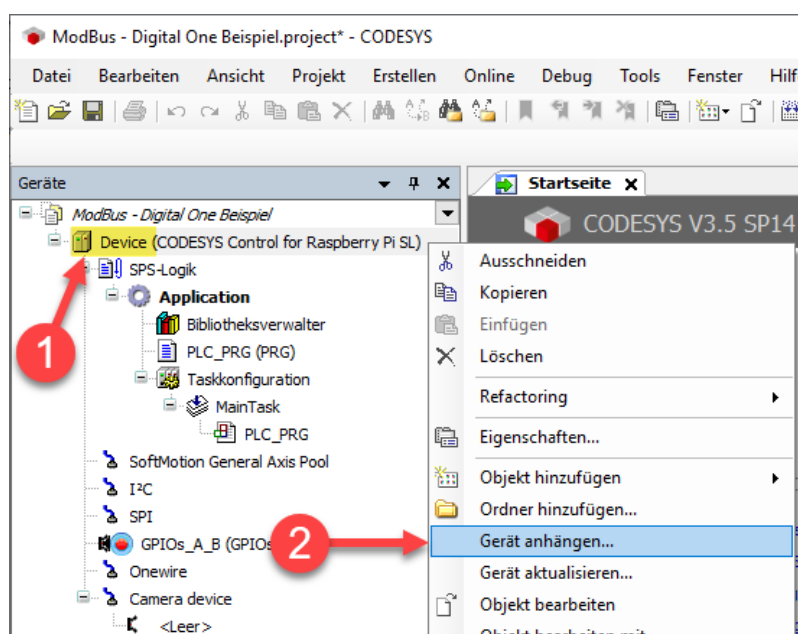


Abbildung 36: CODESYS - Gerät anhängen Menü

Den Eintrag auswählen und auf die Schaltfläche Gerät anhängen, im Fenster unten rechts, klicken.

Das Fenster schließen und im Geräte Fenster den gerade hinzugefügten Modbus_COM_Port Eintrag auswählen.

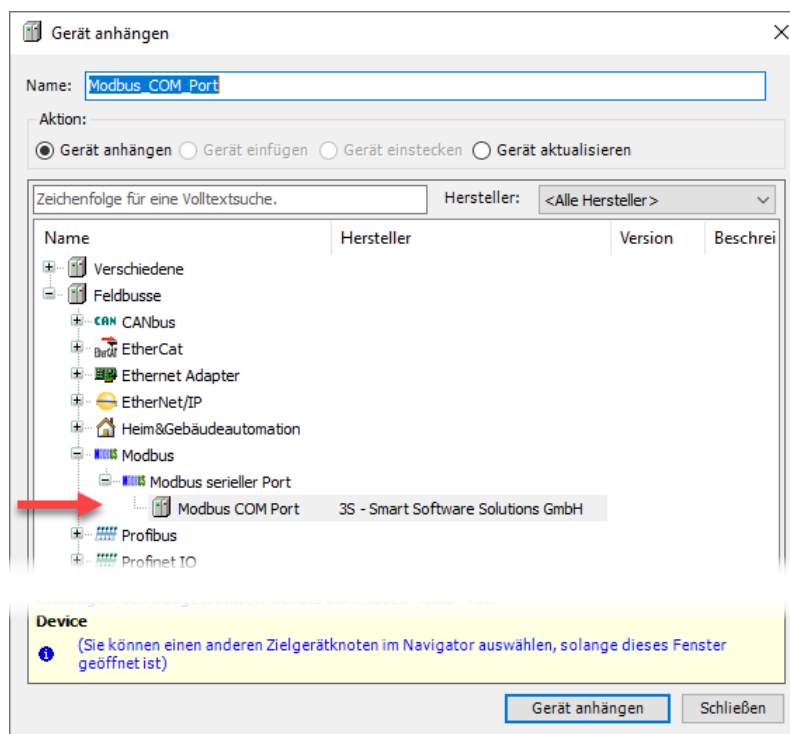


Abbildung 37: CODESYS - Gerät anhängen Fenster

Führen Sie auf dem Eintrag einen Rechtsklick aus und wählen Sie im Menü den Eintrag Gerät anhängen. Im Fenster Gerät anhängen suchen Sie nach dem Eintrag Modbus Master, COM Port und hängen Sie dieses Gerät an den vorhandenen Modbus COM Port an.

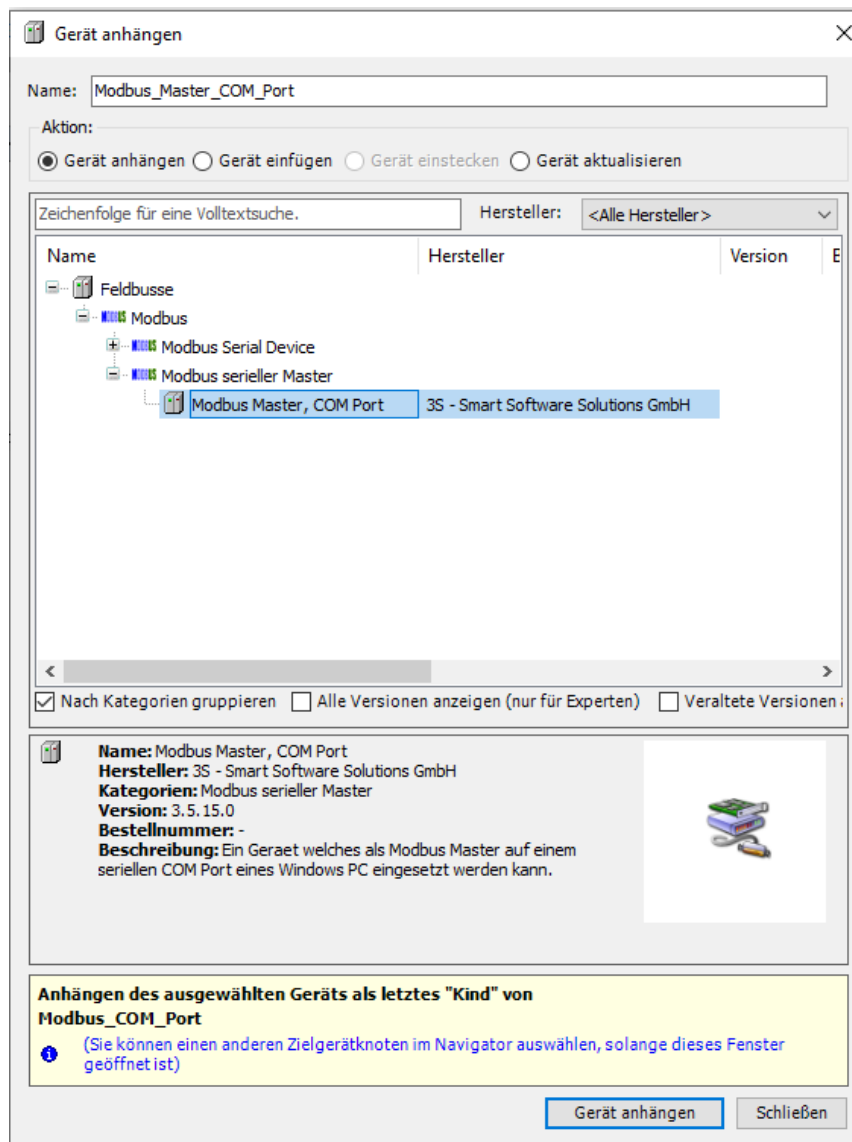


Abbildung 38: CODESYS - Modbus Master anhängen

Schließen Sie das Fenster und wählen Sie im Geräte Fenster den Eintrag Modbus_Master_COM_Port aus. Führen Sie auf dem Eintrag einen Rechtsklick aus und wählen Sie im Menü den Eintrag Gerät anhängen. Im Fenster Gerät anhängen suchen Sie nach dem Eintrag Modbus Slave, COM Port und hängen Sie dieses Gerät an den vorhandenen Modbus_Master_COM_Port an.

Schließen Sie das Fenster, alle Geräte sind nun eingefügt und vorhanden. Jetzt fehlt nur noch die Konfiguration. Im CODESYS Geräte Fenster sollten Sie jetzt eine Konfiguration wie im nachfolgenden Bild vorfinden, unabhängig davon welche SPS Sie haben.

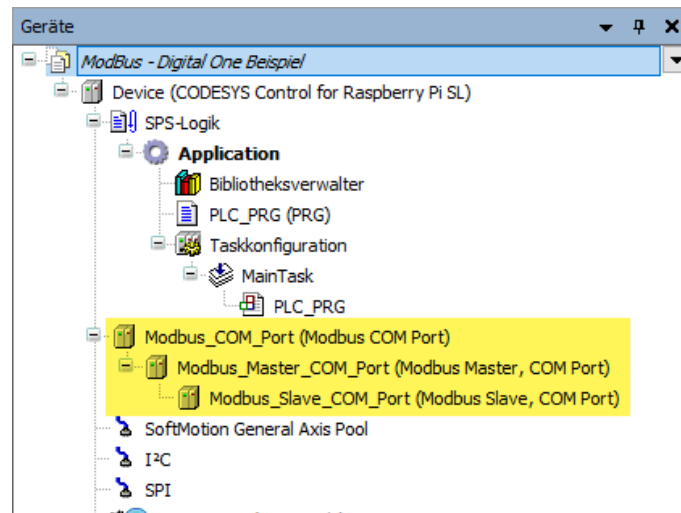


Abbildung 39: CODESYS - Alle Geräte angehängt

9.5.2.3 Modbus COM Port konfigurieren

Führen Sie einen Doppelklick auf den Geräte-Eintrag Modbus_COM_Port aus, um die Einstellungsseite für den seriellen Port zu öffnen. Klicken Sie auf den Reiter Allgemein und stellen Sie folgende Konfiguration ein:

COM-Port: 1

Baurate: 19200, Parität: Even, Daten-Bits: 8 und Stopbits: 1

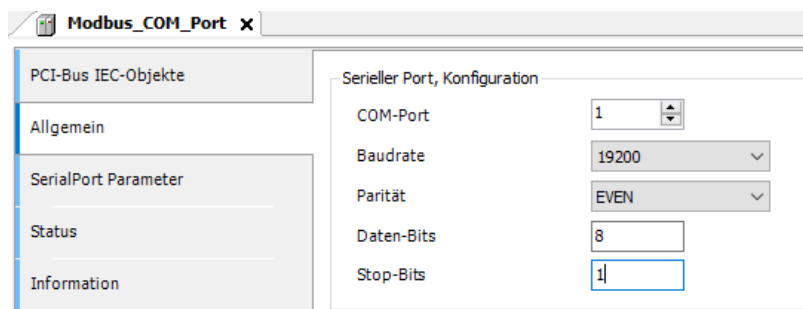


Abbildung 40: CODESYS - Modbus serielle Einstellungen

Sie verwenden kein PiXtend V2 -S-/-L- oder V1.x als SPS, dann schlagen Sie im Handbuch Ihrer SPS nach, welcher COM Port RS485 geeignet ist. Stellen Sie diese Nummer unter COM-Port, anstelle der Nummer 1, ein.

9.5.2.4 Modbus Master COM Port konfigurieren

Im Modbus Master COM Port muss eine Einstellung im Reiter Allgemein vorgenommen werden. Aktivieren Sie hier die Option automatischer Neustart Kommunikation.

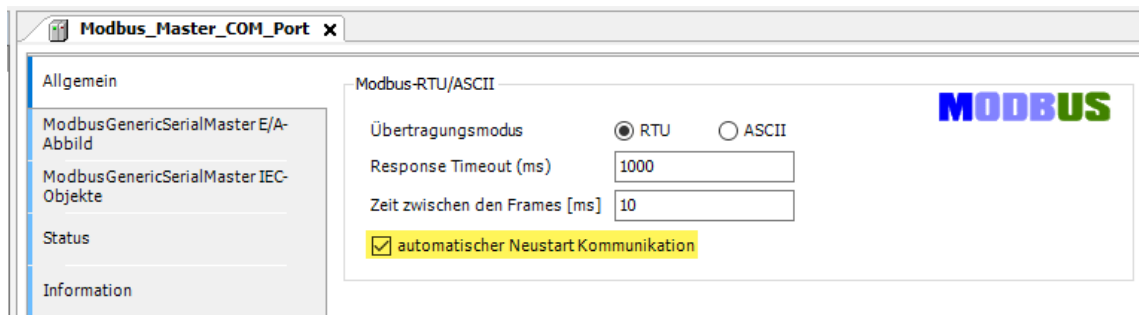


Abbildung 41: CODESYS - Modbus Master Einstellungen

9.5.2.5 Modbus Slave COM Port konfigurieren

Beim Modbus Slave muss im Reiter Allgemein eine Anpassung erfolgen, denn das Analog One Gerät hat ab Werk die Geräte-Adresse 3. Das Timeout bleibt unverändert

Die Einstellung in CODESYS entsprechen dem folgenden Bild:

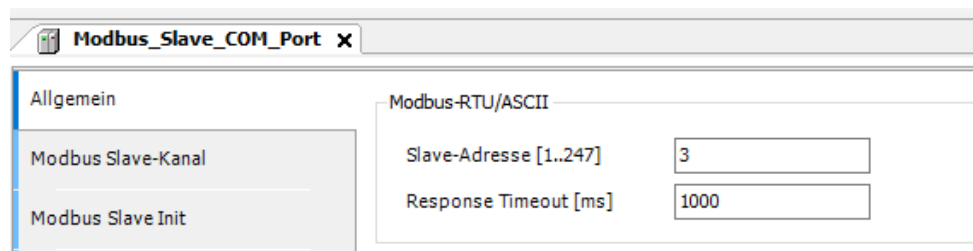


Abbildung 42: CODESYS - Modbus Slave Einstellungen

9.5.2.6 Modbus Slave-Kanal (Funktion Codes) einstellen

Wechseln Sie zum Reiter Modbus Slave-Kanal und klicken Sie unten rechts auf die Schaltfläche Kanal hinzufügen.

Vergeben Sie einen Namen, beispielsweise Analog Output 0, dieser kann später nicht mehr geändert werden. Es ist wichtig, dass Sie bei Zugriffstyp den Funktion Code 6 - Write Single Register einstellen und im Abschnitt WRITE Register einen Offset von 0 eintragen.

Alle anderen Einstellungen bleiben unverändert.

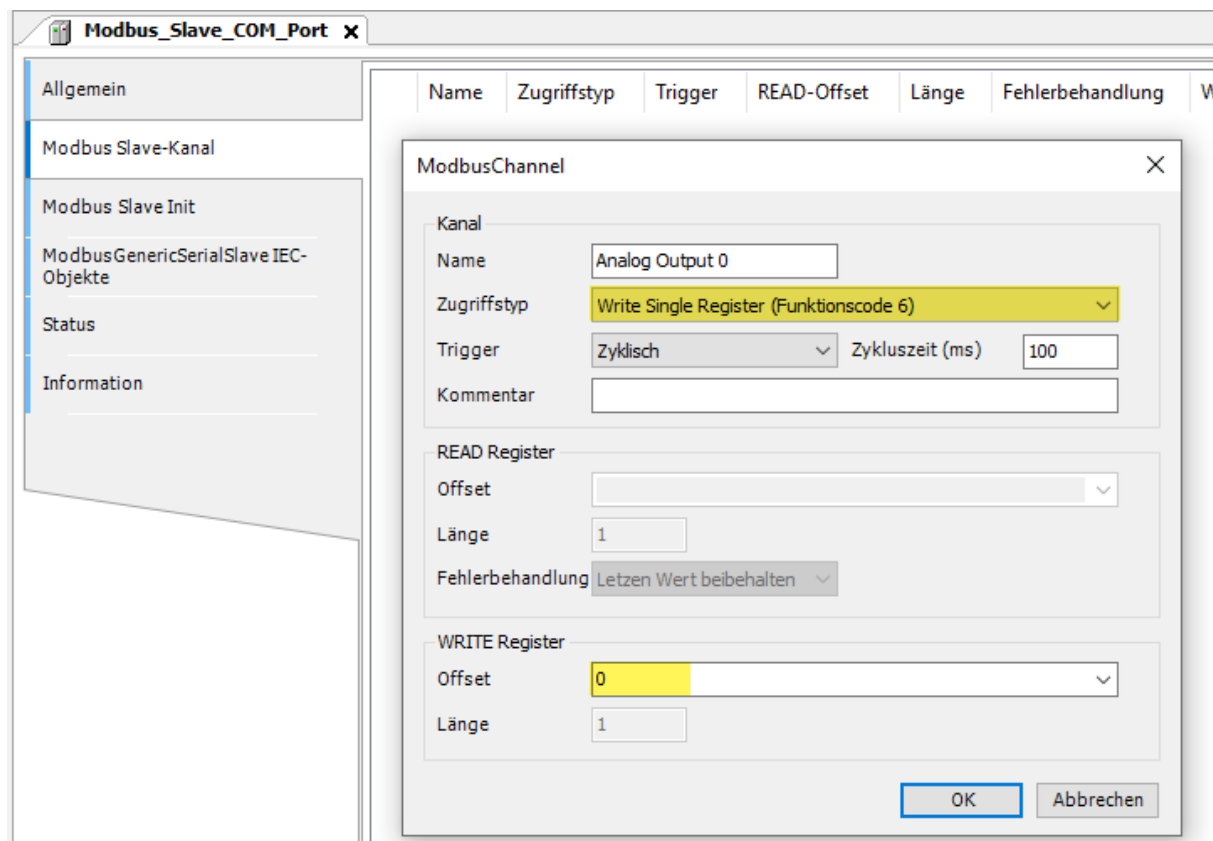


Abbildung 43: CODESYS - Modbus Kommunikationskanal anlegen (1)

Mit einem Klick auf die OK Schaltfläche fügen Sie in CODESYS einen neuen Modbus RTU Kommunikationskanal hinzu.

Klicken Sie im Anschluss nochmals auf die Schaltfläche Kanal hinzufügen... um einen weiteren Modbus RTU Kommunikationskanal einzufügen. Über diesen zweiten Kanal wird der Analogeingang 0 vom Analog One Gerät gelesen

Stellen Sie für den zweiten Kanal unter Zugriffstyp den Funktion Code 4 - Read Input Registers ein. Im Abschnitt READ Register unter Offset eine 0 und bei Länge eine 1 eintragen. Das bedeutet, dass wir nur den ersten analogen Eingang einlesen. Die Einstellung Fehlerbehandlung kann geändert werden. Auf ZERO setzen bedeutet, ist das Analog One Gerät nicht erreichbar, wird der Analogeingang 0 in CODESYS auf 0 gesetzt. Alle anderen Einstellungen bleiben unverändert.

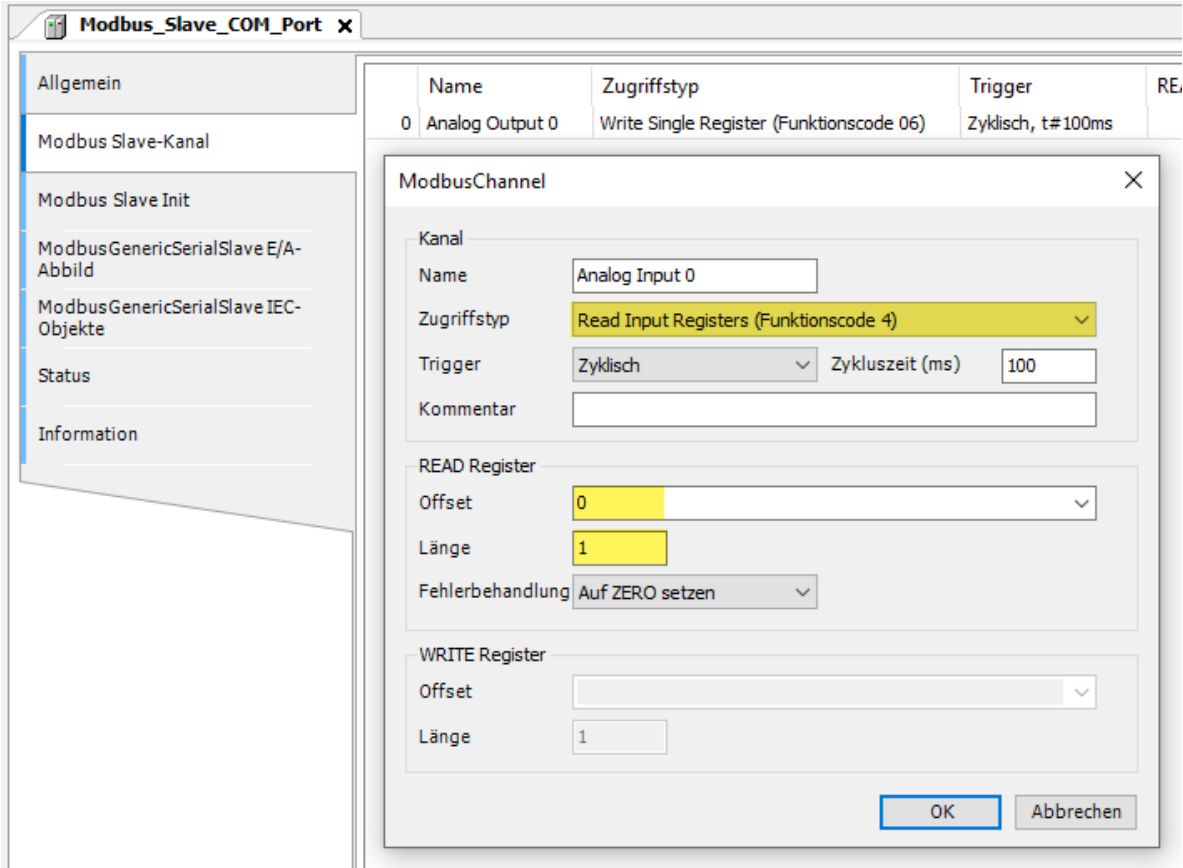


Abbildung 44: CODESYS - Modbus Kommunikationskanal anlegen (2)

Mit einem Klick auf die OK Schaltfläche fügen Sie in CODESYS einen neuen Modbus RTU Kommunikationskanal hinzu.

Die Einstellungen in CODESYS entsprechen folgendem Bild:

	Name	Zugriffstyp	Trigger	READ-Offset	Länge	Fehlerbehandlung	WRITE Offset	Länge
0	Analog Output 0	Write Single Register (Funktionscode 06)	Zyklisch, t#100ms				16#0000	1
1	Analog Input 0	Read Input Registers (Funktionscode 04)	Zyklisch, t#100ms	16#0000	1	Auf ZERO setzen		

Abbildung 45: CODESYS - Modbus Kommunikationskanalübersicht

Als nächstes wird die Variablen Aktualisierung auf die Einstellung Aktiviert 2 umgestellt. Gehen Sie dazu auf den Reiter ModbusGenericSerialSlave E/A-Abbild im Modbus Slave und ändern Sie die Einstellung unten rechts auf den Eintrag Aktiviert 2 ab.

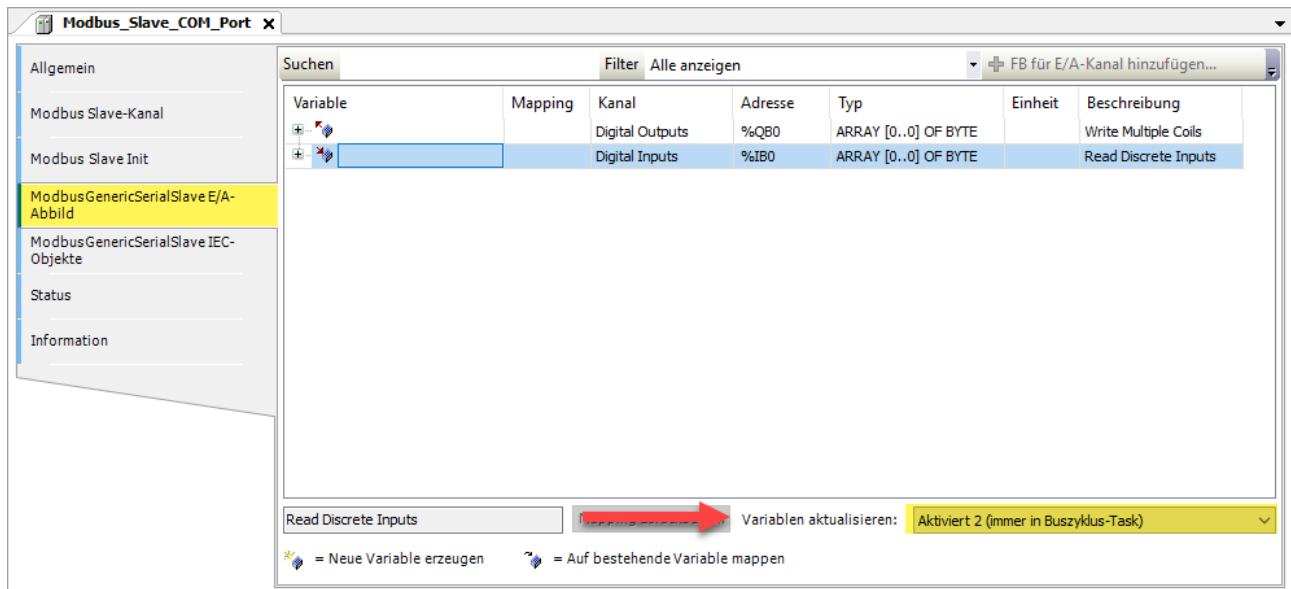


Abbildung 46: CODESYS - Variablen aktualisieren

Nun muss noch ein kleines Programm geschrieben werden, das auf PiXtend V2-L- den seriellen Anschluss von RS232 auf RS485 umschaltet. Verwenden Sie ein PiXtend V2 -S- oder V1.x mit einem RS485 Dongle oder eine andere SPS, dann überspringen Sie einfach den nächsten Schritt.

9.5.2.7 PiXtend V2 -L- Umschaltung serieller Port RS232 zu RS485

Das PiXtend V2 -L- verfügt über eine integrierte RS485 Schnittstelle, diese muss von CODESYS aktiviert werden, da die Standardeinstellung auf der RS232 Schnittstelle liegt.

Über den GPIO18 auf dem Raspberry Pi kann zwischen beiden seriellen Schnittstellentypen gewechselt werden. Richten Sie den GPIO18 des Raspberry Pi als Ausgang (Output) ein und vergeben Sie im GPIO E/A-Abbild den Variablennamen RPI_GPIO_18.

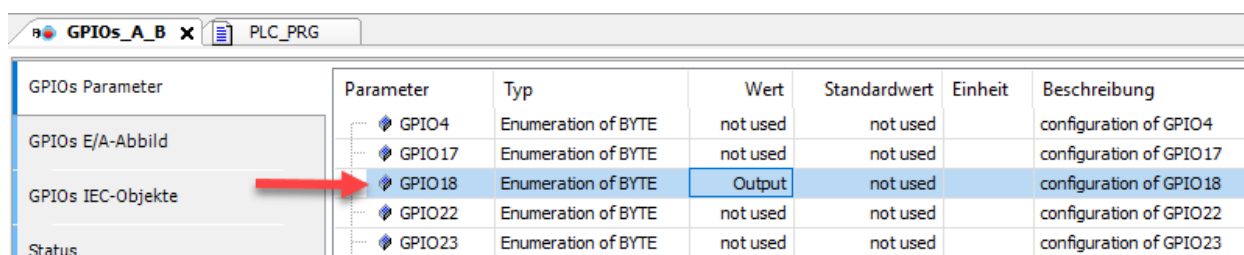


Abbildung 47: CODESYS - GPIO18 als Ausgang definieren

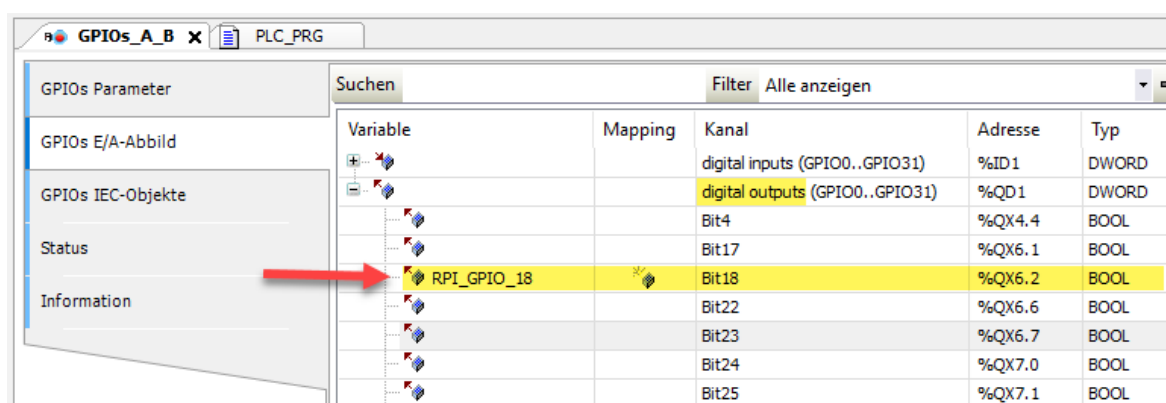
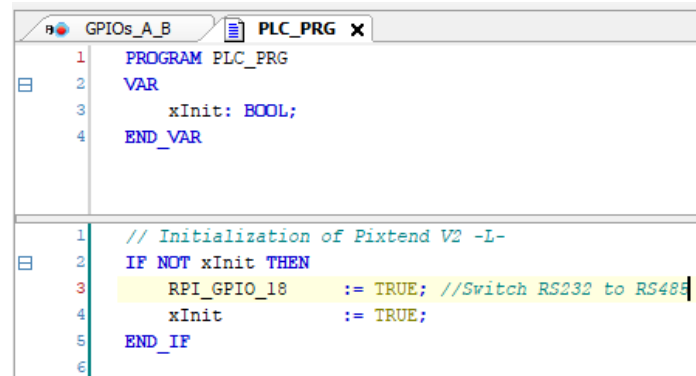


Abbildung 48: CODESYS - GPIO18 als Variable anlegen

Nun fehlt noch ein kurzes Programm, das die RPI_GPIO_18 Variable beim Start auf TRUE setzt. Eine Möglichkeit zur Umsetzung sehen Sie im nachfolgenden Bild:



```

1  PROGRAM PLC_PRG
2  VAR
3      xInit: BOOL;
4  END_VAR

1  // Initialization of PiXtend V2 -L-
2  IF NOT xInit THEN
3      RPI_GPIO_18 := TRUE; //Switch RS232 to RS485
4      xInit := TRUE;
5  END_IF
6

```

Abbildung 49: CODESYS - RS485 aktivieren

Oder Sie schreiben in die erste Zeile des PLC_PRG Bausteins folgende Anweisung:

```
RPI_GPIO_18 := TRUE; //Switch RS232 to RS485
```

Fertig, das Programm auf die Steuerung übertragen, starten und es kann beginnen!

Weitere Informationen zur seriellen Schnittstelle auf dem Raspberry Pi und PiXtend V2 und V1.x Geräten entnehmen Sie den Kapiteln 6.8 Serielle Schnittstelle vorbereiten für eigenes SD Karten Abbild, 6.9 PiXtend V1.3/V2-S- mit USB-zu-RS485 Dongle erweitern und 6.10 PiXtend V2 -L- Serielle Schnittstelle prüfen.

9.5.2.8 Programm übertragen und starten

Verbinden Sie sich mit der Steuerung, übertragen Sie das Programm und starten Sie es über die Start-Taste oder mit F5.

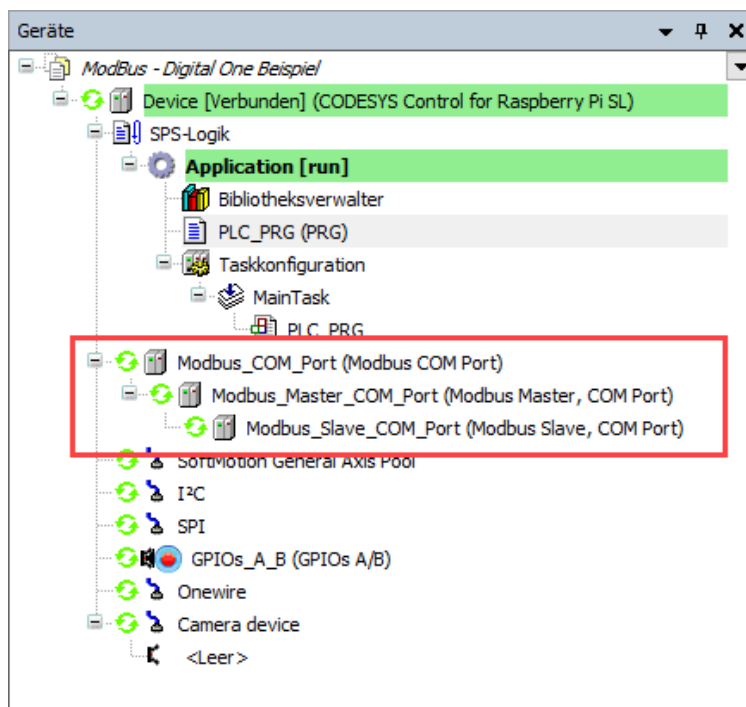


Abbildung 50: CODESYS - Modbus Bus läuft

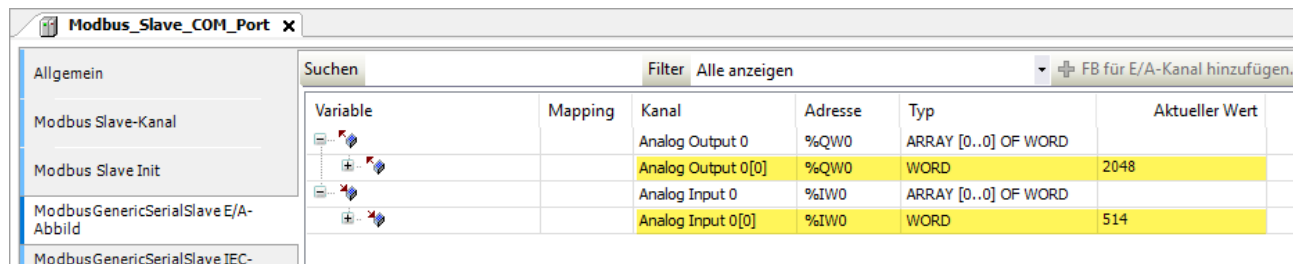
Es hat funktioniert, wenn Sie vor den drei Modbus Geräten im CODESYS Geräte Fenster grüne Kreise aus Pfeilen sehen. Ist die Farbe der Kreise orange, ist keine CODESYS V3.5 Runtime Lizenz auf dem Raspberry Pi installiert, es liegt kein Fehler des Modbus Masters vor.

Es gibt eine Einschränkung, die CODESYS Runtime läuft 2 Stunden und stoppt anschließend. Der Modbus RTU Master hingegen kann nur „30 Minuten“ getestet werden. Er stellt dann seine Arbeit ein und der Raspberry Pi muss neu gestartet werden.

Verwenden Sie eine andere Steuerung, dann konsultieren Sie Ihren Hersteller, sollten nicht alle Kreise grün sein. Gegebenenfalls benötigen Sie eine Modbus RTU Zusatzlizenz um Modbus RTU unter CODESYS auf Ihrer SPS nutzen können.

9.5.2.9 Ausgang auf 5 Volt setzen und Eingang lesen

Das Programm und der Modbus Bus arbeiten. Jetzt setzen Sie den Analogausgang 0 auf 2048, das entspricht in etwa einer Spannung von 5 Volt und können diesen Wert am Analogeingang 0 wieder einlesen. Dafür muss der Analogausgang 0 mit dem Analogeingang 0 am Analog One Gerät verbunden sein. Kehren Sie zum ModbusGenericSerialSlave E/A-Abbild Reiter im Modbus_Slave_COM_Port Gerät zurück und klappen Sie beide Kanäle auf. Haben Sie den Analogausgang 0 auf 2048 gesetzt, dann sollte am Analogeingang 0 ein Wert von etwa 512, das entspricht ungefähr 5 Volt, ersichtlich werden.



Suchen	Filter	Alle anzeigen	+ FB für E/A-Kanal hinzufügen.			
Variable	Mapping	Kanal	Adresse	Typ	Aktueller Wert	
		Analog Output 0	%QW0	ARRAY [0..0] OF WORD		
		Analog Output 0[0]	%QW0	WORD	2048	
		Analog Input 0	%IW0	ARRAY [0..0] OF WORD		
		Analog Input 0[0]	%IW0	WORD	514	

Abbildung 51: CODESYS - Modbus Slave Analogausgang 0 und Analogeingang 0

Besuchen Sie den Download-Bereich auf unserer Homepage für weitere CODESYS-Beispiele sowie Beispiele für anderen Programmiersprachen und Werkzeuge.

9.6. Beispiel Python

Dieses Kapitel zeigt, wie man mit der Programmiersprache Python, dem pyModbus Modul mit dem Digital One Geräte sowie dem Analog One Gerät per Modbus RTU Protokoll sprechen kann. Stellen Sie vor dem Start sicher, dass die Python Packages RPi.GPIO, in der Version 0.6.5 oder später und pyModbus, in der Version 2.2.0 oder später, installiert sind. Deaktivieren Sie über raspi-config die Login-Shell auf der ttyAMA0 bzw. serial0 Schnittstelle. Das installierte Module Python 2.7 lässt sich mit dem Befehl pip freeze prüfen. Verwenden Sie Python 3 dann verwenden Sie pip3 freeze. Sind die Python Module oder Packages nicht vorhanden, verwenden Sie nachfolgenden Befehle, um die benötigten Funktionen zu installieren.

```
pyModbus installieren: sudo pip install -U pymodbus
                        sudo pip3 install -U pymodbus
RPi.GPIO installieren: z.B. so: sudo apt-get install rpi.gpio
```

9.6.1. PiXtend eIO Digital One

Dieses Beispiel zeigt, wie man mit der Programmiersprache Python am Digital One Gerät verschiedene digitale Ausgänge setzen und über die digitalen Eingänge wieder einlesen kann. Wir schreiben die gewünschte Bitfolge auf die Digital One Coils und lesen diese über die Discrete Inputs wieder zurück. Hierzu benötigen wir die Funktion Codes 02 und 15. Der Digitalausgang 0 muss mit dem Digitaleingang 0 über ein Kabel verbunden sein. Weitere Informationen zum Anschluss einer Stromversorgung und Verkabelung eines Digital One, schauen Sie ins PiXtend eIO Hardwarehandbuch.

Arbeiten Sie mit der Programmiersprache Python und verwenden Sie ein PiXtend V2 -L-, dann stehen Ihnen verschiedene Möglichkeiten offen, mit den PiXtend eIO Geräte zu kommunizieren. Im SD-Karten Basis Image ab Version 2.1.6.0 verwenden wir die vollwertige UART (PL011, serial0 oder ttyAMA0 genannt) des Raspberry Pi. Nur diese Schnittstelle erlaubt die Verwendung des Paritätsbits.

Verwenden Sie einen RS485 USB-Dongle, wird Ihr serielles Gerät vermutlich ttyUSB0 heißen. Arbeiten Sie mit der seriellen Schnittstelle auf der Steuerung eines anderen Herstellers, entnehmen Sie die Informationen dem Geräte-Handbuch. Entsprechend der verwendeten Schnittstelle kommentieren Sie die entsprechenden Code-Zeilen ein oder aus beziehungsweise ändern Sie diese entsprechen ab

Das nachfolgende Beispiel basiert auf der Annahme, dass Sie mit einem PiXtend V2 -L-, mit der vollwertigen UART (ttyAMA0) arbeiten und das Basis Image in Version 2.1.6.0 oder später verwenden. Das Digital One Gerät steht auf Werkseinstellung, Geräte-Adresse 1, CONFIG Schalter 1-6 alle auf OFF. Diese Einstellungen entsprechen 19200 Baud, 8 Datenbits, Parität „Even“ und 1 Stoppbit.

Datei: Digital_One_pyModbus_Demo.py

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
import time

# Activate RS485 chip: Only needed for PiXtend V2 -L-
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.output(18, GPIO.HIGH)

# Serial configuration for RPi miniUART ttyS0, does not support parity
#client= ModbusClient(method = "rtu", port="/dev/ttyS0",stopbits = 1, bytesize = 8, parity = 'N', baudrate= 19200)

# Serial configuration for RPi full UART ttyAMA0
client= ModbusClient(method = "rtu", port="/dev/ttyAMA0",stopbits = 1, bytesize = 8, parity = 'E', baudrate= 19200)

# Serial configuration for USB-to-RS485 dongle
#client= ModbusClient(method = "rtu", port="/dev/ttyUSB0",stopbits = 1, bytesize = 8, parity = 'E', baudrate= 19200)

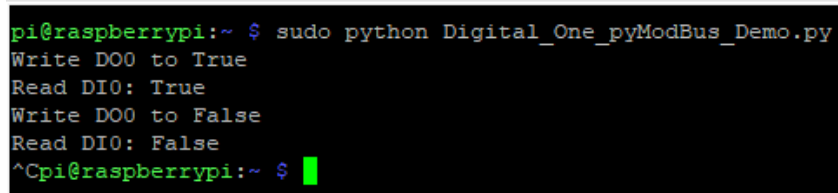
connection = client.connect()

while True:
    try:
        print("Write D00 to True")
        result= client.write_coil(0x00,1, unit= 0x01)
        time.sleep(1)
        result= client.read_discrete_inputs(0x00, count=1, unit= 0x01)
        print("Read DI0: " + str(result.bits[0]))
        time.sleep(1)
        print("Write D00 to False")
        result= client.write_coil(0x00,0, unit= 0x01)
        time.sleep(1)
        result= client.read_discrete_inputs(0x00, count=1, unit= 0x01)
        print("Read DI0: " + str(result.bits[0]))
        time.sleep(1)
    except KeyboardInterrupt:
        #Press Ctrl+C to exit
        client.close()
        time.sleep(0.25)
        client = None
        break
```

Das Programm Python arbeitet am besten mit root-Rechten, somit hat Python auf die serielle Schnittstelle Zugriff. Starten Sie das Programm wie folgt:

- Python 2.7: `sudo python Digital_One_pyModbus_Demo.py`
- Python 3: `sudo python3 Digital_One_pyModbus_Demo.py`

Ein Durchlauf der Hauptprogrammschleife stellt sich in der Konsole wie folgt dar:



```
pi@raspberrypi:~ $ sudo python Digital_One_pyModBus_Demo.py
Write DO0 to True
Read DI0: True
Write DO0 to False
Read DI0: False
^Cpi@raspberrypi:~ $
```

Abbildung 52: Digital One - Python Demo Programm

Zuerst wird der Digitalausgang 0 auf True gesetzt, nach kurzer Zeit wird der Digitaleingang 0 gelesen und liefert True zurück, es liegt ein HIGH-Pegel an. Im Anschluss wird der Digitalausgang 0 auf False gesetzt, nach kurzer Zeit wird der Digitaleingang 0 erneut gelesen und liefert False zurück. Es liegt somit kein HIGH-Pegel mehr an.

Besuchen Sie den Download-Bereich auf unserer Homepage für weitere Python-Beispiele sowie Beispiele für andere Programmiersprachen und Werkzeuge.

9.6.2. PiXtend eIO Analog One

Dieses Beispiel zeigt, wie man mit der Programmiersprache Python und dem pyModbus Modul am Analog One Gerät einen analogen Ausgang setzt und diesen über einen analogen Eingang wieder einliest. Der Analogausgang 0 wird mit dem Analogeingang 0 über ein Kabel verbunden. Weitere Informationen zum Anschluss einer Stromversorgung und Verkabelung eines Analog One, entnehmen Sie dem PiXtend eIO Hardware Handbuch.

Arbeiten Sie mit der Programmiersprache Python und verwenden Sie ein PiXtend V2 -L-, dann stehen Ihnen verschiedene Möglichkeiten offen, mit den PiXtend eIO Geräte zu kommunizieren. Im SD-Karten Basis Image ab Version 2.1.6.0 verwenden wir die vollwertige UART (PL011, auch serial0 oder ttyAMA0 genannt) des Raspberry Pi. Nur diese Schnittstelle erlaubt die Verwendung des Paritätsbits.

Verwenden Sie einen RS485 USB-Dongle, wird Ihr serielles Gerät vermutlich ttyUSB0 heißen. Arbeiten Sie mit der seriellen Schnittstelle auf der Steuerung eines anderen Herstellers, entnehmen Sie die Informationen dem Geräte-Handbuch. Entsprechend der verwendeten Schnittstelle kommentieren Sie die entsprechenden Code-Zeilen ein oder aus beziehungsweise ändern Sie diese entsprechen ab

Das nachfolgende Beispiel basiert auf der Annahme, dass Sie mit einem PiXtend V2 -L-, mit der vollwertigen UART (ttyAMA0) arbeiten und das Basis Image in Version 2.1.6.0 oder später verwenden. Das Analog One Gerät steht Werkseinstellung, Geräte-Adresse 3, CONFIG Schalter 1-6 alle auf OFF. Diese Einstellungen entsprechen 19200 Baud, 8 Datenbits, Parität „Even“ und 1 Stoppbit.

```
Analog_One_pyModbus_Demo.py
#!/usr/bin/env python
import RPi.GPIO as GPIO
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
import time

# Activate RS485: Only needed for PiXtend V2 -L-
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.output(18, GPIO.HIGH)

# Serial configuration for RPi miniUART ttyS0, no parity support
#client= ModbusClient(method = "rtu", port="/dev/ttyS0",stopbits = 1, bytesize = 8, parity = 'N', baudrate= 19200)

# Serial configuration for RPi full UART ttyAMA0
client= ModbusClient(method = "rtu", port="/dev/ttyAMA0",stopbits = 1, bytesize = 8, parity = 'E', baudrate= 19200)

# Serial configuration for USB-to-RS485 dongle
#client= ModbusClient(method = "rtu", port="/dev/ttyUSB0",stopbits = 1, bytesize = 8, parity = 'E', baudrate= 19200)

connection = client.connect()

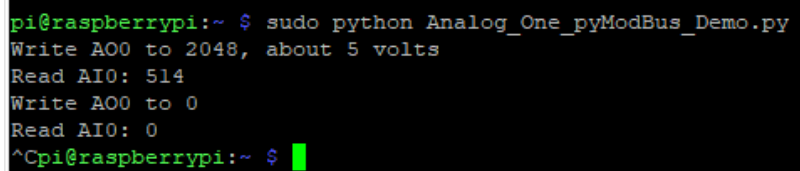
while True:
    try:
        print("Write A00 to 2048, about 5 volts")
        result= client.write_register(0x00, 2048, unit= 0x03)
        time.sleep(1)
        result= client.read_input_registers(0x00, 1, unit= 0x03)
```

```
print("Read AIO: " + str(result.registers[0]))
time.sleep(1)
print("Write A00 to 0")
result= client.write_register(0x00,0, unit= 0x03)
time.sleep(1)
result= client.read_input_registers(0x00, 1, unit= 0x03)
print("Read AIO: " + str(result.registers[0]))
time.sleep(1)
except KeyboardInterrupt:
    client.close()
    time.sleep(0.25)
    client = None
    break
```

Das Programm Python arbeitet am besten mit root-Rechten, so dass Python auf die serielle Schnittstelle Zugriff hat. Starten Sie das Programm wie folgt:

- Python 2.7: `sudo python Analog_One_pyModbus_Demo.py`
- Python 3: `sudo python3 Analog_One_pyModbus_Demo.py`

Ein Durchlauf der Hauptprogrammschleife stellt sich in der Konsole wie folgt dar:



```
pi@raspberrypi:~ $ sudo python Analog_One_pyModbus_Demo.py
Write AO0 to 2048, about 5 volts
Read AI0: 514
Write AO0 to 0
Read AI0: 0
^Cpi@raspberrypi:~ $
```

Abbildung 53: Analog One - Python Demo Programm

Zuerst wird der Analogausgang 0 auf „2048“ (etwa 5 Volt) gesetzt, nach kurzer Zeit wird der Analogeingang 0 gelesen und liefert „514“ zurück, ebenfalls etwa 5 Volt. Im Anschluss wird der Analogausgang 0 auf „0“ gesetzt, nach kurzer Zeit wird der Analogeingang 0 erneut gelesen und liefert jetzt „0“ zurück. Es liegt somit keine Spannung mehr an.

Besuchen Sie den Download-Bereich auf unserer Homepage für weitere Python-Beispiele sowie Beispiele für andere Programmiersprachen und Werkzeuge.

9.7. Beispiel C

9.7.1. PiXtend eIO Digital One

Programmbeispiele für die Programmiersprache C finden Sie im Download-Bereich auf unserer Homepage. Um die Beispiele zu verwenden muss die Bibliothek libModbus installiert sein. Ergänzend wird wiringPi benötigt um auf einem PiXtend V2 -L- den GPIO18 schalten zu können.

Führen Sie folgende Schritte zur Installation von libModbus aus:

- `sudo apt update`
- `sudo apt install libmodbus-dev`

Die Bibliothek wiringPi finden Sie in unserem Download-Bereich zusammen mit einer Installationsanleitung. Im PiXtend V2 Software Handbuch, im Kapitel pxdev – Linux Tools & Library finden Sie weitere Informationen. Nutzen Sie ein Kontron Electronics SD-Karten Abbild, so ist wiringPi bereits vorinstalliert.

Das System ist vorbereitet und Sie können das Mdbus RTU Protokoll in C nutzen. Haben Sie die Beispiele von unserer Homepage geladen, können Sie das *Blinky* Programmbeispiel wie folgt mit dem gcc Compiler übersetzen:

- `gcc -I /usr/include/modbus PiXtend_eIO_Digital_One_ModBus_Demo_Blinky.c -o PiXtend_eIO_Digital_One_ModBus_Demo_Blinky -L/usr/lib/modbus -lmodbus -lwiringPi`

Das Programm wird anschließend mit folgendem Befehl gestartet:

- `sudo ./PiXtend_eIO_Digital_One_ModBus_Demo_Blinky`

9.7.2. PiXtend eIO Analog One

Die Vorbereitungsschritte für die Analog One Programmbeispiele sind gleich wie beim Digital One, siehe Abschnitt 9.7.1.

Die Anweisung zum Übersetzen des Analog One Beispielprogramms.

PiXtend_eIO_Analog_One_ModBus_Demo_A00_to_AIO unterscheidet sich gegenüber der vom Digital One.

Beispiel übersetzen:

- `gcc -I /usr/include/modbus PiXtend_eIO_Analog_One_ModBus_Demo_A00_to_AIO.c -o PiXtend_eIO_Analog_One_ModBus_Demo_A00_to_AIO -L/usr/lib/modbus -lmodbus -lwiringPi`

Das Programm wird anschließend mit folgendem Befehl gestartet:

- `sudo ./PiXtend_eIO_Analog_One_ModBus_Demo_A00_to_AIO`

10. PiXtend eIO Protokoll

10.1. Einführung

Das PiXtend eIO Protokoll ist ein einfaches Klartextprotokoll der Firma Kontron Electronics GmbH, das dafür ausgelegt ist über einen seriellen Bus übertragen zu werden.

Der Anwender kann einfache Klartextanweisungen in ein serielles Programm oder ein Terminal Programm eingeben um dem Gerät einen Befehl zu schicken. Auf jeden Befehl erhält der Anwender eine Rückantwort.

Verschiedene Befehle stehen zur Verfügung, um die digitalen und analogen Ein- und Ausgänge, die jedes Gerät bietet, einzulesen und zu setzen. Jedes Gerät, das sich am Bus befindet bzw. angeschlossen wurde, muss über eine eindeutige Geräte-Adresse verfügen. Der Adressbereich reicht von 0 bis 255, wobei nur eine maximale Anzahl von 32 Geräten an einem RS485 Bus erlaubt sind. Bitte beachten Sie dies bei Ihrer Planung.

Stellen Sie zu sicher, dass der Mode-Schalter im DIP-Block 2 - CONFIG in der Stellung ON, das heißt oben steht.

10.2. Übersicht

Allgemeines - für alle eIO-Geräte

- Ablauf der Kommunikation
- Protokollaufbau

PiXtend eIO Digital One

- Übersicht
- Befehle - Basic
- Befehle - Advanced
- Beispiele - Basic
- Beispiele - Advanced

PiXtend eIO Analog One

- Übersicht
- Befehle - Basic
- Befehle - Advanced
- Beispiele - Basic
- Beispiele - Advanced

10.3. Ablauf der Kommunikation

Das PiXtend eIO Protokoll basiert auf einer Master - Slave Kommunikation, ähnlich dem Modbus RTU Protokoll. Der Master sendet eine Nachricht aus und der betroffene Slave antwortet auf diese Nachricht mit der vom Master geforderten Information. Während der Master auf die Antwort des Slaves wartet, läuft zeitgleich ein Timer der die Kommunikation mit dem Slave überwacht. Antwortet der Slave nicht innerhalb einer bestimmten Zeitspanne, kann der Master die Nachricht erneut senden oder dem Anwender einen Kommunikationsfehler anzeigen. Eine derartige Timeout Überwachung wird z.B. bei Modbus RTU eingesetzt.

Im Gegensatz zum Modbus RTU Protokoll, hier antwortet der Slave so schnell er kann, gibt es beim PiXtend eIO Protokoll eine 10 ms Antwortverzögerung. Aus diesem Grund können PiXtend eIO Geräte mit einem PiXtend V1 Gerät genutzt werden. Die Umschaltung der Sende- und Empfangsrichtung des RS485 Transceivers erfolgt mittels des Raspberry Pi GPIO 22. Es wird nicht automatisch umgestellt, wie beim PiXtend V2 -L- oder unter Verwendung eines USB-zu-RS485 Dongle.

Bei einer seriellen Kommunikation darf immer nur ein Bus-Teilnehmer kommunizieren. Senden 2 oder mehr Teilnehmer gleichzeitig, ist keine geordnete Kommunikation möglich. Beachten Sie, dass nur mit einem Gerät zu einer Zeit gesprochen werden kann. Ist die Kommunikation, der Datenaustausch mit einem Gerät beendet, kann mit dem nächsten Gerät kommuniziert werden. Verwenden Sie immer sehr lange Wartezeiten bzw. Timeouts, um auf die Antwort eines Slaves zu warten, wir empfehlen 500 ms oder 1 Sekunde. Ist die Antwort des Slaves nach über 1 Sekunde Wartezeit nicht eingetroffen, kann davon ausgehen werden, dass etwas nicht in Ordnung ist.

Ablauf der Kommunikation zwischen Master und Slave in grafischer Form:

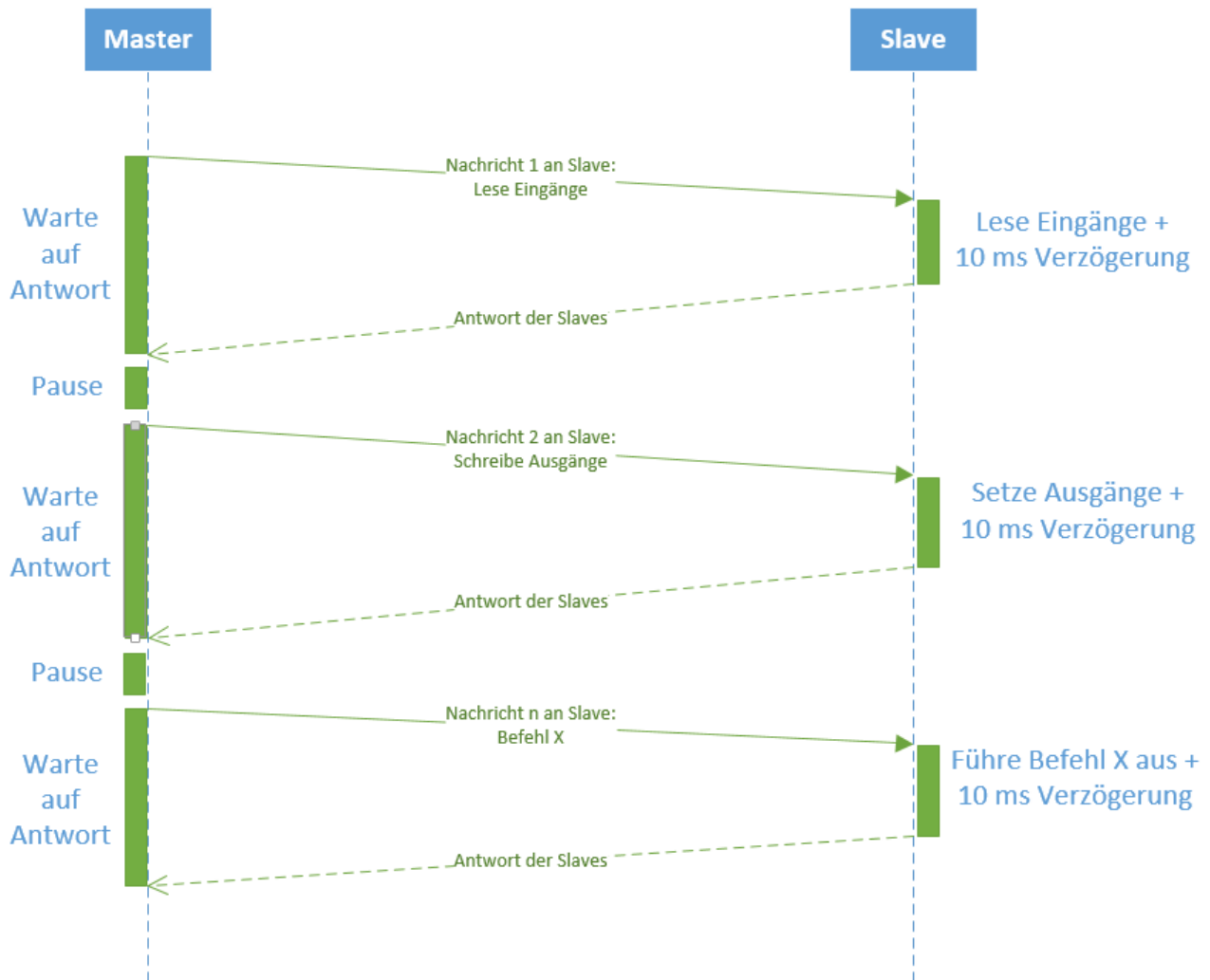


Abbildung 54: PiXtend eIO Protokoll - Kommunikation - Ablaufplan

Hinweise zum Diagramm:

- Master - Warte auf Antwort: bis zu 1000 ms
- Master - Pause für Baudraten unter 9600: ≥ 4 ms
- Master - Pause für Baudraten ab 9600: ≥ 2 ms

10.4. Protokollaufbau

Das PiXtend eIO Protokoll basiert auf einfachen ASCII Zeichen, es hat grundsätzlich folgenden Aufbau und weist pro Gerät immer eine feste Länge auf.

Protokollaufbau:

	Start-Zeichen	Adresse	Trenner	Kommando	Trenner	E/A-Nummer	Trenner	Wert	Ende-Zeichen
•	Start-Zeichen	1 Zeichen, Raute, ASCII 35 _{Dez}						#	
•	Ende-Zeichen	1 Zeichen, Stern, ASCII 42 _{Dez}						*	
•	Trenner	1 Zeichen: Doppelpunkt, ASCII 58 _{Dez}						:	
•	Adresse	3 Zeichen, nur Zahlen, ASCII 48 _{Dez} – 57 _{Dez}						000 bis 999	
•	Kommando	3 Zeichen, nur Zahlen, ASCII 48 _{Dez} – 57 _{Dez}						000 bis 999	
•	E/A-Nummer	2 Zeichen, nur Zahlen, ASCII 48 _{Dez} – 57 _{Dez}						00 bis 99	
•	Wert Länge								
◦	Digital One	3 Zeichen, nur Zahlen, ASCII 48 _{Dez} – 57 _{Dez}						000 bis 999	
◦	Analog One	4 Zeichen, nur Zahlen, ASCII 48 _{Dez} – 57 _{Dez}						0000 bis 9999	
									_{Dez} = Dezimal

Erhält ein Gerät einen Befehl oder eine gültige Nachricht, wird immer eine Antwort zurückgeschickt. Diese Antwort ist nicht immer gleich, sondern unterscheidet sich ob man einen Ausgang setzt, einen Eingang einliest oder einen Status abfragt.

Wird ein Ausgang gesetzt, dann antwortet das Gerät mit dem Befehl bzw. schickt eine Kopie der empfangen Nachricht zurück. Somit lässt sich überprüfen, ob die Nachricht für das betreffende Gerät korrekt angekommen ist oder ob ein Übertragungsfehler vorliegt. Das Gerät nimmt keine Veränderung an den Daten vor.

Soll ein Eingang gelesen werden, erhält man als Antwort alles zurück bis auf den Teil WERT, hier steht der tatsächliche Wert des digital bzw. analog Eingangs.

Es handelt sich um ein Protokoll fester Länge. Bezogen auf das jeweilige Gerät, müssen bei allen Zahlen die Nullen immer angeführt werden, sofern die entsprechende Zahl kleiner 1000 bzw. kleiner 100 bzw. kleiner 10 ist. Die genaue Länge des Protokolls oder einer Nachricht und dessen konkrete Zusammensetzung für ein Gerät, ist im Kapitel des jeweiligen Geräts nachzuschlagen.

Beispiele zu führenden Nullen:

- Zweistellige Werte:
 - 1 → 01
 - 12 → 12
- Dreistellige Werte:
 - 1 → 001
 - 10 → 010
 - 100 → 100
- Vierstellige Werte:
 - 1 → 0001
 - 12 → 0012
 - 120 → 0120
 - 1200 → 1200

Hinweis zu den nachfolgenden Kapiteln:

Das PiXtend eIO Protokoll bietet dem Anwender viele Möglichkeiten die unterschiedlichsten Funktionen der PiXtend eIO Geräte einzustellen, zu verändern sowie ein- und auszuschalten. Zuzüglich der Möglichkeit analoge und digitale Ein- und Ausgänge lesen und schreiben zu können.

Aus diesem Grund ist die Beschreibung der PiXtend eIO Protokoll Befehle in zwei Hauptteile untergliedert, der Basic-Teil und der Advanced-Teil.

Im Basic-Teil finden Sie alle Informationen zum Lesen und Schreiben der analogen und digitalen Ein- und Ausgänge der PiXtend eIO Geräte.

Im Advanced-Teil alle Befehle und Angaben wie sich die erweiterten Funktionen beispielsweise Watchdog-Timer oder Zähler, konfigurieren und verwenden lassen. Dieser Abschnitt richtet sich in erster Linie an fortgeschrittene Anwender, die sich bereits mit PiXtend eIO Geräten auskennen und die Grundfunktionen dieser Geräte beherrschen.

10.5. PiXtend eIO Digital One

Beim Digital One Gerät hat eine Nachricht immer eine Länge von 16 Zeichen (Bytes).

Das Digital One Gerät hat die Gerätekennung: 221 (Hex: DD)

Anzahl der digitalen Eingänge: 8 (0 - 7)

Anzahl der digitalen Ausgänge: 8 (0 - 7)

Beispiel-Nachricht für ein Digital One Gerät:

Beschreibung	START	ADR	TREN	KOM	TREN	E/A- NUM	TREN	WERT	ENDE
ASCII- Zeichen	#	012	:	003	:	02	:	001	*
Byte- Nummer	1	2-3-4	5	6-7-8	9	10-11	12	13-14-15	16

(START = Start, ADR = Adresse, TREN = Trenner, KOM = Kommando, E/A NUM = E/A Nummer, WERT = Wert, ENDE = Ende)

10.5.1. Übersicht unterstützter Befehle - Basic

Die nachfolgenden Befehle können an ein Digital One Gerät per Nachricht geschickt werden, um dessen digitale Eingänge zu lesen oder die digitalen Ausgänge zu setzen.

Kommando	Nummer	Werte-Bereich	Bemerkung
Read Digital Input	1	0/1	Einen digitalen Eingang lesen
Read Digital Input All	2	0 ... 255	Alle digitalen Eingänge lesen. Jedes Bit im Byte entspricht einem Eingang. E/A-NUM wird bei diesem Befehl ignoriert.
Write Digital Output	3	0/1	Einen digitalen Ausgang schreiben/setzen
Write Digital Output All	4	0 ... 255	Alle digitalen Ausgänge schreiben/setzen. Jedes Bit im Byte entspricht einem Ausgang. E/A-NUM wird bei diesem Befehl ignoriert.
Read Device ID	300	0 ... 255	Gerätekennung abfragen, 8 Bit Wert, 1 Byte.
Read Error Register	400	0 ... 255	Fehlerregister abfragen, Aufschlüsselung der Bits im Register siehe Kapitel 10.5.2 Übersicht der Bits im Error Register, 16 Bit Wert E/A-NUM: 0 = Low Byte, 1 = High Byte
Set Config - Quit Errors	401	0	Alle Fehler im Fehlerregister ablöschen, E/A-NUM = 00 und WERT = 000
Read Firmware Version	500	0 ... 999	Version der Geräte Firmware auslesen im Format XXX, z.B. 101 = 1.01, 102 = 1.02

Tabelle 46: Digital One: Übersicht unterstützter Befehle – Basic

Bei den Kommandos 2 (Read Digital Input All) und 4 (Write Digital Output All) kann eine 3-stellige Zahl, 000 bis 255, übertragen werden. Diese Zahl hat auf einer Steuerung/PC den Datentyp Byte (8 Bits). Jedes Bit in diesem Byte entspricht dabei, von LSB nach MSB (Intel Byte Order), einem Eingang bzw. einem Ausgang. Bit 0 = Eingang 0/Ausgang 0 und Bit 7 = Eingang 7/Ausgang 7. Der Protokoll-Teil E/A-NUM wird ignoriert.

Soll ein Eingang gelesen bzw. ein Ausgang gesetzt werden (einzeln, Befehl 1 und 2), sind nur die Zahlen 0 = Aus (000) und 1 = An (001) erlaubt. Hier müssen immer zwei 0 (Nullen) vorangestellt werden.

Das Fehlerregister ist ein 16 Bit Wert, der mit dem PiXtend eIO Protokoll als 2 separate Bytes ausgelesen werden muss. Mit Hilfe des Protokoll-Teils E/A-NUM kann der Anwender auswählen welches Byte er lesen möchte. Die E/A-Nummer 0 entspricht dem Low-Byte und die E/A-Nummer 1 entspricht dem High-Byte.

10.5.2. Übersicht der Bits im Error Register

Das Fehler Register enthält ein Abbild der im Gerät aufgetretenen Fehler. An Hand der Bits im Fehler Register lässt sich feststellen welche Fehler aufgetreten sind. Im laufenden Betrieb bleiben die Bits solange gesetzt, bis sie mit dem Befehl Set Config - Quit Errors abgelöscht werden oder ein Power-Cycle durchgeführt wird. Ausgenommen von dieser Regel ist das Watchdog-Timer Fehlerbit, weitere Informationen stehen im Erklärungstext zu Bit 2.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	VAL OUT ERR	VAL IN ERR	IO NUM ERR	CMD ERR	-	WDT ERR	FRM ERR	CONF ERR
Startwert	0	0	0	0	0	0	0	0

Tabelle 47: Bits im Error Register - Unteres Byte (Low Byte)

Bit 0 - CONF ERR → Konfigurationsfehler

Dieses Bit wird gesetzt, wenn im Gerät ein Konfigurationsfehler festgestellt wird. Im normalen Betrieb taucht dieser Fehler nicht auf, außer es werden die DIP Schalter während des laufenden Betriebs geändert. Um diesen Zustand zu beheben, müssen entweder die DIP-Schalter wieder in ihre ursprüngliche Position geschoben werden oder es erfolgt ein Power Cycle, dadurch wird eine neue Konfiguration vom Gerät übernommen.

NOTICE

Wurden die DIP-Schalter geändert und ein Power Cycle durchgeführt, kommt es zu geänderten Geräte-Einstellung. Die Anpassung muss ebenfalls im Steuerungsprogramm stattfinden, sonst kann das Gerät nicht mehr angesprochen und genutzt werden.

Bit 1 – FRM ERR → Kommunikationsfehler

Stellt das Gerät einen Kommunikationsfehler fest, wird dieses Bit gesetzt. In der Regel handelt es sich um drei Fehlerzustände. Das Gerät kann einen Parity-, Frame- oder Overrun-Fehler feststellen.

Ein Parity-Fehler tritt meistens dann auf, wenn die Einstellungen am Steuerungsgerät nicht korrekt sind. Gleiches gilt für den Frame-Fehler, hier stimmt in den meisten Fällen die Baudrate nicht und das Gerät empfängt keine gültigen Daten. Der Overrun-Fehler hingegen bedeutet, dass zu schnell zu viele Daten über den Bus geschickt werden und das Gerät nicht alles aufnehmen kann.

Prüfen Sie die Einstellungen am Steuergerät, stimmen diese mit der Konfiguration des Geräts überein. Wurde die Verkabelung korrekt durchgeführt und liegen keine Störungen am Bus vor.

Bit 2 – WDT ERR → Der Watchdog-Timer ist abgelaufen

Dieses Bit zeigt an ob vor dem letzten Einschalten der Watchdog-Timer abgelaufen war oder nicht. Da das Gerät bei Ablauf des Watchdog-Timers in den sicheren Zustand geht, kann dieses Bit immer nur nach einem darauffolgenden Power-Cycle abgefragt werden. Das Bit kann mit dem Befehl Set Config - Quit Errors im laufenden Betrieb abgelöscht werden oder es wird ein weiterer Power-Cycle durchgeführt.

Bit 4 – CMD ERR → Falscher oder nicht unterstützter Befehl

Wurde dem Gerät eine Nachricht mit einem nicht unterstützen Befehl geschickt, wird dieses Fehlerbit gesetzt.

Über dieses Fehlerbits lässt sich im Steuerungsprogramm feststellen, ob dieser Fehler seit dem letzten Einschalten aufgetreten ist.

Bit 5 – IO NUM ERR → Falsche oder nicht vorhandene E/A Nummer

Wurde dem Gerät ein Befehl mit einer nicht vorhandenen oder nicht unterstützen E/A-Nummer geschickt, wird dieses Bit gesetzt.

Bit 6 – VAL IN ERR → Nicht erlaubter Wert am Eingang festgestellt

Wird beim Einlesen der Eingänge ein Fehler festgestellt, wird dieses Bit gesetzt.

Bit 7 – VAL OUT ERR → Nicht erlaubter Wert für Ausgang bzw. Ausgänge

Wurde dem Gerät ein Befehl mit einem nicht erlaubten Wert für die digitalen Ausgänge geschickt oder der verwendete Befehl erlaubt nur einen bestimmten Wertebereich, wird dieses Bit gesetzt.

Ein Beispiel hierfür ist: Der Digitalausgang 3 soll auf 3 gesetzt werden, erlaubt sind nur die Werte 0 und 1.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	WDTDO7 ERR	CNTCL ERR	HLDOCL ERR
Startwert	0	0	0	0	0	0	0	0

Tabelle 48: Bits im Error Register - Oberes Byte (High Byte)

Bit 8 - HLDOCL ERR → Hyper-Logic Ausgang Konflikt

Die Hyper-Logic Prozessoren 0 und 1 verwenden die Digitalausgänge 0 und 4. Die entsprechenden Digitalausgänge stehen dem Anwender nicht zur Verfügung, wenn ein Hyper-Logic Prozessor eingeschaltet wurde. Versucht man dennoch die Digitalausgänge zu setzen, wird dieses Bit als Fehlerrückmeldung gesetzt und zeigt an, dass es einen Konflikt zwischen der Einstellung des Geräts und den verwendeten Digitalausgängen gibt.

Bit 9 - CNTCL ERR → Counter Config Konflikt

Die Digitaleingänge 1, 3, 5 und 7 können nicht als 2 Kanal Zähler verwendet bzw. in den 2 Sensorbetrieb geschaltet werden. Wird dies dennoch versucht, wird dieses Bit gesetzt, dadurch erhält man eine Rückmeldung über den unzulässigen Vorgang.

Bit 10 - WDTDO7 ERR → Watchdog-Timer - Digitalausgang 7 Konflikt

Wenn der Watchdog-Timer so konfiguriert wurde, dass er den Digitalausgang 7 als Signalausgang verwendet, dann steht dieser Ausgang dem Anwender nicht zur Verfügung. Versucht der User dennoch den Digitalausgang 7 zu setzen, wird dieses Bit auf 1 gesetzt und zeigt eine Fehlbedienung an.

10.5.3. Beispiel - Eingänge lesen

Anhand der folgenden Beispiele wird gezeigt, wie man mit dem PiXtend eIO Protokoll vom Digital One Gerät die digitalen Eingänge abfragen kann. Zuerst kommt der Befehl des Masters und dann die Antwort des Slaves.

- Digitaleingang 0 von Gerät mit Adresse 1 einlesen:
 - Befehl: #001:001:00:000*
 - Antwort: #001:001:00:001*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 1 geschickt, den Digitaleingang 0 auszulesen und dem Master als Antwort zurück zu schicken. In diesem Beispiel liefert der Digitaleingang 0 eine 1, somit liegt ein HIGH-Pegel am Eingang an.

- Digitaleingang 3 von Gerät mit Adresse 1 einlesen:
 - Befehl: #001:001:03:000*
 - Antwort: #001:001:03:000*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 1 geschickt, den Digitaleingang 3 auszulesen und dem Master als Antwort zurück zu schicken. In diesem Beispiel liefert der Digitaleingang 3 eine 0, somit liegt kein HIGH-Pegel am Eingang an, der Zustand ist LOW.

- Digitaleingang 7 von Gerät mit Adresse 1 einlesen:
 - Befehl: #001:001:07:000*
 - Antwort: #001:001:07:001*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 1 geschickt, den Digitaleingang 7 auszulesen und dem Master als Antwort zurück zu schicken. In diesem Beispiel liefert der Digitaleingang 7 eine 1, somit liegt ein HIGH-Pegel am Eingang an.

- Digitaleingänge 0-7 (alle) von Gerät mit Adresse 1 einlesen:
 - Befehl: #001:002:00:000*
 - Antwort: #001:002:00:021*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 2 geschickt, als Antwort erhalten wir unser Kommando zurück, erhalten jedoch als Wert die Zahl 21 (Binär: 0001_0101). somit sind die Eingänge 0, 2 und 4 gesetzt, hier liegt jeweils ein HIGH-Pegel an.

10.5.4. Beispiel - Ausgänge setzen

Anhand der folgenden Beispiele wird gezeigt, wie man mit dem PiXtend eIO Protokoll am Digital One Gerät die digitalen Ausgänge setzen kann. Zuerst kommt der Befehl des Masters und dann die Antwort des Slaves.

- Digitalausgang 0 am Gerät mit Adresse 1 setzen
 - Befehl: #001:003:00:001*
 - Antwort: #001:003:00:001*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 3 geschickt, den Digitalausgang 0 zu setzen. Als Antwort erhalten wir unseren eigenen Befehl in Kopie zurück und damit die Rückmeldung, dass der Slave den Befehl erkannt und verstanden hat.

- Digitalausgang 0 am Gerät mit Adresse 1 ausschalten
 - Befehl: #001:003:00:000*
 - Antwort: #001:003:00:000*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 3 geschickt, den Digitalausgang 0 auszuschalten. Als Antwort erhalten wir unseren eigenen Befehl in Kopie zurück und damit die Rückmeldung, dass der Slave den Befehl erkannt und verstanden hat.

- Digitalausgang 7 am Gerät mit Adresse 1 setzen
 - Befehl: #001:003:07:001*
 - Antwort: #001:003:07:001*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 3 geschickt, den Digitalausgang 7 zu setzen. Als Antwort erhalten wir unseren eigenen Befehl in Kopie zurück und damit die Rückmeldung, dass der Slave den Befehl erkannt und verstanden hat.

- Digitalausgang 7 am Gerät mit Adresse 1 ausschalten
 - Befehl: #001:003:07:000*
 - Antwort: #001:003:07:000*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 3 geschickt, den Digitalausgang 7 auszuschalten. Als Antwort erhalten wir unseren eigenen Befehl in Kopie zurück und damit die Rückmeldung, dass der Slave den Befehl erkannt und verstanden hat.

- Digitalausgänge 0-7 (alle) am Gerät mit Adresse 1 einschalten
 - Befehl: #001:004:00:255*
 - Antwort: #001:004:00:255*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 4 geschickt, alle Digitalausgänge zu setzen. Als Antwort erhalten wir unseren eigenen Befehl in Kopie zurück und damit die Rückmeldung, dass der Slave den Befehl erkannt und verstanden hat. Der Protokoll-Teil E/A-Nummer wird nicht beachtet, wir setzen ihn auf 00.

- Digitalausgänge 0-7 (alle) am Gerät mit Adresse 1 ausschalten
 - Befehl: #001:004:00:000*
 - Antwort: #001:004:00:000*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 4 geschickt, alle Digitalausgänge auszuschalten. Als Antwort erhalten wir unseren eigenen Befehl in Kopie zurück und damit die Rückmeldung, dass der Slave den Befehl erkannt und verstanden hat. Der Protokoll-Teil E/A-Nummer wird nicht beachtet, wir setzen ihn auf 00.

10.5.5. Beispiel - Error Register auslesen

An Hand der folgenden Beispiele zeigen wir, wie man mit dem PiXtend eIO Protokoll vom Digital One Gerät das Fehlerregister auslesen kann. Hier ist zu beachten, dass es sich beim Error Register um einen 16 Bit Wert handelt, der über jeweils ein separates Byte, Low-Byte und High-Byte, abgerufen werden muss. Die Auswahl des Bytes erfolgt über den Protokoll-Teil E/A-NUM. Das Low-Byte entspricht 00 und das High-Byte 01.

- Fehlerregister von Gerät mit Adresse 1 auslesen - Low-Byte
 - Befehl: #001:400:00:000*
 - Antwort: #001:400:00:003*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 400 geschickt, das Fehlerregister an den Master zu schicken, das Low-Byte. Hier haben wir die Zahl 3 (Binär: 000_0011) zurückbekommen, somit liegt ein Konfigurationsfehler vor und das Gerät hat einen Frame-Error bei der Kommunikation festgestellt.

- Fehlerregister von Gerät mit Adresse 1 auslesen - High-Byte
 - Befehl: #001:400:01:000*
 - Antwort: #001:400:01:000*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 400 geschickt, das Fehlerregister an den Master zu schicken, das High-Byte. Hier haben wir die Zahl 0 zurückbekommen, das heißt, dass seit dem letzten Einschalten kein Fehler aufgetreten ist.

10.6. PiXtend eIO Analog One

Beim Analog One Gerät hat eine Nachricht immer eine **Länge** von 17 Zeichen (Bytes).

Das Analog One Gerät hat die **Geräteerkennung**: 170 (Hex: AA)

Anzahl der analogen Eingänge: 8 (0 - 7), 4 Spannung + 4 Strom

Wertebereich der analogen Eingänge: 0 - 1023

Anzahl der analogen Ausgänge: 6 (0 - 5), 4 Spannung + 2 Strom

Wertebereich der analogen Ausgänge: 0 - 4095

Beispiel-Nachricht für ein Analog One Gerät:

Beschreibung	START	ADR	TREN	KOM	TREN	E/A- NUM	TREN	WERT	ENDE
ASCII-Zeichen	#	012	:	003	:	02	:	0001	*
Byte- Nummer	1	2-3-4	5	6-7-8	9	10-11	12	13-14-15-16	17

(START = Start, ADR = Adresse, TREN = Trenner, KOM = Kommando, E/A NUM = E/A Nummer, WERT = Wert, ENDE = Ende)

10.6.1. Übersicht unterstützter Befehle - Basic

Die nachfolgenden Befehle können an ein Analog One Gerät per Nachricht geschickt werden, um dessen analoge Eingänge zu lesen oder die analogen Ausgänge zu setzen.

Kommando	Nummer	Werte-Bereich	Bemerkung
Read Analog Input	5	0 ... 1023	Einen analogen Eingang lesen.
Write Analog Output	6	0 ... 4095	Einen analogen Ausgang schreiben/setzen
Read Device ID	300	0 ... 255	Geräteerkennung abfragen, 8 Bit Wert, 1 Byte.
Read Error Register	400	0 ... 255	Fehlerregister abfragen, Aufschlüsselung der Bits im Register siehe Kapitel 10.6.2 Übersicht der Bits im Error Register, 16 Bit Wert, E/A-NUM: 0 = Low Byte, kein High-Byte vorhanden.
Set Config - Quit Errors	401	0	Alle Fehler im Fehlerregister ablöschen, E/A-NUM = 00 und WERT = 0000
Read Firmware Version	500	0 ... 999	Version der Geräte Firmware auslesen im Format XXX, z.B. 101 = 1.01, 102 = 1.02

Tabelle 49: Analog One: Übersicht unterstützter Befehle - Basic

Das Fehlerregister ist ein 16 Bit Wert, es werden nur die unteren 8 Bits genutzt, mit dem PiXtend eIO Protokoll wird nur das erste Byte ausgelesen. Mit Hilfe des Protokoll-Teils E/A-NUM wählt der Anwender aus, welches Byte er lesen möchte. Die E/A-Nummer 0 entspricht dem Low-Byte, das High-Byte benötigen wir nicht.

10.6.2. Übersicht der Bits im Error Register

Das Fehler Register enthält ein Abbild der im Gerät aufgetretenen Fehler. An Hand der Bits im Fehler Register lassen sich die aufgetretenen Fehler feststellen. Im laufenden Betrieb bleiben die Bits solange gesetzt, bis sie mit dem Befehl Set Config - Quit Errors abgelöscht werden oder ein Power-Cycle durchgeführt wird. Ausgenommen von dieser Regel ist das Watchdog-Timer Fehlerbit, weitere Informationen stehen im Erklärungstext zu Bit 2.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	VAL OUT ERR	VAL IN ERR	IO NUM ERR	CMD ERR	-	WDT ERR	FRM ERR	CONF ERR
Startwert	0	0	0	0	0	0	0	0

Tabelle 50: Bits im Error Register - Unteres Byte (Low Byte)

Bit 0 - CONF ERR → Konfigurationsfehler

Dieses Bit wird gesetzt, wenn im Gerät ein Konfigurationsfehler festgestellt wird. Im normalen Betrieb taucht dieser Fehler nicht auf, außer es werden die DIP-Schalter während des laufenden Betriebs geändert. Um diesen Zustand zu beheben, müssen die DIP-Schalter in ihre ursprüngliche Position geschoben werden oder es erfolgt ein Power Cycle um die neue Geräte-Konfiguration zu übernehmen.

NOTICE

Wurden die DIP-Schalter geändert und ein Power Cycle durchgeführt, kommt es zu geänderten Geräte-Einstellung. Die Anpassung muss ebenfalls im Steuerungsprogramm stattfinden, sonst kann das Gerät nicht mehr angesprochen und verwendet werden

Bit 1 - FRM ERR → Kommunikationsfehler

Dieses Bit wird gesetzt, wenn das Gerät einen Kommunikationsfehler feststellt. In der Regel handelt es sich um drei Fehlerzustände. Das Gerät kann einen Parity-, Frame- oder Overrun-Fehler feststellen.

Ein Parity-Fehler tritt meistens dann auf, wenn die Einstellungen am Steuerungsgerät nicht korrekt sind. Gleiches gilt für den Frame-Fehler, hier stimmt in den meisten Fällen die Baudrate nicht und das Gerät empfängt keine gültigen Daten. Der Overrun-Fehler bedeutet, dass zu schnell zu viele Daten über den Bus geschickt werden und das Gerät nicht alles aufnehmen kann.

Prüfen Sie die Einstellungen am Steuergerät, stimmen diese mit der Konfiguration des Geräts überein, wurde die Verkabelung korrekt durchgeführt und liegt keine Störungen am Bus vor.

Bit 2 - WDT ERR → Der Watchdog-Timer ist abgelaufen

Dieses Bit zeigt an ob vor dem letzten Einschalten der Watchdog-Timer abgelaufen war oder nicht. Da das Gerät bei Ablauf des Watchdog-Timers in den sicheren Zustand geht, kann dieses Bit immer nur nach einem darauffolgenden Power-Cycle abgefragt werden. Das Bit kann mit dem Befehl Set Config - Quit Errors im laufenden Betrieb abgelöscht werden oder es wird ein weiterer Power-Cycle durchgeführt.

Bit 4 - CMD ERR → Falscher oder nicht unterstützter Befehl

Wurde dem Gerät eine Nachricht mit einem nicht unterstützen Befehl geschickt, dann wird dieses Fehlerbit gesetzt.

Anhand dieses Fehlerbits lässt sich im Steuerungsprogramm feststellen, ob dieser Fehler seit dem letzten Einschalten aufgetreten ist.

Bit 5 - IO NUM ERR → Falsche oder nicht vorhandene E/A Nummer

Wurde dem Gerät ein Befehl mit einer nicht vorhandenen oder nicht unterstützen E/A-Nummer geschickt, wird dieses Bit gesetzt.

Bit 6 - VAL IN ERR → Nicht erlaubter Wert am Eingang festgestellt

Wird beim Einlesen der Eingänge ein Fehler festgestellt, wird dieses Bit gesetzt.

Bit 7 - VAL OUT ERR → Nicht erlaubter Wert für Ausgang bzw. Ausgänge

Wurde dem Gerät ein Befehl mit einem nicht erlaubten Wert für die analogen Ausgänge geschickt oder der verwendete Befehl erlaubt nur einen bestimmten Wertebereich, wird dieses Bit gesetzt.

Ein Beispiel hierfür ist: Analogausgang 3 soll auf 5900 gesetzt werden, erlaubt sind nur Werte zwischen 0 und 4095.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 51: Bits im Error Register - Oberes Byte (High Byte)

Beim Analog One Gerät gibt es im oberen Byte (High-Byte) des Fehlerregisters keine Fehlerbits. Dieses Byte kann ignoriert werden, nur die Bits 0 bis 7 im unteren Byte haben eine Wertigkeit.

10.6.3. Beispiel - Eingänge lesen

Anhand der folgenden Beispiele zeigen wir, wie man mit dem PiXtend eIO Protokoll vom Analog One Gerät die analogen Eingänge abfragen kann. Es ist zu beachten, dass die analogen Eingänge einen Wertebereich von 0 bis 1023 haben. Dieser Rohwert muss zum entsprechenden Spannungs- bzw. Stromwert umgerechnet werden.

- Analogeingang 0 (Spannung) von Gerät mit Adresse 3 einlesen
 - Befehl: #003:005:00:0000*
 - Antwort: #003:005:00:0511*
 - Beschreibung: An den Slave mit Adresse 3 wird der Befehl 5 geschickt, den Analogeingang 0 auszulesen und dem Master als Antwort zurück zu schicken. In diesem Beispiel liefert der Analogeingang 0 den Wert 511 zurück, das sind in etwa 5 Volt im 10 Volt Messbereich des Geräts.
- Analogeingang 4 (Strom) von Gerät mit Adresse 3 einlesen
 - Befehl: #003:005:04:0000*
 - Antwort: #003:005:04:0200*
 - Beschreibung: An den Slave mit Adresse 3 wird der Befehl 5 geschickt, den Analogeingang 4 auszulesen und dem Master als Antwort zurück zu schicken. In diesem Beispiel liefert der Analogeingang 4 den Wert 200 zurück, das sind in etwa 4mA.

Für die Umrechnung der Rohwerte in Strom- und Spannungswerte siehe Kapitel 6.17 Strom und Spannung Umrechnungsfaktor.

10.6.4. Beispiel - Ausgänge setzen

Anhand der folgenden Beispiele zeigen wir, wie man mit dem PiXtend eIO Protokoll am Analog One Gerät die analogen Ausgänge setzen kann. Es ist zu beachten, dass die analogen Ausgänge einen Wertebereich von 0 bis 4095 haben. Um von einem Spannungs- bzw. Stromwert den Rohwert für das Gerät zu erhalten, muss eine Umrechnung stattfinden

- Analogausgang 0 (Spannung) am Gerät mit Adresse 3 setzen
 - Befehl: #003:006:00:1023*
 - Antwort: #003:006:00:1023*
 - Beschreibung: An den Slave mit Adresse 3 wird der Befehl 6 geschickt, den Analogausgang 0 auf den Wert 1023, das sind in etwa 2,5 Volt, zu setzen. Als Rückantwort oder als Bestätigung erhalten wir die gesendete Nachricht zurück.
- Analogausgang 4 (Strom) am Gerät mit Adresse 3 setzen
 - Befehl: #003:006:04:0820*
 - Antwort: #003:006:04:0820*
 - Beschreibung: An den Slave mit Adresse 3 wird der Befehl 6 geschickt, den Analogausgang 4 auf den Wert 820, das sind in etwa 4 mA, zu setzen. Als Rückantwort oder als Bestätigung erhalten wir die gesendete Nachricht zurück.

Für die Umrechnung der Rohwerte in Strom- und Spannungswerte siehe Kapitel 6.17 Strom und Spannung Umrechnungsfaktor.

10.6.5. Beispiel - Error Register auslesen

An Hand der folgenden Beispiele zeigen wir, wie man mit dem PiXtend eIO Protokoll vom Analog One Gerät das Fehlerregister auslesen kann. Es ist zu beachten, dass es sich beim Error Register um einen 16 Bit Wert handelt. Der Wert wird über ein separates Byte, Low-Byte und High-Byte, abgerufen. Beim Analog One Gerät werden die unteren 8 Bits verwendet. Die Auswahl des Bytes erfolgt über den Protokoll-Teil E/A-NUM. Das Low-Byte entspricht 00 und das High-Byte benötigen wir nicht, da es immer null ist.

- Fehlerregister von Gerät mit Adresse 3 auslesen - Low-Byte
 - Befehl: #003:400:00:0000*
 - Antwort: #003:400:00:0032*
 - Beschreibung: An den Slave mit Adresse 3 wird der Befehl 400 geschickt, das Low-Byte des Fehlerregisters an den Master zurückzuschicken. Wir erhalten die Zahl 32 (Binär: 0010_0000), d.h. es wurde dem Gerät in der seit dem letzten Einschalten, Fehlerquittierung oder Power-Cycle, mindestens einmal ein Befehl mit einer ungültigen E/A-Nummer Angabe geschickt.

10.7. PiXtend eIO Protokoll - Advanced

Neben den einfachen Befehlen zum Lesen und Steuern der digitalen und analogen Ein- und Ausgänge der Digital One und Analog One Geräte, gibt für jedes Geräte weitere Befehle. Beispielsweise um den Watchdog-Timer einzustellen und zu starten oder um Impulse auf dem Modul zu zählen. Das Einstellen und Aktivieren der Hyper-Logic wird ebenfalls beschrieben.

10.7.1. PiXtend eIO Digital One

Das Digital One Gerät bietet neben digitalen Ein- und Ausgängen noch weitere Funktionen, die über zusätzliche PiXtend eIO Protokoll Befehle konfiguriert und aktiviert werden können.

10.7.1.1 Übersicht unterstützter Befehle

Die nachfolgenden Befehle können an ein Digital One Gerät per Nachricht geschickt werden, um dessen Zusatzfunktionen einzustellen und diese ein- oder auszuschalten.

Kommando	Num	Bereich	Bemerkung
Read Status	200	0 ... 255	Status vom Modul abfragen. Aufschlüsselung der Bits, siehe Abschnitt 10.7.1.2 Übersicht der Bits im Status-Register. Die Auswahl des abzufragenden Bytes geschieht über die E/A-Nummer: 0 = Low Byte, 1 = High Byte.
Set Config - HL 0	402	0 ... 1	Hyper-Logic Prozessor 0 aktivieren. 0 = Aus, 1 = An, siehe auch Abschnitt 10.7.1.5 Hyper-Logic Konfiguration.
Set Config - HL0 Config	403	0 ... 14	Hyper-Logic Prozessor 0 Verknüpfung von DI0, DI1, DI2 und DI3 mit UND und ODER Gliedern festlegen zum schnellen Setzen von DO0. Siehe Abschnitt 10.7.1.5 Hper-Logic Konfiguration
Set Config - HL 1	404	0 ... 1	Hyper-Logic Prozessor 1 aktivieren. 0 = Aus, 1 = An, siehe Abschnitt 10.7.1.5 Hper-Logic Konfiguration
Set Config - HL 1 Config	405	0 ... 14	Hyper-Logic Prozessor 1 Verknüpfung von DI4, DI5, DI6 und DI7 mit UND und ODER Gliedern festlegen zum schnellen Setzen von DO4. Siehe Abschnitt 10.7.1.5 Hper-Logic Konfiguration.
Set Config - HL	406	0 ... 255	Festlegen wie die Eingänge DI0 - DI7 betrachtet werden. Standard: Positiv (0) oder invertiert/negativ (1). Es sind 8 Bits im Low-Byte, für jeden Eingang ein Bit. Je Bit gibt es die Option: 0 = Aus, 1 = An (negiert), im oberen Byte kann man das smoothing/debouncing der Eingänge festlegen. Siehe Abschnitt 10.7.1.5 Hper-Logic Konfiguration
Set Counter Config	407	0 ... 3	Zähler 0-7 ein- oder ausschalten. Über die E/A-Nummer kann der Zähler ausgewählt werden. Die Optionen sind 0 = Aus, 1 = An (SF), 2 = An (FF), 3 = An (SF + FF) ⁵ , Zähler Nummer entspricht DI Nummer 0 bis 7. Siehe Abschnitt 10.7.1.4 Zählerkonfiguration.
Set Counter Type	408	0 ...3	Zählertyp, die Optionen sind: 0 = Zähler hoch (Up Counter - Standard), 1 = Zähler runter (Down Counter), 2 = Zähler hoch/runter, 2 Sensorbetrieb (2 Kanal Zähler). Siehe Abschnitt 10.7.1.4 Zählerkonfiguration.
Set Counter Reset	409	0	Zähler zurücksetzen, Befehl setzt den Zähler mit der angegebenen EA-Nummer 0 - 7 zurück. Bei der Sonderzahl 8 werden alle Zähler zurückgesetzt. Siehe Abschnitt 10.7.1.4 Zählerkonfiguration.
Set Counter Pre-Set	410	0 ... 255	Zähler Pre-Set Wert (16 Bit Wert) setzen, muss mittels 2 Bytes übertragen werden. Dieser Wert wird auf einen Zähler vorgeladen, wenn ein Zähler zurückgesetzt wird. Standardwert ist 0. Byte Auswahl über E/A-Nummer: 0 = Low Byte und 1 = High Byte. Siehe Abschnitt 10.7.1.4 Zählerkonfiguration.
Read Counter Low Byte	450	0 ... 255	Zähler unteres Byte (Low Byte) lesen. Die Zählereinheiten sind 16 Bit Zähler, jeder Zähler muss mittels 2 separater Bytes ausgelesen werden. Die Zählerauswahl erfolgt über die E/A-Nummer 0 - 7, wird das Low Byte für einen Zähler abgefragt, wird das High Byte dazu eingefroren und muss als nächster Befehl sofort abgefragt werden. Wird nochmals ein Low-Byte abgefragt, wird das vorherige High Byte überschrieben. Wird ein anderer Zähler abfragt, bevor das High Byte der vorherigen Anfrage abgerufen wurde, wird das High Byte überschrieben. Siehe Abschnitt 10.7.1.4 Zählerkonfiguration.
Read Counter High Byte	451	0 ... 255	Zähler oberes Byte (High Byte) lesen. Die Zählereinheiten sind 16

⁵SF = Steigende Flanke, FF = Fallende Flanke

Kommando	Num	Bereich	Bemerkung
			Bit Zähler, jeder Zähler muss mittels 2 separater Bytes ausgelesen werden. Die Zählerauswahl erfolgt über die E/A-Nummer 0 - 7, vor Abfrage des High Byte muss das Low Byte abgefragt werden und dann das High Byte. → Nie das High Byte zuerst abfragen! Siehe Abschnitt 10.7.1.4 Zählerkonfiguration Zählerkonfiguration
Set Watchdog-Timer Time	600	0... 9	Timeoutzeit für den Watchdog-Timer setzen. Siehe Abschnitt 10.7.1.3 Watchdog-Timer Konfiguration, Standard sind 4 Sekunden Timeoutzeit.
Set Watchdog-Timer Behavior	601	0 ... 1	Watchdog-Timer Verhalten ändern, wenn der Watchdog-Timer abläuft gibt es zwei Möglichkeiten: 0 = Alle Ausgänge behalten ihren aktuellen Zustand bei (Voreinstellung), 1 = Alle Ausgänge werden aktiv abgeschaltet. Beachten Sie hierzu auch unsere Hinweise zum sicheren Zustand in Kapitel 6.2 Definition „sicherer Zustand“.
Set Watchdog-Timer Output	602	0 ... 1	Der Watchdog-Timer setzt den Digitalausgang 7 wenn er abläuft, die Optionen sind: 0 = Aus, d.h. der Anwender kann den DO7 selbst steuern oder 1 = An, der Watchdog-Timer übernimmt die Steuerung des DO7, der Anwender hat keinen Einfluss mehr. Siehe Informationen im Kapitel 10.7.1.3 Watchdog-Timer Konfiguration
Set Watchdog-Timer Out Behavior	603	0 ... 1	Watchdog-Timer Digitalausgang 7 Verhalten ändern. In der Voreinstellung setzt der Watchdog-Timer den Digitalausgang 7 auf HIGH, wenn er abläuft. Dieses Verhalten lässt sich umdrehen, im normalen Betrieb ist der Digitalaisgang 7 immer an (HIGH) und wenn der Watchdog-Timer abläuft schaltet er den Digitalausgang 7 aus, geht auf LOW. 0 = HIGH aktiv, 1 = LOW aktiv
Set Watchdog-Timer Enable	604	0 ... 1	Watchdog-Timer Ein-/Ausschalten, der Wert 0 schaltet den Watchdog aus und der Wert 1 schaltet ihn ein.

Tabelle 52: Digital One: Übersicht unterstützter Befehle - Advanced

10.7.1.2 Übersicht der Bits im Status Register

Das Status Register zeigt an, welche Funktion im Gerät aktiv ist und welche nicht. Ein zyklisches Abfragen wird empfohlen, zusätzlich kann geprüft werden ob eine Funktion aktiviert wurde oder nicht.

Das Status Register ist 16 Bits groß und muss daher über zwei separate Bytes ausgelesen werden (Low Byte und High Byte).

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	CNT2	CNT1	CNT0	HL1	HLO	WDT ON	WDT OBE	WDT OUT
Startwert	0	0	0	0	0	0	0	0

Tabelle 53: Bits im Status Register - Unteres Byte (Low Byte)

Bit 0 - WDT OUT → Watchdog-Timer Ausgang aktiv (Digitalausgang 7)

Aus: Wenn der Watchdog-Timer abläuft wird der Digitalausgang 7 nicht gesetzt, er wird als normaler digitaler Ausgang behandelt.

An: Der Digitalausgang 7 wird vom Watchdog-Timer verwendet und steht dem Anwender nicht mehr zur Verfügung. In der Voreinstellung setzt der Watchdog-Timer, wenn er abläuft, den Digitalausgang 7 auf HIGH. Dies kann vom Anwender geändert werden. Siehe Bit 1 - WDT OBE. Siehe weitere Informationen im Abschnitt 10.7.1.3 Watchdog-Timer Konfiguration

Bit 1 - WDT OBE → Watchdog-Timer Ausgang Verhalten

Aus: Der Digitalausgang 7 wird von LOW → HIGH geschaltet, wenn der Watchdog-Timer abläuft. Der Normalzustand für diese Einstellung ist LOW. Diese Einstellung ist das Standardverhalten (Voreinstellung).
Diese Einstellung ist praktisch, wenn zum Beispiel eine Lampe oder Sirene geschaltet werden soll. Läuft der Watchdog-Timer ab, kann ein Fehlzustand der Anlage schneller erkannt werden.

An: Der Digitalausgang 7 wird von HIGH → LOW geschaltet, wenn der Watchdog-Timer abläuft. Der Normalzustand für den Digitalausgang 7 ist HIGH. Diese Einstellung kann praktisch sein, wenn im Betrieb ein Relais angezogen sein muss, damit die Anlage betriebsbereit ist. Läuft der Watchdog-Timer ab, wird der Digitalausgang 7 auf LOW geschaltet und das Relais geht aus. Siehe weitere Informationen im Abschnitt 10.7.1.3 Watchdog-Timer Konfiguration

Bit 2 - WDT ON → Watchdog-Timer aktiv

Aus: Der Watchdog-Timer ist aus.

An: Der Watchdog-Timer ist aktiv und wird je nach Einstellung zurückgesetzt oder löst nach der vorgegebenen Zeit aus. Siehe weitere Informationen im Abschnitt 10.7.1.3 Watchdog-Timer Konfiguration.

Bit 3 - HLO → Hyper-Logic Prozessor 0

Aus: Es findet keine Hyper-Logic Verarbeitung für Hyper-Logic Prozessor 0 statt.

An: Die Hyper-Logic Verarbeitung ist eingeschaltet, der Zustand des Digitalausgangs 0 kann über bis zu vier Digitaleingänge gesteuert werden. Es stehen verschiedene Kombinationsmöglichkeiten der Digitaleingänge 0 bis 3 zur Verfügung, siehe Informationen im Abschnitt 10.7.1.5 Hyper Logic Konfiguration.

Bit 4 - HL1 → Hyper-Logic Prozessor 1

Aus: Es findet keine Hyper-Logic Verarbeitung für Hyper-Logic Prozessor 1 statt.

An: Die Hyper-Logic Verarbeitung ist eingeschaltet und der Zustand des Digitalausgangs 4 kann über bis zu vier Digitaleingänge gesteuert werden. Es stehen verschiedene Kombinationsmöglichkeiten der Digitaleingänge 4 bis 7 zur Verfügung, siehe weitere Informationen im Abschnitt 10.7.1.5 Hyper Logic Konfiguration.

Bit 5 - CNT0 → Zähler 0 aktiv (Counter 0)

Aus: Der Zähler 0 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 0 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

Bit 6 - CNT1 → Zähler 1 aktiv (Counter 1)

Aus: Der Zähler 1 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 1 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

Bit 7 - CNT2 → Zähler 2 aktiv (Counter 2)

Aus: Der Zähler 2 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 2 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	CNT7	CNT6	CNT5	CNT4	CNT3
Startwert	0	0	0	0	0	0	0	0

Tabelle 54: Bits im Status Register - Oberes Byte (High Byte)

Bit 8 - CNT3 → Zähler 3 aktiv (Counter 3)

Aus: Der Zähler 3 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 3 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

Bit 9 - CNT4 → Zähler 4 aktiv (Counter 4)

Aus: Der Zähler 4 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 4 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

Bit 10 - CNT5 → Zähler 5 aktiv (Counter 5)

Aus: Der Zähler 5 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 5 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

Bit 11 - CNT6 → Zähler 6 aktiv (Counter 6)

Aus: Der Zähler 6 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 6 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

Bit 12 - CNT7 → Zähler 7 aktiv (Counter 7)

Aus: Der Zähler 7 ist nicht aktiv und es wird nicht gezählt.

An: Der Zähler 7 ist eingeschaltet. Der 16 Bit Wert des Zählers kann mit den Befehlen 450 (Read Counter Low Byte) und 451 (Read Counter High Byte) abgerufen werden.

10.7.1.3 Watchdog-Timer Konfiguration

Der Watchdog-Timer verfügt auf dem Gerät über verschiedene Einstellungen, allen voran steht die Watchdog-Timer Timeoutzeit. Diese Zeit legt fest wie lange der Watchdog-Timer wartet bis er auslöst. Der Watchdog-Timer beim PiXtend eIO Protokoll wird immer bei Bus-Aktivität zurückgesetzt, hier gibt es keine andere Einstellung. Bitte beachten Sie, dass der Watchdog nach seiner Aktivierung eine zyklische Kommunikation am Bus erfordert.

Um den Watchdog zu verwenden, setzen Sie zuerst die Timeoutzeit, sofern Sie die Voreinstellung von 4 Sekunden nicht verwenden wollen. Die nachfolgende Tabelle hilft Ihnen bei der Auswahl einer Timeoutzeit.

Nummer	Timeoutzeit	Bemerkung
000	16 ms	Nicht empfohlen bzw. sehr häufige Kommunikation notwendig
001	32 ms	Nicht empfohlen bzw. sehr häufige Kommunikation notwendig
002	64 ms	Nicht empfohlen bzw. sehr häufige Kommunikation notwendig
003	0,125 s	
004	0,250 s	
005	0,5 s	
006	1,0 s	
007	2,0 s	
008	4,0 s	Voreinstellung
009	8,0 s	Längste Timeoutzeit, gut bei wenig Bus-Aktivität

Tabelle 55: Digital One: Übersicht Watchdog-Timer Timeoutzeit

Die Watchdog-Timer Timeoutzeit kann mit dem Befehl 600 (Set Watchdog-Timer Time) geändert werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 600, Timeoutzeit 1 Sek.	#001:600:00:006*	#001:600:00:006*
Befehl 600, Timeoutzeit 8 Sek.	#001:600:00:009*	#001:600:00:009*

Tabelle 56: Watchdog-Timer Timeoutzeit setzen - Beispiele

Nachdem die Timeoutzeit eingestellt ist kann der Watchdog-Timer eingeschaltet werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 604, Watchdog-Timer an	#001:604:00:001*	#001:604:00:001*
Befehl 604, Watchdog-Timer aus	#001:604:00:000*	#001:604:00:000*

Tabelle 57: Watchdog-Timer ein / aus - Beispiele

Weitere Funktionen des Watchdog-Timers sind alle Ausgänge abschalten, wenn er abläuft oder einen Digitalausgang einschalten.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 601, Alle Ausgänge aus	#001:601:00:001*	#001:601:00:001*
Befehl 601, Alle Ausgänge bleiben unverändert	#001:601:00:000*	#001:601:00:000*
Befehl 602, Watchdog-Timer setzt D07	#001:602:00:001*	#001:602:00:001*
Befehl 603, Watchdog-Timer D07 invertieren	#001:603:00:001*	#001:603:00:001*
Befehl 603, Watchdog-Timer D07 normal	#001:603:00:000*	#001:603:00:000*

Tabelle 58: Watchdog-Timer Funktionen einstellen - Beispiele

Für weitere Informationen zum Watchdog-Timer siehe Kapitel 6.11 PiXtend eIO Watchdog-Timer im Abschnitt Basis Wissen.

10.7.1.4 Zählerkonfiguration

Über die Auswahl der Flankeneinstellung, welche Flanke eines Signals soll der Zähler beim Zählen betrachten, wird der Zähler mit eingeschaltet. Wie Sie den Zählertyp umstellen können, beschreiben wir auf den folgenden Seiten. Die Auswahl des Zählers erfolgt über den E/A-Nummer Teil des Protokolls.

Auswahl der Flankeneinstellung

Dezimal	Flankeneinstellung
0	Aus, Zähler nicht aktiv
1	Steigende Flanke (Rising Edge), 0 → 1
2	Fallende Flanke (Falling Edge), 1 → 0
3	Steigende + Fallende Flanke (RE + FE), 0 → 1 + 1 → 0

Tabelle 59: Flankeneinstellung - Aufschlüsselung

Zähler ein- und ausschalten:

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 407, Zähler 0 einschalten, steigende Flanke zählen	#001:407:00:001*	#001:407:00:001*
Befehl 407, Zähler 1 einschalten, fallende Flanke zählen	#001:407:01:002*	#001:407:01:002*
Befehl 407, Zähler 2 einschalten, steigende + fallende Flanke zählen	#001:407:02:003*	#001:407:02:003*
Befehl 407, Zähler 2 ausschalten	#001:407:02:000*	#001:407:02:000*
Befehl 407, Zähler 1 ausschalten	#001:407:01:000*	#001:407:01:000*
Befehl 407, Zähler 0 ausschalten	#001:407:00:000*	#001:407:00:000*

Tabelle 60: Zähler ein-/ausschalten mit verschiedenen Einstellungen - Beispiele

Zählertyp Auswahl

Über den Zählertyp lässt sich festgelegt wie der Zähler zählen soll. Es gibt die Einstellung, dass der Zähler hoch zählt (Voreinstellung), runter zählt und 2 Kanäle zählt. Die nachfolgende Tabelle schlüsselt diese Information näher auf.

Dezimal	Zählertyp
0	Zähler zählt hoch, Up Counter (Voreinstellung)
1	Zähler zählt runter, Down Counter
2	Zähler zählt hoch/runter, 2 Sensorbetrieb (2 Kanal Zähler)
3	Reserviert

Tabelle 61: Zählertyp - Auswahl

Die verwendeten Eingänge beim 2 Sensorbetrieb sind immer die Eingänge des jeweiligen Zählers 0, 2, 4 und 6 und der Eingang der direkt daneben liegt, also 1, 3, 5 und 7.

Für weitere Informationen zu den Zählern des Digital One siehe Kapitel 6.12 Zähler Funktion im PiXtend eIO im Abschnitt Basis Wissen.

Zählertyp einstellen:

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 408, Zähler 0 zählt hoch	#001:408:00:000*	#001:408:00:000*
Befehl 408, Zähler 1 zählt runter	#001:408:01:001*	#001:408:01:001*
Befehl 408, Zähler 4 als 2 Kanal Zähler	#001:408:04:002*	#001:408:04:002*

Tabelle 62: Zählertyp einstellen - Beispiele

Zählervorladewert ändern (Counter Pre-Set Value)

Über den Befehl 410 (Set Counter Pre-Set) kann der Wert, der bei einem Reset/Zurücksetzen eines Zählers in den Zählerspeicher geladen wird, geändert werden. Üblicherweise ist dieser Wert 0. Der Zählerspeicher je Zähler ist 16 Bits groß, somit ist der Zählervorladewert 16 Bits groß. Die Übertragung muss mittels 2 separater Bytes erfolgen. Wurde der Zählervorladewert geändert, erscheint dieser Wert nicht sofort bei jedem Zähler. Beim Zurücksetzen eines Zählers wird er in den Zählerspeicher geladen oder wenn alle Zähler gemeinsam zurückgesetzt werden. In beiden Fällen wird dieser Wert von jedem Zähler in den Zählerspeicher geladen.

Die Auswahl zwischen Low-Byte und High-Byte erfolgt über den E/A-Nummer Teil des Protokolls. Eine 0 bedeutet der Wert wird ins Low-Byte und eine 1 bedeutet der Wert wird ins High-Byte geschrieben.

Zählervorladewert - Der Vorladewert wird auf 4095 gesetzt:

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 410, Low-Byte übertragen	#001:410:00:255*	#001:410:00:255*
Befehl 410, High-Byte übertragen	#001:410:01:015*	#001:410:01:015*

Tabelle 63: Zähler Zählervorladewert - Beispiel 1

Zählervorladewert - Der Vorladewert wird auf 0 gesetzt:

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 410, Low Byte übertragen	#001:410:00:000*	#001:410:00:000*
Befehl 410, High Byte übertragen	#001:410:01:000*	#001:410:01:000*

Tabelle 64: Zähler Zählervorladewert - Beispiel 2

Zähler zurücksetzen

Über den Befehl 409 (Set Counter Reset) kann jeder der 8 Zähler einzeln oder alle gemeinsam zurückgesetzt werden. Wird ein Zähler zurückgesetzt, dann wird immer der Zählervorladewert (Counter Pre-Set Value) in den Zählerspeicher geladen. Die Voreinstellung ist hier der Wert 0. Die nachfolgende Tabelle zeigt die möglichen Zahlen zum Rücksetzen der verschiedenen Zählereinheiten. Die Auswahl erfolgt über den E/A-Nummer Teil des Protokolls, der Wert Teil findet keine Beachtung.

Dezimal	Zähler
0	Reset Zähler 0
1	Reset Zähler 1
2	Reset Zähler 2
3	Reset Zähler 3
4	Reset Zähler 4
5	Reset Zähler 5
6	Reset Zähler 6
7	Reset Zähler 7
8	Reset aller Zähler

Tabelle 65: Zähler zurücksetzen

NOTICE

Beim Reset eines Zählers, wird der Wert des internen Speichers, der Zählervorladewert in den Zählerspeicher geladen. Ohne Anwendereingriff ist dieser Wert 0.

Mit diesem Vorgehen lässt sich jeder beliebige Wert zwischen 0 und 65535 in einen Zähler vorladen. Ein Zähler muss nicht bei 0 zu zählen beginnen, er kann nach einem Reset einen anderen Wert annehmen. Bei einem Down Counter ist dieses Verhalten von Vorteil, wenn man 100 Einheiten abzählen möchte. Der Zähler könnte auf 100 voreingestellt werden, bei 0 angekommen, setzt man mit einem Reset den Zähler wieder zurück auf 100.

Beispiele zum Zurücksetzen der Zähler:

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 409, Zähler 0 zurücksetzen	#001:409:00:000*	#001:409:00:000*
Befehl 409, Zähler 1 zurücksetzen	#001:409:01:000*	#001:409:01:000*
Befehl 409, Zähler 2 zurücksetzen	#001:409:02:000*	#001:409:02:000*
Befehl 409, Zähler 7 zurücksetzen	#001:409:07:000*	#001:409:07:000*
Befehl 409, Alle Zähler zurücksetzen	#001:409:08:000*	#001:409:08:000*

Tabelle 66: Zähler zurücksetzen Beispiele

10.7.1.5 Hyper-Logic Konfiguration

Die Hyper-Logic Prozessoren 0 und 1 bieten dem Anwender eine Möglichkeit schnell auf jeweils bis zu 4 unterschiedliche Signale ohne Softwareeingriff zu reagieren, einen Ausgang zu setzen oder abzuschalten.

Das Einschalten und Konfigurieren der Hyper-Logic Prozessoren geht über die Befehle 402 (Set Config – HLO), 403 (Set Config – HLO Config), 404 (Set Config – HL1), 405 (Set Config HL1 Config) und 406 (Set Config – HL). Wir empfehlen für jeden Hyper-Logic Prozessor zuerst die Verknüpfung zu bestimmen und zu konfigurieren und im Anschluss den jeweiligen Prozessor einzuschalten.

Hyper-Logic Prozessor 0 Verknüpfung

Die folgende Tabelle zeigt die Verknüpfungen der Digitaleingänge 0 bis 3 die vom Hyper-Logic Prozessor 0 unterstützt werden. Die Bindungsstärke der logischen Verknüpfungen wird durch Klammern symbolisiert. Ein doppeltes kaufmännisches UND (&&) stellt eine UND-Verknüpfung dar und eine ODER-Verknüpfung wird durch zwei senkrechte Striche (||) ausgedrückt. Die Angabe der Spalte Nummer kann direkt mit dem Befehl 403 übertragen werden. Der Digitalausgang vom Hyper-Logic Prozessor 0 ist immer der Digitalausgang 0. Die jeweilige Verknüpfungsnummer wird mit dem Wert Teil des Protokolls übertragen, der E/A-Nummer Teil spielt keine Rolle und sollte immer 00 sein.

Nummer	Verknüpfung
0	DIO
1	DIO && DI1
2	DIO DI1
3	DIO && DI1 && DI2
4	(DIO && DI1) DI2
5	(DIO DI1) && DI2
6	DIO DI1 DI2
7	DIO && DI1 && DI2 && DI3
8	(DIO && DI1 && DI2) DI3
9	DIO && (DI1 DI2) && DI3
10	DIO DI1 && DI2 && DI3
11	(DIO && DI1) DI2 DI3
12	(DIO DI1) && (DI2 DI3)
13	(DIO DI1 DI2) && DI3
14	DIO DI1 DI2 DI3

Tabelle 67: Hyper-Logic Prozessor 0 - Verknüpfungsübersicht

Hyper-Logic Prozessor 1 Verknüpfung

Die folgende Tabelle zeigt die Verknüpfungen der Digitaleingänge 4 bis 7 die vom Hyper-Logic Prozessor 1 unterstützt werden. Die Bindungsstärke der logischen Verknüpfungen wird durch Klammern symbolisiert. Ein doppeltes kaufmännisches UND (&&) stellt eine UND-Verknüpfung dar und eine ODER-Verknüpfung wird durch zwei senkrechte Striche (||) ausgedrückt. Die Angabe der Spalte Nummer kann direkt mit dem Befehl 405 übertragen werden. Der Digitalausgang vom Hyper-Logic Prozessor 1 ist immer der Digitalausgang 4. Die jeweilige Verknüpfungsnummer wird mit dem Wert Teil des Protokolls übertragen, der E/A-Nummer Teil spielt keine Rolle und sollte immer 00 sein.

Nummer	Verknüpfung
0	DI4
1	DI4 && DI5
2	DI4 DI5
3	DI4 && DI5 && DI6
4	(DI4 && DI5) DI6
5	(DI4 DI5) && DI6
6	DI4 DI5 DI6
7	DI4 && DI5 && DI6 && DI7
8	(DI4 && DI5 && DI6) DI7
9	DI4 && (DI5 DI6) && DI7
10	DI4 DI5 && DI6 && DI7
11	(DI4 && DI5) DI6 DI7
12	(DI4 DI5) && (DI6 DI7)
13	(DI4 DI5 DI6) && DI7
14	DI4 DI5 DI6 DI7

Tabelle 68: Hyper-Logic Prozessor 1 - Verknüpfungsübersicht

Hyper-Logic Prozessor 0 auf Verknüpfung 3 einstellen und einschalten, zusätzlich wird der Digitaleingang 0 invertiert betrachtet. Siehe dazu auch Kapitel 10.7.1.6 Übersicht der Bits im Hyper-Logic Config Register.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 403, Verknüpfung einstellen	#001:403:00:003*	#001:403:00:003*
Befehl 406, DI0 für HL0 invertieren	#001:406:00:001*	#001:406:00:001*
Befehl 402, Hyper-Logic Proz. 0 ein	#001:402:00:001*	#001:402:00:001*

Tabelle 69: Hyper-Logic Prozessor 0 - Beispiel 1

Hyper-Logic Prozessor 1 auf Verknüpfung 14 einstellen und einschalten.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 405, Verknüpfung einstellen	#001:405:00:014*	#001:405:00:014*
Befehl 404, Hyper-Logic Proz. 1 ein	#001:404:00:001*	#001:404:00:001*

Tabelle 70: Hyper-Logic Prozessor 1 - Beispiel 2

Hyper-Logic Prozessor 0 und 1 ausschalten.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 402, Hyper-Logic Proz. 0 aus	#001:402:00:000*	#001:402:00:000*
Befehl 404, Hyper-Logic Proz. 1 aus	#001:404:00:000*	#001:404:00:000*

Tabelle 71: Hyper-Logic Prozessor 0 / 1 - Beispiel 3

10.7.1.6 Übersicht der Bits im Hyper-Logic Config Register

Das Hyper-Logic Config Register ermöglicht es, für jeden digitalen Eingang festzulegen, ob dieser positiv (Voreinstellung) oder negativ (invertiert) betrachtet werden soll. Die 2 Bytes müssen separat übertragen werden. Die Auswahl des Bytes geschieht über den E/A-Nummer Teil des Protokolls.

Diese Einstellung ist hilfreich, wenn man einen Sensor verwendet der LOW aktiv ist, beim Erkennen eines Gegenstandes oder Werkstücks von HIGH auf LOW schaltet.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	HLIN7	HLIN6	HLIN5	HLIN4	HLIN3	HLIN2	HLIN1	HLIN0
Startwert	0	0	0	0	0	0	0	0

Tabelle 72: Bits im Hyper-Logic Config Register - Unteres Byte (Low Byte)

Für jeden der 8 digitalen Eingänge steht ein Bit (HLINx, x = 0 bis 7) zur Verfügung.

Bit 7 .. 0 - HLIN7 .. 0 → Hyper-Logic Prozessor 0 & 1 - Einganginvertierung

Aus: Alle digitalen Eingänge werden positiv betrachtet.

An: Das Signal am gewählten Eingang wird invertiert bzw. negativ gewertet.
Bei dieser Einstellung wird davon ausgegangen, dass ein angeschlossener Sensor oder Schalter im Ruhezustand immer ein HIGH-Pegel liefert. Siehe Kapitel 10.7.1.5 Hyper-Logic Konfiguration

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	HLSM1	HLSM0
Startwert	0	0	0	0	0	0	0	0

Tabelle 73: Bits im Hyper-Logic Config Register - Oberes Byte (High Byte)

Der Hyper-Logic Prozessor 0 und 1 arbeitet mit einer Geschwindigkeit, schneller als die digitalen Eingänge reagieren können, um möglichst zügig eine Reaktion zu erhalten. Für manche Anwendungen ist es jedoch sinnvoll, die Eingangssignale zu entprellen, um ein sichereres und stabileres Ausgangssignal für jeden Hyper-Logic Prozessor zu erhalten.

Bit 8 - HLSM0 → Hyper-Logic Prozessor 0 - Signal-Smoothing (Debounce)

Aus: Es wird kein Smoothing (Debounce) durchgeführt.

An: Die Eingangssignale für den Hyper-Logic Prozessor 0 werden entprellt. Bei den Digitaleingängen 0 bis 3, sofern diese in der gewählten Hyper-Logic Verknüpfung vorkommen, muss jedes Signal länger als 2 ms anliegen bevor es als gültig betrachtet wird. Im Anschluss kann es von der Hyper-Logic verarbeitet werden.

Bit 9 - HLSM1 → Hyper-Logic Prozessor 1 - Signal-Smoothing (Debounce)

Aus: Es wird kein Smoothing (Debounce) durchgeführt.

An: Die Eingangssignale für den Hyper-Logic Prozessor 1 werden entprellt. Bei den Digitaleingängen 4 bis 7, sofern diese in der gewählten Hyper-Logic Verknüpfung vorkommen, muss jedes Signal länger als 2 ms anliegen bevor es als gültig betrachtet wird. Im Anschluss kann es von der Hyper-Logic verarbeitet werden.

10.7.2. Beispiel - Zähler verwenden

Nachfolgend sind verschiedene Beispiele aufgeführt, wie man mit den Zählereinheiten auf einem Digital One Gerät arbeiten kann.

Beispiel zu Zähler 0, fallende Flanken zählen, Zähler 0 wird als Up Counter betrieben

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 408, Zähler 0 als Hochzähler	#001:408:00:000*	#001:408:00:000*
Befehl 410, Zählervorladewert auf 0, Low Byte	#001:410:00:000*	#001:410:00:000*
Befehl 410, Zählervorladewert auf 0, High Byte	#001:410:01:000*	#001:410:01:000*
Befehl 409, Zähler 0 zurücksetzen	#001:409:00:000*	#001:409:00:000*
Befehl 407, Zähler 0 einschalten mit FF	#001:407:00:002*	#001:407:00:002*
Befehl 450, Zähler 0 lesen, Low Byte	#001:450:00:000*	#001:450:00:085*
Befehl 451, Zähler 0 lesen, High Byte	#001:451:00:000*	#001:451:00:037*

Tabelle 74: Beispiel: Zähler verwenden - Zähler 0

In diesem Beispiel zählt der Zähler 0 hoch und wird vor dem Einschalten nochmal zurückgesetzt. Aus diesem Grund wurde der Zählervorladewert auf 0 gesetzt.

Nach dem Einschalten und einer kleinen Zeitspanne wurde der Zähler 0 abgefragt, er hatte einen Zählwert von 9557 erreicht. Der Zähler 0 wird immer weiter zählen, bis er 65535 erreicht hat, dann beginnt er wieder bei 0.

Zur Handhabung bzw. Konvertierung von 16 Bit Werten siehe Kapitel 6.18 Zwei Bytes zu einem 16 Bit Wert kombinieren und 6.19 einen Bit Wert in zwei Bytes zerlegen, im Abschnitt Basis Wissen.

Beispiel zu Zähler 3, steigende Flanken zählen, Zähler 3 wird als Down Counter betrieben und mit dem Wert 10.000 vorgeladen.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 408, Zähler 3 als Runterzähler	#001:408:03:001*	#001:408:03:001*
Befehl 410, Zählervorladewert auf 10000, LB	#001:410:00:016*	#001:410:00:016*
Befehl 410, Zählervorladewert auf 10000, HB	#001:410:01:039*	#001:410:01:039*
Befehl 409, Zähler 3 zurücksetzen	#001:409:03:000*	#001:409:03:000*
Befehl 407, Zähler 3 einschalten mit SF	#001:407:03:001*	#001:407:03:001*
Befehl 450, Zähler 3 lesen, Low Byte	#001:450:03:000*	#001:450:03:136*
Befehl 451, Zähler 3 lesen, High Byte	#001:451:03:000*	#001:451:03:019*

Tabelle 75: Beispiel: Zähler verwenden – Zähler 3

Der Zähler 3 wurde als Down Counter eingestellt und auf den Wert 10000 gesetzt. Mit jeder steigenden Flanke eines HIGH-Pegels an DI3 wird der Zähler verringert. Nach einiger Zeit lesen wir den Zähler aus und erhalten den Wert 5000 zurück, nachdem wir die beiden Bytes, 136 und 19, zu einem 16 Bit-Wert zusammengeführt haben.

Zur Handhabung bzw. Konvertierung von 16 Bit Werten siehe Kapitel 6.18 Zwei Bytes zu einem 16 Bit Wert kombinieren und 6.19 einen Bit Wert in zwei Bytes zerlegen, im Abschnitt Basis Wissen

Beispiel zu Zähler 6, 2 Sensorbetrieb (2 Kanal Zähler):

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 408, Zähler 6 als 2 Kanal Zähler	#001:408:06:002*	#001:408:06:002*
Befehl 410, Zählervorladewert auf 0, Low Byte	#001:410:00:000*	#001:410:00:000*
Befehl 410, Zählervorladewert auf 0, High Byte	#001:410:01:000*	#001:410:01:000*
Befehl 409, Zähler 6 zurücksetzen	#001:409:06:000*	#001:409:06:000*
Befehl 407, Zähler 6 einschalten mit SF	#001:407:06:001*	#001:407:06:001*
Befehl 450, Zähler 6 lesen, Low Byte	#001:450:06:000*	#001:450:06:010*
Befehl 451, Zähler 6 lesen, High Byte	#001:451:06:000*	#001:451:06:000*

Tabelle 76: Beispiel: Zähler verwenden - Zähler 6 - 2 Kanal Zähler

Der Zähler 6 wurde als 2 Kanal Zähler konfiguriert und dann auf 0 zurückgesetzt. An den Digitaleingängen 6 und 7 wurde jeweils ein Kabel des Sensors angeschlossen. Diese Sensoren erfassen ein Signal, sobald sich ein Geber auf einer Scheibe an ihnen vorbei bewegt. Durch diesen Aufbau wird die Drehrichtung erkannt und die Anzahl der Scheiben-Umdrehungen erfasst.

Der Zähler selbst wird mit dem Befehl 407 aktiviert. Zeitgleich geben wir dem Zähler den Befehl, dass er nur steigende Flanken zählt. Der Motor, an dem die Scheibe befestigt ist, wird gestartet und die Scheibe dreht sich nur in eine Richtung.

Nach kurzer Zeit fragen wir den Zähler ab und erhalten den Wert 10, das bedeutet, dass die Scheibe mit dem Geber 10 Umdrehungen vollzogen hat.

10.7.3. Beispiel - Hyper-Logic Prozessor verwenden

Die nachfolgenden Beispiele zu den Hyper-Logic Prozessoren helfen bei deren Verwendung und Konfiguration und zeigen den Aufbau der Nachrichten

Hyper-Logic Prozessor 0 auf Verknüpfung 3 einstellen und einschalten, zusätzlich wird DIO invertiert betrachtet

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 403, Verknüpfung einstellen	#001:403:00:003*	#001:403:00:003*
Befehl 406, DIO für HLO invertieren	#001:406:00:001*	#001:406:00:001*
Befehl 402, Hyper-Logic Proz. 0 ein	#001:402:00:001*	#001:402:00:001*

Tabelle 77: Beispiel: Hyper-Logic Prozessor 0 verwenden

Hyper-Logic Prozessor 1 auf Verknüpfung 14 einstellen und einschalten

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 405, Verknüpfung einstellen	#001:405:00:014*	#001:405:00:014*
Befehl 404, Hyper-Logic Proz. 1 ein	#001:404:00:001*	#001:402:00:001*

Tabelle 78: Beispiel: Hyper-Logic Prozessor 1 verwenden

Hyper-Logic Prozessor 0 und 1 ausschalten

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 402, Hyper-Logic Proz. 0 aus	#001:402:00:000*	#001:402:00:000*
Befehl 404, Hyper-Logic Proz. 1 aus	#001:404:00:000*	#001:402:00:000*

Tabelle 79: Beispiel: Hyper-Logic Prozessor 0 und 1 ausschalten

10.7.4. Beispiel - Watchdog-Timer aktivieren

Die nachfolgenden Beispiele verdeutlichen die Verwendung und Konfiguration des Watchdog-Timers auf dem PiXtend eIO Digital One Gerät.

Die Watchdog-Timer Timeoutzeit kann mit dem Befehl 600 geändert werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 600, Timeoutzeit 1 Sek.	#001:600:00:006*	#001:600:00:006*
Befehl 600, Timeoutzeit 8 Sek.	#001:600:00:009*	#001:600:00:009*

Tabelle 80: Beispiel: Watchdog-Timer Timeoutzeit setzen

Ist die Timeoutzeit eingestellt kann der Watchdog-Timer eingeschaltet werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 604, Watchdog-Timer an	#001:604:00:001*	#001:604:00:001*
Befehl 604, Watchdog-Timer aus	#001:604:00:000*	#001:604:00:000*

Tabelle 81: Beispiel: Watchdog-Timer ein- und ausschalten

Zu den weiteren Funktionen des Watchdog-Timers gehört es, alle Ausgänge abzuschalten, wenn er abläuft oder den Digitalausgang 7 einzuschalten.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 601, Alle Ausgänge aus	#001:601:00:001*	#001:601:00:001*
Befehl 601, Alle Ausgänge bleiben an	#001:601:00:000*	#001:601:00:000*
Befehl 602, Watchdog-Timer setzt D07	#001:602:00:001*	#001:602:00:001*
Befehl 603, Watchdog-Timer D07 invertieren	#001:603:00:001*	#001:603:00:001*
Befehl 603, Watchdog-Timer D07 normal	#001:603:00:000*	#001:603:00:000*

Tabelle 82: Beispiel: Watchdog-Timer Funktionen einstellen

10.7.5. Beispiel - Status auslesen

Anhand der folgenden Beispiele wird gezeigt, wie man mit dem PiXtend eIO Protokoll vom Digital One Gerät den aktuellen Status auslesen kann. Es ist zu beachten, dass es sich beim Status Register um einen 16 Bit Wert handelt, der über jeweils ein separates Byte, Low-Byte und High-Byte, abgerufen werden muss. Die Auswahl des Bytes erfolgt über den Protokoll-Teil E/A-NUM. Das Low-Byte entspricht 00 und das High-Byte 01.

Zuerst kommt der Befehl und dann die Antwort des Slave.

- Status vom Gerät mit Adresse 1 auslesen, Low-Byte
 - Befehl: #001:200:00:000*
 - Antwort: #001:200:00:069*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 200 geschickt, den aktuellen Geräte Status an den Master zu schicken. Als Antwort erhalten wir das Low-Byte mit der Zahl 69 (Binär: 0100_0101). Der Watchdog-Timer wurde aktiviert und entsprechende konfiguriert, sodass er den Digitalausgang 7 verwendet, der Zähler 1 ist aktiv.

- Status vom Gerät mit Adresse 1 auslesen, High-Byte
 - Befehl: #001:200:01:000*
 - Antwort: #001:200:01:001*
 - Beschreibung: An den Slave mit Adresse 1 wird der Befehl 200 geschickt, den aktuellen Geräte Status an den Master zu schicken. Als Antwort im High-Byte erhalten wir die Zahl 1 (Binär: 0000_0001), der Zähler 3 ist aktiv.

10.7.6. PiXtend eIO Analog One

Das Analog One Gerät bietet neben analogen Ein- und Ausgängen weitere Funktionen, die über zusätzliche PiXtend eIO Protokoll Befehle konfiguriert und aktiviert werden können.

10.7.6.1 Übersicht unterstützter Befehle

Die nachfolgenden Befehle können an ein Analog One Gerät als Nachricht versendet werden um dessen Zusatzfunktionen einzustellen und diese ein- oder auszuschalten.

Kommando	Num	Bereich	Bemerkung
Read Status	200	0 ... 255	Status vom Modul abfragen. Aufschlüsselung der Bits, siehe Abschnitt 10.7.6.2 Übersicht der Bits im Status Register. Übersicht der Bits im Status Register Die Auswahl des abzufragenden Bytes erfolgt über die E/A-Nummer: 0 = Low Byte, 1 = High Byte.
Set Watchdog-Timer Time	600	0 ... 9	Timeoutzeit für den Watchdog-Timer setzen. Siehe Abschnitt 10.7.6.3 Watchdog-Timer Konfiguration, Standard sind 4 Sekunden Timeoutzeit.
Set Watchdog-Timer Behavior	601	0 ... 1	Watchdog-Timer Verhalten ändern, wenn der Watchdog-Timer abläuft gibt es zwei Möglichkeiten: 0 = Alle Ausgänge behalten ihren aktuellen Zustand (Voreinstellung), 1 = Alle Ausgänge werden aktiv abgeschaltet. Beachten Sie unsere Hinweise zum sicheren Zustand in Kapitel 6.2 Definition sicherer Zustand.
Set Watchdog-Timer Enable	604	0 ... 1	Watchdog-Timer Ein-/Ausschalten, der Wert 0 schaltet den Watchdog-Timer aus und der Wert 1 schaltet ihn ein.

Tabelle 83: Analog One: Übersicht unterstützter Befehle - Advanced

10.7.6.2 Übersicht der Bits im Status Register

Das Status Register zeigt an, welche Funktion im Gerät aktiv ist und welche nicht. Es wird ein zyklisches Abfragen empfohlen. Zusätzlich wird geprüft, ob eine Funktion aktiviert wurde oder nicht. Das Status Register ist 16 Bits groß und muss daher über zwei einzelne Bytes ausgelesen werden (Low-Byte und High-Byte). Das Analog One Gerät verfügt über Informationen im Low-Byte, das High-Byte kann ignoriert werden, ein Auslesen ist nicht notwendig.

Unteres Byte (Low Byte):

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	WDT ON	A05 OV	A04 OV
Startwert	0	0	0	0	0	0	0	0

Tabelle 84: Bits im Status Register - Unteres Byte (Low Byte)

Bit 0 - A04 OV → Analog Output 4 Overload Feedback Signal

Aus: Der Analogausgang 4 arbeitet normal, eine Last ist vorhanden, es liegt kein Kabelbruch vor.

An: Der Analogausgang 4 hat eine Überlastsituation (Overload) festgestellt, üblicherweise ist keine Last vorhanden oder es liegt ein Kabelbruch vor.

Bit 1 - A05 OV → Analog Output 5 Overload Feedback Signal

Aus: Der Analogausgang 5 arbeitet normal, eine Last ist vorhanden, es liegt kein Kabelbruch.

An: Der Analogausgang 5 hat eine Überlastsituation (Overload) festgestellt, üblicherweise ist keine Last vorhanden oder es liegt ein Kabelbruch vor.

Bit 2 - WDT ON → Watchdog-Timer aktiv

Aus: Der Watchdog-Timer ist aus.

An: Der Watchdog-Timer ist aktiv und wird je nach Einstellung zurückgesetzt und löst nach der vorgegebenen Zeit aus. Siehe weitere Informationen im Abschnitt 10.7.6.3 Watchdog-Timer Konfiguration.

Oberes Byte (High Byte):

Bit	15	14	13	12	11	10	9	8
Name	-	-	-	-	-	-	-	-
Startwert	0	0	0	0	0	0	0	0

Tabelle 85: Bits im Status Register - Oberes Byte (High Byte)

Das obere Byte (High Byte) des Status Registers enthält keine Informationen bzw. Statusbits und kann ignoriert werden.

10.7.6.3 Watchdog-Timer Konfiguration

Der Watchdog-Timer auf dem Gerät verfügt über verschiedene Einstellungen, allen voran steht die Watchdog-Timer Timeoutzeit. Diese Zeit legt fest, wie lange der Watchdog-Timer wartet bis er auslöst. Der Watchdog-Timer beim PiXtend eIO Protokoll wird bei Bus-Aktivität immer zurückgesetzt, es ist keine andere Einstellung möglich. Denken Sie daran, dass der Watchdog-Timer nach seiner Aktivierung eine zyklische Kommunikation am Bus erfordert. Um den Watchdog-Timer zu verwenden, setzen Sie zuerst die Timeoutzeit, wenn Sie die Voreinstellung von 4 Sekunden nicht verwenden möchten. Die nachfolgende Tabelle unterstützt sie bei der Auswahl einer Timeoutzeit.

Nummer	Timeoutzeit	Bemerkung
000	16 ms	Nicht empfohlen bzw. sehr häufige Kommunikation notwendig
001	32 ms	Nicht empfohlen bzw. sehr häufige Kommunikation notwendig
002	64 ms	Nicht empfohlen bzw. sehr häufige Kommunikation notwendig
003	0,125 s	
004	0,250 s	
005	0,5 s	
006	1,0 s	
007	2,0 s	
008	4,0 s	Voreinstellung
009	8,0 s	Längste Timeoutzeit, gut bei wenig Bus-Aktivität

Tabelle 86: Analog One: Übersicht Watchdog-Timer Timeoutzeit

Die Watchdog-Timer Timeoutzeit kann mit dem Befehl 600 (Set Watchdog-Timer Time) geändert werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 600, Timeoutzeit 1 Sek.	#003:600:00:006*	#003:600:00:006*
Befehl 600, Timeoutzeit 8 Sek.	#003:600:00:009*	#003:600:00:009*

Tabelle 87: Watchdog-Timer Timeoutzeit setzen – Beispiele

Ist das Timeout eingestellt kann der Watchdog aktiviert werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 604, Watchdog-Timer an	#003:604:00:001*	#003:604:00:001*
Befehl 604, Watchdog-Timer aus	#003:604:00:000*	#003:604:00:000*

Tabelle 88: Watchdog-Timer ein/aus – Beispiele

Es ist eine weitere Funktion des Watchdog-Timers, alle Ausgänge abzuschalten, wenn er abläuft.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 601, Alle Ausgänge aus	#003:601:00:001*	#003:601:00:001*
Befehl 601, Alle Ausgänge bleiben unverändert	#003:601:00:000*	#003:601:00:000*

Tabelle 89: Watchdog-Timer Funktion einstellen – Beispiele

10.7.7. Beispiel - Watchdog-Timer aktivieren

Die nachfolgenden Beispiele sollen die Verwendung und Konfiguration des Watchdog-Timers auf dem PiXtend eIO Analog One Gerät verdeutlichen.

Die Watchdog-Timer Timeoutzeit kann mit dem Befehl 600 geändert werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 600, Timeoutzeit 1 Sek.	#003:600:00:006*	#003:600:00:006*
Befehl 600, Timeoutzeit 8 Sek.	#003:600:00:009*	#003:600:00:009*

Tabelle 90: Beispiel: Watchdog-Timer Timeoutzeit setzen

Ist die Timeoutzeit eingestellt kann der Watchdog-Timer aktiviert werden.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 604, Watchdog-Timer an	#003:604:00:001*	#003:604:00:001*
Befehl 604, Watchdog-Timer aus	#003:604:00:000*	#003:604:00:000*

Tabelle 91: Beispiel: Watchdog-Timer ein- und ausschalten

Es ist eine weitere Funktionen des Watchdog-Timers alle Ausgänge abzuschalten, wenn er abläuft.

Beschreibung	Befehl an Slave	Antwort vom Slave
Befehl 601, Alle Ausgänge aus	#003:601:00:001*	#003:601:00:001*
Befehl 601, Alle Ausgänge bleiben unverändert	#003:601:00:000*	#003:601:00:000*

Tabelle 92: Beispiel: Watchdog-Timer Funktion einstellen

10.7.8. Beispiel - Status auslesen

An Hand des folgenden Beispiels wird gezeigt, wie man mit dem PiXtend eIO Protokoll den aktuellen Status des Analog One Gerätes auslesen kann. Es ist zu beachten, dass es sich beim Status Register um einen 16 Bit Wert handelt, der über ein separates Byte, Low-Byte und High-Byte, abgerufen werden muss. Die Auswahl des Bytes erfolgt über den Protokoll-Teil E/A-NUM. Das Low-Byte entspricht 00 und das High-Byte 01. Da das Analog One im High Byte keine Informationen enthält, kann dieses Byte ignoriert werden, ein Abrufen ist nicht notwendig.

Zuerst kommt der Befehl des Masters und dann die Antwort des Slaves.

- Status vom Gerät mit Adresse 3 auslesen, Low-Byte
 - Befehl: #003:200:00:000*
 - Antwort: #003:200:00:005*
 - Beschreibung: An den Slave mit Adresse 3 wird der Befehl 200 geschickt, den aktuellen Geräte Status an den Master zu schicken. Als Antwort erhalten wir das Low-Byte mit der Zahl 5 (Binär: 0000_0101). Der Watchdog-Timer ist aktiv und der Analogausgang 4 befindet sich in einer Überlastsituation (Overload).

10.8. PiXtend eIO Protokoll - Fehlernummern

Ist die Device ID korrekt und das Protokoll stimmt, besteht trotzdem die Möglichkeit, dass das Gerät eine Fehlermeldung bzw. Fehlernummer als Rückmeldung versendet.

Diese Nummern sind:

- 994 - Wert - Fehler
 - Der gesendete Wert ist für die gewünschte Aktion nicht passend und nicht zulässig. Wird ein digitaler Ausgang gesetzt, so ist als Wert nur die Auswahl zwischen 0 und 1 möglich. Versendet man den Wert 128, so erhält man diese Fehlernummer zurück. Das trifft ebenso auf das Analog One Gerät zu. Schickt man dem Gerät einen Befehl zum Ansteuern der analogen Ausgänge mit einem Wert zwischen 0 und 4095, ist alles in Ordnung. Verschiedet man den Wert 6300, erhält man diese Fehlernummer als Rückmeldung, der Wert ist unzulässig.
- 995 - E/A-Nummer Falsch
 - Die gewünschte E/A-Nummer ist falsch. Das Digital One Gerät hat 8 Eingänge und 8 Ausgänge (Zahlenbereich 0 bis 7). Wird ein Eingang mit der E/A-Nummer 9 abgefragt, ist das nicht möglich. Dies trifft analog auf die Befehle zu, die keine Hardware ansteuern, sondern Statusinformationen verarbeiten. Das Status Register (16 Bits groß) vom Digital One Gerät wird über 2 separate Bytes abgerufen, das Low-Byte und das High-Byte. Die Auswahl welches Byte abgerufen wird, geschieht über die E/A-Nummer. Bei 2 Bytes steht die die E/A-Nummern 0 und 1 zur Verfügung. Erhält das Gerät eine Byte-Auswahlnummer größer 1, so wird dieser Fehler ausgegeben. Das trifft für alle Befehle zu, die auf dem Gerät Einstellungen oder Daten lesen und schreiben. Prüfen Sie zusätzlich beim jeweiligen Gerät nach, welche Befehle es unterstützt und welche Daten abgerufen oder gesendet werden können.
- 996 - KOM Fehler - Kommando / Befehl wird nicht unterstützt
 - Schickt man dem Gerät eine unbekannt oder nicht unterstützte Befehlsnummer zu, erhält man diese Fehlernummer zurück. An einem Digital One Gerät lassen sich keine analogen Eingänge auslesen oder digitale Ausgänge setzen.

11. Abbildungsverzeichnis

Abbildung 1: RPi - Bluetooth ausschalten.....	27
Abbildung 2: CODESYSControl_User.cfg - Serieller Anschluss ttyAMA	28
Abbildung 3: CODESYS Modbus, Übertragung per Triggervariable (steigende Flanke) ausführen	33
Abbildung 4: Status LEDs "ERR", "COM" & "+5V" auf PiXtend eIO Geräten	37
Abbildung 5: Gerätekonfiguration - Übersicht DIP Schalter - Schema.....	39
Abbildung 6: DIP Block 1 - ADDRESS - Geräte-Adresse	40
Abbildung 7: DIP Block 2 - CONFIG - Serielle Konfiguration.....	47
Abbildung 8: DIP Block 2 - CONFIG - Mode - Modbus RTU	49
Abbildung 9: DIP Block 2 - CONFIG - Mode - PiXtend eIO Protokoll.....	49
Abbildung 10: DIP Block 2 - CONFIG - Busterminierung aus.....	50
Abbildung 11: DIP Block 2 - CONFIG - Busterminierung an.....	50
Abbildung 12: CODESYS V3.5 - Modbus RTU - Gerät meldet Fehler.....	77
Abbildung 13: CODESYS - Datei Menü.....	78
Abbildung 14: CODESYS - Neues Projekt.....	79
Abbildung 15: CODESYS - Standardprojekt - Geräteauswahl	79
Abbildung 16: CODESYS - Gerät anhängen Menü.....	80
Abbildung 17: CODESYS - Gerät anhängen Fenster	80
Abbildung 18: CODESYS - Gerät anhängen - Modbus Master	81
Abbildung 19: CODESYS - Alle Geräte angehängt	82
Abbildung 20: CODESYS - serielle Einstellungen.....	82
Abbildung 21: CODESYS - Modbus Master Einstellungen	83
Abbildung 22: CODESYS - Modbus Slave Einstellungen.....	83
Abbildung 23: CODESYS - Modbus Slave Kommunikationskanal anlegen	84
Abbildung 24: CODESYS - Modbus Slave Kommunikationskanal anlegen (2)	85
Abbildung 25: CODESYS - Modbus Slave Kommunikationskanalübersicht.....	85
Abbildung 26: CODESYS - Variablen aktualisieren	86
Abbildung 27: CODESYS - GPIO18 als Ausgang einstellen.....	87
Abbildung 28: CODESYS - GPIO18 als Variable anlegen.....	87
Abbildung 29: CODESYS - RS485 aktivieren	88
Abbildung 30: CODESYS - Modbus Bus läuft.....	89
Abbildung 31: CODESYS - Modbus Slave Coils & Discrete Inputs Übersicht	90
Abbildung 32: CODESYS - Digital Ausgang und Eingang aktiv.....	90
Abbildung 33: CODESYS - Datei Menü	91
Abbildung 34: CODESYS - Neues Projekt.....	92
Abbildung 35: CODESYS - Standardprojekt - Geräteauswahl.....	92
Abbildung 36: CODESYS - Gerät anhängen Menü.....	93
Abbildung 37: CODESYS - Gerät anhängen Fenster	93
Abbildung 38: CODESYS - Modbus Master anhängen.....	94
Abbildung 39: CODESYS - Alle Geräte angehängt.....	95
Abbildung 40: CODESYS - Modbus serielle Einstellungen.....	95
Abbildung 41: CODESYS - Modbus Master Einstellungen.....	96
Abbildung 42: CODESYS - Modbus Slave Einstellungen.....	96
Abbildung 43: CODESYS - Modbus Kommunikationskanal anlegen (1).....	97
Abbildung 44: CODESYS - Modbus Kommunikationskanal anlegen (2).....	98
Abbildung 45: CODESYS - Modbus Kommunikationskanalübersicht.....	98
Abbildung 46: CODESYS - Variablen aktualisieren	99
Abbildung 47: CODESYS - GPIO18 als Ausgang definieren.....	100
Abbildung 48: CODESYS - GPIO18 als Variable anlegen.....	100
Abbildung 49: CODESYS - RS485 aktivieren	101
Abbildung 50: CODESYS - Modbus Bus läuft.....	102
Abbildung 51: CODESYS - Modbus Slave Analogausgang 0 und Analogeingang 0.....	103
Abbildung 52: Digital One - Python Demo Programm.....	106
Abbildung 53: Analog One - Python Demo Programm.....	109
Abbildung 54: PiXtend eIO Protokoll - Kommunikation - Ablaufplan	113

12. Tabellenverzeichnis

Tabelle 1: Namensdefinition, Abkürzungen und Bezeichnungen.....	18
Tabelle 2: Modbus RTU Objekttypen.....	22
Tabelle 3: Auswahl an Baudraten und deren berechneter Übertragungsdauer	23
Tabelle 4: Modbus RTU - Kommunikationsgeschwindigkeit	24
Tabelle 5: Übersicht Digital One - 2 Kanal Zähler Funktion, Eingänge.....	31
Tabelle 6: LED „ERR“ - Signalisierung von Fehlerzuständen.....	38
Tabelle 7: PiXtend eIO: Auswahltablelle - Geräte-Adresse	40
Tabelle 8: PiXtend eIO: Auswahltablelle - Serielle Gerätekonfiguration.....	47
Tabelle 9: Digital One: Digitale Eingänge als Discrete Inputs einlesen.....	52
Tabelle 10: Digital One: Digitale Ausgänge als Coils lesen/schreiben	52
Tabelle 11: Digital One: Input Register	53
Tabelle 12: Digital One: Holding Register	53
Tabelle 13: Bits im Status Register - Unteres Byte (Low Byte).....	55
Tabelle 14: Bits im Status Register - Oberes Byte (High Byte).....	56
Tabelle 15: Bits im Fehler Register - Unteres Byte (Low Byte).....	58
Tabelle 16: Bits im Fehler Register - Oberes Byte (High Byte).....	59
Tabelle 17: Bits im Watchdog-Timer Register - Unteres Byte (Low Byte)	60
Tabelle 18: Watchdog-Timer Timeoutzeit Übersicht.....	60
Tabelle 19: Bits im Watchdog-Timer Register - Oberes Byte (High Byte).....	61
Tabelle 20: Bits im Config Register - Unteres Byte (Low Byte).....	62
Tabelle 21: Hyper-Logic Prozessor 0 - Verknüpfungsübersicht	63
Tabelle 22: Bits im Config Register - Oberes Byte (High Byte).....	63
Tabelle 23: Hyper-Logic Prozessor 1 - Verknüpfungsübersicht.....	64
Tabelle 24: Config Register - Bits: Zähler zurücksetzen	65
Tabelle 25: Bits im Counter Config Register 0 - Unteres Byte (Low Byte)	66
Tabelle 26: Bits im Counter Config Register 0 - Oberes Byte (High Byte).....	66
Tabelle 27: Counter Config Register 0 - Flankeneinstellung - Aufschlüsselung.....	66
Tabelle 28: Bits im Counter Config Register 1 - Unteres Byte (Low Byte).....	67
Tabelle 29: Bits im Counter Config Register 1 - Oberes Byte (High Byte).....	67
Tabelle 30: Counter Config Register 1 - Zählertyp - Aufschlüsselung.....	67
Tabelle 31: Bits im Hyper-Logic Config Register - Unteres Byte (Low Byte).....	68
Tabelle 32: Bits im Hyper-Logic Config Register - Oberes Byte (High Byte).....	68
Tabelle 33: Analog One: Overload Feedback Signale als Discrete Inputs.....	69
Tabelle 34: Analog One: Input Register	70
Tabelle 35: Analog One: Holding Register	71
Tabelle 36: Bits im Status Register - Unteres Byte (Low Byte).....	72
Tabelle 37: Bits im Status Register - Oberes Byte (High Byte).....	72
Tabelle 38: Bits im Fehler Register - Unteres Byte (Low Byte).....	72
Tabelle 39: Bits im Fehler Register - Oberes Byte (High Byte)	73
Tabelle 40: Bits im Watchdog-Timer Register - Unteres Byte (Low Byte).....	74
Tabelle 41: Watchdog-Timer Timeoutzeit Übersicht.....	74
Tabelle 42: Bits im Watchdog Register - Oberes Byte (High Byte)	75
Tabelle 43: Bits im Config Register - Unteres Byte (Low Byte).....	76
Tabelle 44: Bits im Config Register - Oberes Byte (High Byte).....	76
Tabelle 45: Modbus RTU Fehlernummern (Exception Codes).....	77
Tabelle 46: Digital One: Übersicht unterstützter Befehle - Basic	116
Tabelle 47: Bits im Error Register - Unteres Byte (Low Byte).....	117
Tabelle 48: Bits im Error Register - Oberes Byte (High Byte).....	119
Tabelle 49: Analog One: Übersicht unterstützter Befehle - Basic	123
Tabelle 50: Bits im Error Register - Unteres Byte (Low Byte).....	124
Tabelle 51: Bits im Error Register - Oberes Byte (High Byte).....	125
Tabelle 52: Digital One: Übersicht unterstützter Befehle - Advanced.....	128
Tabelle 53: Bits im Status Register - Unteres Byte (Low Byte).....	130
Tabelle 54: Bits im Status Register - Oberes Byte (High Byte).....	131
Tabelle 55: Digital One: Übersicht Watchdog-Timer Timeoutzeit	133
Tabelle 56: Watchdog-Timer Timeoutzeit setzen - Beispiele	133
Tabelle 57: Watchdog-Timer ein / aus - Beispiele	133
Tabelle 58: Watchdog-Timer Funktionen einstellen - Beispiele	134
Tabelle 59: Flankeneinstellung - Aufschlüsselung.....	135
Tabelle 60: Zähler ein- / ausschalten mit verschiedenen Einstellungen - Beispiele.....	135
Tabelle 61: Zählertyp - Auswahl.....	135
Tabelle 62: Zählertyp einstellen - Beispiele.....	136
Tabelle 63: Zähler Zählervorladewert - Beispiel 1	136
Tabelle 64: Zähler Zählervorladewert - Beispiel 2.....	136

Tabelle 65: Zähler zurücksetzen	137
Tabelle 66: Zähler zurücksetzen Beispiele	137
Tabelle 67: Hyper-Logic Prozessor 0 - Verknüpfungsübersicht	138
Tabelle 68: Hyper-Logic Prozessor 1 - Verknüpfungsübersicht	139
Tabelle 69: Hyper-Logic Prozessor 0 - Beispiel 1.....	139
Tabelle 70: Hyper-Logic Prozessor 1 - Beispiel 2.....	139
Tabelle 71: Hyper-Logic Prozessor 0 / 1 - Beispiel 3.....	140
Tabelle 72: Bits im Hyper-Logic Config Register - Unteres Byte (Low Byte)	140
Tabelle 73: Bits im Hyper-Logic Config Register - Oberes Byte (High Byte)	140
Tabelle 74: Beispiel: Zähler verwenden - Zähler 0.....	142
Tabelle 75: Beispiel: Zähler verwenden - Zähler 3	143
Tabelle 76: Beispiel: Zähler verwenden - Zähler 6 - 2 Kanal Zähler.....	144
Tabelle 77: Beispiel: Hyper-Logic Prozessor 0 verwenden.....	145
Tabelle 78: Beispiel: Hyper-Logic Prozessor 1 verwenden	145
Tabelle 79: Beispiel: Hyper-Logic Prozessor 0 und 1 ausschalten	145
Tabelle 80: Beispiel: Watchdog-Timer Timeoutzeit setzen	146
Tabelle 81: Beispiel: Watchdog-Timer ein- und ausschalten	146
Tabelle 82: Beispiel: Watchdog-Timer Funktionen einstellen.....	146
Tabelle 83: Analog One: Übersicht unterstützter Befehle - Advanced.....	148
Tabelle 84: Bits im Status Register - Unteres Byte (Low Byte).....	149
Tabelle 85: Bits im Status Register - Oberes Byte (High Byte).....	149
Tabelle 86: Analog One: Übersicht Watchdog-Timer Timeoutzeit.....	150
Tabelle 87: Watchdog-Timer Timeoutzeit setzen - Beispiele	150
Tabelle 88: Watchdog-Timer ein / aus - Beispiele	150
Tabelle 89: Watchdog-Timer Funktion einstellen - Beispiele	150
Tabelle 90: Beispiel: Watchdog-Timer Timeoutzeit setzen	151
Tabelle 91: Beispiel: Watchdog-Timer ein- und ausschalten.....	151
Tabelle 92: Beispiel: Watchdog-Timer Funktion einstellen.....	151