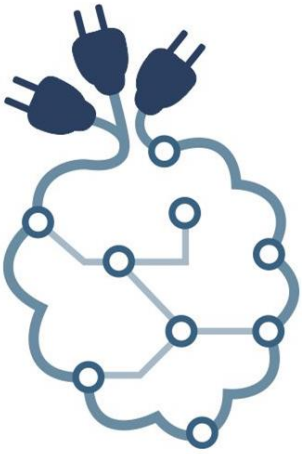
SOFTWARE MANUAL



PiXtend®

PiXtend eIO

kontron

Version History

Version	Date	Description	Editor
1.00	23.10.2019	Document created	RT
1.01	21.08.2023	Minor corrections, address and logo change	Tur

Table of Contents

1. About this Documentation

Keep this documentation in a safe place for future reference!

This documentation is part of the product and is to be kept for the entire duration of the product's usage. If the product is passed on or sold, this document must be handed over to the next user; this also includes any updates and/or changes to this documentation.

1.1. Scope of Application

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

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

The documents for the previous products can be found on our website at https://www.pixtend.de/downloads/

Notice:

If only the name PiXtend eIO is used in this documentation, the information mentioned applies equally to both products: PiXtend eIO Digital One and PiXtend eIO Analog One.

1.2. Copyright

1.3. Trademarks

- "Raspberry Pi" and its logo are registered trademarks of the Raspberry Pi Foundation - www.raspberrypi.org.
- "CODESYS" and its logo are registered trademarks of the 3S-Smart Software GmbH - www.codesys.com.
- "PiXtend", "ePLC" and its logo are registered trademarks of the Kontron Electronics GmbH – www.kontron-electronics.de.
- "AVR", "ATmega" and its logo are registered trademarks of the www.atmel.comMicrochip Technology Corporation www.microchip.com.
- "Debian" and "Raspbian" are registered trademarks of the Debian Project – www.debian.org.
- "I2C" and "I²C" are registered trademarks of NXP Semiconductors – www.nxp.com.
- "Arduino" is a registered trademark of the Arduino AG – www.arduino.cc.
- "Modbus" and its logo are registered trademarks of the Modbus Organization, Inc. - www.modbus.org.
- "Node-RED" and its logo are registered trademarks of the JS Foundation – js.foundation - www.nodered.org.
- "Java" and its logo are registered trademarks of the Oracle Corporation - www.oracle.com.
- The "C++" logo is a registered trademark of the Standard C++ Foundation – www.isocpp.org.
- "Python" and its logo are registered trademarks of the Python Software Foundation - https://www.python.org/psf/.
- "Netbeans" and its logo are registered trademarks of the Apache Software Foundation - www.netbeans.org.
- C# was developed and published by Microsoft; the associated logo was published by Chris McKee under the CC BY-SA 4.0 license.
  https://www.microsoft.com - https://creativecommons.org/licenses/by-sa/4.0/

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

1.4. Symbols

1.4.1. General symbols

NOTICE	NOTICE indicates a particular characteristic.
⚠CAUTION	CAUTION indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.
⚠DANGER	DANGER indicates a hazardous situation which, if not avoided, will result in death or serious injury.

1.4.2. Special warning symbols

HOT surface!
Do NOT touch! Allow to cool before servicing.

Electric Shock!
This symbol and title warn of hazards due to electrical shocks (> 60V) when touching products or parts of products. Failure to observe the precautions indicated and/or prescribed by the law may endanger your life/health and/or result in damage to your equipment.

ESD Sensitive Device!
This symbol and title inform that the electronic boards and their components are sensitive to static electricity. Care must therefore be taken during all handling operations and inspections of this product in order to ensure product integrity at all times.

2. Important Information

This chapter contains information on legal principles, the intended use of the product described here, the technical condition at delivery and important safety instructions.

2.1. Subject to Change

Kontron Electronics GmbH reserves the right to revise or amend this documentation in whole or in part if this serves the technical progress or if existing software components are changed or new ones have been created. The latest version of this documentation is always available at https://www.pixtend.de/downloads/.

2.2. Intended Use

PiXtend eIO devices fulfill the function of input and output units for electrical signals. Sensor signals can be detected and evaluated, and the devices can activate actuators. The specifications of the inputs and outputs comply with the international standard for programmable logic controllers, PLC - IEC/EN 61131-2. In addition, the devices comply with the EMC requirements of Zone B (general industrial environment) from the cited standard.

Communication between the PiXtend eIO and other devices is via a serial interface (RS485/EIA485). It is a bus system in which up to 32 participants can communicate with each other. The PiXtend eIO fulfills the function of a "slave" in such a bus system (also known as a network). This means that information can be sent to or requested from devices. A slave device always only becomes active on the bus if it is addressed by a "master", i.e., the master makes a communicate request. There is not a direct data transfer between the slave devices. Possible master devices for the PiXtend eIO devices are personal computers (PC), industrial controllers (PLC, IPC), Raspberry Pi computers, Arduino computers, embedded devices, specially equipped smartphones and tablets and many more computer systems that have an RS485 interface or can be retrofitted with this kind of interface.

Of particular importance is the combination of PiXtend eIO devices with the PiXtend V2 -L- controller. This is the reference bus master recommended by Kontron Electronics. The PiXtend eIO devices are able to use two protocols - "Modbus RTU" and the "PiXtend eIO ASCII protocol". Basic settings for operation on the bus (baud rate, parity, stop bits, address, protocol and terminator) are set using mechanical switches on the device before use. The programming of inputs, outputs and special functions can be done, among other things, with the software "CODESYS V3" from the company 3S Smart Software Solutions GmbH.

Kontron Electronics also offers other automation, IT and home automation programming languages and systems that customers can use to control PiXtend eIO devices (C, C ++, C #, Python, Java, FHEM, Node-RED). Instructions and examples have been created by Kontron Electronics for this purpose. Both protocols are disclosed and can be implemented by the customer - in their own responsibility - in other programming languages and systems.

Unlike previous Kontron Electronics products, the PiXtend eIO devices are delivered exclusively as finished devices that are ready for use and do not require final assembly by the customer.

PiXtend eIO devices are designed for dry indoor environments - protection classes IP20 (ePLC Pro), IP00 (ePLC Basic). Operation outdoors and in humid/wet rooms is not permitted, except when devices are installed in a suitable housing. The devices are not designed for hazardous areas or safety critical systems/installations with particularly high requirements.

PiXtend eIO devices can be used in industrial/commercial environments, in educational facilities and in residential areas alike.

Intended uses are:

- Mechanical, plant and equipment engineering and construction
- Testing equipment
- Laboratory and production automation
- Home automation/smart home
- Prototyping in industry (development departments)
- Use in education (technical schools, colleges and universities)
- IoT (Internet of Things), IIoT (Industrial Internet of Things) concepts and solutions
- Industry 4.0 concepts and solutions
- Mobile applications - auxiliary functions in the automotive sector

Apart from this, PiXtend elO devices are suitable for all persons aged 14 and over who have read and understood the safety data sheet and the manuals.

Use in educational facilities must be supervised by qualified and authorized personnel. Power supplies and accessories used must be approved for the country in which the PiXtend elO system is to be installed and used.

2.3. Technical Condition

Each PiXtend elO device is supplied with a pre-defined configuration, independent of its model:

PiXtend elO Digital One

- All digital inputs are configured for 24V (no jumper is set).
- Address: 1, Baud rate 19200 baud, Parity bit: even, Stop bit: 1 (8E1)
- Termination inactive, Mode/Protocol: Modbus RTU
- The micro-controller firmware is always the latest version released by Kontron Electronics. The current version can be found on our website.

PiXtend elO Analog One

- All analog voltage inputs are configured for 0 to 10V (no jumper is set).
- Address: 3, Baud rate 19200 baud, Parity bit: even, Stop bit: 1 (8E1)
- Termination inactive, Mode/Protocol: Modbus RTU
- The micro-controller firmware is always the latest version released by Kontron Electronics GmbH. The current version can be found on our website.

Each device is available in two different versions:

Basic

- Basic version without housing

Pro

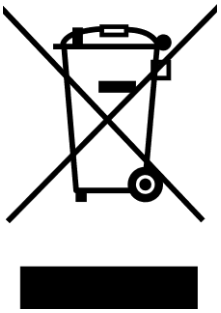
- Professional version with stainless-steel cover and DIN rail housing

If you need another version or a different hardware and software combination, please send your request directly to us (info@pixtend.de).

2.4. Certifications

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

Warning:
Changes and modifications to the product, as well as a non-compliance with the information contained in the manuals and safety data sheets, will lead to the loss of certification.



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

2.5. Safety information

Read the complete document and the safety and connection instructions before connecting or operating PiXtend eIO devices. Save this document even after you have set up all components.

⚠CAUTION

Kontron Electronics GmbH does not accept any liability for damage of any kind resulting from disregarding the data sheets and operating instructions. The guarantee and warranty claim will be void if the data sheets and operating instructions are disregarded.

- PiXtend eIO devices may only be operated with the specified voltage (24V DC ±20%) and a power supply with the VDE and CE mark (for Europe). The power supply must comply with the legal requirements of the country in which PiXtend eIO is used.
- The device is only designed for use in dry and clean rooms and is not suitable for outdoor use or in damp areas.
- The permissible operating temperature is between 0° C and 60° C.
- PiXtend eIO devices as well as all cables, connectors and power supplies must be kept away from liquids.
- PiXtend eIO devices must not be used in the vicinity of flammable liquids, gases or dusts.
- Only original or recommended spare parts may be used for repairs. If anything is unclear, contact Kontron Electronics – support@pixtend.de
- No 230V, 115V alternating voltage or any other dangerous voltage greater than 50V may be connected. **Caution: Dangerous to life!**
- PiXtend eIO and accessories are to be kept out of reach of children under 14.
- The operation in schools, hobby workshops and educational facilities is to be supervised by trained personnel.

2.6. Disclaimer

The information contained in this documentation has been compiled, checked and tested with the greatest possible care with the software and hardware described herein. Nevertheless, discrepancies cannot be ruled out completely. Kontron Electronics GmbH is not liable for any damages that may result from the use of the software, software components, hardware or the steps described in this documentation.

2.7. Contact Information

Our postal address:

Kontron Electronics GmbH
Max-Planck-Str. 6
D-72636 Frickenhausen

How to reach us:

Telephone: +49 7022 40570
info@kontron-electronics.de
www.kontron-electronics.de

2.8. Assistance

If you have any questions, please feel free to contact us by e-mail (support@pixtend.de). You will receive an answer as soon as possible.

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

3. Overview

PiXtend elO (economic Input and Output) is the I/O extension system for those looking for inexpensive, general-purpose inputs and outputs - digital and analog. The data connection is made with the robust RS485 bus. If not available, it can be retrofitted to any computer system.

The elO devices and their inputs and outputs comply with the industry standard IEC 61131-2 and are therefore intended for rough environment use in laboratories, industry and commercial enterprises. Compliance with the worldwide industry standard has great advantages in terms of stability and insensitivity to interference. All common sensors and actuators are easily connected to the PiXtend elO.

Innovative features, particularly fast digital "Hyper Logic", offer additional added value to your project. Simple logical connections between inputs and outputs can thus be carried out by the devices themselves, without the need for data transmission via the bus. Digital inputs can also be used to count events. A watchdog timer monitors the data transmission secured by the CRC checksum and, if necessary, puts the outputs in an adjustable, safe state.

The name "PiXtend" is known from our Raspberry Pi based PLC, IoT devices. The elO-System fits optically, haptically and functionally perfectly to these controls, but it can also be connected to almost any other computer system and embedded device.

In terms of communication and protocols, PiXtend elO is simple and user-friendly. With Modbus RTU support, elO devices can also be integrated into existing systems. The protocol is safe and has been tried and tested for decades. In addition, the elO devices can also be addressed with the rather simple and lean "PiXtend elO protocol", which transmits the data in plain text (ASCII). We offer open source sample programs for C, C #, Python, CODESYS (61131-3) and Java. The integration into own applications is a breeze.

The basic settings can be made directly and easily on the device, without software and without prior knowledge.

4. Requirements

These are the minimum requirements that must be met in order to setup or use a PiXtend eIO device.

- System requirements for development computers (PC)
  - Microsoft Windows 7/8/10/11 (32/64 bit), Linux, macOS
  - Sufficient disk space for installing new programs
  - The necessary rights (e.g., administrator rights) that allow installation of a new program
  - Serial RS485 connection or USB-to-Serial RS485 dongle, if the device is to be controlled directly from the computer. Otherwise a PLC or an embedded device with serial RS485 connection is required.
  - Terminal or console program such as "Putty" with access permission to the serial ports. This can be helpful if the PiXtend eIO protocol is to be used for direct entry of commands.
  - Modbus RTU requires Modbus RTU Master software on the PC.

- System requirements for PLC
  - Serial RS485 connection
  - If there is no RS485 interface, alternatively a USB-to-Serial dongle or can be used to retrofit a serial interface or a bus converter which provides the PLC with an RS485 interface can used.
  - Modbus RTU Master Block (or function) for Modbus RTU communication or serial communication using the simple and free PiXtend eIO ASCII protocol.

- Required hardware
  - PiXtend eIO device (www.pixtend.de)
  - Modbus RTU - RS485 compatible cable
  - 24-volt power supply

You should also have basic knowledge of one or more of the following programming languages or programming tools to use PiXtend eIO devices efficiently. The following list represents a small excerpt. Any programming language can be used that provides access to a serial interface on the desired target system.

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

For these languages and programming systems, there are various examples and demo projects in the download section on our website. Those who also work on a PC with Visual Studio 2017 or later can use a Windows example with a graphical user interface. Without going deeper into the programming of the PiXtend eIO protocol, this program can be used to send simple commands to a PiXtend eIO Analog One and/or PiXtend eIO Digital One device. The state of the digital and analog outputs can be changed, or the state of the respective inputs can be queried.

5. Licenses

All examples and programs for using the PiXtend elO system were published by Kontron Electronics GmbH under the MIT license and can be used and modified freely. The use in one's own projects as well as for commercial use, in connection with PiXtend elO products, is permitted.

There are currently sample projects for the PiXtend elO protocol for the programming languages "C#", Python, Node-RED, Java (Netbeans 11), CODESYS V3.5 and "C" for the Raspberry Pi computer.

Modbus RTU examples are available for the programming languages Python, C for the Raspberry Pi computer and CODESYS V3.5.

The files may be freely distributed and used in any number; the license notice and the company name, Kontron Electronics GmbH, must be retained and may not be removed.

If you use additional libraries or packages like pySerial, pyModbus or libModbus, please note the respective license terms.

6. Basic knowledge

The Basic Knowledge chapter contains important information about the PiXtend eIO system and should always be read carefully and completely before starting to operate PiXtend eIO devices.

6.1. Name definitions, abbreviations and labels

The following table gives an overview of different names, abbreviations and labels used in this documentation. This table is intended to serve as a guide to help better understand the documentation.

Label	Description
A	A capital A stands for the word output (German: *Ausgang*). This abbreviation may occur in connection with analog and/or digital outputs.
E	A capital E stands for the word input (German: *Eingang*). This abbreviation may occur in connection with analog and/or digital inputs.
I	A capital I stands for the word input.
O	A capital O stands for the word output.
AI	AI stands for analog input.
AO	AO stands for analog output.
DI	DI stands for digital input.
DO	DO stands for digital output.
HIGH	In this documentation, the term HIGH, to be found as HIGH level or HIGH signal, describes the state when a corresponding electrical signal is applied to a digital input, which falls within the range of a logical 1, also called On or True. The exact electrical regulations can be found in the PiXtend eIO Hardware Manual.
LOW	In this documentation, the term LOW, to be found as LOW level or LOW signal, describes the state when a corresponding electrical signal is applied to a digital input, which falls within the range of a logical 0, also called Off or False. The exact electrical regulations can be found in the PiXtend eIO Hardware Manual.
On	Describes the state that a PiXtend eIO device function is switched on or active. Other names with the same meaning are the number 1, the logical state 1 or the Boolean expression True.
Off	Describes the state that a PiXtend eIO device function is switched off or is inactive. Other names with the same meaning are the number 0, the logical state 0 or the Boolean expression False.
1	Refers to On.
0	Refers to Off.
True	Refers to On. Refer to the Wikipedia entry on: Propositional calculus
False	Refers to Off. Refer to the Wikipedia entry on: Propositional calculus
MC	Micro-controllers (µController, µC, MCU) are semiconductor chips that contain a processor and peripheral functions at the same time. In many cases, the working and program memory is partly or completely located on the same chip. A micro-controller is a single-chip computer system. The term system-on-a-chip or SoC is used for some micro-controllers. Modern micro-controllers often contain complex peripheral functions such as CAN (Controller Area Network), LIN (Local Interconnect Network), USB (Universal Serial Bus), I²C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), serial or Ethernet interfaces, PWM outputs, LCD controllers and drivers and analog-to-digital converters. Some micro-controllers have programmable digital and/or analog or hybrid function blocks. (Wikipedia, https://en.wikipedia.org/wiki/Microcontroller, accessed in October 2019)
Power Cycle	This is when a PiXtend eIO device is disconnected from the power supply and after a short waiting period is supplied with power again. This process resets all states in the PiXtend eIO device, and all configuration settings are applied on restart.

Label	Description
Watchdog	The term watchdog (also known as a watchdog timer) describes a function for failure detection of a digital system, mainly in control applications. If a potential malfunction is detected, this is signaled to other components in accordance with the system agreement (e.g., switchover to a redundant system), a suitable jump instruction or a reset is initiated for automatic correction of the failure or a safe shutdown is initiated. (Wikipedia, https://en.wikipedia.org/wiki/Watchdog_timer, accessed in October 2019)
Watchdog timer	Refer to Watchdog.
Message	In the network technology on which this documentation is based, a message is understood as a network packet. This network packet may be, for example, a command or query for a terminal on a data bus. When the terminal receives a message, it can evaluate it and respond to it. More detailed information on this topic is available on Wikipedia at https://en.wikipedia.org/wiki/Network_packet (accessed in October 2019).
Telegram	Refer to Message.
RS485	Also known as EIA-485, it is the designation for a physical interface for serial data transmission. On the basis of this interface, a wide variety of devices can be connected to each other and also enable communication between the devices. More information on RS485 is available on Wikipedia at https://en.wikipedia.org/wiki/RS-485 (accessed in October 2019). Information about Bus systems: https://en.wikipedia.org/wiki/Bus_(computing)
RS232	This designates a serial interface for data transmission between two devices or participants. More information on RS232 is available on Wikipedia at https://en.wikipedia.org/wiki/Serial_port (accessed in October 2019).
USB-to-RS485	This designates a special adapter with which a computer can add an RS485 using a USB interface. Refer to RS485 and Wikipedia at https://en.wikipedia.org/wiki/USB for more information on USB.
Baud rate	The baud rate describes the speed of transmission of a symbol per second in a physical medium. The baud rate must always be the same for the sender and receiver. More information is available on Wikipedia at https://en.wikipedia.org/wiki/Baud (accessed in October 2019).
Parity	The parity bit is added to a sequence of bits for error detection and corresponds to the total number of 1-bits in the sequence, even or odd. The parity of a sequence of bits is called even if the number of bits in the sequence is even, otherwise odd. More information is available on Wikipedia at https://en.wikipedia.org/wiki/Parity_bit (accessed in October 2019).
even	Refer to parity
odd	Refer to parity
Stop bit	One to two stop bits are used in asynchronous serial data transmission to enable the receiver to synchronize to each transmitted character. The start and stop bits always are the same but have different logical levels. A start bit is sent before the data word to be transmitted, the stop bit or bits after it. More information is available on Wikipedia at https://en.wikipedia.org/wiki/Asynchronous_serial_communication (accessed in October 2019). Refer also to parity and baud rate.
Overflow	The overflow bit indicates to counters that they have reached the end of their number space or counted beyond it. This applies to the positive and negative counting. Depending on the system and device, an overflow event can cause an error or, as is often the case in automation technology, the counters simply continue counting. Not every system has and provides an overflow bit.

Label	Description
	More information is available on Wikipedia at https://en.wikipedia.org/wiki/Integer_overflow (accessed in October 2019).
Rising edge	This changes a signal from a logical 0 to a logical 1.
Falling edge	This changes a signal from a logical 1 to a logical 0.
RE	Abbreviation for rising edge
FE	Abbreviation for falling edge
RE	REMOVE Rising edge
FE	REMOVE Falling edge

Table 1: Name definitions, abbreviations and labels

6.2. Definition “Safe State”

The PiXtend elO system has a user-enabled watchdog timer. If the user has activated this function and a communication failure occurs, then the watchdog timer is triggered after the user-defined waiting time. If the waiting time has been passed and the watchdog timer has been triggered, the micro-controller enters a "safe state." This state has the following characteristics:

- All digital and analog outputs remain in their current state (default setting). This can be changed by the user so that all outputs are switched off or set to 0V and 0mA.
- The ERR LED flashes 3 times briefly and 3 times longer, then there is a 1-second pause before the sequence repeats.
- The micro-controller or the PiXtend elO device has to be restarted (power cycle).
- It is no longer possible to communicate with the device; this means it is also not possible to do a remote reset. The device must be disconnected from the power supply on site.

6.3. Modbus - Short introduction and overview[1]

Modbus is communication protocol based on a master/slave protocol - client/server architecture. Modbus has become a *de facto* standard communication protocol because it is openly published and royalty-free. Since 2007, the Modbus TCP version is part of the IEC 61158 standard.

A master (a PC) and several slaves (measurement and control systems) can be connected via Modbus. There are two versions: One for the serial interface (EIA-232 and EIA-485) and one for Ethernet.

We distinguish between two modes of operation for data transmission:

- Modbus RTU
- Modbus TCP

Each bus device must have a unique address. The address 0 is reserved for a broadcast. Each device is allowed to send messages via the bus. The communication is always initiated by the master and an addressed slave responds.

[1]Text from Wikipedia, as of August 2019, https://en.wikipedia.org/wiki/Modbus

Read and write accesses are possible for the following object types:

Object type	Access	Size
Single input/output coil	Read and write	1 bit
Single discrete input	Only read	1 bit
(Analog) inputs "input register"	Only read	16 bits
(Analog) inputs/outputs "holding register"	Read and write	16 bits

Table 2: Modbus RTU object types

Modbus RTU (RTU: Remote Terminal Unit) transmits the data in binary form. This ensures a good data throughput. However, the data cannot be evaluated directly by humans, but must first be converted into a readable format. The data transmission is also protected with a cyclic redundancy check (often called "CRC"), so that both the master and a slave can check the accuracy of the data received. If an error is detected, the data is discarded.

6.4. Modbus RTU and PiXtend eIO Protocol Definitions

Both communication protocols are based on a master/slave architecture. The master must always send a request (message or telegram) to a slave; only this one slave may respond. Slaves cannot and must not communicate with each other; the master controls the entire communication and distributes the information.

With the Modbus RTU protocol for master and slave, there are other names that can lead to confusion.

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

Note these different names when reading other literature, notes or information regarding Modbus RTU.

6.5. PiXtend eIO Protocol - Short introduction and overview

The PiXtend eIO protocol is a simple plain text protocol from Kontron Electronics GmbH designed to be transmitted via a serial bus.

The basic idea behind the PiXtend eIO protocol is that you can connect to the PiXtend eIO devices directly from a PC, e.g. via an USB-to-RS485 dongle. The user can simply use a serial program or a terminal program with access to a serial interface to send a command to the device, as the commands can be entered in plain text using the keyboard.

Various commands are available to read and set the digital and analog inputs and outputs.

If an RS485 bus system is available, but there is no communication via Modbus RTU, then the PiXtend eIO protocol is perfectly suitable. It adds digital and analog inputs and outputs to the existing system. In addition, the PiXtend eIO protocol is particularly easy to integrate into one's own programs. There is a variety of examples and demo projects to help support quick implementation.

6.6. RS-485/EIA-485 - Serial communication overview

RS-485[2], also referred to as EIA-485, is an industry standard for a physical interface for asynchronous serial data transmission. The symmetrical line increases the electromagnetic compatibility.

RS-485 uses a pair of wires to transmit the inverted and non-inverted levels of a 1-bit data signal. At the receiver, the original data signal is reconstructed from the difference between the two voltage levels. This has the advantage that common-mode interference does not affect the transmission and thus the interference immunity is increased.

Unlike other buses, RS-485 only defines the electrical interface conditions. The protocol can be selected for specific applications. As for example with the PiXtend eIO devices, these can be addressed with the Modbus RTU protocol or alternatively with Kontron Electronics' own PiXtend eIO protocol. Since the transmission is serial and depends on the selected speed, time must be allowed for the communication from the devices with their response.

The information in the table is based on a message to a Digital One and an Analog One device with the PiXtend eIO protocol. The response of the slave is not taken into account.

Baud rate	Time per bit	Time per byte	D-One message (16 bytes)	A-One message (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

Table 3: Selection of baud rates and their calculated transmission time

With the PiXtend eIO protocol, the master receives a response from the slave after ten milliseconds (plus the transmission time from table 3), this circumstance allows the use of PiXtend eIO with PiXtend V1.x boards.

NOTICE

There must always be a sufficient break between two messages or telegrams. The master sends a request/command to the slave and waits for the response. Once the master has received the response from the slave, a pause must occur.

The following times are recommended:

- Slave response waiting time: 1000 ms
- Pause time for baud rates below 9600: >= 4 ms
- Pause time for baud rates above 9600: >= 2 ms

This recommendation applies to the PiXtend eIO protocol and is independent of the number of bus devices.

For Modbus RTU, see Modbus specification 1.02, chapter Modbus Message RTU Framing.

[2]Information from Wikipedia https://en.wikipedia.org/wiki/RS-485, accessed in August 2019.

6.7. Modbus RTU speed

When using the Modbus RTU protocol, little data is generated depending on the function code. The binary coding makes the process more efficient than with the PiXtend elO protocol for the same task (see previous page). The table below shows how fast each device communicates and how fast and how often each device can be polled per second.

Device	FC	Number of I/Os	Updates	Comment
Digital One	02/15	8 I or 8 O	166/s	Read 8 inputs or write 8 outputs
Digital One	02/15	8 I and 8 O	100/s	Read or write all I/Os
Analog One	04/06	1 I or O	100/s	One register contains 16 bits (2 bytes); read 1 input or write 1 output
Analog One	04/16	8 I and 6 O	41/s	Read or write all I/Os

Table 4: Modbus RTU – communication speed

The update and cycle times per second mentioned here always refer to just one single device with a baud rate of 115200 with eight data bits, no parity and one stop bit.

The times were determined under CODESYS V3.5 SP14 on a Raspberry Pi without a concurrent application or visualization (HMI). When using extended functions or larger programs, the times mentioned may differ, especially if there are several bus devices.

The update frequency under CODESYS V3.5 depends essentially on three factors. On the one hand, the bus cycle task of the Modbus master determines how fast the Modbus protocol is processed and the associated data is processed. Data includes the request to the slave and its answer.

Furthermore, the set cycle time of the respective Modbus RTU channel plays a role; this time setting determines the interval at which communication with a slave takes place. By default, CODESYS V3.5 always suggests 100 ms.

The last point is the setting "Time between Frames [ms]" in the CODESYS V3.5 ModBus Master. This setting is usually ten milliseconds; this means the Modbus master always sends a message to a slave every ten milliseconds, no matter what is configured as a trigger in the Modbus slave channels.

For a better understanding of the timings under Modbus RTU, refer to Modbus Specification 1.02, chapter Modbus Message RTU Framing.

6.8. Prepare the serial interface for one's own SD card image

If you do not want to use our prepared CODESYS SD card images or prefer to program in Python or C and use our Basic image, then the following description will help you to set the serial interface of the Raspberry Pi computer accordingly. This section is also important if you are using your own SD card image with the Raspbian operating system.

To use the serial port of the Raspberry Pi computer for your own purposes, it must first be activated and configured. If you have not previously used the serial interface, the following commands activate and adjust it; it does not send any data by itself. Access to the Linux console via the serial port is also disabled by this procedure. This conversion is important to ensure correct and consistent data transmissions.

The serial interface of the Raspberry Pi is activated or adjusted using the raspi-config program:

```
sudo raspi-config
```

Switch to the following in the program:

```
Interfacing Options --> Serial --> <No> --> <Yes> --> <Ok>
```

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

```
enable_uart=1
```

The UART or serial interface is activated and the login shell is deactivated. The program raspi-config does all the work.

Then restart the Raspberry Pi computer and use the serial port exclusively in your own applications.

A restart can be performed with the following command:

```
sudo reboot
```

NOTICE

The steps above are intended for the use of a PiXtend V2 -L- board with the serial RS485 interface to communicate with the PiXtend eIO devices. The PiXtend V2 -S- and PiXtend V1.x need a USB-to-RS485 dongle to communicate with PiXtend eIO devices. The serial interface of the PiXtend V2 -S- and PiXtend V1.x are not Modbus RTU capable.

If you use the PiXtend eIO protocol, you can use the RS485 interface on the PiXtend V1.x. PiXtend eIO devices take a ten-millisecond pause before sending the response to the master. This gives the user time to switch the transceiver from transmit to receive.

6.9. Adding a USB-to-RS485 dongle to V2 -S-

The PiXtend V2 -L- already has an RS485 interface, which can be activated by setting the GPIO 18 of the Raspberry Pi. The Modbus RTU protocol can therefore be used very easily.

PiXtend V2 -S- requires an additional USB-to-RS485 dongle, which provides the appropriate hardware to use Modbus RTU. PiXtend V2 -S- only has a RS232 interface. A USB-to-RS485 dongle can be purchased from our shop.

To gain access to the additional serial interface under CODESYS V3.5 and to be able to use Modbus RTU, the following must be observed. The CODESYS SD card images are set by default to always use the Raspberry Pi serial port with the device name ttyAMA0 (also known as PL011 or serial0). If you want to use the serial interface provided by the USB dongle, the device name is usually ttyUSB0. The zero can be another number if another dongle is already plugged into the Raspberry Pi.

We recommend the following steps to use the RS485 USB dongle under CODESYS.

- Turn on the PiXtend and start CODESYS
- Log in via console application (e.g. Putty) or directly on the Raspberry Pi via monitor, mouse and keyboard
- Now plug in the RS485 USB dongle to the Raspberry Pi
- Enter the following in the console: dmesg
- A lot of information is displayed; look at the end of the text and look for a line of text that looks something like this:
  - FTDI USB Serial Device converter now attached to ttyUSB0
- The RS485 USB dongle from our shop registers as FTDI USB Serial Device. If you have another dongle, the name may be different. At the end of the line is the device name with which the RS485 USB dongle was registered on the Raspberry Pi. In the example, the name is ttyUSB0; instead of the zero there can be another number. Note this device name.
- Note: The number behind the device name ttyUSB must be between "0" and "98"; otherwise, the RS485 USB dongle cannot be used in CODESYS.
- In the next step the CODESYS user configuration file must be adjusted; here we define which serial interface CODESYS uses:
  - Execute the following command: sudo nano /etc/CODESYSControl_User.cfg
  - In the file CODESYSControl_User.cfg search for the section [SysCom]
  - Change the section as follows from:

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

  - to

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

  - remove the two semicolons, change the device name from ttyS to ttyUSB; do not add a number at the end of the device name, CODESYS will do this later itself.
  - `Save the changes by pressing Ctrl+O → Enter`
  - Exit the editor by pressing Ctrl+x
  - Reboot the Raspberry Pi with this command: sudo reboot

Now access the new serial interface under CODESYS V3.5. Please note that CODESYS starts with the COM port numbering at one and not at zero. If your RS485 USB dongle is registered in the Raspbian operating system with the device name ttyUSB0, use COM port 1 in CODESYS. If the dongle is registered with the device name ttyUSB3, use COM port 4 in CODESYS etc.

Please refer to the PiXtend V2 Software Manual, Chapter 7, sections Preparation Linux and Adjusting the Interface Settings for more information.

6.10. PiXtend V2 -L- – Checking the serial interface

To work with the PiXtend elO devices without restrictions, we recommend using the full serial interface of the Raspberry Pi, also called PL011 or serial0, with the device name ttyAMA0. Only this interface allows the parity bit to be used in serial communication.

All our SD card images from version 2.1.6.0 for the Basic image and the PiXtend V2 -S- CODESYS image or version 2.1.1.1L for the PiXtend V2 -L- CODESYS image use the ttyAMA0 interface for communication.

If you are using an older image or an older Raspbian version, or you are not sure whether you are already using a new image, it is best to look in the config.txt file in the Raspbian boot partition.

Under Windows, insert the SD card into a card reader, open the SD card with Windows Explorer and look in the config.txt file to see if it contains the entry: dtoverlay=pi3-disable-bt (starting with Raspberry Pi OS Bullseye use dtoverlay=disable-bt instead) at the end of the file.

Use the following command to check it directly on the Raspberry Pi:

→ sudo nano /boot/config.txt

At the end of the file, there is the entry dtoverlay=pi3-disable-bt (starting with Raspberry Pi OS Bullseye use dtoverlay=disable-bt instead). If it is available, then you are already using the device ttyAMA0 as serial interface and do not need to do anything else.

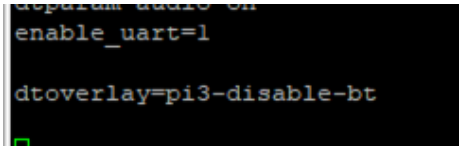


Figure 1: RPi - Turning off Bluetooth®

If this entry does not exist, insert it and carry out the following steps:

- sudo nano/boot/config.txt
- Go to the end of the file
- Add: dtoverlay=pi3-disable-bt (starting with Raspberry Pi OS Bullseye use dtoverlay=disable-bt instead)
- Save the changes by pressing Ctrl+O → Enter
- Exit the editor by press Ctrl+X
- Perform a restart: sudo reboot

In order to control PiXtend elO devices with CODESYS V3.5, a further adjustment is necessary after completing the previous change to use the correct serial interface.

- Execute the following command: sudo nano /etc/CODESYSControl_User.cfg
- In the file CODESYSControl_User.cfg select the section [SysCom]
- Change the section as follows from:

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

- to

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

- Remove the two semicolons, change the device name from ttyS to ttyAMA. Never add a number to the end of the device name; CODESYS will do this later itself.
- Save the changes by pressing Ctrl+O → Enter

- Exit the editor by pressing Ctrl+X
- Reboot the Raspberry Pi with this command: sudo reboot

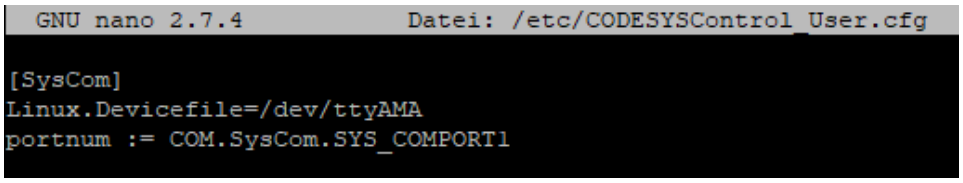


Figure 2: CODESYSControl_User.cfg - serial connection ttyAMA

Please remember to deactivate the login shell on the serial0/ttyAMA0 interface. Refer to chapter 6.8 Prepare the serial interface for one's own SD card image.

6.11. PiXtend eIO Watchdog timer

6.11.1. Watchdog timer behavior

If the watchdog timer on a device is activated and the time expires, the device goes into a safe state. In the default setting, all outputs retain their current state, no action is taken. In addition, the ERR LED displays a blink code.

The user can adjust the following functions of the watchdog timer:

- Watchdog timer timeout (16 ms to 8 seconds)
  - Modbus RTU protocol default setting: Depends on the holding register, see Table 18: Watchdog timer timeout overview
  - PiXtend eIO protocol default setting: 4 seconds timeout
- Active the switch-off of all outputs and set to 0V/0mA
- Only Digital One device:
  Set digital output 7 when the watchdog timer expires
  - Default setting: Digital output 7 is set from LOW to HIGH, in the normal state the output is always LOW.
  - Alternative: Digital output 7 is set from HIGH to LOW, i.e., inverted, in the normal state the output is always HIGH.
  - If the user has activated this option of the watchdog timer, digital output 7 is no longer available for one's own applications but is used exclusively by the watchdog timer.

6.11.2. Watchdog timer mode

Each PiXtend eIO device has a watchdog timer that can be used to monitor communication on the bus or with the device itself.

The watchdog timer basically has two modes.

In the first mode the watchdog timer "only" monitors the communication on the bus. This means that if data is sent by the master, another device (slave) on the bus responds and data exchange takes place. If the user does not change the watchdog timer configuration, this mode is preset

In the second mode, the watchdog timer monitors the actual communication with the PiXtend eIO device itself. The Modbus RTU Master communicates with the device at least once within a specified time and sends, for example, a message that reads the input registers or sets outputs. Only if the message is directly intended for the device, the watchdog timer is reset and the waiting time of the watchdog timer starts again. Now the master sends the next message to the device within the waiting time; after each received message the cycle starts again.

The watchdog timer mode is useful to determine if the Modbus RTU Master is fast enough to continuously communicate with the PiXtend eIO device within the specified time period. In large Modbus networks, up to 32 devices are allowed on one Modbus RTU bus. When the master communicates with many PiXtend eIO devices, it could happen that the device is no longer or only very rarely communicated with. The maximum possible waiting time for the watchdog timer is 8 seconds.

6.11.3. Watchdog timer output

Note: This section only applies to PiXtend eIO Digital One devices.

The watchdog timer can be configured to set the digital output 7 when it expires. This happens regardless of whether the user wants the watchdog timer to switch off all outputs. Output 7 would still remain active, even if the remaining seven outputs became inactive.
This function can be used to switch on a signal lamp or horn, for example.

Additionally, this "Watchdog timer has expired signal" can be inverted. In normal operation the digital output 7 is always HIGH. If the watchdog timer expires and is triggered, the signal changes to LOW. It is conceivable here that this circumstance could be used to give another device a ready signal in normal operation. In the event of a problem, this signal is turned off automatically.

NOTICE

When this option is activated, digital output 7 is no longer available as a controllable output. This output is now controlled by the watchdog timer.

⚠CAUTION

If the power supply is disconnected, digital output 7 cannot switch a signal. The connection of an actuator is conceivable. Make sure that this is not a safety-relevant device or that it can be reached directly.

6.11.4. Watchdog timer - turning off outputs

The default setting of the watchdog timer is, when it expires, that all outputs maintain their current state, no action is taken. This applies to every device, for all analog and digital outputs. The user has the possibility to instruct the watchdog timer to actively switch off all outputs via the watchdog timer configuration. With the PiXtend eIO Digital One device all digital outputs are set to LOW or switched off and with the PiXtend eIO Analog One all outputs are set to 0V or 0 mA.

6.12. Counter function of the PiXtend eIO Digital One

The Digital One device has a total of 8 counter units, each digital input represents a counter unit. Each of the 8 counters can be set differently. These are 16-bit counters, which cover a number range from 0 to 65535. When reaching the highest or lowest value, depending on the counting direction (up or down), an overflow happens automatically, and the respective counter continues counting. The counters have a setting option that allows the user to define which edges of a signal are counted.

6.12.1. Standard functions of the counters

Each of the 8 counters in the PiXtend eIO Digital One has various settings called standard functions.

These standard functions include setting which edge or edges of a digital input signal should be counted. The counters can count rising and falling edges, or both. If both edges are counted, the count will be twice as big as when only one edge is counted; this means the counters count each edge individually.

Additionally, you may define for every counter if it counts up, the actual value is incremented by 1 or if it counts down, the actual value is decremented by 1. The setting of the counting direction offers a third option, the 2-channel counter. This special function is described in the next chapter.

Each counter can be reset individually, or all counters can be reset at the same time. Every counter may be set to a preset with a value, for example if a count should not start at zero.

6.12.2. Function of the 2-channel counter

The PiXtend eIO Digital One device has a total of 8 counter units; 4 counters can be used as 2-channel counters (called 2-sensor operation). In this special mode, the respective digital inputs for the counters 0, 2, 4 and 6 are combined.

Counter Unit	First input	Second input	Comment
Counter 0	DI0	DI1	Only RE/FE[3], counter 1 deactivated
Counter 2	DI2	DI3	Only RE/FE, counter 3 deactivated
Counter 4	DI4	DI5	Only RE/FE, counter 5 deactivated
Counter 6	DI6	DI7	Only RE/FE, counter 7 deactivated

Table 5: Overview Digital One - 2-channel counter function, inputs

In this mode you may connect 2 sensors, switches or buttons for one counter unit and thus determine by the order of the signals whether the counter counts up or down. The PiXtend eIO Digital One device counts up. If a signal is detected at the first input, before the signal at the second input, the situation is reversed, and it counts down.

NOTICE

Please note that both digital inputs do not change their level simultaneously; the level changes occur one after the other. The time interval between both level changes must be at least 2.5 ms.

The overview shows which signal sequence of the digital inputs has which effect.

- First input → Second input → Counter counts up
- Second input → First input → Counter counts down

Two-channel counters are designed so that in principle you can also connect a quadrature encoder, which has an A and B track. Make sure that a complete rotation is always performed before changing the direction of the rotation; otherwise, it is possible that a rotation or count is lost.

NOTICE

The counter inputs are designed for a maximum frequency of 400 Hz. If the signals are faster, they are no longer reliably detected and counted.

Please regard that in the 2-channel counter mode the counter unit of the second digital input cannot be used separately. This applies to counters 1, 3, 5 and 7 when the other 4 counter units are configured as 2-channel counters. The user can read out the current status of the 8 digital inputs at any time, regardless of the set counter configuration.

[3] RE = rising edge, FE = falling edge

6.13. Hyper logic I/Os

The digital hyper logic is an innovative feature of PiXtend elO Digital One devices. With simple logic links between inputs and outputs that the user can specify in the device, the devices themselves can switch digital outputs without the need to transfer data over the bus.

Each device has two hyper logic processors (logical processing units, abbreviated HL), each of which can control one digital output. The control of each output is based on a logical link of up to four digital inputs. The user has a choice of different AND and OR links between the various digital inputs to determine at which state a digital output should be switched. In addition, it can be defined for each digital input whether it should be negated (inverted). This makes it possible to build and use negative switching logic as well.

The hyper logic processor 0, the first hyper logic unit, is assigned the digital output 0 and the digital inputs 0 to 3.

The hyper logic processor 1, the second hyper logic unit, is assigned the digital output 4 and the digital inputs 4 to 7.

When the hyper logic function is activated, the user can no longer control the respective digital output assigned to a hyper logic unit. The states of the digital inputs can be queried via the bus without restriction, regardless of the hyper logic processor setting.
If a hyper logic unit has been activated, it continues to run autonomously on the PiXtend elO device. Continuous communication is not required for the hyper logic to do its task.

6.14. Device configuration using Modbus RTU in CODESYS V3.5

PiXtend elO devices offer various settings and configuration options. For most settings, it is sufficient if they are transferred once to the PiXtend elO device when starting the main program. Cyclic transmission of the configuration values is not necessary and the bus load can be reduced during regular operation, as less communication with the devices is required.

Under CODESYS V3.5, it is therefore advisable to do the transmission of the individual holding registers, which contain setting information, only on a rising edge of a trigger variable.

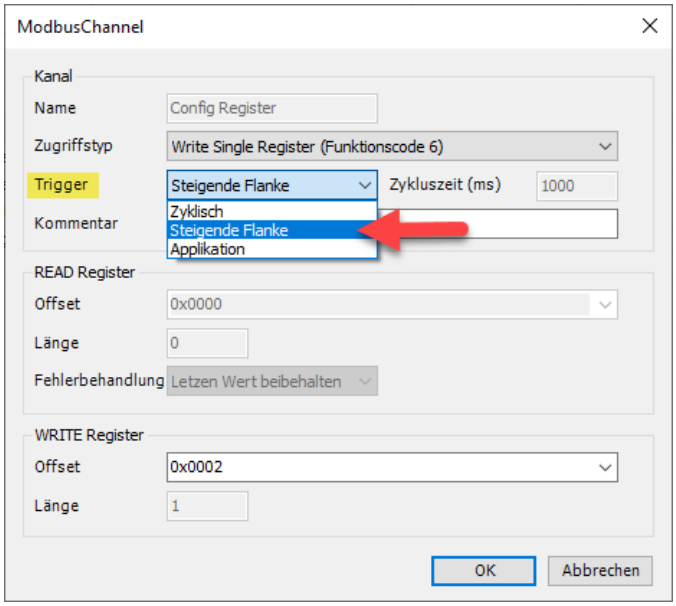


Figure 3: CODESYS Modbus, Transfer via trigger variable (rising edge)

If a setting is subsequently changed during operation, this can be done in CODESYS. By re-setting the trigger variable for the desired holding register, the content is transferred again. The transfer takes place only when required and not periodically.

Whether a main function such as the watchdog timer is active can be indicated by reading the status register, which should always be read periodically.

6.15. Number of supported devices

The Modbus RTU protocol as well as the PiXtend elO protocol are both designed for the use of up to 32 devices on a bus.

Please note that for Modbus RTU the device address must be in the range from 1 to 247 and for the PiXtend elO protocol it can be between 0 and 255.

NOTICE

Each device address may occur only once on a serial RS485 bus system. Double allocation of device addresses leads to malfunctions and data loss.

6.16. Multi-purpose output – digital output 0, 4 and 7

Note: This section only applies to Digital One device.

Digital One device have various additional functions which, when activated, take control or sovereignty over one of the digital outputs mentioned above. The user then no longer has any influence on the corresponding output, i.e., it can no longer be controlled by software (Modbus RTU or PiXtend elO protocol).

Please refer to the list below for information on which function controls which digital output when activated.

- Hyper logic processor 0 → Digital output 0
- Hyper logic processor 1 → Digital output 4
- Watchdog timer, watchdog timer output function → Digital output 7

6.17. Current and voltage conversion factors

With the Analog One device, the user only receives the raw values from the device. These values must be converted into current or voltage values. Below are the corresponding formulas.

- Current Inputs: Raw $value_{in}$ * 0.020158400229358 = mA_{in}
- Voltage inputs 10 volt range: (Raw $value_{in}$ * 10)/1023 = $Volt_{in}$
- Voltage inputs 5 volt range: (Raw $value_{in}$ * 5)/1023 = $Volt_{in}$

To set the analog outputs, a conversion is also necessary, this time in the other direction. The desired current and voltage values must be converted into raw values. The appropriate formulas are listed below.

- Current outputs: mA_{out} * (4095/20) = raw $value_{out}$
- Voltage outputs: V_{out} * (4095/10) = raw $value_{out}$

6.18. Combining two bytes to a 16-bit value

When working with the PiXtend elO protocol, 16-bit values, numbers from 0 to 65535, cannot be transmitted at once; they must be retrieved via two separate bytes. The user must combine these two bytes into a 16-bit value again upon receipt. Code examples are listed below.

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. Splitting a 16-bit value into two bytes

When working with the PiXtend elO protocol, 16-bit values, numbers from 0 to 65535, cannot be transmitted at once; they must be transmitted via two separate bytes. The user must extract these two bytes from the 16-bit value. Code examples are listed below.

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. Error signaling with PiXtend eIO devices

All PiXtend eIO devices are designed for cyclic operation but can also be used and controlled via sporadic individual commands.

If a message is sent to a device, the information is evaluated, and the device responds. If an error or problem is detected, the communication works without errors and the ERR LED provides a signal for an error code.

The feedback of the ERR LED always refers to just one message. If no error is detected with the next message and everything is fine in the device itself, the ERR LED goes off again. If there is no continuous error, the display of the last error is automatically acknowledged.

In cyclic operation, this behavior can cause the ERR LED to be switched on and immediately switched off again, the ERR LED starts blinking. This behavior can be due to the fact that different messages are sent to the device and among these messages is one or more messages that indicate an error.

If you notice that the ERR LED keeps turning on and off, the blinking may vary in speed, then stop communication with the device. Check if the DIP switches are in the correct position. Perform a power cycle on the device and then test all Modbus RTU function codes or PiXtend eIO protocol commands individually to determine which is faulty.

In such a situation, read the error register to determine which errors have occurred. All errors that occur are recorded until the user acknowledges them or performs a power cycle.

7. Device LEDs – Signaling

Figure 4: Status LEDs “ERR”, “COM” and “+5V” on PiXtend eIO devices

7.1. Signaling: supply voltage LED “+5V”

The green LED labeled "+5V" indicates the presence of the internal supply voltage of 5V DC.

State (Signal)	Designation	Meaning
Off/faint light	not ready	supply voltage not available and insufficient
On	ready	supply voltage available

If this LED does not light or only glows slightly, then the power supply to the device is not available or insufficient. In this case, check the wiring, the voltage between the terminals "VCC" and "GND" and check the requirements of the power supply. More detailed information can be found in the Hardware Manual.

7.2. Signaling: communication LED “COM”

The orange LED with the label "COM" indicates data transmissions on the RS485 bus. It is irrelevant whether the transmitted data is intended for the respective device or not. All communication on the bus causes blinking. If there are no data transmission, this LED remains dark.

State (Signal)	Designation	Meaning
off	idle	
blinking	communication	There is data transmissions/communication on the RS485 bus.

In regard to the blinking frequency, a tendency can be observed (the higher the frequency, the more data or the faster data is being transferred on the bus). However, the exact baud rate or amount of data cannot be determined by this LED.

7.3. Signaling: error LED “ERR”

Some error states that may occur with the PiXtend eIO are not able to be reported back to the bus master via RS485, e.g. in case of faulty communication or if the watchdog of the PiXtend eIO has been activated. For these situations, a red "ERR" LED has been placed on the PiXtend eIO device to help the user to determine the source of a problem.

The meaning of the different LED (L1) signals can be found below:

State (Signal)	Designation	Meaning
Off	Ready, no errors	-
on (continuous)	Telegram errors	Modbus RTU protocol: Incorrect function code (FC) Number of registers, coils or discretes is wrong
	Message error	PiXtend eIO ASCII protocol: frame error, parity error, overrun error, incorrect baud rate defined
Blinking fast (0.05 s)	Communication error Configuration error	With Modbus RTU and PiXtend eIO ASCII protocol: data transfer error Configuration via DIP switch not permitted or changed during operation
Blinking (0.1 s)	Command error	Only with PiXtend eIO ASCII protocol: the command sent is unknown or is not supported
Blinking slow (0.2 s)	I/O error	Only with PiXtend eIO ASCII protocol: the desired input or output is not available (e.g. when trying to set output 9, but there are only 8 outputs)
Blinking 3x fast, 3x slow, 1 second break	Watchdog timer	With Modbus RTU and PiXtend eIO ASCII protocol: triggered by the device's Watchdog timer

Table 6: LED “ERR” – signaling of error states

The one-time red blinking when starting the devices is normal and does not indicate an error. Similarly, the orange "COM" LED blinks briefly during startup.

8. PiXtend eIO – device configuration

8.1. Overview

All PiXtend eIO devices have 2 DIP[4] switch blocks with 8 switches each. The first DIP block is marked ADDRESS, these switches are used to set the device address. Using the 8 switches, bus addresses in the range 1 - 247 (Modbus RTU) or 0 - 255 (PiXtend eIO protocol) can be set.
Each device address may only be used once on the bus!
The specification of the device address is done as mentioned above using 8 switches, the arrangement of the switch goes from left to right, numbers 1 to 8 (see figure 5).
If there is a switch (white) at the top of a DIP block, it is in the ON position and thus active. A complete list of all possible device addresses can be found in chapter 8.2 Device Addresses.

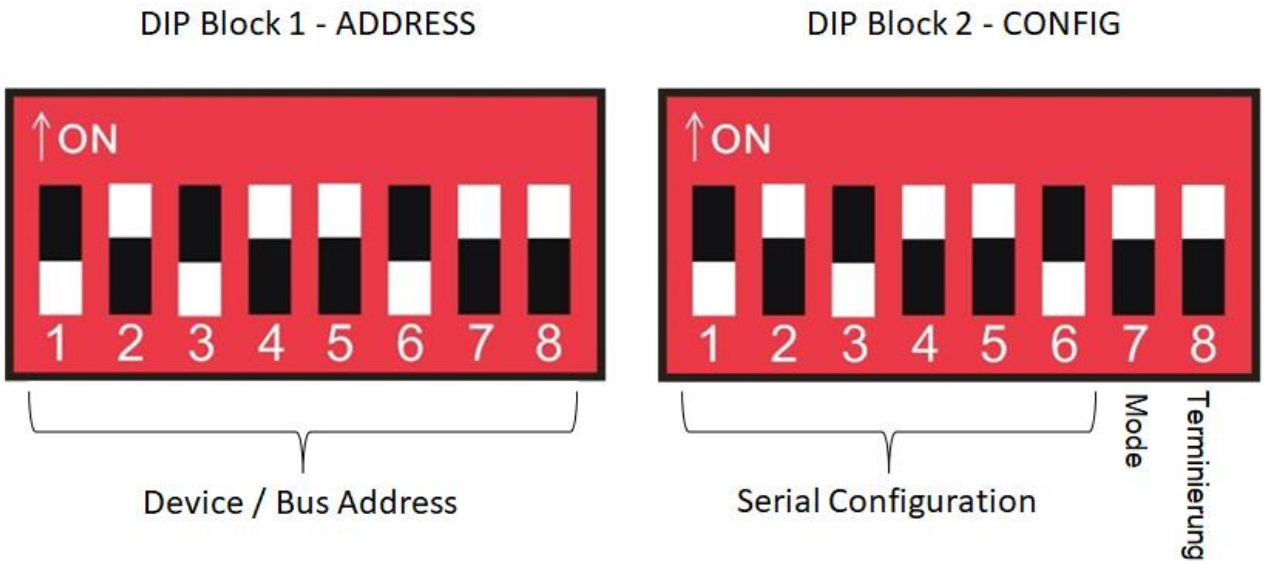


Figure 5: Device configuration - overview DIP switches - scheme

The second DIP block is labeled CONFIG to set the serial configuration as well as to select the communication protocol (mode) and to switch the bus termination on or off.

The serial interface is configured via the first six switches (number 1 to 6). Switch 7 is used to select the communication protocol. In the OFF position, when the switch is down, the Modbus RTU protocol is selected (active) and used by the device for communication. In the ON position, the device uses Kontron Electronics GmbH's own PiXtend eIO protocol for data exchange with other devices.

Bus termination can be switched on via switch 8, the switch must be pushed into ON position. If it is at the bottom, bus termination is switched off.

For more information on communication settings, see chapter 8.3 Serial configuration, 8.4 Protocol selection and 8.5 Bus termination.

[4]DIP is the abbreviation for Dual In-line Package.

8.2. Device address

The device address enables the bus master to always address exactly one device on the bus. Each message sent by the bus master contains a unique device address. This ensures which device is addressed and that several devices never respond simultaneously.

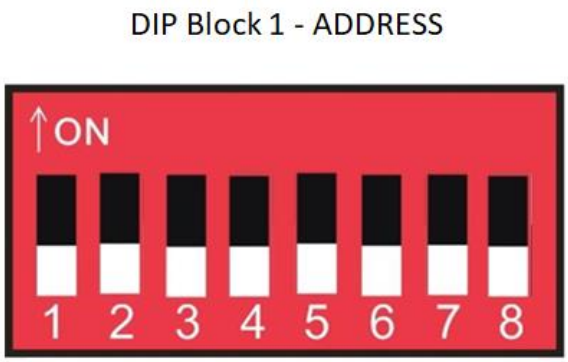


Figure 6: DIP block 1 - ADDRESS – device address

The following table lists all 255 possible device addresses, together with the corresponding switch position on DIP block 1 for all 8 switches. A 0 corresponds to the Off position when the switch is in the lower position and a 1 corresponds to the ON position when the switch is in the upper position. The order of the switches goes from left to right. Note the numbering, this is from 1 (switch 1) to 8 (switch 8).

Device address	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

Device address	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

Device address	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

Device address	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

Device address	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

Device address	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

Device address	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

Table 7: PiXtend elO: Selection table - device addresses

8.3. Serial configuration

So that the device can be integrated into as many RS485 networks as possible, there are a total of 6 switches (numbers 1 - 6) on DIP block 2 which can be used to configure the serial interface of the device. Various combinations of baud rate, parity and stop bit are available. In total there are 45 setting options, whereby only one can be set at a time via the 6 switches.
A 0 corresponds to the Off position when the switch is in the lower position and a 1 corresponds to the On position when the switch is in the upper position. The order of the switches goes from left to right, numbers 1 - 6.

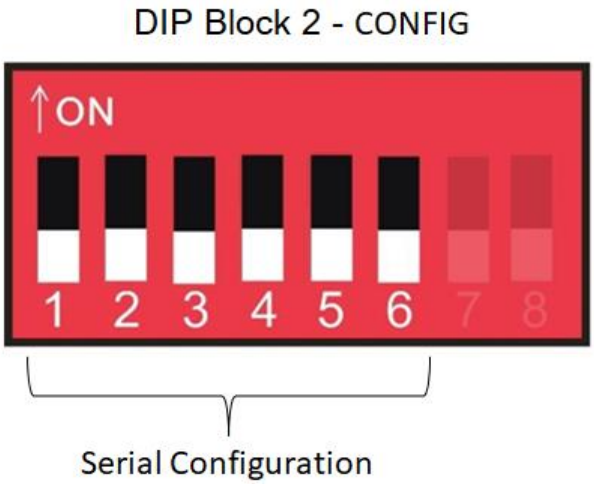


Figure 7: DIP block 2 - CONFIG – serial configuration

Select the desired combination of baud rate, parity and stop bit(s) from the following table and then transfer the corresponding switch order of the table to the first 6 switches of DIP block 2 from left to right according to the assignment 1 (switch 1) to 6 (switch 6). Switches 7 and 8 can be disregarded here, they will be discussed in the following chapters.

Number	Baud rate	Parity	Stop bit	1	2	3	4	5	6
0	19200	even	1	0	0	0	0	0	0
1	2400	none	1	1	0	0	0	0	0
2	4800	none	1	0	1	0	0	0	0
3	9600	none	1	1	1	0	0	0	0
4	14400	none	1	0	0	1	0	0	0
5	19200	none	1	1	0	1	0	0	0
6	28800	none	1	0	1	1	0	0	0
7	38400	none	1	1	1	1	0	0	0
8	57600	none	1	0	0	0	1	0	0
9	76800	none	1	1	0	0	1	0	0
10	115200	none	1	0	1	0	1	0	0
11	230400	none	1	1	1	0	1	0	0
12	2400	none	2	0	0	1	1	0	0
13	4800	none	2	1	0	1	1	0	0
14	9600	none	2	0	1	1	1	0	0

Number	Baud rate	Parity	Stop bit	1	2	3	4	5	6
15	14400	none	2	1	1	1	1	0	0
16	19200	none	2	0	0	0	0	1	0
17	28800	none	2	1	0	0	0	1	0
18	38400	none	2	0	1	0	0	1	0
19	57600	none	2	1	1	0	0	1	0
20	76800	none	2	0	0	1	0	1	0
21	115200	none	2	1	0	1	0	1	0
22	230400	none	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

Table 8: PiXtend elO: Selection table - serial device configuration

8.4. Protocol selection

Switch 7 in DIP block 2 - CONFIG is used to select the communication protocol for the PiXtend eIO device. The well-known Modbus RTU protocol and Kontron Electronics GmbH's own PiXtend eIO protocol are available.

In switch position OFF, when the switch is in the down position, the Modbus RTU protocol (default setting) is selected and active.

Modbus RTU active

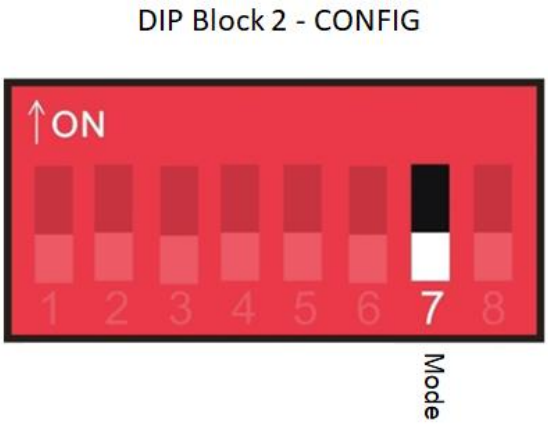


Figure 8: DIP block 2 - CONFIG – mode – Modbus RTU

In switch position ON, when the switch is in the upper position, the PiXtend eIO protocol is selected and is used for communication instead of the Modbus RTU protocol.

PiXtend eIO protocol active

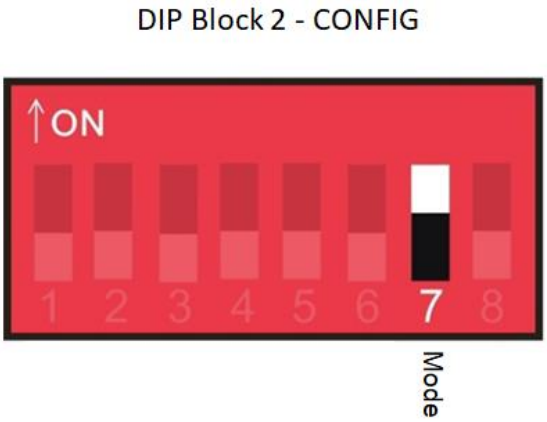


Figure 9: DIP block 2 - CONFIG – mode – PiXtend eIO Protocol

8.5. Bus termination

Switch 8 in DIP block 2 - CONFIG is used to turn the bus termination on and off for the PiXtend eIO device. The selection depends on whether the device is the first or last device on the bus. In switch position OFF, when the switch is in the down position, bus termination is off (default setting).

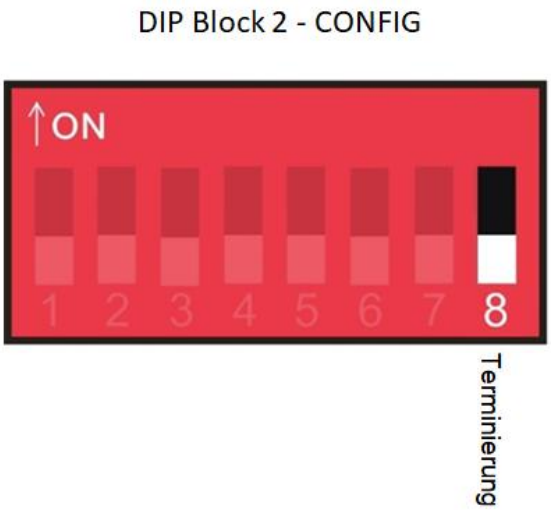


Figure 10: DIP block 2 - CONFIG - bus termination off

In switch position ON, when the switch is in the down position, bus termination is on. Termination only needs to be on for the first and the last device on the bus.

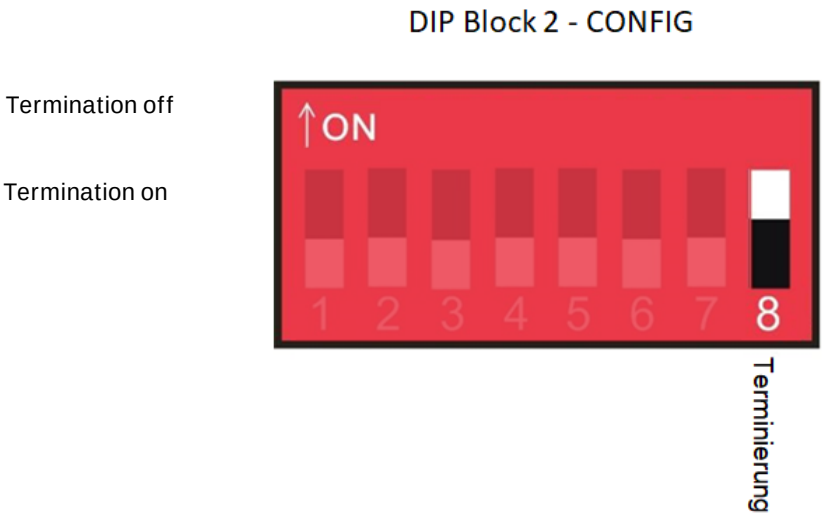


Figure 11: DIP block 2 - CONFIG – bus termination on

9. Modbus RTU - Protocol

9.1. Introduction

In this chapter, there is information about the Modbus addresses, registers and functions supported by the respective PiXtend elO device. This information allows the user to switch digital and analog outputs and read digital and analog inputs. In addition, certain registers and configuration settings can be used to activate other functions in the device.

9.2. Overview

General - for all elO devices

- Supported function codes
- Modbus RTU error numbers

PiXtend elO Digital One

- Modbus addresses and functions
- Example CODESYS
- Example Python
- Example C

PiXtend elO Analog One

- Modbus addresses and functions
- Example CODESYS
- Example Python
- Example C

9.3. Supported function codes

The following function codes are supported by PiXtend eIO devices.

- FC02 - Read discrete inputs
- FC01|FC05|FC15 – Read/write of coils
  - Only Digital One device
- FC04 – Read the input register
- FC03|FC06|FC16 – Read/write of the holding register

9.3.1. PiXtend eIO Digital One - Modbus addresses and functions

9.3.1.1 Discrete inputs - reading digital inputs

Digital inputs can be read individually or all simultaneously. In CODESYS, each input is provided as one of the bits in a byte; other software platforms and Modbus RTU implementations may differ. Please refer to the relevant documentation for your system.

Use the Modbus RTU function code 2 to read the values.

Name	Address	Data type	Comment
Digital Input 0-7	0-7	Bit	Each digital input is provided as a single bit with its own discrete input address

Table 9: Digital One: Reading digital inputs as discrete inputs

9.3.1.2 Coils (outputs) - setting digital outputs

The digital outputs can be set or written individually or all simultaneously. In CODESYS, each output is provided as one of the bits in a byte; other software platforms and Modbus RTU implementations may differ. Please refer to the relevant documentation for your system.

Use the Modbus RTU function code 1, 5 and 15 to read and write values.

Name	Address	Data type	Comment
Digital output 0-7	0-7	bit	Each digital output is provided as a single bit with its own coil address

Table 10: Digital One: Reading/writing digital outputs as coils

9.3.1.3 Input register reading of various information and states

The input registers allow various information to be read from PiXtend eIO devices. All available Modbus input registers are listed in the table below. References to breakdowns of the various bits that may be present in a register are in the comment column. Use function code 4 to read the input registers of the device.

The addresses of the input registers start counting at address 0. The device has 12 addresses or 12 input registers. The data type of each register is 16 bits of type WORD, 2 bytes long and can contain the numerical values from 0 to 65535.

Name	Address	Comment
Digital inputs	0	This is an alternative to reading the digital inputs if you do not want to use discrete inputs. Only the first (lower) 8 bits in the WORD are used by the device.
Status register	1	This register contains status information from the device as feedback to the user to check whether a function is activated or not. In this way, it can be determined whether the watchdog timer is active. The table in chapter 9.3.1.5 Bits in the status register contains a breakdown of the status bits.
Error register	2	This contains an image of the errors detected in the device, which cannot be read directly via the ERR LED afterwards, because the faulty state is no longer present. The table in chapter 9.3.1.6 Bits in the error register contains a breakdown of the error bits.
Counter 0 register	3	Current value of counter 0, value range 0 to 65535
Counter 1 register	4	Current value of counter 1, value range 0 to 65535
Counter 2 register	5	Current value of counter 2, value range 0 to 65535
Counter 3 register	6	Current value of counter 3, value range 0 to 65535
Counter 4 register	7	Current value of counter 4, value range 0 to 65535
Counter 5 register	8	Current value of counter 5, value range 0 to 65535
Counter 6 register	9	Current value of counter 6, value range 0 to 65535
Counter 7 register	10	Current value of counter 7, value range 0 to 65535
Version register	11	Version of the firmware as numbers, 101 = 1.01, 102 = 1.02 etc.

Table 11: Digital One: Input register

9.3.1.4 Holding register – setting outputs and configuration of bits

The holding registers allow the individual outputs to be set and the setting of various configurations, such as activating the counters or the watchdog timer. All available Modbus holding registers are listed in the table below. References to breakdowns of the various bits that may be present in a register are in the comment column. Use function code 6 and 16 to write in the holding registers of the device.

The addresses of the holding registers start counting at address 0. The device has 7 device addresses or 7 holding registers. The data type of each register is 16 bits of type WORD, 2 bytes long and can contain the numerical values from 0 to 65535.

Name	Address	Comment
Digital outputs	0	If the appropriate bit has been set in the config register, the outputs of the Digital One device can be set using the first eight bits in that register. This is an alternative to the use and control via coils.
Watchdog timer register	1	The watchdog timer register allows the setting and switching of the device watchdog timer for monitoring the bus communication. The table in chapter 9.3.1.7 Bits in the watchdog timer register contains a breakdown of the configuration bits.
Config register	2	In the config register various settings can be made in the device, the internal error memory can be cleared, the hyper logic can be activated, the counters can be reset and it can be determined whether the outputs should be set via a register instead of by coils. The table in chapter 9.3.1.8 Bits in the config register contains a breakdown of the configuration bits.
Counter config register 0	3	This register allows the switching on of a counter (0 - 7) and the definition of edge detection. Rising edges, falling edges or both edges can be counted. The table in chapter 9.3.1.9 Bits in the counter config register contains a breakdown of the configuration bits.
Counter config register 1	4	In this register you may define a counter function, a type for every counter. There are options to count up and down, with two inputs each.

Name	Address	Comment
		The up counter is preset. The table in chapter 9.3.1.10 Bits in the counter config register contains a breakdown of the configuration bits.
Counter pre-set value register	5	Preload counter 0 - 7 with a value, e.g., counting down to 0 must be done in conjunction with the reset bits in the config register. See also chapter 9.3.1.8 Bits in the config register.
Hyper logic config register	6	This register is used to configure the hyper logic in Digital One device. The table in chapter 9.3.1.11 Bits in the hyper logic config register contains a breakdown of the configuration bits.

Table 12: Digital One: Holding register

NOTICE

If counters 0, 2, 4, 6 are operated in the 2-sensor mode, the other counters cannot be used. The device prevents this internally and switches off the affected counters automatically.

NOTICE

Counters 1, 3, 5, 7 cannot be switched to "2-sensor mode". The device ignores this configuration and deactivates the affected counters automatically.

9.3.1.5 Bits in the status register

The status register shows which function is active in the device and which is not. A cyclic request is recommended; additionally, it can be checked in this way whether a function has been activated or not.

Low byte:

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

Table 13: Bits in the status register - low byte

Bit 0 - WDT ON → Watchdog time active

Off: The watchdog timer is off.

On: The watchdog timer is active and, depending on the setting, is reset or triggered after the specified time has expired.

Bit 1 - WDT RST → Watchdog time reset - type setting

Off: The watchdog timer is always reset during bus activity. If there is no communication on the bus because the master is no longer transmitting, the watchdog timer is triggered after the timeout has expired.

On: The watchdog timer is reset when the device receives a complete Modbus RTU message from the master within the set timeout time. This is useful if many Modbus RTU devices are connected to a bus and you want to determine if the Modbus RTU Master is addressing the device within the timeout time. The timeout can be set up to a maximum of 8 seconds.

Bit 2 - WDT OUT → Watchdog time output active (digital output 7)

Off: When the watchdog timer expires, the digital output 7 is not set but is treated as a normal digital output.

On: The digital output 7 is used by the watchdog timer and is no longer available to the user. In the default setting, the watchdog timer sets the digital output 7 to HIGH when it expires. This can be changed by the user. See bit 3 - WDT OBE.

Bit 3 - WDT OBE → Watchdog time output behavior

Off: Digital output 7 is switched from LOW → HIGH when the watchdog timer expires. The normal state for this setting is LOW. This setting corresponds to the standard behavior (default setting).
This setting is useful if, for example, a lamp or siren is to be switched. If the watchdog timer expires, a fault in the system can be detected more quickly.

On: Digital output 7 is switched from HIGH → LOW when the watchdog timer expires. The normal state for the digital output is HIGH. This setting is useful if a relay must be energized during operation for the system to be operational. When the watchdog timer expires, digital output 7 is switched to LOW and the relay turns off.

Bit 4 - HL0 → Hyper logic processor 0

Off: There is no hyper logic processing for hyper logic processor 0.

On: Hyper logic processing is enabled and the state of digital output 0 can be controlled by up to four digital inputs. Various combinations of digital inputs 0 to 3 are available; see further information in section 9.3.1.8 Bits in the config register.

Bit 5 - HL1 → Hyper logic processor 1

Off: There is no hyper logic processing for hyper logic processor 1.

On: Hyper logic processing is enabled and the state of digital output 4 can be controlled by up to four digital inputs. Various combinations of digital inputs 4 to 7 are available; see further information in section 9.3.1.8 Bits in the config register.

Bit 6 - CNT0 → Counter 0 active

Off: Counter 0 is not active and there is no counting.

On: Counter 0 is activated. The 16-bit value of the counter is in the input register counter 0 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1. Further information on this can be found in the sections 9.3.1.9 Bits counter config register 0 and 9.3.1.10 Bits in counter config register 1.

Bit 7 - CNT1 → Counter 1 active

Off: Counter 1 is not active and there is no counting.

On: Counter 1 is activated. The 16-bit value of the counter is in the input register counter 1 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1. Further information on this can be found in the sections 9.3.1.9 Bits in the counter config register 0 and 9.3.1.10 Bits in counter config register 1.

High byte:

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

Table 14: Bits in the status register - high byte

Bit 8 - CNT2 → Counter 2 active

Off: Counter 2 is not active and there is no counting.

On: Counter 2 is activated. The 16-bit value of the counter is in the input register counter 2 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1. Further information on this can be found in the sections 9.3.1.9 Bits in the counter config register and 9.3.1.10 Bits in counter config register 1.

Bit 9 - CNT3 → Counter 3 active

Off: Counter 3 is not active and there is no counting.

On: Counter 3 is activated. The 16-bit value of the counter is in the input register counter 3 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1. Further information on this can be found in the sections 9.3.1.9 Bits in the counter config register 0 and 9.3.1.10 Bits in counter config register 1.

Bit 10 - CNT4 → Counter 4 active

Off: Counter 4 is not active and there is no counting.

On: Counter 4 is activated. The 16-bit value of the counter is in the input register counter 4 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1. Further information on this can be found in the sections 9.3.1.9 Bits in the counter config register 0 and 9.3.1.10 Bits in counter config register 1.

Bit 11 - CNT5 → Counter 5 active

Off: Counter 5 is not active and there is no counting.

On: Counter 5 is activated. The 16-bit value of the counter is in the input register counter 5 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1. Further information on this can be found in the sections 9.3.1.9 Bits in the counter config register 0 and 9.3.1.10 Bits in counter config register 1.

Bit 12 - CNT6 → Counter 6 active

Off: Counter 6 is not active and there is no counting.

On: Counter 6 is activated. The 16-bit value of the counter is in the input register counter 6 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1. Further information on this can be found in the sections 9.3.1.9 Bits in the counter config register 0 and 9.3.1.10 Bits in counter config register 1.

Bit 13 - CNT7 → Counter 7 active

Off: Counter 7 is not active and there is no counting.

On: Counter 7 is activated. The 16-bit value of the counter is in the input register counter 7 register.
The behavior of the counter can be influenced by the user via the holding register counter config register 0 and counter config register 1 . Further information on this can be found in the sections 9.3.1.9 Bits in the counter config register 0 and 9.3.1.10 Bits in counter config register 1.

Bit 14 - DOREG ON → Set digital outputs via register

Off: Digital outputs 0 to 7 can only be controlled via coils (FC01, FC05, FC15).

On: Digital outputs 0 to 7 can only be controlled via the holding register digital outputs. Setting or changing the digital outputs via coils is not possible with this setting.

9.3.1.6 Bits in the error register

The error register contains a copy of the errors that have occurred in the device. The bits in the error register can be used to determine which errors have occurred. During operation the bits remain set until they are cleared by the QUIT ERR bit in the config register or a power cycle is executed. Exempted from this rule is the watchdog timer error bit; further information can be found in the explanation text for bit 4.

Low byte:

Bit	7	6	5	4	3	2	1	0
Name	COM ERR	COF ERR	TEL ERR	WDT ERR	HLDOCL ERR	CNTCL ERR	WDTDO7 ERR	-
Start value	0	0	0	0	0	0	0	0

Table 15: Bits in the error register - low byte

Bit 1 - WDTDO7 ERR → Watchdog timer - digital output 7 conflict

If the watchdog timer has been configured to use digital output 7 as the signal output, then this output will not be available to the user. If the user still tries to set digital output 7, this bit is set to 1 and indicates an operating error.

Bit 2 - CNTCL ERR → Counter configuration conflict

Digital inputs 1, 3, 5 and 7 cannot be used as a 2-channel counter or switched into the 2-sensor operation. If this is nevertheless attempted, this bit is set and an acknowledgment of the invalid operation is received.

Bit 3 - HLDOCL ERR → Hyper logic output conflict

Hyper logic processors 0 and 1 use digital outputs 0 and 4. The corresponding digital outputs are not available to the user when a hyper logic processor is turned on. If an attempt is made to set the digital outputs nevertheless, this bit is set as error feedback and indicates that there is a conflict between the device setting and the digital outputs used.

Bit 4 - WDT ERR → Watchdog timer has expired
This bit indicates whether or not the watchdog timer had expired before the last power-up. Since the device enters the safe state when the watchdog timer expires, this bit can only be queried after a subsequent power cycle. The bit can be cleared during operation via the QUIT ERR bit in the config register or when another power cycle can be executed.

Bit 5 - TEL ERR → Function code or address error

If the device has been sent a telegram or message with an unsupported function code or the received Modbus RTU message contains a non-existent address, then this error bit is set.
By means of this error bit, the control program can determine whether this error has occurred since the last power-up.

Bit 6 - COF ERR → Configuration error

This bit is set if a configuration error is detected in the device. In normal operation, this error will not occur unless the DIP switches are changed during operation. To remedy this situation, either the DIP switches must be returned to their original position or a power cycle is performed, in which case the new configuration is adopted by the device.

NOTICE

If the DIP switches were adjusted and a power cycle was performed, then the setting of the device was changed! If the change does not take place in the control program, the device can neither be addressed nor used.

Bit 7 - COM ERR → communication error
This bit is set if the device detects a communication error. As a rule, there are three fault states. The device may detect a parity, frame, or overrun error.

A parity error usually occurs when the settings on the control device are not correct. The same applies to the frame error; in most cases the baud rate is incorrect and the device does not receive any valid data. The overrun error, on the other hand, means that too much data is sent over the bus too quickly and the device cannot record everything.

In such cases, check the settings on the control box to see if they match the configuration of the device, if the wiring is correct, and that there are no faults on the bus.

High byte:

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

Table 16: Bits in the error register - high byte

The high byte of the error register contains no information or error bits and can be ignored.

9.3.1.7 Bits in the watchdog timer register

The watchdog timer register allows the user to influence the behavior of the device's watchdog timer. The setting options range from the timeout time to the watchdog timer setting an output when it expires.

If the watchdog timer is switched on without further configuration, a timeout of 16 milliseconds is set. The watchdog timer is reset as soon as activity is detected on the bus, regardless of whether the data is intended for the device or not.

Low byte:

Bit	7	6	5	4	3	2	1	0
Name	WDTO	WDTRST	WDTEN	WB	WT3	WT2	WT1	WT0
Start value	0	0	0	0	0	0	0	0

Table 17: Bits in the watchdog timer register - low byte

Bit 3...0 - WT3...WT0 → Watchdog timer timeout

The watchdog timer time determines how long the watchdog timer runs before it is triggered.
The following table shows which bit pattern must be set so that the watchdog timer takes over the timeout time when activated and works with it. If no bits are set, the watchdog timer uses the shortest time of 16 ms. This time span is much too short for Modbus RTU communication. Select a larger value in any case, e.g. 4 seconds.

Watchdog timer timeout:

WT3	WT2	WT1	WT0	Timeout
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

Table 18: Watchdog timer timeout overview

Procedure for setting/changing the watchdog time during operation:

- Watchdog is deactivated (turned off)
- Select the desired watchdog timer timeout
- Turn on the watchdog

Bit 4 - WB → Watchdog timer behavior

Off: The state of the digital outputs remains as it is when the watchdog timer expires.

On: All digital outputs are actively switched off when the watchdog timer expires.

Bit 5 - WDTEN → Watchdog timer enabled

Off: The watchdog timer is off.

On: The watchdog timer is activated or switched on.

Bit 6 - WDTRST → Watchdog timer reset type

Off: The watchdog timer reset is executed during bus activity, no matter what data is involved and whether it is intended for the device or not. This setting can be used to determine whether the master is basically communicating or whether it has failed. If there is no communication on the bus within the selected watchdog timer time, the watchdog timer expires and is triggered.

On: The watchdog timer reset is only executed when the device receives a valid Modbus RTU message from the master. If the device is not addressed by the master within the watchdog timer time with a valid Modbus RTU message, the watchdog timer expires.
This setting is useful to determine that each device is addressed within the selected watchdog timer time in a large Modbus RTU network.

Bit 7 - WDTO → Watchdog timer output

Off: The digital output 7 is not used by the watchdog timer and is available to the user.

On: Digital output 7 is used by the watchdog timer as a signal output. By default, the watchdog timer sets this output to HIGH when it expires.

High byte:

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

Table 19: Bits in the watchdog timer register - high byte

Bit 8 - WDTOB → Watchdog timer output behavior

Off: Digital output 7 is switched from LOW to HIGH when the watchdog timer expires. The normal state during normal operation is LOW.

On: Digital output 7 is switched from HIGH to LOW when the watchdog timer expires. The normal state during normal operation is HIGH.

For further information about the watchdog timer see chapter 6.11 PiXtend eIO Watchdog Timer, section Basic knowledge.

9.3.1.8 Bits in the config register

The config register allows you to set and activate various functions of the device. The bits can be used to reset the internal error memory during operation. The hyper logic processors can be set and activated. It is possible to reset the counters and to switch the digital outputs from coils to holding register 0.

Low byte:

Bit	7	6	5	4	3	2	1	0
Name	HL1CFG0	HL0CFG3	HL0CFG2	HL0CFG1	HL0CFG0	HL1ACT	HL0ACT	QUIT ERR
Start value	0	0	0	0	0	0	0	0

Table 20: Bits in the config register - low byte

Bit 0 - QUIT ERR → Quit errors

Off: The device sets the error bits in the error memory.

On: If this bit is set to 1, then the internal error memory of the device is cleared.
It is best to check the error register after setting this bit to make sure that all errors have been cleared.

Bit 1 - HL0ACT → Hyper logic processor 0 active

Off: The hyper logic processor 0 is switched off and not working.

On: The hyper logic processor 0 is active and controls the digital output 0 according to the selected hyper logic link. See bit 6 to 3. Digital output 0 is no longer available to the user; it is used exclusively by the hyper logic processor 0.
The status register can be checked to see whether the hyper logic processor 0 is active.

Bit 2 - HL1ACT → Hyper logic processor 1 active

Off: The hyper logic processor 1 is switched off and not working.

On: The hyper logic processor 1 is active and controls the digital output 4 according to the selected hyper logic link. See bit 10 to 7. Digital output 4 is no longer available to the user; it is used exclusively by the hyper logic processor 1.
The status register can be checked to see whether the hyper logic processor 1 is active.

Bits 6 ... 3 - HL0CFG3 **... 0** → Hyper logic processor 0 link

The following table shows which links of digital inputs 0 to 3 are supported by hyper logic processor 0. The binding strength of the logical links is symbolized by brackets. A double ampersand (&&) represents an AND operation, and an OR operation is expressed by two vertical lines (||). The specification of the columns HL0CFG3 .. 0 can be transferred directly to these 4 bits just as the bits in the register are defined. The digital output from the hyper logic processor 0 is always the digital output 0.

Num	HL0CFG3	HL0CFG2	HL0CFG1	HL0CFG0	Link
0	0	0	0	0	DI0
1	0	0	0	1	DI0 && DI1
2	0	0	1	0	DI0 || DI1
3	0	0	1	1	DI0 && DI1 && DI2
4	0	1	0	0	(DI0 && DI1) || DI2
5	0	1	0	1	(DI0 || DI1) && DI2
6	0	1	1	0	DI0 || DI1 || DI2
7	0	1	1	1	DI0 && DI1 && DI2 && DI3
8	1	0	0	0	(DI0 && DI1 && DI2) || DI3
9	1	0	0	1	DI0 && (DI1 || DI2) && DI3
10	1	0	1	0	DI0 || DI1 && DI2 && DI3
11	1	0	1	1	(DI0 && DI1) || DI2 || DI3
12	1	1	0	0	(DI0 || DI1) && (DI2 || DI3)
13	1	1	0	1	(DI0 || DI1 || DI2) && DI3
14	1	1	1	0	DI0 || DI1 || DI2 || DI3

Table 21: Hyper logic processor 0 - link overview

High byte:

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

Table 22: Bits in the config register - high byte

Bits 10 ... 7 - HL1CFG3 ... 0 → Hyper logic processor 1 link

The following table shows which links of digital inputs 4 to 7 are supported by hyper logic processor 1. The binding strength of the logical links is symbolized by brackets. A double ampersand (&&) represents an AND operation, and an OR operation is expressed by two vertical lines (||). The specification of the columns HL1CFG3... 0 can be transferred directly to these 4 bits just as the bits in the register are defined. The digital output from the hyper logic processor 1 is always the digital output 4.

Num	HL1CFG3	HL1CFG2	HL1CFG1	HL1CFG0	Link
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

Table 23: Hyper logic processor 1 - link overview

Bits 14 ... 11 - CRST3 ... 0 → Counter reset

Every counter may be reset individually or all counters at once with the four counter reset bits. The following table shows both the decimal number and the binary code for resetting each counter.
The specification of the columns CRST3 ... 0 can be transferred directly to these four bits, with the LSB on the right and the MSB on the left, according to the arrangement of the bits in the register.

Number	CRST3	CRST2	CRST1	CRST0	Counter
0	0	0	0	0	Off, no action
1	0	0	0	1	Reset counter 0
2	0	0	1	0	Reset counter 1
3	0	0	1	1	Reset counter 2
4	0	1	0	0	Reset counter 3
5	0	1	0	1	Reset counter 4
6	0	1	1	0	Reset counter 5
7	0	1	1	1	Reset counter 6
8	1	0	0	0	Reset counter 7
9	1	0	0	1	Reset all counters

Table 24: Config register - Bits: Counter reset

NOTICE

When a counter is reset, the value of the counter pre-set value register is loaded into the counter register. Without user intervention, this value is 0.

The user can preload any value between 0 and 65535 into a counter. A counter does not have to start counting at 0; after a reset it may take another value. This behavior is advantageous for a down counter, if you always want to count 100 units. The counter can be preset to 100; when it reaches 0, a reset resets the counter to 100.

Bit 15 - DOREG → Setting the digital outputs via holding register 0

Off: The digital outputs can only be set via coils.

On: The digital outputs can only be set via the holding register 0, instead of coils.

9.3.1.9 Bits counter config register 0

The counter config register 0 allows you to set the edge detection of the counter, i.e. to which edge the counter should react, increase or decrease the count value. Setting the edge setting activates the counter at the same time. For each counter two bits can be selected in the counter config register 0 via the edge setting. Each counter has four states. According to the breakdown of the bits, the setting for one counter is shown, which applies analogously to all counters.

Low byte:

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

Table 25: Bits in the config register 0 - low byte

High byte:

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

Table 26: Bits in the config register 0 - high byte

Number	CEDX1	CEDX0	Edge setting
0	0	0	Off, counter not active
1	0	1	Rising edge, 0 → 1
2	1	0	Falling edge, 1 → 0
3	1	1	Rising + falling edge (RE + FE), 0 → 1 + 1 → 0

Table 27: Counter config register 0 - Edge setting - breakdown

The "x" in the abbreviation CEDX1 respectively CEDX0 represents one of the eight counters (0 to 7).

NOTICE

The edge setting number 3 (Binary 11 - rising and falling edge) cannot be used for 2-sensor operation (2-channel counter). If this setting is nevertheless selected, the device only evaluates the rising edges; the falling edges are ignored. The device behaves in the same way as for edge setting number 1 (binary 01 - rising edge).

9.3.1.10 Bits counter config register 1

A counter type can be defined for each counter with the counter config register 1. By default, each counter counts up. For each counter, there are three counter types. According to the breakdown of the bits, the setting for one counter is shown, which applies analogously to all counters.

Low byte:

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

Table 28: Bits in the config register 1 - low byte

High byte:

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

Table 29: Bits in the config register 1 - high byte

Number	CFTX1	CFTX0	Counter type
0	0	0	Counter counts up, up counter (default)
1	0	1	Counter counts down, down counter
2	1	0	Counter counts up/down, 2-sensor operation (2-channel counter)
3	1	1	reserved

Table 30: Counter config register 1 - Counter type - breakdown

The "x" in the abbreviation CFTX1 respectively CFTX0 represents one of the eight counters (0 to 7).

Further information and notes on the counters can be found in chapter 6.12 Counter function in PiXtend eIO Digital One, section Basic knowledge.

9.3.1.11 Bits in the hyper logic config register

In the hyper logic config register, the user can define for each digital input whether it should be considered positive (default) or negative (inverted). This setting is helpful when using a sensor that is LOW active and switches from HIGH to LOW when detecting an object or part.

Low byte:

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

Table 31: Bits in the hyper logic config register - low byte

One bit (HLINx, x = 0 to 7) is available for each of the eight digital inputs.

Bit 7 ... 0 - HLIN7 ... 0 → Hyper logic processor 0 & 1 - input inversion

Off: All digital inputs are considered positive.

On: The signal at the selected input is inverted or evaluated negatively.
With this setting, it is assumed that a connected sensor or switch always delivers a HIGH level in its idle state.

High byte:

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

Table 32: Bits in the hyper logic config register - high byte

The hyper logic processors 0 and 1 typically operate at a rate faster than the digital inputs can respond to achieve the fastest possible throughput. For some applications, however, it may be useful to debounce the input signals for a more secure and stable output for each hyper logic processor.

Bit 8 - HLSM0 → Hyper logic processor 0 - Signal smoothing (debounce)

Off: No smoothing (debounce) is performed.

On: The input signals for the hyper logic processor 0 are debounced, i.e. for digital inputs 0 to 3, if they occur in the hyper logic operation selected by the user. Each signal must be present for more than 2 ms before it is considered valid and can be processed by the hyper logic.

Bit 9 - HLSM1 → Hyper logic processor 1 - Signal smoothing (debounce)

Off: No smoothing (debounce) is performed.

On: The input signals for the hyper logic processor 1 are debounced, i.e. for digital inputs 4 to 7, if they occur in the hyper logic operation selected by the user. Each signal must be present for more than 2 ms before it is considered valid and can be processed by the hyper logic.

9.3.2. PiXtend eIO Analog One - Modbus addresses and functions

9.3.2.1 Discrete inputs - reading overload feedback signals

The Analog One device has two discrete inputs, via which the overload feedback signals of the two current outputs can be recorded, for example to detect cable breaks. In CODESYS, each input is provided as one of the bits in a byte; other software platforms and Modbus RTU implementations may differ. Please refer to the relevant documentation for your system.
Use the Modbus RTU function code 2 to read the values.

Name	Address	Data type	Comment
Analog output 4 overload feedback	0	Bit	Overload feedback signal of the analog current output 4
Analog output 5 overload feedback	1	Bit	Overload feedback signal of the analog current output 5

Table 33: Analog One: Overload feedback signals as discrete inputs

If everything is OK and a load is connected to the analog current output, the overload feedback bits deliver a 0. Under CODESYS V3.5 this corresponds to the value FALSE.

On the other hand, if there is a cable break or no load has been connected, the overload feedback bits deliver a 1. Under CODESYS V3.5 this corresponds to the value TRUE, and there is an overload situation. In such cases check the wiring for faults.

9.3.2.2 Input register - Reading the analog inputs and further information

The input registers allow various information to be read from PiXtend elO devices. All available Modbus input registers are listed in the table below. References to breakdowns of the various bits are in the comment column.

Use function code 4 to read the input registers of the device.
The addresses of the input registers start counting at address 0. The device has 11 addresses or 11 input registers. The data type of each register is 16 bits of type WORD, i.e. 2 bytes long and can contain the numerical values from 0 to 65535.

Name	Address	Comment
Analog Input 0	0	Analog voltage input 0, a value range of 0 - 1023 is supported; this corresponds to the range: 0V - 10V
Analog Input 1	1	Analog voltage input 1, a value range of 0 - 1023 is supported; this corresponds to the range: 0V - 10 V
Analog Input 2	2	Analog voltage input 2, a value range of 0 - 1023 is supported; this corresponds to the range: 0V - 10V
Analog Input 3	3	Analog voltage input 3, a value range of 0 - 1023 is supported; this corresponds to the range: 0V - 10V
Analog Input 4	4	Analog current input 4, a value range of 0 - 1023 is supported; this corresponds to the range: 0mA - 20mA
Analog Input 5	5	Analog current input 5, a value range of 0 - 1023 is supported; this corresponds to the range: 0mA - 20mA
Analog Input 6	6	Analog current input 6, a value range of 0 - 1023 is supported; this corresponds to the range: 0mA - 20mA
Analog Input 7	7	Analog current input 7, a value range of 0 - 1023 is supported; this corresponds to the range: 0mA - 20mA
Status register	8	This register contains status information from the device as feedback to the user. It can be checked whether a function is switched on or not. In this way, it can be determined whether the watchdog timer is active. The table in chapter 9.3.2.4 Bits in the status register contains a breakdown of the status bits.
Error register	9	This contains a copy of the errors detected in the device. The table in chapter 9.3.2.5 Bits in the error register contains a breakdown of the error bits.
Version register	10	Version of the firmware as numbers, 101 = 1.01, 102 = 1.02 etc

Table 34: Analog One: Input register

9.3.2.3 Holding register – setting analog outputs and settings

The holding registers allow the setting of the individual outputs and the modification of various configurations, for example the activation of the watchdog timer. All available Modbus holding registers are listed in the table below. References to breakdowns of the various bits are in the comment column.

Use function code 6 and 16 to write in the holding registers of the device.
The addresses of the holding registers start counting at address 0. The device has 8 device addresses or 8 holding registers. The data type of each register is 16 bits of type WORD, 2 bytes long and can contain the numerical values from 0 to 65535.

Name	Address	Comment
Analog Output 0	0	Analog voltage output 0, a value range of 0 - 4095 is supported; this corresponds to the range: 0V - 10V
Analog Output 1	1	Analog voltage output 1, a value range of 0 - 4095 is supported; this corresponds to the range: 0V - 10V
Analog Output 2	2	Analog voltage output 2, a value range of 0 - 4095 is supported; this corresponds to the range: 0V - 10V
Analog Output 3	3	Analog voltage output 3, a value range of 0 - 4095 is supported; this corresponds to the range: 0V - 10V
Analog Output 4	4	Analog current output 4, a value range of 0 - 4095 is supported; this corresponds to the range: 0mA - 20mA
Analog Output 5	5	Analog current output 5, a value range of 0 - 4095 is supported; this corresponds to the range: 0mA - 20mA
Watchdog timer register	6	The watchdog timer register allows for the setting and switching of the device watchdog timer for monitoring the bus communication. The table in chapter 9.3.2.6 Bits in the watchdog timer register contains a breakdown of the configuration bits.
Config register	7	In the config register, various settings in the device can be changed. The table in chapter 9.3.2.7 Bits in the config register contains a breakdown of the configuration bits.

Table 35: Analog One: Holding register

9.3.2.4 Bits in the status register

The status register shows which function is active in the device and which is not. A cyclic request is recommended; it can be checked whether a function has been activated or not.

Low byte:

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

Table 36: Bits in the status register - low byte

Bit 0 - WDT ON → Watchdog time active

Off: The watchdog timer is off.

On: The watchdog timer is active and, depending on the setting, is reset or triggered after the specified time has expired.

High byte:

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

Table 37: Bits in the status register - high byte

The high byte of the status register contains no information or status bits and can be ignored.

9.3.2.5 Bits in the error register

The error register contains a copy of the errors that have occurred in the device. During operation the bits remain set until they are cleared by the QUIT ERR bit in the config register or a power cycle is executed. Exempted from this rule is the watchdog timer error bit; further information can be found in the explanation text for bit 4.

Low byte:

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

Table 38: Bits in the error register - low byte

Bit 4 - WDT ERR → Watchdog timer has expired

This bit indicates whether or not the watchdog timer had expired before the last power-up. Since the device enters the safe state when the watchdog timer expires, this bit can only be queried after a subsequent power cycle. The bit can be cleared during operation via the QUIT ERR bit in the config register or when another power cycle can be executed.

Bit 5 - TEL ERR → Function code or address error

If the device has been sent a telegram or message with an unsupported function code or the received Modbus RTU message contains a non-existent address, then this error bit is set. In the control program, it is possible to subsequently determine whether one of these errors has occurred since the last power-up, even if the ERR LED is not lit.

Bit 6 - COF ERR → Configuration error

This bit is set if a configuration error is detected in the device. In normal operation, this error will not occur unless the DIP switches are changed during operation. To remedy this situation, either the DIP switches must be returned to their original position or a power cycle is performed, in which case the new configuration is adopted.

Bit 7 - COM ERR → Communication error

This bit is set if the device detects a communication error. As a rule, there are three fault states. The device may detect a parity, frame, or overrun error.

A parity error usually occurs when the settings on the control device are not correct. The same applies to the frame error; in most cases the baud rate is incorrect and the device does not receive any valid data. The overrun error means that too much data is sent over the bus too quickly and the device cannot record everything.

In such cases, check the settings on the control box to see if they match the configuration of the device, if the wiring is correct, and that there are no faults on the bus.

High byte:

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

Table 39: Bits in the error register - high byte

The high byte of the error register contains no information or error bits and can be ignored.

9.3.2.6 Bits in the watchdog timer register

The watchdog timer register allows the user to influence the behavior of the device's watchdog timer. The possible settings range from the timeout time to the behavior of the watchdog timer.

If the watchdog timer is activated without further configuration, a timeout time of 16 milliseconds is preset. The watchdog timer is reset as soon as activity is detected on the bus, regardless of whether the data is intended for the device or not.

Low byte:

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

Table 40: Bits in the watchdog timer register - low byte

Bit 3...0 - WT3...WT0 → Watchdog timer timeout

The watchdog timer time defines how long the watchdog timer is active before it is triggered.
The following table shows which bit pattern must be set so that the watchdog timer takes over the timeout time when activated and works with it. Please note, if no bits are set, the watchdog timer uses the shortest time, 16 ms. This time span is much too short for Modbus RTU communication. Select a larger value in any case, e.g. 4 seconds.

Watchdog timer timeout:

WT3	WT2	WT1	WT0	Timeout
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

Table 41: Watchdog timer timeout overview

Procedure for setting/changing the watchdog time during operation:

- Watchdog is deactivated
- Select the desired watchdog timer timeout
- Turn on the watchdog

Bit 4 - WB → Watchdog timer behavior

Off:	The analog outputs remain in their current state when the watchdog timer expires.
On:	All analog outputs are actively switched to the value 0 when the watchdog timer expires. This corresponds to 0V, 0mA.

Bit 5 - WDTEN → Watchdog timer enabled

Off:	The watchdog timer is off.
On:	The watchdog timer is activated or switched on.

Bit 6 - WDTRST → Watchdog timer reset type

Off:	The watchdog timer reset is executed during bus activity, regardless of what data is involved. This setting can be used to determine whether the master is basically communicating or whether it has failed. If there is no communication on the bus within the selected watchdog timer time, the watchdog timer expires and is triggered.
On:	The watchdog timer reset is only executed when the device receives a valid Modbus RTU message from the master. If the device is not addressed by the master within the watchdog timer time with a valid Modbus RTU message, the watchdog timer expires. This setting is useful to determine that each device is addressed within the selected watchdog timer time in a large Modbus RTU network.

High byte:

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

Table 42: Bits in the watchdog register - high byte

The high byte of the watchdog register contains no information or configuration bits and can be ignored.

9.3.2.7 Bits in the config register

The internal error memory can be reset during operation with the bit in this register.

Low byte:

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

Table 43: Bits in the config register - low byte

Bit 0 - QUIT ERR → Quit errors

Off: The device sets the error bits in the error memory.

On: If this bit is set to 1, the internal device error memory is cleared. After setting the bit, check whether all errors in the error register have been cleared.

High byte:

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

Table 44: Bits in the config register - high byte

The high byte of the config register contains no information or configuration bits and can be ignored.

9.4. Modbus RTU error numbers

The Modbus RTU protocol uses different error numbers (exception codes) which a master can receive as response from a slave.

PiXtend elO devices have two error numbers that are reported. The following table provides an overview of these errors.

Error numbers:

Name	Number	Comment
Illegal function	1	The function code used is not supported.
Illegal data value	3	The transmitted data contains an error or the specified coil/discrete inputs/register address is incorrect or does not exist.

Table 45: Modbus RTU error numbers (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

Figure 12: CODESYS V3.5 - Modbus RTU - Device reports an error

9.5. Example CODESYS V3.5

This chapter shows a simple example for each PiXtend eIO device with the PLC programming tool CODESYS V3.5.

In the examples, we assume that the devices are set to default settings. The Digital One device has the device address 1 and the Analog One device the device address 3. The switches 1 - 6 of the DIP block are set to 2 - CONFIG in the lower position (they are switched off). COM port 1 in CODESYS V3.5 is set with a baud rate of 19200, with parity "even", one stop bit and 8 data bits (8E1). The respective device is wired and connected to the RS485 port of the PiXtend V2 -L-. The PiXtend V2 -S- and PiXtend V1.x need a USB-to-RS485 dongle.

9.5.1. PiXtend eIO Digital One

We show how to set different digital outputs with the CODESYS V3.5 Modbus RTU Master on the Digital One device and how to read them again via the digital inputs. The desired bit sequence is written to the Digital One coils and we read back the state via the discrete inputs. We use the function codes 02 and 15 in this example.

First the digital output 0 must be connected to the digital input 0 via a cable. Refer to the PiXtend eIO Hardware Manual for further details on connecting a power supply and wiring a Digital One device.

9.5.1.1 Create a new project and select PLC

Open the

CODESYS file menu and select

New project...

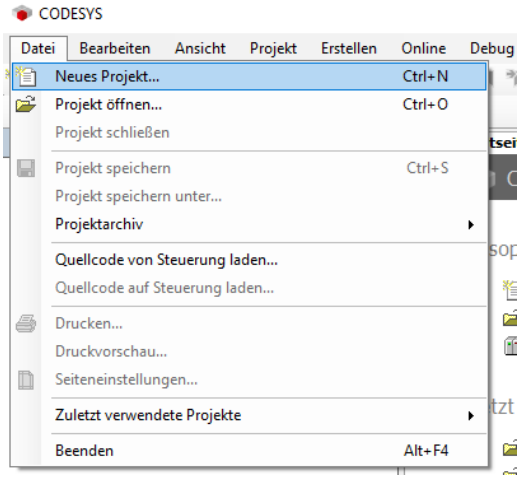


Figure 13: CODESYS - File menu

In the New Project menu, select the Project category and select the Standard Project entry under Templates (right). Enter a project name, for example, Modbus - Digital One example. Now select the location to save the project.

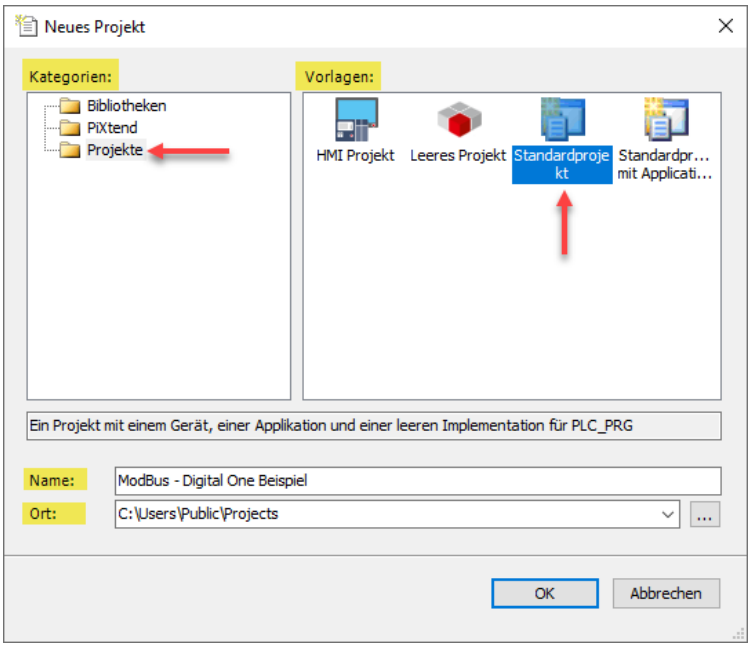


Figure 14: CODESYS - New project

In the Standard Project window, select your PLC. All PiXtend V2 users need to select the Raspberry Pi as the device.

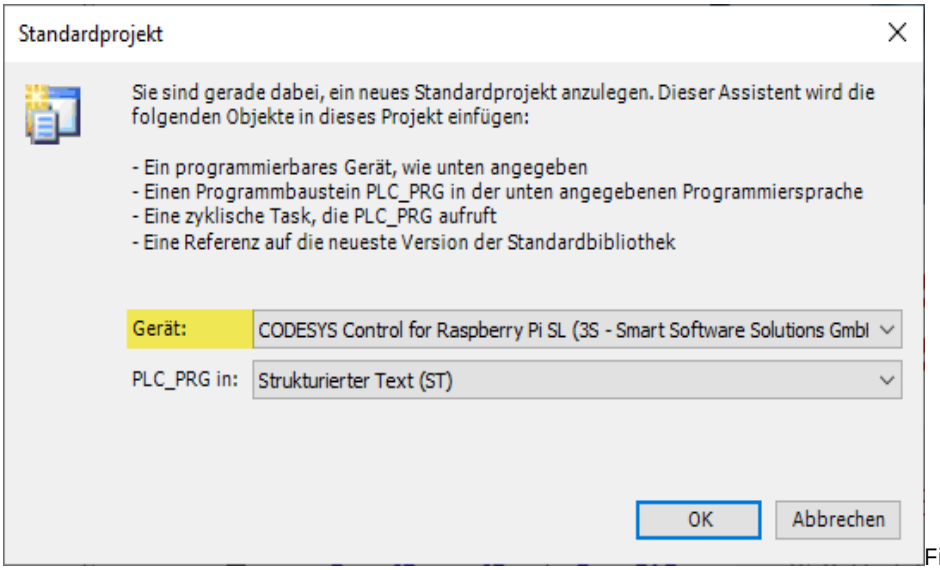


Figure 15: CODESYS - Standard project - device selection

Click on OK and the new project is created.

9.5.1.2 Add Modbus COM Port

Your project has been created so now click on the entry Device (1) in the Device window. Right-click on the entry and select the Add Device entry... (2).

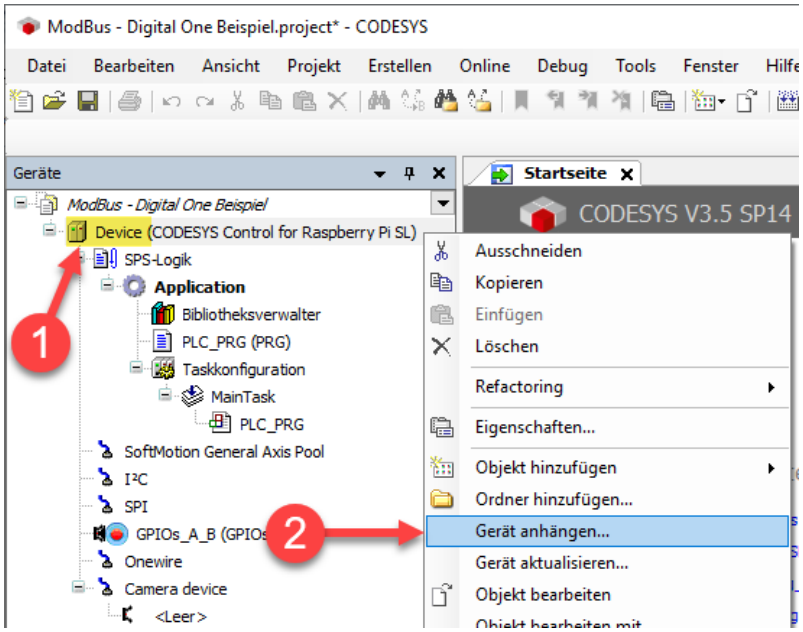


Figure 16: CODESYS - Add device menu

In the Add Device window, use the plus sign to open the Fieldbus entry. Open the Modbus entry and select the device Modbus COM port.

Select this entry and click the Add device button in the lower right corner of the window.

Close the window and select the just added Modbus_COM_Port entry in the device window.

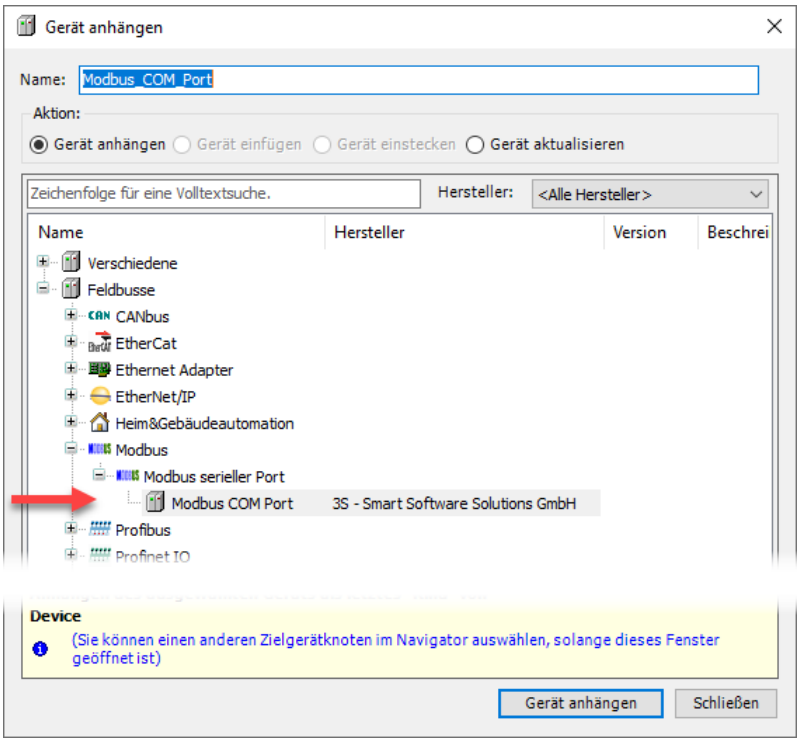


Figure 17: CODESYS - Add device window

Right-click on the entry and select Add Device from the menu. In the window Add Device, search for the entry Modbus Master, COM port. Add this device to the existing Modbus COM port.

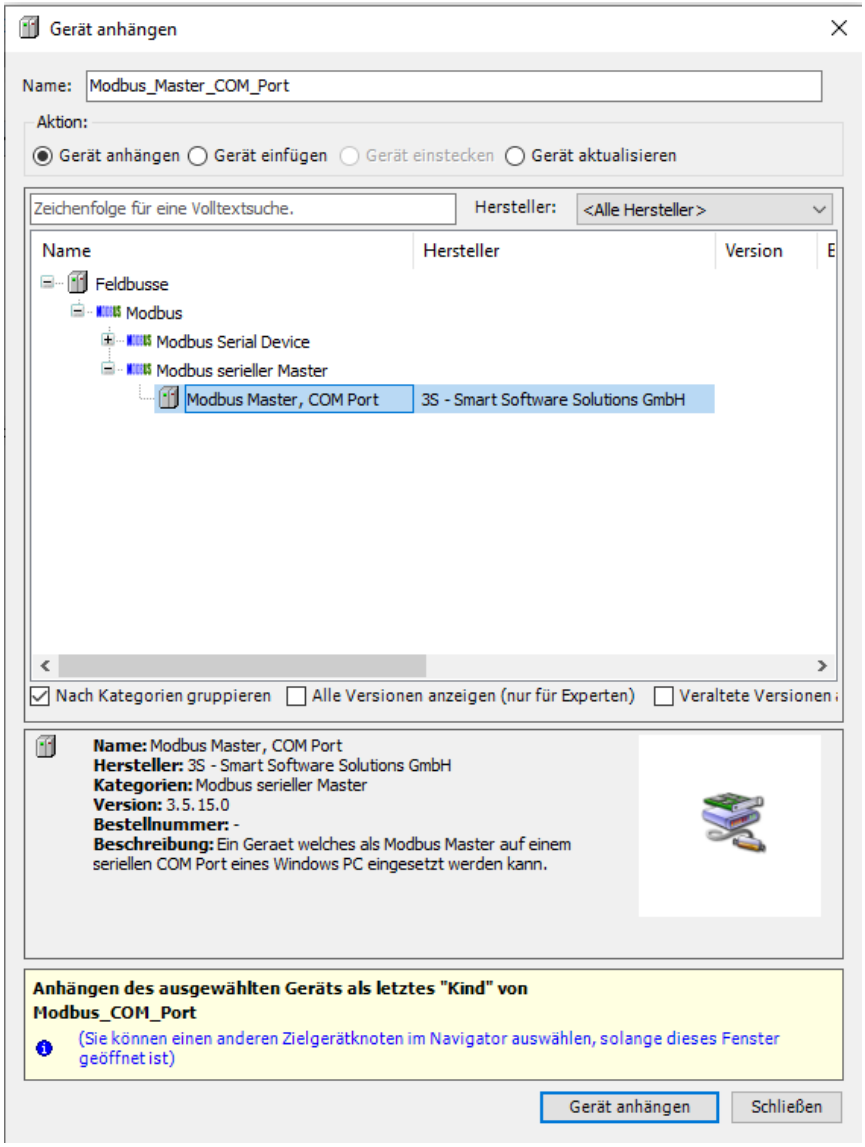


Figure 18: CODESYS - Add device - Modbus master

Close the window, select the entry Modbus_Master_COM_Port in the device window. Right-click on the entry and select Add Device from the menu. In the Add Device window, search for the entry Modbus Slave, COM port and add this device to the existing Modbus_Master_COM_Port.

Close the window. All devices have been added and are available. Now only the configuration is missing.

In the CODESYS devices window, you should find a configuration as shown in figure 19, independent of your PLC.

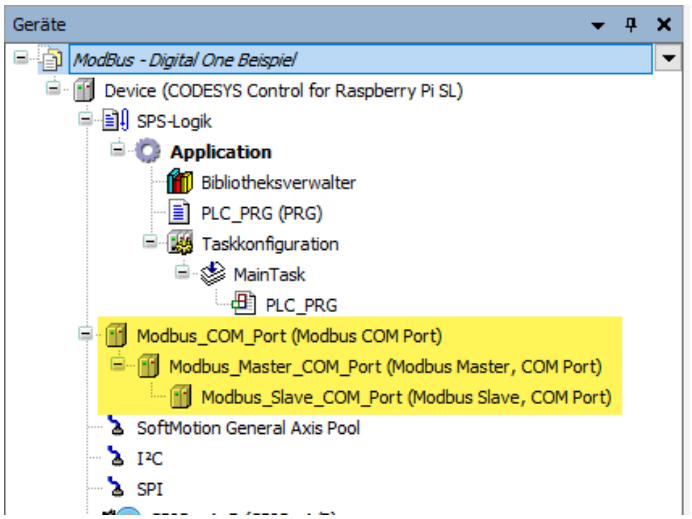


Figure 19: CODESYS - All devices added

9.5.1.3 Configure the Modbus COM Port

Double click on the device entry Modbus_COM_Port to open the settings page for the serial port. Select the General tab and make the following configuration:

COM port: 1
Baud rate: 19200, Parity: even, Data bits: 8 and Stop bits: 1

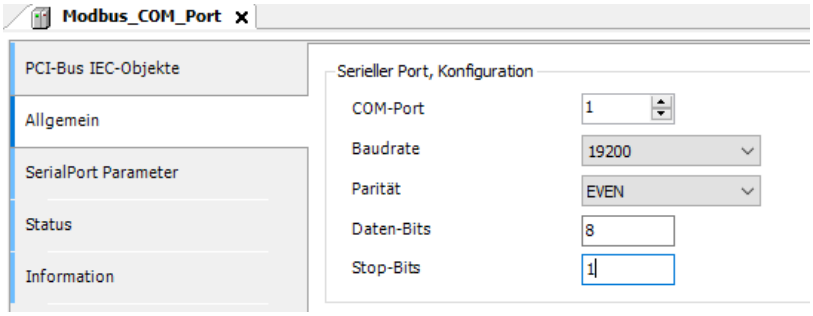


Figure 20: CODESYS - serial settings

If you do not use a PiXtend V2 -S- / -L- or V1.x as the PLC, refer to the manual of your PLC to find out which COM port is suitable for RS485 and set this number under COM port.

9.5.1.4 Configuring Modbus Master COM Port

In the Modbus Master COM port, a configuration must be made in the General tab. Activate the option "automatic restart communication".

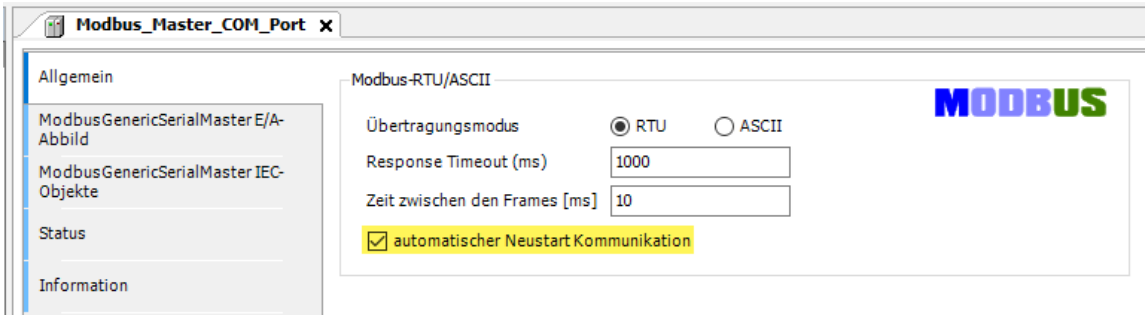


Figure 21: CODESYS - Modbus Master settings

9.5.1.5 Configuring Modbus Slave COM Port

No adjustment is made in the General tab for the Modbus slaves. The Digital One device has device address 1 by default and the timeout remains unchanged.

The settings in CODESYS correspond to the following figure:

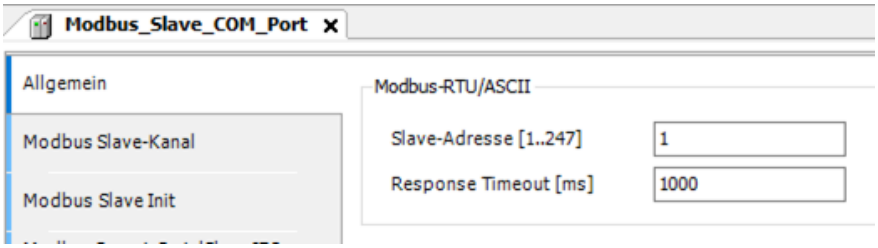


Figure 22: CODESYS - Modbus Slave settings

9.5.1.6 Configure Modbus slave channel (function codes)

Change to the tab Modbus Slave Channel and click on the button Add Channel at the bottom right.
Assign a name, for example Digital Outputs. Please note, this cannot be changed later. It is important that you set the access type to function code 15 - Write Multiple Coils and enter an offset of 0 and a length of 8 in the WRITE Register section.
All other settings remain unchanged.

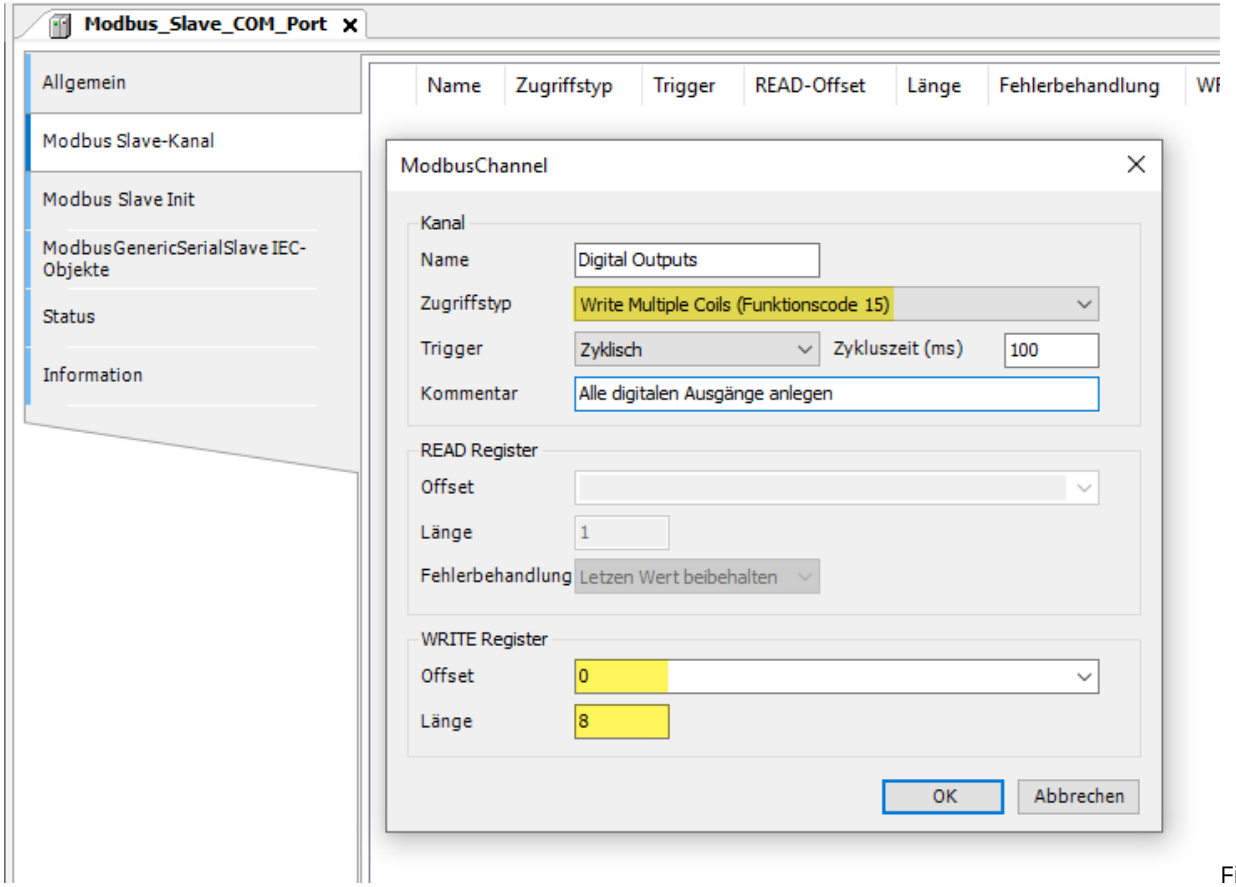


Figure 23: CODESYS - Create Modbus Slave communication channel

Click on the OK button to add a new Modbus RTU communication channel in CODESYS.
Then click the Add Channel button immediately afterwards to add another Modbus RTU communication channel.

This second channel is to read the digital inputs from the Digital One device.

For this second channel, set the Function Code 2 - Read Discrete Inputs under Access Type. In the READ Register section, enter a 0 for the Offset and an 8 for the Length. Thus there is one bit for each digital input. You can change the Error Handling setting to Set to ZERO. If the Digital One is not accessible, all digital inputs in CODESYS are set to FALSE. The other settings remain unchanged.

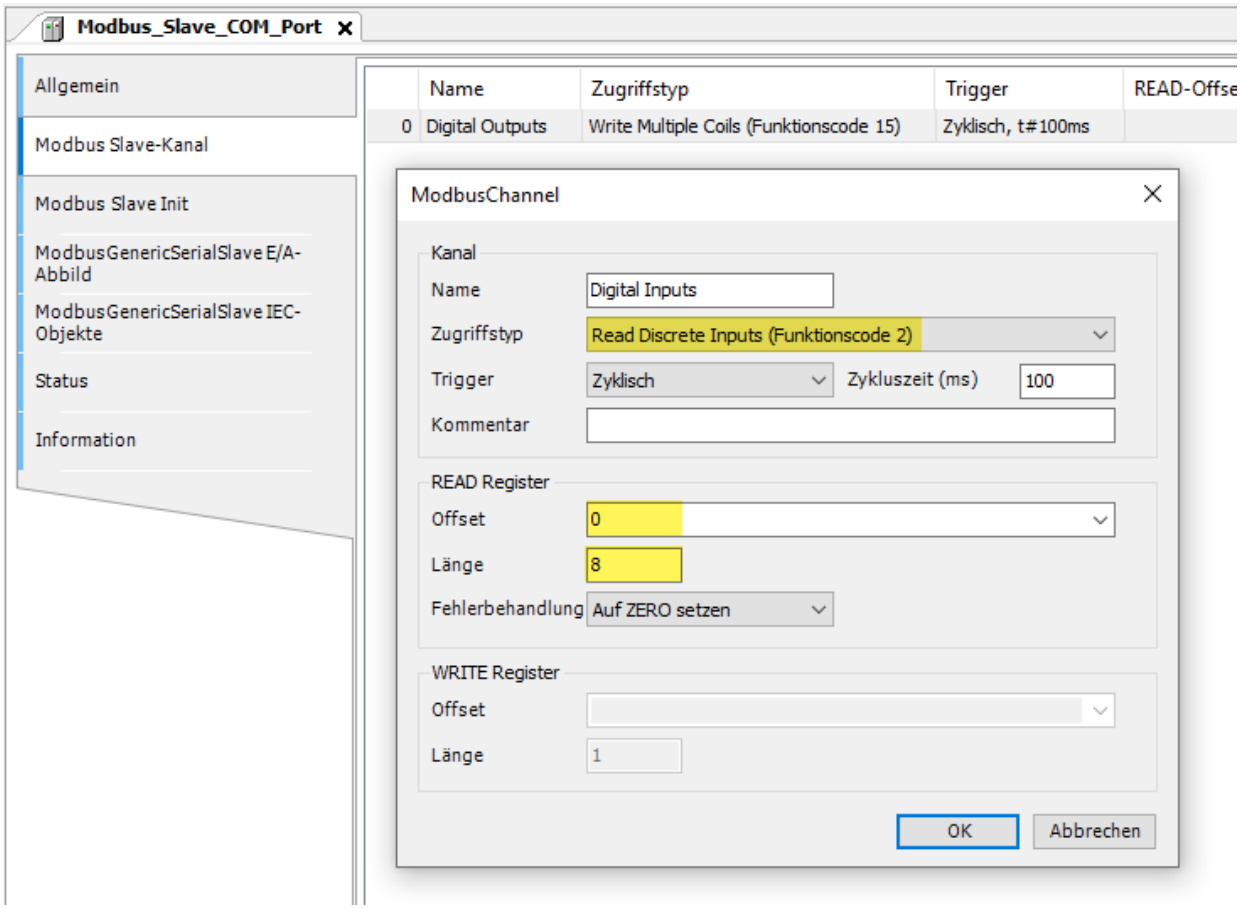


Figure 24: CODESYS - Create Modbus Slave communication channel (2)

Click on the OK button to add a new Modbus RTU communication channel in CODESYS.

The settings in CODESYS correspond to the following figure:

	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		

Figure 25: CODESYS - Modbus Slave communication channel overview

The next step is to change the variable update to the setting Enabled 2. Go to the tab ModbusGenericSerialSlave I/O Mapping in the Modbus Slave and change the setting to the entry Enabled 2 in the lower right corner.

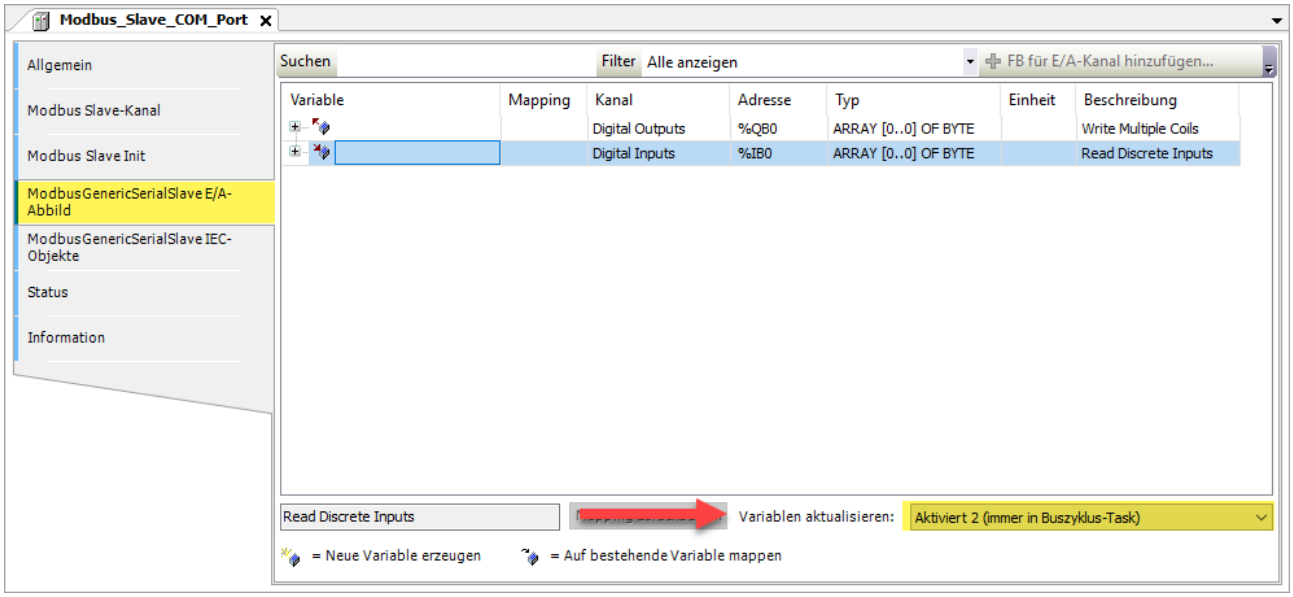


Figure 26: CODESYS - Update variables

Now a small program must be written, which switches the serial port on PiXtend V2-L- from RS232 to RS485. If you are using a PiXtend V2 -S- or V1.x with an RS485 dongle or another PLC, you can skip the next step.

9.5.1.7 PiXtend V2 -L- Switching the serial port RS232 to RS485

The PiXtend V2 -L- has an integrated RS485 interface; this must be activated by CODESYS, as the RS232 interface is activated as the standard setting. With the GPIO18 on the Raspberry Pi, you can choose between both serial interface types. Set up the GPIO18 of the Raspberry Pi as output and assign the variable name RPI_GPIO_18 in the GPIO I/O mapping.

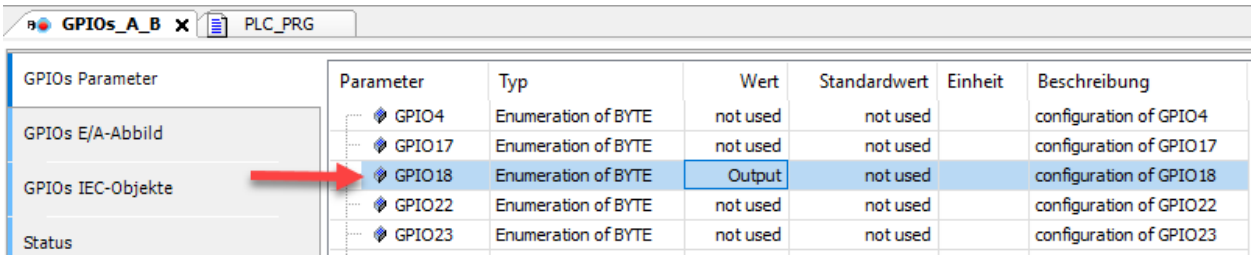


Figure 27: CODESYS - GPIO18 as an output

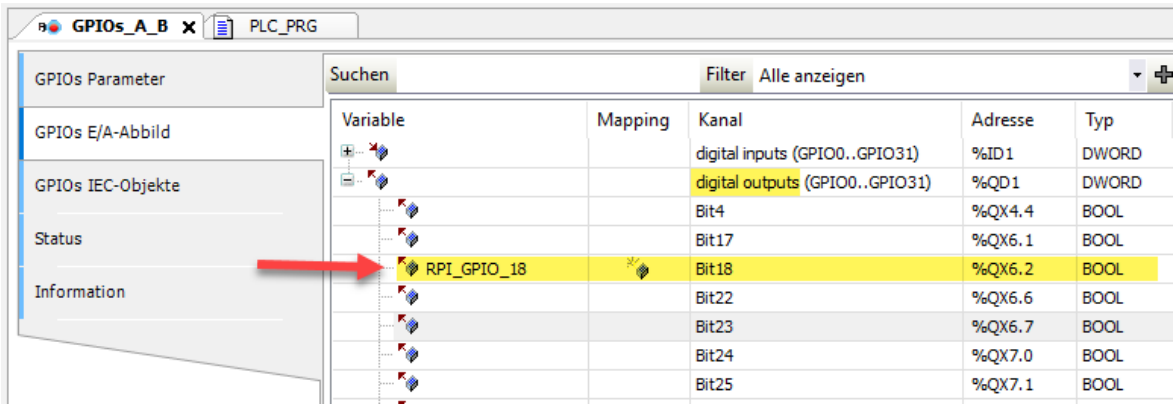


Figure 28: CODESYS - GPIO18 as a variable

All that is missing now is a very short program that sets the RPI_GPIO_18 variable to TRUE at startup. One possibility on how this can work is in the following picture:

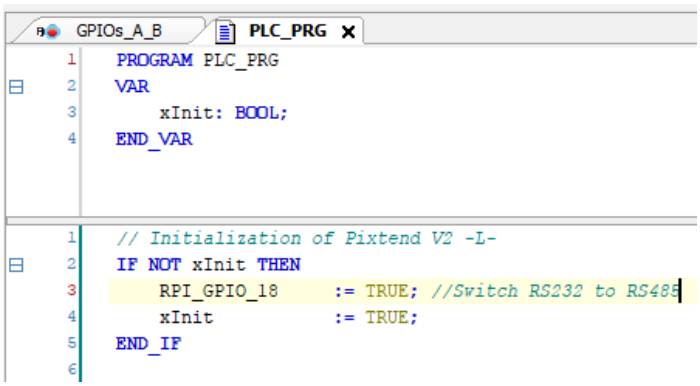


Figure 29: CODESYS - activating RS485

Another option is to write the following instruction in the first line of the PLC_PRG block:

RPI_GPIO_18 := TRUE; //Switch RS232 to RS485

That's it, now the program just has to be transferred to the controller and started and off you go!

Further information about the serial interface on the Raspberry Pi and PiXtend V2 devices can be found in the chapters 6.8 Prepare the serial interface for one's own SD card image, 6.9 Adding a USB-to-RS485 dongle to PiXtend V2 -S- and 6.10 PiXtend V2 -L- - Checking the serial interface.

9.5.1.8 Transfer and start the program

Connect to the controller and transfer the program. If everything went well, then start the program by pressing the start button or F5.

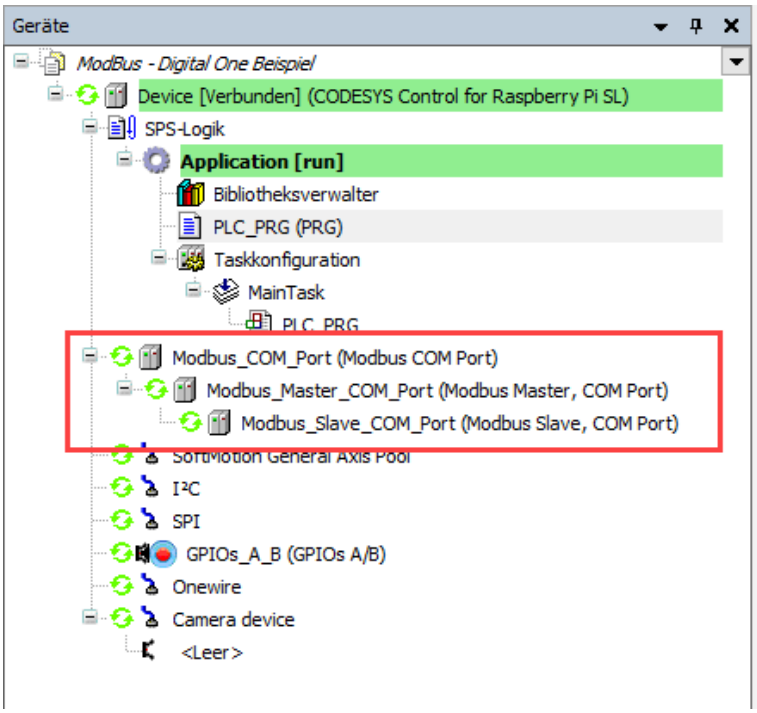


Figure 30: CODESYS - Modbus Bus running

It worked when you see green circles of arrows in front of the three Modbus devices in the CODESYS device window. If the color of the circles is orange, there is no CODESYS V3.5 Runtime License installed on the Raspberry Pi; there is not an error with the Modbus master.

There is a restriction; the CODESYS Runtime runs for two hours and then stops. The Modbus RTU Master, however, can only be tested for 30 minutes. It then stops working and the Raspberry Pi must be restarted.

If you use a different controller, then consult the manufacturer if all circles are not green. You may need an additional Modbus RTU license to use Modbus RTU under CODESYS on your PLC.

9.5.1.9 Switching output and reading input

The program and the Modbus bus are working. Now when we switch the digital output 0 to TRUE, the digital input 0 should also switch to TRUE. For this, digital output 0 must be connected to digital input 0. Go back to the ModbusGenericSerialSlave I/O Mapping tab in the Modbus_Slave_COM_Port device. Open both channels until all input and output bits of the slave are visible.

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

Figure 31: CODESYS - Modbus Slave coils and discrete inputs overview

Use the Prepared Value column to set Bit0, which is the digital output 0 on the Digital One device. After a short time, Bit0 of the digital inputs should become TRUE, which means that a HIGH level is present at digital input 0.

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
		Digital Inputs[0]	%IB0	BYTE	1
		Bit0	%IX0.0	BOOL	TRUE
		Bit1	%IX0.1	BOOL	FALSE

Figure 32: CODESYS - Digital output and input active

9.5.2. PiXtend eIO Analog One

In the example, we show how to set an analog output with the CODESYS V3.5 Modbus RTU Master on the Analog One device and how to read it in again via an analog input. We use the function codes 04 and 06 in this example. The analog output 0 must be connected via a cable to the analog input 0. Refer to the PiXtend eIO Hardware Manual for further information on connecting a power supply and wiring an Analog One device.

9.5.2.1 Create a new project and select PLC

Open the CODESYS file menu and select
New project...

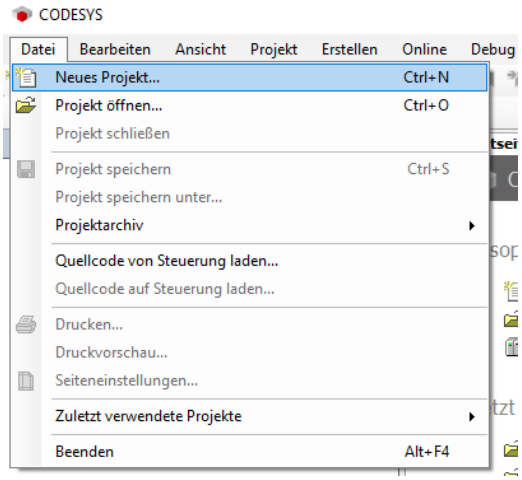


Figure 33: CODESYS - File menu

Select the Projects category in the New Project window and select Default Project under Templates (right). Enter a project name, for example, Modbus - Digital Analog One example. Now select the location to save the project.

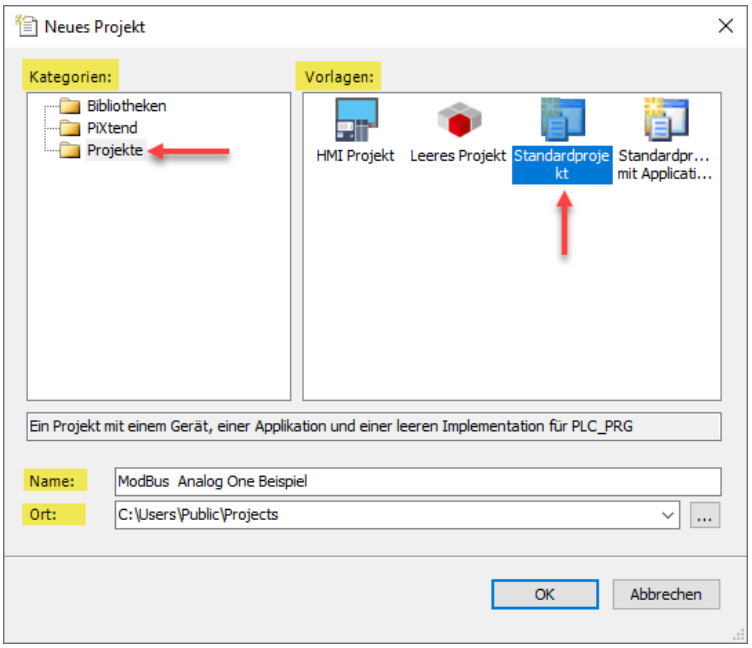


Figure 34: CODESYS - New project

In the Standard Project window, select your PLC. PiXtend V2 users need to select the Raspberry Pi as the device.

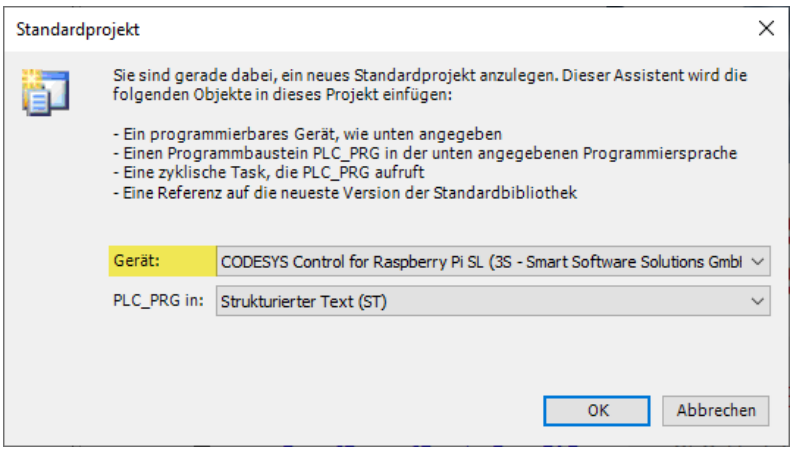


Figure 35: CODESYS - Standard project - device selection

Click on OK and the new project is created.

9.5.2.2 Adding Modbus COM Port

After the project has been created, click on the entry Device (1) in the device window to select it. Right-click on the entry and select Add Device in the menu that appears (2).

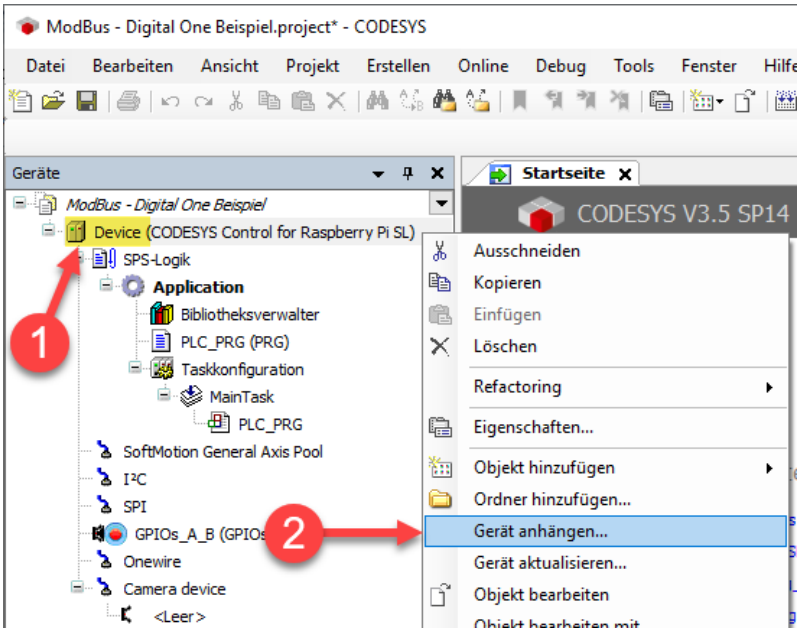


Figure 36: CODESYS - Add device menu

In the Add Device window, open the Fieldbus entry with the plus sign. Open the Modbus entry and find the device Modbus COM port.

Select the entry and click on the Add Device button in the bottom right window.

Close the window and select the Modbus_COM_Port entry just added in the device window.

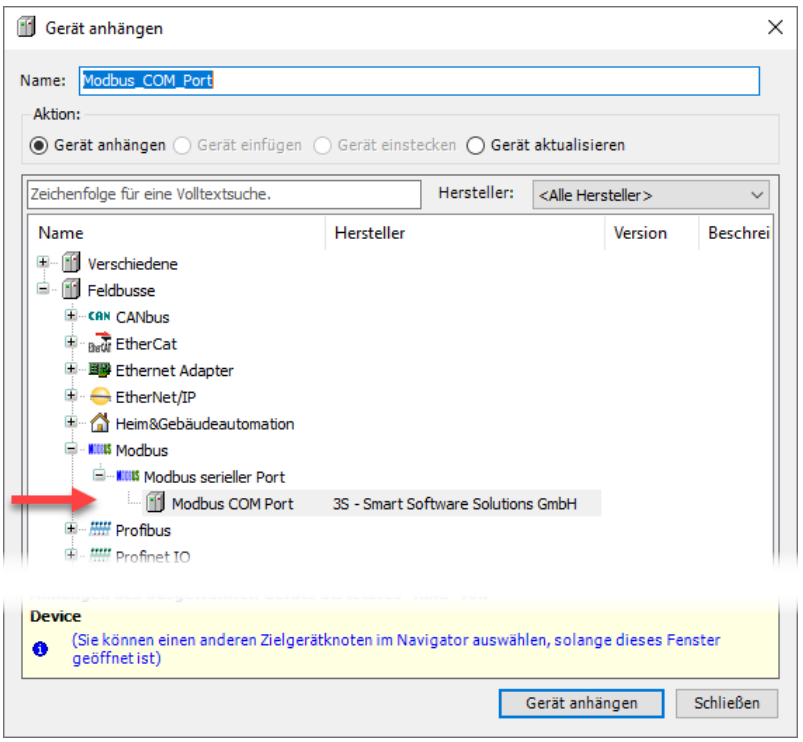


Figure 37: CODESYS - Add device window

Right-click on the entry and select Add Device from the menu. In the Add Device window, search for the entry Modbus Master, COM port and add this device to the existing Modbus COM port.

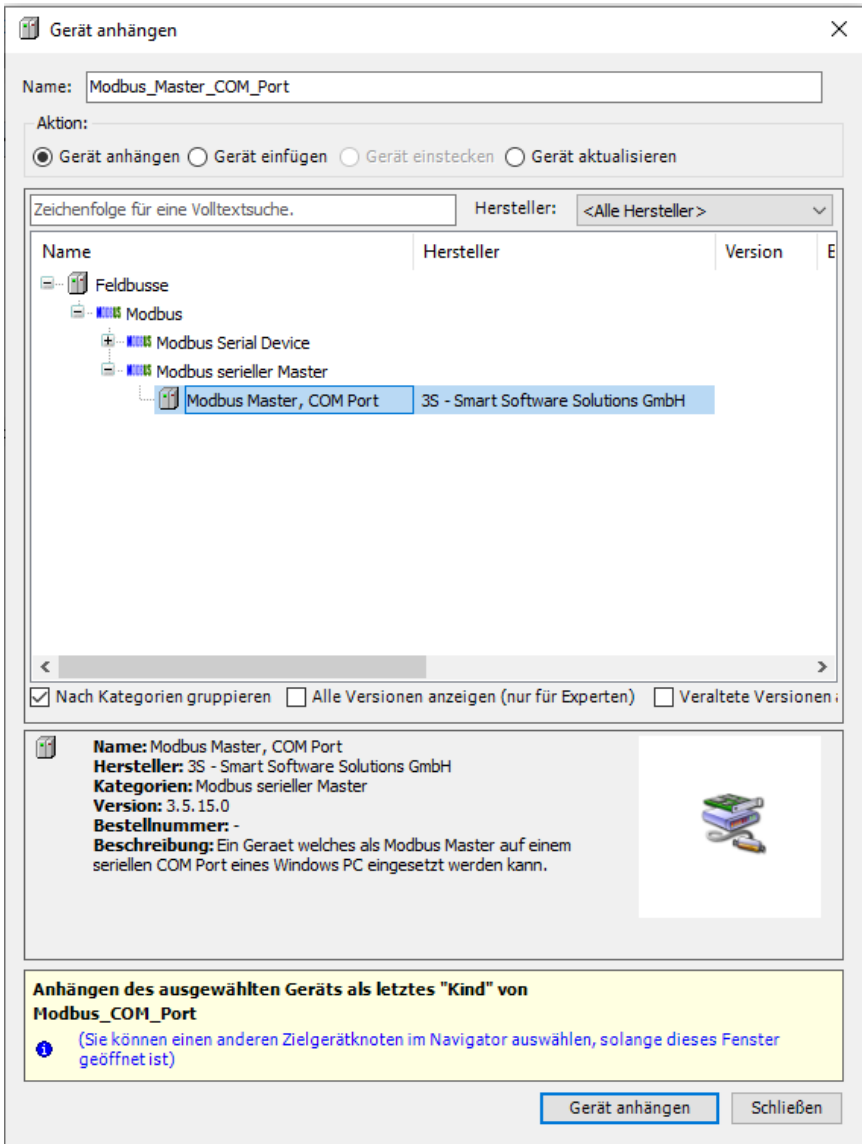


Figure 38: CODESYS - Add Modbus master

Close the window and select the entry Modbus_Master_COM_Port in the device window. Right-click on the entry and select Add Device from the menu. In the Add Device window, search for the entry Modbus Slave, COM port and add this device to the existing Modbus_Master_COM_Port.

Close the window; all devices have been added and are available. Now only the configuration is missing.
In the CODESYS devices window, you should find a configuration as shown in following figure, independent of your PLC.

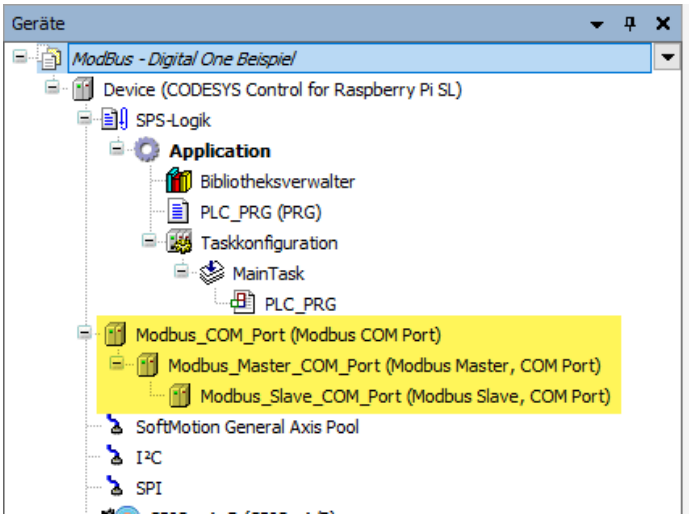


Figure 39: CODESYS - All devices added

9.5.2.3 Configuring Modbus COM Port

Double click on the device entry Modbus_COM_Port to open the settings page for the serial port. Click on the General tab and make the following configuration:

COM port: 1

Baud rate: 19200, Parity: even, Data bits: 8 and stop bit: 1

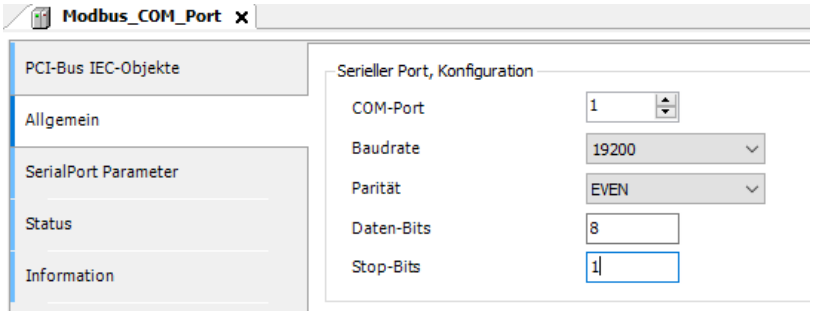


Figure 40: CODESYS - Modbus serial settings

If you do not use a PiXtend V2 -S- / -L- as the PLC, please refer to the manual of your PLC to see which COM port is suitable for RS485 and set this number under COM port.

9.5.2.4 Configuring Modbus Master COM Port

In the Modbus Master COM port, a configuration must be made in the General tab.
Activate the option "automatic restart communication".

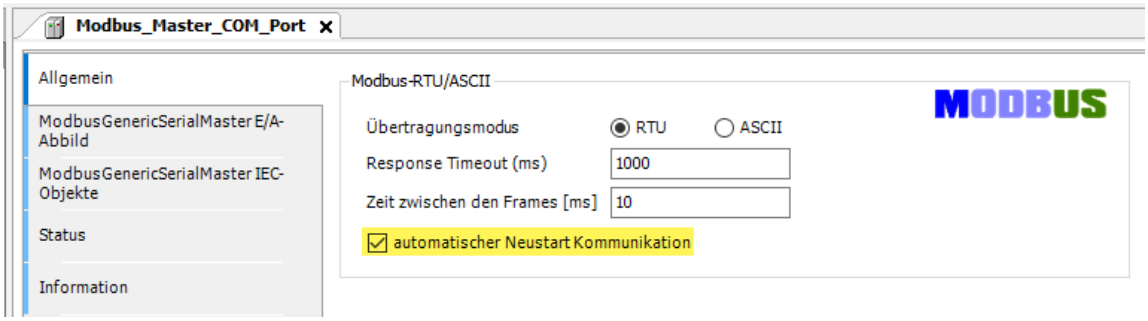


Figure 41: CODESYS - Modbus Master settings

9.5.2.5 Configuring Modbus Slave COM Port

For the Modbus slave, an adjustment must be made in the General tab, because the Analog One device has device address 3 by default. The timeout remains unchanged.

The settings in CODESYS correspond to the following figure:

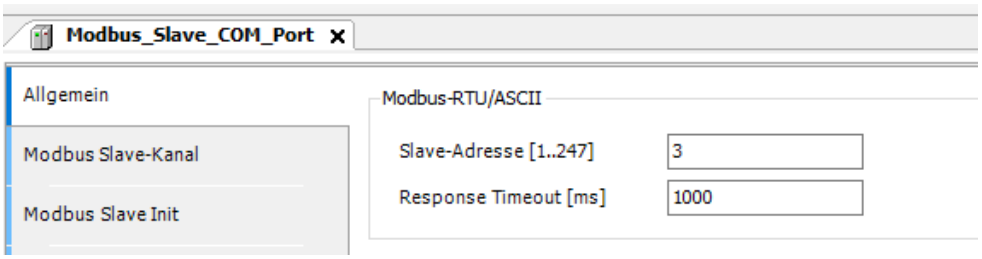


Figure 42: CODESYS - Modbus Slave settings

9.5.2.6 Configure Modbus slave channel (function codes)

Change to the tab Modbus Slave Channel and click on the button Add Channel at the bottom right.

Assign a name, for example Analog Output 0. Please note, this cannot be changed later. It is important that you set the access type to function code 6 - Write Single Register and enter an offset of 0 in the WRITE Register section.
All other settings remain unchanged.

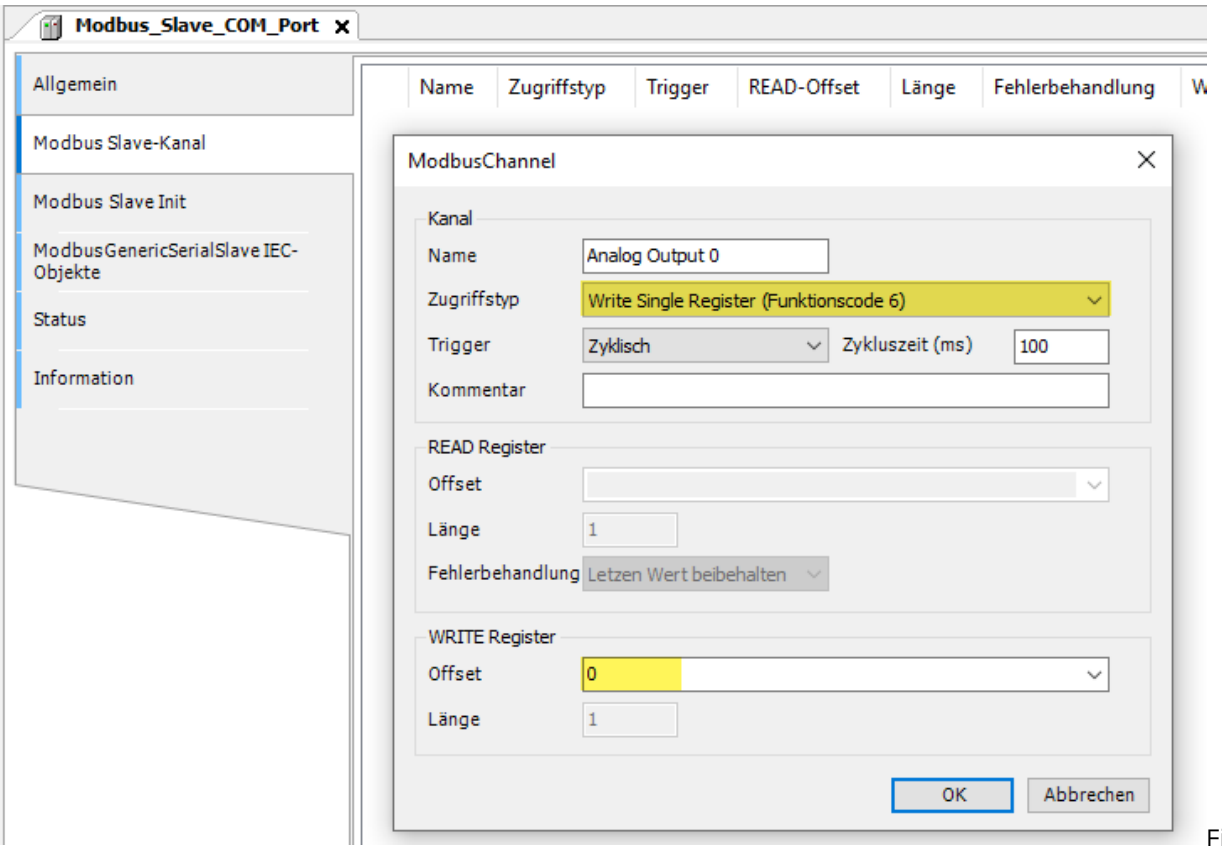


Figure 43: CODESYS - Create Modbus communication channel (1)

Click on the OK button to add a new Modbus RTU communication channel in CODESYS.

Then click the Add Channel button again afterwards to add another Modbus RTU communication channel. This second channel is to read the analog input 0 from the Analog One device .

Set the function code 4 - Read Input Registers for the second channel under Access Type. In the READ Register section, enter a 0 for the Offset and a 1 for the Length. This means that we only read in the first analog input. You can change the Error Handling setting. Set to ZERO means, if the Analog One device is not accessible, the analog input 0 in CODESYS is set to 0. All other settings remain unchanged.

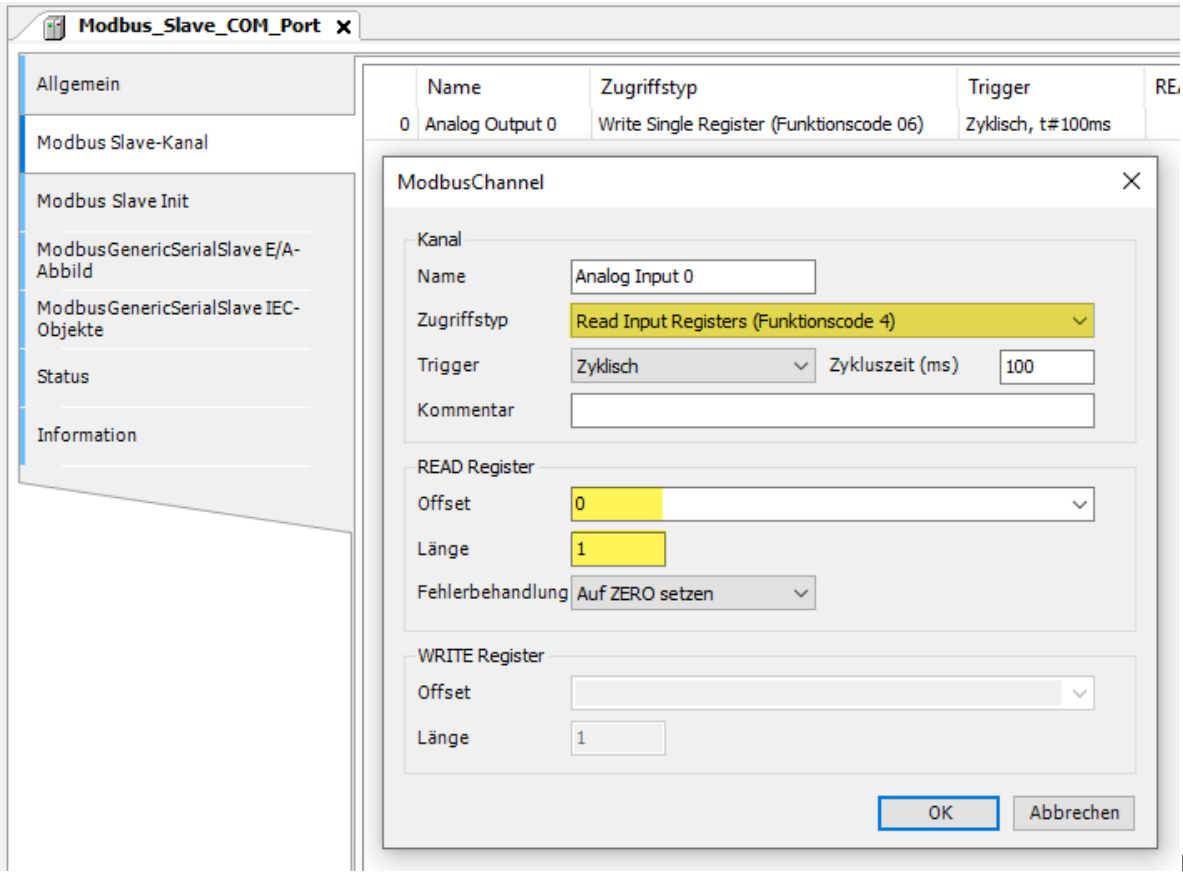


Figure 44: CODESYS - Create Modbus communication channel (2)

Click on the OK button to add a new Modbus RTU communication channel in CODESYS.

The settings in CODESYS correspond to the following figure:

	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		

Figure 45: CODESYS - Modbus communication channel overview

The next step is to change the variable update to the setting Enabled 2. Go to the tab ModbusGenericSerialSlave I/O Mapping in the Modbus Slave and change the setting to the entry Enabled 2 in the lower right corner.

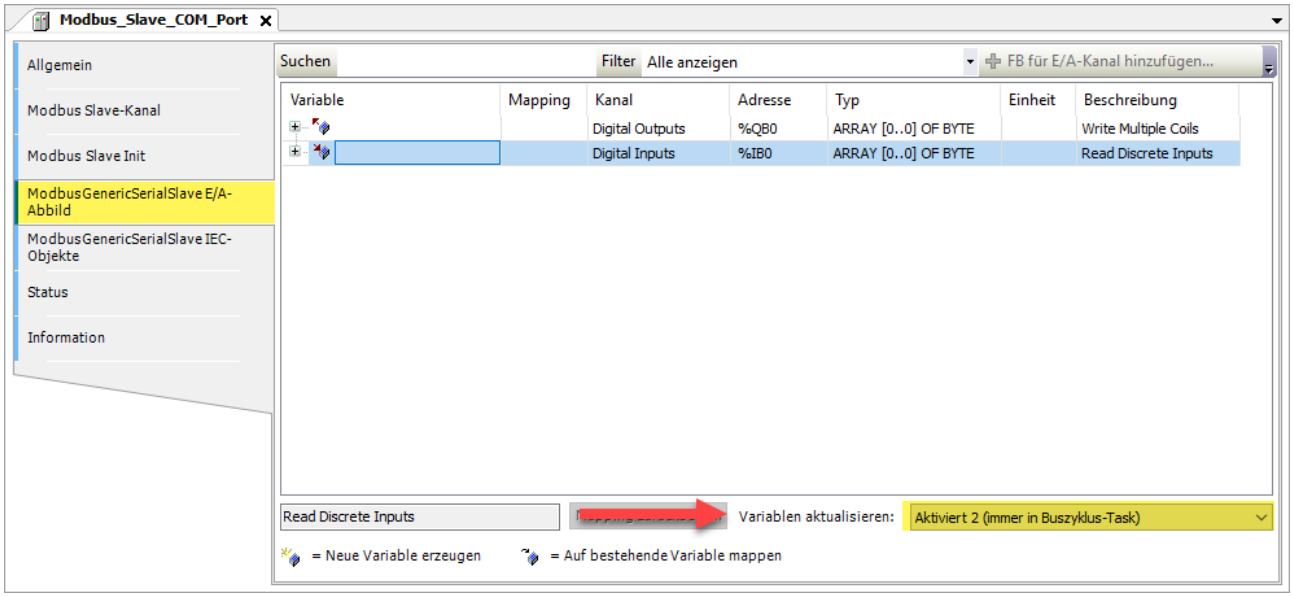


Figure 46: CODESYS - Update variables

Now a small program must be written, which switches the serial port on PiXtend V2-L- from RS232 to RS485. If you are using a PiXtend V2 -S- with an RS485 dongle or another PLC, you can skip the next step.

9.5.2.7 PiXtend V2 -L- Switching the serial port RS232 to RS485

The PiXtend V2 -L- has an integrated RS485 interface; this must be activated by CODESYS, as the RS232 interface is activated as the standard setting.

With the GPIO18 on the Raspberry Pi, you can switch between both serial interface types. Set up the GPIO18 of the Raspberry Pi as output and assign the variable name RPI_GPIO_18 in the GPIO I/O mapping.

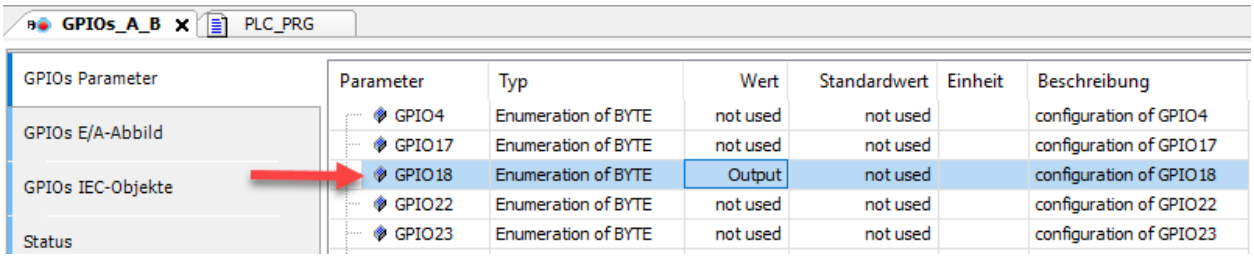


Figure 47: CODESYS - GPIO18 defined as an output

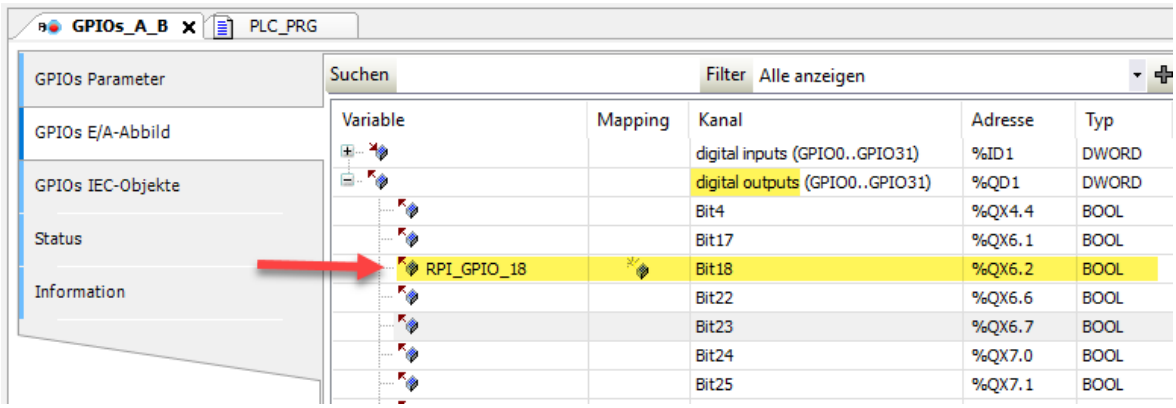


Figure 48: CODESYS - GPIO18 as a variable

All that is missing now is a very short program that sets the RPI_GPIO_18 variable to TRUE at startup. You can see one possibility for conversion in the following figure:

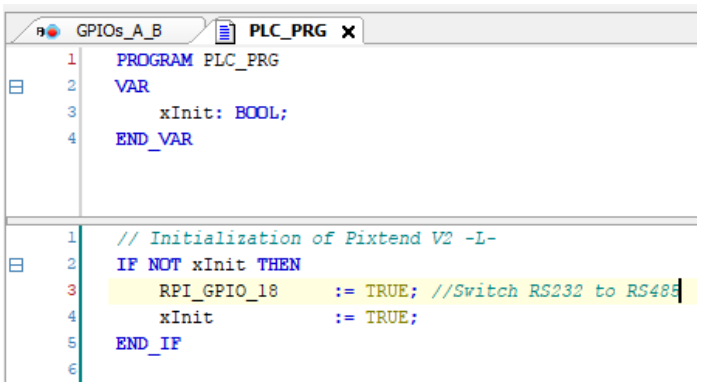


Figure 49: CODESYS - activating RS485

Another option is to write the following instruction in the first line of the PLC_PRG block:

RPI_GPIO_18 := TRUE; //Switch RS232 to RS485

That's it, now the program just has to be transferred to the controller and started and off you go!

Further information about the serial interface on the Raspberry Pi and PiXtend V2 devices can be found the chapters 6.8 Prepare the serial interface for one's own SD card image, 6.9 Adding a USB-to-- RS485 dongle to PiXtend V2 -S- and 6.10 PiXtend V2 -L- - Checking the serial interface.

9.5.2.8 Transfer and start the program

Connect to the controller, transfer the program and start it with the start button or with F5.

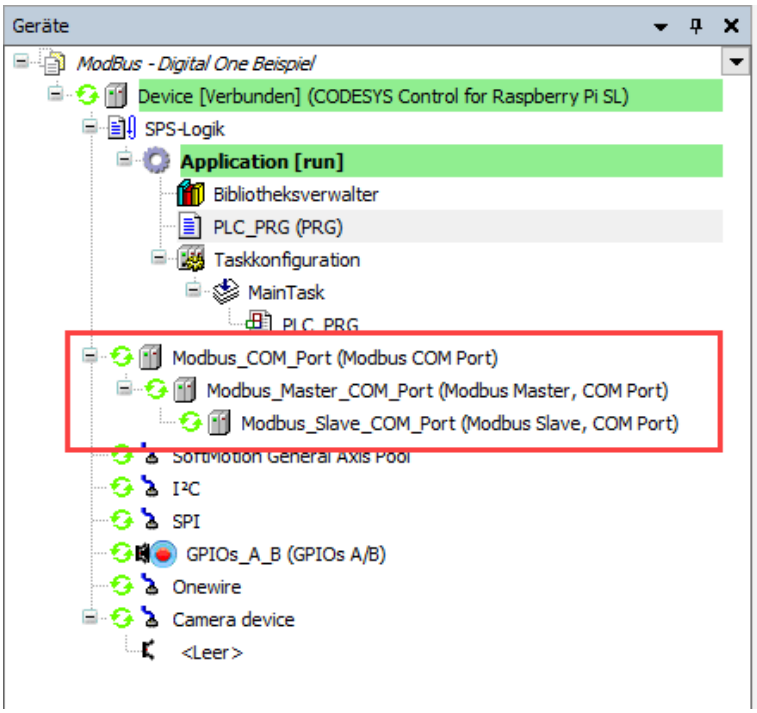


Figure 50: CODESYS - Modbus Bus running

It worked when you see green circles of arrows in front of the three Modbus devices in the CODESYS device window. If the color of the circles is orange, there is no CODESYS V3.5 Runtime License installed on the Raspberry Pi; there is not an error with the Modbus master.

There is a restriction; the CODESYS Runtime runs for two hours and then stops. The Modbus RTU Master, however, can only be tested for 30 minutes. It then stops working and the Raspberry Pi must be restarted.

If you use a different controller, then consult the manufacturer if all circles are not green. You may need an additional Modbus RTU license to use Modbus RTU under CODESYS on your PLC.

9.5.2.9 Set output to 5 volts and read input

The program and the Modbus bus are working. Now set the analog output 0 to 2048; this corresponds approximately to a voltage of 5 volts and you can read this value at the analog input 0 again. For this, analog output 0 must be connected to analog input 0 on the Analog One device. Go back to the ModbusGenericSerialSlave I/O Mapping tab in the Modbus_Slave_COM_Port device and open both channels. If you have set the Analog Output 0 to 2048, a value of about 512, which corresponds to about 5 volts, should be visible at the Analog Input 0.

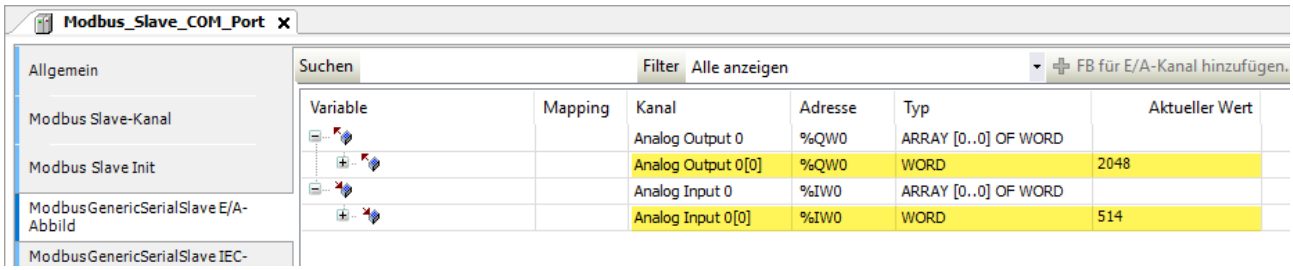


Figure 51: CODESYS - Modbus slave analog output 0 and analog input 0

Check out the download section on our website for more CODESYS examples and examples of other programming languages and tools.

9.6. Example Python

This chapter shows how to use the Python programming language, the pyModbus module with the Digital One device and the Analog One device to communicate with the Modbus RTU protocol. Before starting, make sure that the Python packages RPi.GPIO, version 0.6.5 or later and pyModbus, version 2.2.0 or later, are installed. Use raspi-config to disable the login shell on the ttyAMA0 or serial0 interface. The installed Python 2.7 module can be checked with the command pip freeze. If using Python 3, then use pip3 freeze. If the Python modules or packages are not available, use the following commands to install the required features:

Install pyModbus: sudo pip install -U pymodbus

sudo pip3 install -U pymodbus

Install RPi.GPIO: e.g.: sudo apt-get install rpi.gpio

9.6.1. PiXtend elO Digital One

This example shows how to use the Python programming language to set various digital outputs on the Digital One device and read them back via the digital inputs. We write the desired bit sequence on the Digital One coils and read it back via the discrete inputs. We use the function codes 02 and 15 in this example. The digital output 0 must be connected to the digital input 0 via a cable. Refer to the PiXtend elO Hardware Manual for further information on connecting a power supply and wiring a Digital One.

If working with the programming language Python and using a PiXtend V2 -L-, you have several possibilities to communicate with the PiXtend elO devices. In the SD card Basic image from version 2.1.6.0, we use the full-fledged UART (called PL011, serial0 or ttyAMA0) of the Raspberry Pi. Only this interface allows the parity bit to be used.

If you use an RS485 USB dongle, your serial device will probably be called ttyUSB0. If using the serial interface on the controller of another manufacturer, please refer to its device manual for information. Depending on which interface you use, comment in or out the corresponding lines of code or modify them as you need them.

The following example is based on the assumption that you are working with a PiXtend V2 -L-, with the full-fledged UART (ttyAMA0) and use the Basic image version 2.1.6.0 or later. The Digital One device has the default settings, device address 1, CONFIG switch 1-6 all to OFF. These settings correspond to 19200 baud, 8 data bits, parity "even" and 1 stop bit.

File: 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 DO0 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 DO0 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
```

The Python program works best with root privileges so that Python has access to the serial port.
Start the program as follows:

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

One cycle of the main program loop looks like this in the console:

```
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:~ $
```

Figure 52: Digital One - Python Demo Program

First the digital output 0 is set to True; after a short time, the digital input 0 is read and returns True, a HIGH level is present. Then the digital output 0 is set to False; after a short time, the digital input 0 is read and returns False. A HIGH level is no longer present.

Check out the download section on our website for more Python examples and examples of other programming languages and tools.

9.6.2. PiXtend elO Analog One

This example shows how to use the programming language Python and the pyModbus module to set an analog output on the Analog One device and read it back via an analog input. The analog output 0 must be connected to the analog input 0 via a cable. Refer to the PiXtend elO Hardware Manual for further information on connecting a power supply and wiring an Analog One device.

If working with the programming language Python and using a PiXtend V2 -L-, you have several possibilities to communicate with the PiXtend elO devices. In the SD card Basic image from version 2.1.6.0, we use the full-fledged UART (called PL011, serial0 or ttyAMA0) of the Raspberry Pi. Only this interface allows the parity bit to be used.

If you use an RS485 USB dongle, your serial device will probably be called ttyUSB0. If using the serial interface on the controller of another manufacturer, please refer to its device manual for information. Depending on which interface you use, comment in or out the corresponding lines of code or modify them as you need them.

The following example is based on the assumption that you are working with a PiXtend V2 -L-, with the full-fledged UART (ttyAMA0) and use the Basic image version 2.1.6.0 or later. The Analog One device has the default settings, device address 3, CONFIG switch 1-6 all to OFF. These settings correspond to 19200 baud, 8 data bits, parity "even" and 1 stop bit.

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 AO0 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 AI0: " + str(result.registers[0]))
        time.sleep(1)
        print("Write AO0 to 0")
        result= client.write_register(0x00,0, unit= 0x03)
        time.sleep(1)
        result= client.read_input_registers(0x00, 1, unit= 0x03)
        print("Read AI0: " + str(result.registers[0]))
        time.sleep(1)
    except KeyboardInterrupt:
        client.close()
        time.sleep(0.25)
        client = None
        break
```

The Python program works best with root privileges so that Python has access to the serial port.
Start the program as follows:

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

One cycle of the main program loop looks like this in the console:

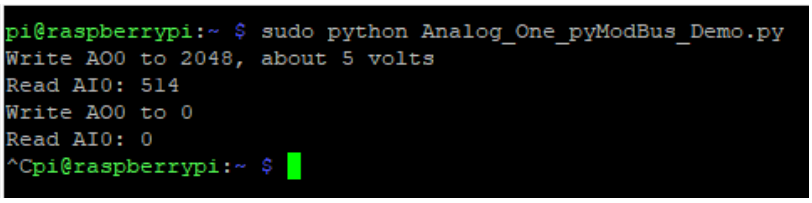


Figure 53: Analog One - Python Demo Program

First the analog output 0 is set to "2048" (about 5 Volt); after a short time, the analog input 0 is read and returns "514", also about 5 Volt. Then the analog output 0 is set to "0"; after a short time, the analog input 0 is read and returns "0". There is therefore no more voltage present.

Check out the download section on our website for more Python examples and examples of other programming languages and tools.

9.7. Example C

9.7.1. PiXtend eIO Digital One

Program examples for the programming language C can be found in the download section on our website.
To use the examples, the library libModbus must be installed. Additionally, wiringPi is required to switch the GPIO18 on a PiXtend V2 -L-.

Perform the following steps to install the libModbus:

- sudo apt update
- sudo apt install libmodbus-dev

The library wiringPi can be found in our download section together with an installation guide. The PiXtend V2 Software Manual, chapter pxdev - Linux Tools and Library contains further information. If you use a Kontron Electronics SD card image, wiringPi is already pre-installed.

The system is prepared, and you can use the Mdbus RTU protocol in C. If you have downloaded the examples from our website, you can compile the Blinky program example with the gcc compiler as follows:

- 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

The program is then started with the following command:

- sudo ./PiXtend_eIO_Digital_One_ModBus_Demo_Blinky

9.7.2. PiXtend eIO Analog One

The preparation steps for the Analog One program examples are the same as for Digital One, see section 9.7.1.

The instruction for compiling the Analog One example program PiXtend_eIO_Analog_One_ModBus_Demo_AO0_to_AI0 differs from that of the Digital One.

Example compile command:

- gcc -I /usr/include/modbus PiXtend_eIO_Analog_One_ModBus_Demo_AO0_to_AI0.c -o PiXtend_eIO_Analog_One_ModBus_Demo_AO0_to_AI0 -L/usr/lib/modbus -lmodbus -lwiringPi

The program is then started with the following command:

- sudo ./PiXtend_eIO_Analog_One_ModBus_Demo_AO0_to_AI0

10. PiXtend eIO protocol

10.1. Introduction

The PiXtend eIO protocol is a simple plain text protocol from Kontron Electronics GmbH designed to be transmitted via a serial bus.

The user can enter simple plain text instructions into a serial program or a terminal program to send a command to the device. The user receives a response to every command.

Various commands are available to read and set the digital and analog inputs and outputs of each device. Each device that is connected to the bus must have a unique device address. The address range is from 0 to 255, whereby only a maximum number of 32 devices are permitted on one RS485 bus. Please take this into account in your planning.

Make sure that the mode switch in DIP block 2 - CONFIG is in the ON position, i.e., at the top.

10.2. Overview

General - for all eIO devices

- Communication procedure
- Protocol structure

PiXtend eIO Digital One

- Overview
- Commands - Basic
- Commands - Advanced
- Examples - Basic
- Examples - Advanced

PiXtend eIO Analog One

- Overview
- Commands - Basic
- Commands - Advanced
- Examples - Basic
- Examples - Advanced

10.3. Communication procedure

The PiXtend elO protocol is based on a master-slave communication similar to that of the Modbus RTU protocol. The master sends a message and the addressed slave responds to this message with the information requested by the master. While the master is waiting for the response of the slave, a timer is running at the same time that monitors the communication with the slave. If the slave does not respond within a certain period of time, the master can resend the message or indicate a communication error to the user. This type of a timeout monitoring is used, for example, with Modbus RTU.

In contrast to the Modbus RTU protocol, where the slave responds as quickly as possible, the PiXtend elO protocol has a 10 ms response delay. For this reason, PiXtend elO devices can be used with a PiXtend V1 device. The switching the send and receive direction of the RS485 transceiver is done by the Raspberry Pi GPIO 22. It is not switched automatically, as with the PiXtend V2 -L- or the USB-to-RS485 dongle.

With serial communication only one bus participant may communicate at a time. If two or more participants try to communicate simultaneously, orderly communication is not possible. Note that only one device can be addressed at a time. When the communication with one device is finished, the next device can communicate. Always use very long waiting times or timeouts to wait for a slave response; we recommend 500 ms or 1 second. If the response from the slave has not arrived after a waiting time of more than 1 second, it can be assumed that something is wrong.

Communication process between master and slave in graphical form:

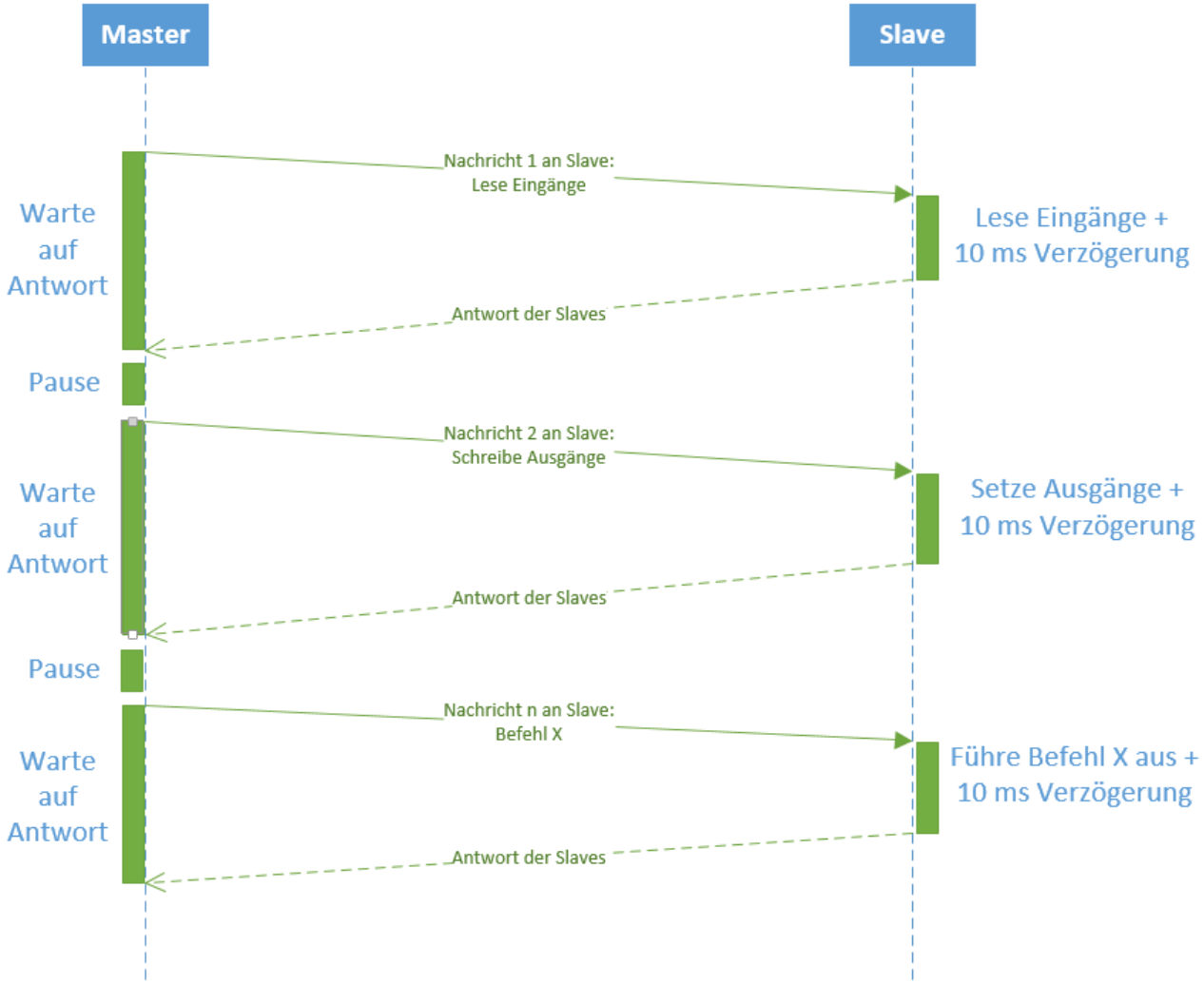


Figure 54: PiXtend eIO protocol - Communication - process overview

Notes about the diagram:

- Master - Waits for a response: up to 1000 ms
- Master - Pause for baud rates below 9600: >= 4 ms
- Master - Pause for baud rates above 9600: >= 2 ms

10.4. Protocol structure

The PiXtend elO protocol is based on simple ASCII characters; it basically has the following structure and always has a fixed length per device.

Protocol structure:

Start character	Address	Separator	Command	Separator	I/O number	Separator	Value	End character

• Start character	1 character, hash, ASCII 35_{Dec}	#
• End character	1 character, asterisk, ASCII 42_{Dec}	*
• Separator	1 character: Colon, ASCII 58_{Dec}	:
• Address	3 characters, numbers only, ASCII 48_{Dec} - 57_{Dec}	000 to 999
• Command	3 characters, numbers only, ASCII 48_{Dec} - 57_{Dec}	000 to 999
• I/O Number	2 characters, numbers only, ASCII 48_{Dec} - 57_{Dec}	00 to 99
• Value length		
◦ Digital One	3 characters, numbers only, ASCII 48_{Dec} - 57_{Dec}	000 to 999
◦ Analog One	4 characters, numbers only, ASCII 48_{Dec} - 57_{Dec}	0000 to 9999
		$_{Dec}$ = decimal

When a device receives a command or a valid message, a response is always sent back. The answer is not always the same, but differs between setting an output, reading in an input or requesting a status.

If an output is set, the device responds with the command or sends back a copy of the received message. This makes it possible to check whether the message for the device in question also arrived correctly or if there was a transmission error. The device makes no changes to the data.

If an input is to be read, the response is to return everything except the VALUE part, which contains the actual value of the digital or analog input.

It is a fixed length protocol. In relation to the respective device, the zeros must always be given for all numbers, provided that the corresponding number is less than 1000 or less than 100 or less than 10. The exact length of the protocol or a message and its specific composition for a device can be found in the chapter of the respective device.

Examples for leading zeros:

- Two-digit values:
  - 1 → 01
  - 12 → 12
- Three-digit values:
  - 1 → 001
  - 10 → 010
  - 100 → 100
- Four-digit values:
  - 1 → 0001
  - 12 → 0012
  - 120 → 0120
  - 1200 → 1200

Note on the following chapters:
The PiXtend elO protocol offers the user many options for setting, changing and switching on and off the various functions of the PiXtend elO devices. It also offers the ability to read and write analog and digital inputs and outputs.

For this reason, the description of the PiXtend elO protocol commands is divided into two main parts, Basic and Advanced.

The Basic section contains all the information needed to read and write the analog and digital inputs and outputs of PiXtend elO devices.

The Advanced section contains all commands and information on how to configure and use the extended functions, e.g., watchdog timer or counter. This section is intended primarily for advanced users who are already familiar with PiXtend elO devices and the basic functions of these devices.

10.5. PiXtend eIO Digital One

With the Digital One device a message always has a **length** of 16 characters (bytes).

The Digital One device has the **device ID:** 221 (Hex: DD)

Number of the digital inputs: 8 (0 - 7)

Number of the digital outputs: 8 (0 - 7)

Example message for a Digital One device:

Description	START	ADR	SEP	COM	SEP	I/O NUM	SEP	VALUE	END
ASCII char	Part Below	012	:	003	:	02	:	001	*
Byte number	1	2-3-4	5	6-7-8	9	10-11	12	13-14-15	16

(START = Start, ADR = Address, SEP = Separator, COM = Command, I/O NUM = I/O Number, VALUE = Value, END = End)

10.5.1. Overview of supported commands - Basic

The following commands can be sent to a Digital One device by message to read its digital inputs or set its digital outputs.

Command	Number	Value range	Comment
Read Digital Input	1	0/1	Reads one digital input
Read Digital Input All	2	0 to 255	Reads all digital inputs. Each bit in the byte corresponds to one input. I/O NUM is ignored with this command.
Write Digital Output	3	0/1	Write/set one digital output.
Write Digital Output All	4	0 to 255	Write/set all digital outputs. Each bit in the byte corresponds to one output. I/O NUM is ignored with this command.
Read Device ID	300	0 to 255	Query device ID, 8-bits value, 1 byte.
Read Error Register	400	0 to 255	Query error register, for a breakdown of the bits in the register see chapter 10.5.2 Overview of the bits in the error register, 16-bit value I/O NUM: 0 = Low byte, 1 = High byte
Set Config - Quit Errors	401	0	Clear all errors in the error register, I/O NUM = 00 and VALUE = 000
Read Firmware Version	500	0 to 999	Version of the device's firmware in XXX format, e.g. 101 = 1.01, 102 = 1.02

Table 46: Digital One: Overview of supported commands – Basic

With the commands 2 (Read Digital Input All) and 4 (Write Digital Output All), a 3-digit number, 000 to 255, can be transmitted. This number has the data type byte (8 bits) on a controller/PC. Every bit in this byte corresponds to an input or an output, from LSB to MSB (Intel Byte Order). Bit 0 = input 0/output 0 and bit 7 = input 7/output 7. The protocol part I/O NUM is ignored.

If an input is to be read or an output is to be set (individually, commands 1 and 2), only the numbers 0 = Off (000) and 1 = On (001) are permitted. Here, two 0 (zeroes) must always be placed in front.

The error register is a 16-bit value which must be read as two separate bytes using the PiXtend elO protocol. With help of the protocol part I/O NUM, the user can select which byte to read. I/O number 0 corresponds to the low byte and I/O number 1 corresponds to the high byte.

10.5.2. Overview of the bits in the error register

The error register contains a copy of the errors that have occurred in the device. The bits in the error register can be used to determine which errors have occurred. During operation the bits remain set until they are cleared with the command Set Config - Quit Errors or a power cycle is executed. Exempted from this rule is the watchdog timer error bit; further information can be found in the explanation text for bit 2.

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
Start value	0	0	0	0	0	0	0	0

Table 47: Bits in the error register - low byte

Bit 0 - CONF ERR → Configuration error

This bit is set if a configuration error is detected in the device. In normal operation, this error will not occur unless the DIP switches are changed during active operation. To remedy this situation, either the DIP switches must be returned to their original position or a power cycle is performed, in which case the new configuration is adopted by the device.

NOTICE

If the DIP switches have been changed and a power cycle has been performed, the device settings will be changed. The change must also take place in the control program; otherwise, the device can no longer be addressed and used.

Bit 1 - FRM ERR → Communication error

If the device detects a communication error, this bit is set. As a rule, there are three fault states. The device may detect a parity, frame, or overrun error.

A parity error usually occurs when the settings on the control device are not correct. The same applies to the frame error; in most cases the baud rate is incorrect and the device does not receive any valid data. The overrun error, on the other hand, means that too much data is sent over the bus too quickly and the device cannot record everything.

Check the settings on the control device to see if they match the configuration of the device. Check if the wiring is correct and that there is no interference on the bus.

Bit 2 - WDT ERR → Watchdog timer has expired

This bit indicates whether or not the watchdog timer had expired before the last power-up. Since the device enters the safe state when the watchdog timer expires, this bit can only be queried after a subsequent power cycle. The bit can be cleared during operation with the command Set Config - Quit Errors or when another power cycle is executed.

Bit 4 - CMD ERR → Incorrect or unsupported command

If a message was sent to the device with an unsupported command, this error bit is set.

By means of this error bit, the control program can determine whether this error has occurred since the last power-up.

Bit 5 - IO NUM ERR → Incorrect or missing I/O number

If a command was sent to the device with a non-existent or unsupported I/O number, this bit is set.

Bit 6 - VAL IN ERR → Non-permitted value detected at input

If an error is detected when reading the inputs, this bit is set.

Bit 7 - VAL OUT ERR → Non-permitted value at output(s)

If a command with a non-permitted value for the digital outputs was sent to the device or the command used only permits a certain value range, this bit is set.

An example of this is: Digital output 3 should be set to 3, only the values 0 and 1 are allowed.

High byte:

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

Table 48: Bits in the error register - high byte

Bit 8 - HLDOCL ERR → Hyper logic output conflict

Hyper logic processors 0 and 1 use digital outputs 0 and 4. The corresponding digital outputs are not available to the user when a hyper logic processor is turned on. If an attempt is made to set the digital outputs nevertheless, this bit is set as error feedback and indicates that there is a conflict between the device setting and the digital outputs used.

Bit 9 - CNTCL ERR → Counter configuration conflict

Digital inputs 1, 3, 5 and 7 cannot be used as a 2-channel counter or switched into the 2-sensor operation. If this is nevertheless attempted, this bit is set and an acknowledgment of the invalid operation is received.

Bit 10 - WDTDO7 ERR → Watchdog timer - digital output 7 conflict

If the watchdog timer has been configured to use digital output 7 as the signal output, then this output will not be available to the user. If the user still tries to set digital output 7, this bit is set to 1 and indicates an operating error.

10.5.3. Example - Reading inputs

The following examples show how the PiXtend elO protocol can be used to query the digital inputs of the Digital One device. First comes the command from the master and then the response of the slave.

- Read digital input 0 from device with address 1:
  - Command: #001:001:00:000*
  - Response: #001:001:00:001*
  - Description: Command 1 is sent to the slave with address 1 to read digital input 0 and send it back to the master as a response. In this example the digital input 0 delivers a 1, thus a HIGH level is present at the input.

- Read digital input 3 from device with address 1:
  - Command: #001:001:03:000*
  - Response: #001:001:03:000*
  - Description: Command 1 is sent to the slave with address 1 to read digital input 3 and send it back to the master as a response. In this example the digital input 3 delivers a 0, thus a HIGH level is present not at the input, but a LOW.

- Read digital input 7 from device with address 1:
  - Command: #001:001:07:000*
  - Response: #001:001:07:001*
  - Description: Command 1 is sent to the slave with address 1 to read digital input 7 and send it back to the master as a response. In this example the digital input 7 delivers a 1, thus a HIGH level is present at the input.

- Read digital input 0-7 (all) from device with address 1:
  - Command: #001:002:00:000*
  - Response: #001:002:00:021*
  - Description: Command 2 is sent to the slave with address 1, as a response the command received back, but the number 21 is received as a value (binary: 0001_0101). Thus the inputs 0, 2 and 4 are set, here a HIGH level is present in each case.

10.5.4. Example - Setting outputs

The following examples show how the PiXtend elO protocol can be used to set the digital outputs of the Digital One device. First comes the command from the master and then the response of the slave.

- Set digital output 0 from device with address 1
  - Command: #001:003:00:001*
  - Response: #001:003:00:001*
  - Description: The command 3 is sent to the slave with address 1 to set digital output 0. In response, we receive a copy of our own command and thus the feedback that the slave has recognized and understood the command.

- Switching off digital output 0 from device with address 1
  - Command: #001:003:00:000*
  - Response: #001:003:00:000*
  - Description: The command 3 is sent to the slave with address 1 to switch off digital output 0. In response, we receive a copy of our own command and thus the feedback that the slave has recognized and understood the command.

- Set digital output 7 from device with address 1
  - Command: #001:003:07:001*
  - Response: #001:003:07:001*
  - Description: The command 3 is sent to the slave with address 1 to set digital output 7. In response, we receive a copy of our own command and thus the feedback that the slave has recognized and understood the command.

- Switching off digital output 7 from device with address 1
  - Command: #001:003:07:000*
  - Response: #001:003:07:000*
  - Description: The command 3 is sent to the slave with address 1 to switch off digital output 7. In response, we receive a copy of our own command and thus the feedback that the slave has recognized and understood the command.

- Switching on digital output 0-7 (all) from device with address 1
  - Command: #001:004:00:255*
  - Response: #001:004:00:255*
  - Description: The command 4 is sent to the slave with address 1 to set all digital outputs. In response, we receive a copy of our own command and thus the feedback that the slave has recognized and understood the command. The protocol part I/O number is ignored, we set it to 00.

- Switching off digital output 0-7 (all) from device with address 1
  - Command: #001:004:00:000*
  - Response: #001:004:00:000*
  - Description: The command 4 is sent to the slave with address 1 to switch off all digital outputs. In response, we receive a copy of our own command and thus the feedback that the slave has recognized and understood the command. The protocol part I/O number is ignored, we set it to 00.

10.5.5. Example - Reading the error register

The following examples show how to read the error register from the Digital One device using the PiXtend elO protocol. Please note that the error register is a 16-bit value that must be retrieved via a separate low byte and high byte. The byte is selected via the protocol part I/O NUM. The low byte corresponds to 00 and the high byte to 01.

- Read error register from device with address 1 - low byte
  - Command: #001:400:00:000*
  - Response: #001:400:00:003*
  - Description: The command 400 is sent to the slave with address 1 to send the error register to the master, the low byte. Here we got back the number 3 (binary: 000_0011), so there is a configuration error and the device has detected a frame error during communication.

- Read error register from device with address 1 - high byte
  - Command: #001:400:01:000*
  - Response: #001:400:01:000*
  - Description: The command 400 is sent to the slave with address 1 to send the error register to the master, the high byte. Here we got back the number 0, which means that no error has occurred since the last power-up.

10.6. PiXtend elO Analog One

With the Analog One device a message always has a **length** of 17 characters (bytes).

The Analog One device has the **device ID:** 170 (Hex: AA)

Number of the analog inputs: 8 (0 - 7), 4 voltage + 4 current

Value range of the analog inputs: 0 - 1023

Number of the analog outputs: 6 (0 - 5), 4 voltage + 4 current

Value range of the analog outputs: 0 - 4095

Example message for an Analog One device:

Description	START	ADR	SEP	COM	SEP	I/O NUM	SEP	VALUE	END
ASCII char	Part Below	012	:	003	:	02	:	0001	*
Byte number	1	2-3-4	5	6-7-8	9	10-11	12	13-14-15-16	17

(START = Start, ADR = Address, SEP = Separator, COM = Command, I/O NUM = I/O Number, VALUE = Value, END = End)

10.6.1. Overview of supported commands - Basic

The following commands can be sent to an Analog One device by message to read its analog inputs or set its analog outputs.

Command	Number	Value range	Comment
Read Analog Input	5	0 to 1023	Read one analog input.
Write Analog Output	6	0 to 4095	Write/set one analog output.
Read Device ID	300	0 to 255	Query Device ID, 8-bits value, 1 byte.
Read Error Register	400	0 to 255	Query error register, for a breakdown of the bits in the register see section 10.6.2 Overview of the bits in the error register, 16-bit value, I/O NUM: 0 = low byte, no high byte present.
Set Config - Quit Errors	401	0	Clear all errors in the error register, □ I/O NUM = 00 and VALUE = 0000
Read Firmware Version	500	0 to 999	Version of the device's firmware in XXX format, e.g. 101 = 1.01, 102 = 1.02

Table 49: Analog One: Overview of supported commands - Basic

The error register is a 16-bit value; only the lower 8 bits are used. With the PiXtend elO protocol only the first byte is read. With help of the protocol part I/O NUM, the user can select which byte to read. The I/O number 0 corresponds to the low byte; the high byte is not required.

10.6.2. Overview of the bits in the error register

The error register contains a copy of the errors that have occurred in the device. The bits in the error register can be used to determine which errors have occurred. During operation the bits remain set until they are cleared with the command Set Config - Quit Errors or a power cycle is executed. Exempted from this rule is the watchdog timer error bit; further information can be found in the explanation text for bit 2.

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
Start value	0	0	0	0	0	0	0	0

Table 50: Bits in the error register - low byte

Bit 0 - CONF ERR → Configuration error

This bit is set if a configuration error is detected in the device. In normal operation, this error will not occur unless the DIP switches are changed during active operation. To remedy this situation, either the DIP switches must be returned to their original position or a power cycle is performed, in which case the new configuration is adopted.

NOTICE

If the DIP switches have been changed and a power cycle has been performed, the device settings will be changed. The change must also take place in the control program; otherwise, the device can no longer be addressed and used

Bit 1 - FRM ERR → Communication error

This bit is set if the device detects a communication error. As a rule, there are three fault states. The device may detect a parity, frame, or overrun error.

A parity error usually occurs when the settings on the control device are not correct. The same applies to the frame error; in most cases the baud rate is incorrect and the device does not receive any valid data. The overrun error means that too much data is sent over the bus too quickly and the device cannot record everything.

In such cases, check the settings on the control box to see if they match the configuration of the device, if the wiring is correct, and that there are no faults on the bus.

Bit 2 - WDT ERR → Watchdog timer has expired

This bit indicates whether or not the watchdog timer had expired before the last power-up. Since the device enters the safe state when the watchdog timer expires, this bit can only be queried after a subsequent power cycle. The bit can be cleared during operation with the command Set Config - Quit Errors or when another power cycle is executed.

Bit 4 - CMD ERR → Incorrect or unsupported command

If a message was sent to the device with an unsupported command, this error bit is set.

By means of this error bit, the control program can determine whether this error has occurred since the last power-up.

Bit 5 - IO NUM ERR → Incorrect or missing I/O number

If a command was sent to the device with a non-existent or unsupported I/O number, this bit is set.

Bit 6 - VAL IN ERR → Non-permitted value detected at input

If an error is detected when reading the inputs, this bit is set.

Bit 7 - VAL OUT ERR → Non-permitted value at output(s)

If a command with a non-permitted value for the analog outputs was sent to the device or the command used only permits a certain value range, this bit is set.

An example of this is: Analog output 3 is to be set to 5900, only values between 0 and 4095 are permitted.

High byte:

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

Table 51: Bits in the error register - high byte

With the Analog One device, there are no error bits in the upper byte (high byte) of the error register. This byte can be ignored; only the bits 0 to 7 in the lower byte have a value.

10.6.3. Example - Reading inputs

The following examples show how the PiXtend elO protocol can be used to query the analog inputs of the Analog One device. Please note that the analog inputs have a value range from 0 to 1023. This raw value must be converted to the corresponding voltage or current value.

- Read analog input 0 (voltage) from device with address 3
  - Command: #003:005:00:0000*
  - Response: #003:005:00:0511*
  - Description: Command 5 is sent to the slave with address 3 to read analog input 0 and send it back to the master as a response. In this example, analog input 0 returns the value 511, which is approximately 5 volts in the 10 volt measuring range of the device.

- Read analog input 4 (current) from device with address 3
  - Command: #003:005:04:0000*
  - Response: #003:005:04:0200*
  - Description: Command 5 is sent to the slave with address 3 to read analog input 4 and send it back to the master as a response. In this example, analog input 4 returns the value 200, ca. 4mA.

For the conversion of raw values into current and voltage values, see chapter 6.17 Current and voltage conversion factor.

10.6.4. Example - Setting outputs

The following examples show how the PiXtend elO protocol can be used to set the analog outputs of the Analog One device. Please note that the analog outputs have a value range from 0 to 4095. To obtain the raw value for the device from a voltage or current value, a conversion must take place

- Set analog output 0 (voltage) from device with address 3
  - Command: #003:006:00:1023*
  - Response: #003:006:00:1023*
  - Description: Command 6 is sent to the slave with address 3 to set the analog output 0 to the value 1023, which is approximately 2.5 volts. The message sent is received as a reply or confirmation.

- Set analog output 4 (current) from device with address 3
  - Command: #003:006:04:0820*
  - Response: #003:006:04:0820*
  - Description: Command 6 is sent to the slave with address 3 to set the analog output 4 to the value 820, which is approximately 4 mA. The message sent is received as a reply or confirmation.

For the conversion of raw values into current and voltage values, see chapter 6.17 Current and voltage conversion factor.

10.6.5. Example - Reading the error register

The following examples show how to read the error register from the Analog One device using the PiXtend eIO protocol. Please note that the error register is a 16-bit value. The value is retrieved via a separate byte, low byte and high byte. With the Analog One device, the lower 8 bits are used. The byte is selected via the protocol part I/O NUM. The low byte corresponds to 00 and we do not need the high byte because it is always zero.

- Read error register from device with address 3 - low byte
  - Command: #003:400:00:0000*
  - Response: #003:400:00:0032*
  - Description: The command 400 is sent to the slave with address 3 to return the low byte of the error register to the master. We receive the number 32 (binary: 0010_0000), i.e., the device has received a command with an invalid I/O number at least once since the last power-up, error acknowledgment or power cycle.

10.7. PiXtend eIO Protocol - Advanced

Besides the simple commands for reading and controlling the digital and analog inputs and outputs of the Digital One and Analog One devices, there are additional commands for each device. For example, to set and start the watchdog timer or to count pulses on the module. Setting and activating the hyper logic is also described.

10.7.1. PiXtend eIO Digital One

In addition to digital inputs and outputs, the Digital One device offers other functions that can be configured and activated via additional PiXtend eIO protocol commands.

10.7.1.1 Overview of supported commands

The following commands can be sent to a Digital One device by message to set its additional functions and switch them on or off.

Command	Number	Range	Comment
Read Status	200	0 to 255	Request status from the device. Breakdown of the bits see section 10.7.1.2 Overview of the bits in the status register. The byte to be queried is selected via the I/O number: 0 = Low byte, 1 = High byte.
Set Config - HL 0	402	0 to 1	Activate hyper logic processor 0. 0 = Off, 1 = On, see also the section 10.7.1.5 Hyper logic configuration.
Set Config - HL0 Config	403	0 to 14	Hyper logic processor 0 link of DI0, DI1, DI2 and DI3 with AND and OR sets links for the fast setting of DO0. See section 10.7.1.5 Hyper logic configuration
Set Config - HL 1	404	0 to 1	Activate hyper logic processor 1. 0 = Off, 1 = On, see section 10.7.1.5 Hyper logic configuration
Set Config - HL 1 Config	405	0 to 14	Hyper logic processor 1 link of DI4, DI5, DI6 and DI7 with AND and OR define links for the fast setting of DO4. See section 10.7.1.5 Hyper logic configuration.
Set Config - HL	406	0 to 255	Defines how inputs DI0 - DI7 are considered. Standard: Positive (0) or inverted/negative (1). There are 8 bits in the low byte, one bit for each input. There is one option per bit: 0 = Off, 1 = On (negated), the smoothing/debouncing of the inputs can be defined in the upper byte. See section 10.7.1.5 Hyper logic configuration
Set Counter Config	407	0 to 3	Switch counter 0-7 on or off. The counter can be selected via the I/O number. The options are 0 = Off, 1 = On (RE), 2 = On (FE), 3 = On (RE + FE)[5], counter number corresponds to DI number 0 to 7. See section 10.7.1.4 Counter configuration.
Set Counter Type	408	0 - 3	Counter type, the options are: 0 = counter up (Up Counter - standard), 1 = counter down (Down Counter), 2 = counter up/down, 2-sensor operation (2-channel counter). See section 10.7.1.4 Counter configuration.
Set Counter Reset	409	0	Reset counter, command resets the counter with the specified I/O number 0 - 7. All counters are reset with the special number 8. See section 10.7.1.4 Counter configuration.
Set Counter Pre-Set	410	0 to 255	Set counter pre-set value (16-bit value), must be transmitted using two bytes. This value is preloaded to a counter when a counter is reset. The standard value is 0. Byte selection via I/O number: 0 = Low byte, 1 = High byte. See section 10.7.1.4 Counter configuration.
Read Counter Low Byte	450	0 to 255	Read counter low byte. The counter units are 16-bit counters; each counter must be read by two separate bytes. The counter selection is done via the I/O number 0 - 7. If the low byte is queried for a counter, the high byte is frozen for this purpose and must be queried immediately as the next command. If a low byte is queried again, the previous high byte is overwritten. If another counter is polled before the high byte of the previous request is retrieved, the high byte is overwritten. See section 10.7.1.4 Counter configuration.
Read Counter High Byte	451	0 to 255	Read counter high byte. The counter units are 16-bit counters;

[5] RE = rising edge, FE = falling edge

Command	Number	Range	Comment
			each counter must be read by two separate bytes. The counter selection is done via I/O number 0 - 7. Before querying the high byte, the low byte must be queried and then the high byte. Never query the high byte first! See section 10.7.1.4 Counter configuration Counter configuration.
Set Watchdog-Timer Time	600	0 to 9	Set timeout time for the watchdog timer. See section 10.7.1.3 Watchdog timer configuration. The default is 4 seconds timeout time.
Set Watchdog-Timer Behavior	601	0 to 1	Change watchdog timer behavior when the watchdog timer expires. There are two possibilities: 0 = All outputs maintain their current state (default setting), 1 = All outputs are actively switched off. Please also observe our notes on the safe state in chapter 6.2 Definition "Safe State".
Set Watchdog-Timer Output	602	0 to 1	The watchdog timer sets digital output 7 when it expires. The options are 0 = Off, i.e., the user can control the DO7 or 1 = On, the watchdog timer takes over control of the DO7, the user has no more influence. See information in chapter 10.7.1.3 Watchdog timer configuration
Set Watchdog-Timer Out Behavior	603	0 to 1	Change behavior of the watchdog timer digital output 7. By default, the watchdog timer sets this digital output 7 to HIGH when it expires. This behavior can be reversed; in normal operation the digital relay output 7 is always on (HIGH) and when the watchdog timer expires it switches the digital output 7 off, goes to LOW. 0 = HIGH active, 1 = LOW active
Set Watchdog-Timer Enable	604	0 to 1	Switch watchdog timer on/off; value 0 switches the watchdog off and value 1 switches it on.

Table 52: Digital One: Overview of supported commands - Advanced

10.7.1.2 Overview of the bits in the status register

The status register shows which function is active in the device and which is not. A cyclic request is recommended; it can be checked whether a function has been activated or not.

The status register is 16 bits in size and must therefore be read out via two separate bytes (low byte and high byte).

Low byte:

Bit	7	6	5	4	3	2	1	0
Name	CNT2	CNT1	CNT0	HL1	HL0	WDT ON	WDT OBE	WDT OUT
Start value	0	0	0	0	0	0	0	0

Table 53: Bits in the status register - low byte

Bit 0 - WDT OUT → Watchdog time output active (digital output 7)

Off: When the watchdog timer expires, the digital output 7 is not set but is treated as a normal digital output.

On: The digital output 7 is used by the watchdog timer and is no longer available to the user. In the default setting, the watchdog timer sets the digital output 7 to HIGH when it expires. This can be changed by the user. See bit 1 - WDT OBE. See information in section 10.7.1.3 Watchdog timer configuration

Bit 1 - WDT OBE → Watchdog time output behavior

Off: Digital output 7 is switched from LOW → HIGH when the watchdog timer expires. The normal state for this setting is LOW. This setting corresponds to the standard behavior (default setting).
This setting is useful if, for example, a lamp or siren is to be switched. If the watchdog timer expires, a fault in the system can be detected quickly.

On: Digital output 7 is switched from HIGH → LOW when the watchdog timer expires. The normal state for the digital output is HIGH. This setting is useful if a relay must be energized during operation for the system to be operational. When the watchdog timer expires, digital output 7 is switched to LOW and the relay turns off. See information in section 10.7.1.3 Watchdog timer configuration

Bit 2 - WDT ON → Watchdog time active

Off: The watchdog timer is off.

On: The watchdog timer is active and, depending on the setting, is reset or triggered after the specified time has expired. See information in section 10.7.1.3 Watchdog timer configuration.

Bit 3 - HL0 → Hyper logic processor 0

Off: There is no hyper logic processing for hyper logic processor 0.

On: Hyper logic processing is enabled and the state of digital output 0 can be controlled by up to four digital inputs. Various combinations of digital inputs 0 to 3 are available; see further information in section 10.7.1.5 Hyper logic configuration.

Bit 4 - HL1 → Hyper logic processor 1

Off: There is no hyper logic processing for hyper logic processor 1.

On: Hyper logic processing is enabled and the state of digital output 4 can be controlled by up to four digital inputs. Various combinations of digital inputs 4 to 7 are available; see further information in section 10.7.1.5 Hyper logic configuration.

Bit 5 - CNT0 → Counter 0 active

Off: Counter 0 is not active and there is no counting.

On: Counter 0 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

Bit 6 - CNT1 → Counter 1 active

Off: Counter 1 is not active and there is no counting.

On: Counter 1 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

Bit 7 - CNT2 → Counter 2 active

Off: Counter 2 is not active and there is no counting.

On: Counter 2 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

High byte:

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

Table 54: Bits in the status register - high byte

Bit 8 - CNT3 → Counter 3 active

Off: Counter 3 is not active and there is no counting.

On: Counter 3 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

Bit 9 - CNT4 → Counter 4 active

Off:	Counter 4 is not active and there is no counting.
On:	Counter 4 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

Bit 10 - CNT5 → Counter 5 active

Off:	Counter 5 is not active and there is no counting.
On:	Counter 5 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

Bit 11 - CNT6 → Counter 6 active

Off:	Counter 6 is not active and there is no counting.
On:	Counter 6 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

Bit 12 - CNT7 → Counter 7 active

Off:	Counter 7 is not active and there is no counting.
On:	Counter 7 is activated. The 16-bit value of the counter can be retrieved with the commands 450 (Read Counter Low Byte) and 451 (Read Counter High Byte).

10.7.1.3 Watchdog timer configuration

The watchdog timer has different settings on the device, most importantly the watchdog timer timeout time. This time determines how long the watchdog timer waits until it is triggered. The watchdog timer for the PiXtend elO protocol is always reset when there is bus activity; there is no other setting here. Please note that the watchdog requires cyclic communication on the bus after its activation.

To use the watchdog, first set the timeout time if you do not want to use the default setting of 4 seconds. The following table helps you to select a timeout time.

Number	Timeout	Comment
000	16 ms	Not recommended or very frequent communication necessary
001	32 ms	Not recommended or very frequent communication necessary
002	64 ms	Not recommended or very frequent communication necessary
003	0.125 s	
004	0.250 s	
005	0.5 s	
006	1.0 s	
007	2.0 s	
008	4.0 s	Default setting
009	8.0 s	Longest time-out setting, good with little bus activity

Table 55: Digital One: Overview watchdog timer timeout

The watchdog timer timeout time can be changed with the command 600 (Set Watchdog-Timer Time).

Description	Command to slave	Reply from slave
Command 600, time-out 1 sec.	#001:600:00:006*	#001:600:00:006*
Command 600, time-out 8 sec.	#001:600:00:009*	#001:600:00:009*

Table 56: Setting watchdog timer timeout - examples

After the timeout has been set, the watchdog can be activated.

Description	Command to slave	Reply from slave
Command 604, watchdog on	#001:604:00:001*	#001:604:00:001*
Command 604, watchdog off	#001:604:00:000*	#001:604:00:000*

Table 57: Watchdog timer on/off - examples

Further functions of the watchdog timer are to switch off all outputs when it expires or to switch on a digital output.

Description	Command to slave	Reply from slave
Command 601, all outputs off	#001:601:00:001*	#001:601:00:001*
Command 601, all outputs remain unchanged	#001:601:00:000*	#001:601:00:000*
Command 602, watchdog triggers DO7	#001:602:00:001*	#001:602:00:001*
Command 603, watchdog inverts DO7	#001:603:00:001*	#001:603:00:001*
Command 603, watchdog DO7 normal	#001:603:00:000*	#001:603:00:000*

Table 58: Setting watchdog timer functions - examples

For further information about the watchdog timer, see chapter 6.11 PiXtend elO Watchdog timer, section Basic knowledge.

10.7.1.4 Counter configuration

By selecting the edge setting, which edge of a signal the counter should consider when counting, the counter is also switched on. The following pages describe how to change the counter type. The counter is selected via the I/O number part of the protocol.

Edge Setting Selection

Number	Edge setting
0	Off, counter not active
1	Rising edge, 0 → 1
2	Falling edge, 1 → 0
3	Rising + falling edge (RE + FE), 0 → 1 + 1 → 0

Table 59: Edge setting - breakdown

Switch the counter on and off:

Description	Command to slave	Reply from slave
Command 407, switch on counter 0, count rising edge	#001:407:00:001*	#001:407:00:001*
Command 407, switch on counter 1, count falling edge	#001:407:01:002*	#001:407:01:002*
Command 407, switch on counter 2, count falling edge	#001:407:02:003*	#001:407:02:003*
Command 407, switch off counter 2	#001:407:02:000*	#001:407:02:000*
Command 407, switch off counter 1	#001:407:01:000*	#001:407:01:000*
Command 407, switch off counter 0	#001:407:00:000*	#001:407:00:000*

Table 60: Switching counters on/off with different settings - examples

Counter type selection
The counter type defines how the counter should count. There is the setting that the counter counts up (default), counts down and counts two channels. The following table breaks down this information in more detail.

Number	Counter type
0	Counter counts up, up counter (default setting)
1	Counter counts down, down counter
2	Counter counts up/down, 2-sensor operation (2-channel counter)
3	reserved

Table 61: Counter type – selection

The inputs used for 2-sensor operation are always the inputs of the respective counters 0, 2, 4 and 6 and the input which is directly adjacent, i.e., 1, 3, 5 and 7.

For more information about the counters of the Digital One, see chapter 6.12 Counter function in PiXtend elO in section Basic knowledge.

Set counter type:

Description	Command to slave	Reply from slave
Command 408, counter 0 counts up	#001:408:00:000*	#001:408:00:000*
Command 408, counter 1 counts down	#001:408:01:001*	#001:408:01:001*
Command 408, counter 4 as 2-channel counter	#001:408:04:002*	#001:408:04:002*

Table 62: Set counter type - examples

Change counter pre-set value

Command 410 (Set Counter Pre-Set) can be used to change the value that is loaded into the counter memory when a counter is reset. This value is usually 0. The counter memory for each counter has a size of 16 bits, so the counter pre-set value is 16 bits. The transmission must take place via two separate bytes. If the counter pre-set value was changed, this value does not appear immediately with every counter. When a counter is reset, it is loaded into the counter memory or when all counters are reset together. In both cases, this value is loaded into the counter memory by every counter.

The selection between low byte and high byte is done via the I/O number part of the protocol. A 0 means the value is written to the low byte and a 1 means the value is written to the high byte.

Counter pre-set value - The pre-set value is set to 4095:

Description	Command to slave	Reply from slave
Command 410, transmit low-byte	#001:410:00:255*	#001:410:00:255*
Command 410, transmit high-byte	#001:410:01:015*	#001:410:01:015*

Table 63: Counter pre-set value - example 1

Counter pre-set value - The pre-set value is set to 0:

Description	Command to slave	Reply from slave
Command 410, transmit low-byte	#001:410:00:000*	#001:410:00:000*
Command 410, transmit high-byte	#001:410:01:000*	#001:410:01:000*

Table 64: Counter pre-set value - example 2

Counter reset

With the command 409 (Set Counter Reset), each of the 8 counters can be reset individually or all together. When a counter is reset, the counter pre-set value is always loaded into the counter memory. The default value is 0. The following table shows the possible numbers for resetting the various counter units. The selection is made via the I/O number part of the protocol, the value part is ignored.

Number	Counter
0	Reset counter 0
1	Reset counter 1
2	Reset counter 2
3	Reset counter 3
4	Reset counter 4
5	Reset counter 5
6	Reset counter 6
7	Reset counter 7
8	Reset all counters

Table 65: Counter reset

NOTICE

When resetting a counter, the value of the internal memory, the counter pre-set value is loaded into the counter memory. Without user intervention, this value is 0.

With this procedure, any value between 0 and 65535 can be pre-set into a counter. A counter does not have to start counting at 0; it may take another value after a reset. With a down counter this behavior is advantageous if you want to count 100 units. The counter can be preset to 100; when it reaches 0, a reset resets the counter to 100.

Examples for resetting the counters:

Description	Command to slave	Reply from slave
Command 409, reset counter 0	#001:409:00:000*	#001:409:00:000*
Command 409, reset counter 1	#001:409:01:000*	#001:409:01:000*
Command 409, reset counter 2	#001:409:02:000*	#001:409:02:000*
Command 409, reset counter 7	#001:409:07:000*	#001:409:07:000*
Command 409, reset all counters	#001:409:08:000*	#001:409:08:000*

Table 66: Counter reset examples

10.7.1.5 Hyper logic configuration

The hyper logic processors 0 and 1 offer the user the possibility to react quickly to up to 4 different signals each without software intervention to set or switch off an output.

The hyper logic processors can be switched on and configured with the commands 402 (Set Config - HL0), 403 (Set Config - HL0 Config), 404 (Set Config - HL1), 405 (Set Config HL1 Config) and 406 (Set Config - HL). We recommend that you first determine and configure the link for each hyper logic processor and then switch on the respective processor.

Hyper logic processor 0 link

The following table shows which links of digital inputs 0 to 3 are supported by hyper logic processor 0. The binding strength of the logical links is symbolized by brackets. A double ampersand (&&) represents an AND operation, and an OR operation is expressed by two vertical lines (||). The specification of the column number can be transferred directly with the command 403. The digital output from the hyper logic processor 0 is always the digital output 0. The respective link number is transmitted with the value part of protocol; the I/O number part is not important and should always be 00.

Number	Link
0	DI0
1	DI0 && DI1
2	DI0 || DI1
3	DI0 && DI1 && DI2
4	(DI0 && DI1) || DI2
5	(DI0 || DI1) && DI2
6	DI0 || DI1 || DI2
7	DI0 && DI1 && DI2 && DI3
8	(DI0 && DI1 && DI2) || DI3
9	DI0 && (DI1 || DI2) && DI3
10	DI0 || DI1 && DI2 && DI3
11	(DI0 && DI1) || DI2 || DI3
12	(DI0 || DI1) && (DI2 || DI3)
13	(DI0 || DI1 || DI2) && DI3
14	DI0 || DI1 || DI2 || DI3

Table 67: Hyper logic processor 0 - link overview

Hyper logic processor 1 link

The following table shows which links of digital inputs 4 to 7 are supported by hyper logic processor 1. The binding strength of the logical links is symbolized by brackets. A double ampersand (&&) represents an AND operation, and an OR operation is expressed by two vertical lines (||). The specification of the column number can be transferred directly with the command 405. The digital output from the hyper logic processor 1 is always the digital output 4. The respective link number is transmitted with the value part of protocol; the I/O number part is not important and should always be 00.

Number	Link
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

Table 68: Hyper logic processor 1 - link overview

Set and switch on hyper logic processor 0 to link 3, in addition the digital input 0 is considered inverted. Refer to the chapter 10.7.1.6 Overview the bits in the hyper logic config register for more information.

Description	Command to slave	Reply from slave
Command 403, Set link	#001:403:00:003*	#001:403:00:003*
Command 406, Invert DI0 for HL0	#001:406:00:001*	#001:406:00:001*
Command 402, Hyper logic proc. 0 on	#001:402:00:001*	#001:402:00:001*

Table 69: Hyper logic processor 0 - example 1

Set hyper logic processor 1 to link 14 and switch it on.

Description	Command to slave	Reply from slave
Command 405, Set link	#001:405:00:014*	#001:405:00:014*
Command 404, Hyper logic proc. 1 on	#001:404:00:001*	#001:404:00:001*

Table 70: Hyper logic processor 1 - example 2

Switch off hyper logic processor 0 and 1.

Description	Command to slave	Reply from slave
Command 402, Hyper logic proc. 0 off	#001:402:00:000*	#001:402:00:000*
Command 404, Hyper logic proc. 1 off	#001:404:00:000*	#001:404:00:000*

Table 71: Hyper logic processor 0 / 1 - example 3

10.7.1.6 Overview the bits in the hyper logic config register

The hyper logic config register allows you to specify for each digital input whether it should be considered positive (default) or negative (inverted). The two bytes must be transferred separately. The byte is selected via the I/O number part of the protocol.

This setting is helpful when using a sensor that is LOW active and switches from HIGH to LOW when detecting an object or part.

Low byte:

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

Table 72: Bits in the hyper logic config register - low byte

One bit (HLINx, x = 0 to 7) is available for each of the eight digital inputs.

Bit 7 .. 0 - HLIN7 **..** 0 → Hyper logic processor 0 & 1 - input inversion

Off: All digital inputs are considered positive.

On: The signal at the selected input is inverted or evaluated negatively.
With this setting, it is assumed that a connected sensor or switch always delivers a HIGH level in its idle state. See chapter 10.7.1.5 Hyper logic configuration

High byte:

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

Table 73: Bits in the hyper logic config register - high byte

The hyper logic processors 0 and 1 typically operate at a rate faster than the digital inputs can respond to achieve the fastest possible reaction. For some applications, however, it is useful to debounce the input signals for a more secure and stable output for each hyper logic processor.

Bit 8 - HLSM0 → Hyper logic processor 0 - Signal smoothing (debounce)

Off: No smoothing (debounce) is performed.

On: The input signals for the hyper logic processor 0 are debounced. For digital inputs 0 to 3, if they occur in the selected hyper logic link, each signal must be present for longer than 2 ms before it is considered valid. It can then be processed by the hyper logic.

Bit 9 - HLSM1 → Hyper logic processor 1 - Signal smoothing (debounce)

Off: No smoothing (debounce) is performed.

On: The input signals for the hyper logic processor 1 are debounced. For digital inputs 4 to 7, if they occur in the selected hyper logic link, each signal must be present for longer than 2 ms before it is considered valid. It can then be processed by the hyper logic.

10.7.2. Example - Use counter

Below are several examples of how to work with the counter units on a Digital One device.

Example for counter 0, count falling edges, counter 0 is operated as up counter

Description	Command to slave	Reply from slave
Command 408, counter 0 an up counter	#001:408:00:000*	#001:408:00:000*
Command 410, pre-set value to 0, low byte	#001:410:00:000*	#001:410:00:000*
Command 410, pre-set value to 0, high byte	#001:410:01:000*	#001:410:01:000*
Command 409, reset counter 0	#001:409:00:000*	#001:409:00:000*
Command 407, switch on counter 0 with RE	#001:407:00:002*	#001:407:00:002*
Command 450, read counter 0, low byte	#001:450:00:000*	#001:450:00:085*
Command 451, read counter 0, low byte	#001:451:00:000*	#001:451:00:037*

Table 74: Example: Use counter - counter 0

In this example the counter 0 counts up and is reset before power-on. For this reason, the counter pre-set value was set to 0.

After switching on and waiting a short period, counter 0 was queried, it had reached a count value of 9557. Counter 0 will continue to count until it reaches 65535, then it starts at 0 again.

For the handling or conversion of 16-bit values, see chapter 6.18 Combining two bytes to a 16-bit value and 6.19 Splitting a bit value into two bytes in the section Basic knowledge.

Example for counter 3, counting rising edges, counter 3 is operated as down counter and pre-set with the value 10000.

Description	Command to slave	Reply from slave
Command 408, Counter 3 as down counter	#001:408:03:001*	#001:408:03:001*
Command 410, pre-set value to 10000, LB	#001:410:00:016*	#001:410:00:016*
Command 410, pre-set value to 10000, HB	#001:410:01:039*	#001:410:01:039*
Command 409, reset counter 3	#001:409:03:000*	#001:409:03:000*
Command 407, switch on counter 3 with RE	#001:407:03:001*	#001:407:03:001*
Command 450, read counter 3, low byte	#001:450:03:000*	#001:450:03:136*
Command 451, read counter 3, low byte	#001:451:03:000*	#001:451:03:019*

Table 75: Example: Use counter - counter 3

Counter 3 was set as down counter and set to the value 10000. With every rising edge of a HIGH level at DI3 the counter is decremented. After some time, we read the counter and get back the value 5000 after merging the two bytes, 136 and 19, to a 16-bit value.

For the handling or conversion of 16-bit values, see chapter 6.18 Combining two bytes to a 16-bit value and 6.19 Splitting a bit value into two bytes in the section Basic knowledge.

Example for counter 6, 2-sensor operation (2-channel counter):

Description	Command to slave	Reply from slave
Command 408, counter 6 as 2-channel counter	#001:408:06:002*	#001:408:06:002*
Command 410, pre-set value to 0, low byte	#001:410:00:000*	#001:410:00:000*
Command 410, pre-set value to 0, high byte	#001:410:01:000*	#001:410:01:000*
Command 409, reset counter 6	#001:409:06:000*	#001:409:06:000*
Command 407, switch on counter 6 with RE	#001:407:06:001*	#001:407:06:001*
Command 450, read counter 6, low byte	#001:450:06:000*	#001:450:06:010*
Command 451, read counter 6, low byte	#001:451:06:000*	#001:451:06:000*

Table 76: Example: Use counter- counter 6 – 2-channel counter

Counter 6 was configured as a 2-channel counter and then reset to 0. One cable each from a sensor was connected at the digital inputs 6 and 7. These sensors detect a signal when an encoder on a disc moves past them. This design allows us to detect the direction of rotation and detect the number of revolutions of the disc.

The counter itself is switched on with the command 407. At the same time, we give the counter the command that it only counts rising edges. We start the motor to which the disc is attached. The disc only turns in one direction.

After a short time, we query the counter and get the value 10, which means the disk with the encoder completed 10 turns.

10.7.3. Example - Use hyper logic processor

The following examples for the hyper logic processors help with their use and configuration and show the structure of the messages.

Set hyper logic processor 0 to link 3 and switch it on, in addition the digital input 0 (DI0) is considered inverted.

Description	Command to slave	Reply from slave
Command 403, Set link	#001:403:00:003*	#001:403:00:003*
Command 406, Invert DI0 for HL0	#001:406:00:001*	#001:406:00:001*
Command 402, Hyper logic proc. 0 on	#001:402:00:001*	#001:402:00:001*

Table 77: Example: Use hyper logic processor 0

Set hyper logic processor 1 to link 14 and switch it on

Description	Command to slave	Reply from slave
Command 405, Set link	#001:405:00:014*	#001:405:00:014*
Command 404, Hyper logic proc. 1 on	#001:404:00:001*	#001:402:00:001*

Table 78: Example: Use hyper logic processor 1

Switch off hyper logic processor 0 and 1

Description	Command to slave	Reply from slave
Command 402, Hyper logic proc. 0 off	#001:402:00:000*	#001:402:00:000*
Command 404, Hyper logic proc. 1 off	#001:404:00:000*	#001:402:00:000*

Table 79: Example: Switch off hyper logic processor 0 and 1

10.7.4. Example - Activate watchdog timer

The following examples clarify the use and configuration of the watchdog timer on PiXtend elO Digital One devices.

The watchdog timer timeout time can be changed with the command 600.

Description	Command to slave	Reply from slave
Command 600, time-out 1 sec.	#001:600:00:006*	#001:600:00:006*
Command 600, time-out 8 sec.	#001:600:00:009*	#001:600:00:009*

Table 80: Example: Setting watchdog timer timeout

If the timeout is set, the watchdog can be activated.

Description	Command to slave	Reply from slave
Command 604, watchdog on	#001:604:00:001*	#001:604:00:001*
Command 604, watchdog off	#001:604:00:000*	#001:604:00:000*

Table 81: Example: Switch watchdog timer on and off

Other functions of the watchdog timer include switching off all outputs when it expires or switching on digital output 7.

Description	Command to slave	Reply from slave
Command 601, all outputs off	#001:601:00:001*	#001:601:00:001*
Command 601, all outputs remain on	#001:601:00:000*	#001:601:00:000*
Command 602, watchdog triggers DO7	#001:602:00:001*	#001:602:00:001*
Command 603, watchdog inverts DO7	#001:603:00:001*	#001:603:00:001*
Command 603, watchdog DO7 normal	#001:603:00:000*	#001:603:00:000*

Table 82: Example: Setting watchdog timer functions

10.7.5. Example - Read status

The following examples show how to use the PiXtend elO protocol to read the current status from the Digital One device. Please note that the status register is a 16-bit value that must be retrieved via a separate low byte and high byte. The byte is selected via the protocol part I/O NUM. The low byte corresponds to 00 and the high byte to 01.

First comes the command and then the response of the slave.

- Status from device with address 1, low byte
  - Command: #001:200:00:000*
  - Response: #001:200:00:069*
  - Description: The command 200 is sent to the slave with address 1 to send the current device status to the master. In response, we get the low byte with the number 69 (binary: 0100_0101). The watchdog timer has been activated and configured accordingly so that it uses digital output 7, which means counter 1 is active.

- Status from device with address 1, high byte
  - Command: #001:200:01:000*
  - Reply: #001:200:01:001*
  - Description: The command 200 is sent to the slave with address 1 to send the current device status to the master. As response in the high byte, we get the number 1 (binary: 0000_0001), which means counter 3 is active.

10.7.6. PiXtend elO Analog One

In addition to analog inputs and outputs, the Analog One device offers other functions that can be configured and activated via additional PiXtend elO protocol commands.

10.7.6.1 Overview of supported commands

The following commands can be sent to an Analog One device by message to set its additional functions and switch them on or off.

Command	Num	Range	Comment
Read Status	200	0 to 255	Request status from the device. Breakdown of the bits see section 10.7.6.2 Overview of the bits in the status register. Overview of the bits in the status register. The byte to be queried is selected via the I/O number: 0 = Low byte, 1 = High byte.
Set Watchdog-Timer Time	600	0 to 9	Set timeout time for the watchdog timer. See section 10.7.6.3 Watchdog timer configuration. The default is 4 seconds timeout time.
Set Watchdog-Timer Behavior	601	0 to 1	Change watchdog timer behavior when the watchdog timer expires. There are two possibilities: 0 = All outputs maintain their current state (default setting), 1 = All outputs are actively switched off. Please observe our notes on the safe state in chapter 6.2 Definition "Safe State".
Set Watchdog-Timer Enable	604	0 to 1	Switch watchdog timer on/off, value 0 switches the watchdog timer off and value 1 switches it on.

Table 83: Analog One: Overview of supported commands - Advanced

10.7.6.2 Overview of the bits in the status register

The status register shows which function is active in the device and which is not. Cyclic queries are recommended. It also checks whether a function has been activated or not. The status register is 16 bits in size and must therefore be read out via two individual bytes (low byte and high byte). The Analog One device has information in the low byte; the high byte can be ignored; it is not necessary to read it.

Low byte:

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	WDT ON	AO5 OV	AO4 OV
Start value	0	0	0	0	0	0	0	0

Table 84: Bits in the status register - low byte

Bit 0 - AO4 OV → Analog Output 4 Overload Feedback Signal

Off: The analog output 4 operates normally, a load is present, there is no cable break.

On: The analog output 4 has detected an overload situation, usually there is no load or there is a cable break.

Bit 1 - AO5 OV → Analog Output 5 Overload Feedback Signal

Off: The analog output 5 operates normally, a load is present, there is no cable break.

On: The analog output 5 has detected an overload situation, usually there is no load or there is a cable break.

Bit 2 - WDT ON → Watchdog time active

Off: The watchdog timer is off.

On: The watchdog timer is active and, depending on the setting, it is reset and triggered after the specified time has expired. See information in section 10.7.6.3 Watchdog timer configuration.

High byte:

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

Table 85: Bits in the status register - high byte

The high byte of the status register contains no information or status bits and can be ignored.

10.7.6.3 Watchdog timer configuration

The watchdog timer has different settings on the device, most importantly the watchdog timer timeout time. This time determines how long the watchdog timer waits until it is triggered. The watchdog timer for the PiXtend elO protocol is always reset when there is bus activity; there is no other setting here. Please note that the watchdog time requires cyclic communication on the bus after its activation.
To use the watchdog time, first set the timeout time if you do not want to use the default setting of 4 seconds. The following table helps you to select a timeout time.

Number	Timeout	Comment
000	16 ms	Not recommended or very frequent communication necessary
001	32 ms	Not recommended or very frequent communication necessary
002	64 ms	Not recommended or very frequent communication necessary
003	0.125 s	
004	0.250 s	
005	0.5 s	
006	1.0 s	
007	2.0 s	
008	4.0 s	Default setting
009	8.0 s	Longest time-out setting, good with little bus activity

Table 86: Analog One: Overview watchdog timer timeout

The watchdog timer timeout time can be changed with the command 600 (Set Watchdog-Timer Time).

Description	Command to slave	Reply from slave
Command 600, time-out 1 sec.	#003:600:00:006*	#003:600:00:006*
Command 600, time-out 8 sec.	#003:600:00:009*	#003:600:00:009*

Table 87: Setting watchdog timer timeout - examples

If the timeout is set, the watchdog can be activated.

Description	Command to slave	Reply from slave
Command 604, watchdog on	#003:604:00:001*	#003:604:00:001*
Command 604, watchdog off	#003:604:00:000*	#003:604:00:000*

Table 88: Watchdog timer on/off - examples

It is another function of the watchdog timer to switch off all outputs when it expires.

Description	Command to slave	Reply from slave
Command 601, all outputs off	#003:601:00:001*	#003:601:00:001*
Command 601, all outputs remain unchanged	#003:601:00:000*	#003:601:00:000*

Table 89: Setting watchdog timer function - examples

10.7.7. Example - Activate watchdog timer

The following examples should clarify the use and configuration of the watchdog timer on PiXtend elO Analog One devices.

The watchdog timer timeout time can be changed with the command 600.

Description	Command to slave	Reply from slave
Command 600, time-out 1 sec.	#003:600:00:006*	#003:600:00:006*
Command 600, time-out 8 sec.	#003:600:00:009*	#003:600:00:009*

Table 90: Example: Setting watchdog timer timeout

If the timeout is set, the watchdog can be activated.

Description	Command to slave	Reply from slave
Command 604, watchdog on	#003:604:00:001*	#003:604:00:001*
Command 604, watchdog off	#003:604:00:000*	#003:604:00:000*

Table 91: Example: Switch watchdog timer on and off

It is another function of the watchdog timer to switch off all outputs when it expires.

Description	Command to slave	Reply from slave
Command 601, all outputs off	#003:601:00:001*	#003:601:00:001*
Command 601, all outputs remain unchanged	#003:601:00:000*	#003:601:00:000*

Table 92: Example: Setting watchdog timer function

10.7.8. Example - Read status

The following example shows how the current status of the Analog One device can be read with the PiXtend elO protocol. Please note that the status register is a 16-bit value that must be retrieved via a separate low byte and high byte. The byte is selected via the protocol part I/O NUM. The low byte corresponds to 00 and the high byte to 01. Since the Analog One high byte does not contain any information, this byte can be ignored, a query is not necessary.

First comes the command from the master and then the response of the slave.

- Status from device with address 3, low byte
  - Command: #003:200:00:000*
  - Response: #003:200:00:005*
  - Description: The command 200 is sent to the slave with address 3 to send the current device status to the master. In response, we get the low byte with the number 5 (binary: 0000_0101). The watchdog timer is active and the analog output 4 is in an overload situation.

10.8. PiXtend elO protocol- error numbers

If the Device ID is correct and the protocol is correct, it is still possible that the device sends an error message or error number as feedback.

The numbers are:

- 994 - Value - Error
    - The value sent is not appropriate for the desired action and is not allowed. If a digital output is set, only the selection between 0 and 1 is possible as a value. If the value 128 is sent, this error number is returned. This also applies to the Analog One device. If you send the device a command to control the analog outputs with a value between 0 and 4095, everything is fine. If you send the value 6300, you receive this error number as a response; the value is invalid.

- 995 - I/O number incorrect
    - The desired I/O number is incorrect. The Digital One device has 8 inputs and 8 outputs (number range 0 to 7). If an input with I/O number 9 is queried, this is not possible. This same applies to the commands that do not control hardware but process status information. The status register (16 bits in size) of the Digital One device is accessed via two separate bytes: the low byte and the high byte. The selection of which byte is retrieved is done via the I/O number. With two bytes, the I/O numbers 0 and 1 are available. If the device receives a byte selection number greater than 1, this error is returned. This applies to all commands that read and write settings or data on the device. In addition, check the respective device to see which commands it supports and which data can be retrieved or sent.

- 996 - CMD Error - Command is not supported
    - If you send the device a command number that it does not know or does not support, then you get this error number back. For example, you cannot read analog inputs on a Digital One device or set digital outputs on an Analog One device.

11. List of Figures

12. List of Tables