Precaution/Disclaimer:

This Hardware is designed for educational purposes.

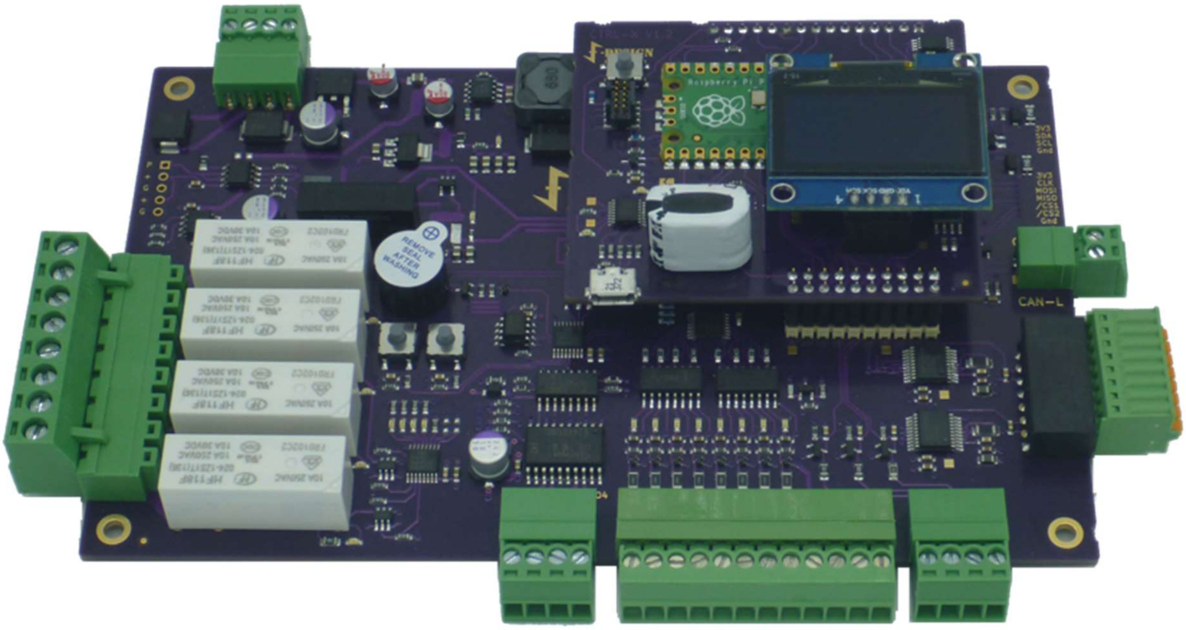

Figure 1: Ctrl-X micro-PLC

1 Table of Content

2 List of Figures

3 Brief Features

- Dimension Main-PCB: 100x160mm
- 8x 24V/Digital-input, isolated with status LED
- 4x 230V/10Amps heavy-duty relay with status LED
- 4x 24V/1Amp Digital-input, isolated
- 3x Analog-Input 0..10V
- 2x Pt100-RTD (3-Wire)
- 1x Buzzer
- 2x Onboard-User-Buttons
- 4x User-LEDs
- 1x PWM-Output 5V/20ms
- 1x external I2C (iso)
- 1x external SPI (iso)
- 1x CAN 2.0 (iso)
- 1x PWM/Servo-Output 5V/24V
- real time clock (RTC), battery buffered
- SRAM for Retain-Variables, battery buffered
- EEPROM for Parameter storage
- Brown-Out detection
- HW-Watchdog
- FTDI-UART, USB-C 2.0
- I2C-Display 64x128 (optional)
- Reset button

4 Supply

- Main-Supply: 24V DC
- Digital-Output-Supply: 24V DC (max. 1A/Channel, each)

5 Boards

The Ctrl-X-PLCY is divided into two functional units.
The main board contains all the power electronics, the digital and analogue peripherals, internal power supplies and interfaces. In addition, some protective components.

The MCU board contains the Raspberry Pi Pico MCU and various devices to turn the Ctrl-X-PLC into a "real" PLC. (e.g. RTC, EEPROM, SRAM).
It is possible to operate the MCU-PCB in standalone mode, for whatever reason

5.1 Main-PCB

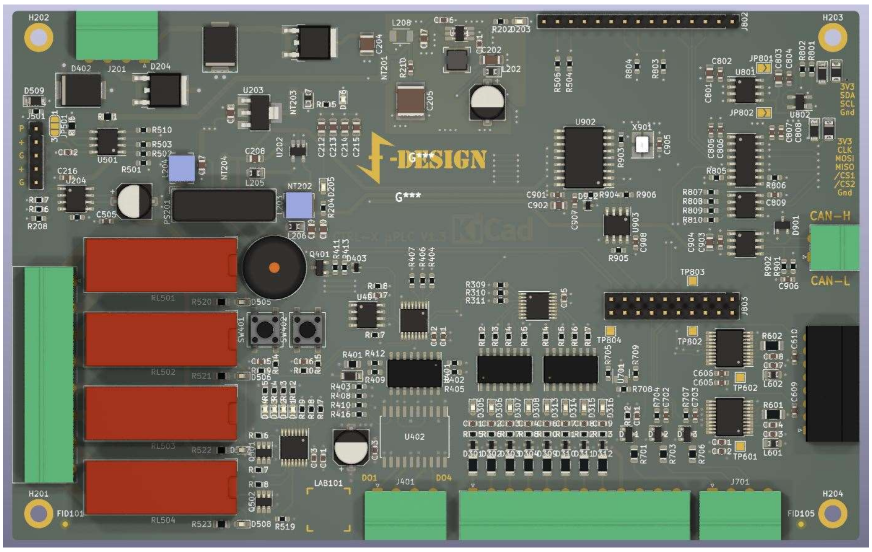

Figure 2: Layout Main-Board

Functions/Components

5.2 MCU-PCB

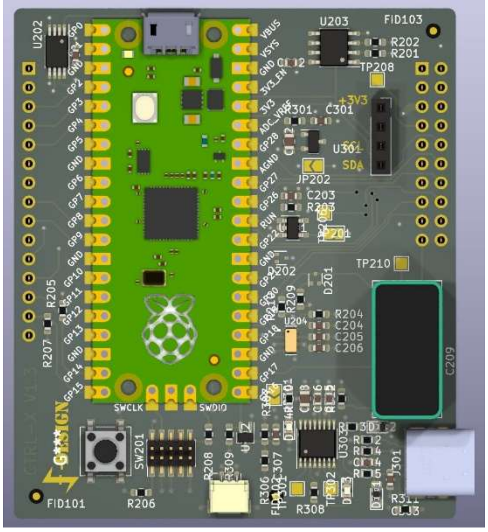

Figure 3: Layout MPU-Board

Functions/Components

6 Connectors Main-PCB

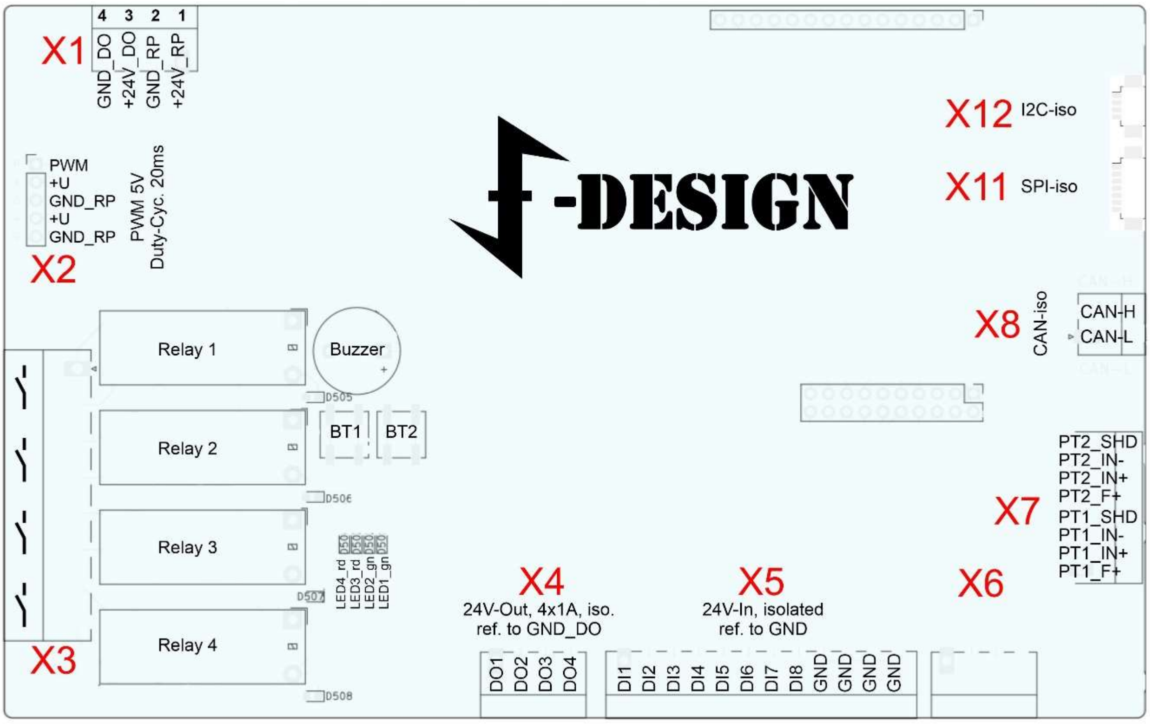

Figure 4: Main-PCB, Terminals

6.1 Main-Supply (X1:1,2)

Operating Voltage: 24V DC, power consumption: max. 400mA
Terminal X1-1: +24V DC, internal
Terminal X1-2: Ground,_internal

6.2 Digital-Output-Supply (X1:3,4) (isolated)

Operating Voltage: 24V DC, power consumption: max. 4A
Terminal X1-3: +24V DC
Terminal X1-4: Ground,_digital out

6.3 Relay-Outputs (X3)

Form: 1Form C, V1, normally open
Contact rating (Res. load): 10A 250V AC/30V DC
Max. switching voltage: 440V AC/125V DC
Terminal X3-1/2: Relay 1, n/o
Terminal X3-3/4: Relay 2, n/o
Terminal X3-5/6: Relay 3, n/o
Terminal X3-7/8: Relay 4, n/o

6.4 Digital-Input (isolated) (X5)

Operating Voltage: 24V DC, power consumption: max. 5,4mA each, reverse polarity protected
Terminal X3-1..8: Digital input
Terminal X3-9..12: Ground_digital_in

6.5 Digital-Ouput (isolated) (X4)

Operating Voltage: 24V DC, nominal load current max. 1A/Channel
Terminal X2-1..4, Digital output

6.6 Analog-Input (X6)

Input range: 0..10V
Terminal X4-1: Analog In 1
Terminal X4-2: Analog In 2
Terminal X4-3: Analog In 3
Terminal X4-4..12: Ground_internal

6.7 RTD-Sensor-Input (X7)

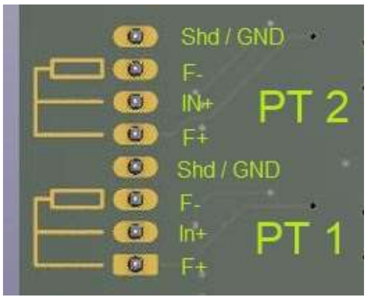

Figure 5: Pt100 3-Wire-Connection Bottom View

The Temperature-Inputs are optimized for Pt100-Sensors (RTDs). At the Bottom of Main-PCB the required wiring is shown. Total Accuracy Over All Operating Conditions is 0.5°C or 0.05% of Full Scale, maximum. Maximum Conversion Time is 21ms. In noisy environment it is recommended to use shielded sensor cable. The max. Cable Resistance (per lead) is 50Ω.The Voltage-Protection for Sensor-Inputs is ±45V.
WARNING: Sensor-Inputs will not survive Powerline-Voltage.

6.8 CAN 2.0-Interface (isolated) (X8)

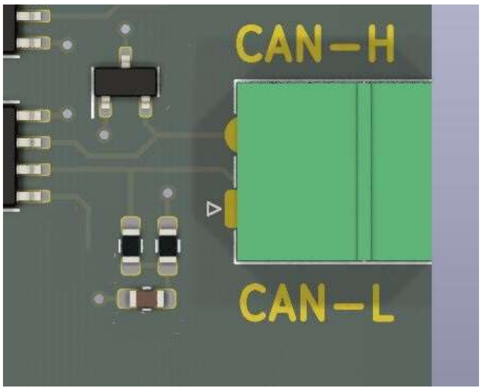

Figure 6: CAN2.0-Wiring Top View

The 2-Wire-CAN-Interface is based on MCP2561-H/SN-Driver. CAN-High/CAN-Low-Line is balanced and terminated with 59Ω/each. CAN-Interface (I2C-Interface as well) is decoupled by Quad-Channel Digital Isolator from Raspberry Pi Pico Supply-Voltage. The interface is protected by ESD Protection Diodes. The CAN-Interface is connected to Raspberry Pi Pico “SPI0”.

An opto-coupled hardware interrupt enables a fast response to communication Rx-Requests. A software implementation of the CAN interface is available at https://github.com/runout0.

6.9 External I2C-Interface (isolated) (X12)

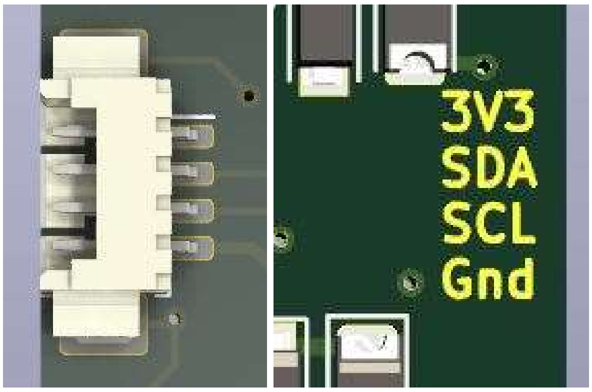

Figure 7: external I2C-Interface

The supply via the 3V3 pin is required to use the external I2C interface.
3V3-Supply Voltage is protected by a Polyfuse und a Clamp Diode.

The 3V3 pin of the I2C interface and the 3V3 pin of the SPI interface are identical, so it is sufficient to supply only one of them.

6.10 External SPI-Interface (isolated) (X11)

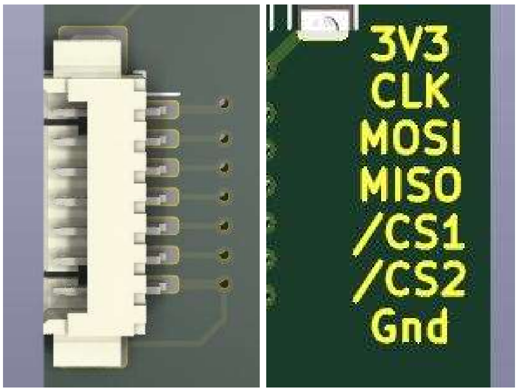

Figure 8: external SPI-Interface

The supply via the 3V3 pin is required to use the external SPI interface.
3V3-Supply Voltage is protected by a Polyfuse und a Clamp Diode.

The 3V3 pin of the SPI interface and the 3V3 pin of the I2CI interface are identical, so it is sufficient to supply only one of them.

6.11 PWM/Servo-Interface (X2)

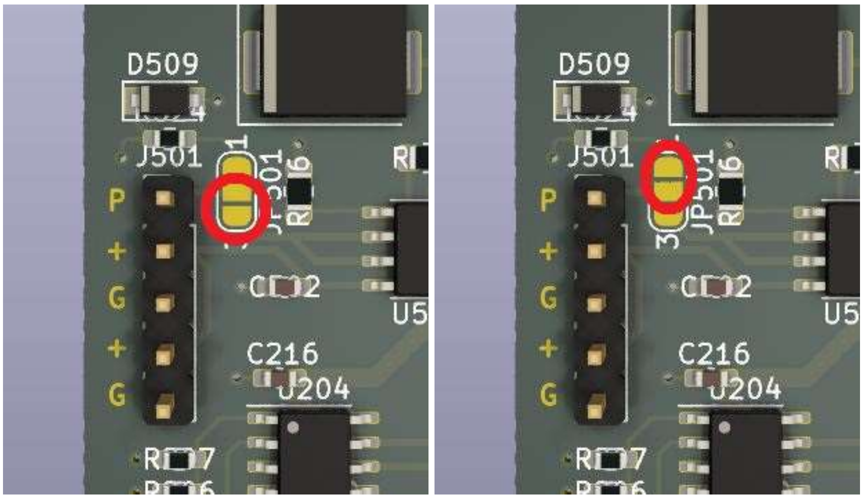

Figure 9: PWM-Interface/Fast Digital-Input

SolderJumper

- Lower Position: P = PWM-Output (5V) – Default Setting
- Upper Position: P = Fast Digital-Input (Interrupt), Remark: not isolated, not protected, 3V3

External PWM-Power

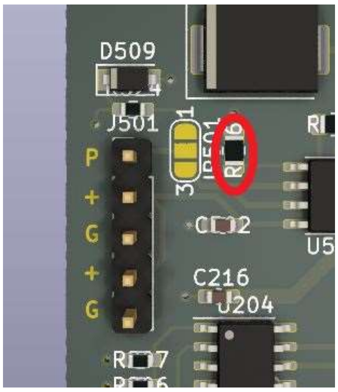

Figure 10: External PWM-Power, 0-Ohm Resistor

The PWM-Output (5V) is limited to approx. 100mA.

If the starting current is too high, remove the marked resistor and supply the servo (e.g.) via (+) pins.

7 Connectors MCU-PCB

- Internal Connector 1: Pin-Header 1x15, 2,54mm
- Internal Connector 2: Pin-Header 2x10, 2,54mm

7.1 Program-Interface (SWD)

Connector for SWD programming/debugging with e.g. Segger J-Link™ or ST-Link™.
(a 20pin-10pin adapter is required, e.g. ARM-JTAG-20-10)

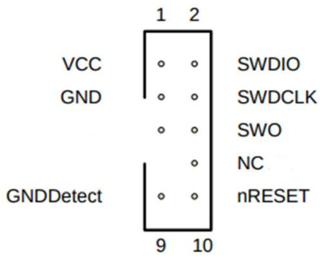

Figure 11: Standard Cortex Debug (10-pin)

A simpler, if somewhat inconvenient, solution is to use a Raspberry Pi debug probe.

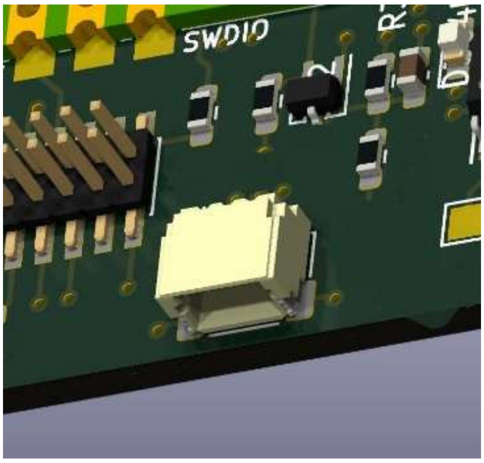

Figure 12: Pico-Debug-Probe SWD (3-pin)

In harsh environments, it may make sense to use a USB isolator for programming/debugging.

7.2 USB-UART-Interface

The USB2.0 Interface is based on a FTDI-Chip and connected to Raspberry Pi Pico UART0.

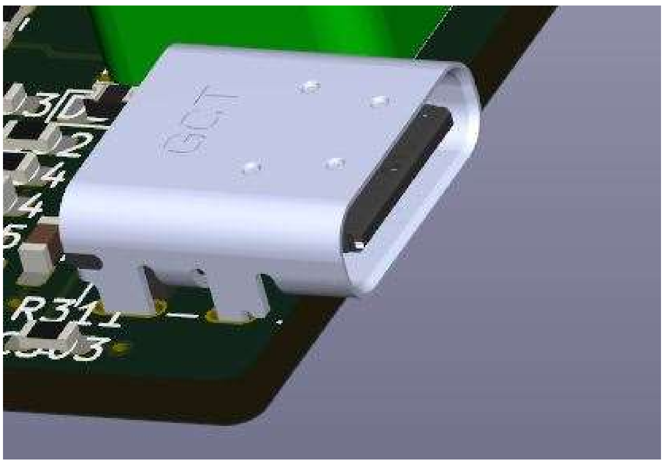

Figure 13: USBC 2.0 Interface

7.3 I2C-Interface (Display)

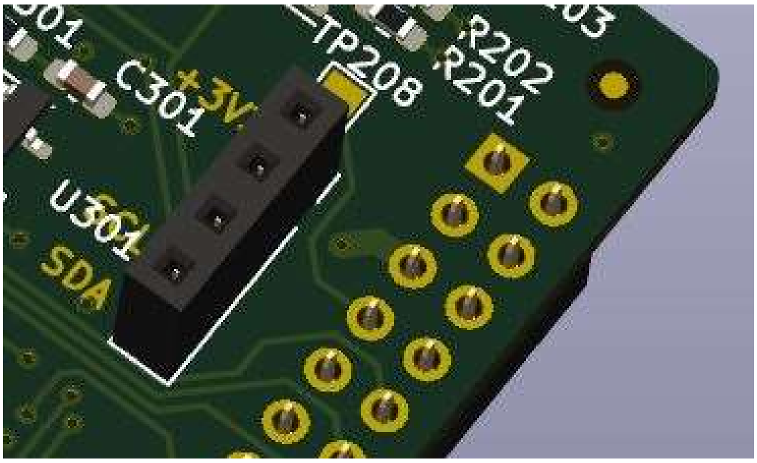

Figure 14: I2C-Display-Interface

The I2C-Interface on MCP-Board is designed to plug&play a 1,3"-Graphic-Display.
The I2C-Interface is connected to Raspberry Pi Pico "I2C1".

A software implementation of the display interface is available at Github.

8 Data-Structure

8.1 I/O-Structure

8.1.1 Digital input

DI-Variable	PLC-Name	Beremiz-Name
DI1	XPLC.IO.di_int[DI_INTERN0].DI[0]	%IX0.0
DI2	XPLC.IO.di_int[DI_INTERN0].DI[1]	%IX0.1
DI3	XPLC.IO.di_int[DI_INTERN0].DI[2]	%IX0.2
DI4	XPLC.IO.di_int[DI_INTERN0].DI[3]	%IX0.3
DI5	XPLC.IO.di_int[DI_INTERN0].DI[4]	%IX0.4
DI6	XPLC.IO.di_int[DI_INTERN0].DI[5]	%IX0.5
DI7	XPLC.IO.di_int[DI_INTERN0].DI[6]	%IX0.6
DI8	XPLC.IO.di_int[DI_INTERN0].DI[7]	%IX0.7
BTN1	XPLC.IO.di_int[DI_INTERN0].DI[8]	%IX0.8
BTN2	XPLC.IO.di_int[DI_INTERN0].DI[9]	%IX0.9
STATUS_DO	XPLC.IO.di_int[DI_INTERN0].DI[10]	%IX0.10

8.1.2 Digital output

DO-Variable	PLC-Name	Beremiz-Name
LED1_GN	XPLC.IO.do_int[DO_INTERN0].DO[0]	%QX0.0
LED2_GN	XPLC.IO.do_int[DO_INTERN0].DO[1]	%QX0.1
LED3_RD	XPLC.IO.do_int[DO_INTERN0].DO[2]	%QX0.2
LED4_RD	XPLC.IO.do_int[DO_INTERN0].DO[3]	%QX0.3
RELAY1	XPLC.IO.do_int[DO_INTERN0].DO[4]	%QX0.4
RELAY2	XPLC.IO.do_int[DO_INTERN0].DO[5]	%QX0.5
RELAY3	XPLC.IO.do_int[DO_INTERN0].DO[6]	%QX0.6
RELAY4	XPLC.IO.do_int[DO_INTERN0].DO[7]	%QX0.7
DO1	XPLC.IO.do_int[DO_INTERN0].DO[8]	%QX0.8
DO2	XPLC.IO.do_int[DO_INTERN0].DO[9]	%QX0.9
DO3	XPLC.IO.do_int[DO_INTERN0].DO[10]	%QX0.10
DO4	XPLC.IO.do_int[DO_INTERN0].DO[11]	%QX0.11
BUZZER	XPLC.IO.do_int[DO_INTERN0].DO[12]	%QX0.12

8.1.3 Analog-Input

AI-Variable	PLC-Name	Beremiz-Name
AI_ADC0	XPLC.IO.ai_int[AI_INTERN0].AI[0]	%IW0.0
AI_ADC1	XPLC.IO.ai_int[AI_INTERN0].AI[1]	%IW0.1
AI_ADC2	XPLC.IO.ai_int[AI_INTERN0].AI[2]	%IW0.2
AI_PT1	XPLC.IO.ai_int[AI_INTERN0].PT[0]	%IW0.3
AI_PT2	XPLC.IO.ai_int[AI_INTERN0].PT[1]	%IW0.4

8.1.4 Analog-Output

AO-Variable	PLC-Name	Beremiz-Name
AO_PWM	XPLC.IO.ao_int[OI_INTERN0].AO[0]	%QW0.0

8.2 Config-Structure

tbd