Position Sensor Evaluation Software

Version 0.1.0

Table of Contents

Contents:

Introduction 4

Installation 6

GUI Reference 11

- Main GUI 11
- Dev GUI 12

GUI Usage 15

- AS5173 15

API Reference 18

- QMH Module 18
- PosSensor Module 18

API Usage 38

- Getting Started 38
- Valid Sensor Names 38
- Coding examples 38

Folder Structure 46

INI Configuration Reference 47

- TCP_Server 47
- AS5171_Analog 47
- AS5171_Digital 48
- AS5172 48
- AS5173 15
- AS5270_Analog 49
- AS5270_Digital 49

Welcome to Position Sensor Software's documentation!

Introduction

The Position Sensor Software is an evaluation software developed to help testing position sensor designed by Infineon.

Infineon has a wide portfolio of position sensors, the following table shows the supported sensors “referred to as Devices too”

		Drivers				
		SD4Yv02	SD4Yv01	UART_PCB	sim	PSI5_PCB
Devices	AS5171_Analog	✓	✓	✓	✓	NS
	AS5171_Digital	✓	✓	✓	✓	NS
	AS5270_Analog	✓	✓	✓	✓	NS
	AS5270_Digital	✓	✓	✓	✓	NS
	AS5172	✓	NS	NS	✓	✓
	AS5173	✓	NS	NS	✓	✓

NS: Not supported by hardware

Devices:

AS5171 : SIP package with analog and digital output interfaces.

AS5270 : Dual Die package with analog and digital output interfaces.

AS5172 : SIP package with PSI5 output.

AS5173 : SIP package with PSI5 output and an integrated filter angle.

Drivers:

Drivers are the hardware between the PC and the sensor, it has the sensor’s specific interface. Drivers can be an evaluation PCB or a production ready programmer.

Available drivers:

SD4Yv01 : Production ready programmer.

SD4Yv02 : Production ready programmer.

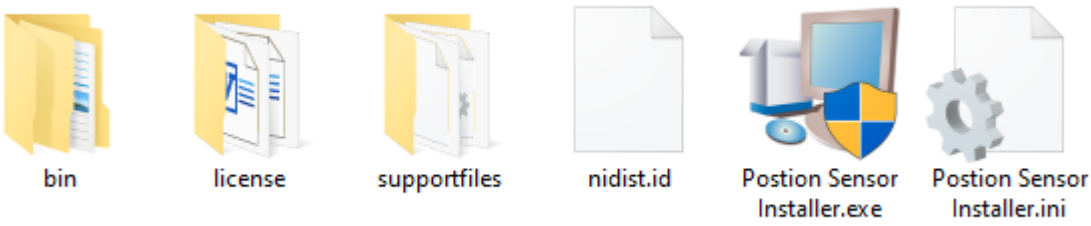
UART_PCB : Evaluation PCB.

PSI5_PCB : Evaluation PCB.

Installation

This section explains how to Install the position sensor evaluation software.

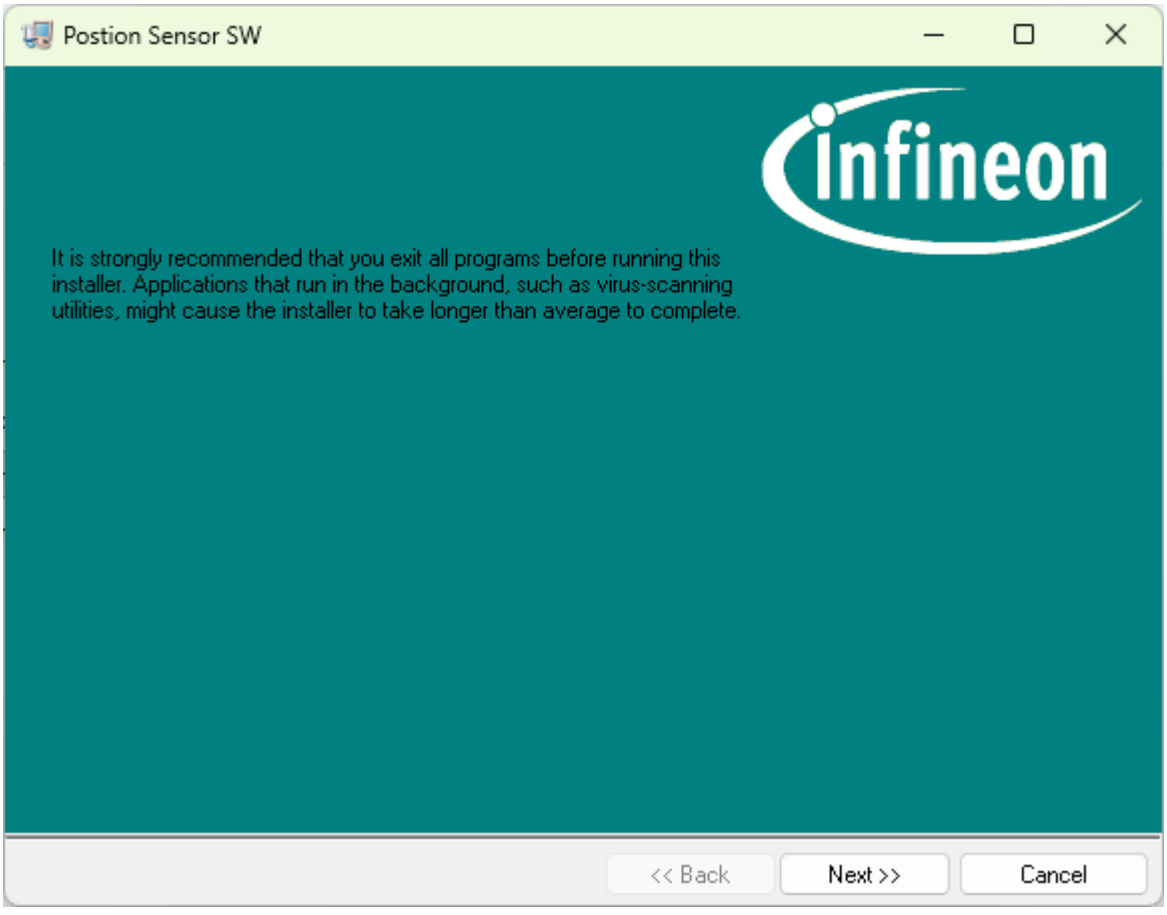
After downloading the zip file and extracting it, it will look like this:



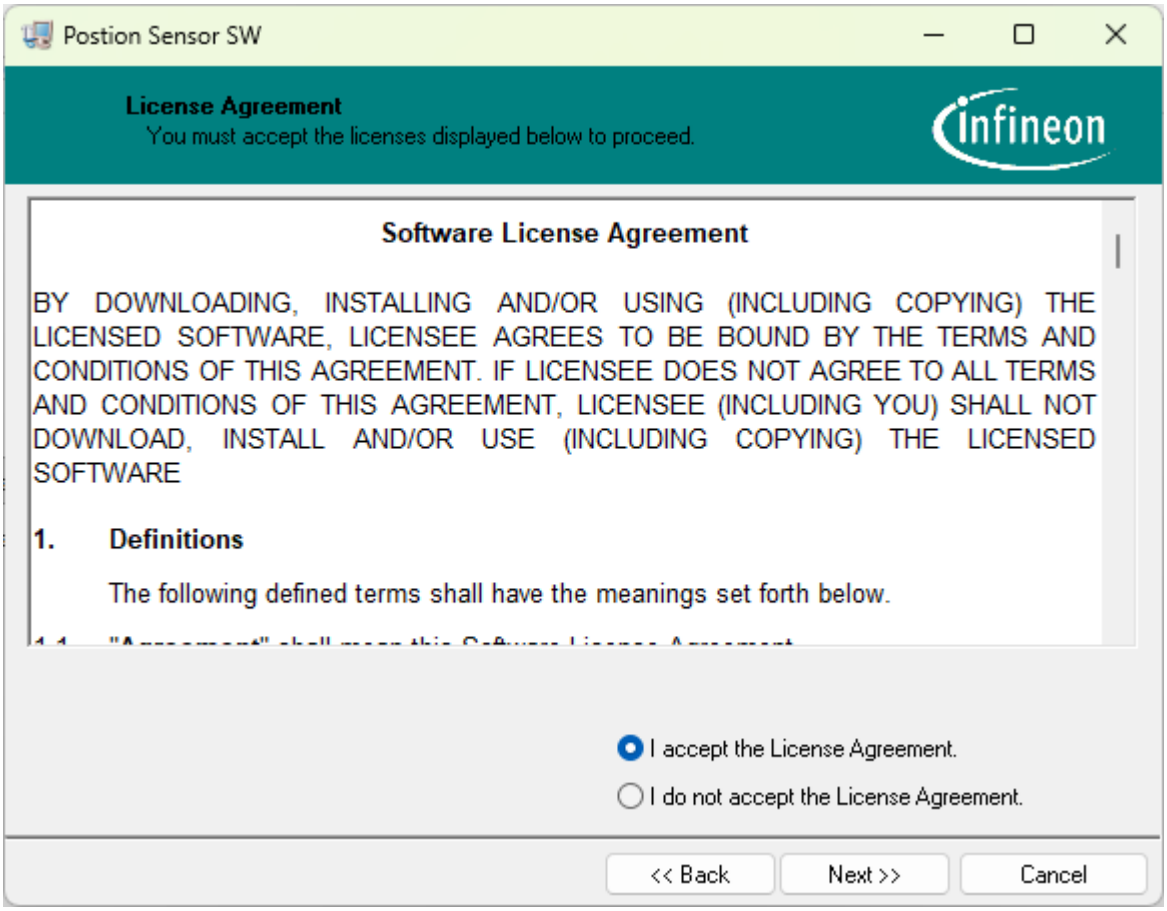
Start Position Sensor Installer.exe

And follow the following steps:

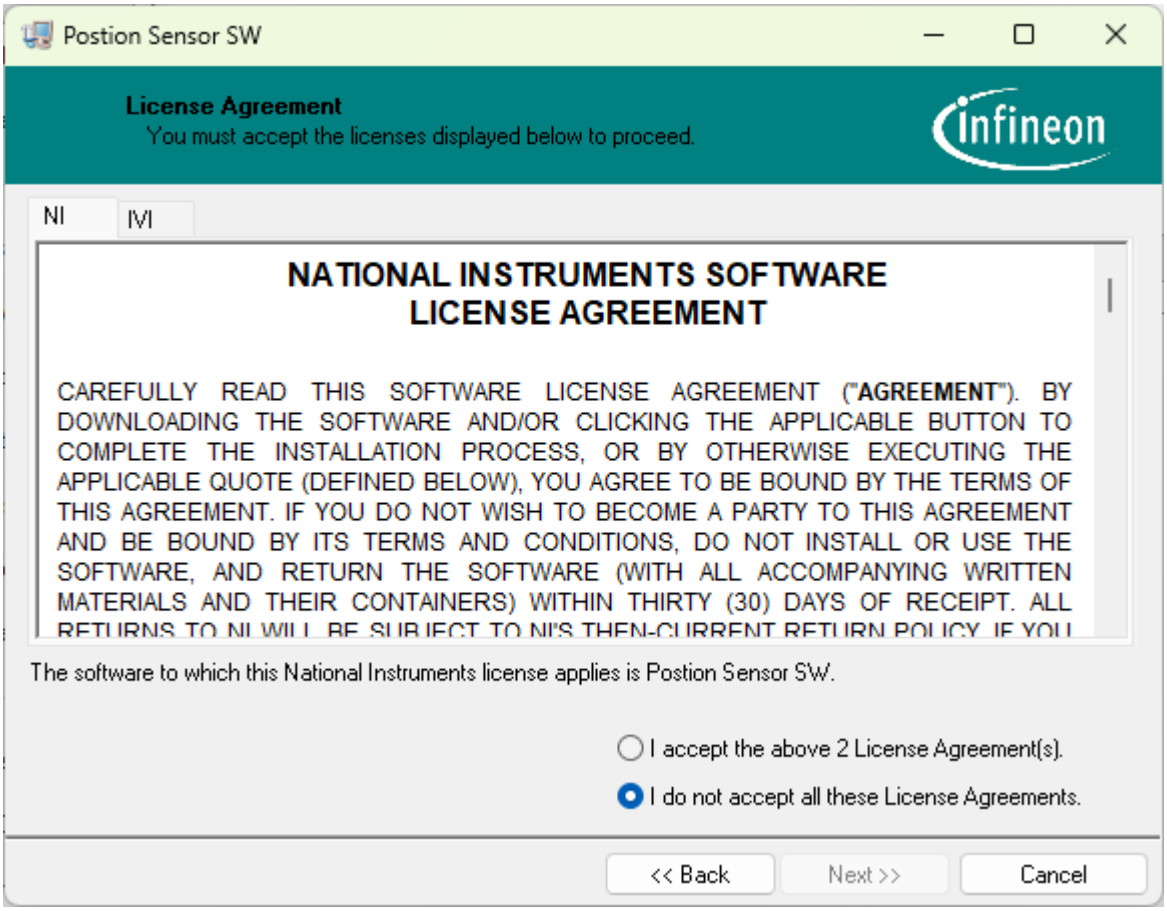
- Click Next



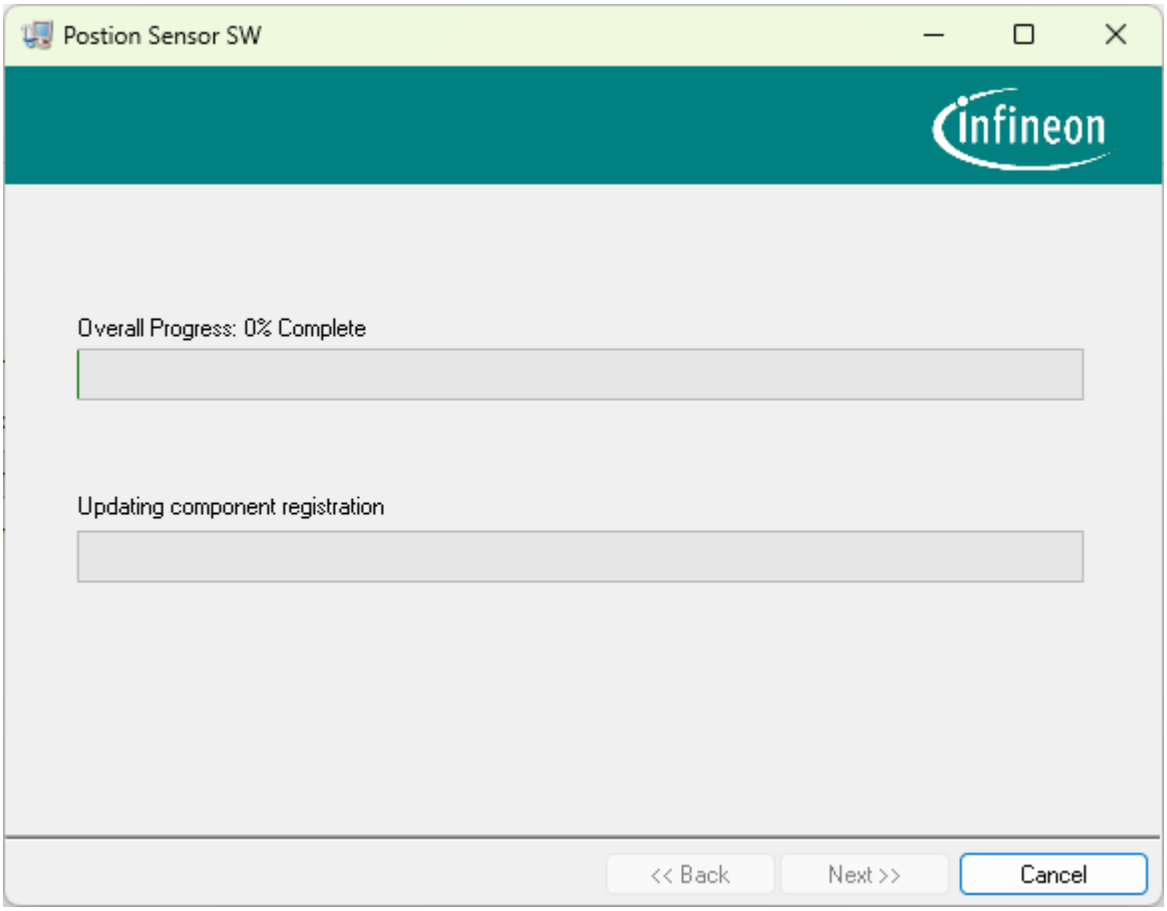
- Read and accept the software license agreement and click Next



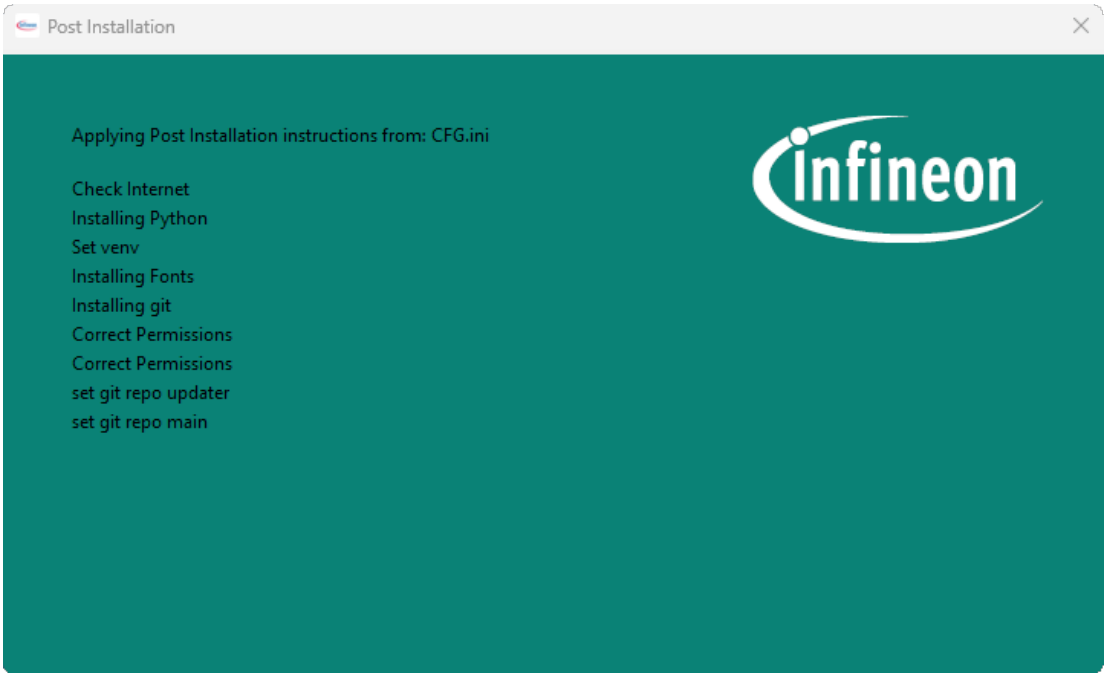
・Read and accept the NI components license agreement and click Next



・The installation will start



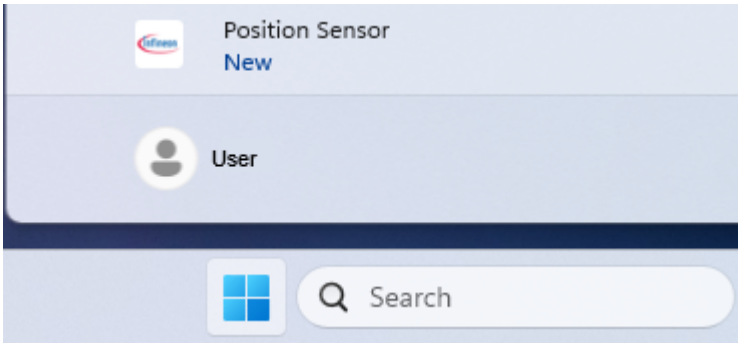
- Afterwords, the Post-Installation window will install and configuration other components needed by the software. **“This step requires internet access”**



- After this step, the installer will ask you to reboot the PC.

- The application can be found on the Desktop:

or from the Start Menu:

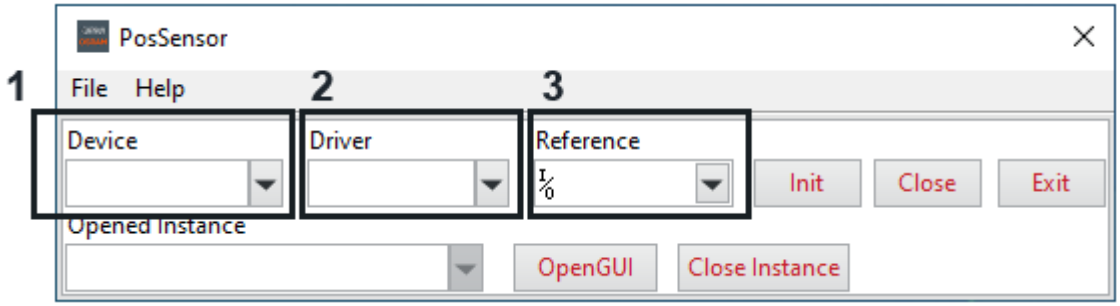


GUI Reference

This section explains how to use the GUI in details.

Main GUI

In the image below, the APP main GUI is shown

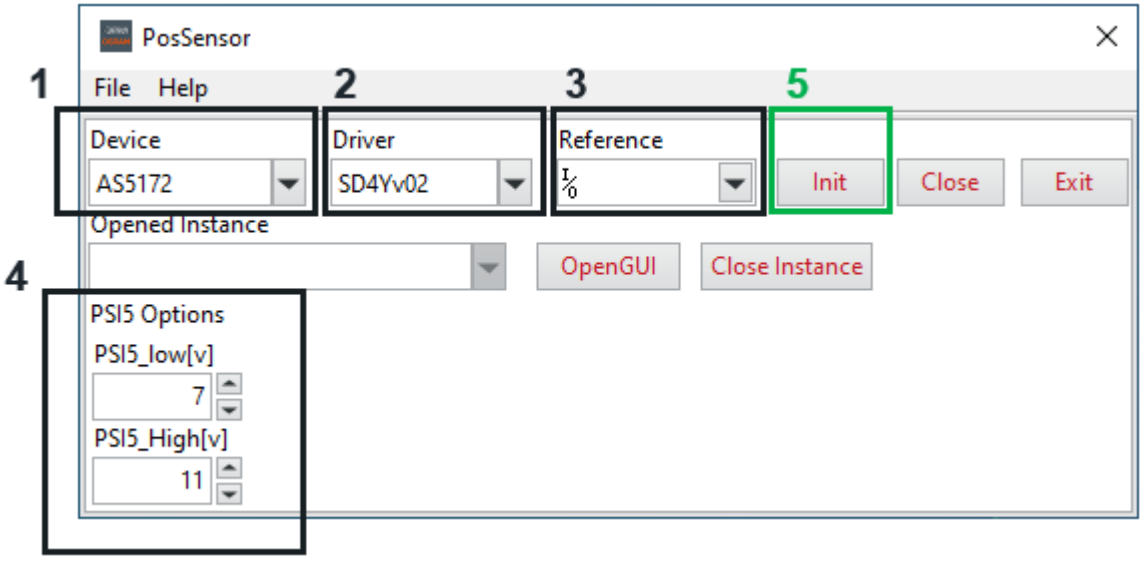


To start a connection with a sensor, do the following

1. Select a device.

2. Select a driver.

3. Select the com-port for the driver.



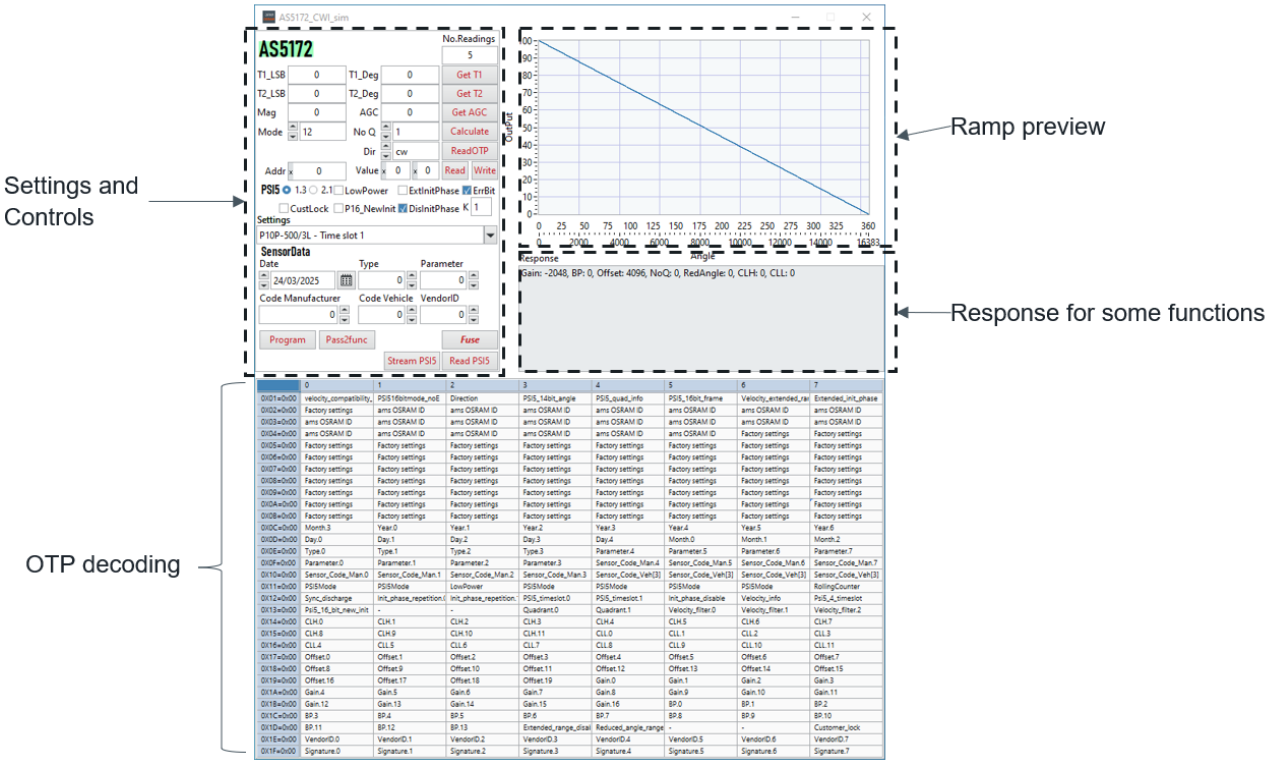
1. For some Device/Driver combinations, there will be optional settings.

2. Click Init.

Dev GUI

After successful initialization of the driver, a new windows popup will appear. It is called Device GUI

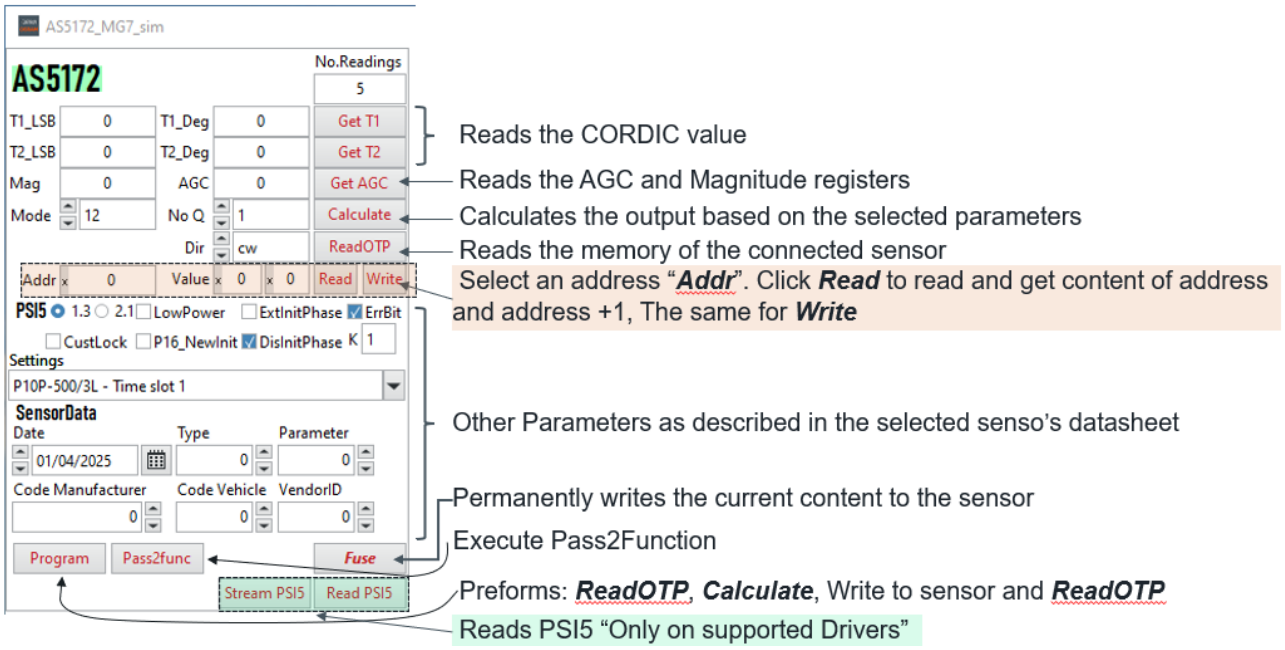
The Device GUI is shown in the image below:



It has 4 components

1. Controls and settings: It sets all desired settings and have all the controls.

In the image below an example of the Controls and settings with explanation of each function



1. Ramp Preview: Shows the output based on the user's input.

2. Response: Shows a feedback to certain functions.

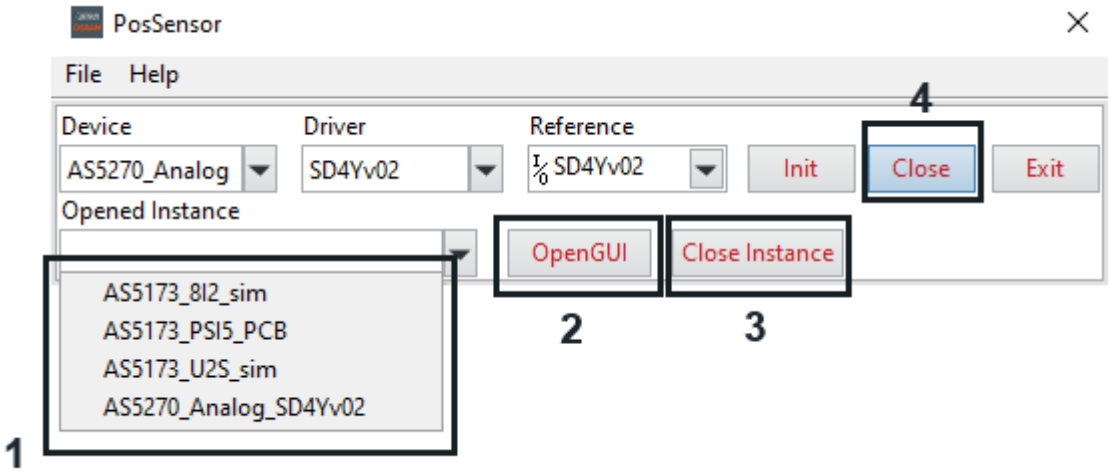
3. OTP decoding: Shows the device memory after each read or write operation.

In the image below, the OTP of device is shown. If a bit is green it means it has 1 otherwise a zero is written to it.:

	0	1	2	3	4	5	6	7
0X01=0x00	velocity_compatibility_	PSI516bitmode_noE	Direction	PSI5_14bit_angle	PSI5_quad_info	PSI5_16bit_frame	Velocity_extended_ra	Extended_init_phase
0X02=0x28	Factory settings	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID
0X03=0xCC	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID
0X04=0x13	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	ams OSRAM ID	Factory settings	Factory settings
0X05=0x10	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings
0X06=0x00	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings
0X07=0x4D	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings
0X08=0x04	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings
0X09=0x41	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings
0X0A=0x08	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings
0X0B=0x0F	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings	Factory settings
0X0C=0x32	Month.3	Year.0	Year.1	Year.2	Year.3	Year.4	Year.5	Year.6
0X0D=0x81	Day.0	Day.1	Day.2	Day.3	Day.4	Month.0	Month.1	Month.2
0X0E=0x00	Type.0	Type.1	Type.2	Type.3	Parameter.4	Parameter.5	Parameter.6	Parameter.7
0X0F=0x00	Parameter.0	Parameter.1	Parameter.2	Parameter.3	Sensor_Code_Man.4	Sensor_Code_Man.5	Sensor_Code_Man.6	Sensor_Code_Man.7
0X10=0x00	Sensor_Code_Man.0	Sensor_Code_Man.1	Sensor_Code_Man.2	Sensor_Code_Man.3	Sensor_Code_Veh[3]	Sensor_Code_Veh[3]	Sensor_Code_Veh[3]	Sensor_Code_Veh[3]
0X11=0x00	PSI5Mode	PSI5Mode	LowPower	PSI5Mode	PSI5Mode	PSI5Mode	PSI5Mode	RollingCounter
0X12=0x20	Sync_discharge	Init_phase_repetition.(	Init_phase_repetition.'	PSI5_timeslot.0	PSI5_timeslot.1	Init_phase_disable	Velocity_info	Psi5_4_timeslot
0X13=0x00	Psi5_16_bit_new_init	-	-	Quadrant.0	Quadrant.1	Velocity_filter.0	Velocity_filter.1	Velocity_filter.2
0X14=0x00	CLH.0	CLH.1	CLH.2	CLH.3	CLH.4	CLH.5	CLH.6	CLH.7
0X15=0x00	CLH.8	CLH.9	CLH.10	CLH.11	CLL.0	CLL.1	CLL.2	CLL.3
0X16=0x00	CLL.4	CLL.5	CLL.6	CLL.7	CLL.8	CLL.9	CLL.10	CLL.11
0X17=0x00	Offset.0	Offset.1	Offset.2	Offset.3	Offset.4	Offset.5	Offset.6	Offset.7
0X18=0x10	Offset.8	Offset.9	Offset.10	Offset.11	Offset.12	Offset.13	Offset.14	Offset.15
0X19=0x00	Offset.16	Offset.17	Offset.18	Offset.19	Gain.0	Gain.1	Gain.2	Gain.3
0X1A=0x80	Gain.4	Gain.5	Gain.6	Gain.7	Gain.8	Gain.9	Gain.10	Gain.11
0X1B=0x1F	Gain.12	Gain.13	Gain.14	Gain.15	Gain.16	BP.0	BP.1	BP.2
0X1C=0x00	BP.3	BP.4	BP.5	BP.6	BP.7	BP.8	BP.9	BP.10
0X1D=0x00	BP.11	BP.12	BP.13	Extended_range_disal	Reduced_angle_range	-	-	Customer_lock
0X1E=0x00	VendorID.0	VendorID.1	VendorID.2	VendorID.3	VendorID.4	VendorID.5	VendorID.6	VendorID.7
0X1F=0x4D	Signature.0	Signature.1	Signature.2	Signature.3	Signature.4	Signature.5	Signature.6	Signature.7

The GUI allows multiple connections at the same time. So if two or more drivers are connected to the PC, both of them can be initiated.

The main GUI as in the image below, has a drop down menu (1), that shows all the opened instances:



By clicking on OpenGUI (2), the Device GUI will come to the front. Close instance will close the selected and make the driver free.

If the user wants to close all the opened instances, Close button (4) can be used.

The Device GUI uses the same names and terminology according to the datasheet.

GUI Usage

This section explains how to use the GUI through some examples.

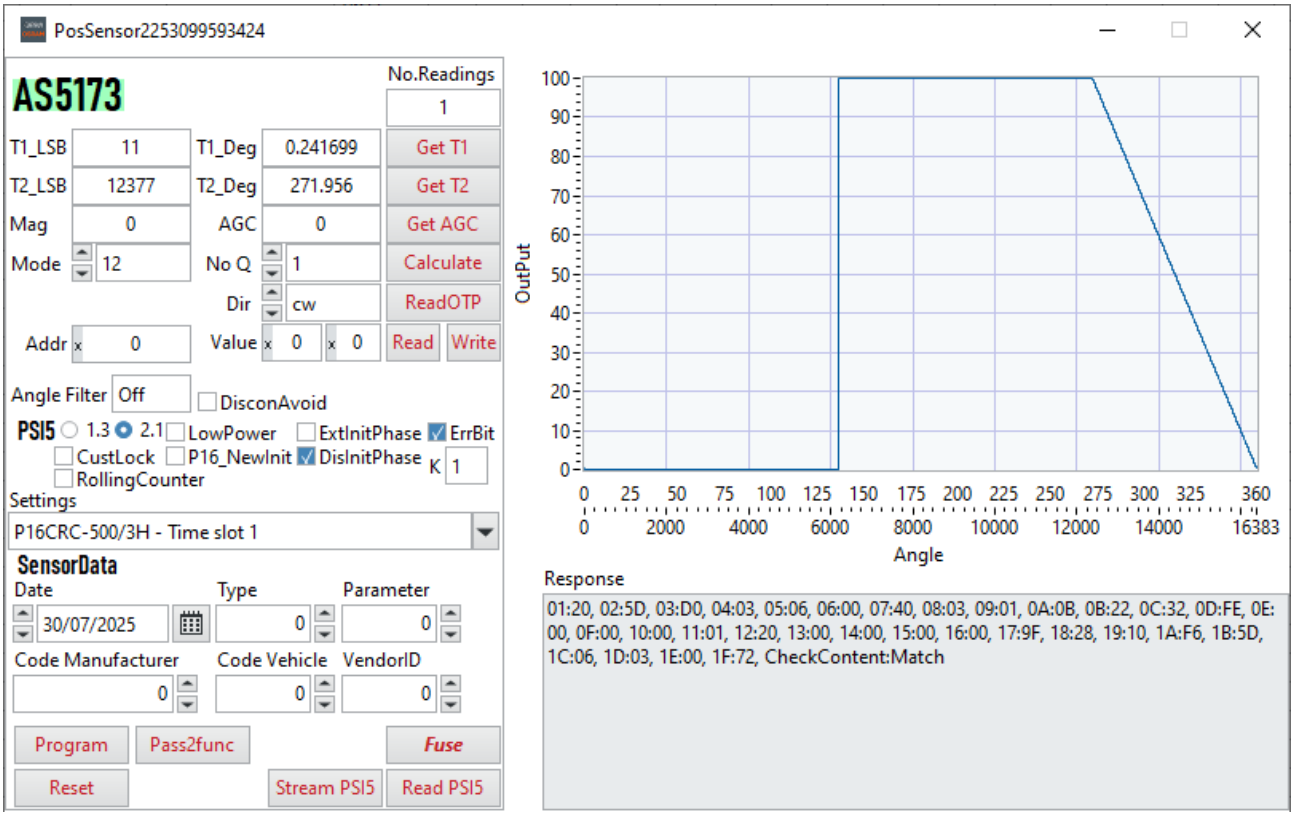
AS5173

Example 1

- Segment: 90 °
- PSI5 configuration: P16CRC-500/3H - Time slot1
- Output resolution: 12 bit

Steps:

- Move to the mechanical position one and click *Get T1* .
- Move to the mechanical position two and click *Get T2* .
- Select Mode > *12*
- Select PSI5 > *2.1*
- Select Settings > *P16CRC-500 - Time slot1*
- Click *Program*



- If the sensor is to be programmed permanently, Click Fuse. Otherwise use Pass2func.

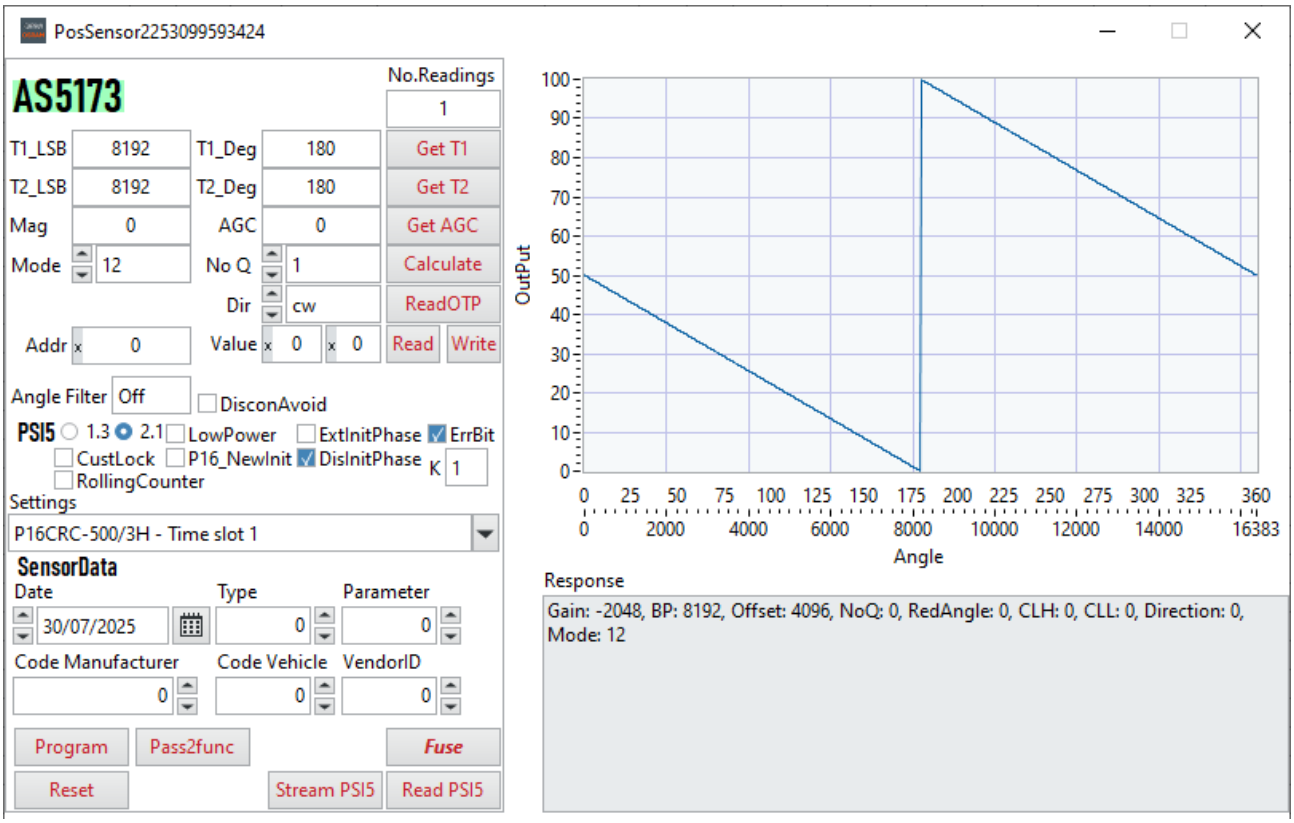
Example 2

- Segment: 360 °
- Zero position: modified
- PSI5 configuration: P16CRC-500/3H - Time slot1
- Output resolution: 12 bit

Steps:

- Move to the mechanical position of zero position.
- Click *Get T1* .
- Copy the same value from *T1_LSB* to *T2_LSB* .
- Select Mode > *12*
- Select PSI5 > *2.1*
- Select Settings > P16CRC-500 - Time slot1.

- Other metadata can be selected such as Date, Type ,etc.
- If the device will be fused, don't forget to select *CustLock* .
- By default PSI5 initialization data is disabled, if this data is needed, deselect *DisInitPhase* .
- Click *Program*



API Reference

QMH Module

This module provides the connection between the Python environment and the Position Software.

class qmh_pyclient.QMH. QMH

Bases: `object`

The main class that connects Python to the API.

ServerConn (*serverPort : int = 50007*) → None

Connects to the Python API server.

Parameters :

- **serverHost** (*str*) – IP address or hostname of the PC where the Position Software is running, defaults to *'localhost'* .
- **serverPort** (*int*) – Port number to connect to, defaults to *50007* .

Returns :

None

PosSensor Module

This module provides position sensor interfaces.

Driver Classes

The Driver class is used to define the programmer to be used with the sensor.

class qmh_pyclient.PosSensor. SD4Yv01 (*REF : str* , *Baudrate : int = 9600*)

Bases: `PosSensorDriver`

Driver class for the SD4Yv01 programmer.

Variables :

- **REF** (*str*) – Com port assigned to the SD4Yv01 programmer.
- **Baudrate** (*int*) – Baudrate between PC and the programmer, default is 9600.

class qmh_pyclient.PosSensor. SD4Yv02 (*REF : str* , *Baudrate : int = 115200* , *V_PSI5_h : float = 11* , *V_PSI5_l : float = 5.5* , *Sensor_Baud_Rate : int = 39000*)

Bases: `PosSensorDriver`

Driver class for the SD4Yv02 position sensor.

Variables :

- **REF** (*str*) – Com port assigned to the SD4Yv02 programmer.
- **Baudrate** (*int*) – Baudrate between PC and the programmer, default is 115200.
- **V_PSI5_h** (*float*) – PSI5 sync Pulse Voltage/ UARToverPSI5 high voltage, default is 11.
- **V_PSI5_l** (*float*) – PSI5 Idle voltage/ UARToverPSI5 low voltage, default is 5.5.
- **Sensor_Baud_Rate** (*int*) – Sensor-specific baud rate setting, default is 39000.

class qmh_pyclient.PosSensor. PSI5_PCBv01 (*REF : str* , *Baudrate : int = 38400*)

Bases: `PosSensorDriver`

Driver Class for PSI5_PCBv01.

Parameters :

- **REF** (*str*) – Com port assigned to the PSI5_PCBv01 programmer.

- **Baudrate** (*int*) – Baudrate between PSI5_PCBv01 and the Sensor (defaults to 38400).

class qmh_pyclient.PosSensor. UART_PCB (*REF : str* , *Baudrate : int = 9600* , *Connection : str = 'Top'*)

Bases: `PosSensorDriver`

Driver Class for UART_PCB.

Variables :

- **REF** (*str*) – Com port assigned to the PSI5_PCBv01 programmer.
- **Baudrate** (*int*) – Baudrate between UART_PCB and the Sensor (defaults to 9600).
- **Connection** (*str*) – Where is the sensor connected to the board, options are Top or Bot (defaults to Top).

class qmh_pyclient.PosSensor. Sim

Bases: `PosSensorDriver`

Driver Class for simulated driver. It is used without any hardware.

Position Sensor Classes

The PosSen class is used to define the sensor aka device to be used.

class qmh_pyclient.PosSensor. PosSen (*Sensor : str* , *Driver : PosSensorDriver*)

Bases: `object`

Base Class for position sensors.

init (*Sensor* , *Driver* , *Name = ''*)

Initiates a Sensor/Programmer combination based on inputs.

Parameters :

- **Sensor** (*str*) – Sensor name.
- **Driver** (Driver Classes) – A compatible programer for the sensor.

Write (*Reg* , *Data*)

Writes Data to a register address.

Parameters :

- **Reg** (*int*) – The address of the register.
- **Data** (*int*) – The data to be written to the register.

Return type :

None

Read (*Reg*)

Reads a register address.

Parameters :

Reg (*int*) – The address of the register.

Returns :

A list of Address's value, Address+1's value:

Return type :

list

Close ( )

Closes all opened connections to the programmers.

Return type :

None

CloseDev ( )

Closes the opened connection to the programmer.

Return type :

None

Pass2Func ( )

Execute the pass2function command

Return type :

None

Fuse ( ) → bool

Start the fusing process “Burn Command”

Returns :

True if the burn command was executed correctly.

Return type :

boolean

REG_Read_OTP (*CheckContent = False*)

Reads the OTP Content. Optionally it can check if the acquired data matches what was written. Sensor.OTP{i:02x} will be also available i.e Sensor.OTP02 with contents of 0x02 or Sensor.OTP1e for address 0x1E.

Returns :

If CheckContent=False:

- A tuple of:
  - A list containing the OTP values and
  - None.

If CheckContent=True:

- A tuple of:
  - A tuple of the OTP values
  - Boolean result of CheckContent.

Return type :

tuple

REG_Write_OTP ( )

Writes and returns OTP Content.

Returns :

A list containing the OTP values.

Return type :

list

REG_Write_Array (*Array*)

Writes an array to the OTP

Read_Cordic (*NoOfReads = 10*)

Reads the CORDIC value.

Parameters :

NoOfReads (*int*) – The number of readings.

Returns :

A list containing CORDIC value in lsb.

Return type :

list

Read_Mag_AGC (*NoOfReads = 10*)

Reads the Magnitude and the AGC registers.

Parameters :

NoOfReads (*int*) – The number of readings.

Returns :

A list containing:

- Mag (list int): list of Mag readings.
- AGC (list int): list of AGC readings.

Return type :

list

Sim_ChangeAngleRel (*Target*)

For a simulated device, the angle can changed to relative to the current position.

Parameters :

Target (*float*) – Relative position.

Return type :

None

Sim_ChangeAngleAbs (*Target*)

For a simulated device, the angle can changed to an absolute position.

Parameters :

Target (*float*) – Absolute position.

Return type :

None

param_calc (*T1* , *T2* , *Direction* , *No_Q* , *CLH* , *CLL*)

Calculates the values of Gain, BP, Offset, NoQ, RedAngle, CLH, CLL. Works for AS5171, AS5172, AS5270

Parameters :

- **T1** (*int*) – Angle 1.
- **T2** (*int*) – Angle 2.
- **Direction** (*str*) – Sets rising or falling ramp output.
  Valid inputs are: 'cw', 'ccw'
- **No_Q** (*int*) – No of quadrants
- **CLH** (*int*) – Clamping level high.
- **CLL** (*int*) – Clamping level high.

Returns :

A tuple containing:

- Gain (int): Output gain factor.
- BP (int): Output Breakpoint.
- Offset (int): Output offset value.
- NoQ (int): Number of quadrants.
- RedAngle (float): Reduced angle enabled.
- CLH (float): clamping limit high.
- CLL (float): clamping limit low.

Return type :

tuple

AS5171 Class

The AS5171 class is the base class used to define the sensor as AS5171.

class qmh_pyclient.PosSensor. AS5171 (*Driver* , *** kwargs*)

Bases: `PosSen`

OTP_1st_Addr *= 2*

Uses the methods from the base class `PosSen`

Write (*Reg* , *Data* , *Data_1 = 0*)

Writes Data to a register address “Reg” and Data_1 to address+1.

Parameters :

- **Reg** (*int*) – The address of the register for Data.
- **Data** (*int*) – The data to be written to the register.
- **Data_1** (*int*) – The data to be written to the register with address+1.

Return type :

None

RampCalc (*T1* , *T2* , *No_Q* , *Direction* , *CLH* , *CLL*)

Calculates the values of Gain, BP, Offset, NoQ, RedAngle, CLH, and CLL.

This function shows the output preview on the GUI. It uses `qmh_pyclient.PosSensor.PosSen.param_calc()` .

Parameters :

- **T1** (*int*) – Angle 1.
- **T2** (*int*) – Angle 2.
- **No_Q** (*int*) – No of quadrants.
- **Direction** (*str*) – Sets rising or falling ramp output. Valid inputs are: ‘cw’, ‘ccw’.
- **CLH** (*int*) – The desired high clamping level.
- **CLL** (*int*) – The desired low clamping level.

Returns :

A tuple containing:

- Gain (int): Output gain factor.
- BP (int): Output Breakpoint.
- Offset (int): Output offset value.
- NoQ (int): Number of quadrants.
- RedAngle (float): Reduced angle enabled.
- CLH (float): Clamping limit high.
- CLL (float): Clamping limit low.

Return type :

tuple

class qmh_pyclient.PosSensor. AS5171A (*Driver* , *** kwargs*)

Bases: `AS5171`

class qmh_pyclient.PosSensor. AS5171B (*Driver* , *** kwargs*)

Bases: `AS5171`

AS5270 Class

The AS5270 class is the base class used to define the sensor as AS5270.

class qmh_pyclient.PosSensor. AS5270 (*Driver* , *** kwargs*)

Bases: `PosSen`

OTP_1st_Addr *= 2*

Uses the methods from the base class `PosSen` , besides to the following methods:

REG_Read_OTP (*** cls*)

Reads the OTP Content. Optionally it can check if the acquired data matches what was written. TBD:Sensor.OTP{i:02x} will be also available i.e AS5270.Top.OTP02 with contents of 0x02 or Axxx.Top.OTP1e.

Returns :

If CheckContent=False:

- A tuple of:
  - A list containing the OTP values and
  - None.

If CheckContent=True:

- A tuple of:
  - A tuple of the OTP values
  - Boolean result of CheckContent.

Return type :

tuple

Read_Cordic (* *cls*)

Reads the CORDIC value for AS5270.

Parameters :

NoOfReads (*int*) – The number of readings.

Returns :

A list of two lists containing CORDIC value for Top/ Bottom in lsb.

Return type :

list of two list

Read_Mag_AGC (* *cls*)

Reads the Magnitude and the AGC registers.

Parameters :

NoOfReads (*int*) – The number of readings.

Returns :

A list containing two lists:

- Top Die List contains:
  - Mag (list int): list of Mag readings.
  - AGC (list of int): list of AGC readings.
- Bottom Die List contains:
  - Mag (list int): list of Mag readings.
  - AGC (list of int): list of AGC readings.

Return type :

list

class qmh_pyclient.PosSensor. AS5270A (*Driver* , *** kwargs*)

Bases: `AS5270`

class qmh_pyclient.PosSensor. AS5270B (*Driver* , *** kwargs*)

Bases: `AS5270`

AS5172 Class

The AS5172 class is used to define the sensor as AS5172.

class qmh_pyclient.PosSensor. AS5172 (*Driver* , *** kwargs*)

Bases: `PosSen`

OTP_1st_Addr *= 1*

** kwargs: Name > An ID to be assigned to the sensor that will appear in the Device GUI

RampCalc (*T1* , *T2* , *No_Q* , *Direction* , *PSI5_mode*)

Calculates the values of Gain, BP, Offset, NoQ, RedAngle, CLH, and CLL.

This function shows the output preview on the GUI. It uses `qmh_pyclient.PosSensor.PosSen.param_calc()` .

Parameters :

- **T1** (*int*) – Angle 1.
- **T2** (*int*) – Angle 2.
- **No_Q** (*int*) – No of quadrants
- **Direction** (*str*) – Sets rising or falling ramp output. Valid inputs are: ‘cw’, ‘ccw’
- **PSI5_mode** (*int*) – the PSI5 output resolution. Valid inputs are 12, 14

Returns :

A tuple containing:

- Gain (int): Output gain factor.
- BP (int): Output Breakpoint.
- Offset (int): Output offset value.
- NoQ (int): Number of quadrants.
- RedAngle (float): Reduced angle enabled.
- CLH (float): Clamping limit high.
- CLL (float): Clamping limit low.

Return type :

tuple

Write (*Reg* , *Data* , *Data_1 = 0*)

Writes Data to a register address “Reg” and Data_1 to address+1.

Parameters :

- **Reg** (*int*) – The address of the register for Data.
- **Data** (*int*) – The data to be written to the register.
- **Data_1** (*int*) – The data to be written to the register with address+1.

Return type :

None

SetCustomerParam (*Direction = 0* , *PSI5_14bit_angle = 0* , *PSI5_quad_info = 0* , *PSI5_16bit_frame = 0* , *Extended_init_phase = 0* , *Day = 0* , *Month = 0* , *Year = 0* , *SensorType = 0* , *SensorParameter = 0* , *SensorCodeManufacturer = 0* , *SensorCodeVehicle = 0* , *PSI5Mode = 0* , *Sync_discharge = 0* , *Init_phase_repetition = 0* , *PSI5_timeslot = 0* , *Init_phase_disable = 0* , *Extended_range_disable = 0* , *Customer_lock = 0* , *VendorID = 0*)

Configure customer-specific parameters for the AS5172 device.

Each parameter corresponds to specific configuration bits or registers as defined in the datasheet, allowing detailed customization of sensor behavior and PSI5 protocol options.

Parameters :

- **Direction** (*int*) – Direction setting (bit 2 at register 0x01).
- **PSI5_14bit_angle** (*int*) – Enable/disable 14-bit PSI5 angle mode (bit 3 at 0x01).
- **PSI5_quad_info** (*int*) – Quadrants info flag for PSI5 (bit 4 at 0x01).
- **PSI5_16bit_frame** (*int*) – Enable 16-bit PSI5 frame mode (bit 5 at 0x01).
- **Extended_init_phase** (*int*) – Extended initialization phase flag (bit 7 at 0x01).
- **Day** (*int*) – Day value (bits 0:4 at 0x0D).
- **Month** (*int*) – Month value (bits 5:7 at 0x0D plus bit 0 at 0x0C).

- **Year** (*int*) – Year value (bits 1:7 at 0x0C).
- **SensorType** (*int*) – Sensor type identifier (bits 0:3 at 0x0E).
- **SensorParameter** (*int*) – Sensor parameter bits (bits 0:3 at 0x0F plus bits 4:7 at 0x0E).
- **SensorCodeManufacturer** (*int*) – Manufacturer code (bits 0:3 at 0x10 plus bits 4:7 at 0x0F).
- **SensorCodeVehicle** (*int*) – Vehicle code (bits 4:7 at 0x10).
- **PSI5Mode** (*int*) – PSI5 protocol mode setting (bits 0:7 at 0x11).
- **Sync_discharge** (*int*) – Synchronization discharge flag (bit 0 at 0x12).
- **Init_phase_repetition** (*int*) – Initialization phase repetition count (bits 1:2 at 0x12).
- **PSI5_timeslot** (*int*) – PSI5 timeslot setting (bits 3:4 at 0x12).
- **Init_phase_disable** (*int*) – Disable initialization phase flag (bit 5 at 0x12).
- **Extended_range_disable** (*int*) – Disable extended range flag (bit 3 at 0x1D).
- **Customer_lock** (*int*) – Customer lock flag (bit 7 at 0x1D).
- **VendorID** (*int*) – Vendor identifier (bits 0:7 at 0x1E).

Returns :

None

class qmh_pyclient.PosSensor. AS5172A (*Driver* , *** kwargs*)

Bases: `AS5172`

P10P_5003L_Timeslot1 *= 0*
P10P_5003L_Timeslot2 *= 1*
P10P_5003L_Timeslot3 *= 2*

P10P_5003L_Discharge_Timeslot2 *= 3*
P10P_5003L_Discharge_Timeslot3 *= 4*
P10P_5003L_HighResMode_Timeslot1_2 *= 5*
P10P_5003L_HighResMode_Timeslot2_3 *= 6*
P10P_5003L_Discharge_HighResMode_Timeslot2_3 *= 7*
A10P_5001L *= 8*
A10P_2501L *= 9*
A10P_5001L_HighResMode *= 10*
A10P_2501L_HighResMode *= 11*
P20CRC_5001L_Timeslot1 *= 12*
P20CRC_5002L_Timeslot2 *= 13*
P20CRC_5002H_Timeslot1 *= 14*
P20CRC_5002H_Timeslot2 *= 15*
P20CRC_5003H_Timeslot1 *= 16*
P20CRC_5003H_Timeslot2 *= 17*
P20CRC_5003H_Timeslot3 *= 18*
P16CRC_5003H_Timeslot1 *= 19*
P16CRC_5003H_Timeslot2 *= 20*
P16CRC_5003H_Timeslot3 *= 21*
A20CRC_2001H *= 22*
A20CRC_5001H *= 23*
A20CRC_2002H *= 24*
A20CRC_3001L *= 25*

SetPSI5Mode (*PSI5Mode* , *LowPower* , *K_times = 0* , *RollingCounter = 0*)

Configure the PSI5 communication mode related parameters for AS5172EA/B.

Parameters :

- **PSI5Mode** (*int*) – PSI5 mode setting value.
- **LowPower** (*int*) – Low power mode enable flag.
- **ErrBit** (*int*) – Error bit enable flag (default is 1).
- **K_times** (*int*) – K-times configuration parameter (default is 0).
- **P16_new_init** (*int*) – P16 new initialization flag (default is 0).
- **RollingCounter** (*int*) – Rolling counter enable flag (default is 0).

Returns :

None

Read_PSI5 (*PSI5Mode* , *LowPower = 0* , *Mode_14bit = 0* , *NoSamples = 1*) → tuple

Read PSI5 sensor data according to the specified mode and options.

Parameters :

- **PSI5Mode** (*int*) – PSI5 communication mode setting.
- **LowPower** (*int*) – Enable low power mode if set to 1 (default is 0).
- **Mode_14bit** (*int*) – Enable 14-bit mode if set to 1 (default is 0).
- **NoSamples** (*int*) – Number of samples to read (default is 1).

Returns :

A tuple containing the PSI5 sensor data samples read.

Return type :

tuple

class qmh_pyclient.PosSensor. AS5172E (*Driver* , *** kwargs*)

Bases: `AS5172`

P10P_5003L_Timeslot1 *= 0*
P10P_5003L_Timeslot2 *= 1*
P10P_5003L_Timeslot3 *= 2*
P10P_5003L_Discharge_Timeslot2 *= 3*
P10P_5003L_Discharge_Timeslot3 *= 4*
P10P_5003L_HighResMode_Timeslot1_2 *= 5*
P10P_5003L_HighResMode_Timeslot2_3 *= 6*
P10P_5003L_Discharge_HighResMode_Timeslot2_3 *= 7*
A10P_5001L *= 8*
A10P_2501L *= 9*
A10P_5001L_HighResMode *= 10*
A10P_2501L_HighResMode *= 11*
P20CRC_5001L_Timeslot1 *= 12*
P20CRC_5002L_Timeslot2 *= 13*
P20CRC_5002L_AngleandVelocity *= 14*

P20CRC_5002H_Timeslot1 *= 15*
P20CRC_5002H_Timeslot2 *= 16*
P20CRC_5002H_AngleandVelocity *= 17*
P20CRC_5003H_Timeslot1 *= 18*
P20CRC_5003H_Timeslot2 *= 19*
P20CRC_5003H_Timeslot3 *= 20*
P20CRC_5003H_AngleandVelocity_Timeslot1and2 *= 21*
P20CRC_5003H_AngleandVelocity_Timeslot2and3 *= 22*
P20CRC_5003H_AngleandVelocity_Timeslot1and3 *= 23*
P16CRC_5003H_Timeslot1 *= 24*
P16CRC_5003H_Timeslot2 *= 25*
P16CRC_5003H_Timeslot3 *= 26*
P16CRC_5003H_AngleandVelocity_Timeslot1and2 *= 27*
P16CRC_5003H_AngleandVelocity_Timeslot2and3 *= 28*
P16CRC_5003H_AngleandVelocity_Timeslot1and3 *= 29*
P16CRC_10004H_AngleandVelocity_Timeslot1and2 *= 30*
P16CRC_10004H_AngleandVelocity_Timeslot3and4 *= 31*
A20CRC_2001H *= 32*
A20CRC_5001H *= 33*
A20CRC_2002H *= 34*
A20CRC_3001L *= 35*

SetPSI5Mode (*PSI5Mode* , *LowPower* , *ErrBit = 1* , *K_times = 0* , *P16_new_init = 0* , *RollingCounter = 0*)

Configure the PSI5 communication mode related parameters for AS5172E/F.

Parameters :

- **PSI5Mode** (*int*) – PSI5 mode setting value.
- **LowPower** (*int*) – Low power mode enable flag.
- **ErrBit** (*int*) – Error bit enable flag (default is 1).
- **K_times** (*int*) – K-times configuration parameter (default is 0).
- **P16_new_init** (*int*) – P16 new initialization flag (default is 0).
- **RollingCounter** (*int*) – Rolling counter enable flag (default is 0).

Returns :

None

Read_PSI5 (*PSI5Mode* , *LowPower = 0* , *ErrBit = 1* , *Mode_14bit = 0* , *NoSamples = 1* , *NoQ = 1*) → tuple

Read PSI5 sensor data according to the specified configuration.

Parameters :

- **PSI5Mode** (*int*) – PSI5 communication mode setting.
- **LowPower** (*int*) – Enable low power mode if set to 1 (default is 0).
- **ErrBit** (*int*) – Enable error bit handling if set to 1 (default is 1).
- **Mode_14bit** (*int*) – Use 14-bit mode if set to 1 (default is 0).
- **NoSamples** (*int*) – Number of samples to read (default is 1).
- **NoQ** (*int*) – Number of Quadrants (default is 1).

Returns :

A tuple containing the PSI5 sensor data samples read.

Return type :

tuple

AS5173 Class

The AS5173 class is used to define the sensor as AS5173.

class qmh_pyclient.PosSensor. AS5173 (*Driver* , *** kwargs*)

Bases: `AS5172E`

param_calc (*T1* , *T2* , *Direction* , *No_Q* , *CLH* , *CLL* , *Mode*)

Calculates the values of Gain, BP, Offset, NoQ, RedAngle, CLH, CLL. Works for AS5173.

Parameters :

- **T1** (*int*) – Angle 1.
- **T2** (*int*) – Angle 2.
- **Direction** (*int*) – Input direction.
- **No_Q** (*int*) – Number of quadrants.
- **CLH** (*int*) – Clamping limit high.
- **CLL** (*int*) – Clamping limit low.
- **Mode** (*int*) – Sensor mode.

Returns :

A tuple containing:

- Gain (int): Output gain factor.
- BP (int): Output breakpoint.
- Offset (int): Output offset value.
- NoQ (int): Number of quadrants.
- RedAngle (float): Reduced angle enabled.
- CLH (float): Clamping limit high.
- CLL (float): Clamping limit low.
- Direction (int): Returned direction.
- Mode (int): Returned mode.

Return type :

tuple

API Usage

This section explains how to use the API.

Getting Started

Before calling any function, the `qmh_pyclient.QMH.QMH.ServerConn()` must be called first.

Example:

```
from qmh_pyclient import *

QMH.ServerConn(serverHost='localhost',serverPort=50007)
Prog=SD4Yv02(REF='SD4Yv02',Baudrate=115200,V_PSI5_h=11,V_PSI5_l=5.5)
# REF is the com port number
SENSOR=AS5172E(Driver=Prog,Name="AS5172E")
```

Valid Sensor Names

Some base classes should not be called directly, hence the following classes can be directly initiated:

- `AS5171A`
- `AS5171B`
- `AS5270A`
- `AS5270B`
- `AS5172A`
- `AS5172E`
- `AS5173`

Coding examples

AS5171

The following code can be used to program an `AS5171A` / `AS5171B` .

```
from qmh_pyclient import *
import numpy

QMH.ServerConn(serverHost='localhost',serverPort=50007)

## Settings ##
Burn=False  ## Use burn with caution is it can happen only once ##
Custlock=False ## If Burn is True it is better to custom lock the part, so it will start in functional mode ##

# Prog=SD4Yv02(REF='SD4Yv02',Baudrate=115200)
# Prog=SD4Yv01(REF='SD4Yv02')
# Prog=UART_PCB(REF='com21',Baudrate=9600)
Prog=Sim()
SENSOR=AS5171A(Driver=Prog,Name="AS5171A")
OTP_dict,_=SENSOR.REG_Read_OTP()
print("Initial OTP::")
print(OTP_dict)

MAG,AGC=SENSOR.Read_Mag_AGC(NoOfReads=5)
print(f"MAG={numpy.average(MAG)}, AGC={numpy.average(AGC)}") # logs Mag and Agc or use them to check if magnet in range/exist.
T1=int(numpy.average(SENSOR.Read_Cordic(5)))
## The following code calculates the Gain,BP and other parameters need for defining the output ##
Gain, BP, Offset, NoQ, RedAngle, CLH, CLL=SENSOR.RampCalc(T1=round(T1),T2=T1+1,Direction='cw',No_Q=1,CLH=0,CLL=0)


print(f"Gain={Gain}, BP={BP}, Offset={Offset}, NoQ={NoQ}, RedAngle={RedAngle}, CLH={CLH}, CLL={CLL}")

CLH_lo=CLH&0x0FF
CLH_hi=(CLH&0xF00)>>8 # Shift Left by 8 (multiply)
SENSOR.OTP14|=CLH_lo
SENSOR.OTP15|=CLH_hi

CLL_lo=(CLL&0x00F)<<4 # Shift Left by 4 (multiply)
CLL_hi=(CLL&0xFF0)>>4 # Shift Right by 4 (divide)
SENSOR.OTP15|=CLL_lo
SENSOR.OTP16|=CLL_hi


Offset_lo= Offset&0x0000FF
Offset_mi=(Offset&0x00FF00)>>8 # Shift Right by 8 (divide)
Offset_hi=(Offset&0xF0000)>>16 # Shift Right by 16 (divide)
SENSOR.OTP17|=Offset_lo
SENSOR.OTP18|=Offset_mi
SENSOR.OTP19|=Offset_hi
```

```
45
46
47
48 Gain_lo=(Gain & 0x00000F)<<4  # Shift Left by 4 (multiply) as it
is stored at [4:7]
49 Gain_mi=(Gain & 0x000FF0)>>4  # Shift Right by 4 (divide)
50 Gain_hi=(Gain & 0x01F000)>>12 # Shift Right by 12 (divide)
51 SENSOR.OTP19|=Gain_lo
52 SENSOR.OTP1a|=Gain_mi
53 SENSOR.OTP1b|=Gain_hi
54
55
56
57 BP_lo =(BP & 0x0007)<<5  # Shift Left by 5 (multiply) as it is
stored at [5:7]
58 BP_mi =(BP & 0x07F8)>>3  # Shift Right by 3 (divide)
59 BP_hi =(BP & 0x3800)>>11 # Shift Right by 11 (divide)
60 SENSOR.OTP1b|=BP_lo
61 SENSOR.OTP1c|=BP_mi
62 SENSOR.OTP1d|=BP_hi
63
64 if Burn==True:
65     if Custlock == True:
66         SENSOR.OTP1d|=0x80
67
68 ## Other settings can be changed as described in the datasheet of
the product ##
69
70 OTP_Array = [getattr(SENSOR, f'OTP{format(i, "02x")}', None) for
i in range(0x1,0x1E)]
71 SENSOR.REG_Write_Array(OTP_Array) ## Writing the OTP with
signature calculated ##
72 _1E=SENSOR.Signature_Calc() ## Signature calculation ##
73 SENSOR.Write(0x1E,_1E<<8)
74 time.sleep(5)
75 OTP_dict_final,_=SENSOR.REG_Read_OTP() ## Final read to check if
the data was written correctly ##
76 ## Checks if the data written is equal to the last write step ##
77 # OTP_Array.append(_1E)
78 OTP_Array[-1]=_1E
79
80 if OTP_dict_final==OTP_Array:
81     print("Data was written")
82 else:
83     raise NameError("The written data does not match")
84 print(OTP_dict_final)
85 if Burn==True:
86     if SENSOR.Fuse():
87         print("Data was fused")
88     else:
89         raise NameError("The fused data does not match")
90 else:
```

```
    SENSOR.Pass2Func()

SENSOR.CloseDev()
```

AS5172

The following code can be used to program an `AS5172E` .

```
from qmh_pyclient import *
import numpy

QMH.ServerConn(serverHost='localhost',serverPort=443)


#
Prog=SD4Yv02(REF='SD4Yv02',Baudrate=115200,V_PSI5_h=11,V_PSI5_l=5.5)
# Prog=PSI5_PCBv01(REF='com21',Baudrate=38400)
Prog=Sim()
SENSOR=AS5172E(Driver=Prog,Name="AS5172E")

MAG,AGC=SENSOR.Read_Mag_AGC(NoOfReads=5)
print(f"MAG={numpy.average(MAG)}, AGC={numpy.average(AGC)}") #
logs Mag and Agc or use them to check if magnet in range/exist.
T1=int(numpy.average(SENSOR.Read_Cordic(5)))

## The following code calculates the Gain,BP and other parameters
need for defining the output ##

Gain, BP, Offset, Quad, RedAngle, CLH,
CLL=SENSOR.RampCalc(T1=round(T1),T2=T1+1,Direction='cw',No_Q=1,PSI5_mode=14)

Velocity_Filter=0
velocity_range=1250
ExtRangeDis=False
RedAngle=False
FilterCFG=0
InitPhaseDis=True
DirBit=False
BitMode14=False
PSI5Mode=SENSOR.P16CRC_5003H_Timeslot1
LowPower=0
Pass2Func=True

OTP,_=SENSOR.REG_Read_OTP()
# MSB >> LSB #

## DirectionBit ##
# 0x01[2]
# 0000 0X00
if DirBit==False:
```

```
    SENSOR.OTP01&=0xFB
elif DirBit==True:
    SENSOR.OTP01|=0x04

## VelocityRange 0x01.[0] & 0x01[6]
    # 0x01[0]
    # 0x01[6]
    # 0X00 000X
if velocity_range==1250:
    SENSOR.OTP01&=0xBE
elif velocity_range==4608:
    SENSOR.OTP01&=0xBE
    SENSOR.OTP01|=0x01
elif velocity_range==5000:
    SENSOR.OTP01&=0xBE
    SENSOR.OTP01|=0x40
else:
    raise NameError(f'velocity_range
({velocity_range}) is Out of range') # RAISE An Error


## 14BitMode ##
# 0x01[3]
# 0000 X000
if BitMode14==False:
    SENSOR.OTP01&=0xF7
elif BitMode14==True:
        SENSOR.OTP01|=8

## InitPhaseDisable ##
# 0x12[5]
# 00X0 0000
if InitPhaseDis==False:
    SENSOR.OTP12&=0xDF
elif InitPhaseDis==True:
    SENSOR.OTP12|=0x20

## Quad ##
    # 0x13[3:4]
    # 000x x000
if (Quad>=0) and (Quad <= 3):
    Quad=Quad <<3 ## Bit shift
    SENSOR.OTP13|=Quad
else:
    raise NameError(f'Quad ({Quad}) is Out of range') # RAISE An
Error


## Velocity Filter ##
    # 0x13[5:7]
    # XXX0 0000
if (Velocity_Filter>=0) and (Velocity_Filter <= 7):
```

```
 88     Velocity_Filter=Velocity_Filter<<5 # Shift Left by 5
(multiply) as it is stored at [5:7]
 89     SENSOR.OTP13|=Velocity_Filter
 90 else:
 91     raise NameError(f'Velocity_Filter ({Velocity_Filter}) is Out
of range') # RAISE An Error
 92
 93 ## CLH ##
 94     # CLH[00:07] > 0x14[0:7]
 95     # CLH[08:11] > 0x15[0:3]
 96     # xxxx xxxx
 97     # 0000 xxxx
 98 if (CLH>=0) and (CLH <= (2**12)-1):
 99     CLH_lo= CLH & 0x00FF
100     CLH_hi=(CLH & 0x0F00)>>8 # Shift Right by 8 (divide)
101     SENSOR.OTP14|=CLH_lo
102     SENSOR.OTP15|=CLH_hi
103 else:
104     raise NameError(f'CLH ({CLH}) is Out of range') # RAISE An
Error
105
106 ## CLL ##
107     # CLL[00:03] > 0x15[4:7]
108     # CLL[04:11] > 0x16[0:3]
109     # xxxx 0000
110     # xxxx xxxx
111 if (CLL>=0) and (CLL <= (2**12)-1):
112     CLL_lo=(CLL & 0x000F)<<4 # Shift Left by 4 (multiply) as it
is stored at [4:7]
113     CLL_hi=(CLL & 0x0FF0)>>4 # Shift Right by 4 (divide)
114     SENSOR.OTP15=SENSOR.OTP15|CLL_lo
115     SENSOR.OTP16=SENSOR.OTP16|CLL_hi
116 else:
117     raise NameError(f'CLL ({CLL}) is Out of range') # RAISE An
Error
118
119 ## Offset ##
120     # Offset[00:07] > 0x17[0:7]
121     # Offset[08:15] > 0x18[0:7]
122     # Offset[16:19] > 0x19[0:3]
123     # XXXX XXXX
124     # XXXX XXXX
125     # 0000 XXXX
126 if (Offset>=0) and (Offset <= (2**20)-1):
127     Offset_lo= Offset & 0x0000FF
128     Offset_mi=(Offset & 0x00FF00)>>8  # Shift Right by 8
(divide)
129     Offset_hi=(Offset & 0x0F0000)>>16 # Shift Right by 8
(divide)
130     SENSOR.OTP17|=Offset_lo
131     SENSOR.OTP18|=Offset_mi
132     SENSOR.OTP19|=Offset_hi
```

```
133
134 else:
135     raise NameError(f'Offset ({Offset}) is Out of range')
# RAISE An Error
136
137 ## Gain ##
138     # Gain[00:03] > 0x19[4:7]
139     # Gain[04:11] > 0x1A[0:7]
140     # Gain[12:16] > 0x1B[0:3]
141     # XXXX XXXX
142     # XXXX XXXX
143     # 000X XXXX
144 if -(2**16) <= Gain <= (2**16 - 1):
145     Gain_lo=(Gain & 0x00000F)<<4
# Shift Left by 4 (multiply) as it is stored at [4:7]
146     Gain_mi=(Gain & 0x000FF0)>>4  # Shift Right by 4 (divide)
147     Gain_hi=(Gain & 0x01F000)>>12 # Shift Right by 12 (divide)
148     SENSOR.OTP19|=Gain_lo
149     SENSOR.OTP1a|=Gain_mi
150     SENSOR.OTP1b|=Gain_hi
151
152 else:
153     raise NameError(f'Gain ({Gain}) is Out of range') # RAISE An
Error
154
155 ## BP ##
156     # BP[00:02] > 0x1B[5:7]
157     # BP[03:10] > 0x1C[0:7]
158     # BP[11:13] > 0x1D[0:2]
159     # XXX0 0000
160     # XXXX XXXX
161     # 0000 0XXX
162 if (BP>=0) and (BP <= (2**14)-1):
163     BP_lo =(BP & 0x0007)<<5  # Shift Left by 5 (multiply) as it
is stored at [5:7]
164     BP_mi =(BP & 0x07F8)>>3  # Shift Right by 3 (divide)
165     BP_hi =(BP & 0x3800)>>11 # Shift Right by 11 (divide)
166     SENSOR.OTP1b|=BP_lo
167     SENSOR.OTP1c|=BP_mi
168     SENSOR.OTP1d|=BP_hi
169 else:
170     raise NameError(f'BP ({BP}) is Out of range') # RAISE An
Error
171
172 ## ExtRangeDis ##
173     # 0x1D[3]
174 if ExtRangeDis==False:
175     SENSOR.OTP1d&=0xF7
176 elif ExtRangeDis==True:
177     SENSOR.OTP1d|=0x08
178
179 ## RedAngle ##
```

```
    # 0x1D[4]
if RedAngle==False:
    SENSOR.OTP1d&=0xEF
elif RedAngle==True:
    SENSOR.OTP1d|=0x10


OTP_Array=[]
i=1
for reg in OTP:
    _=eval(f'SENSOR.OTP{i:02x}')
    if (_>=0) and (_<=255):
        OTP_Array.append(_)
        i=i+1
    else:
        raise NameError(f'({AS5172}.OTP{i:02x}=({_}) is Out of range') # RAISE An Error
print(f"OTP_Array={OTP_Array}")
SENSOR.REG_Write_Array(OTP_Array)
SENSOR.REG_Read_OTP(CheckContent=True)

SENSOR.SetPSI5Mode(PSI5Mode,LowPower=LowPower)

SENSOR.Signature_Calc()
SENSOR.REG_Write_OTP()
time.sleep(.5)
result=SENSOR.REG_Read_OTP(CheckContent=True)
OTP_Final=result[0]
Match=result[1]
if Match==True:
    pass
else:
    raise Exception("OPT Content Do not match")
OTP_Final_str=';'.join(str(hex(_)) for _ in OTP_Final)
Report.Add('OTP_Final',OTP_Final_str,1)
Print(f"OTP_Final={OTP_Final}")
if Pass2Func==True:
    SENSOR.Pass2Func()

SENSOR.CloseDev()
```

Folder Structure

The installer will install files at **C:\Infineon\POS** as follows

```
└──POS
    ├──App
    │   ├──Position Sensor.exe
    ├──Settings
    │   ├──Settings.ini
    ├──Updater
    └──Venv
        ├──.venv
        │
```

App : Main application location

Any folders/files saved here by the user will be deleted during the update process.

Settings\Settings.ini : User settings

User settings file. More details are in User Settings section.

Venv : Python created virtual environment for using the API.

INI Configuration Reference

This document describes all configuration sections and keys used in the **Settings\Settings.ini** INI file.

TCP_Server

```
[TCP_Server]
port = "50007"
```

Keys

port
: The default port for Python API. It can be changed if not available. Default: `"50007"` .

AS5171_Analog

```
[AS5171_Analog]
GuiCTRLs = "LoadFromFile"
Path = ""
```

Keys

GuiCTRLs
: GUI control loading mode. `"LoadDefault"` means control definitions are always default wheb loading the device GUI. `"LoadFromFile"` means control definitions are read from a file that is stored in **Settings** . `"LoadFromPath"` means control definitions are read from a file that is specified in Path.

Path
: File system path to the GUI control definition file. Empty string means no file is assigned.

AS5171_Digital

```
[AS5171_Digital]
GuiCTRLs = "LoadFromFile"
Path = ""
```

Keys

GuiCTRLs
: Same behavior as before.

Path
: Same behavior as before.

AS5172

```
[AS5172]
GuiCTRLs = "LoadFromFile"
Path = ""
```

Keys

GuiCTRLs
: Same behavior as before.

Path
: Same behavior as before.

AS5173

```
[AS5173]
GuiCTRLs = "LoadFromFile"
Path = ""
```

Keys

GuiCTRLs
: Same behavior as before.

Path

Same behavior as before.

AS5270_Analog

```
[AS5270_Analog]
GuiCTRLs = "LoadFromFile"
Path = ""
```

Keys

GuiCTRLs

Same behavior as before.

Path

Same behavior as before.

AS5270_Digital

```
[AS5270_Digital]
GuiCTRLs = "LoadFromFile"
Path = ""
```

Keys

GuiCTRLs

Same behavior as before.

Path

Same behavior as before.

Indices and tables

- Index
- Module Index
- Search Page