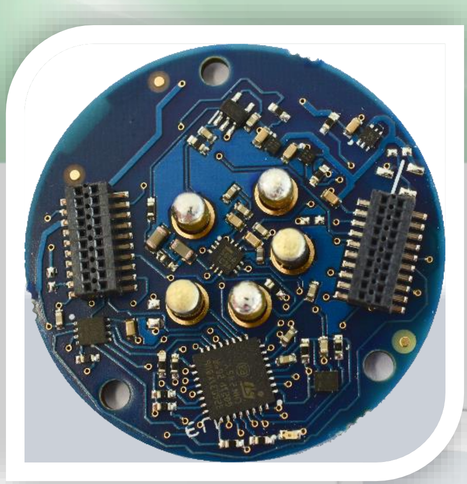
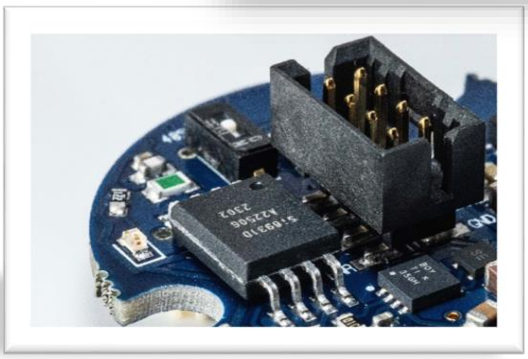
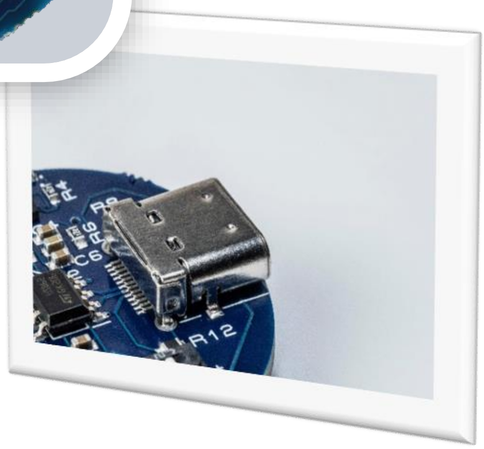
Integration Kits User Manual

Version 1.0

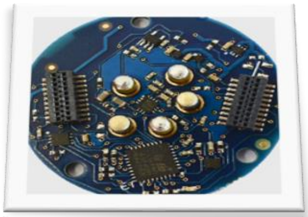






Quick Start Guide

Box Contents



ION CONNECT Integration Kit

- ION CONNECT PCB
- ION PROGRAM PCB
- ION GAS HOOD
- Accessories (Screws, Cable, Allen key and O-rings)

ION TRANSMIT Integration Kit

- ION CONNECT PCB
- ION TRANSMIT PCB
- ION PROGRAM PCB
- ION GAS HOOD
- Accessories (Screws, Cables, Allen key and O-rings)

Individual PCBs and Gas hoods are also available for purchase based on need.

ION PC

Download and install the ION PC software from the product pages:
https://ionsense.com/ion-connect
https://ionsense.com/ion-transmit

ION TRANSMIT

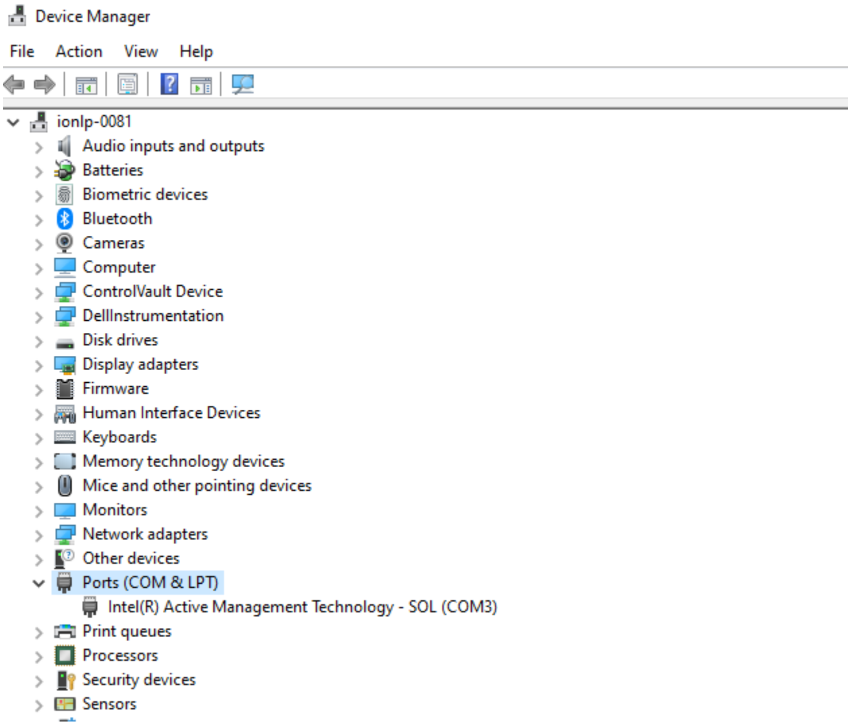
1. Connect the ION CONNECT PCB and ION TRANSMIT PCBs together
2. Connect the wire to the ION TRANSMIT PCB
3. Connect the wire to an RS485 to USB module
4. Connect to a power supply
5. Connect to a USB
6. Supply power

ION PROGRAM

1. Connect the ION CONNECT PCB and ION PROGRAM PCBs together
2. Connect the USB C to USB A cable to PC and ION PROGRAM PCB

ION PC

1. Once connected to the PC check the COM port which is being used in the Device Manager.



2. Open ION PC
3. Click “Select Instrument”
4. Click “ION CONNECT”
5. Select the COM Port, Baud rate and Slave ID. Factory settings for these will be 9600 Baud rate and Slave ID 1.
6. Click Connect and you will see Settings, Calibration and Data logs appear in the side bar.

Calibration

1. Select Calibration in the side bar.
2. Select the New Calibration button in the top bar of this page. A side panel will appear.
3. Apply zero air to the sensor. Once the readings stabilize click SET ZERO.
4. Type the Response Factor (RF) of the calibration gas and click SET CAL GAS RF.
5. Type the concentration of the Span 1 gas and click SET SPAN 1.
6. Apply Span gas. Once the readings stabilize click the SPAN 1 RESPONSE.
7. Repeat steps 5 and 6 for up to four different span gas calibration points if doing multi point calibration.
8. Once you have applied all your span gas calibration points click END CALIBRATION.
9. Click the button in the top bar to write the calibration to the memory.

Technical Support

If you require technical support with any aspect of the ION SENSE® Integration kits, please contact sensors@ionsense.com

Contents

Symbols

WARNING!

USED TO INDICATE DANGER WARNINGS WHERE THERE IS A RISK OF INJURY OR DEATH.

WARNING! - DANGER OF ELECTRIC SHOCK

Used to indicate danger warnings where there is a risk of injury or death from electric shock.

CAUTION

Used to indicate a caution where there is a risk of damage to equipment.

PROHIBITED ACTION

Used to indicate actions that are NOT PERMITTED, e.g. 'You Must NEVER'.

INFORMATION

Important information or useful hints about usage.

Recycling and Disposal

RECYCLING

Recycle all packaging.

WEEE REGULATIONS

Ensure that waste electrical equipment is disposed of correctly.

Statements

Validity of this Manual

This User Manual gives information and procedures for the firmware and software versions shown in the manual update log later in this manual.

If you have different versions of firmware, please get in touch to obtain the correct firmware version.

Responsibility for Correct Use

ION SENSE® accepts no responsibility for incorrect adjustments, configurations or installations that cause harm or damage to persons or property.

Use the equipment in accordance with this manual, and compliance with local safety standards.

Reduced performance of gas detection might not be obvious, so equipment must be inspected and maintained regularly. ION SENSE® recommends:

- you use a schedule of regular checks to ensure it performs within calibration limits.
- you keep a record of calibration check data.

ION SENSE® integration boards are not authorised for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-critical applications include, without limitation, life support devices and systems, equipment or systems for operation in Hazardous Areas or Zones. ION SENSE® products are neither designed nor intended for use in military applications. The customer acknowledges and agrees that any such use of ION SENSE® products is solely at the customer's risk and that the customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

Warnings

1. Read and understand this manual fully before you install or operate any of the Integration board.
2. For safety, the integration boards must only be installed by qualified personnel.
3. All electrical work must only be carried out by competent persons.
4. Substitution of components can result in unsafe conditions and will invalidate the warranty.
5. It is the responsibility of the user to ensure that PCBs are adequately protected from sources of EMI (Electro-magnetic index).
6. It is the responsibility of the user to ensure that PCBs are adequately protected from unsafe voltage conditions that could be transmitted into any connected equipment or systems.
7. It is the user's responsibility to assess risk when working with toxic and flammable compounds. Relevant protective equipment must be used.

Quality Assurance

The integration boards (ION CONNECT PCB, ION TRANSMIT PCB and ION PROGRAM PCB) are manufactured using business systems complying with the ISO 9001 standard, which ensures that the equipment is:

- Designed and assembled reproducibly, from traceable components.
- Functionally tested to the stated standards before it leaves the factory.

Disposal

Dispose of the integration boards (ION CONNECT PCB, ION TRANSMIT PCB and ION PROGRAM PCB) and its components in accordance with all local and national safety and environmental requirements. This includes the European WEEE (Waste Electrical and Electronic Equipment) directive. ION Science® offers a take-back service. Please contact us for more information.

Legal Notice

Whilst every attempt is made to ensure the accuracy of the information contained in this manual, ION Science® accepts no liability for errors or omissions, or any consequences deriving from the use of information contained herein. It is provided "as is" and without any representation, term, condition or warranty of any kind, either expressed or implied. To the extent permitted by law, ION Science shall not be liable to any person or entity for any loss or damage which may arise from the use of this manual. We reserve the right at any time and without any notice to remove, amend or vary any of the content which appears herein.

Warranty

ION Science® warrants that its products will conform to the Specifications. This warranty lasts for one (1) year from the date of the sale.

ION Science® shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the user, including improper installation or testing, or for any products that have been altered or modified in any way by the user. Moreover, ION Science shall not be liable for any defects that result from the customer's design, specifications, installations or instructions or incorrect interfacing for such products. Testing and other quality control techniques are used to the extent ION Science deems necessary.

If any ION Science® product fails to conform to the warranty set forth above, ION Science sole liability shall be to replace such products. ION Science liability shall be limited to products that are determined by ION Science not to conform to such warranty. If ION Science elects to replace such products, ION Science shall be given a reasonable time to provide replacements. Replaced products shall be warranted for a new full warranty period. ION Science shall not be liable for any freight costs incurred.

In no event shall ION Science® be liable to the customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether ION Science has been advised of the possibility of such damages. This indemnity will survive the termination of the warranty period.

Contact Details

UK Head Office
ION Science Ltd
The Hive, Butts Lane
Fowlmere
Cambridge
SG8 7SL
UNITED KINGDOM

Tel: +44 (0)1763 208503
Fax: +44 (0) 1763 208814
Email: sensors@ionsense.com
Web: www.ionsense.com

USA Corporate HQ
ION Science Inc
4153 Bluebonnet Drive
Stafford
TX 77477
USA

Tel : +1 (877) 864 7710
Email : info@ionscienceusa.com
Web : www.ionscienceusa.com

India Office
ION Science India Pvt. Ltd
#1-90/B/C/3/1, G-10 Charmy,
Ganesh Nilayam, Vittal Rao Nagar
Image Hospital Lane, Madhapur,
Hyderabad – 500 081
Telangana State
INDIA

Tel: +91 40 48536129
Email: kschari@ionscience.com
Web: www.ionscience-india.com

Italy Office
ION Science Italia
Via Emilia 51/c
40011 Anzola Emilia
Bologna
ITALY

Tel: +39 051 0561850
Fax: +39 051 0561851
Email: info@ionscience.it
Web: www.ionscience.it

China Office
ION Science China Ltd
1101, Building B
Far East International Plaza
No. 317 Xiaxia Road
Shanghai
CHINA

Tel : +86 21 52545988
Fax : +86 21 52545986
Email : info@ionscience.cn
Web : ionscience.cn

Introduction to ION SENSE® Integration Kits

The ION SENSE® integration boards allow users to rapidly evaluate and seamlessly integrate ION SENSE® PID sensors. ION CONNECT PCB adds smart functionality to ION PIDS, including calibration, alarms, adjustable response factors and allows communication via I2C or UART. ION TRANSMIT PCB pairs with CONNECT PCB to allow long range communication via RS485 MODBUS RTU or 4-20mA. PROGRAM PCB allows PC configuration and data logging via ION PC software.

ION CONNECT PCB

ION CONNECT PCB connects directly with the sensor. This 32mm diameter PCB converts the analogue sensors analogue signal to digital. The PCB has board to board connectors which allow the user to communicate using UART and I2C. ION CONNECT is used as a MODBUS RTU slave device. The registers include calibration points, alarms, adjustable response factor, dead band, scaled outputs, duty cycle and error states. For more information on the MODBUS registers, see the Modbus Interface section in this manual. ION CONNECT has been designed to work in both diffuse and impressed flow applications. There is an ION Gas Hood designed to deliver impressed flow to the ION SENSE® PID sensors (MiniPID 2 & ION SENSE® PID).

ION TRANSMIT PCB

ION TRANSMIT PCB connects to ION CONNECT PCB and allows the user to transmit all the ION CONNECT PCB features over long-range protocols. This PCB gives the user access to 4-20mA and RS485 in a 32mm diameter PCB.

ION PROGRAM PCB

ION PROGRAM PCB connects to ION CONNECT PCB and allows the user to connect directly to the PC using a USB C to USB A cable. This gives the user quick access to the ION PC software, giving them the ability to program and update the firmware on ION CONNECT PCB.

Technical Specification

ION CONNECT PCB

Dimensions	32mm diameter
Weight	4g
Nominal Voltage	3.3 – 5.5 VDC
Supply Connectors	Samtec CLE-110-01-G-DV-P-TR
Operating Humidity:	0 – 99 RH% (non-condensing)
Operating Temperature	-20°C to +60°C

ION TRANSMIT PCB

Dimensions	32mm diameter – 12.92mm height
Weight	4g
Nominal Voltage	12V to 24VDC

Typical Power	At 24V 25mA unloaded, 31mA with Sensor At 12V 31mA unloaded, 44mA with Sensor
Supply Cable	MOLEX 2185101081 (MILLI-GRID DR R-S 8CKT 100MM DIS)
Maximum Contact Load	2A
Operating Humidity:	0 – 99 RH% (non-condensing)
Operating Temperature	-20°C to +60°C
Mating ION CONNECT Connectors	Samtec FTE-110-01-G-DV-ES-A-P-TR

ION GAS HOOD

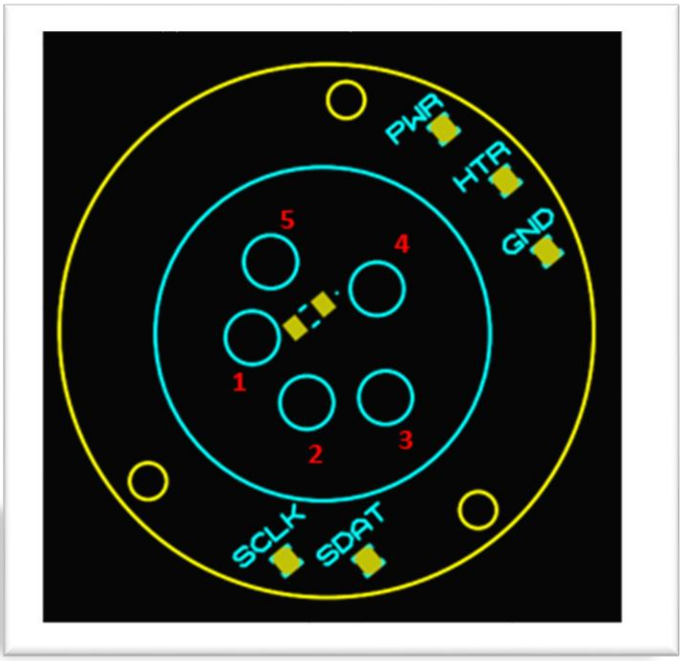
Dimensions	32mm diameter x 26mm
Pipe Connection	1/16" OD barb push fit 1/16" ID/1/8" OD fluorinated tubing recommended
Seal material	Viton
Flow Rate (max)	500ml/min
Pressure (max)	-15mBar

Compatible Sensors

	Part Number
MiniPID 2 PPM (3.0 – 3.2 V)	MP3SMLLAU2
MiniPID 2 PPM (3.2 – 3.6 V)	MP3SMLLBU2
MiniPID 2 PPM WR (3.0 – 3.2 V)	MP3SWMLLAU2
MiniPID 2 PPM WR (3.2 – 3.6 V)	MP3SWMLLBU2
MiniPID 2 PPB (3.0 – 3.2 V)	MP3SBLBAU2
MiniPID 2 PPB (3.2 – 3.6 V)	MP3SBLBBU2
MiniPID 2 PPB WR (3.0 – 3.2 V)	MP3SWBLBAU2
MiniPID 2 PPB WR (3.2 – 3.6 V)	MP3SWBLBBU2
MiniPID 2 PPB XF (3.0 – 3.2 V)	MP3SBLBXAU2
MiniPID 2 PPB XF (3.2 – 3.6 V)	MP3SBLBXBU2
MiniPID 2 HS (3.0 – 3.2 V)	MP3SHLHSAU2
MiniPID 2 HS (3.2 – 3.6 V)	MP3SHLHSBU2
MiniPID 2 10.0 eV (3.0 – 3.2 V)	MP3SBL0AU2
MiniPID 2 10.0 eV (3.2 – 3.6 V)	MP3SBL0BU2
MiniPID 2 11.7 eV (3.0 – 3.2 V)	MP3SB7BAU2
MiniPID 2 11.7 eV (3.2 – 3.6 V)	MP3SB7BBU2

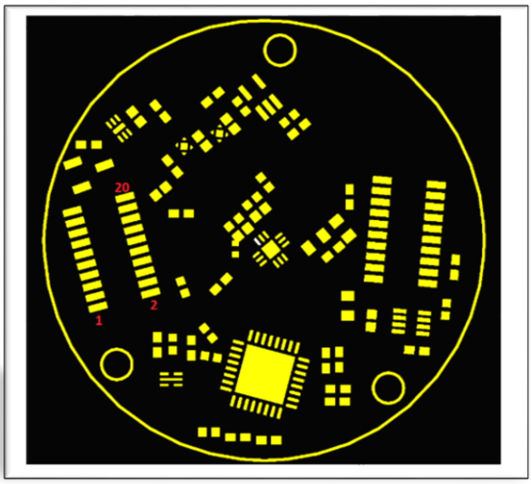
System Integration

ION CONNECT PCB (Top VIEW)

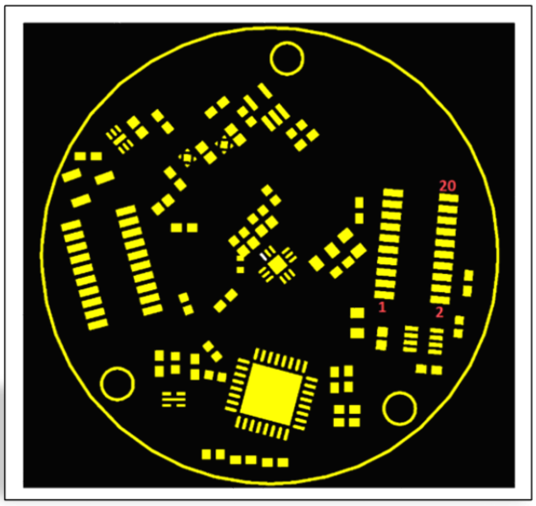


ION CONNECT PCB Top View		
Pin #	Mode (I/O)	Pin Description/Function
1 – GND	-	Ground
2 – Vin	I	Voltage provided to supported sensor (3.2V).
3 – Rx	-	Reserved for future product – Do not use
4 – Signal	O	Analog voltage output from supported sensor. Voltage range between 0-3.2V.
5 – Tx	-	Reserved for future product – Do not use
PWR	-	Reserved for future product – Do not use
HTR	-	Reserved for future product – Do not use
GND	-	Ground
SCLK	-	Reserved for future product – Do not use
SDAT	-	Reserved for future product – Do not use

ION CONNECT PCB (Bottom VIEW)



ION CONNECT PCB Bottom View		
J1 Pinout		
Pin #	**Mode (I/O)**	**Pin Description/Function**
1	-	Not used
2 - GND	-	GND
3 – RS485 Toggle	O	RS485 Transceiver toggle
4 – Error	O	Pin is driven high when ION Connect is in an error state
5 – USART_RX	I	USART Receive (from the perspective of ION Connect).
6	-	Not used
7 – USART_TX	O	USART Transmit (from the perspective of ION Connect)
8 – Alarm	O	Pin is driven high when ION Connect is in an alarm state
9 – DAC Output	O	Analog voltage is converted to current for 4-20mA when connected to ION Transmit.
10 – CONT/POLL	I	Set pin high (1.9-3.3V), ION Connect outputs data over UART continuously at 1 second intervals (see Continuous output section for data output format). Set to GND to stop continuous output mode. This pin is ignored when connected to ION Transmit.
11	-	Not used
12 – BD Type	I	GND
13	-	Not used
14 – PID Signal	O	Analog voltage from PID sensor.
15 – GND	-	GND
16	-	Not used
17	-	Not used
18 – Vin	-	3.3-5.5V (Recommend 5V input)
19	-	Not used
20	-	Not used



ION CONNECT PCB Bottom View		
J2 Pinout		
Pin #	**Mode (I/O)**	**Pin Description/Function**
1	-	Not used
2 - GND	-	GND
3 – SPI DI	I	SPI data in pin for ADC and DAC
4	-	Not used
5	-	Not used
6 – DAC CS	O	DAC chip select
7 – ADC CS	O	ADC chip select
8 – SPI CLK	O	SPI clock
9 – BOOT0	I	Boot 0 pin to initiate bootloader. This pin should be tied to GND.
10 – SWDIO	-	Used for Connect programming.
11 – RST	I	Set low to reset ION Connect
12 – I2C SCL	O	I2C serial clock
13	-	Not used
14 – I2C SDA	I/O	I2C serial data
15 – SPI DO	O	SPI data out
16 – 3V3 Output	O	3.3V Output
17 – PID Switch	-	Not used
18 – 3V2 Output	O	3.2V Output
19	-	Not used
20	-	Not used

When integrating ION CONNECT onto a custom PCB we recommend the following pins be left floating:

J1 – 14.

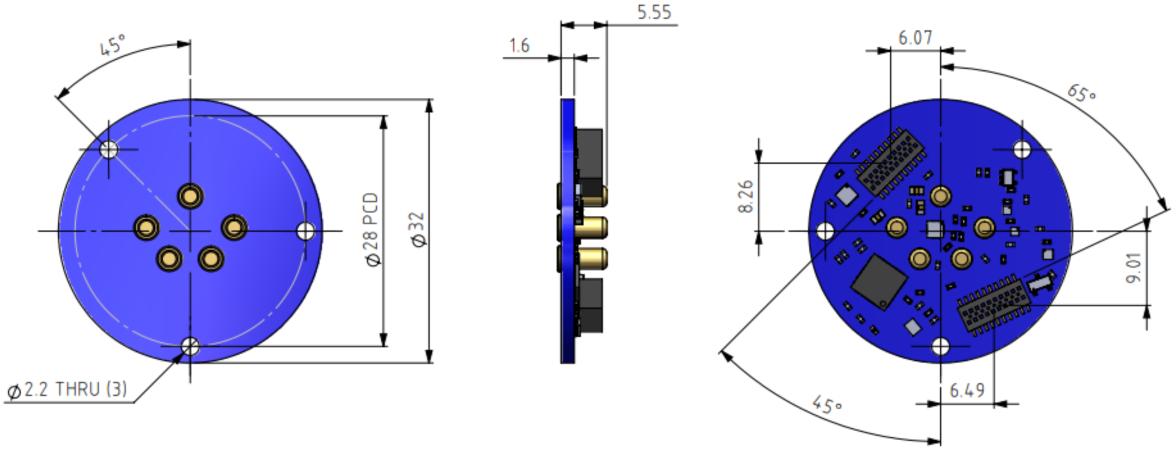
J2 – 3, 6, 7, 8, 10, 15 & 17.

ION TRANSMIT PCB (Bottom view)

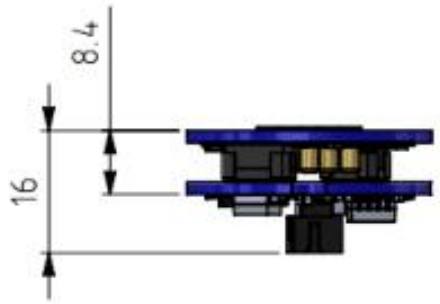
ION TRANSMIT PCB Bottom View		
J3 Pinout		
Pin #	**Mode (I/O)**	**Pin Description/Function**
1 – GND	-	Ground
2 – Error Pin	O	Error output, provides a voltage output of 3.3V when error is active.
3 – Alarm Pin	O	Alarm output, provides a voltage output of 3.3V when alarm is active.
4 – GND	-	Ground
5 – B	-	This is the non-inverting line of the differential pair of RS485 communication (Data +).
6 – A	-	This is the inverting line of the differential pair of RS485 communication (Data -).
7 – 4-20mA	O	4-20mA output is an analogue representation of the calibrated sensor value.
8 – Power	-	Voltage Input 12V-24V. Voltage provided to Transmit and Connect.

Dimensions for Installation

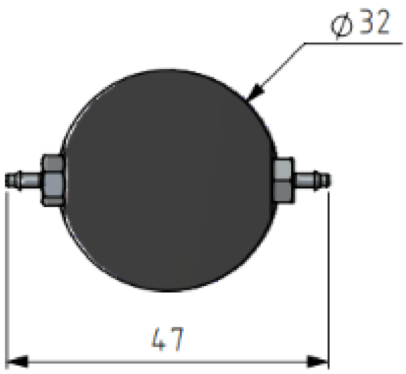
- ION CONNECT PCB



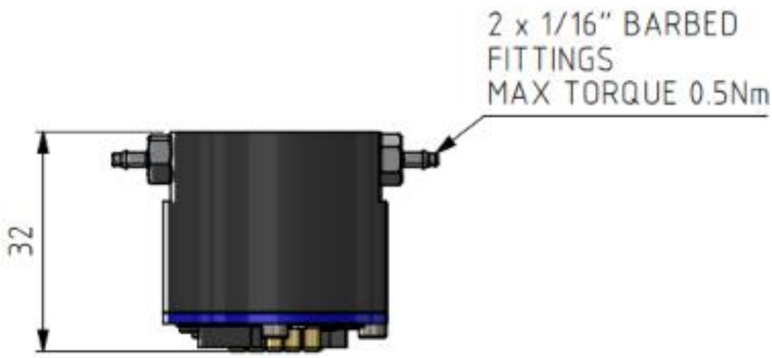
- ION TRANSMIT PCBs

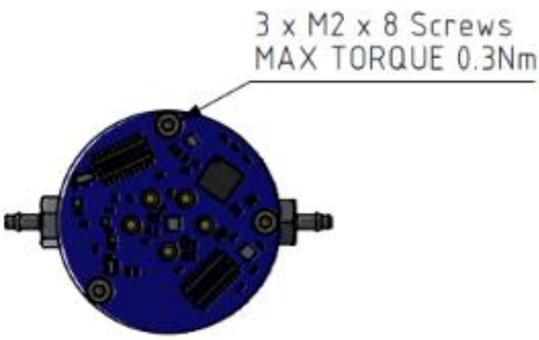


❖ ION GAS HOOD

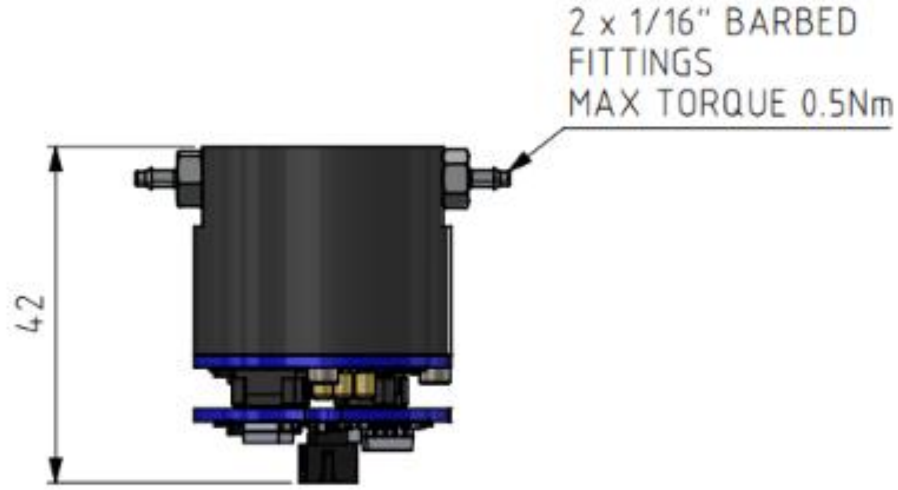


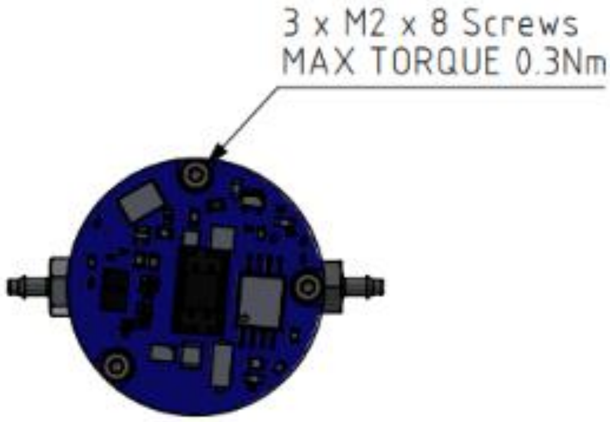
❖ ION CONNECT PCB with ION GAS HOOD



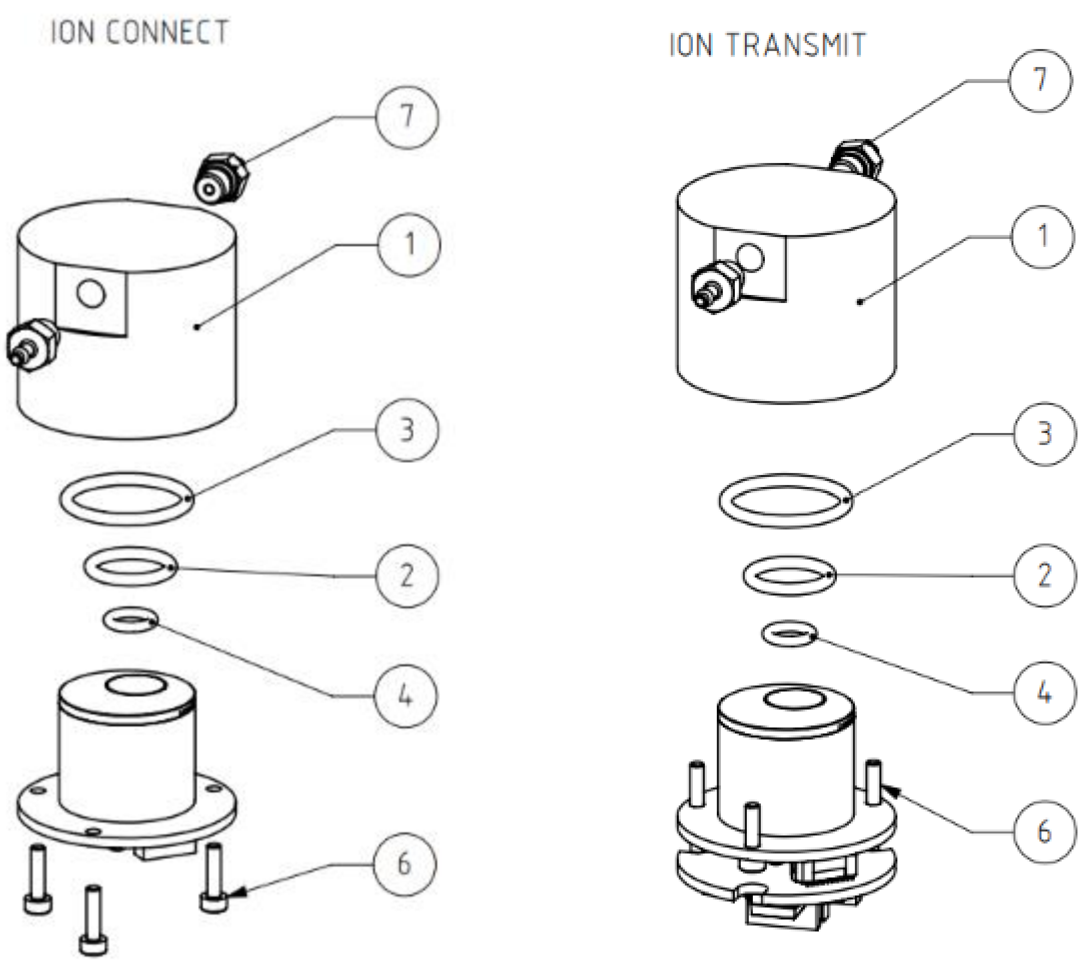


❖ **ION TRANSMIT PCBs with IONJ GAS HOOD**





Pneumatic Integration



Electronic Integration

- It is recommended that the enablers are housed in a metallic enclosure to shield against EMI from external sources.
- ION TRANSMIT must be powered from a stable 12-24VDC supply.
- ION CONNECT must be powered from a stable 3.3-5.5VDC supply.
- It is recommended that you use screened cables to protect against EMI.

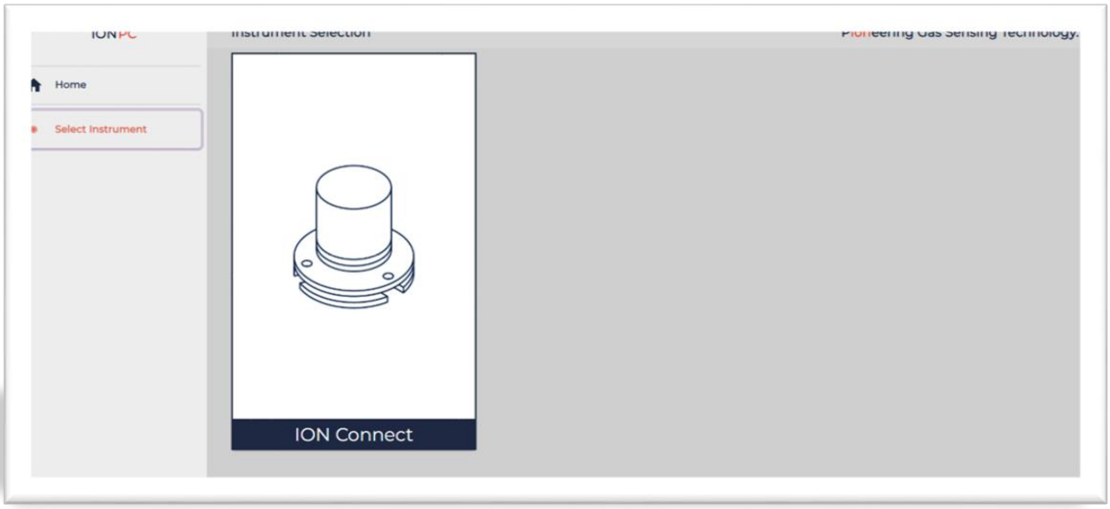
ION PC Software

ION PC is a windows based software tool that configures the Integration Kits. Please visit https://ionsense.com/ion-connect to download the software. The software is compatible with Windows 10 and 11.

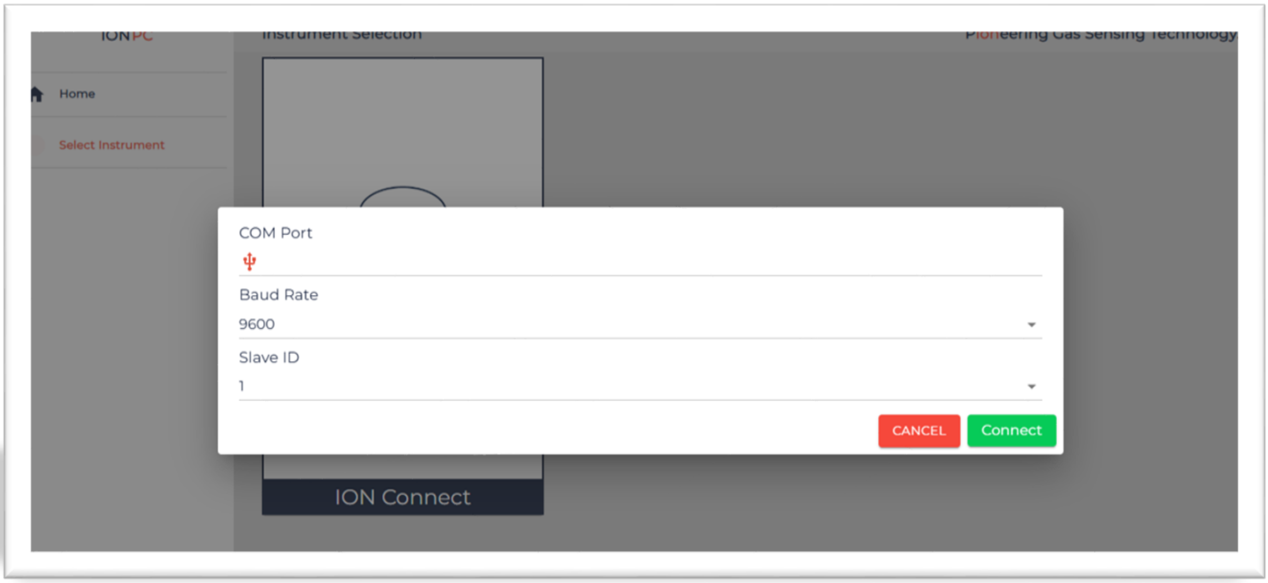
1. Download the software.
2. Open the .zip file that contains the software.
3. Install the software to your system.

Connection

1. Once connected to the PC check the COM Port which is being used in the Device Manager.
2. Open ION PC
3. Click "Select Instrument"
4. Click "ION Connect"

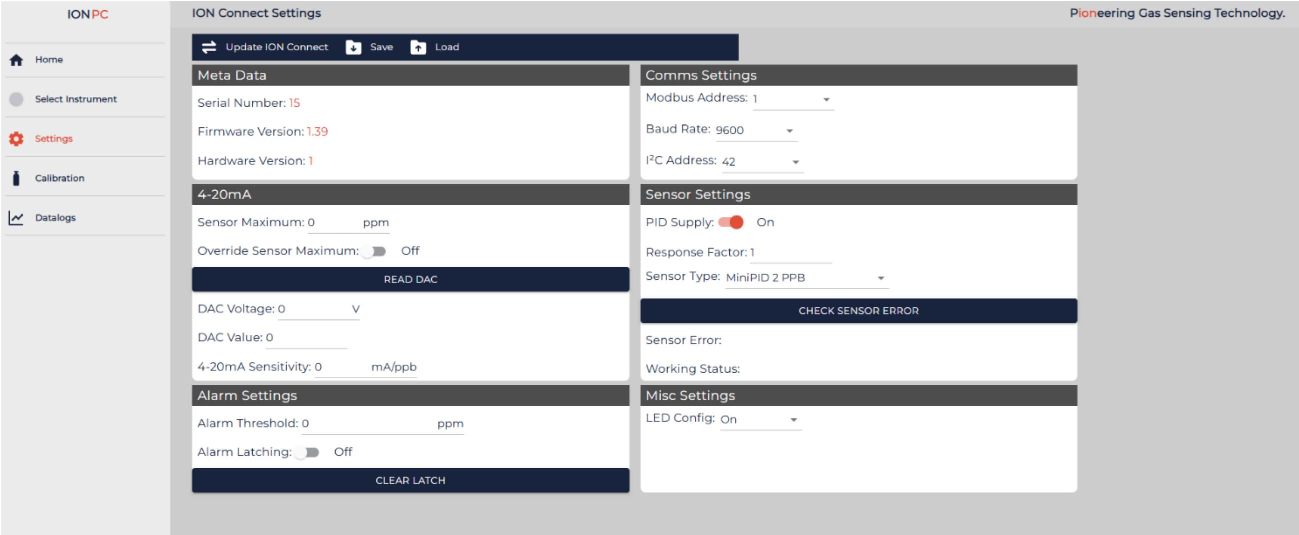


5. Select the COM Port, Baud rate and Slave ID. Factory settings for these will be 9600 Baud rate and Slave ID 1.



Settings

The Settings Page allows the user to configure their ION CONNECT. There are two types of settings, read settings and read and write settings. Read settings can be found in the read only MODBUS RTU registers. The read and write settings will allow the user to read and write specific parameters relevant to their application. If you change any read and write settings, you will need to click the Update ION CONNECT button in the top bar of this page. This top bar will also allow you to Save and Load any configurations which you have already done.



Meta Data

The Meta Data is read only information which has been input by the factory upon manufacture. These are pieces of data which relate to your ION CONNECT and its current state.

- Serial number
- Firmware version
- Hardware version

Communication

The communication settings are read and write settings which allow you to change the address and the speed of communication.

- Modbus Address (User can select addresses between 1 and 247)
- Baud Rate (9600, 19200, 38400 and 115200)
- I2C Address (User can select addresses between 1 and 127)

4-20mA

The 4-20mA section is split into a read and write section and a read section. The read and write section is for configuring the range of 4-20mA. The other settings allow the user to see the live 4-20mA readings as well as the DAC.

- Sensor Maximum
- Override Sensor Maximum
- Read DAC
- DAC Voltage
- DAC Value
- 4-20mA Sensitivity

Sensor Settings

This read and write section of the settings page is the for the sensor specific settings.

- PID Power Supply
- Response Factor
- Sensor Type

Check Sensor Error

- Sensor error
- Working status

Alarm Settings

The Alarm settings are read and write settings which will allow the user to add an alarm threshold and decide whether it stays on (Alarm Latching) once it has been activated.

- Alarm Threshold
- Alarm Latching

MISC Settings

These are settings which will allow you to change Misc settings. These are read and write settings.

- LED Config

PID Setup and Calibration

An example of what would you need to calibrate the ION SENSE® sensor is given below. For more details, please refer to the relevant section of the ION SENSE® sensors manual or get in touch.

- 1 x cylinder of ultra-high purity air (zero gas) fitted with a 300ml/min fixed flow regulator.
- 1 to 4 x cylinders of calibration gas (depending on the number of calibration points which are required) fitted with a 300 ml/min fixed flow regulator.
- Tubes to connect the regulators to the gas delivery hood. The barb on the gas delivery hood is suitable for 1/16" inner diameter tubing (Fluorinated tubing is recommended).
- ION SENSE® sensor connected to the ION CONNECT PCB.

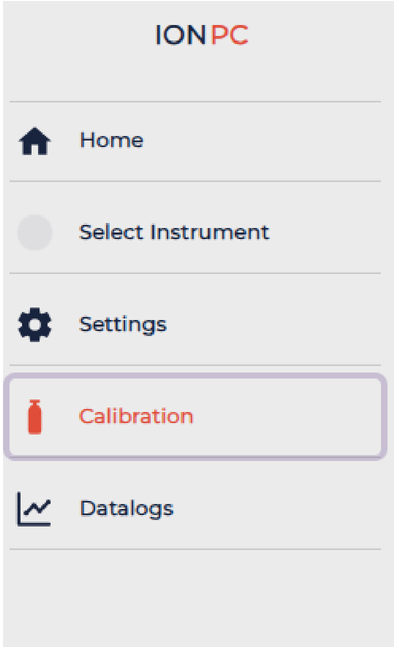
If using a pump, the following must be considered.

- Pump upstream of Gas hood: Choose a pump with suitable materials to not affect the reading. Many pumps have soft parts that will adsorb and desorb VOCs.
- Pump downstream of Gas hood: Any leaks in the pneumatic system upstream of the gas hood will result in dilution of the sample.
- The flow rate of the pump must match the flow rate of the fixed flow regulator. Over pressure can be relived using suitable T-piece.

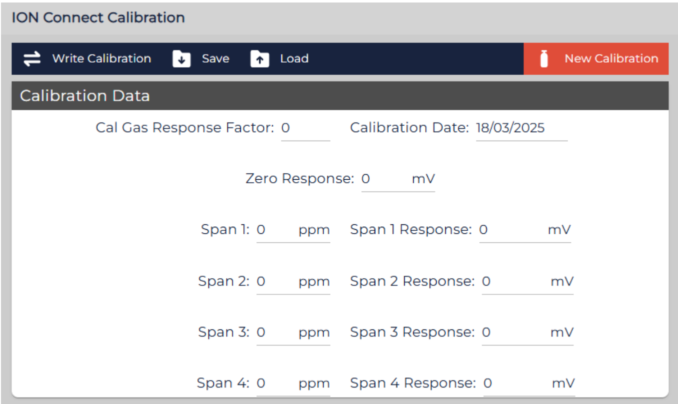
Calibration

Note: Calibrations can be saved and re-loaded to ION CONNECT via ION PC.

1. Select Calibration in the side bar.



2. Select the New Calibration button in the top bar of this page. A side panel will appear.



3. Apply zero air to the sensor. Once the readings stabilize click SET ZERO.

4. Type the Response Factor (RF) of the calibration gas and click SET CAL GAS RF.

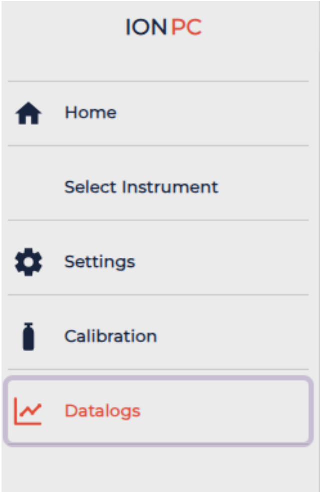
5. Type the concentration of the Span 1 gas and click SET SPAN 1.

6. Apply Span gas. Once the readings stabilize click the SPAN 1 RESPONSE.
7. Repeat steps 5 and 6 for up to four different span gas calibration points.
8. Once you have applied all you span gas calibration points click END CALIBRATION.
9. Click the button in the top bar to Write the calibration to the memory.

10. You will now be able to see the live PID readings in the Data logs section on the side bar.

Data logs

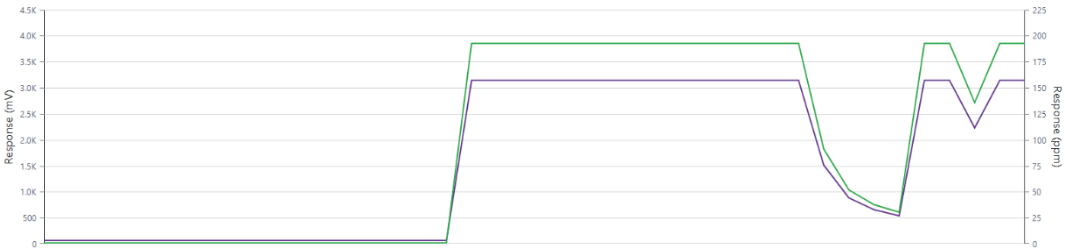
1. Select the Data logs in the side bar.



2. In the top bar of the Data logs page, you will find the start button.

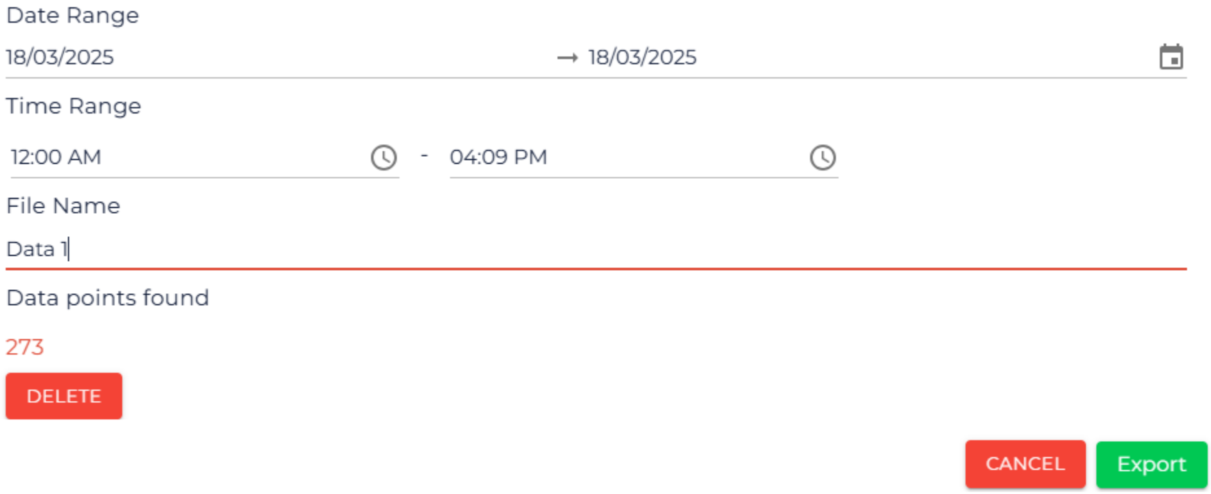
3. Select the Sampling Interval you would like to use.

4. Select the Slave Address of up to 4 ION CONNECT devices.
5. Click START
6. The live sensor readings will now appear on the graph.



7. You can toggle the mV and ppm readings on and off.
8. When you are finished with the readings click Stop in the top bar.

9. To export the data that was just recorded then click Export.



Note: The sampling time may need to be increased if packets are not read correctly during data sampling.

MiniPID Spares

Part Description	Part Number
MiniPID 2 Electrode Stack Blue	A-846486
MiniPID 2 Electrode Stack White	A-846267
MiniPID 2 Electrode Stack White + Gold	A-846417
MiniPID 2 Electrode Stack Red	A-846695
MiniPID 2 Electrode Stack PPB XF	A-846920
MiniPID 2 Electrode Stack Removal Tool	846216
MiniPID2 Long Life Lamp (3.0V to 3.2V) 10.6eV	LA4SXL3.0
MiniPID2 Long Life Lamp (3.2V to 3.6V) 10.6eV	LA4SXL3.2
MiniPID 2 Lamp Spring	846600
MiniPID 2 Lamp Cleaning Kit	A-31063

Communication Protocols

Libraries Download

Go to the website, Python Libraries and download the Python Libraries as needed. These are pre-written code and functions which help to extend the functionality of the Python language.

Modbus RTU

Introduction

For the next two sections of the libraries, we will need to do some introductions about Modbus. The Modbus is based on Modbus RTU communication protocol. It has a master-slave architecture.

Note: Non-volatile Modbus registers have write-protection when in Normal mode. The Operation Mode register (0x52) must be set to Config mode to allow successful writes to non-volatile registers. Changes take effect when the Operation Mode is set back to Normal mode.

Modbus RTU protocol description

The Modbus RTU message frame consists of the address of the Slave ID device, the function code, the special data, and the CRC checksum. The frame structure includes the following elements:

Slave ID	Function code	Data	CRC
8 bits	8 bits	N x 8 bits	16 bits

Slave Address (1-byte): This field is a unique identifier assigned to each slave device in the network, allowing the master device to communicate with specific slave devices. Since it's an 8-bit value, it accepts the range 1-247, addresses from 248 to 255 are reserved for other purpose.

Function Code (1-byte): It is a numerical code that specifies the type of action or request being made by the master device, such as reading or writing data.

Data: The data field is the actual Modbus message being transmitted, which can be the values being written/read from the slave registers. The size which is referred to as N can vary in length depending on the function code and the specific operation to be performed.

Error Check (2-byte): Modbus RTU uses a cyclic redundancy check (CRC) for checking errors. It is used to verify the integrity of the transmitted data and detect potential communication errors. The CRC which is used is a CRC16 by the Modbus RTU standard.

Silent Period: at the start and end of the frame, the >=3.5 Characters silent period is the idle time between consecutive frames, during which the communication line remains quiet. It serves as a minimum gap period that separates two frames, indicating the start and end of the message. The silent period plays a crucial role in frame synchronization and helps ensure the reliable transmission of Modbus RTU frames over the communication line.

Modbus RTU Function Codes

Function codes play a crucial role in Modbus RTU communication, as they define the type of action or request made by the master device.

Data Addresses: Modbus RTU uses four primary types of data objects: coils, discrete inputs, input registers, and holding registers. Each data object has a specific address range, allowing the master device to access the desired data within a slave device. In our application, we only use holding registers for data addressing.

Read Function Codes

Read function codes allow the master device to request data from slave devices. only Read Holding Registers is selected:

Read Holding Registers (0x03): This function code is used to read the values of multiple holding registers in a slave device.

Write Function Codes

Write function codes that enable the master device to send data to slave devices, modifying their internal states or settings. only Write Multiple Registers is selected:

Write Multiple Registers (0x10): This function code is used to write the values of multiple holding registers in a slave device.

Modbus RTU Error Handling

Effective error handling is vital in Modbus RTU to promptly identify and resolve any communication errors between the master and slave devices.

Error Detection

Modbus RTU uses a cyclic redundancy check (CRC) to detect errors in the transmitted data. The CRC is a mathematical algorithm that calculates a checksum value based on the frame's content. The sender appends this checksum to the frame, and the receiver recalculates the CRC upon receiving the frame. If the calculated CRC matches the received CRC, the frame is considered error-free. To calculate the CRC, we suggest understanding CRC16 as this is the type of CRC used within Modbus RTU.

Error Codes

When a slave device encounters an error while processing a request from the master device, it responds with an exception message containing an error code. These error codes provide information about the nature of the error, allowing the master device to take appropriate action. Some common Modbus RTU error codes include:

Error Code	Description
0x01	**Illegal Function:** The error code indicates that the requested function code is not supported by the slave device.
0x02	**Illegal Data Address:** The error code indicates that the requested data address is not valid or out of the allowable range for the slave device.
0x03	**Illegal Data Value:** The error code indicates that the data value provided in the request is not valid or allowed by the slave device.
0x04	**Slave Device Failure:** The error code indicates that the slave device encountered an internal error while processing the request.

0x05	**Acknowledge:** The error is sent by the slave device to indicate that it has received the request but needs additional time to process it.
0x06	**Slave device busy:** Th error is sent by the slave device to indicate that it is busy executing some other command. The master should send the request once the slave device is available.

Timeout and Retransmission

If the master device does not receive a response within this period, it assumes that an error has occurred, such as a lost frame or a non-responsive slave device. In such cases, the master device can attempt to retransmit the request or take other appropriate actions, such as reporting the error to the user or initiating a fault recovery process.

How to send a Modbus RTU command to read Holding Registers

In this example we will be reading from Slave ID Address 17 (HEX = 0x11), the first register address we will be reading from is 107 (HEX = 0x6B), the number of registers to read 3 (HEX = 0x03) consisting of 2 bytes per register, the following is the request string from master:

11 03 006B 0003 7687

In response to the Modbus RTU from Slave device we get:

11 03 06 AE41 5652 4340 49AD

Byte	**Request**	**Byte**	**Answer**
(Hex)	Field Name	(Hex)	Field Name
11	Device address	11	Device address
03	Functional code	03	Functional code
00	Address of the first register Hi bytes	06	Length, Number of bytes more
6B	Address of the first register Lo bytes	AE	Register value Hi 0x006B
00	Number of registers Hi bytes	41	Register value Lo 0x006B
03	Number of registers Lo bytes	56	Register value Hi 0x006C
76	Checksum CRC Lo Byte	52	Register value Lo 0x006C
87	Checksum CRC Hi Byte	43	Register value Hi 0x006D
		40	Register value Lo 0x006D
		49	Checksum CRC Lo Byte
		AD	Checksum CRC Hi Byte

How to send a Modbus RTU command to write Multiple Holding Registers

In this example we will be writing to the Slave ID Address 17 (HEX = 0x11) and be reading two registers. The first address we will be 1 (HEX = 0x01) and since we are reading two registers, we will set the number of registers to 2 (HEX = 0x02). The number of bytes need to be specified in this message, which would be 4, since we are reading 2 bytes for each register. The value which will be stored will sent to the first register is 10 (HEX = 0x000A) and the second register 258 (0x0102).

11 10 0001 0002 04 000A 0102 C6F0

In response to the Modbus RTU from Slave device we get:

11 10 0001 0002 1298

Byte	Request	Byte	Answer
(Hex)	Field Name	(Hex)	Field Name
11	Device address	11	Device address
10	Functional code	10	Functional code
00	Address of the first register Hi bytes	00	Address of the first register Hi bytes
01	Address of the first register Lo bytes	01	Address of the first register Lo bytes
00	Number of registers Hi bytes	00	Number of recorded registers Hi bytes
02	Number of registers Lo bytes	02	Number of recorded registers Lo bytes
04	Length, Number of Bytes more	12	Checksum CRC Lo Byte
00	Value Hi 0x01	98	Checksum CRC Hi Byte
0A	Value Lo 0x01		
01	Value Hi 0x02		
02	Value Lo 0x02		
C6	Checksum CRC Lo Byte		
F0	Checksum CRC Hi Byte		

What are the errors of the Modbus request

If the device receives a request, but the request cannot be processed, the device will respond with an error code.

According to Modbus RTU, the response will contain the modified Function code, the high-order bit will be 1 (HEX = 0x01). Thus, the return modified function code will be:

(HEX = 0x80) + original function code.

Example:

IT WAS	IT BECOME
Function code in request	Function error code in response
01 (01 hex) 0000 0001	129 (81 hex) 1000 0001
03 (03 hex) 0000 0011	131 (83 hex) 1000 0011
…	…
16 (10 hex) 0001 0000	144 (90 hex) 1001 0000

Sample request and response with error:

Byte	Request	Byte	Answer
(Hex)	Field Name	(Hex)	Field Name
0A	Device address	0A	Device address
01	Functional code	81	Functional code, added 0x80
04	Address of the first register Hi bytes	02	Error Code
A1	Address of the first register Lo bytes	B0	Checksum CRC Lo Byte
00	Number of registers Hi bytes	53	Checksum CRC Hi Byte
01	Number of registers Lo bytes		
AC	Checksum CRC Lo Byte		
63	Checksum CRC Hi Byte		

In error response string, the error code follows immediately after Function Code with no other data. The whole response bytes string length equals to 5 bytes.

Device address + ((HEX = 0x80) + function code) + error code + CRC checksum.

Register Map

Register with Modbus address in the table below can be accessed through Modbus Function code 3 (HEX = 0x3) for read and 16 (HEX = 0x10) for write.

The Modbus registers can be read/written as groups, or each parameter can be accessed individually. The main types of formats which we will be using are 32-bit floats, 32-bit unsigned integers and 16 unsigned integers. To write/read a register you will need to specify the Modbus Address you would like to read/write from, the available registers can be found listed below. All the registers will be using one of these formats, the structure of the data format should be Big-Endian for registers.

All registers address below have been grouped into sections,

To adjust configurations, the operation mode register must be changed the value 2 (HEX = 0x02). To save the configurations, the operation mode register must be set back to 0 (HEX = 0x00) from 2 (HEX = 0x02).

Modbus Registers Map

With the Modbus introduction done we are now able to understand the second and third sections of the ION CONNECT Libraries. The second section covers the Modbus registers. The third section is a list of defined functions for reading from and writing to the Modbus registers on ION CONNECT. The Modbus address registers are listed below, the register map has colour indicators to help support with block address sections and locations.

Modbus Address		Register	Type	RW*	Registers	Length Byte	
Non-Volatile	Board Information						
	0x00	Board Serial Number	U32	RO	2	4	

	0x02	Board Batch Number	U32	RO	2	4	
	0x04	Production Date	U32	RO	2	4	
	0x06	PCB Hardware version	U16	RO	1	2	
	0x07	Firmware Version	U16	RO	1	2	
	0x08	Sensor Serial Number	U32	RO	2	4	
Non-Volatile	Board Configuration						
	0x0B	Sensor Type	U16	RW	1	2	
	0x0C	Response Factor Value	float	RW	2	4	
	0x0E	Sensor Maximum Value	float	RW	2	4	
	0x10	Use Sensor Maximum Value	U16	RW	1	2	
	0x11	Alarm Threshold Value	float	RW	2	4	
	0x13	Alarm Latch	U16	RW	1	2	
	0x14	LED Control	U16	RW	1	2	
	0x15	Modbus Slave Address	U16	RW	1	2	
	0x16	UART/Modbus Baud Rate	U16	RW	1	2	
	0x17	I2C Slave Address	U16	RW	1	2	
Non-Volatile	Calibration Data						
	0x1C	Calibration Date	U32	RW	2	4	
	0x1E	Calibration Response Factor	float	RW	2	4	
	0x20	Zero Response	float	RW	2	4	
	0x22	Span 1 Concentration	float	RW	2	4	
	0x24	Span 1 Response	float	RW	2	4	
	0x26	Span 2 Concentration	float	RW	2	4	
	0x28	Span 2 Response	float	RW	2	4	
	0x2A	Span 3 Concentration	float	RW	2	4	
	0x2C	Span 3 Response	float	RW	2	4	
	0x2E	Span 4 Concentration	float	RW	2	4	
	0x30	Span 4 Response	float	RW	2	4	
Volatile	Sensing Telemetry						
	0x34	Sensor mV Value	float	RO	2	4	
	0x36	Sensor PPM Value	float	RO	2	4	
	0x3E	4mA Current Loop	float	RO	2	4	
	0x40	20mA Current loop	float	RO	2	4	

	0x42	4-20mA Current loop Sensitivity	float	RO	2	4	
	0x44	DAC Value	float	RO	2	4	
	0x46	DAC Data	U16	RO	1	2	
Volatile	Status Registers						
	0x47	Working Status	U16	RO	01	2	
	0x48	Sensor Error Status	U16	R0	1	2	
Volatile	Control Registers						
	0x50	PID Supply	U16	RW	1	2	
	0x51	Alarm Latch Clear	U16	RW	1	2	
	0x52	Operation Mode	U16	RW	1	2	
	0x54	New Relative Zero	float	RW	1	4	
	0x56	Use new Relative Zero	U16	RW	1	2	
Volatile	System Commands						
	0x58	Software Reset	U16	RW	1	2	

RW= read/write; RO=read only

Register Description

The following section will be explaining each individual register and how they are used by the user.

Note: Non-volatile Modbus registers have write-protection when in Normal mode. The Operation Mode register (0x52) must be set to Config mode to allow successful writes to non-volatile registers. Changes take effect when the Operation Mode is set back to Normal mode.

Board Information

The board information section contains the information for each board produced; this can be used to identify details.

Board Serial Number – 0x00

The board serial number provided will be used to identify each individual board, this is a read only register which is pre-populated and will be read as a 32-bit unsigned integer.

Board Batch Number – 0x02

The board batch number provided will be used to identify each individual board, this is a read only register which is pre-populated and will be read as a 32-bit unsigned integer.

Production Date – 0x04

The board batch number provided will be used to identify each individual board, this is a read only register which is pre-populated and will be read as a 32-bit unsigned integer formatted as YYYYMMDD.

PCB Hardware Version – 0x06

The PCB hardware version provided will be used to identify each individual board, this is a read only register which is pre-populated and will be read as a 16-bit unsigned integer.

Firmware Version – 0x07

The firmware version provided will be used to identify what firmware version has been loaded onto the board this populated when firmware upgrades occur and will be read as a 16-bit unsigned integer.

Board Configuration

The board configuration section is containing information regarding the board set-up, the following section will explain each individual section and how they can be used. To modify any of the sections, the operation mode must be set to configure.

Sensor Type – 0x0B

The sensor type register is used to determine which type of sensor is being used as a 16-bit unsigned integer; this will allow the user to select from the list below. The sensor types will have different sensor error status references, which will be dependent on the sensor's specifications. The sensor type will also have different sensor maximum values, which will affect the sensitivity ranges such as 4-20mA sensitivity and Sensor PPM sensitivity. The types of sensor type configurations have been listed below:

- Sensor HS: 1
- Sensor PPB: 2
- Sensor PPB WR: 3
- Sensor PPM: 4
- Sensor PPM WR: 5
- Sensor 10 eV: 6
- Sensor 11 eV: 7

Response Factor Value – 0x0C

The response factor value register is used to adjust the sensor ppm value dependent on factor scaling. The response factor value is a 32-bit float, which will stored in the non-volatile memory.

Sensor Maximum Value – 0x0E

The sensor maximum value register is used to determine the upper limit of the sensor ppm output, which will scale the 4-20mA sensitivity to the input value. This register is a 32-bit float and requires the 'use sensor maximum value' register to be set to value '1'.

Use Sensor Maximum Value – 0x10

The use sensor maximum value register is used to enable the upper limit of the sensor ppm output, which will scale the 4-20mA sensitivity to the input value. This register requires the 'sensor maximum value' register to have a value set. The register is a 16-bit unsigned integer register, the modes it can change between are listed below:

- Disable use sensor maximum value: 0
- Enable use sensor maximum value: 1

Alarm Threshold Value – 0x11

The alarm threshold value register is a 32-bit float register which enables the alarm pin on the board when exceeded.

Latch Alarm – 0x13

When in alarm latch mode, the alarm pin will be latched high when an alarm condition occurs. It will be cleared only with the alarm latch clear command; when not in alarm latch mode, the alarm pin will be cleared when the alarm condition has disappeared. The alarm latch register is a 16-bit unsigned integer register, the modes it can change between are listed below:

- Disable alarm latch: 0
- Enable alarm latch: 1

LED Control – 0x14

The LED control configuration allows the user to adjust the LED on-board to perform different actions for indication. There are three modes which can occur, the LED being set to off, the LED being set on and LED set to pulsing for 1 second intervals. The LED control register is a 16-bit unsigned integer register, the modes it can change between are listed below:

- LED off: 0
- LED on: 1
- LED pulse 1 second: 2

Modbus Slave Address – 0x15

The modbus slave address register will allow the user to change the modbus slave address for the board. The register is a 16-bit unsigned integer which accepts values between 1 and 127 for this register.

UART/Modbus Baud Rate – 0x16

The UART/Modbus baud rate register will allow the user to change the baud rate of the board. When change the baud rate, this will affect both the UART and Modbus as they are not independent from one another.

The following information will remain the same for all the communications which will be used.

- Parity: Even
- Stop bits: 1

The register is a 16-bit unsigned integer which accepts values listed below:

- Baud rate 9600: 1
- Baud rate 19200: 2
- Baud rate 38400: 3
- Baud rate 115200: 4

I2C Slave Address – 0x17

The I2C slave address register will allow the user to change the I2C slave address for the board. The register is a 16-bit unsigned integer which accepts values between 10 and 127 for this register.

Calibration Data

The calibration data section contains information regarding the calibration of the sensor, the user to adjust and perform calibrations to their needs. The following section will explain each individual section and how they can be used.

Calibration Date – 0x1C

The calibration date register stores the calibration date provided by the user; the date is a 32-bit unsigned integer which is formatted as YYYYMMDD.

Calibration Response Factor – 0x1E

The calibration response factor is a 32-bit float which stores the response factor at calibration, this will change dependent on gas provided.

Zero Response – 0x20

The zero response is a 32-bit float which stores the zero response of the calibration. To set the zero response, the ‘sensor mv’ at the first point of calibration is to be stored in the register.

Span 1 Concentration – 0x22

The span 1 concentration is a 32-bit float which sets the ‘span 1 response’ to a concentration value. To set the span 1 concentration, the concentration being applied at ‘span 1 response’ needs to be inserted here.

Span 1 Response – 0x24

The span 1 response is a 32-bit float which stores the span 1 response of the calibration. To set the span 1 response, the ‘sensor mv’ at the second point of calibration is to be stored in the register.

Span 2 Concentration – 0x26

The span 2 concentration is a 32-bit float which sets the ‘span 2 response’ to a concentration value. To set the span 2 concentration, the concentration being applied at ‘span 2 response’ needs to be inserted here.

Span 2 Response – 0x28

The span 2 response is a 32-bit float which stores the span 2 response of the calibration. To set the span 2 response, the ‘sensor mv’ at the third point of calibration is to be stored in the register.

Span 3 Concentration – 0x2A

The span 3 concentration is a 32-bit float which sets the ‘span 3 response’ to a concentration value. To set the span 3 concentration, the concentration being applied at ‘span 3 response’ needs to be inserted here.

Span 3 Response – 0x2C

The span 3 response is a 32-bit float which stores the span 3 response of the calibration. To set the span 3 response, the 'sensor mv' at the fourth point of calibration is to be stored in the register.

Span 4 Concentration – 0x2E

The span 4 concentration is a 32-bit float which sets the 'span 4 response' to a concentration value. To set the span 4 concentration, the concentration being applied at 'span 4 response' needs to be inserted here.

Span 4 Response – 0x30

The span 4 response is a 32-bit float which stores the span 4 response of the calibration. To set the span 4 response, the 'sensor mv' at the fifth point of calibration is to be stored in the register.

Sensing Telemetry

The sensing telemetry section contains information received from the sensors and output values. The registers are read-only, the following section will explain each individual section.

Sensor mV Value – 0x34

The sensor mv value register outputs the sensor voltage as a 32-bit float.

Sensor PPM Value – 0x36

The sensor ppm value register outputs the sensor ppm using the conversions from the calibration response factor and response factor as a 32-bit float.

Note: Without a valid calibration on the board (at least a zero response and 1 span response) PPM readings will show 0.

4mA Current Loop – 0x3E

The 4mA current loop register outputs the 4mA current loop value, which is the lower side of the 4-20mA, this value is set in ppm format.

20mA Current Loop – 0x40

The 20mA current loop register outputs the 20mA current loop value, which is the higher side of the 4-20mA, this value is set in ppm format. This is affected by the sensor type; this will be overwritten to use sensor maximum value register if the register 'use sensor maximum value' is set to '1'.

4-20mA Current Loop Sensitivity – 0x42

The 4-20mA current loop register outputs the sensor defined sensor maximum value / difference between the 4mA and 20mA current loop. If customer uses custom sensor maximum value, then sensor defined sensor maximum value is no longer used.

DAC Value – 0x44

The DAC value register outputs the voltage output at sensor ppm concentrations, this is using the 4-20mA current loop sensitivity per ppm. When the sensor ppm is 0, the DAC value will be 0.4V. The DAC will be used for the 4-20mA current loop out if transmit board is connected.

DAC Data – 0x46

The DAC data register outputs the raw analogue to digital to the register as a 16-bit unsigned integer.

Status Registers

The status registers will be used to provide information regarding both the board and sensor.

Working Status – 0x47

The working status register is used to identify the state of the board, this can provide information regarding issues occurring and modes. The working status register is a 16-bit unsigned register which can be in the states listed below:

- Normal operation: 0
- Memory Error: 1
- Uninitialized: 2
- PID unpowered: 3
- UART continuous: 4
- Alarm active: 5
- Error occurred: 225

Sensor Error Status – 0x48

The sensor error status register is used to identify the state of the sensor, this can provide information regarding issues occurring and modes. The sensor error pin will be pulled high when the sensor error status is active, otherwise it will be pulled low. The working status register is a 16-bit unsigned register which can be in the states listed below:

- No error occurred: 0
- Lamp not illuminated due to electrode stack: 1
- Oscillator not working on electrode stack: 2
- Oscillator overload: 3
- Power removed from PID: 4
- Indicated alarm condition has occurred: 16

Control Registers

PID Supply – 0x50

The function of this register is to turn off power to the sensor, this is formatted as a 16-bit unsigned integer. When this is done the sensor mV should output 0V, consideration should be taken when powering sensor to allow warm-up for sensor to take place.

- Disable PID supply: 0
- Enable PID supply: 1

Alarm Latch Clear – 0x51

The following register will be used to clear the alarm pin for when the alarm threshold is no longer exceeded but the latch alarm is still active, this will pull the alarm pin low until threshold is exceeded again. This register works as a rising edge, the value will set back to '0' after completion of task without needing any input value after the value below is provided:

- Clear latch alarm: 1

Operation Mode – 0x52

There are two operation modes: normal mode and config mode. Normal mode will be used for general operation which will only provide limited registers to be modified such as relative zero, PID supply and alarm latch clear etc. Config mode will allow the user to adjust the calibration data and configurations such as response factor, sensor type and alarm thresholds, etc. The register is a 16-bit unsigned integer which can be the following values:

- Normal mode: 0
- Config mode: 1

New Relative Zero – 0x54

The new relative zero value is a 32-bit float register which contains a new zero value adjusting the calibration data to an offset set from the zero response and the new relative zero value. This will use the value from the new relative zero register and adjust the current calibration and sensor PPM value taking the zero into consideration. This requires the 'use new relative zero' register to be set to '1' to perform this.

Use New Relative Zero – 0x56

In other to use the functionality of new relative zero, the user is required to change the state to '1' to enable the relative zero, this will use the value from the new relative zero register and adjust the current calibration and sensor PPM value taking the zero into consideration. If the value is set back to '0' the zero will return to the factory zero response reference, the sensor PPM value and calibration will no longer take the relative zero as a reference.

- Disables the relative zero value: 0
- Enables the relative zero value: 1

Software Reset – 0x58

This functionality will perform a hardware reset using software, which will take approximately 3 seconds to perform. When the value '1' is inserted to the register, the instrument will stop responding for approximately 2 seconds to reset and then will perform as if the instrument had just been powered on. This register works as a rising edge, the value will set back to '0' after completion of task without needing any input value after the value below is provided:

- Software reset: 1

Modbus Class

The Modbus class is a bundle of functions which can be called within code to perform key actions. At the start of this third section, you will find the functions which allow you to initialize the Modbus connection. After this, it is functions to get data from the registers and to set data into the registers. To begin with you will need to write,

```
import ION_Connect  # Importing ION_Connect module for Modbus communication (ensure this module is installed)
```

This will import the ION CONNECT Libraries. Once the libraries are imported the Modbus class needs to be initialized and opened.

```
# Initialize and open Modbus connection
modbus = ION_Connect.Modbus(port, baudrate, timeout) # Initialize Modbus connection with specified port, baudrate and timeout
modbus.open()  # Open the Modbus connection
```

From here you can configure the different Modbus Registers. To read the registers you will need to use one of the get functions.

```
modbus.get_sensor_mv() # Get sensor mV value
modbus.get_sensor_ppm() # Get sensor PPM value
```

To write to one of the Modbus registers you will need to first change the operation mode to allow you to set the registers. From here you will be able to use the set functions to write information into the Modbus registers.

```
# Configure Modbus settings
modbus.set_operation_mode(1)  # Enter configure mode
modbus.set_sensor_type(4)  # Set miniPID sensor ppm configuration - refer to manual
modbus.set_response_factor(1)  # Set response factor to 1.0
modbus.set_alarm_threshold_value(2000)  # Set alarm threshold to 2000 ppm
modbus.set_alarm_latch_state(1)  # Enable alarm latch
modbus.set_led_control(2)  # Set on-board LED to blink/flash at 1-second intervals
modbus.set_operation_mode(0)  # Exit configure mode

modbus.close() # Close the Modbus connection
```

I2C

Introduction

Connect supports I2C protocol and is configured as a slave device with configurable address. The address is stored in EEPROM and is used during initialization to set the I2C slave address. If no value is found in EEPROM then 0x2A is used as a default address.
I2C should support fast mode (400KHz) and clock stretching.
This document refers to the protocol used for the external I2C on pins A11 & A12.
Note: <u>There are no pull-up resistors on the Ion Connect PCB</u>. The I2C master is responsible for implementing pull-up resistors on the SCL and SDA lines.

Note: Non-volatile Modbus registers have write-protection when in Normal mode. The Operation Mode register (0x52) must be set to Config mode to allow successful writes to non-volatile registers. Changes take effect when the Operation Mode is set back to Normal mode.

Protocol

Below describes the read and write protocol between an I2C master and the I2C slave (Connect). A command frame is sent which is then followed up by a read or write frame.

Write 4 Bytes

Master	Start	I²C Address	Write		CMD [47:40]		Data0 [39:32]		Data1 [31:24]		Data2 [23:16]		Data3 [15:8]		CRC [7:0]		STOP
Slave				ACK		ACK		ACK		ACK		ACK		ACK		ACK	

Write 2 Bytes

Master	Start	I²C Address	Write		CMD [31:24]		Data0 [23:16]		Data1 [15:8]		CRC [7:0]		STOP
Slave				ACK		ACK		ACK		ACK		ACK	

Write 1 Byte

Master	Start	I²C Address	Write		CMD [23:16]		Data0 [15:8]		CRC [7:0]		STOP
Slave				ACK		ACK		ACK		ACK	

Read 4 Bytes

Master	Start	I²C Address	Read		CMD [47:40]			ACK		ACK		ACK		ACK		ACK	STOP
Slave				ACK		ACK	Data0 [39:32]		Data1 [31:24]		Data2 [23:16]		Data3 [15:8]		CRC [7:0]		

Read 2 Bytes

Master	Start	I²C Address	Read		CMD [31:24]			ACK		ACK		ACK	STOP

Slave				ACK		ACK	Data0 [23:16]		Data 1 [15:8]		CRC [7:0]		

Read 1 Byte

Master	Start	I²C Address	Read		CMD [23:16]			ACK		ACK	STOP
Slave				ACK		ACK	Data 0 [15:8]		CRC [7:0]		

The I2C protocol includes a CRC, which will be a CRC-8/NRSC-5.
The CRC-8/NRSC-5 protocol is a cyclic redundancy check (CRC) algorithm used for error detection in data transmission. It operates on each byte of data to ensure integrity. In this protocol, the data packet consists of five, three or two bytes, each serving a specific purpose:

In the case of five bytes:

1. CMD (Command Byte): The first byte in the sequence, referred to as CMD, contains the command or instruction for the data packet.
2. Data[0]: The second byte, labelled data[3], holds the most significant part of the data.
3. Data[1]: The third byte, labelled data[2], contains the next significant part of the data.
4. Data[2]: The fourth byte, labelled data[1], holds the next part of the data.
5. Data[3]: The fifth byte, labelled data[0], contains the least significant part of the data.

In the case of three bytes:

1. CMD (Command Byte): The first byte in the sequence, referred to as CMD, contains the command or instruction for the data packet.
2. Data[0]: The second byte, labelled data[1], holds the next part of the data.
3. Data[1]: The third byte, labelled data[0], contains the least significant part of the data.

In the case of two bytes:

1. CMD (Command Byte): The first byte in the sequence, referred to as CMD, contains the command or instruction for the data packet.
2. Data[0]: The second byte, labelled data[0], contains the least significant part of the data.

Find below an example of how the CRC-8/NRSC-5 is calculated:

```
def _calculate_crc8(data: list) -> int:
    """
    Calculates CRC-8/NRSC-5.

    :param data: CMD & Data bytes received from buffer.
    :type data: list
    :returns: CRC value calculated from buffer using CRC-8/NRSC-5.
    :rtype: int
    """
    crc = 0xff
    for byte in data:
```

```
        crc ^= byte
        for _ in range(8):
            if (crc & 0x80) != 0:
                crc = (crc << 1) ^ 0x31
            else:
                crc <<= 1
            crc &= 0xff  # Ensure crc remains an 8-bit value
    return crc
```

I2C Class

The next section to the ION CONNECT Libraries is the I2C class. Like the previous section this one will begin with functions to initialize the I2C connection. First you will import the ION_CONNECT Library.

```
import ION_Connect  # Importing ION_Connect module for I2C communication (ensure this module is installed)
```

This will import the ION CONNECT Libraries. Once the libraries are imported the I2C class needs to be initialized and opened.

```
# Initialize I2C bus connection with specified address and bus
i2c_address = 0x15
i2c_bus = 1

# Initialize and open I2C bus connection
i2c = ION_Connect.I2C(i2c_address, i2c_bus)  # Initialize I2C bus connection with specified address and bus
i2c.open()  # Open the I2C bus
```

From here you can configure the different I2C Registers. To read the registers you will need to use one of the get functions.

```
i2c.get_sensor_mv() # Get sensor mV value
i2c.get_sensor_ppm() # Get sensor PPM value
```

To write to one of the I2C registers you will need to first change the operation mode to allow you to set the registers. From here you will be able to use the set functions to write information into the I2C registers.

```
# Configure I2C settings
i2c.set_operation_mode(1)  # Enter configure mode
i2c.set_sensor_type(4)  # Set miniPID sensor ppm configuration - refer to manual
i2c.set_response_factor(1)  # Set response factor to 1.0
i2c.set_alarm_threshold_value(2000)  # Set alarm threshold to 2000 ppm
i2c.set_alarm_latch_state(1)  # Enable alarm latch
i2c.set_led_control(2)  # Set on-board LED to blink/flash at 1-second intervals
i2c.set_operation_mode(0)  # Exit configure mode

i2c.close() # Close the I2C bus
```

UART

Introduction

Connect supports UART continuous output protocol and is configured using a GPIO referred to as CONT/POLL which is J1 pin 10 which needs to tie to ground to activate this mode.

If Transmit is connected, the continuous output mode will not function.

Note: Non-volatile Modbus registers have write-protection when in Normal mode. The Operation Mode register (0x52) must be set to Config mode to allow successful writes to non-volatile registers. Changes take effect when the Operation Mode is set back to Normal mode.

Protocol

The UART continuous protocol includes a CRC, which will be a CRC-16.

Each byte in the string is listed in detail as below:

Byte Sequence	Field Name	Description
0	Device address	Default as 0x1, but user programable 0 ~ 247
1	Functional code = 03	0x03
2	Data Length	Total bytes length (including CRC)
3	Sensor PPM	Float, 4 bytes
4	Sensor PPM	
5	Sensor PPM	
6	Sensor PPM	
7	Sensor mV	Float, 4 bytes
8	Sensor mV	
9	Sensor mV	
10	Sensor mV	
11	DAC value	Float, 4 bytes
12	DAC value	
13	DAC value	
14	DAC value	
15	Sensor Error Status	1 byte
16	Sensor Type	1 byte
17-28	Not used	
29	Checksum CRC Hi Byte	
30	Checksum CRC Lo Byte	

UART Continuous Class

The next section to the ION CONNECT Libraries is the UART Continuous class. Like the Modbus section this one will begin with functions to initialize the Serial port connection. First you will import the ION_CONNECT Library.

```
import ION_Connect  # Importing ION_Connect module for UART Continuous communication (ensure this module is installed)
```

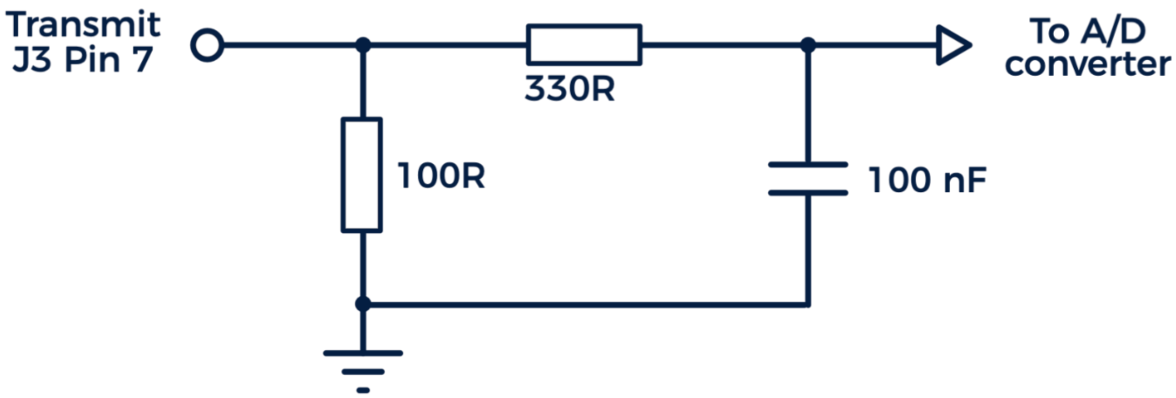
This will import the ION CONNECT Libraries. Once the libraries are imported the UART Continuous class needs to be initialized and opened.

```
# Initialize UART Continuous Serial connection with specified port, baudrate, timeout and parity
port = '/dev/ttyUSB0'
baudrate = 9600
timeout = 1
parity = 'E'

# Initialize and open UART Continuous Serial connection
uart_continuous = ION_Connect.UARTContinuous(port, baudrate, timeout, parity)  # Initialize UART Continuous Serial with specified port and baudrate
uart_continuous.open()  # Open the Serial connection
```

From here you can read the different UART Continuous Registers, there is no writeable functions within this class.

```
uart_continuous.get_sensor_mv() # Get sensor mV value
uart_continuous.get_sensor_ppm() # Get sensor PPM value
uart_continuous.get_sensor_type() # Get miniPID sensor ppm configuration - refer to manual

uart_continuous.close() # Close the Serial connection
```

4-20 mA Wiring Diagram



Part numbers

Part Description	Part Number
ION SENSE® CONNECT Kit	ION SENSE CONNECT kit
ION SENSE® TRANSMIT Kit	ION SENSE TRANSMIT Kit
ION CONNECT PCB	A-952100
ION TRANSMIT PCB	A-952101
ION PROGRAM PCB	A-952102
ION Gas Hood	A-952204

FAQ

Sensor PPM and 4-20mA output not changing or incorrect

A valid calibration must be stored on the ION CONNECT for it to output ppm values and 4-20mA to an ION TRANSMIT board. A valid calibration must have a span response value, a zero response and at least one span response/concentration.

Verify the selected sensor is correct within the 'Sensor Type' register.

If the sensor maximum value is being overridden, check the 'Sensor Maximum Value' and 'Use Sensor Maximum Value" registers have accurate values assigned.

Writing into the registers and some of them aren't updating

Non-volatile Modbus registers have write-protection when in Normal mode. The Operation Mode register (0x52) must be set to Config mode to allow successful writes to non-volatile registers. Changes take effect when the Operation Mode is set back to Normal mode.

How to enable continuous output mode

Continuous output mode works when connected to ION CONNECT and the following conditions are met:

- ION TRANSMIT is not connected.
- Pin 10 of connector J1 must be set high (1.9-3.3V).

Modbus RTU troubleshooting

Each slave device must be unique in address on the bus and ensure slave id is correct. Default slave id is 1.

Verify serial settings match (baud rate, 1 Stop Bit & Even parity). Default baud rate is 9600.

If using third party RS485 to USB converters, ensure drivers are correctly installed.

Increase timeouts and introduce a retry policy on dropped packets.

Manual Update Log

Manual Version	Amendment	Issue Date	Instrument Firmware	PC Software
V 1.0	Original Version	23/07/2025