AtlasScientific™
Environmental Robotics
V 1.9
Revised 12/22

EZO-CO2™

Embedded NDIR CO2 Sensor

Reads	Gaseous CO2
Range	0 – 10,000 ppm
Calibration	Factory calibrated
Pressure	Atmosphere only
Response time	1 reading per second
Resolution	1 ppm
Accuracy	(+/- 5%) + (+/- 50 ppm)
Connector	5 lead data cable
Warmup time	10 seconds
Cable length	1 meter
Data protocol	UART & I²C
Default I²C address	105 (0x69)
Data format	ASCII
Operating voltage	3.3V – 5V
Life expectancy	~5.5 years

Written by Jordan Press
Designed by Noah Press

This is an evolving document, check back for updates.

Table of contents

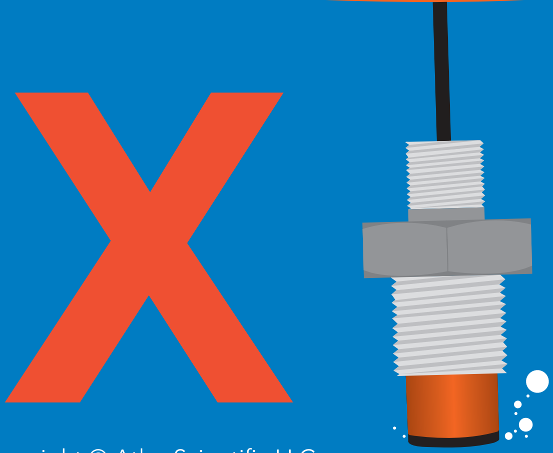
UART

I²C

Attention

The EZO-CO2™ is 100% operational out of the box.
CALIBRATION IS UNNECESSARY

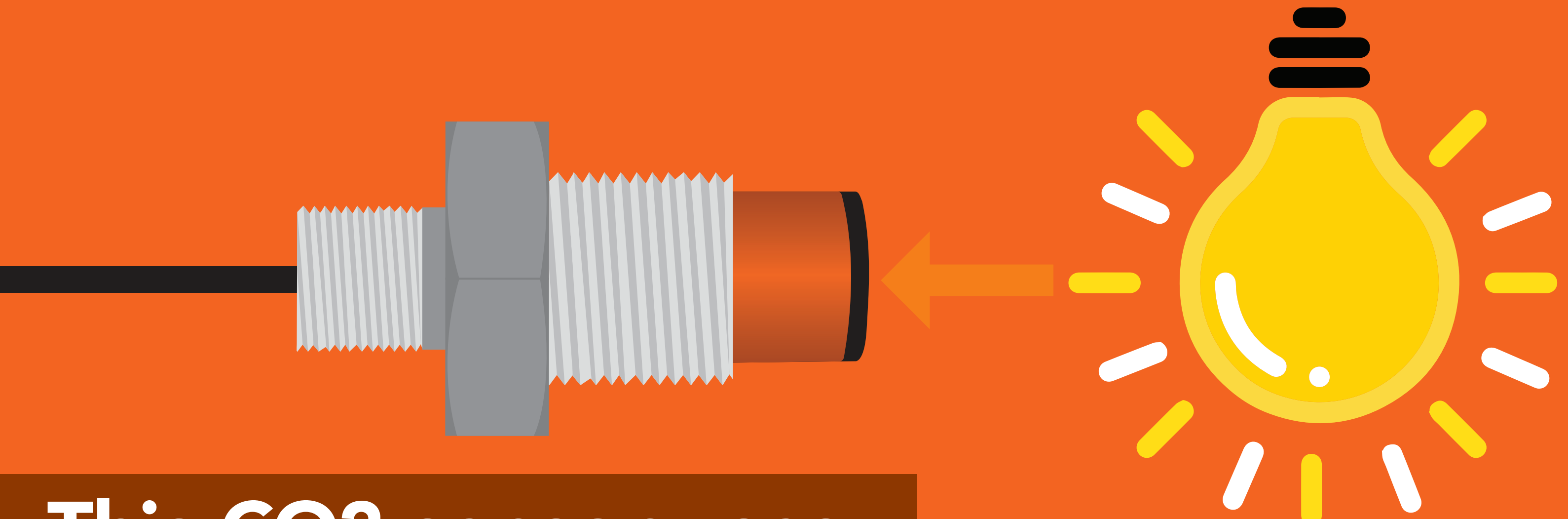
This sensor detects
GASEOUS CO2



This sensor does not read dissolved CO2.
DO NOT SUBMERGE!

Attention

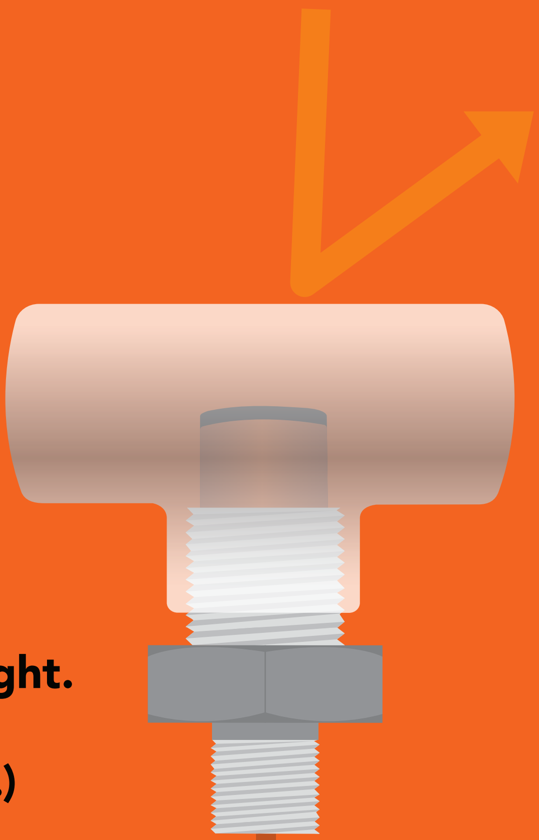
Do not point the sensor ***directly*** at bright lights



This CO2 sensor uses IR light to detect CO2.

Pointing the sensor directly at a bright light will give false readings.

(it will not damage the sensor.)



If the CO2 sensor is returning false readings when in a bright environment, try attaching a PVC Tee to the sensor, to block the direct light.

(or just don't point the sensor at bright lights.)

AtlasScientific™
Environmental Robotics

Attention

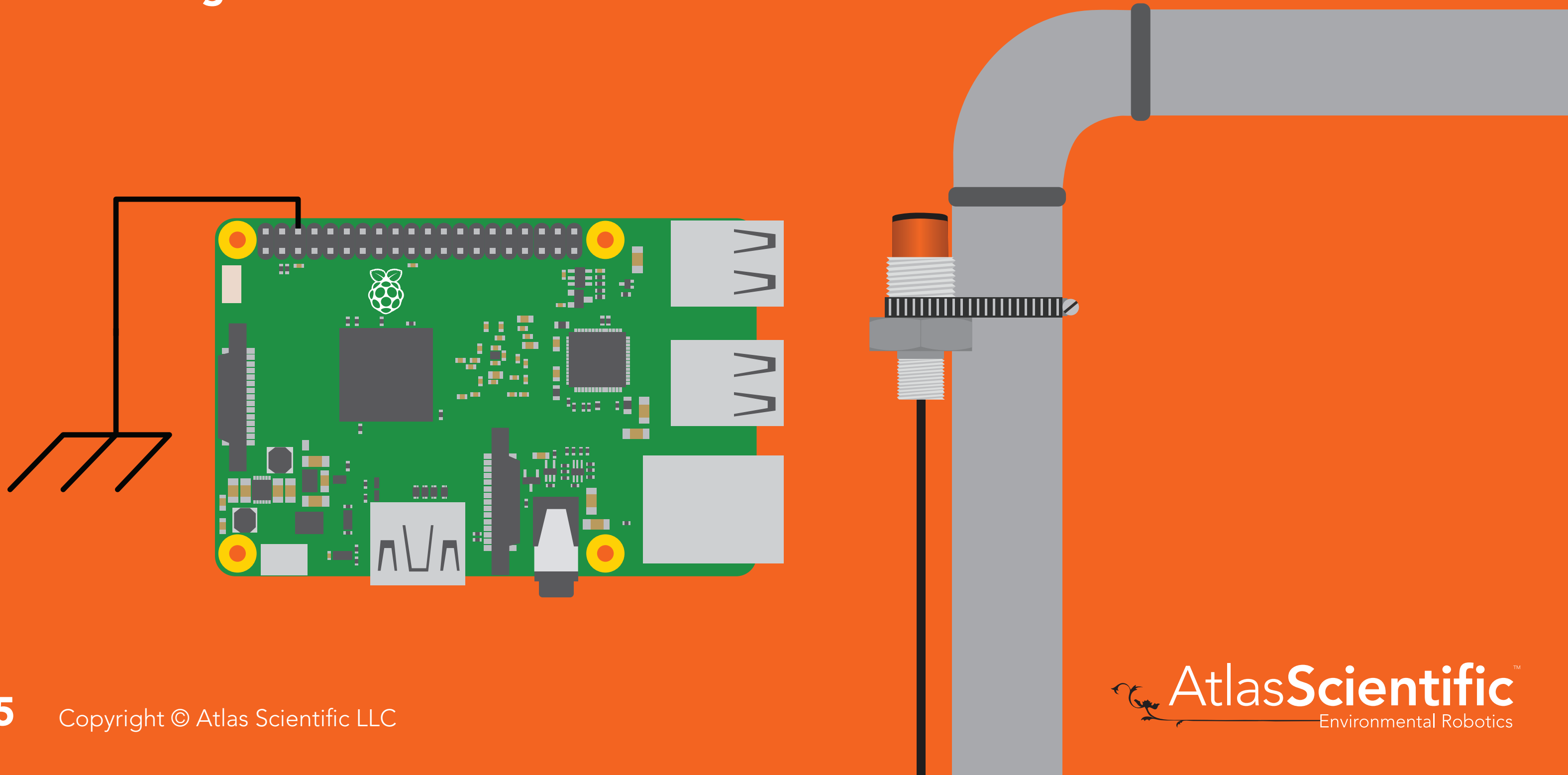
This CO2 sensor is sensitive to ground loops.

Put simply, a ground loop is when the ground line is not actually 0 volts. (It's the buzzing you hear in audio equipment)

If your system has a ground loop you will see readings that are between 100 and 250 ppm higher than expected. Atlas Scientific has detected ground loops on many different Raspberry Pi's. If this sensor is connected to a Raspberry Pi you should expect to have a ground loop.

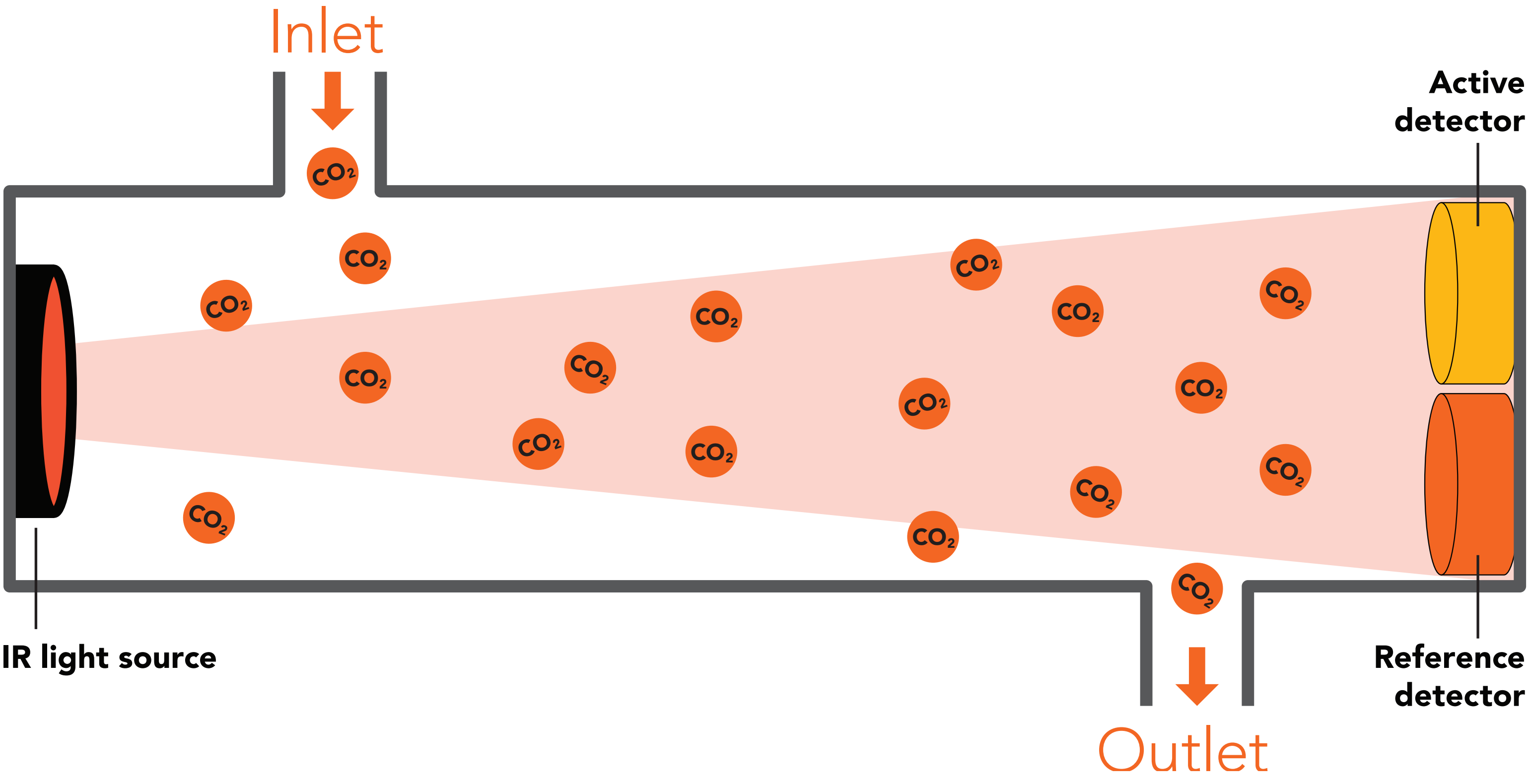
There are two ways to fix this problem

1. Connect a ground pin from the Raspberry Pi (or other device) to an earth ground.
2. Connect the body of the CO2 sensor to a metal object that is connected to an earth ground.

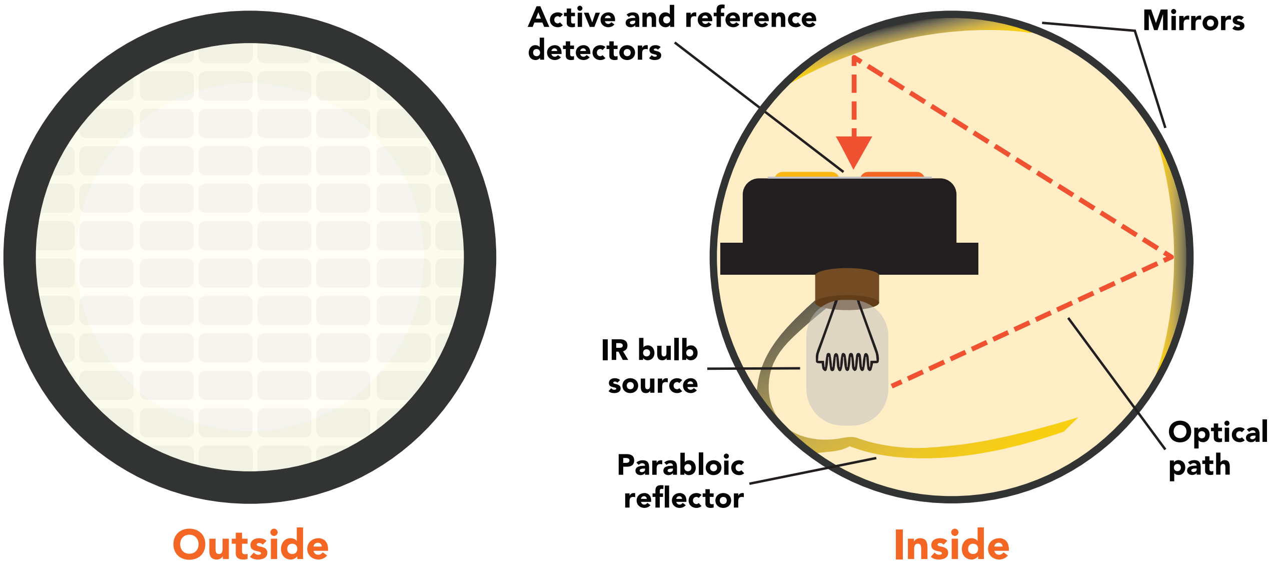


Operating principle

The Atlas Scientific EZO-CO2™ Embedded CO2 Sensor uses a non-dispersive infra-red (NDIR) gas detection cell to derive CO2 content in a gaseous matrix. The NDIR detection cell is a single wavelength spectrophotometer that has been specifically designed to detect 4.2μm infrared radiation.

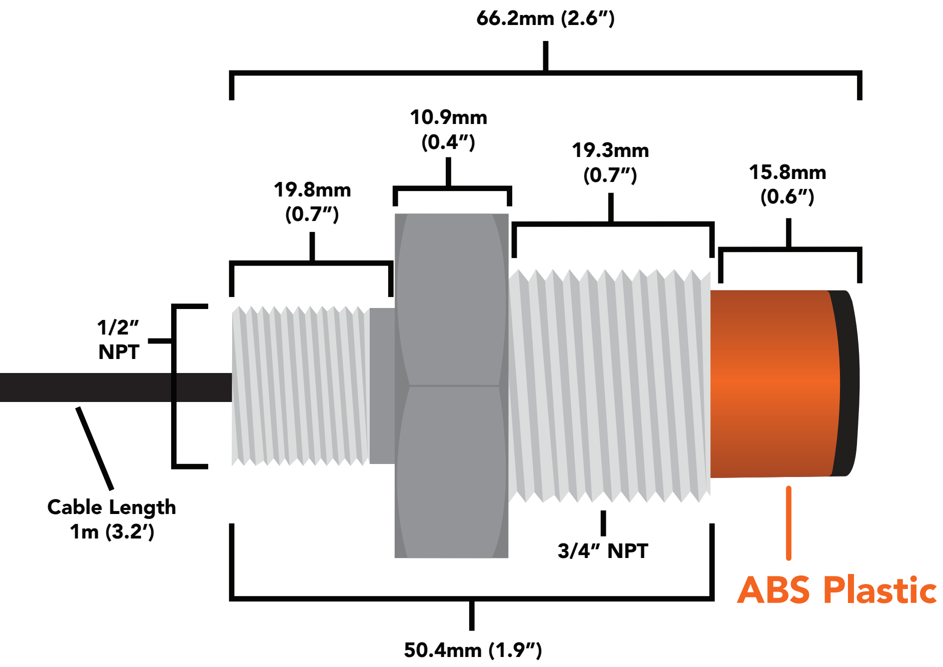


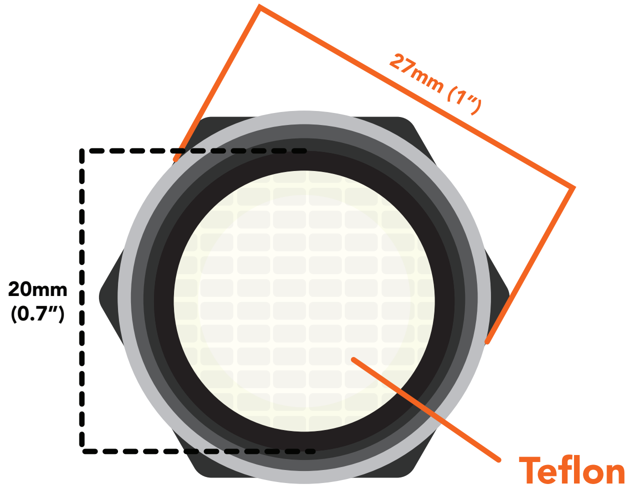
Gaseous CO2 has a prominent absorption band centered at 4.2μm. CO2 content is derived by quantifying how much light energy has been lost when it travels through a gaseous matrix over a fixed distance.

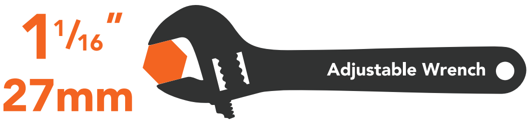


Physical properties

The EZO-CO2™ sensor only detects gaseous CO2 levels. This device cannot read dissolved CO2 levels. *DO NOT SUBMERGE IN LIQUID.*





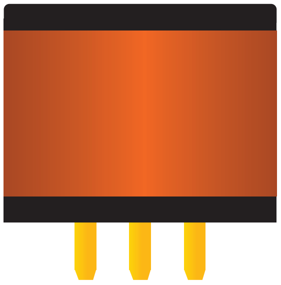


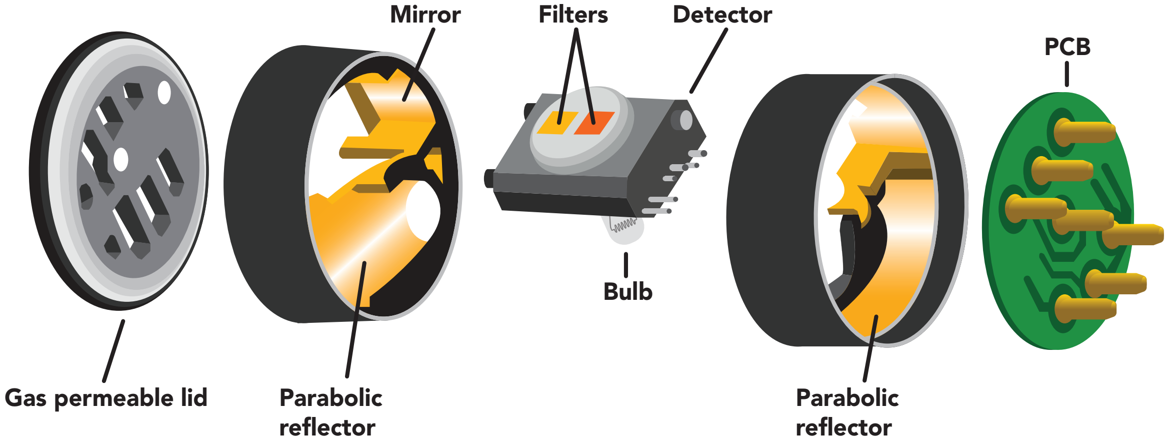
Weight 133g

Body 316 Stainless Steel

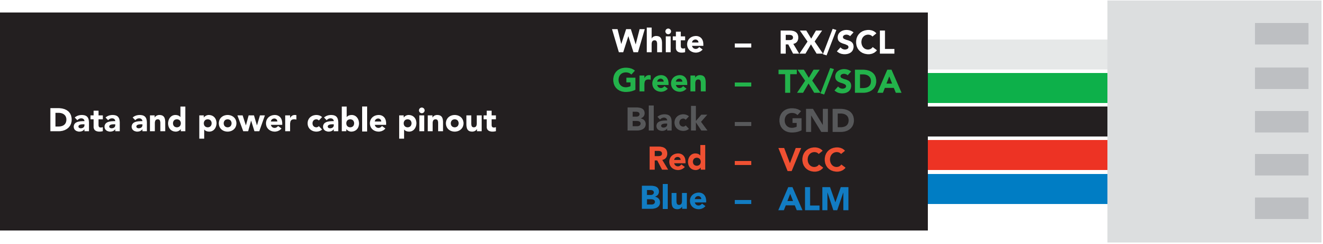
IP60

Sensor properties



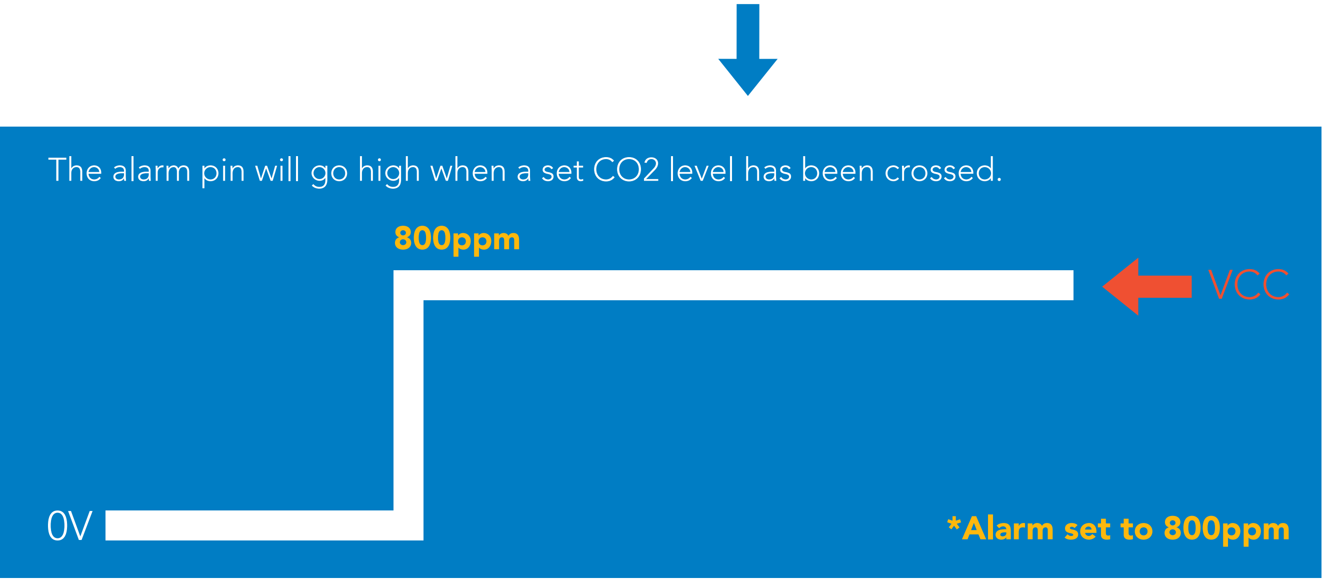


Pin out



Data and power cable pinout

White	–	RX/SCL
Green	–	TX/SDA
Black	–	GND
Red	–	VCC
Blue	–	ALM



The alarm pin will go high when a set CO2 level has been crossed.

*Alarm set to 800ppm

If unused leave **ALM** floating. Do not connect **ALM** to **VCC** or **GND**.

See page **23** to enable CO2 level alarm in UART mode.
See page **46** to enable CO2 level alarm in I2C mode.

Power consumption

	LED	MAX	SLEEP
5V	ON	45 mA	3.4 mA
	OFF	44 mA	
3.3V	ON	42 mA	3.0 mA
	OFF	41 mA	

Absolute max ratings

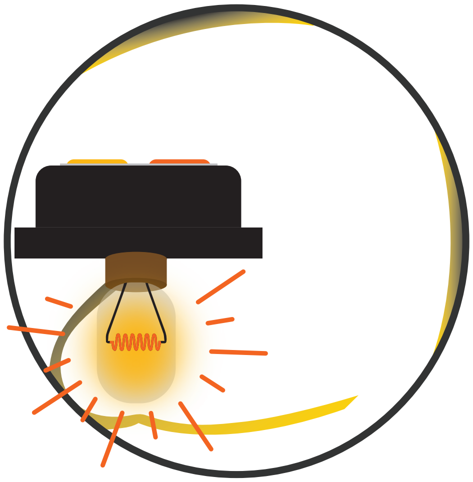
Parameter	MIN	TYP	MAX
Storage temperature	-65 °C		75 °C
Operational temperature	-20 °C	25 °C	50 °C
VCC	3.3V	3.3V	5.5V

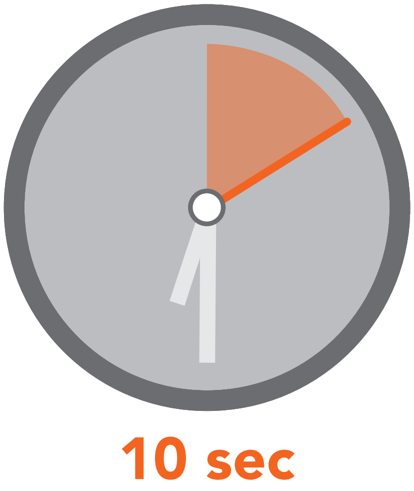
Humidity Range 0 to 95% rh non-condensing

IP60

Sensor warm-up

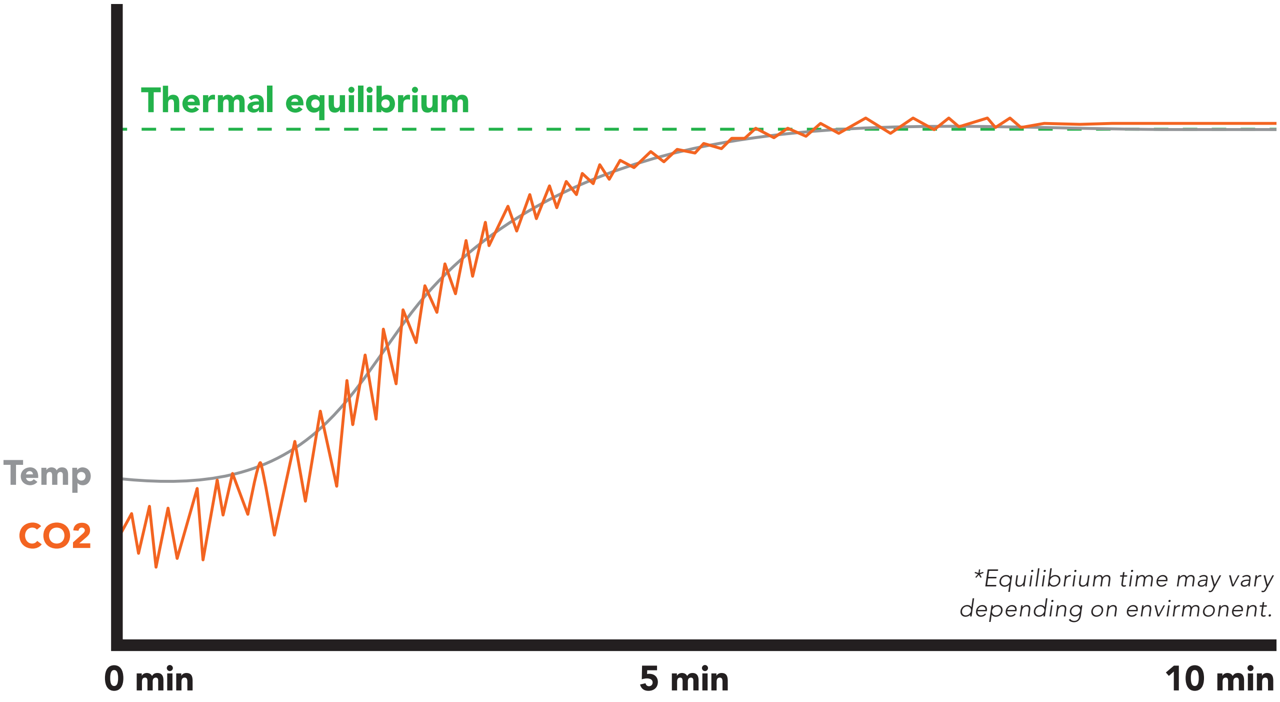
When the Atlas Scientific EZO-CO2™ Embedded CO2 Sensor is first powered on *(or wakes up from sleep mode)* the sensor must warm-up before it can output readings. The warm-up process takes 10 seconds to complete.





During the first 10 seconds of operation the output will be: ***warm**

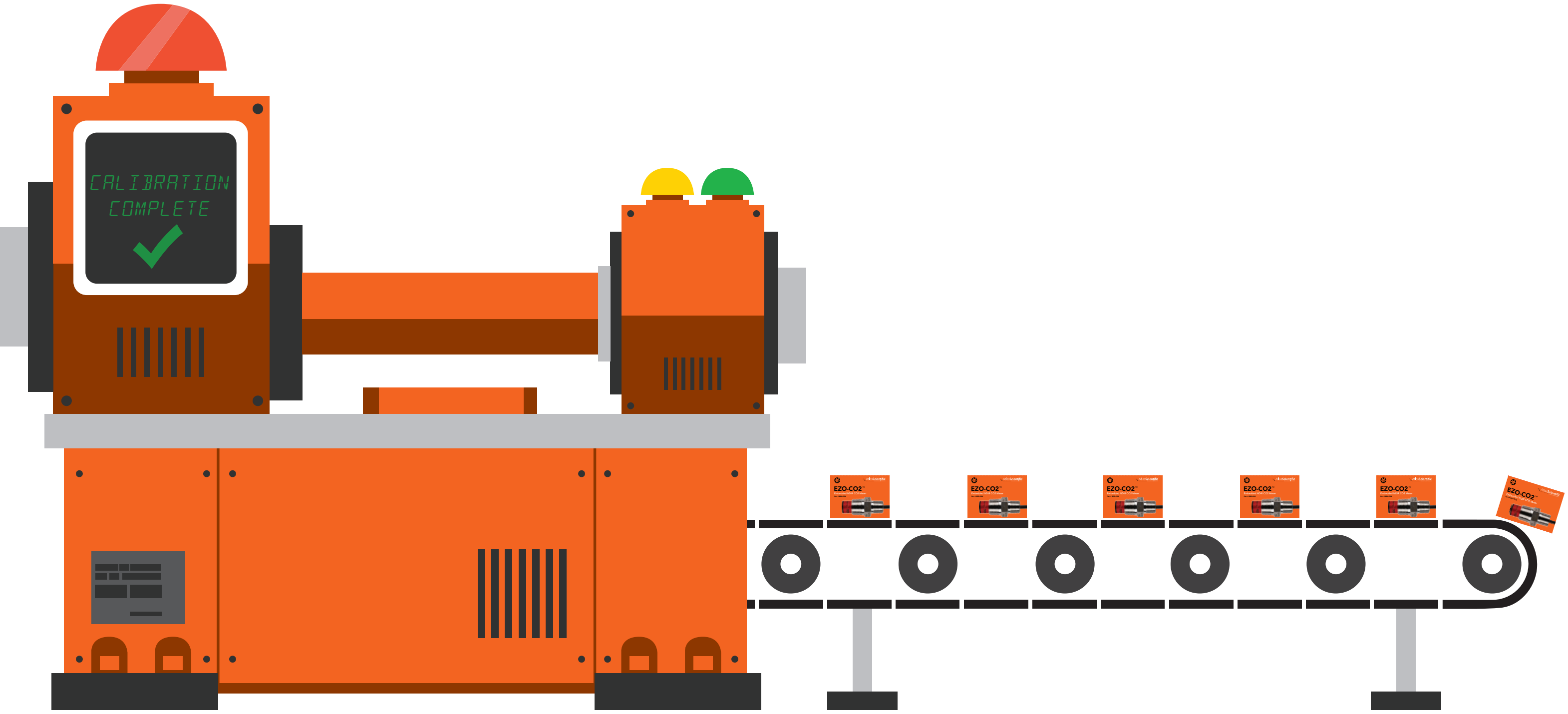
Once warming is finished, CO2 readings will be output. The device will continue to warm-up over several minutes. As the internal temperature stabilizes, so will the CO2 readings.



To see the internal temperature of the sensor and watch as it stabilizes, use the 'O' command found on page 24.

Calibration theory

The Atlas Scientific EZO-CO2™ Embedded CO2 Sensor comes pre-calibrated, and does not need to be recalibrated. Atlas Scientific performs a two-point factory calibration as part of the manufacturing process.



Low point calibration = 0 ppm
High point calibration = 4,000 ppm

The factory calibration data is permanently stored in the sensor and cannot be erased.

Custom calibration

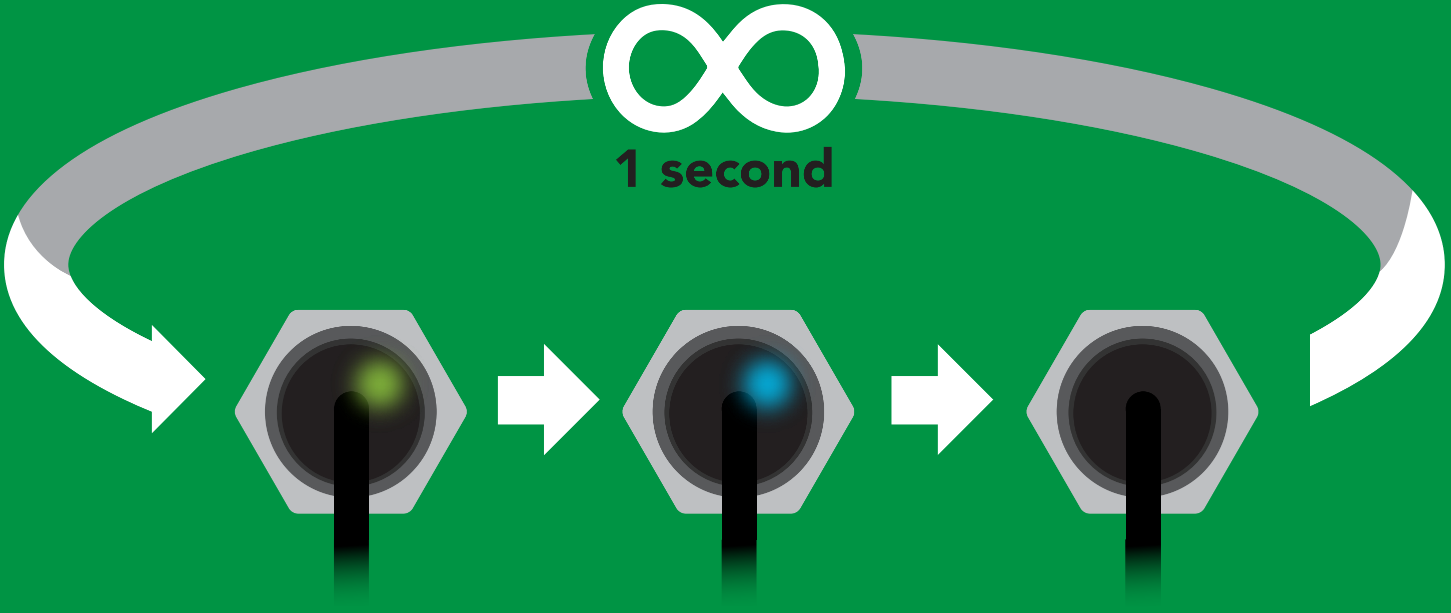
One or two-point calibration can be done at any time. When custom calibration is used, factory calibration will be ignored. To revert back to the factory calibration simply clear the custom calibration.

See page **24** or **47** for custom calibration commands.

Default state

UART mode

Baud	9,600
Readings	continuous
Speed	1 second
LED	on



1 second

Green
Standby

Cyan
Taking reading

Transmitting

✓ Available data protocols

UART

default

I^{2}C

✗ Unavailable data protocols

SPI

Analog

RS-485

Mod Bus

4–20mA

UART mode

Settings that are retained if power is cut

Baud rate
Calibration
Continuous mode
Device name
Enable/disable response codes
Hardware switch to I²C mode
LED control
Protocol lock
Software switch to I²C mode

Settings that are *NOT* retained if power is cut

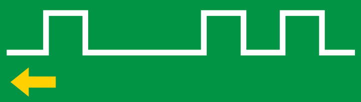
Sleep mode

UART mode

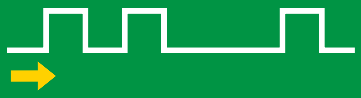
8 data bits	no parity
1 stop bit	no flow control

Baud
300
1,200
2,400
9,600 default
19,200
38,400
57,600
115,200

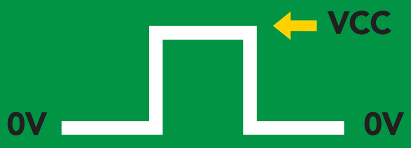
RX
Data in

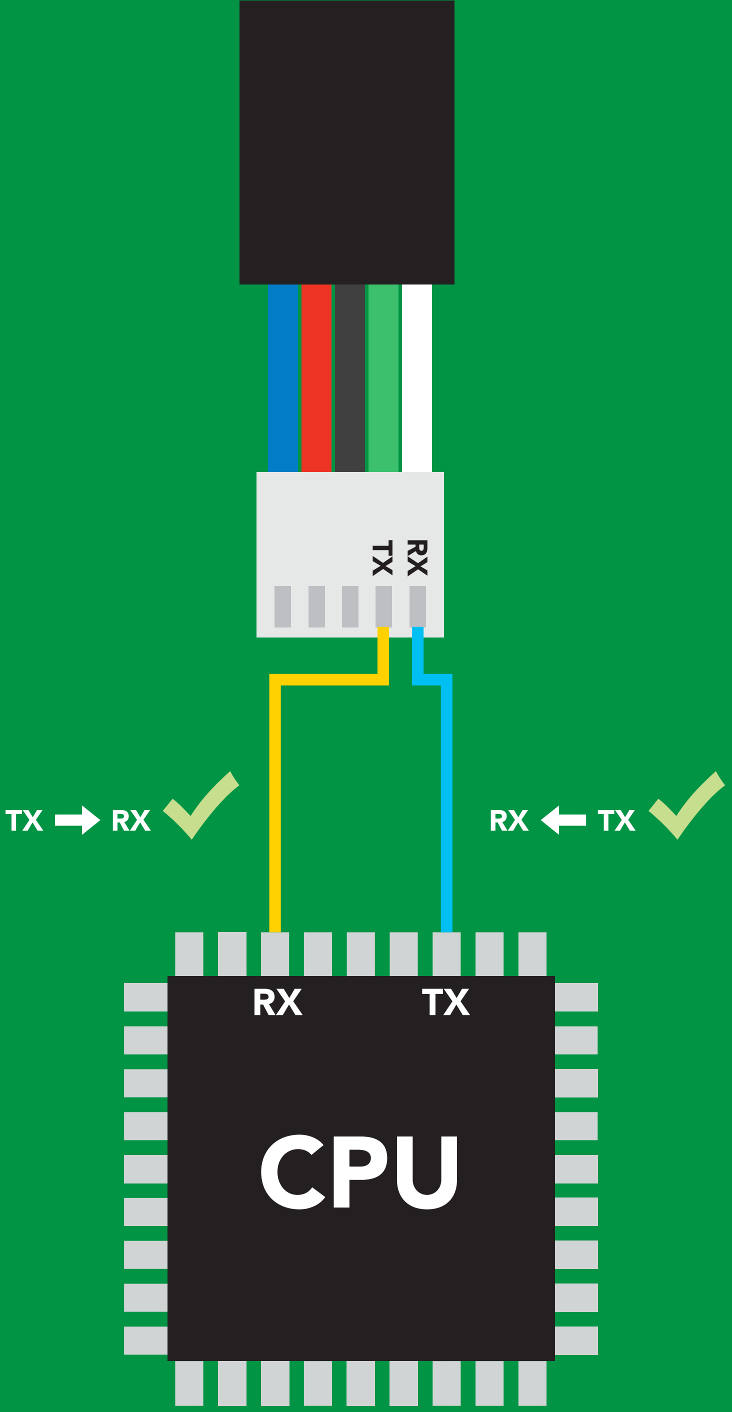


TX
Data out



Vcc 3.3V – 5V





Data format

Reading	Gaseous CO2	Data type	unsigned int
Units	PPM	Decimal places	0
Encoding	ASCII	Smallest string	2 characters
Format	string	Largest string	12 characters
Terminator	carriage return		

Receiving data from device

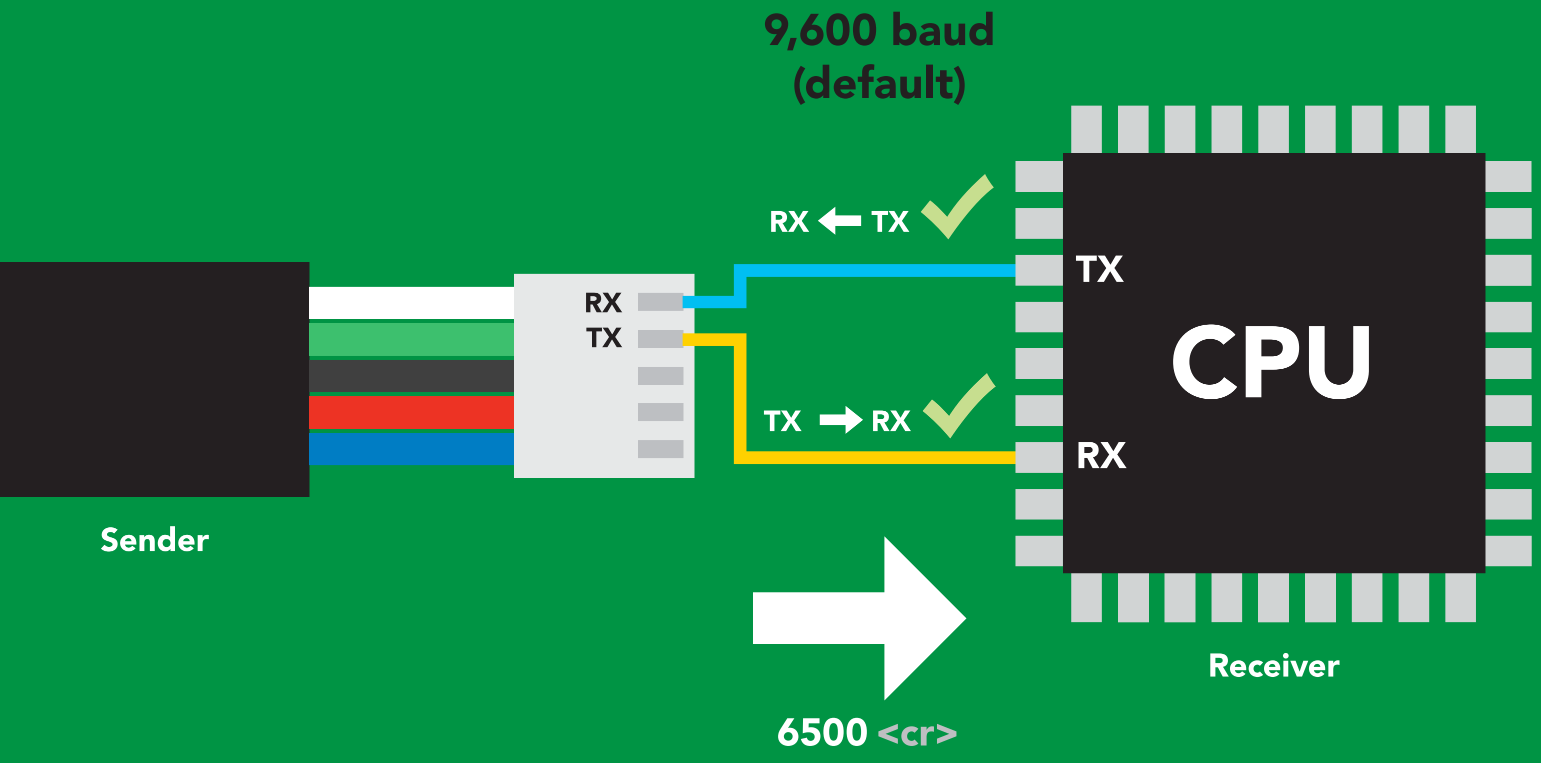
2 parts

ASCII data string

Command

Carriage return <cr>

Terminator



Advanced

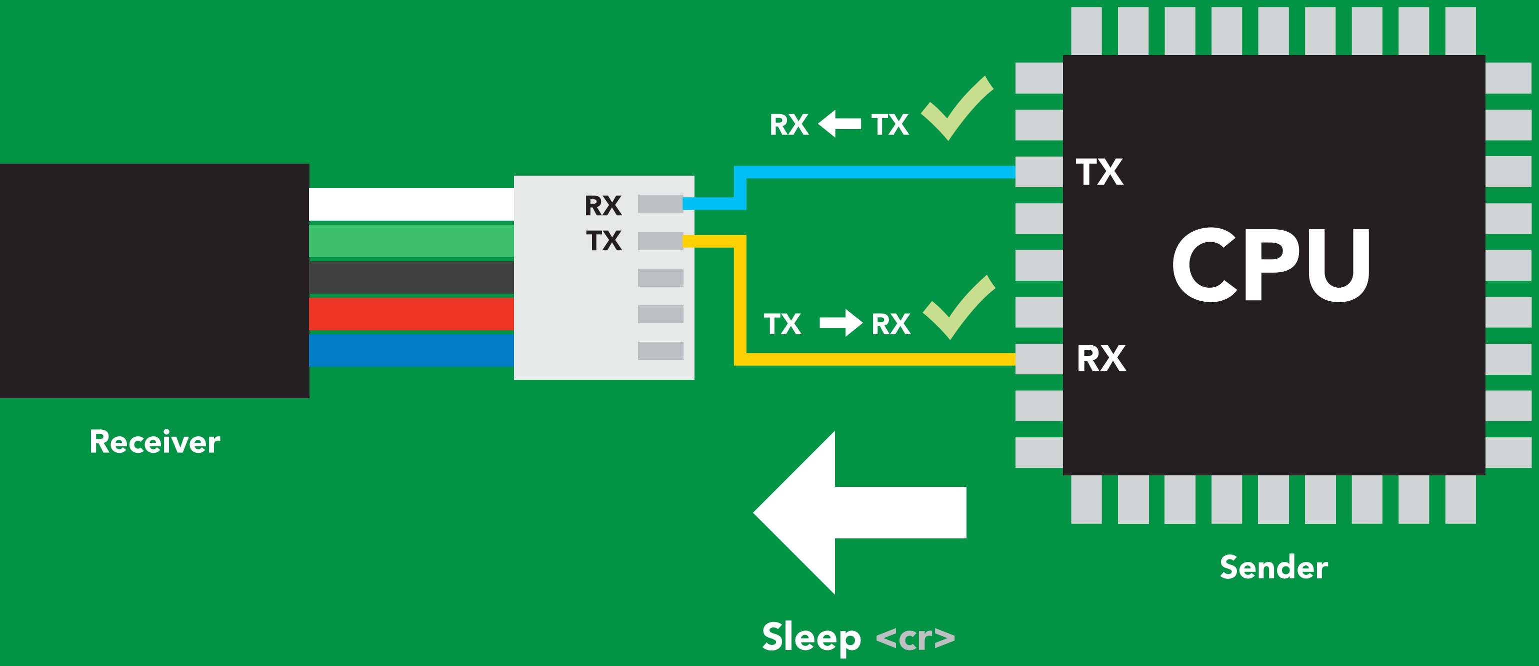
ASCII:	6	5	0	0	<cr>
Hex:	36	35	30	30	0D
Dec:	54	53	48	48	13

Sending commands to device

2 parts

Command (not case sensitive)
ASCII data string

Carriage return <cr>
Terminator



Advanced

ASCII:	S	l	e	e	p	<cr>
Hex:	53	6C	65	65	70	0D
Dec:	83	108	101	101	112	13

LED color definition

Green
UART standby

Cyan
Taking reading

Purple
Changing baud rate

Red
Command not understood

White
Find

	LED ON
5V	**+2.5 mA**
3.3V	**+1 mA**

UART mode

command quick reference

All commands are ASCII strings or single ASCII characters.

Command	Function		Default state
Alarm	enable/disable alarm	pg. 23	n/a
Baud	change baud rate	pg. 31	9,600
C	enable/disable continuous mode	pg. 21	enabled
Cal	performs custom calibration	pg. 24	n/a
Factory	enable factory reset	pg. 33	n/a
Find	finds device with blinking white LED	pg. 20	n/a
i	device information	pg. 27	n/a
I2C	change to I^{2}C mode	pg. 34	not set
L	enable/disable LED	pg. 19	enabled
Name	set/show name of device	pg. 26	not set
O	enable/disable internal temperature	pg. 25	disabled
Plock	enable/disable protocol lock	pg. 32	n/a
R	returns a single reading	pg. 22	n/a
Sleep	enter sleep mode/low power	pg. 30	n/a
Status	retrieve Status Information	pg. 29	n/a
*OK	enable/disable response codes	pg. 28	n/a

LED control

Command syntax

```
L,1  <cr>  LED on  default
L,0  <cr>  LED off
L,?  <cr>  LED state on/off?
```

Example	Response
L,1 <cr>	*OK <cr>
L,0 <cr>	*OK <cr>
L,? <cr>	?L,1 <cr> or ?L,0 <cr> *OK <cr>

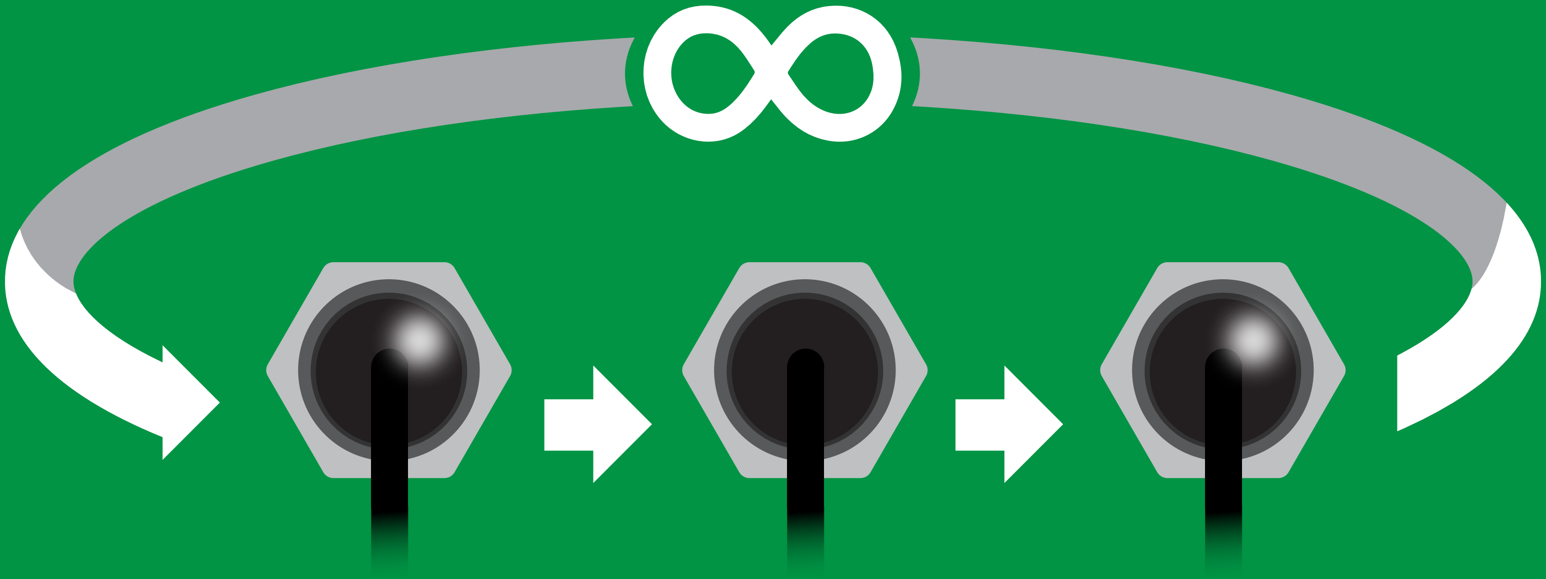
Find

Command syntax

This command will disable continuous mode
Send any character or command to terminate find.

Find <cr> LED rapidly blinks white, used to help find device

Example	Response
Find <cr>	*OK <cr>



Continuous mode

Command syntax

C,1	<cr>	enable continuous readings once per second	default
C,n	<cr>	continuous readings every n seconds (n = 2 to 99 sec)	
C,0	<cr>	disable continuous readings	
C,?	<cr>	continuous reading mode on/off?	

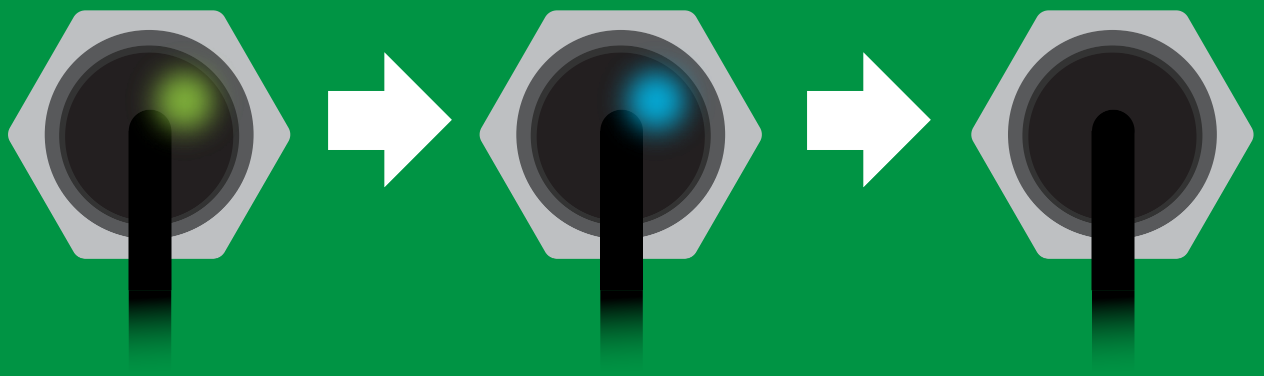
Example	Response
C,1 <cr>	*OK <cr> CO2 (1 sec) <cr> CO2 (2 sec) <cr> CO2 (n sec) <cr>
C,30 <cr>	*OK <cr> CO2 (30 sec) <cr> CO2 (60 sec) <cr> CO2 (90 sec) <cr>
C,0 <cr>	*OK <cr>
C,? <cr>	?C,1 <cr> or ?C,0 <cr> or ?C,30 <cr> *OK <cr>

Single reading mode

Command syntax

R <cr> takes single reading

Example	Response
R <cr>	6500 <cr> *OK <cr>



Green
Standby

Cyan
Taking reading

Transmitting

1 second

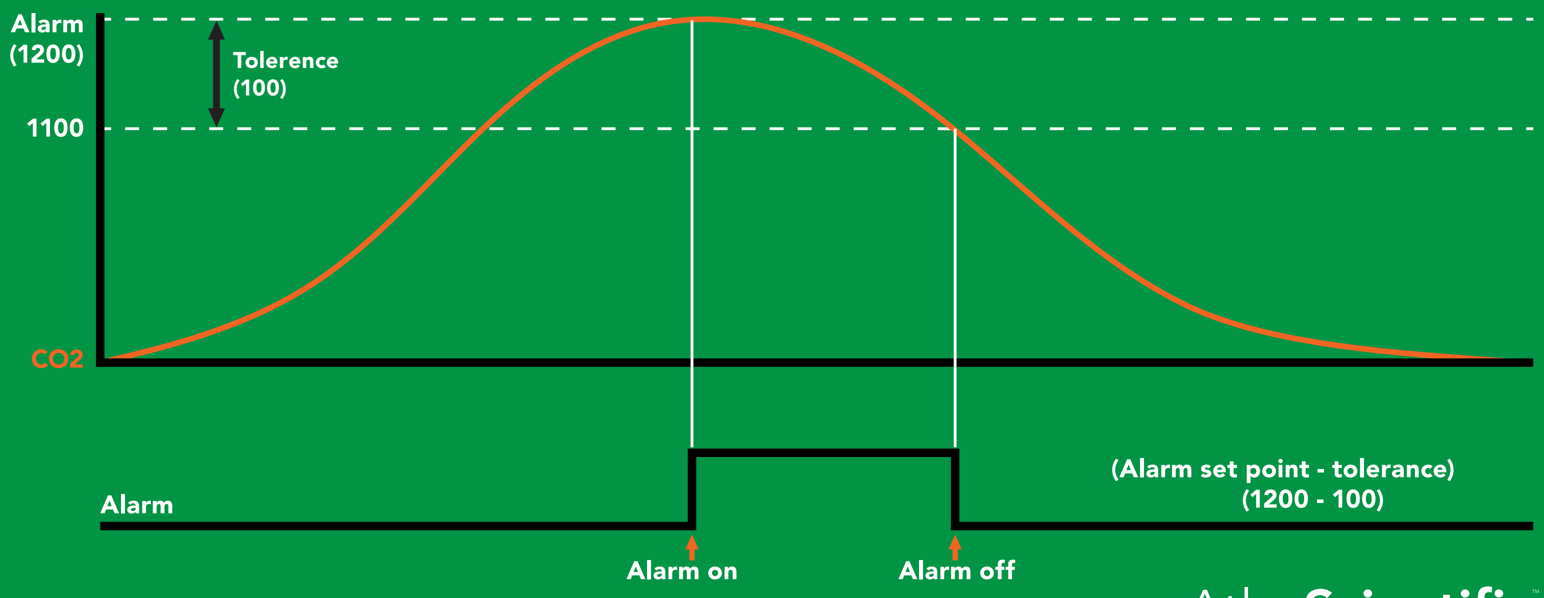
Alarm

Command syntax

The alarm pin will = 1 when CO2 levels are > alarm set point. Alarm tolerance sets how far below the set point CO2 levels need to drop before the pin will = 0 again.

Alarm,en,[1,0]	<cr>	enable / disable alarm
Alarm,n	<cr>	sets alarm
Alarm,tol,n	<cr>	sets alarm tolerance (0-500 ppm)
Alarm,?	<cr>	alarm set?

Example	Response
Alarm,en,1 <cr>	*OK <cr> Enable alarm
Alarm,1200 <cr>	*OK <cr>
Alarm,tol,100 <cr>	*OK <cr> CO2 level must fall 100 ppm below set point for alarm to reset.
Alarm,? <cr>	?,alarm,1200,100,1 <cr> if all are enabled



Custom calibration

Command syntax

High point calibration can be from 3,000 ppm to 5,000 ppm. Calibration outside of that range my lead to accuracy issues.

Cal,n	<cr>	calibrates the high point
Cal,0	<cr>	calibrates the zero point
Cal,clear	<cr>	restores calibration to factory settings
Cal,?	<cr>	device calibrated?

Example	Response
Cal,3900 <cr>	*OK <cr>
Cal,0 <cr>	*OK <cr>
Cal,clear <cr>	*OK <cr>
Cal,? <cr>	?Cal,0 <cr> or ?Cal,1 <cr> or ?Cal,2 <cr> or no calibration only zero point calibration only high point calibration ?Cal,3 <cr> *OK <cr> zero and high point calibration

This device comes pre-calibrated.

Custom calibration should not be performed without scientific grade calibration gasses.

Enable/disable internal temperature from output string

Command syntax

O,t,[1,0]	<cr>	enable or disable internal temperature

Example	Response
O,t,1 <cr>	*OK <cr> enable temperature
O,t,0 <cr>	*OK <cr> disable temperature
O,? <cr>	?O,ppm,t <cr> if internal temp is enabled

Enabling the internal temperature should only be used to confirm that the device is at thermal equilibrium. Refer to page 6

Naming device

Command syntax

Do not use spaces in the name

Name,n	<cr>	set name
Name,	<cr>	clears name
Name,?	<cr>	show name

n = _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

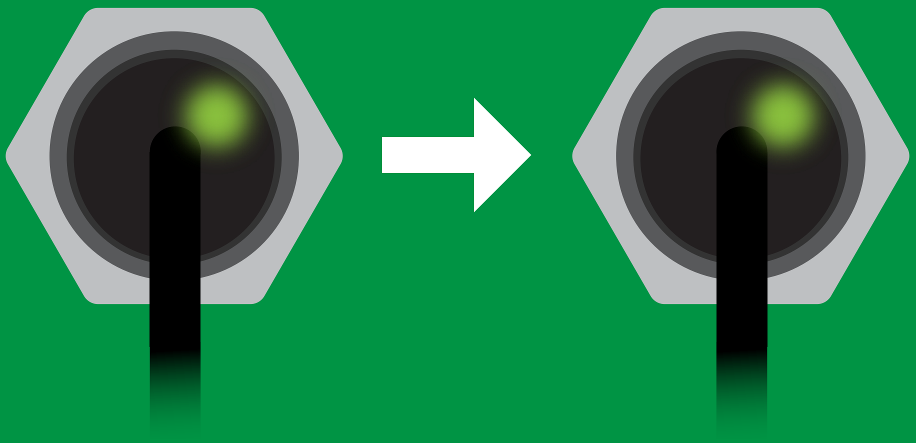
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Up to 16 ASCII characters

Example	Response
Name, <cr>	*OK <cr> name has been cleared
Name,zzt <cr>	*OK <cr>
Name,? <cr>	?Name,zzt <cr> *OK <cr>

Name,zzt

Name,?



*OK <cr>

?Name,zzt <cr>
*OK <cr>

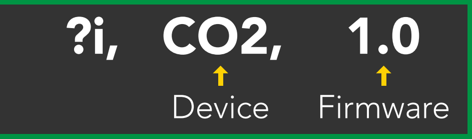
Device information

Command syntax

i <cr> **device information**

Example	Response
i <cr>	?i,CO2,1.0 <cr> *OK <cr>

Response breakdown



?i, CO2, 1.0

Device

Firmware

Response codes

Command syntax

*OK,1	<cr>	enable response	default
*OK,0	<cr>	disable response	
*OK,?	<cr>	response on/off?	

Example	Response
R <cr>	6,500 <cr> *OK <cr>
*OK,0 <cr>	no response, *OK disabled
R <cr>	6,500 <cr> *OK disabled
*OK,? <cr>	?*OK,1 <cr> or ?*OK,0 <cr>

Other response codes

*ER	unknown command
*OV	over volt (VCC>=5.5V)
*UV	under volt (VCC<=3.1V)
*RS	reset
*RE	boot up complete, ready
*SL	entering sleep mode
*WA	wake up

These response codes cannot be disabled

Reading device status

Command syntax

Status <cr> **voltage at Vcc pin and reason for last restart**

Example	Response
Status <cr>	**?Status,P,5.038** <cr> ***OK** <cr>

Response breakdown

?Status,	P,	5.038
	Reason for restart	Voltage at Vcc

Restart codes

P	powered off
S	software reset
B	brown out
W	watchdog
U	unknown

Sleep mode/low power

Command syntax

Send any character or command to awaken device.

Sleep <cr> enter sleep mode/low power

Example	Response
Sleep <cr>	*OK <cr> *SL <cr>
Any command	*WA <cr> wakes up device

	MAX	SLEEP
5V	45 mA	3.4 mA
3.3V	42 mA	3.0 mA

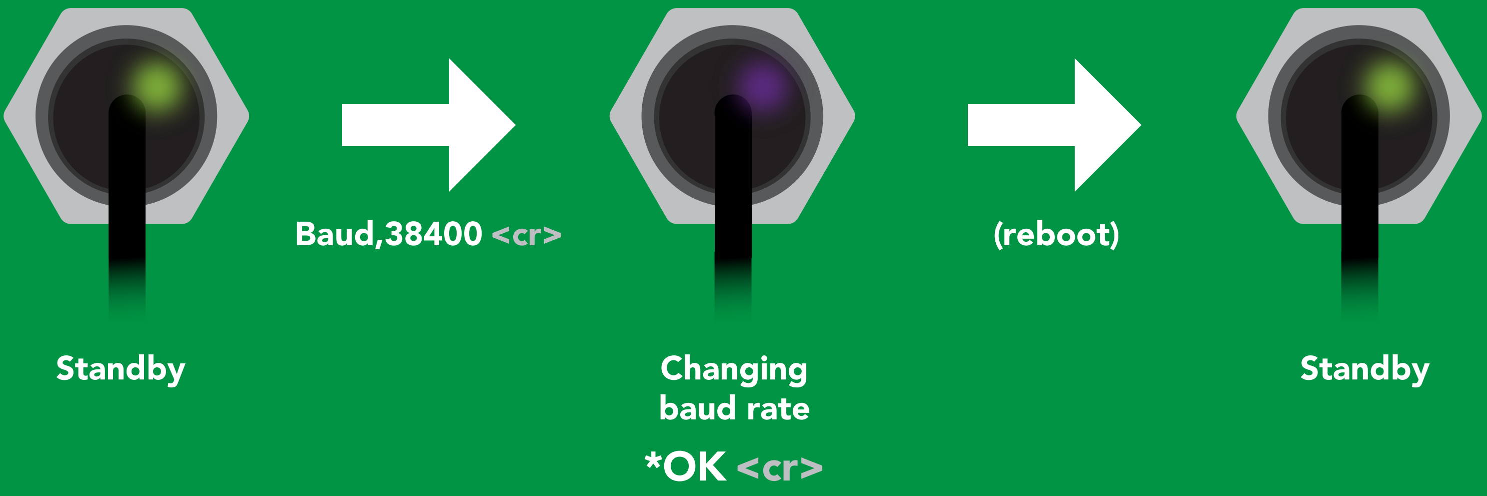
Change baud rate

Command syntax

Baud,n <cr> **change baud rate**

Example	Response
Baud,38400 <cr>	*OK <cr>
Baud,? <cr>	?Baud,38400 <cr> *OK <cr>

n =
- 300
- 1200
- 2400
- 9600 default
- 19200
- 38400
- 57600
- 115200



Protocol lock

Command syntax

Locks device to UART mode.

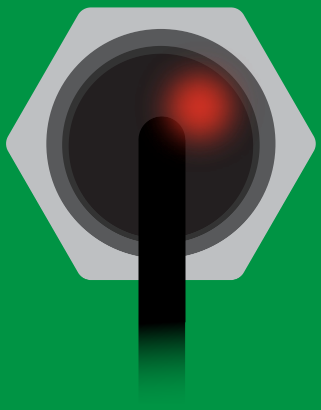
Plock,1	<cr>	enable Plock	
Plock,0	<cr>	disable Plock	default
Plock,?	<cr>	Plock on/off?	

Example	Response
Plock,1 <cr>	*OK <cr>
Plock,0 <cr>	*OK <cr>
Plock,? <cr>	?Plock,1 <cr> or ?Plock,0 <cr>

Plock,1

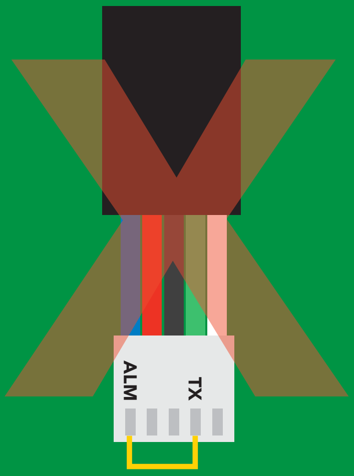
*OK <cr>

I2C,100



cannot change to I^2C

*ER <cr>



cannot change to I^2C

Factory reset

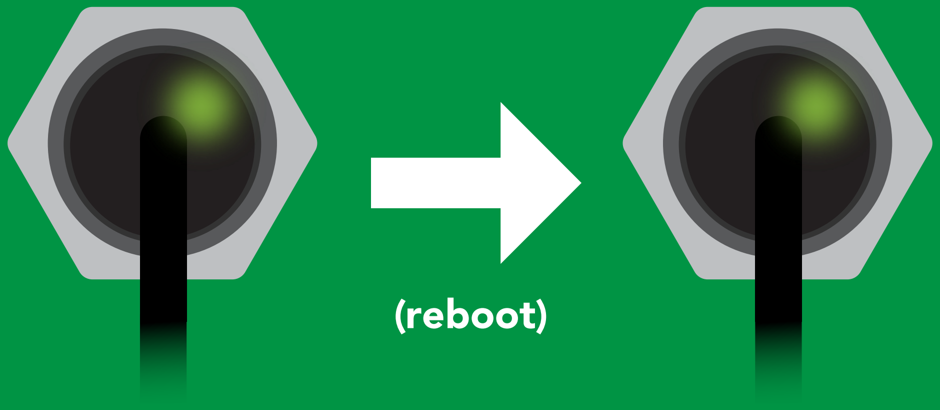
Command syntax

Clears custom calibration
"*OK" enabled

Factory <cr> enable factory reset

Example	Response
Factory <cr>	*OK <cr>

Factory <cr>



(reboot)

*OK <cr>

*RS <cr>
*RE <cr>

Baud rate will not change

Change to I^2C mode

Command syntax

Default I^2C address 105 (0x69)

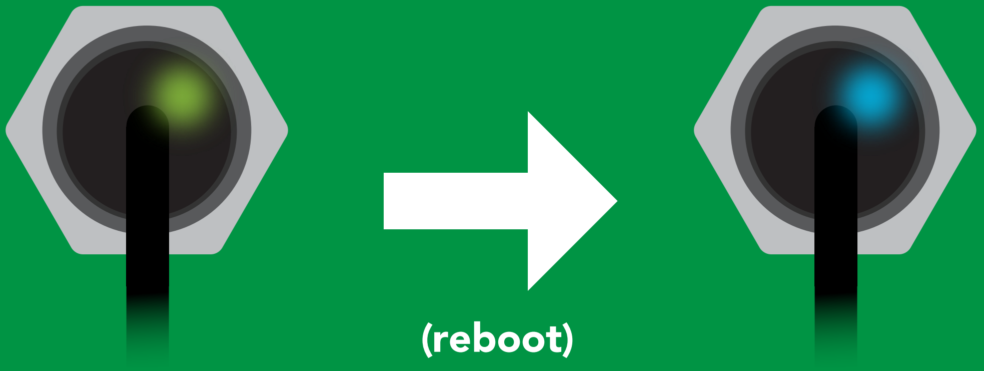
I2C,n <cr> sets I^2C address and reboots into I^2C mode

n = any number 1 – 127

Example	Response
I2C,100 <cr>	*OK (reboot in I^2C mode)

Wrong example	Response
I2C,139 <cr> n ≯ 127	*ER <cr>

I2C,100



(reboot)

Green
*OK <cr>

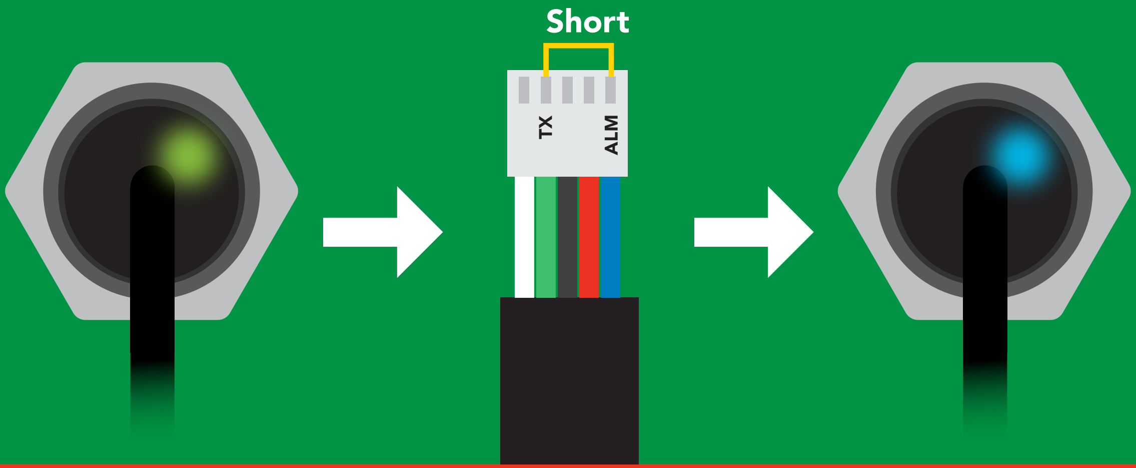
Blue
now in I^2C mode

Manual switching to I²C

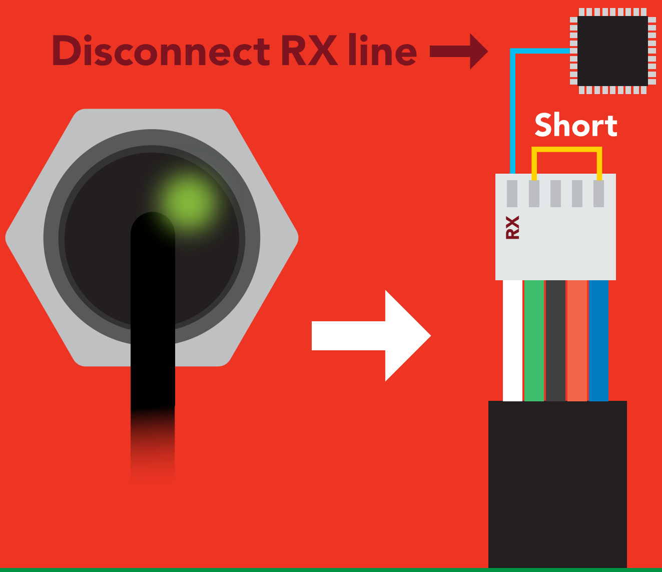
- Disconnect ground (power off)
- Disconnect TX and RX
- Connect TX to ALM
- Confirm RX is disconnected
- Connect ground (power on)
- Wait for LED to change from Green to Blue
- Disconnect ground (power off)
- Reconnect all data and power

Manually switching to I²C will set the I²C address to 105 (0x69)

Example



Wrong Example



I^{2}C mode

The I^{2}C protocol is *considerably more complex* than the UART (RS–232) protocol. Atlas Scientific assumes the embedded systems engineer understands this protocol.

To set your EZO™ device into I^{2}C mode click here

Settings that are retained if power is cut

Calibration
Change I^{2}C address
Hardware switch to UART mode
LED control
Protocol lock
Software switch to UART mode

Settings that are *NOT* retained if power is cut

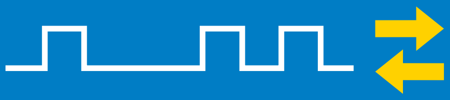
Sleep mode

AtlasScientific™
Environmental Robotics

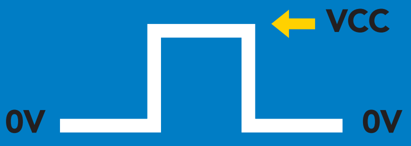
I^{2}C mode

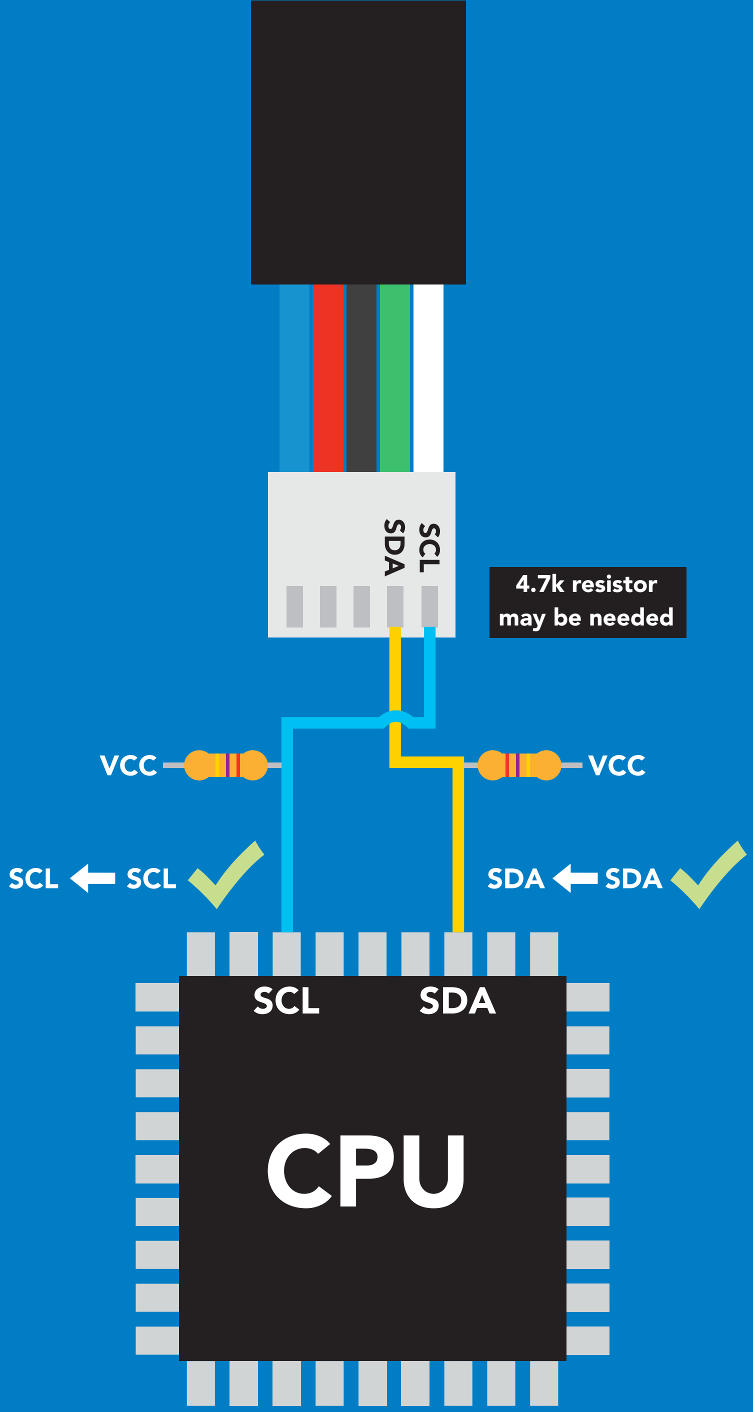
I^{2}C address	(0x01 – 0x7F) 105 (0x69) **default**
Vcc	3.3V – 5.5V
Clock speed	100 – 400 kHz

SDA



SCL

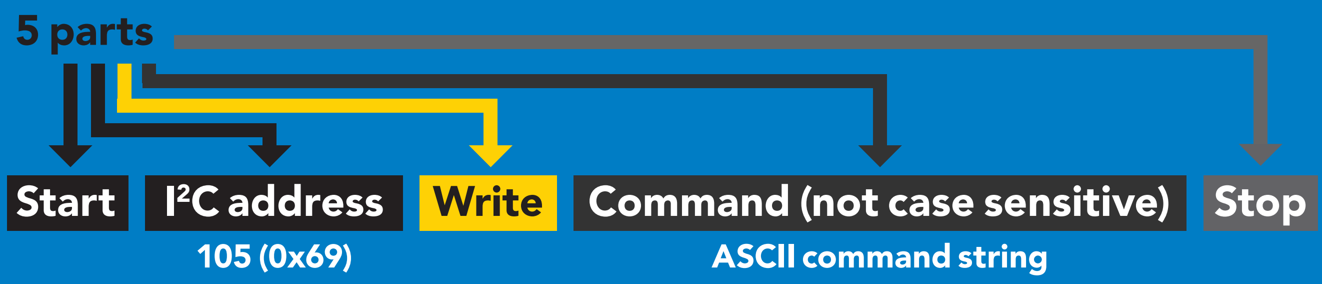




Data format

Reading	Gaseous CO2	Data type	unsigned int
Units	PPM	Decimal places	0
Encoding	ASCII	Smallest string	2 characters
Format	string	Largest string	12 characters

Sending commands to device

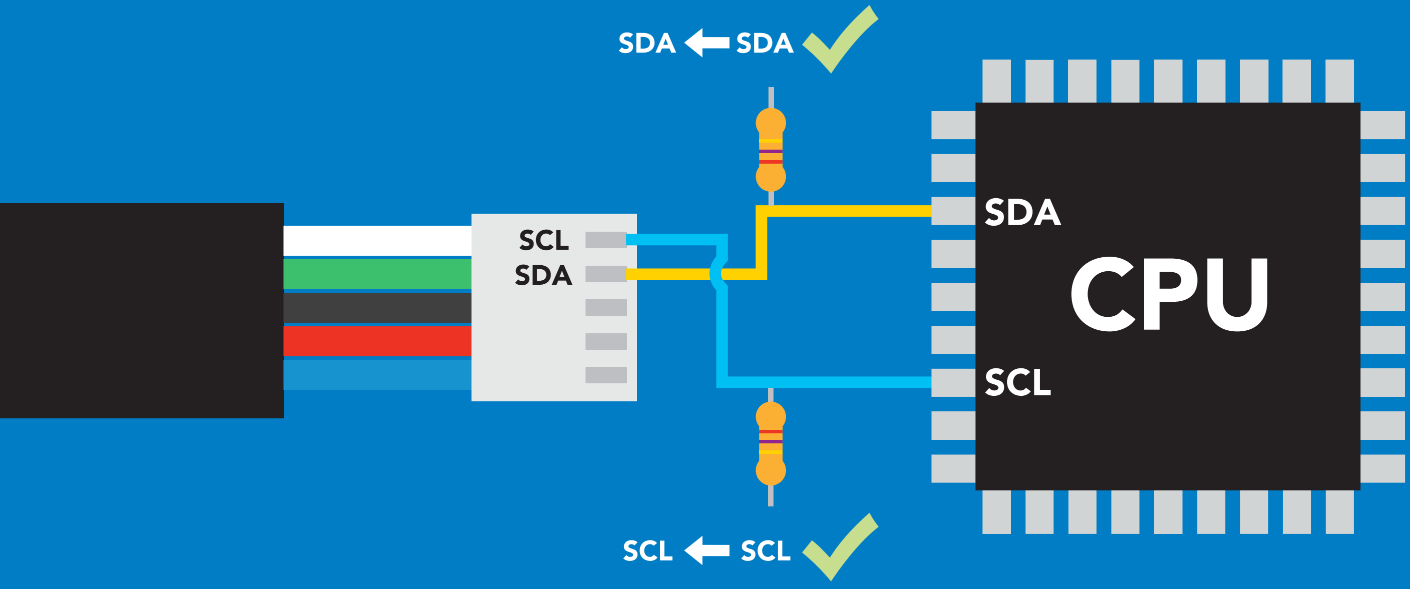


5 parts

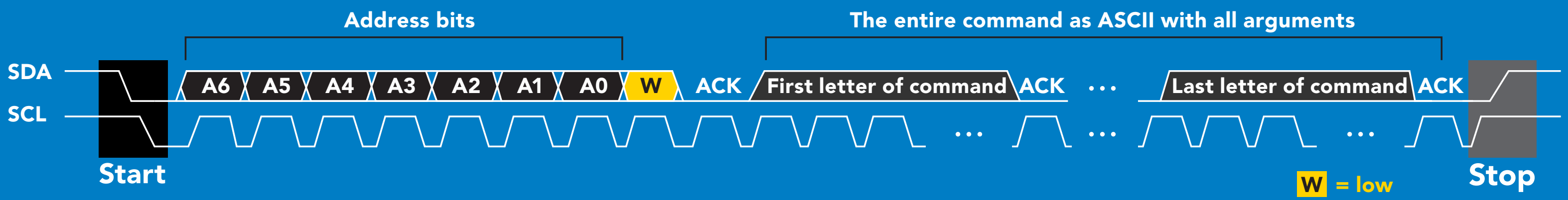
Start | I²C address 105 (0x69) | Write | Command (not case sensitive) ASCII command string | Stop

Example

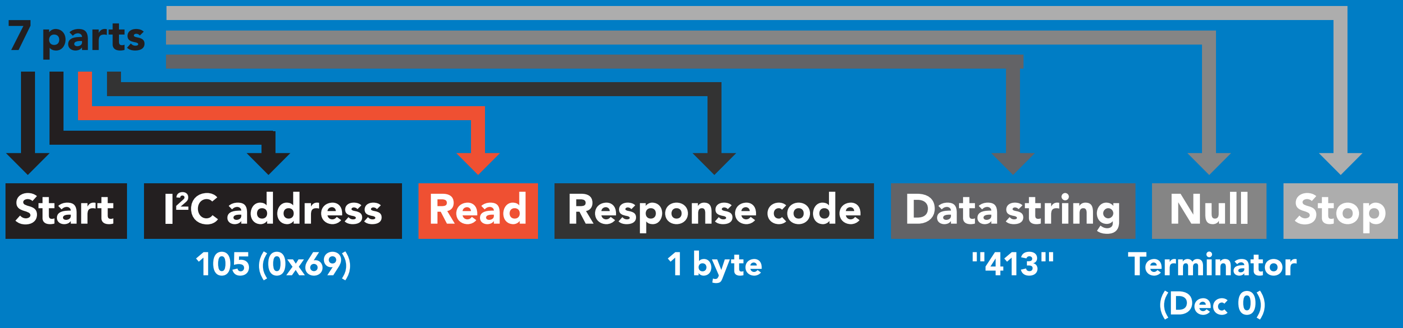
Start | 105 (0x69) I²C address | Write | Sleep Command | Stop



Advanced



Requesting data from device



7 parts

Start

I²C address
105 (0x69)

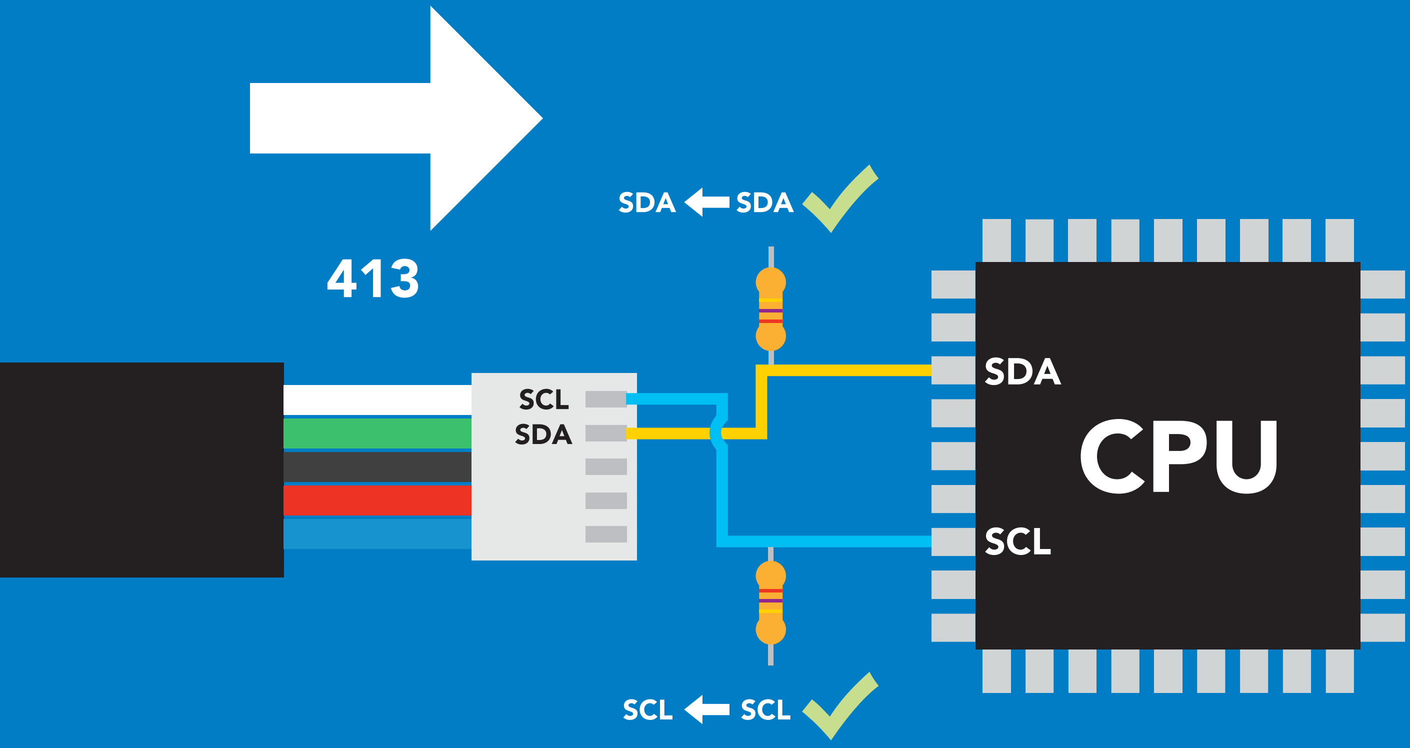
Read

Response code
1 byte

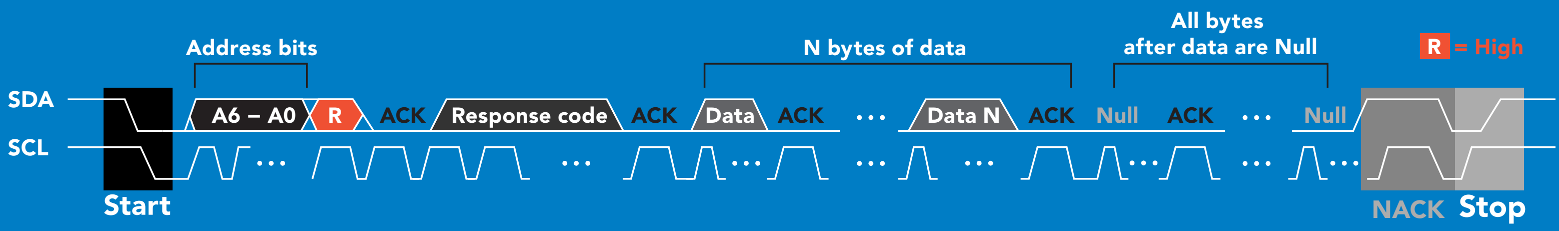
Data string
"413"

Null
Terminator
(Dec 0)

Stop



Advanced



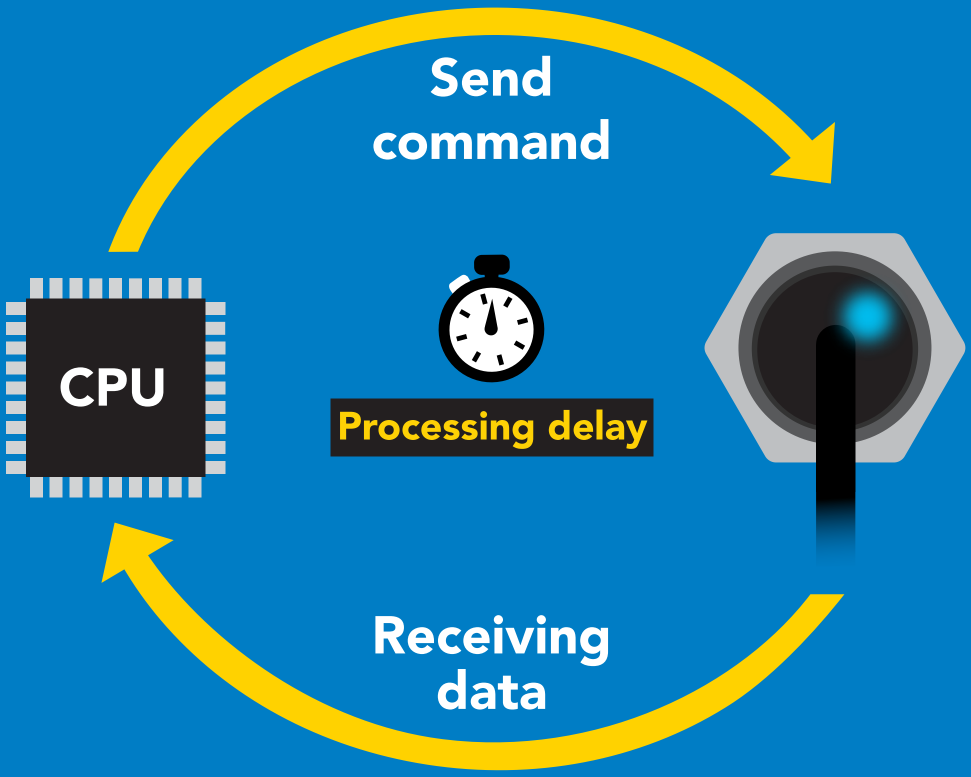
1 | 52 | 49 | 51 | 0 = 413

Dec | ASCII | Dec

Response codes & processing delay

After a command has been issued, a 1 byte response code can be read in order to confirm that the command was processed successfully.

Reading back the response code is completely optional, and is not required for normal operation.



Send command

CPU

Processing delay

Receiving data

Example

```
I2C_start;
I2C_address;
I2C_write(EZO_command);
I2C_stop;

delay(300);   Processing delay

I2C_start;
I2C_address;
Char[ ] = I2C_read;
I2C_stop;
```

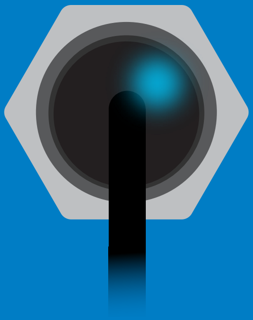
If there is no processing delay or the processing delay is too short, the response code will always be 254.

Response codes

Single byte, not string

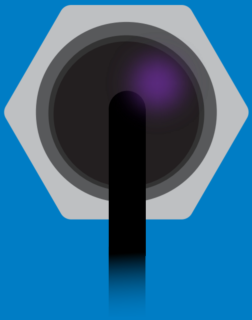
255	no data to send
254	still processing, not ready
2	syntax error
1	successful request

LED color definition

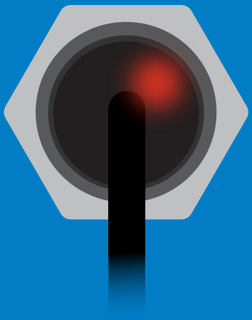


Blue
I^2C standby

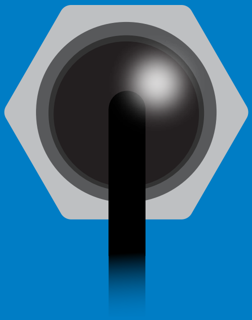
Green
Taking reading



Purple
Changing I^2C address



Red
Command not understood



White
Find

	LED ON
5V	**+2.5 mA**
3.3V	**+1 mA**

I^{2}C mode
command quick reference

All commands are ASCII strings or single ASCII characters.

Command	Function	
Alarm	enable/disable alarm	pg. 46
Baud	switch back to UART mode	pg. 56
Cal	performs custom calibration	pg. 47
Factory	enable factory reset	pg. 55
Find	finds device with blinking white LED	pg. 44
i	device information	pg. 50
I2C	change I^{2}C address	pg. 54
L	enable/disable LED	pg. 43
Name	set/show name of device	pg. 49
O	enable/disable internal temp	pg. 48
Plock	enable/disable protocol lock	pg. 57
R	returns a single reading	pg. 45
Sleep	enter sleep mode/low power	pg. 52
Status	retrieve status information	pg. 51

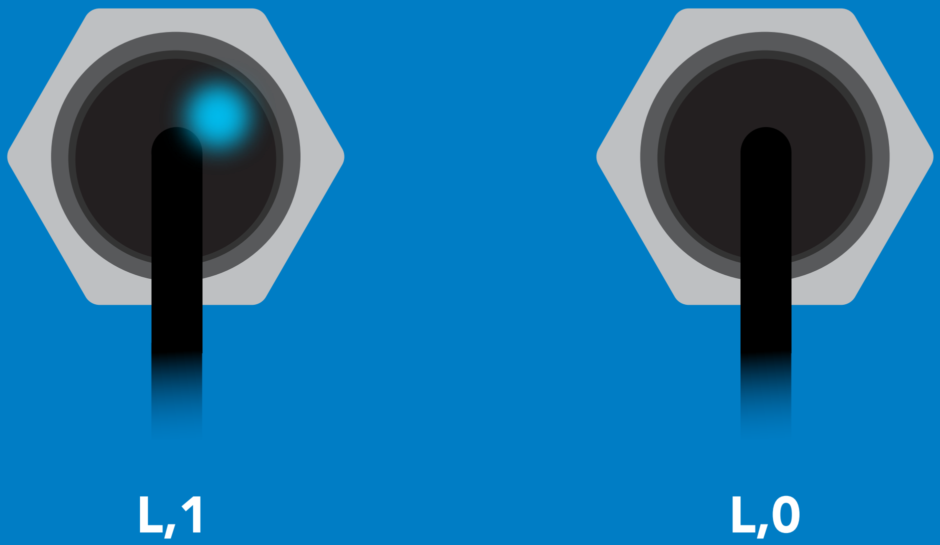
LED control

Command syntax

300ms ⏱ processing delay

L,1	LED on	default
L,0	LED off	
L,?	LED state on/off?	

Example	Response
L,1	⏱ Wait 300ms · 1 Dec · 0 Null
L,0	⏱ Wait 300ms · 1 Dec · 0 Null
L,?	⏱ Wait 300ms · 1 Dec · ?L,1 ASCII · 0 Null **or** ⏱ Wait 300ms · 1 Dec · ?L,0 ASCII · 0 Null



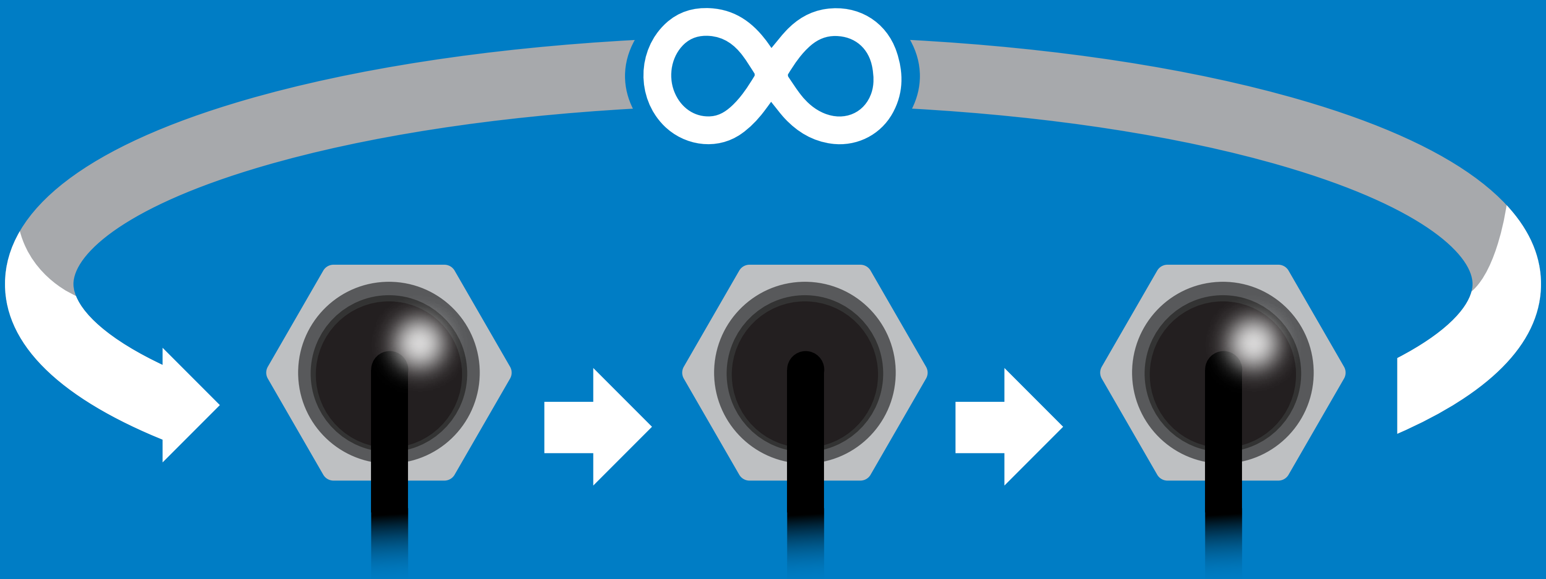
Find

300ms ⏱ processing delay

Command syntax

Find	LED rapidly blinks white, used to help find device

Example	Response
Find	⏱ Wait 300ms 1 Dec 0 Null



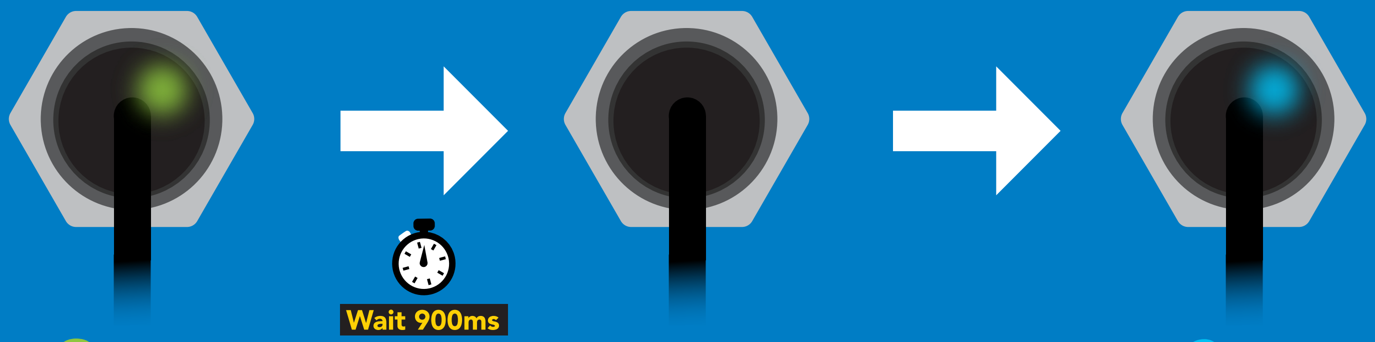
Taking reading

Command syntax

900ms ⏱ processing delay

R return 1 reading

Example	Response
R	⏱ Wait 900ms 1 Dec 800 ASCII 0 Null



Green
Taking reading

Transmitting

Cyan
Standby

Alarm

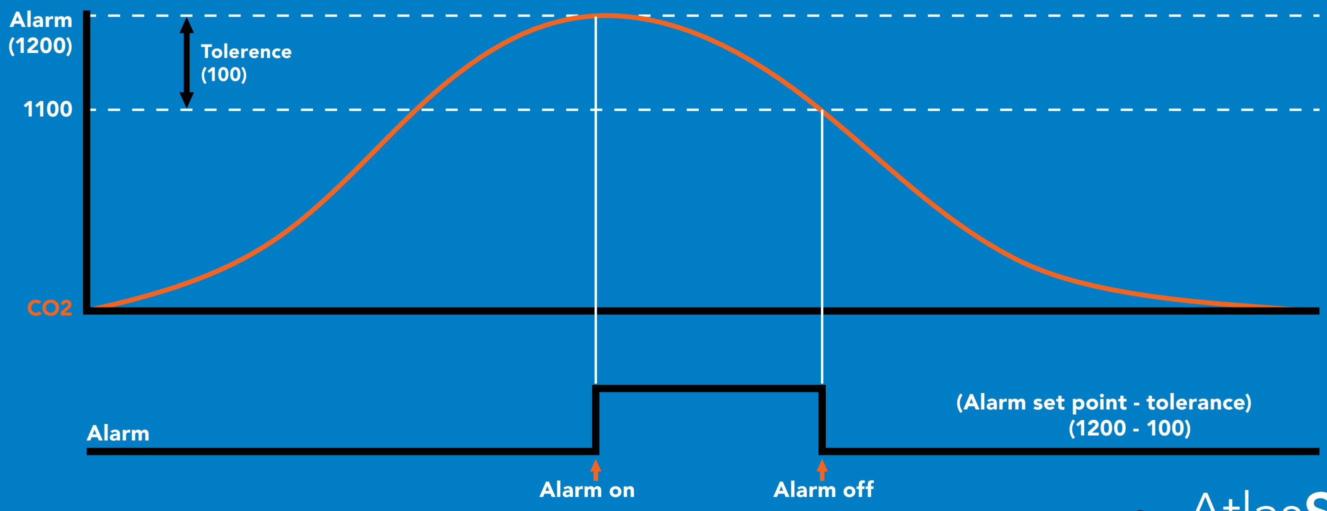
300ms ⏱ processing delay

Command syntax

The alarm pin will = 1 when CO2 levels are > alarm set point. Alarm tolerance sets how far below the set point CO2 levels need to drop before the pin will = 0 again.

Alarm,en,[1,0]	enable / disable alarm
Alarm,n	sets alarm
Alarm,tol,n	sets alarm tolerance (0-500 ppm)
Alarm,?	alarm set?

Example	Response
Alarm,en,1	⏱ Wait 300ms · 1 Dec · 0 Null · Enable alarm
Alarm,1200	⏱ Wait 300ms · 1 Dec · 0 Null
Alarm,tol,100	⏱ Wait 300ms · 1 Dec · 0 Null · CO2 level must fall 100 ppm below set point for alarm to reset.
Alarm,?	⏱ Wait 300ms · 1 Dec · ?,alarm,1200,100,1 ASCII · 0 Null · if all are enabled



Custom calibration

900ms ⏱ processing delay

Command syntax

High point calibration can be from 3,000 ppm to 5,000 ppm. Calibration outside of that range my lead to accuracy issues.

Cal,n	calibrates the high point
Cal,0	calibrates the zero point
Cal,clear	restores calibration to factory settings
Cal,?	device calibrated?

Example	Response
Cal,3900	⏱ Wait 900ms 1 Dec 0 Null
Cal,0	⏱ Wait 900ms 1 Dec 0 Null
Cal,clear	⏱ Wait 300ms 1 Dec 0 Null
Cal,?	⏱ Wait 300ms 1 Dec ?Cal,0 ASCII 0 Null no calibration or 1 Dec ?Cal,1 ASCII 0 Null only zero point calibration or 1 Dec ?Cal,2 ASCII 0 Null only high point calibration or 1 Dec ?Cal,3 ASCII 0 Null zero and high point calibration

This device comes pre-calibrated.

Custom calibration should not be performed without scientific grade calibration gasses.

Enable/disable internal temperature from output string

Command syntax

300ms ⏱ processing delay

O,t,[1,0]	enable or disable internal temperature

Example	Response	
O,t,1	⏱ Wait 300ms 1 Dec 0 Null	enable temperature
O,t,0	⏱ Wait 300ms 1 Dec 0 Null	disable temperature
O,?	⏱ Wait 300ms 1 Dec ?O,ppm,t ASCII 0 Null	if internal temp is enabled

Enabling the internal temperature should only be used to confirm that the device is at thermal equilibrium. Refer to page 6

Naming device

300ms ⏱ processing delay

Command syntax

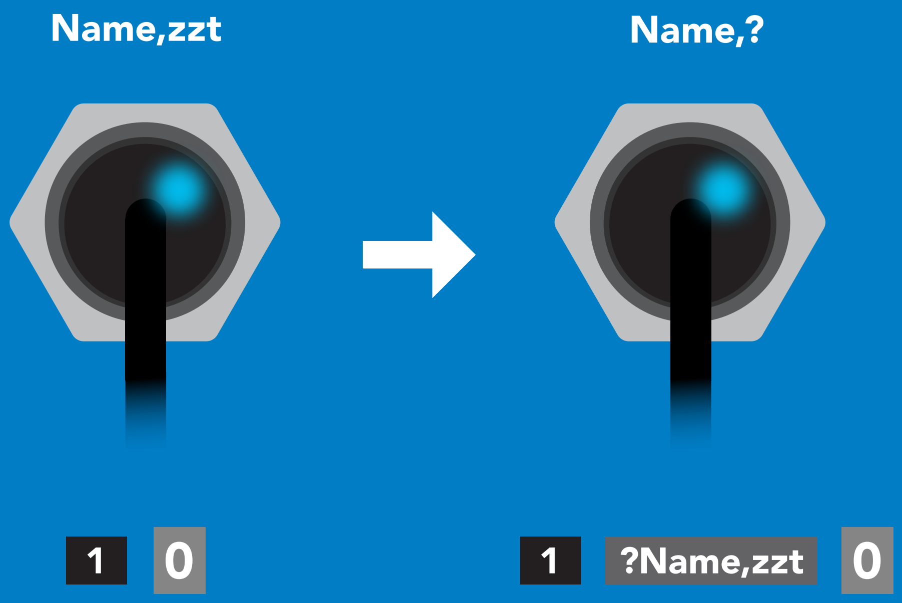
Do not use spaces in the name

Name,n	set name
Name,	clears name
Name,?	show name

n = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Up to 16 ASCII characters

Example	Response
Name,	⏱ Wait 300ms · 1 Dec · 0 Null · name has been cleared
Name,zzt	⏱ Wait 300ms · 1 Dec · 0 Null
Name,?	⏱ Wait 300ms · 1 Dec · ?Name,zzt ASCII · 0 Null



Device information

Command syntax

300ms ⏱ processing delay

i device information

Example

i

Response

⏱
Wait 300ms

1	?i,CO2,1.00	0
Dec	ASCII	Null

Response breakdown

?i, CO2, 1.00

CO2 ↑ Device

1.00 ↑ Firmware

Reading device status

Command syntax

300ms ⏱ processing delay

Status	voltage at Vcc pin and reason for last restart

Example	Response
Status	⏱ Wait 300ms 1 Dec ?Status,P,5.038 ASCII 0 Null

Response breakdown

?Status,	P,	5.038
	↑ Reason for restart	↑ Voltage at Vcc

Restart codes	
P	powered off
S	software reset
B	brown out
W	watchdog
U	unknown

Sleep mode/low power

Command syntax

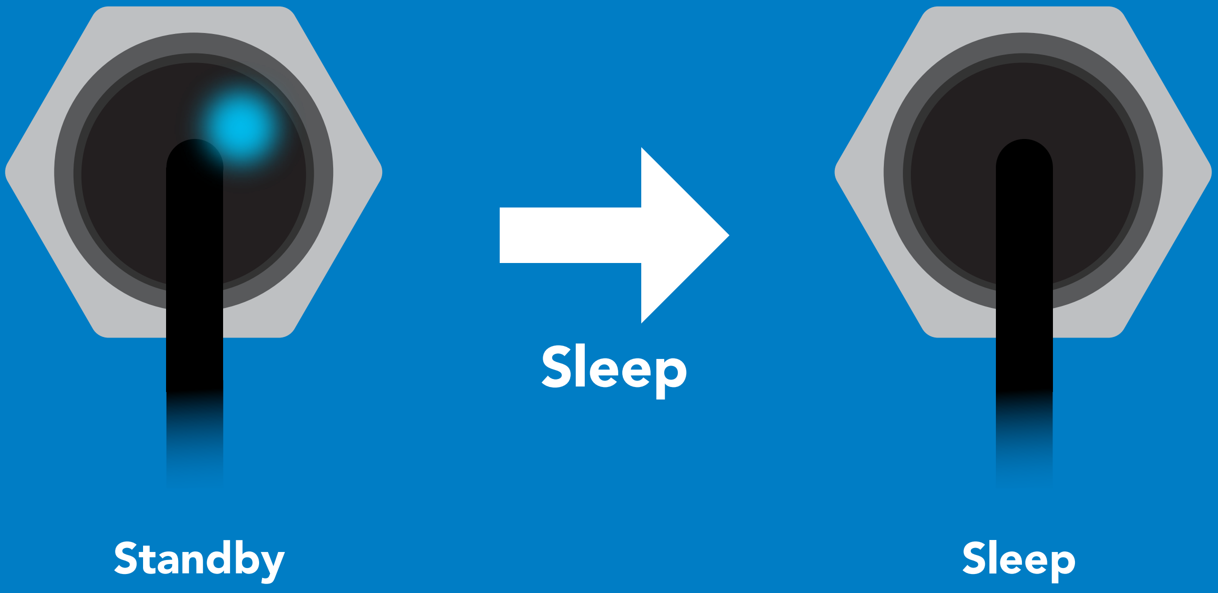
Sleep enter sleep mode/low power

Send any character or command to awaken device.

Example	Response
Sleep	no response
Any command	wakes up device

Do not read status byte after issuing sleep command.

	STANDBY	SLEEP
5V	45 mA	3.4 mA
3.3V	42 mA	3.0 mA



Protocol lock

Command syntax

300ms processing delay

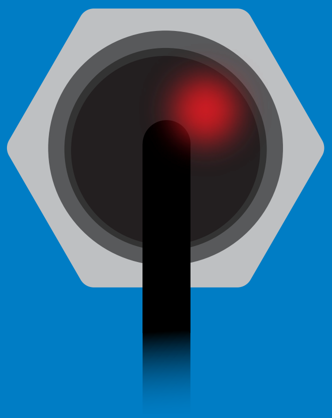
Plock,1	enable Plock	
Plock,0	disable Plock	default
Plock,?	Plock on/off?	

Locks device to I^2C mode.

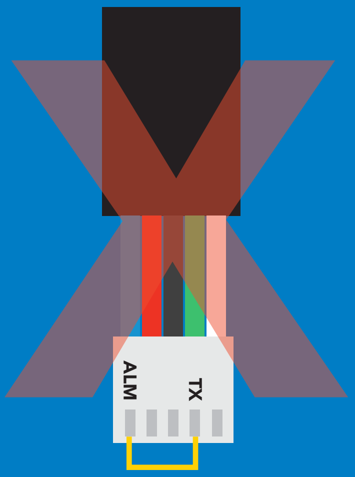
Example	Response			
Plock,1	Wait 300ms	1 Dec	0 Null	
Plock,0	Wait 300ms	1 Dec	0 Null	
Plock,?	Wait 300ms	1 Dec	?Plock,1 ASCII	0 Null

Plock,1

Baud, 9600



cannot change to UART



cannot change to UART

I^2C address change

Command syntax

300ms ⏱ processing delay

I2C,n sets I^2C address and reboots into I^2C mode

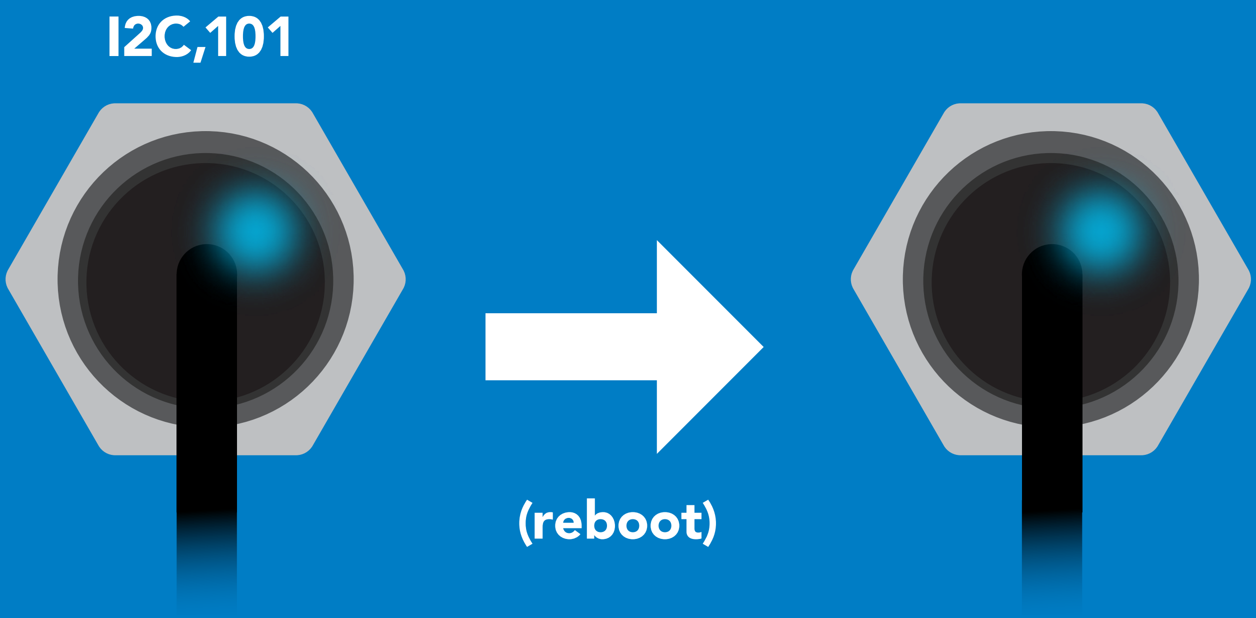
Example	Response
I2C,101	device reboot (no response given)

Warning!

Changing the I^2C address will prevent communication between the circuit and the CPU until the CPU is updated with the new I^2C address.

Default I^2C address is 105 (0x69).

n = any number 1 – 127



Factory reset

Command syntax

Factory reset will not take the device out of I²C mode.

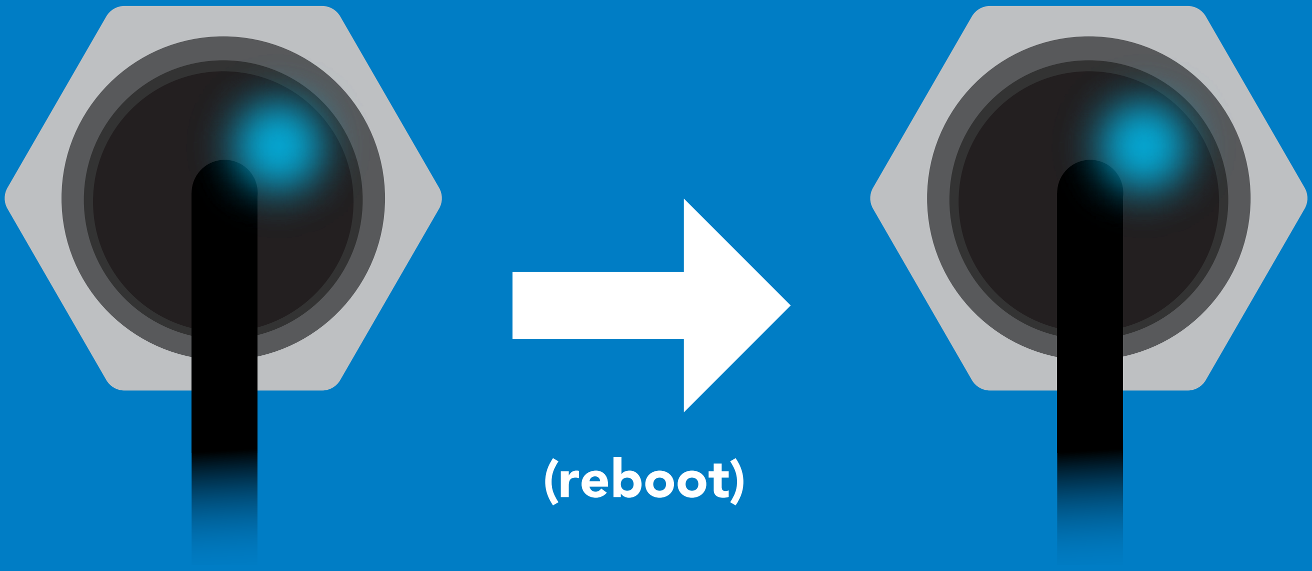
Factory	**enable factory reset**

I²C address will not change

Example	Response
Factory	device reboot (no response given)

Clears custom calibration
LED on
Response codes enabled

Factory



Change to UART mode

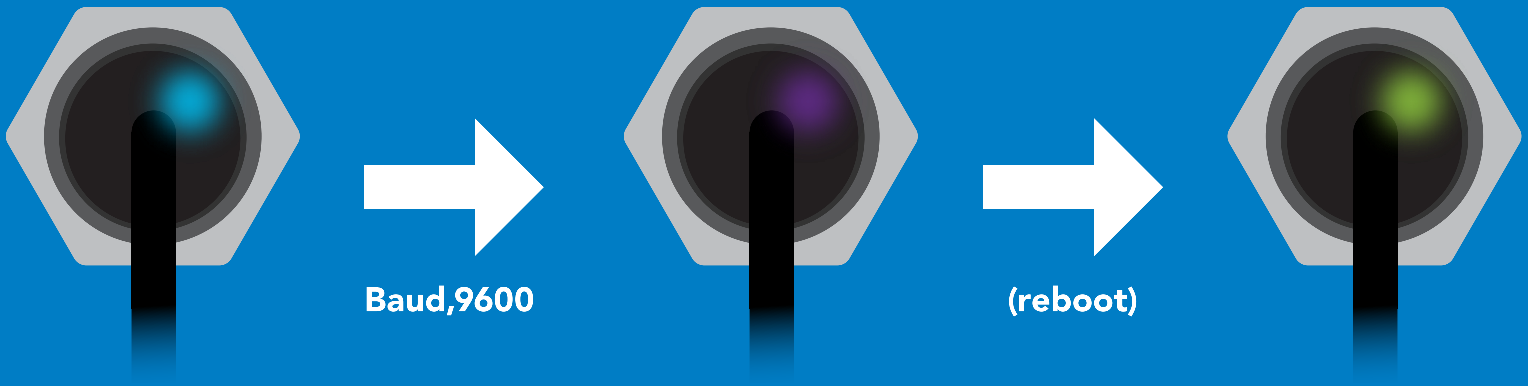
Command syntax

Baud,n	**switch from I^2C to UART**

Example	Response
Baud,9600	reboot in UART mode (no response given)

n =

- 300
- 1200
- 2400
- 9600
- 19200
- 38400
- 57600
- 115200



Baud,9600

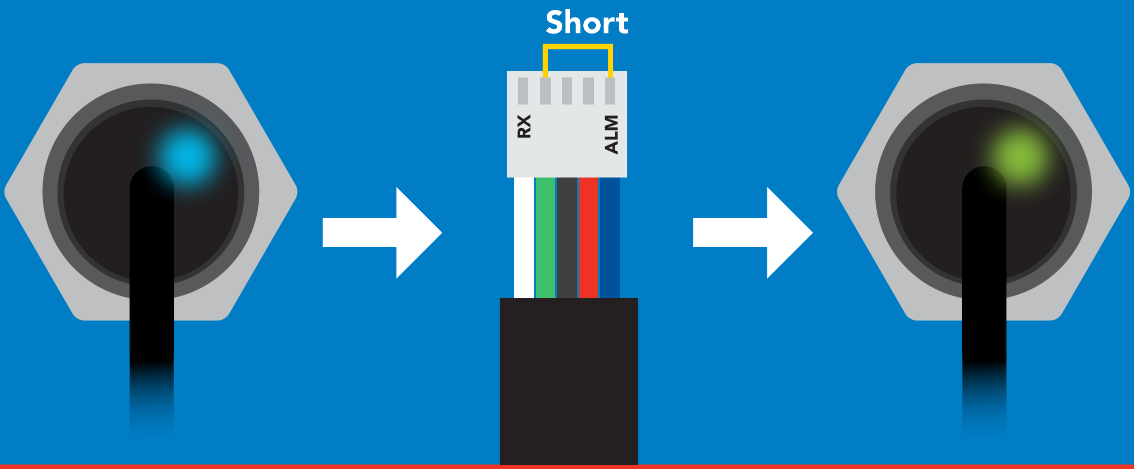
(reboot)

Changing to
UART mode

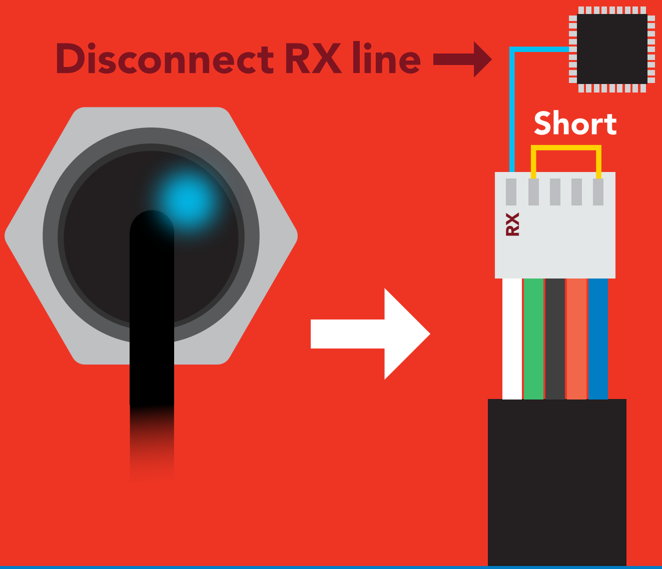
Manual switching to UART

- Disconnect ground (power off)
- Disconnect TX and RX
- Connect TX to ALM
- Confirm RX is disconnected
- Connect ground (power on)
- Wait for LED to change from Blue to Green
- Disconnect ground (power off)
- Reconnect all data and power

Example



Wrong Example


Disconnect RX line

Datasheet change log

Datasheet V 1.9

Revised info on the cover page

Datasheet V 1.8

Revised accuracy listed on cover page.

Datasheet V 1.7

Removed Import/Export commands from datasheet.

Datasheet V 1.6

Revised naming device info on pages 28 & 53.

Datasheet V 1.5

Revised info for "Pin out" on page 8.

Datasheet V 1.4

Added life expectancy to the cover page, and moved Default state to pg 11.

Datasheet V 1.3

Added page about pointing the CO2 sensor at bright lights on pg 4.

Datasheet V 1.2

Revised response for the sleep command in UART mode on pg 29.

Datasheet V 1.1

Added more information on the Export calibration and Import calibration commands.

Datasheet V 1.0

New datasheet

Firmware updates

V1.00 – (Sept 12, 2018)

- Initial release

V2.00 – (Jan 24, 2020)

- Changes the lamp power supply to 5V with boost converter, stops CO2 readings from going below 0.

V2.01 – (Nov 06, 2020)

- Adjusts lamp frequency to fit the lamp signal into the ADC range more consistently.

Warranty

Atlas Scientific™ Warranties the EZO-CO2™ Embedded NDIR CO2 Sensor to be free of defect during the debugging phase of device implementation, or 30 days after receiving the EZO-CO2™ Embedded NDIR CO2 Sensor (which ever comes first).

The debugging phase

The debugging phase as defined by Atlas Scientific™ is the time period when the EZO-CO2™ Embedded NDIR CO2 Sensor is connected into a bread board, or shield. If the EZO-CO2™ Embedded NDIR CO2 Sensor is being debugged in a bread board, the bread board must be devoid of other components. If the EZO-CO2™ Embedded NDIR CO2 Sensor is being connected to a microcontroller, the microcontroller must be running code that has been designed to drive the EZO-CO2™ Embedded NDIR CO2 Sensor exclusively and output the EZO-CO2™ Embedded NDIR CO2 Sensor data as a serial string.

It is important for the embedded systems engineer to keep in mind that the following activities will void the EZO-CO2™ Embedded NDIR CO2 Sensor warranty:

- **Soldering any part to the EZO-CO2™ Embedded NDIR CO2 Sensor.**
- **Running any code, that does not exclusively drive the EZO-CO2™ Embedded NDIR CO2 Sensor and output its data in a serial string.**
- **Embedding the EZO-CO2™ Embedded NDIR CO2 Sensor into a custom made device.**
- **Removing any potting compound.**

Reasoning behind this warranty

Because Atlas Scientific™ does not sell consumer electronics; once the device has been embedded into a custom made system, Atlas Scientific™ cannot possibly warranty the EZO-CO2™ Embedded NDIR CO2 Sensor, against the thousands of possible variables that may cause the EZO-CO2™ Embedded NDIR CO2 Sensor to no longer function properly.

Please keep this in mind:

1. **All Atlas Scientific™ devices have been designed to be embedded into a custom made system by you, the embedded systems engineer.**

2. **All Atlas Scientific™ devices have been designed to run indefinitely without failure in the field.**

3. **All Atlas Scientific™ devices can be soldered into place, however you do so at your own risk.**

Atlas Scientific™ is simply stating that once the device is being used in your application, Atlas Scientific™ can no longer take responsibility for the EZO-CO2™ Embedded NDIR CO2 Sensor continued operation. This is because that would be equivalent to Atlas Scientific™ taking responsibility over the correct operation of your entire device.