

ezo-o2TM

Embedded Oxygen Sensor

Reads	Gaseous O₂
Range	0 – 42% (2x atmospheric O₂ levels)
Calibration	Factory calibrated
Pressure	Atmosphere only
Response time	1 reading per second
Resolution	0.01
Accuracy	+/- 0.01 (0.2 PPT)
Connector	5 lead data cable
Cable length	1 meter
Data protocol	UART & I²C
Default I ² C address	108 (0x6c)
Data format	ASCII
Operating voltage	3.3V – 5V
Life expectancy	~3.5 years



Table of contents

Operating principle	4	Calibration theory	6
Physical properties	4	Custom calibration	6
Pin out	5	Default state	7
Power consumption	5	Available data protocol	8
Absolute max ratings	5		

UART

UART mode	10
Receiving data from device	11
Sending commands to device	12
LED color definition	13
UART quick command page	14
LED control	15
Find	16
Continuous mode	17
Single reading mode	18
Alarm	19
Calibration	20
Temperature compensation	21
Enable/disable parameters	22
Naming device	23
Device information	24
Response codes	25
Reading device status	26
Sleep mode/low power	27
Change baud rate	28
Protocol lock	29
Factory reset	30
Change to I2C mode	31
Manual switching to I2C	32

I²C

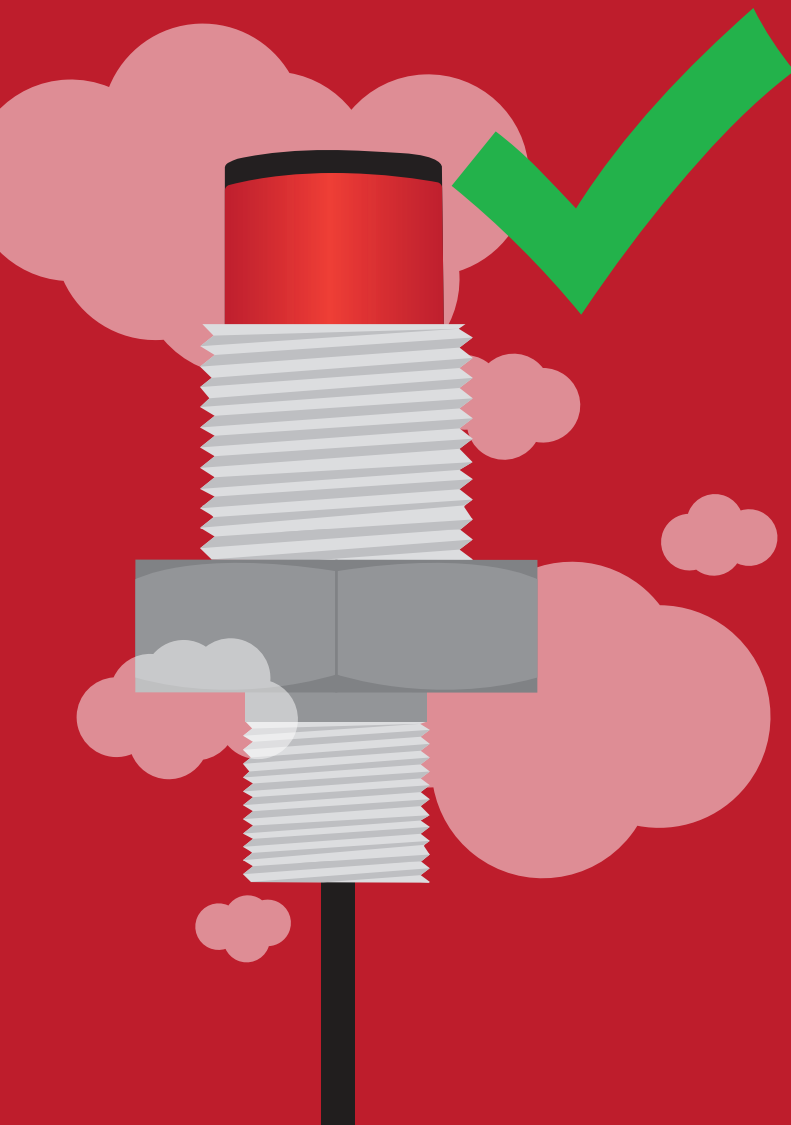
I ² C mode	34
Sending commands	35
Requesting data	36
Response codes	37
Processing delay	37
LED color definition	38
I²C quick command page	39
LED control	40
Find	41
Taking reading	42
Alarm	43
Calibration	44
Temperature compensation	45
Enable/disable parameters	46
Naming device	47
Device information	48
Reading device status	49
Sleep mode/low power	50
Protocol lock	51
I ² C address change	52
Factory reset	53
Change to UART mode	54
Manual switching to UART	55

Datasheet change log	56
Firmware updates	56
Warranty	57

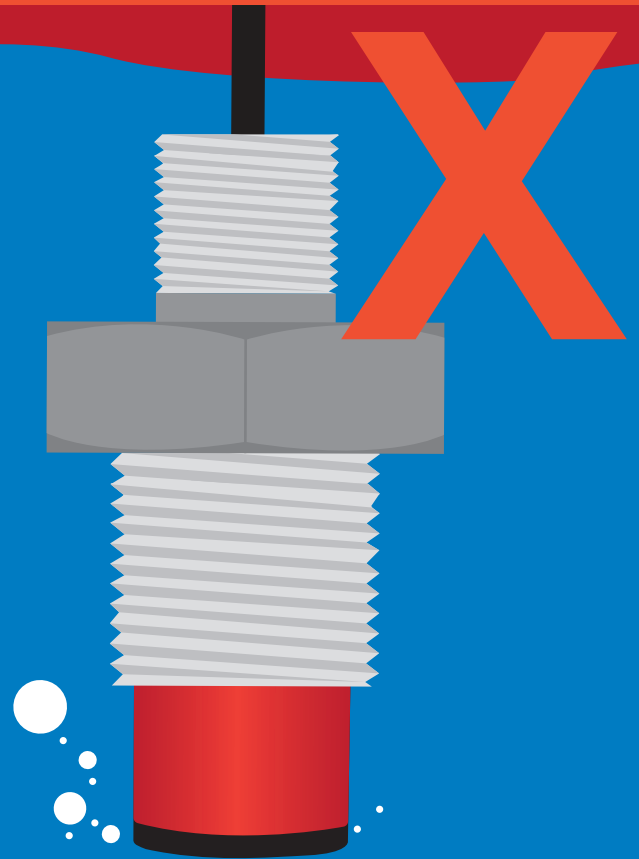
Attention

The EZO-O2™ is 100% operational out of the box.
CALIBRATION IS UNNECESSARY

This sensor detects
GASEOUS O₂



This sensor does not
read dissolved O₂

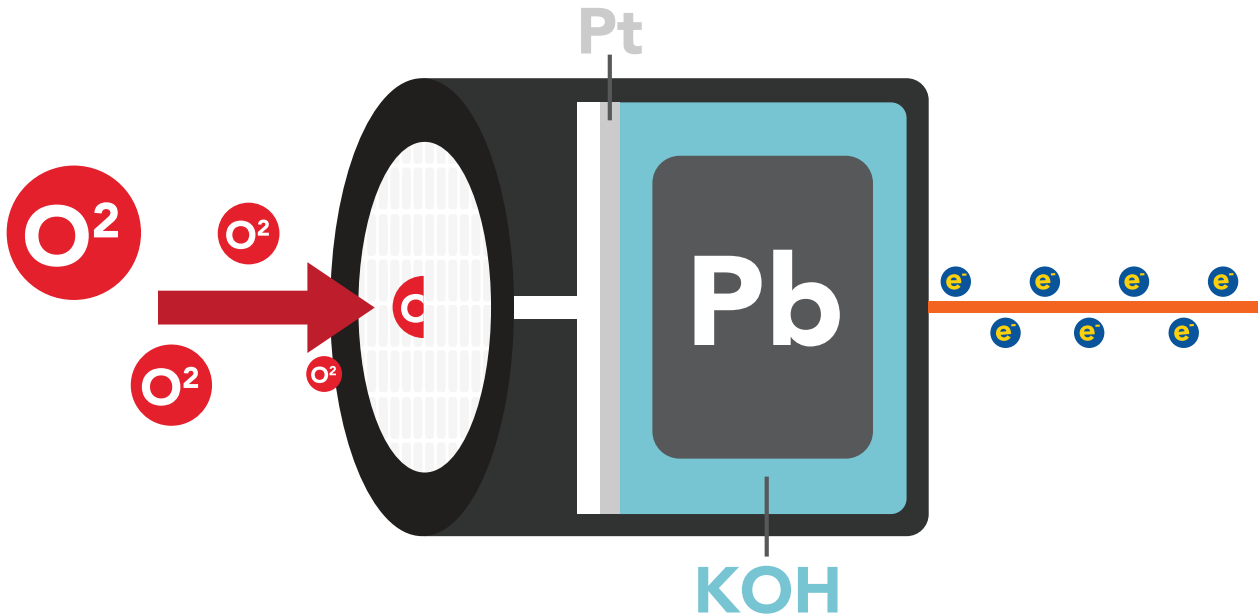


DO NOT SUBMERGE!

**Click here for our line of
Dissolved Oxygen sensors.**

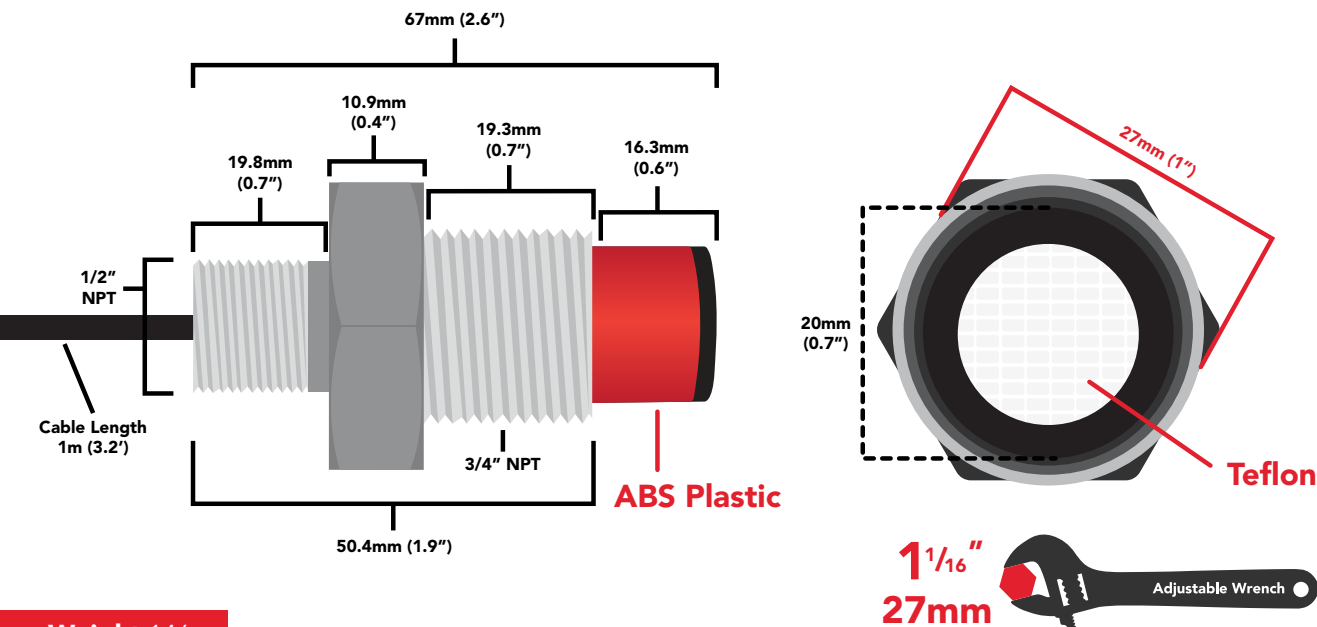
Operating principle

The Atlas Scientific EZO-O2™ Embedded Oxygen Sensor is an electrochemical sensing device that detects the partial pressure of oxygen through reduction. The sensor can be thought of as a small fuel cell. When the oxygen comes in contact with the sensor, the "fuel cell" begins to produce a current. A teflon membrane ensures that the oxygen enters the sensor at a steady rate.



Physical properties

The EZO-O2™ sensor only detects gaseous oxygen levels. This device cannot read dissolved O2 levels. **DO NOT SUBMERGE IN LIQUID.**



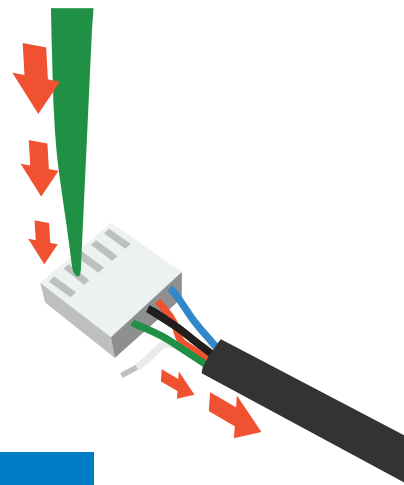
Weight 146g

Body 316 Stainless Steel

Pin out

Data and power cable pinout

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



Should you need to remove this connector from the data cable, follow the provided illustration.

The alarm pin will go high when a set O₂ level has been crossed.



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

See page **19** to enable O₂ level alarm in UART mode.

See page **43** to enable O₂ level alarm in I2C mode.

Power consumption

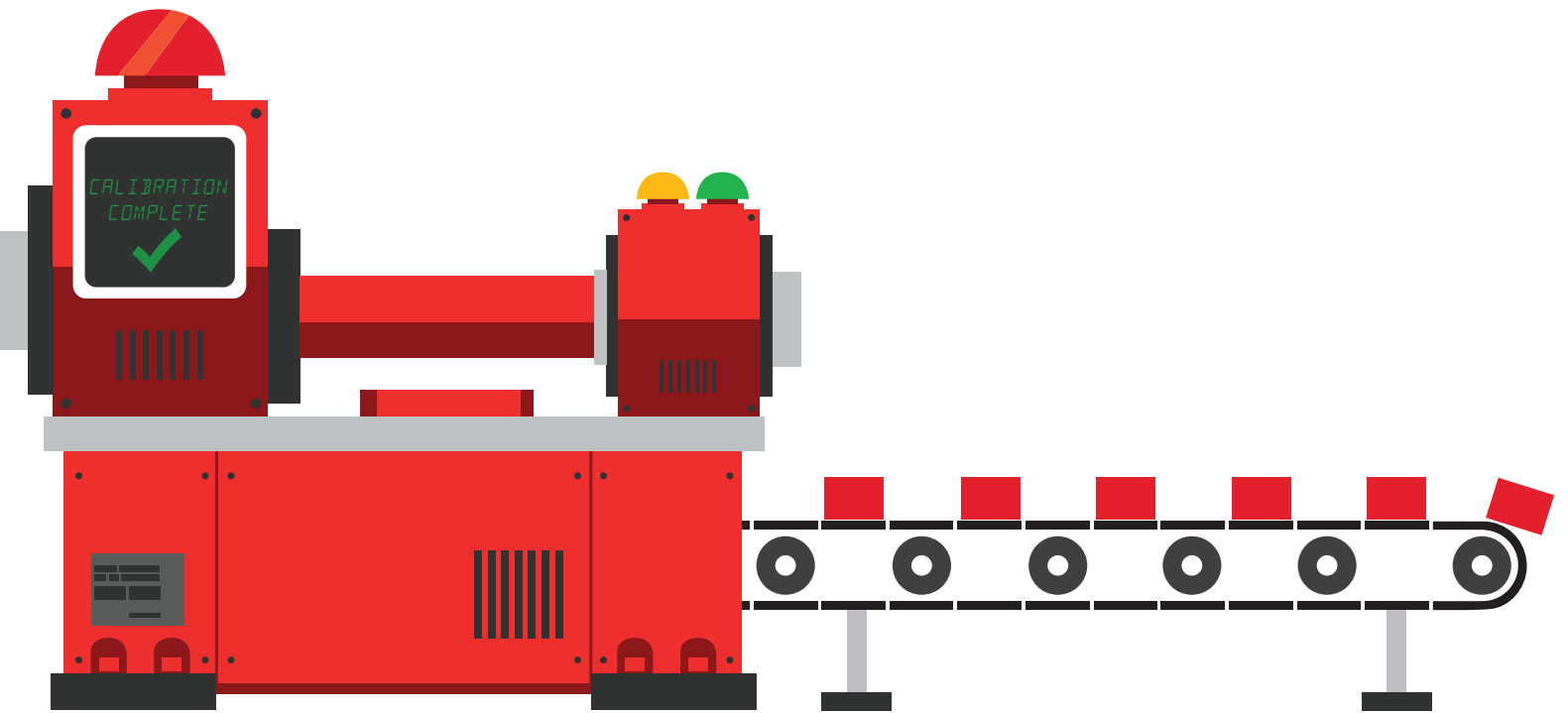
	LED	MAX	SLEEP
5V	ON	14.6 mA	0.5 mA
	OFF	13.9 mA	
3.3V	ON	13.7 mA	0.4 mA
	OFF	13.5 mA	

Absolute max ratings

Parameter	MIN	TYP	MAX
Storage temperature	-30 °C		75 °C
Operational temperature	-20 °C	25 °C	50 °C
VCC	3.3V	3.3V	5.5V

Calibration theory

The Atlas Scientific EZO-O2™ Embedded Oxygen Sensor comes pre-calibrated. As part of the manufacturing process Atlas Scientific performs a two-point factory calibration.



Low point calibration = 0% O₂
High point calibration = 20.95%

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

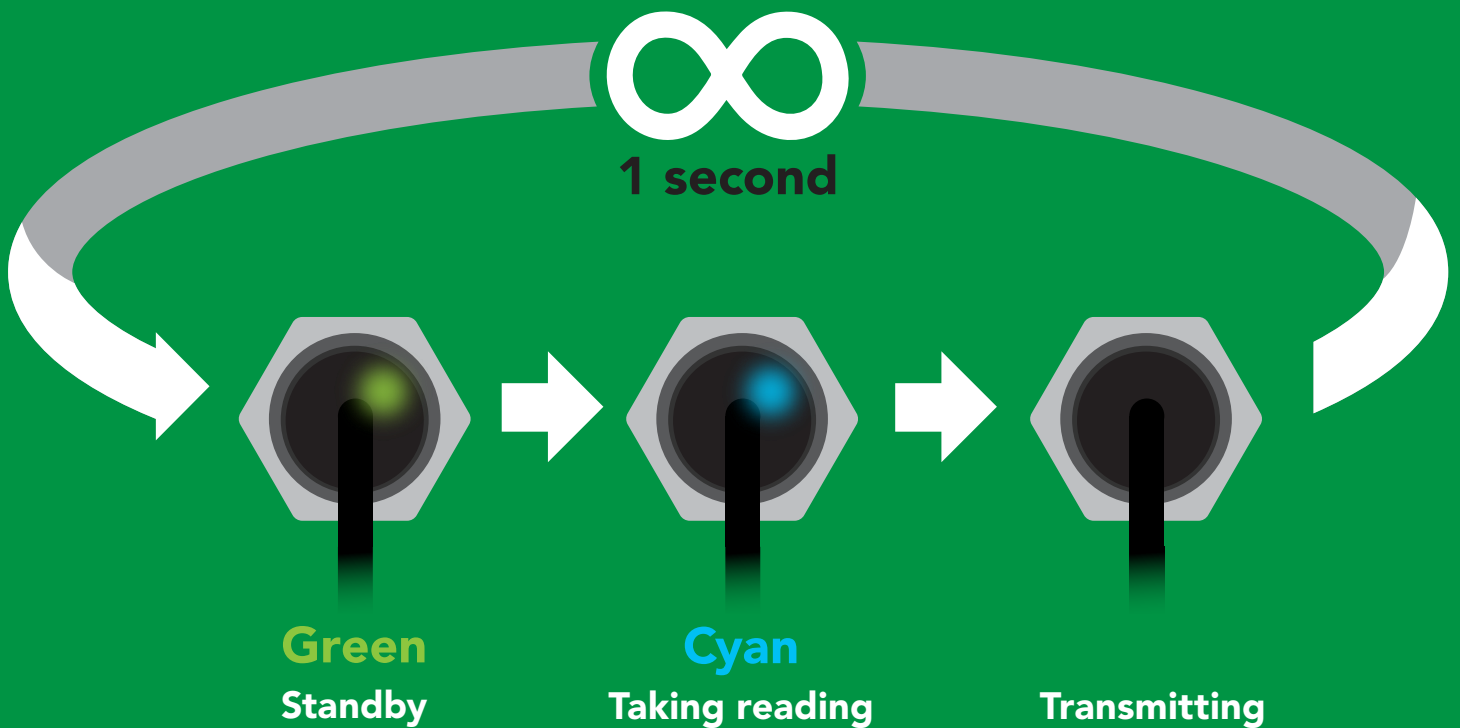
Custom calibration

After ~12 months of operation the EZO-O2™ Embedded Oxygen Sensor may need to be re-calibrated. A simple single point recalibration to the atmospheric O₂ level is all that's needed.

Default state

UART mode

Baud	9,600
Readings	continuous
Speed	1 second
LED	on



✓ Available data protocols

UART

default

I²C

✗ 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

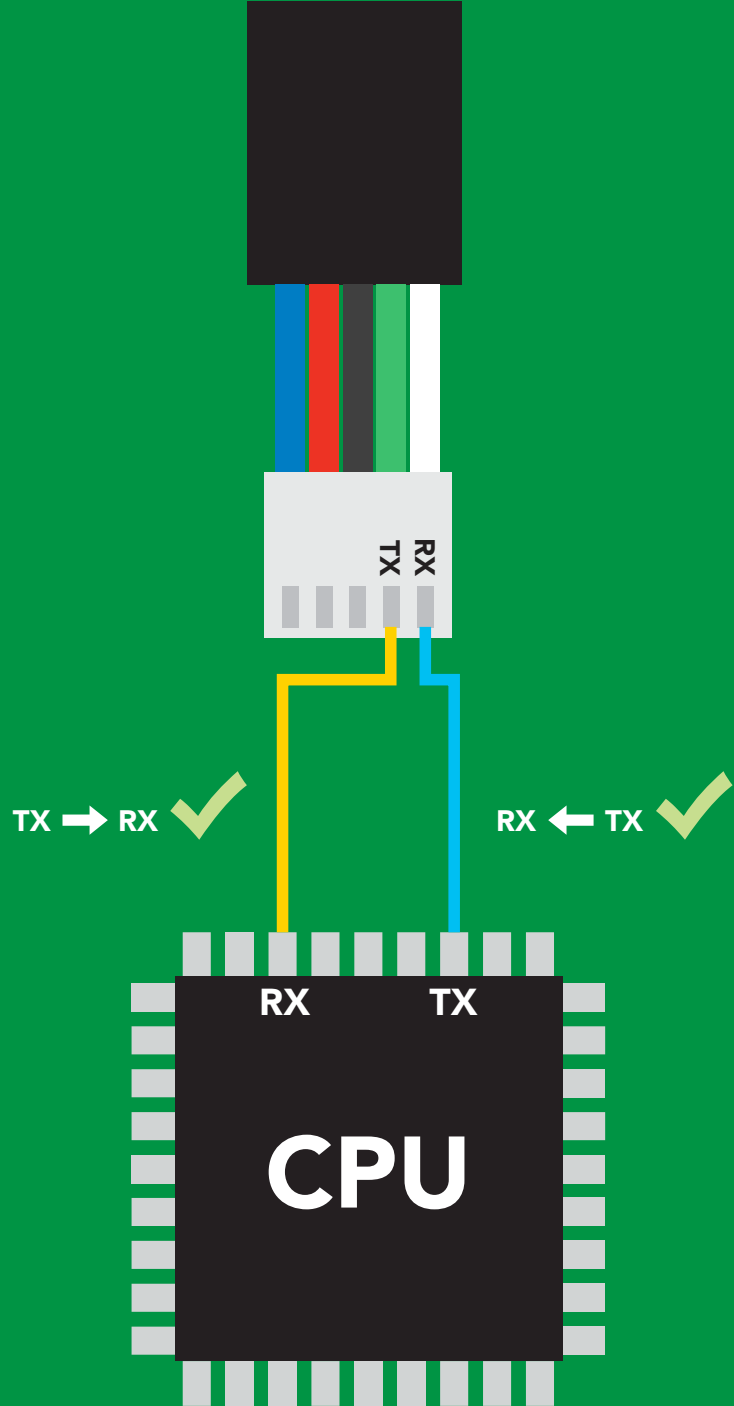
UART mode

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



Vcc 3.3V – 5V



Data format

Reading	Gaseous O₂	Data type	Floating point
Units	percent concentration & PPT (when enabled)	Decimal places	2
Encoding	ASCII	Smallest string	4 characters
Format	string (CSV string when PPT is enabled)	Largest string	16 characters
Terminator	carriage return		

Receiving data from device

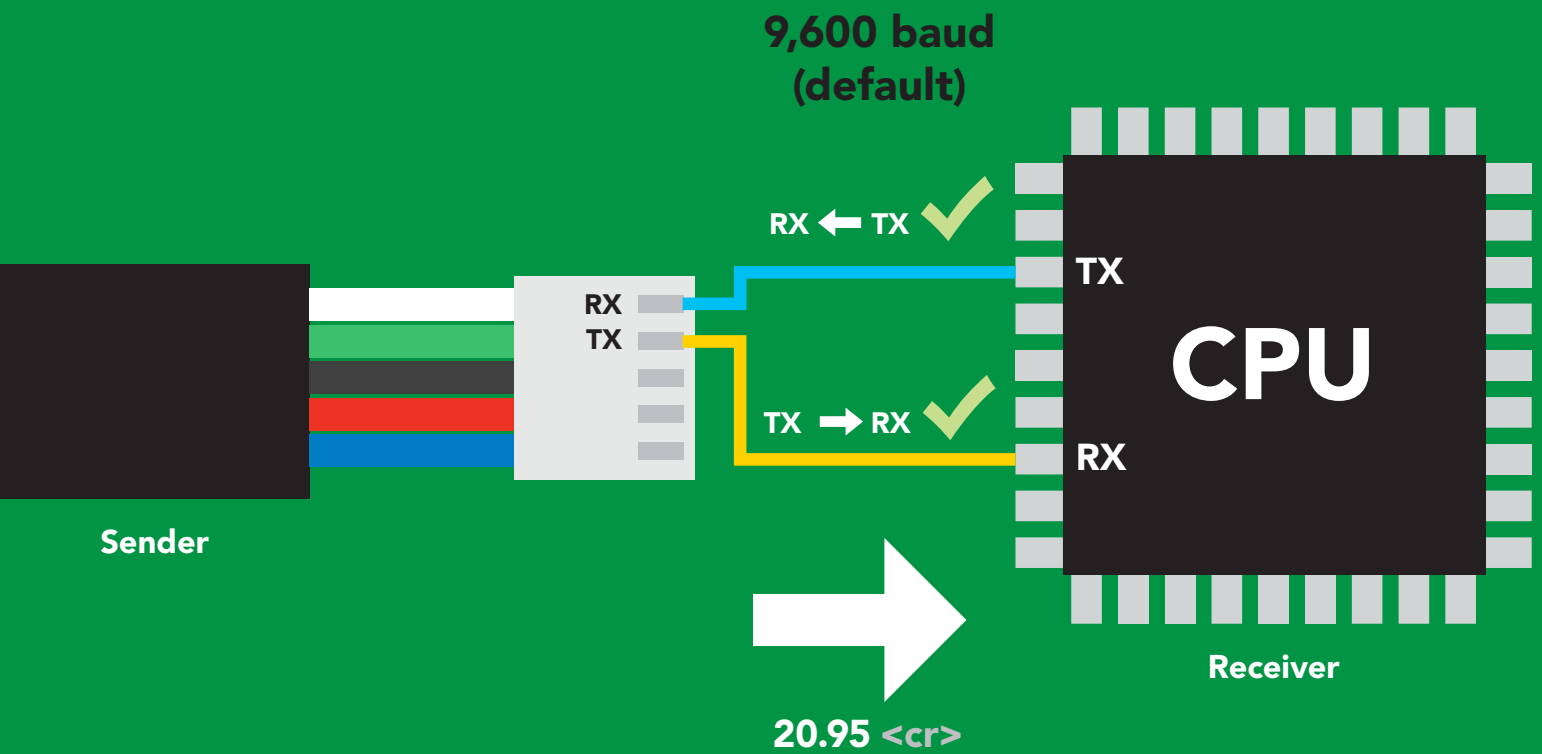
2 parts

ASCII data string

Command

Carriage return <cr>

Terminator



Advanced

ASCII: 2 0 . 9 5 <cr>

Hex: 32 30 2E 39 35 0D

Dec: 50 48 46 57 53 13

Sending commands to device

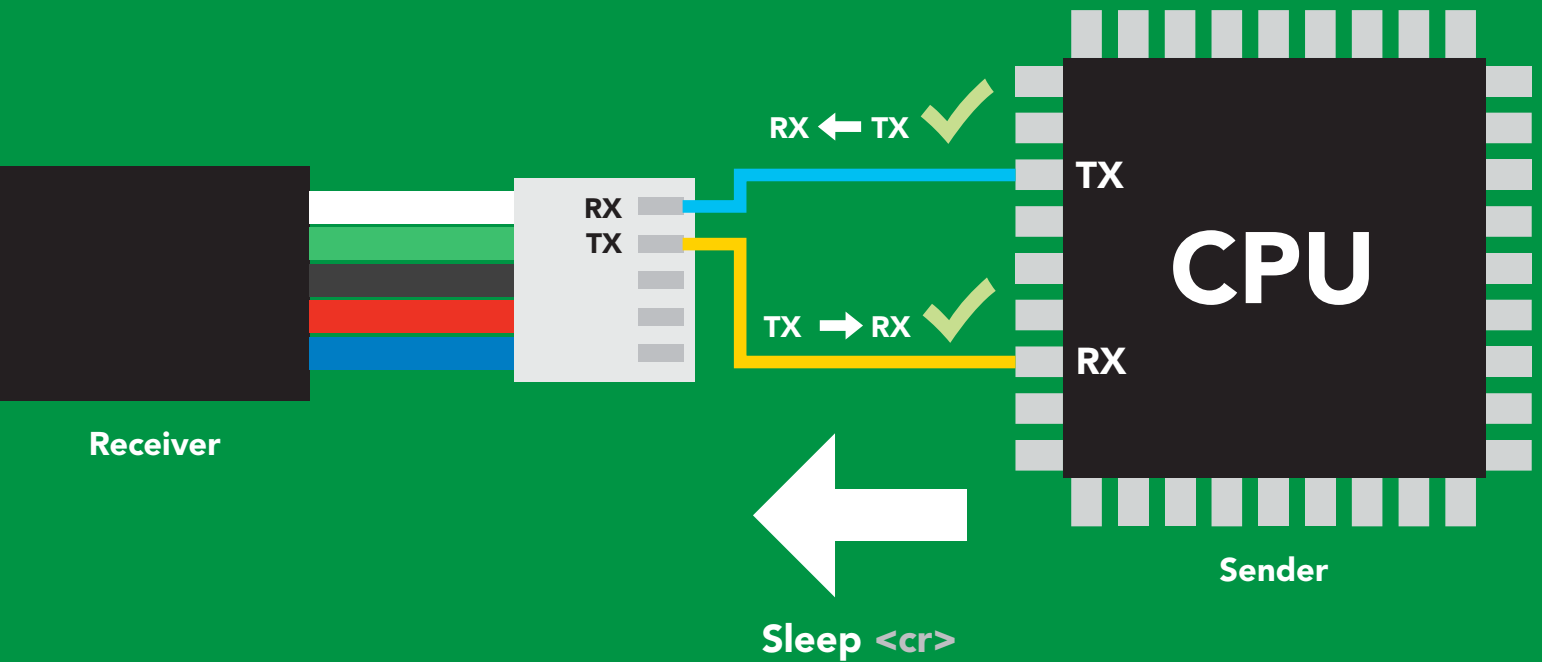
2 parts

Command (not case sensitive)

ASCII data string

Carriage return <cr>

Terminator



Advanced

ASCII: **S** **I** **e** **e** **p** **<cr>**

Hex: **53** **6C** **65** **65** **70** **0D**

Dec: **83** **108** **101** **101** **112** **13**

LED color definition



Green

UART standby



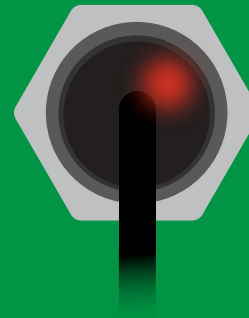
Cyan

Taking reading



Purple

Changing
baud rate



Red

Command
not understood



White

Find

5V

LED ON
+0.7 mA

3.3V

+0.2 mA

UART mode

command quick reference

All commands are ASCII strings or single ASCII characters.

Command	Function		Default state
Alarm	enable/disable alarm	pg. 19	n/a
Baud	change baud rate	pg. 28	9,600
C	enable/disable continuous mode	pg. 17	enabled
Cal	performs calibration	pg. 20	n/a
Factory	enable factory reset	pg. 30	n/a
Find	finds device with blinking white LED	pg. 16	n/a
i	device information	pg. 24	n/a
I2C	change to I ² C mode	pg. 31	not set
L	enable/disable LED	pg. 15	enabled
Name	set/show name of device	pg. 23	not set
O	enable/disable parameters	pg. 22	O2
Plock	enable/disable protocol lock	pg. 29	n/a
R	returns a single reading	pg. 18	n/a
Sleep	enter sleep mode/low power	pg. 27	n/a
Status	retrieve Status Information	pg. 26	n/a
T	Temperature compensation	pg. 21	n/a
*OK	enable/disable response codes	pg. 25	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>



L,1



L,0

Find

Command syntax

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

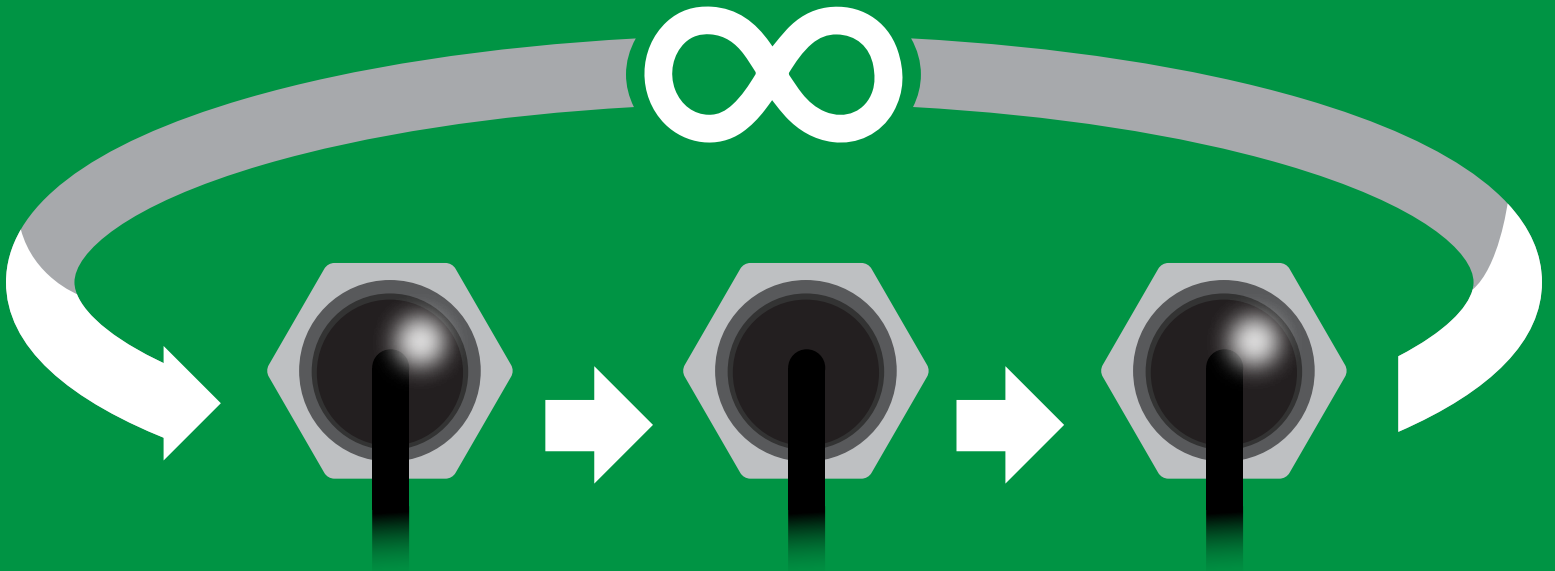
Find <cr> LED rapidly blinks white, used to help find device

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>**
O2 (1 sec) <cr>
O2 (2 sec) <cr>
O2 (n sec) <cr>

C,30 <cr>

***OK <cr>**
O2 (30 sec) <cr>
O2 (60 sec) <cr>
O2 (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

R <cr>

Response

20.95 <cr>

*OK <cr>



Green
Standby



Cyan
Taking reading



Transmitting



1 second

Alarm

Command syntax

The alarm pin will = 1 when O2 levels are > alarm set point. Alarm tolerance sets how far below the set point O2 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 – 60)

Alarm,? <cr> alarm set?

Example

Response

Alarm,en,1 <cr>

*OK <cr> Enable alarm

Alarm,5.5 <cr>

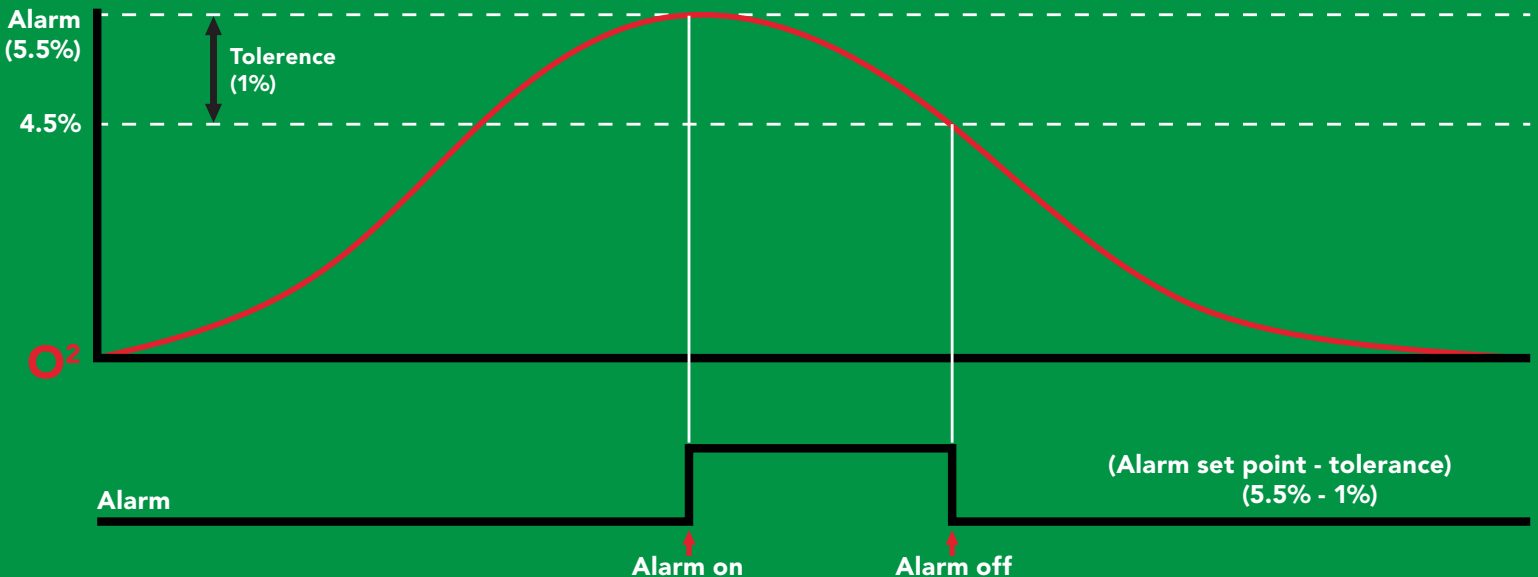
*OK <cr>

Alarm,tol,1 <cr>

*OK <cr> O2 level must fall one percentage point below set point for alarm to reset.

Alarm,? <cr>

?,alarm,5.50,1.00,1 <cr> if all are enabled



Calibration

Command syntax

After ~1 year the sensor may need re-calibration. A single point calibration to atmospheric O2 levels is all that's needed. 0 point calibration can also be done if accuracy at low O2 levels is needed.

- Cal,nn.nn <cr>** calibration to O2 levels at your altitude. nn.nn =%o2
- Cal,0 <cr>** calibrate device to 0 oxygen
- Cal,clear <cr>** delete calibration data
- Cal,? <cr>** device calibrated?

Example

Response

Cal,20.95 <cr>

***OK <cr>** Calibrated to O2 concentration at sea level

Cal,0 <cr>

***OK <cr>**

Cal,clear <cr>

***OK <cr>**

Cal,? <cr>

?Cal,0 <cr> or **?Cal,1 <cr>** or **?Cal,2 <cr>**
*OK <cr> single point two point

Altitude (feet)	Altitude (meters)	%
1,000	305	20.1
5,000	1,524	17.3
10,000	3,048	14.3

Temperature compensation

Command syntax

Air temperature affects how the sensor works, not the actual O2 concentration in the air.

T,n <cr> n = any value; floating point or int

T,? <cr> compensated temperature value?

RT,n <cr> set temperature compensation and take a reading

Example

Response

T,19.5 <cr>

***OK** <cr>

RT,19.5 <cr>

***OK** <cr>

20.95 <cr> Temperature compensated O2 reading

T,? <cr>

?T,19.5 <cr>

***OK** <cr>

Enable/disable parameters from output string

Command syntax

O, [parameter],[1,0] <cr> enable or disable output parameter

O,? <cr> enabled parameter?

Example

O,PPT,1 / O,PPT,0 <cr>

O,%,1 / O,%,0 <cr>

O,? <cr>

Response

*OK <cr> enable / disable PPT

*OK <cr> enable / disable percent concentration

?,O,%,PPT <cr> if both are enabled

Parameters

PPT O₂ in parts per thousand
% O₂ in percent concentration

Followed by 1 or 0

1 enabled
0 disabled

*** If you disable all possible data types your readings will display "no output".**

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 <cr>



*OK <cr>

Name,? <cr>



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

Device information

Command syntax

```
i <cr> device information
```

Example

```
i <cr>
```

Response

```
?i,O2,1.0 <cr>  
*OK <cr>
```

Response breakdown

?i,	O2,	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>

20.95 <cr>
***OK** <cr>

***OK,0** <cr>

no response, ***OK** disabled

R <cr>

20.95 <cr> ***OK** disabled

***OK,?** <cr>

?*OK,1 <cr> or **?*OK,0** <cr>

Other response codes

- *ER** unknown command
- *OV** over volt ($VCC \geq 5.5V$)
- *UV** under volt ($VCC \leq 3.1V$)
- *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

```
Status <cr>
```

Response

```
?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

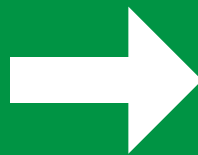
*WA <cr> wakes up device

5V

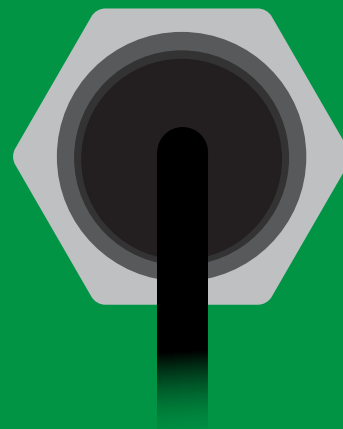
MAX	SLEEP
14.6 mA	0.5 mA

3.3V

13.7 mA	0.4 mA
---------	--------



Sleep <cr>



Change baud rate

Command syntax

Baud,n <cr> change baud rate

Example

Baud,38400 <cr>

Response

*OK <cr>

Baud,? <cr>

?Baud,38400 <cr>

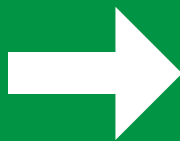
*OK <cr>

n =

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



Standby

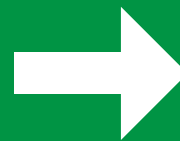


Baud,38400 <cr>



Changing
baud rate

*OK <cr>



(reboot)



Standby

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>`

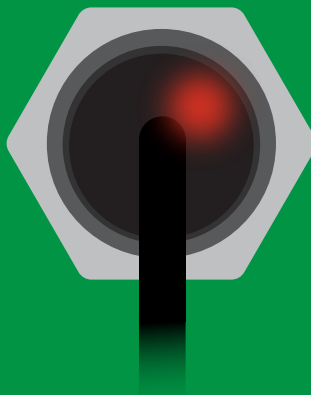
`?Plock,1 <cr>` or `?Plock,0 <cr>`

`Plock,1`



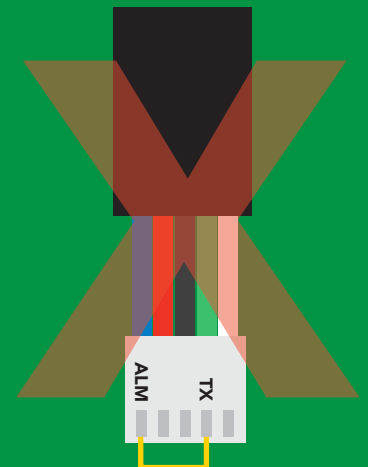
`*OK <cr>`

`I2C,100`



cannot change to I²C

`*ER <cr>`



cannot change to I²C

Factory reset

Command syntax

Clears custom calibration
"*OK" enabled

Factory <cr> enable factory reset

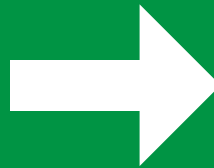
Example

Response

Factory <cr>

*OK <cr>

Factory <cr>



(reboot)



*OK <cr>

*RS <cr>

*RE <cr>

Baud rate will not change

Change to I²C mode

Command syntax

Default I²C address 108 (0x6C)

I2C,n <cr> sets I²C address and reboots into I²C mode

n = any number 1 – 127

Example

Response

I2C,100 <cr>

*OK (reboot in I²C mode)

Wrong example

Response

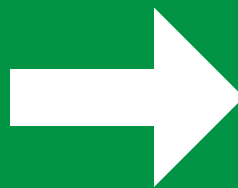
I2C,139 <cr> n ≠ 127

*ER <cr>

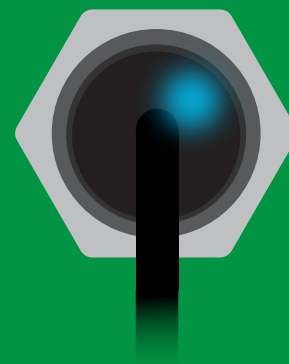
I2C,100



Green
*OK <cr>



(reboot)



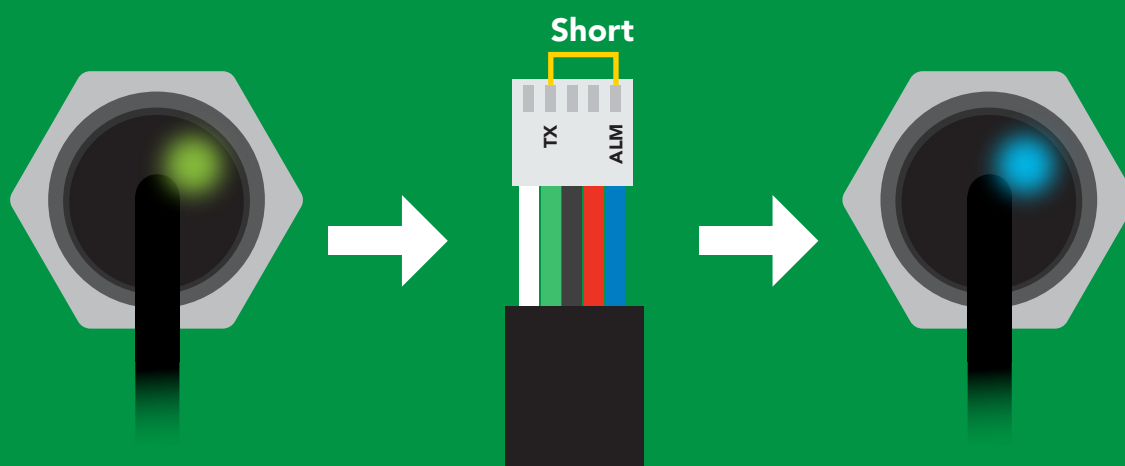
Blue
now in I²C mode

Manual switching to I²C

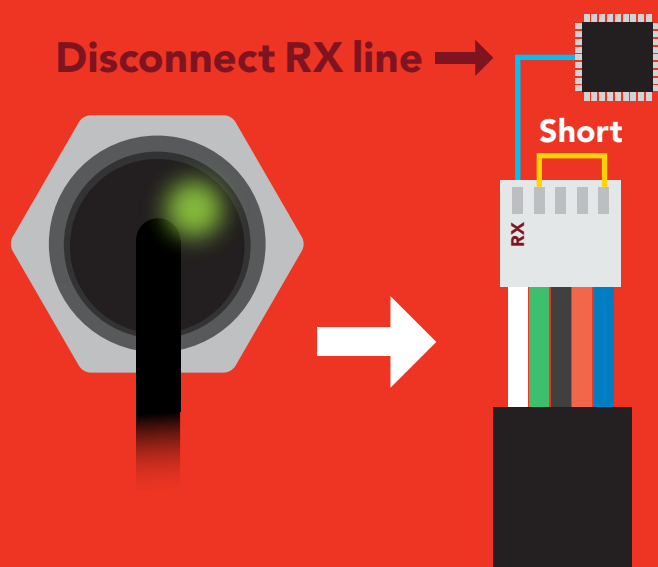
- Disconnect ground (power off)
- Disconnect TX and RX
- Connect TX to ALM
- Confirm RX is disconnected
- Connect ground (power on)
- Wait for LED to change from **Green** to **Blue**
- Disconnect ground (power off)
- Reconnect all data and power

Manually switching to I²C will set the I²C address to 108 (0x6C)

Example



Wrong Example



I²C mode

The I²C 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 I²C mode click [here](#)

Settings that are retained if power is cut

- Calibration
- Change I²C 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

I²C mode

I²C address (0x01 – 0x7F)
108 (0x6C) default

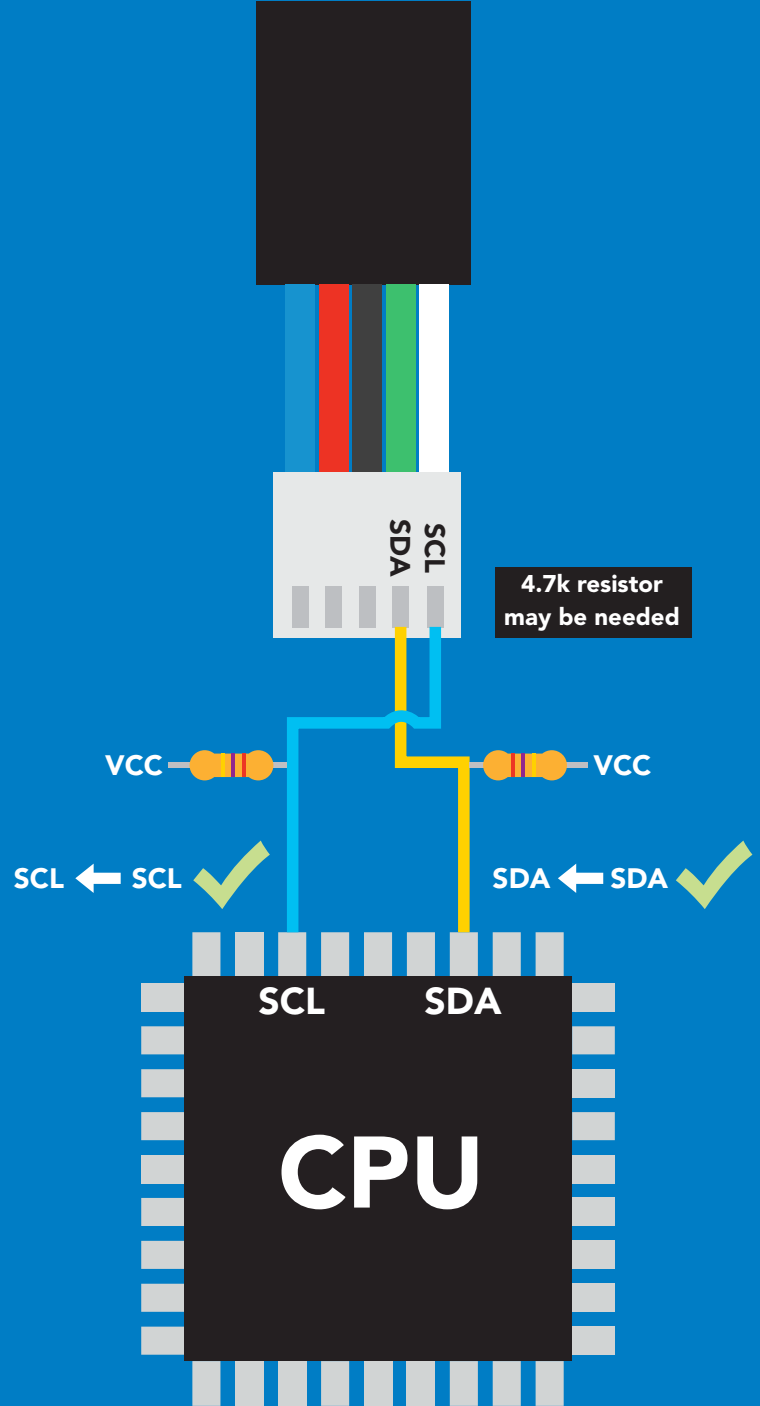
Vcc 3.3V – 5.5V

Clock speed 100 – 400 kHz

SDA 

SCL 





Data format

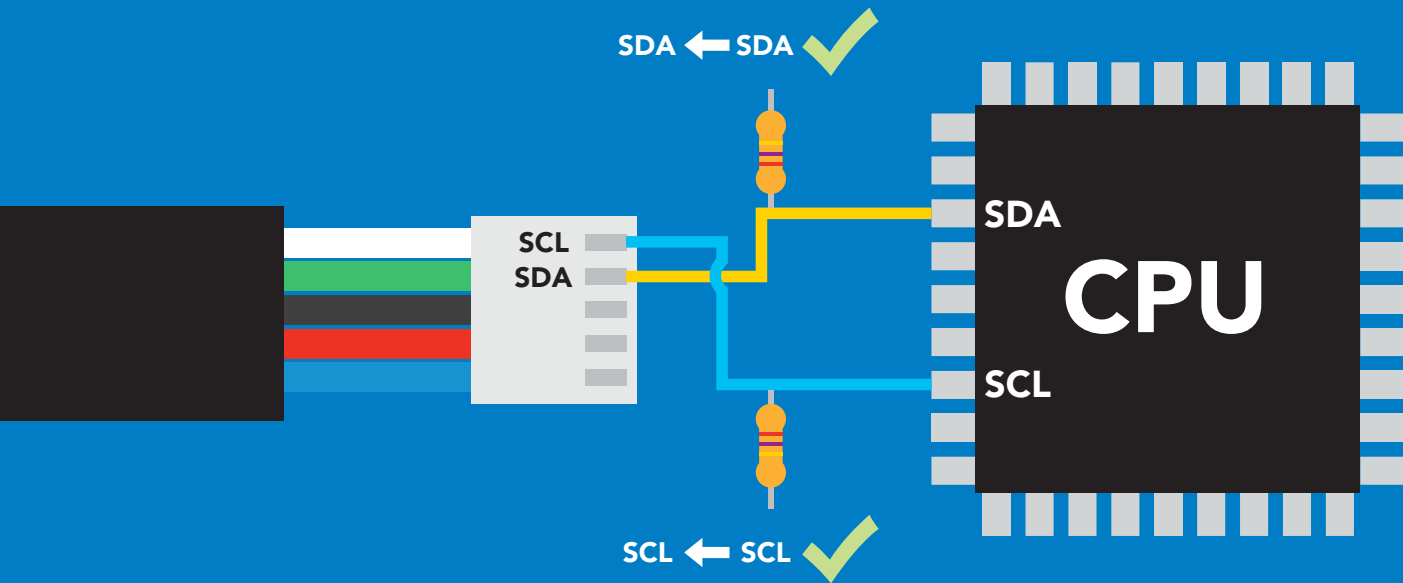
Reading **Gaseous O₂**
Units percent concentration & PPT (when enabled)
Encoding **ASCII**
Format **string**
(CSV string when PPT is enabled)

Data type **Floating point**
Decimal places **2**
Smallest string **4 characters**
Largest string **16 characters**

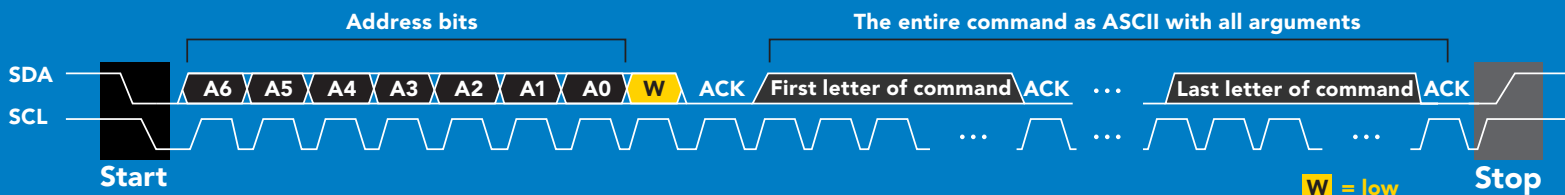
Sending commands to device



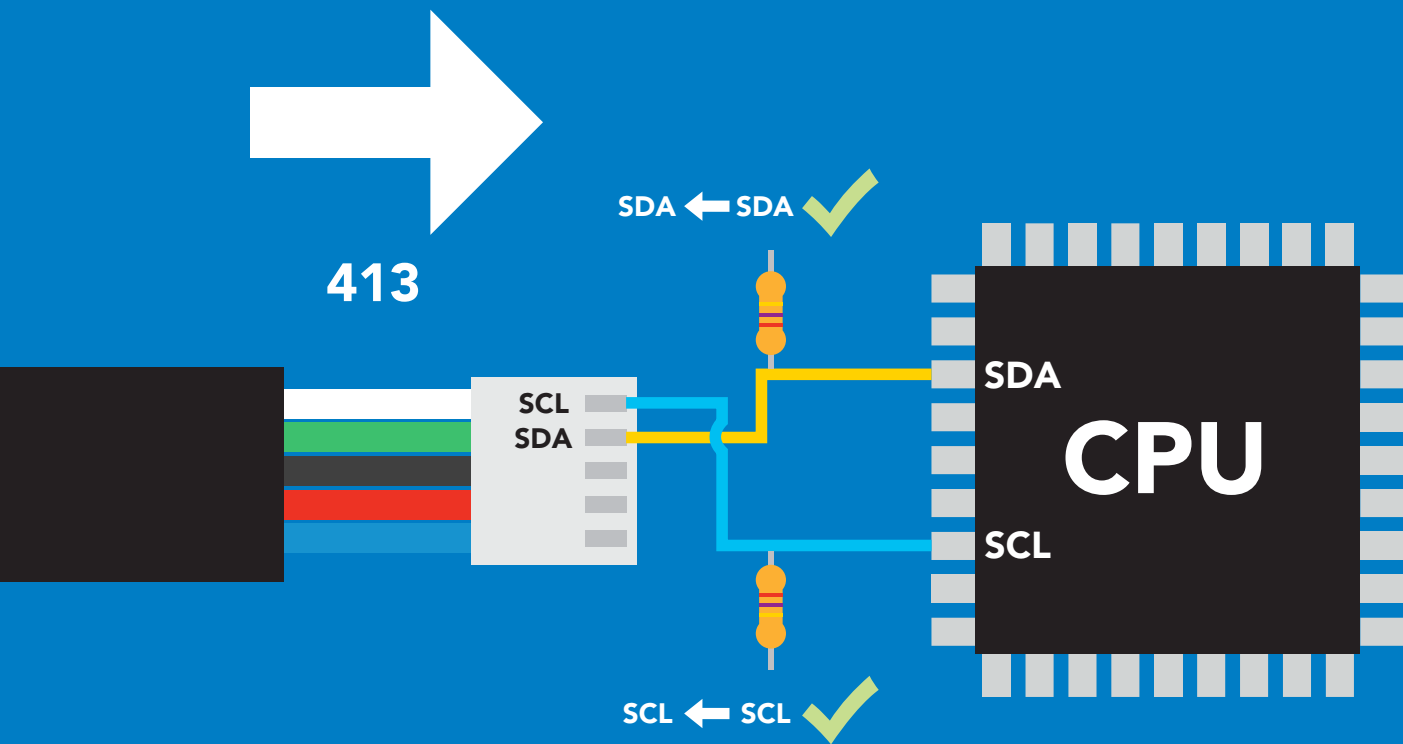
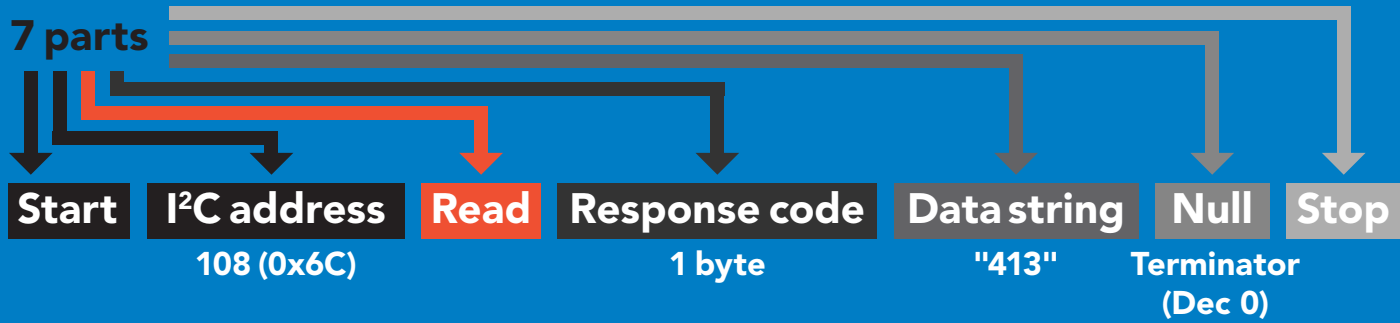
Example



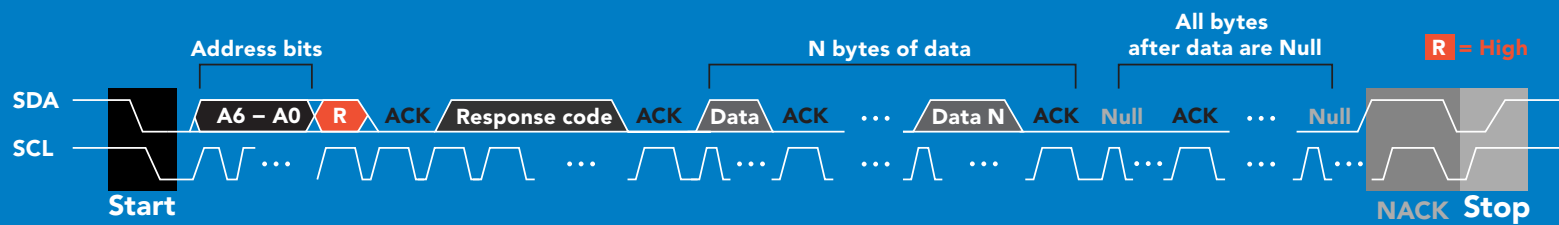
Advanced



Requesting data from device



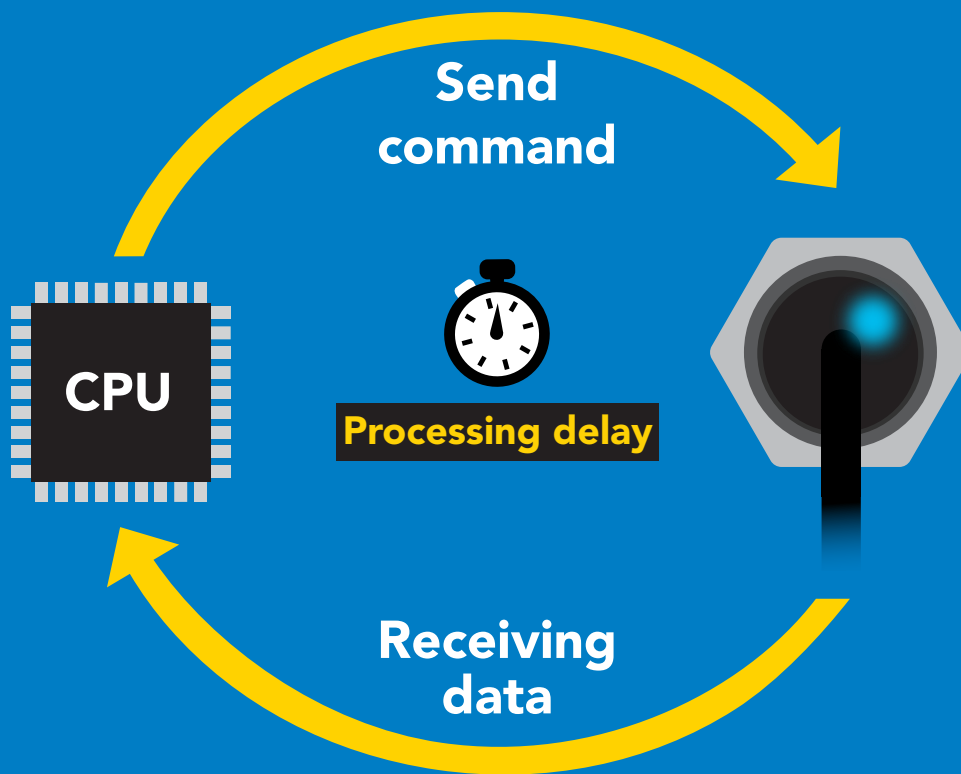
Advanced



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.



Example

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

```
delay(300);
```



```
Processing delay
```

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

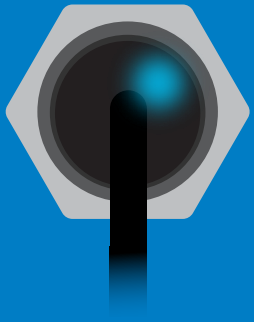
If there is no processing delay or the processing delay is too short, the response code will always be 254.

Response codes

Single byte, not string

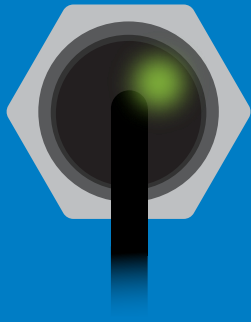
255	no data to send
254	still processing, not ready
2	syntax error
1	successful request

LED color definition



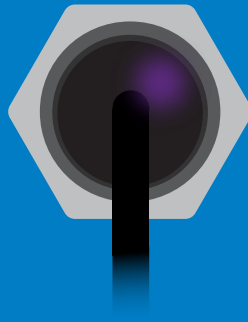
Blue

I²C standby



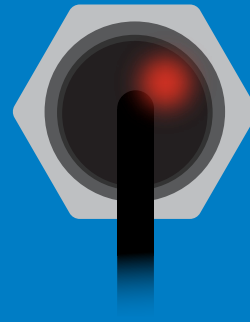
Green

Taking reading



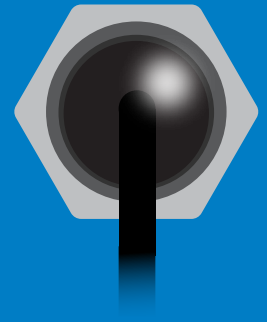
Purple

Changing
I²C address



Red

Command
not understood



White

Find

5V

LED ON
+0.7 mA

3.3V

+0.2 mA

I²C mode

command quick reference

All commands are ASCII strings or single ASCII characters.

Command	Function	
Alarm	enable/disable alarm	pg. 43
Baud	switch back to UART mode	pg. 54
Cal	performs calibration	pg. 44
Factory	enable factory reset	pg. 53
Find	finds device with blinking white LED	pg. 41
i	device information	pg. 47
I2C	change I ² C address	pg. 52
L	enable/disable LED	pg. 40
Name	set/show name of device	pg. 47
O	enable/disable parameters	pg. 46
Plock	enable/disable protocol lock	pg. 51
R	returns a single reading	pg. 42
Sleep	enter sleep mode/low power	pg. 50
Status	retrieve status information	pg. 49
T	enter sleep mode/low power	pg. 45

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

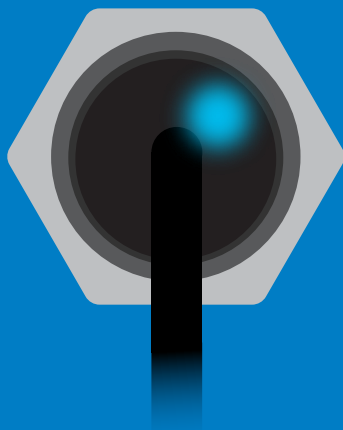
 **Wait 300ms** **1** **0**
Dec Null

L,0

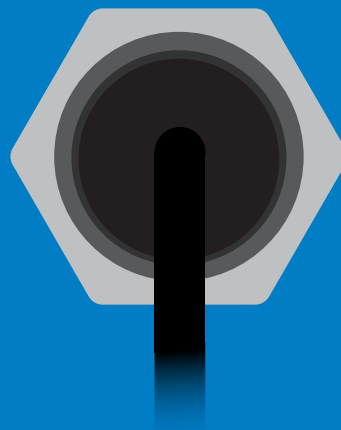
 **Wait 300ms** **1** **0**
Dec Null

L,?

 **Wait 300ms** **1** **?L,1** **0** or  **Wait 300ms** **1** **?L,0** **0**
Dec ASCII Null Dec ASCII Null



L,1



L,0

Find

300ms  processing delay

Command syntax

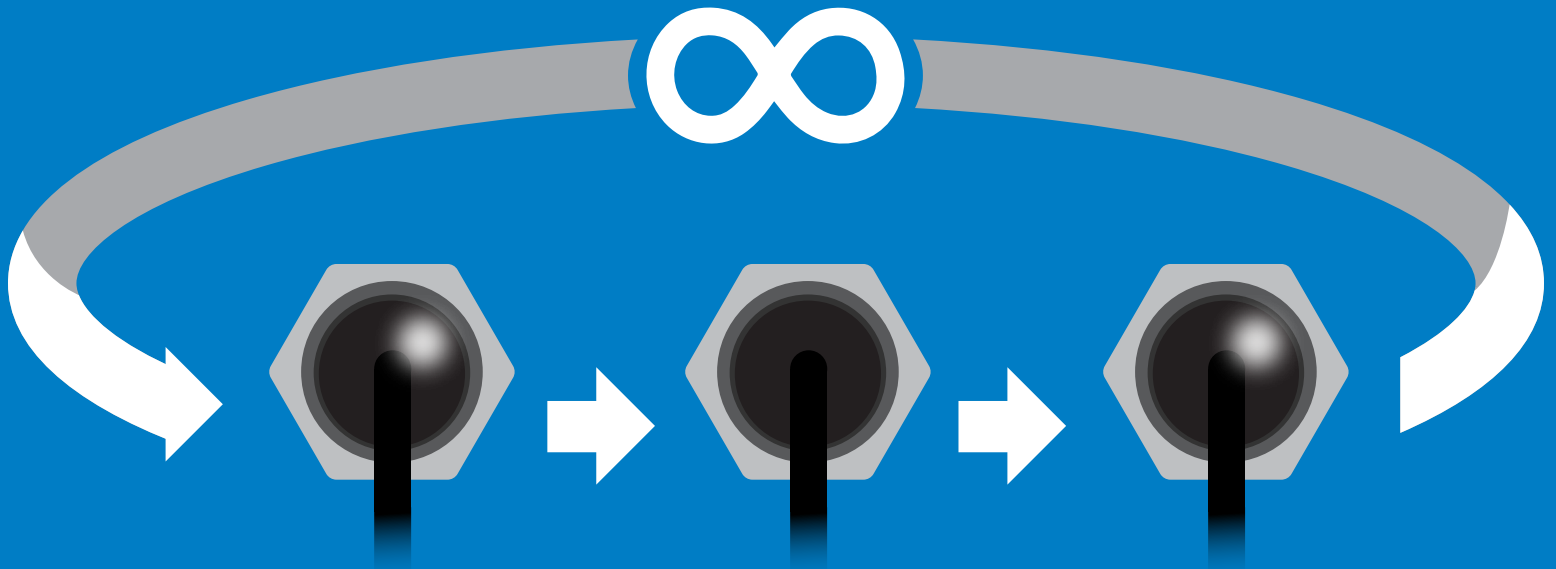
Find LED rapidly blinks white, used to help find device

Example

Response

Find

 Wait 300ms **1** Dec **0** Null



Taking reading

Command syntax

900ms  processing delay

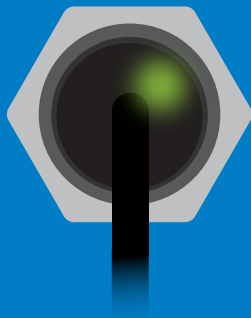
R return 1 reading

Example

Response

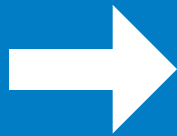
R

 Wait 900ms	1 Dec	20.95 ASCII	0 Null
---	-----------------	-----------------------	------------------

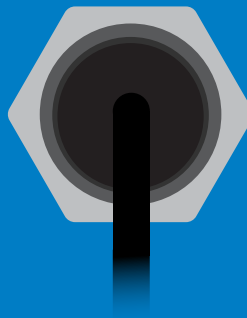


Green

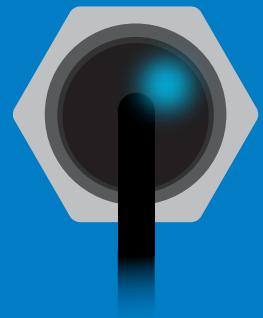
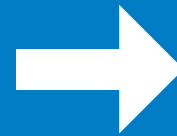
Taking reading



Wait 900ms



Transmitting



Cyan

Standby

Alarm

300ms  processing delay

Command syntax

The alarm pin will = 1 when O2 levels are > alarm set point. Alarm tolerance sets how far below the set point O2 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 – 60)
- Alarm,? alarm set?

Example

Response


Alarm,en,1

 Wait 300ms **1** **0** Enable alarm
Dec Null

Alarm,5.5

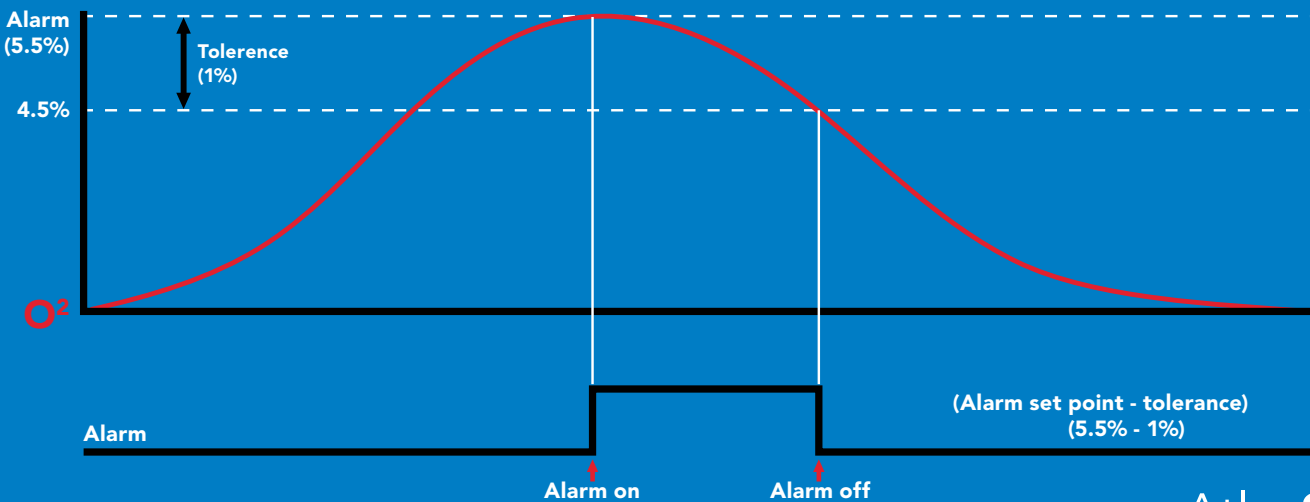
 Wait 300ms **1** **0**
Dec Null

Alarm,tol,1

 Wait 300ms **1** **0** O2 level must fall one percentage point below set point for alarm to reset.
Dec Null

Alarm,?

 Wait 300ms **1** **?,alarm,5.50,1.00,1** **0** if all are enabled
Dec ASCII Null



Calibration

1300ms  processing delay

After ~1 year the sensor may need re-calibration. A single point calibration to atmospheric O2 levels is all that's needed. 0 point calibration can also be done if accuracy at low O2 levels is needed.

Command syntax

- Cal,nn.nn calibration to O2 levels at your altitude. nn.nn =‰O2
- Cal,0 calibrate device to 0 dissolved oxygen
- Cal,clear delete calibration data
- Cal,? device calibrated?


Example

Response

Cal,20.95

 Wait 1300ms **1** **0**
Dec Null
Calibrated to O2 concentration at sea level


Cal,0

 Wait 1300ms **1** **0**
Dec Null

Cal,clear

 Wait 300ms **1** **0**
Dec Null

Cal,?

 Wait 300ms **1** **?Cal,0** **0** or **1** **?Cal,1** **0**
Dec ASCII Null Dec ASCII Null
or **1** **?Cal,2** **0**
Dec ASCII Null

Altitude (feet)	Altitude (meters)	%
1,000	305	20.1
5,000	1,524	17.3
10,000	3,048	14.3

Temperature compensation

Command syntax

Air temperature affects how the sensor works, not the actual O2 concentration in the air.

- T,n n = any value; floating point or int 300ms  processing delay
- T,? compensated temperature value?
- RT,n set temperature compensation and take a reading

Example

Response

T,19.5

 **Wait 300ms** **1** **0**
Dec Null

RT,19.5

 **Wait 900ms** **1** **20.95** **0** Temperature compensated O2 reading
Dec ASCII Null

T,?

 **Wait 300ms** **1** **?T,19.5** **0**
Dec ASCII Null

Enable/disable parameters from output string

Command syntax

300ms  processing delay

O, [parameter],[1,0]

enable or disable output parameter

O,?

enabled parameter?

Example

Response

O,PPT,1 / O,PPT,0



1 **0**
Dec Null

enable / disable PPT

O,%,1 / O,%,0



1 **0**
Dec Null

enable / disable
percent concentration

O,?



1 **? , O , % , PPT** **0**
Dec ASCII Null

if both are enabled

Parameters

PPT O₂ in parts per thousand
% O₂ in percent concentration

Followed by 1 or 0

1 enabled
0 disabled

*** If you disable all possible data types your readings will display "no output".**

Naming device

300ms  processing delay

Command syntax

Do not use spaces in the name

Name,n	set name	n =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Name,	clears name		Up to 16 ASCII characters															
Name,?	show name																	

Example

Response

Name,



1 0
Dec Null

name has been cleared

Name,zzt



1 0
Dec Null

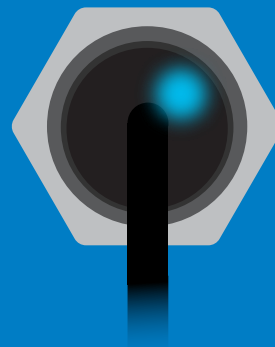
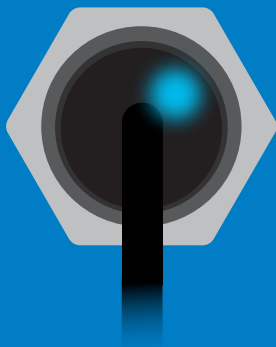
Name,?



1 ?Name,zzt 0
Dec ASCII Null

Name,zzt

Name,?



1 0

1 ?Name,zzt 0

Device information

Command syntax

300ms  processing delay

i device information

Example

Response

i



Wait 300ms

1

Dec

?i,O2,1.00

ASCII

0

Null

Response breakdown

?i, O2, 1.00
↑ ↑
Device Firmware

Reading device status

Command syntax

300ms  processing delay

Status voltage at Vcc pin and reason for last restart

Example

Response

Status

 **1** **?Status,P,5.038** **0**
Wait 300ms Dec ASCII 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

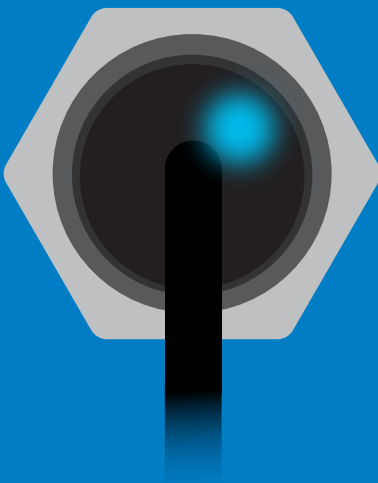
no response

Do not read status byte after issuing sleep command.

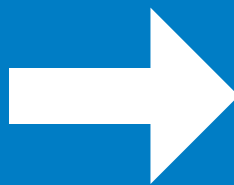
Any command

wakes up device

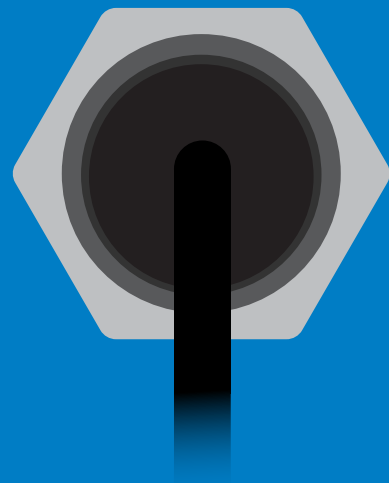
5V	STANDBY 14.6 mA	SLEEP 0.5 mA
3.3V	13.7 mA	0.4 mA



Standby



Sleep



Sleep

Protocol lock

Command syntax

300ms  processing delay

Plock,1 enable Plock

Plock,0 disable Plock

Plock,? Plock on/off?

Locks device to I²C mode.

default

Example

Response

Plock,1


Wait 300ms


1	0
Dec	Null

Plock,0


Wait 300ms

1	0
Dec	Null

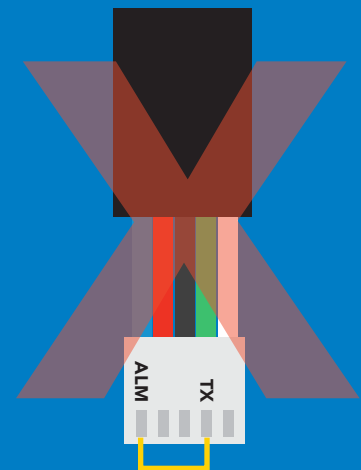
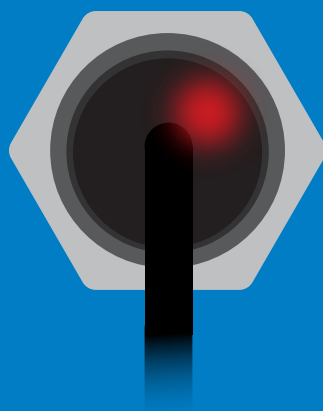
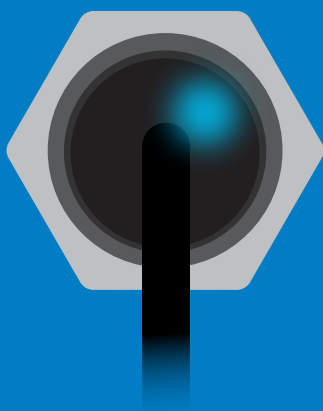
Plock,?


Wait 300ms

1	?Plock,1	0
Dec	ASCII	Null

Plock,1

Baud, 9600



cannot change to UART

cannot change to UART

I²C address change

Command syntax

300ms  **processing delay**

I2C,n sets I²C address and reboots into I²C mode

Example

Response

I2C,101

device reboot
(no response given)

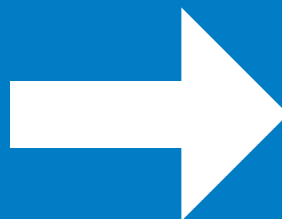
Warning!

Changing the I²C address will prevent communication between the circuit and the CPU until the CPU is updated with the new I²C address.

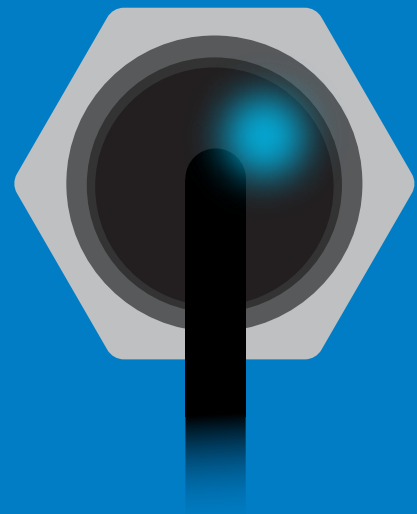
Default I²C address is **108 (0x6C)**.

n = any number 1 – 127

I2C,101



(reboot)



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

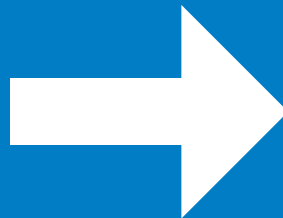
Response

Factory

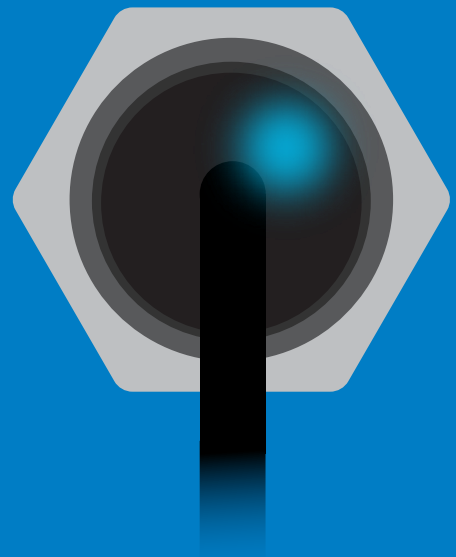
device reboot
(no response given)

Clears custom calibration
LED on
Response codes enabled

Factory



(reboot)



Change to UART mode

Command syntax

Baud,n switch from I²C to UART

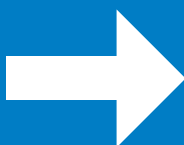
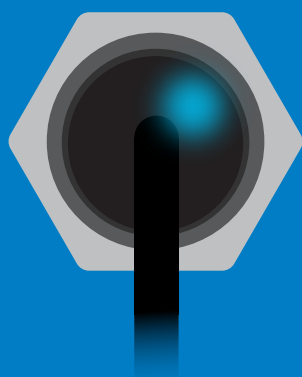
Example

Baud,9600

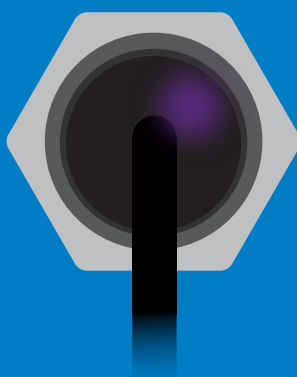
Response

reboot in UART mode
(no response given)

n = [300
1200
2400
9600
19200
38400
57600
115200



Baud,9600



(reboot)

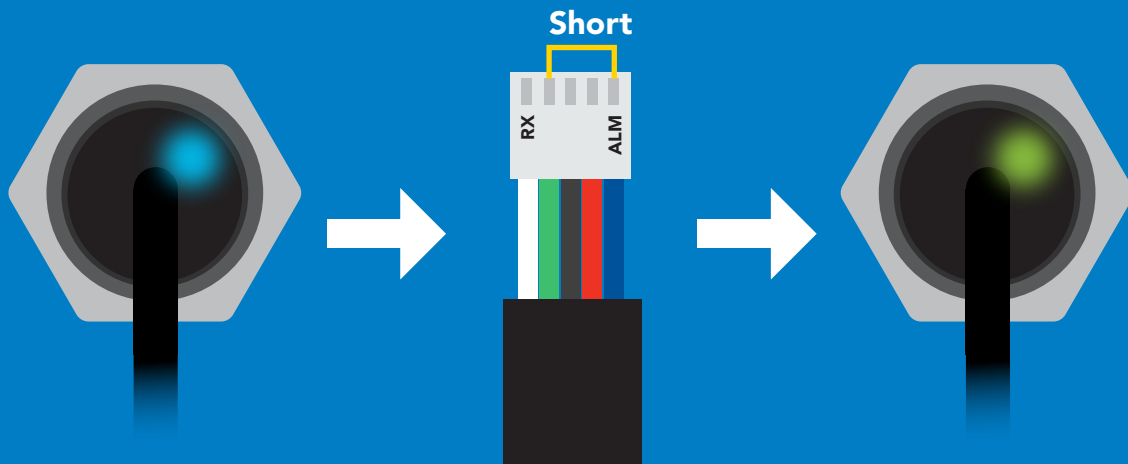


Changing to
UART mode

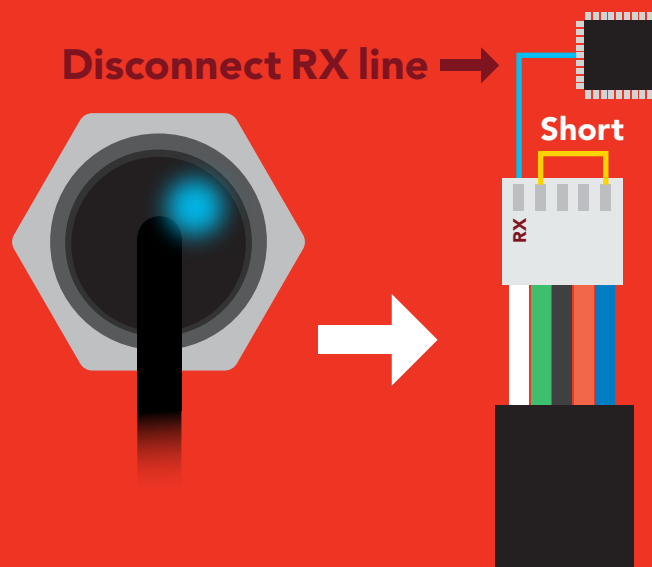
Manual switching to UART

- 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



Datasheet change log

Datasheet V 1.6

Revised artwork on page 5.

Datasheet V 1.5

Corrected typos found pages 14 & 39.

Datasheet V 1.4

Revised info on the cover page

Datasheet V 1.3

Revised naming device info on pages 23 & 47.

Datasheet V 1.2

Revised info for "Pin out" on page 5.

Datasheet V 1.1

Revised info for the Alarm command on pages 19 & 43.

Datasheet V 1.0

New datasheet

Firmware updates

V1.0 – Initial release (June 3, 2020)

V1.01 – Initial release (June 18, 2020)

- Fixed bug with the alarm command not working in certain circumstances.

Warranty

Atlas Scientific™ Warranties the EZO-O2™ Embedded Oxygen Sensor to be free of defect during the debugging phase of device implementation, or 30 days after receiving the EZO-O2™ Embedded Oxygen Sensor (which ever comes first).

The debugging phase

The debugging phase as defined by Atlas Scientific™ is the time period when the EZO-O2™ Embedded Oxygen Sensor is connected into a bread board, or shield. If the EZO-O2™ Embedded Oxygen Sensor is being debugged in a bread board, the bread board must be devoid of other components. If the EZO-O2™ Embedded Oxygen Sensor is being connected to a microcontroller, the microcontroller must be running code that has been designed to drive the EZO-O2™ Embedded Oxygen Sensor exclusively and output the EZO-O2™ Embedded Oxygen 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-O2™ Embedded Oxygen Sensor warranty:

- **Soldering any part to the EZO-O2™ Embedded Oxygen Sensor.**
- **Running any code, that does not exclusively drive the EZO-O2™ Embedded Oxygen Sensor and output its data in a serial string.**
- **Embedding the EEZO-O2™ Embedded Oxygen 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-O2™ Embedded Oxygen Sensor, against the thousands of possible variables that may cause the EZO-O2™ Embedded Oxygen 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-O2™ Embedded Oxygen Sensor continued operation. This is because that would be equivalent to Atlas Scientific™ taking responsibility over the correct operation of your entire device.