Atlas Scientific™
Environmental Robotics

V 3.4

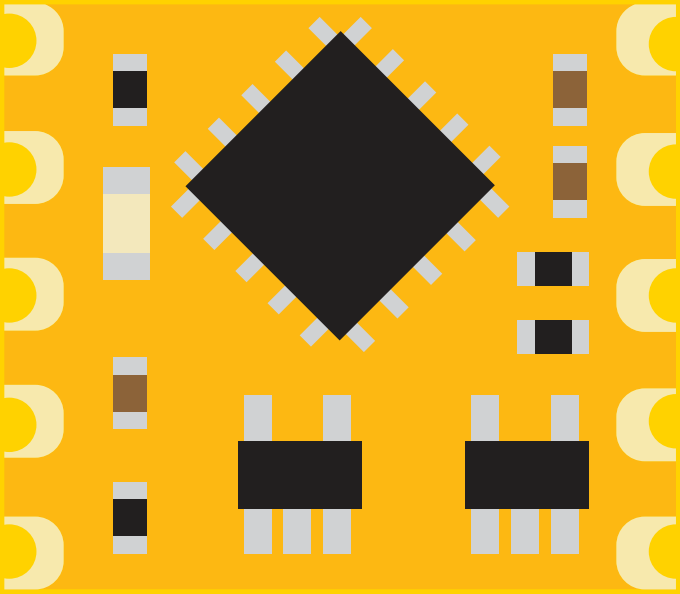
Revised 8/23

OEM-DO™

Embedded Dissolved Oxygen Circuit

ISO 5814 Compliant

Reads	**Dissolved Oxygen**
Range	**0.01 – 100+ mg/L** **0.01 – 400+ % saturation**
Accuracy	**+/– 0.05 mg/L**
Response time	**1 reading every 420ms**
Supported probes	**Any galvanic probe**
Calibration	**1 or 2 point**
Temperature, salinity and pressure compensation	**Yes**
Data protocol	**SMBus/I²C**
Default I²C address	**0x67**
Operating voltage	**3.0V – 5.5V**
Data format	**ASCII**



RoHS COMPLIANT

PATENT PROTECTED

Written by Jordan Press
Designed by Noah Press

This is an evolving document, check back for updates.

STOP

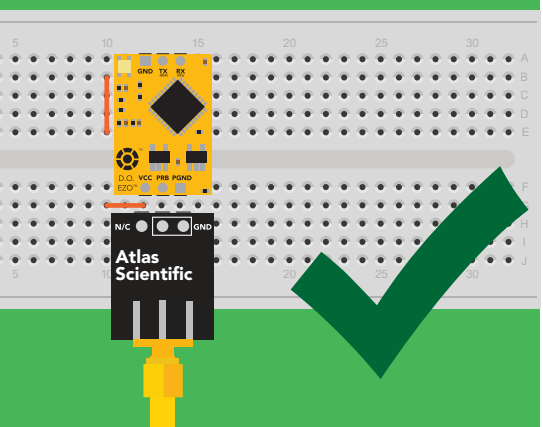
SOLDERING THIS DEVICE VOIDS YOUR WARRANTY.

Before purchasing the Dissolved Oxygen OEM™ read this data sheet in its entirety. This product is designed to be surface mounted to a PCB of your own design.

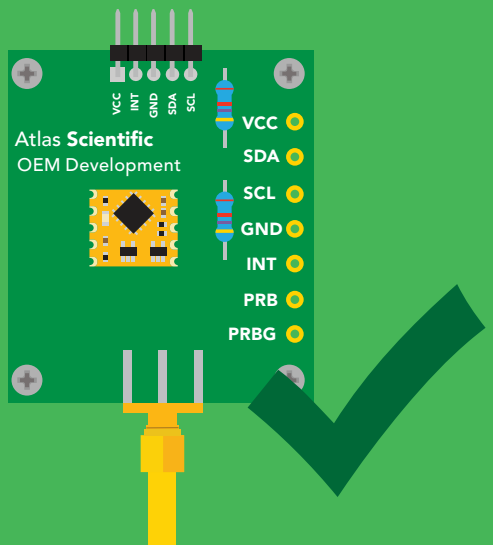
This device is designed for electrical engineers who are familiar with embedded systems design and programing. If you, or your engineering team are not familiar with embedded systems design and programing, Atlas Scientific does not recommend buying this product.

Unfamiliar with DO sensing? Try our EZO™ DO circuit first.

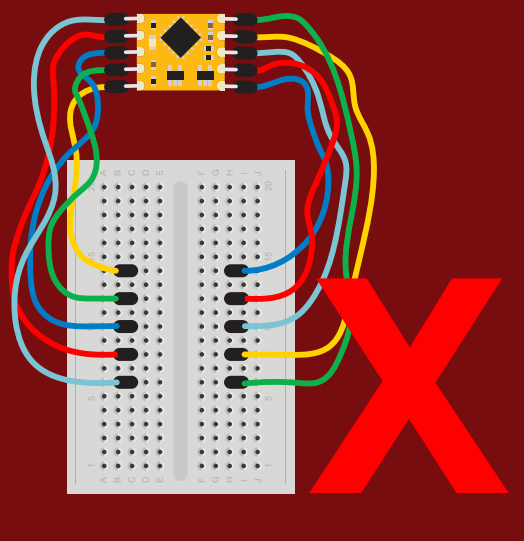
It's much easier to use, and provides a good working reference.

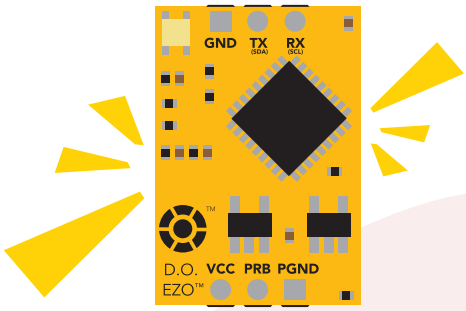


Get this device working in our OEM Development board first!



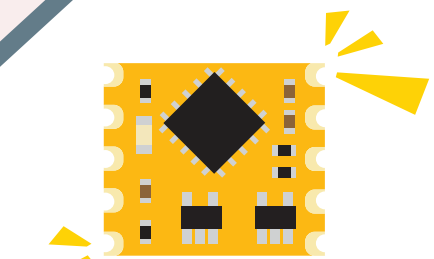
Do not solder wires to this device.





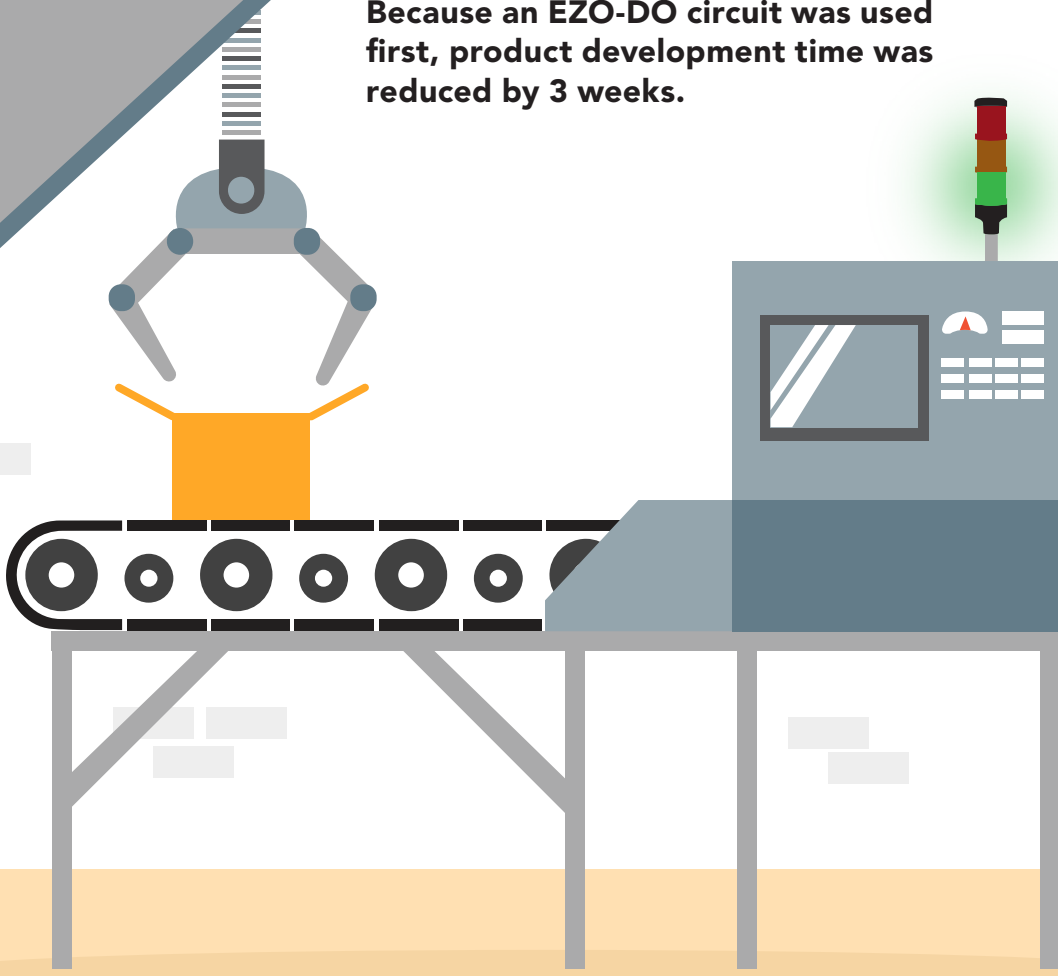
EZO-DO

Getting an EZO-DO circuit working first will significantly speed up OEM development. It will act as a working reference model; making it easy to debug OEM issues.



DO-OEM

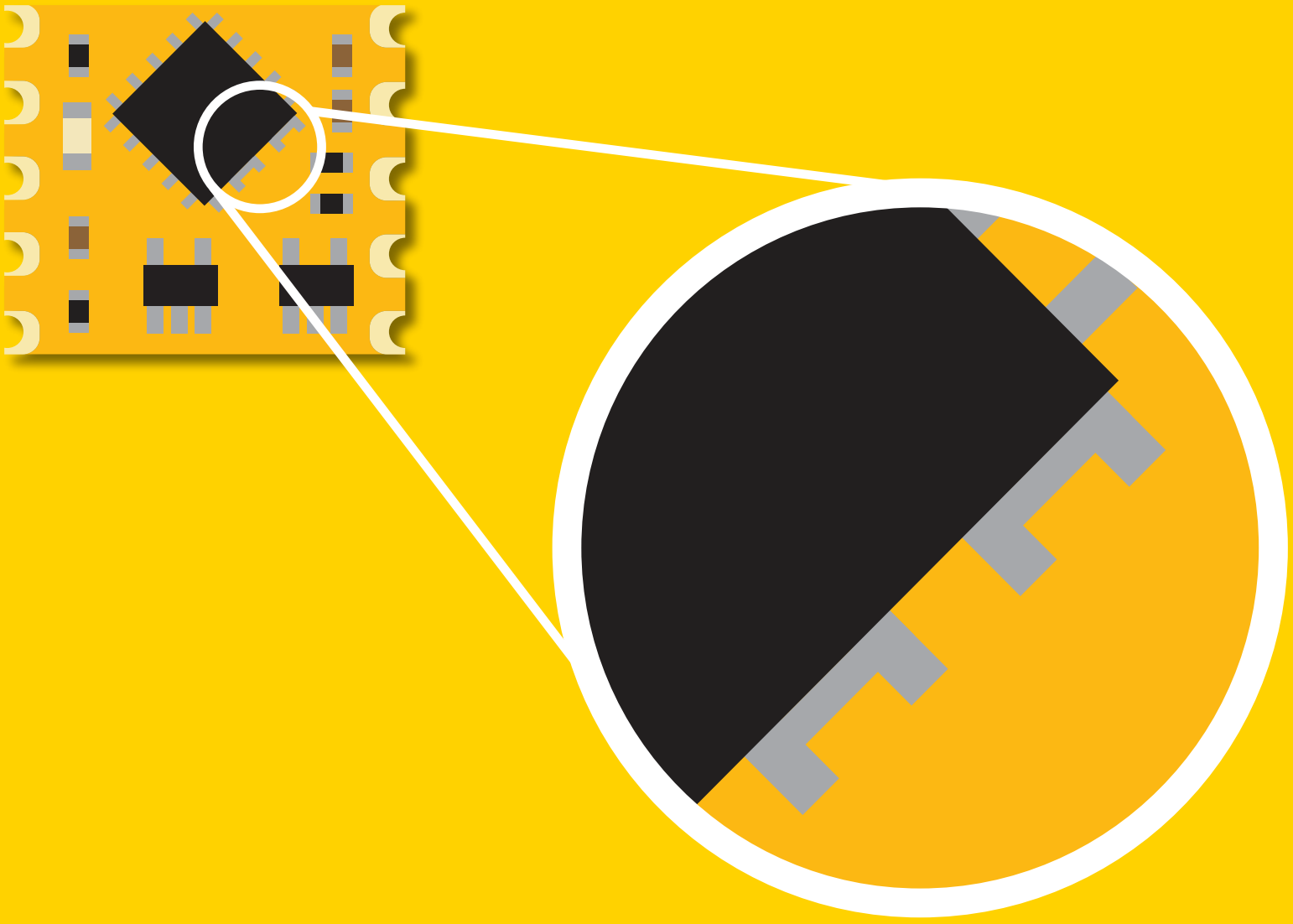
Because an EZO-DO circuit was used first, product development time was reduced by 3 weeks.



Attention

You may see these four pins soldered together.

THIS IS NOT A MISTAKE.



AtlasScientific
Environmental Robotics

Table of contents

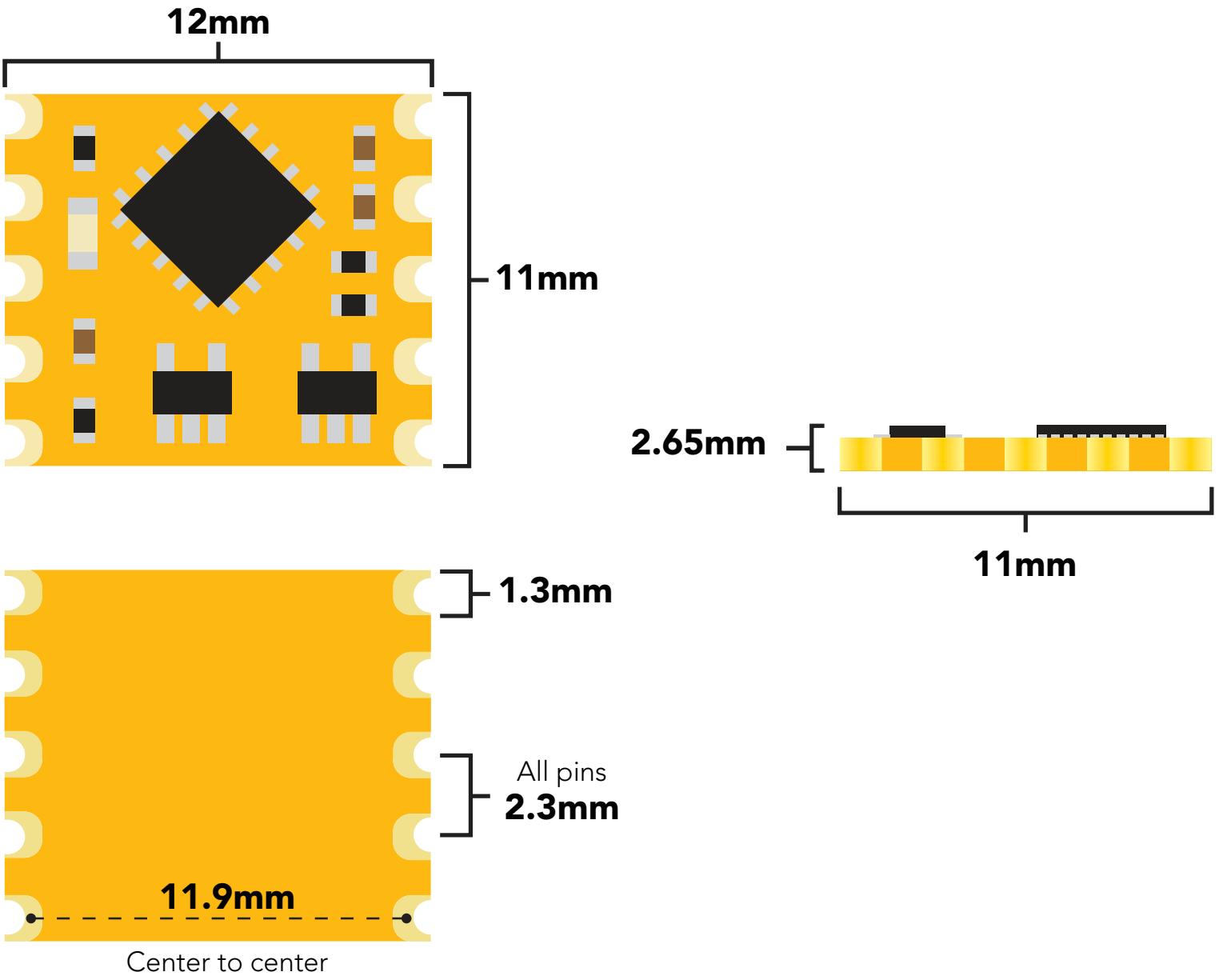
OEM circuit dimensions 6
Power consumption 6
Absolute max ratings 6
Pin out 7
Resolution 7
Power on/start up 7
System overview 8
Reading register values 9
Writing register values 10
Sending floating point numbers 11
Receiving floating point numbers 12

REGISTERS

0x00 Device type register 14
0x01 Device version register 14
0x02 Address lock/unlock register 15
0x03 Address register 16
0x04 Interrupt control register 17
0x05 LED control register 19
0x06 Active/hibernate register 19
0x07 New reading available register 20
0x08 Calibration register 21
0x09 Calibration confirmation register 21
0x0A – 0x0D Salinity compensation registers 22
0x0E – 0x11 Pressure compensation registers 23
0x12 – 0x15 Temperature compensation registers 24
0x16 – 0x19 Salinity confirmation registers 25
0x1A – 0x1D Pressure confirmation registers 25
0x1E – 0x21 Temperature confirmation registers 25
0x22 – 0x25 D.O. in mg/L registers 26
0x26 – 0x29 D.O. in saturation registers 26

OEM electrical isolation 28
Designing your product 29
Designing your PCB 31
Recommended pad layout 32
IC tube measurements 32
Recommended reflow soldering profile 33
Pick and place usage 34
Datasheet change log 35
Firmware update 36

OEM circuit dimensions



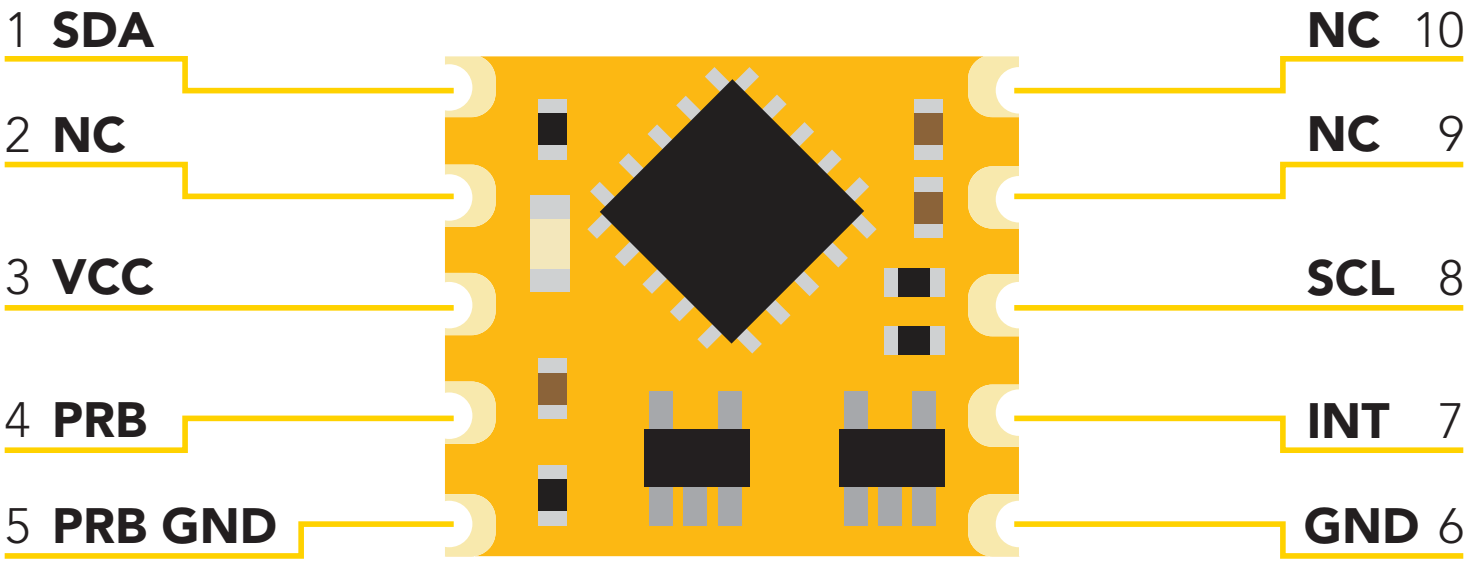
Power consumption

	LED	OPERATIONAL	HIBERNATION
3.3V	ON	2.46 mA	2.40 mA
	OFF	2.11 mA	2.09 mA

Absolute max ratings

Parameter	MIN	TYP	MAx
Storage temperature	-60 °C		150 °C
Operational temperature	-40 °C	25 °C	125 °C
VCC	3.0V	3.3V	5.5V

Pin out



Resolution

The resolution of a sensor is the smallest change it can detect in the quantity that it is measuring. The Atlas Scientific™ Dissolved Oxygen OEM™ will always produce a reading with a resolution of two decimal places.

Example
1.02 mg/L or 4.01%
9.47 mg/L or 98.78%

Power on/start up

Once the Atlas Scientific™ Dissolved Oxygen OEM™ is powered on it will be ready to receive commands and take readings after 1 ms. Communication is done using the SMBus/I²C protocol at speeds of 10 – 100 kHz.

Settings that are retained if power is cut

Calibration
I²C address

Settings that are *NOT* retained if power is cut

Active/Hibernation mode
LED control
Interrupt control
Salinity compensation
Pressure compensation
Temperature compensation

System overview

The Atlas Scientific™ Dissolved Oxygen OEM™ Class Embedded Circuit is the core electronics needed to read Dissolved Oxygen of water from any off the shelf galvanic dissolved oxygen probe. The Dissolved Oxygen OEM™ Embedded Circuit will meet, or exceed the capabilities and accuracy found in all models of bench top laboratory grade Dissolved Oxygen meters.

The Dissolved Oxygen OEM™ is an SMBus/I²C slave device that communicates to a master device at a speed of 10 – 100 kHz. Read and write operations are done by accessing **42** different 8 bit registers.

Accessible registers

Device information

0x00: Device type	R
0x01: Firmware version	R

Device address

0x02: SMBus/I²C address lock/unlock	R/W
0x03: SMBus/I²C address	R/W

Control

0x04: Interrupt control	R/W
0x05: LED control	R/W
0x06: Active/hibernate	R/W
0x07: New reading available	R/W

Calibration

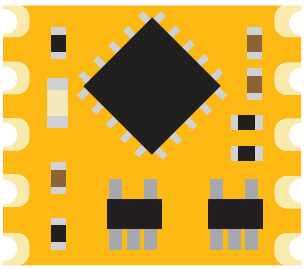
0x08: Calibration	W
0x09: Calibration confirm	R

Compensation

0x0A: Salinity compensation MSB	R/W
0x0B: Salinity compensation high byte	R/W
0x0C: Salinity compensation low byte	R/W
0x0D: Salinity compensation LSB	R/W

0x0E: Pressure compensation MSB	R/W
0x0F: Pressure compensation high byte	R/W
0x10: Pressure compensation low byte	R/W
0x11: Pressure compensation LSB	R/W

0x12: Temperature compensation MSB	R/W
0x13: Temperature compensation high byte	R/W
0x14: Temperature compensation low byte	R/W
0x15: Temperature compensation LSB	R/W



Read only R
Write only W
Read and write R/W

Confirmation

0x16: Salinity confirmation MSB	R
0x17: Salinity confirmation high byte	R
0x18: Salinity confirmation low byte	R
0x19: Salinity confirmation LSB	R

0x1A: Pressure confirmation MSB	R
0x1B: Pressure confirmation high byte	R
0x1C: Pressure confirmation low byte	R
0x1D: Pressure confirmation LSB	R

0x1E: Temperature confirmation MSB	R
0x1F: Temperature confirmation high byte	R
0x20: Temperature confirmation low byte	R
0x21: Temperature confirmation LSB	R

Sensor Data

0x22: D.O. in mg/L MSB	R
0x23: D.O. in mg/L high byte	R
0x24: D.O. in mg/L low byte	R
0x25: D.O. in mg/L LSB	R

0x26: D.O. in saturation MSB	R
0x27: D.O. in saturation high byte	R
0x28: D.O. in saturation low byte	R
0x29: D.O. in saturation LSB	R

Each Dissolved Oxygen reading takes 420ms

The default device address is **0x67**
This address can be changed.

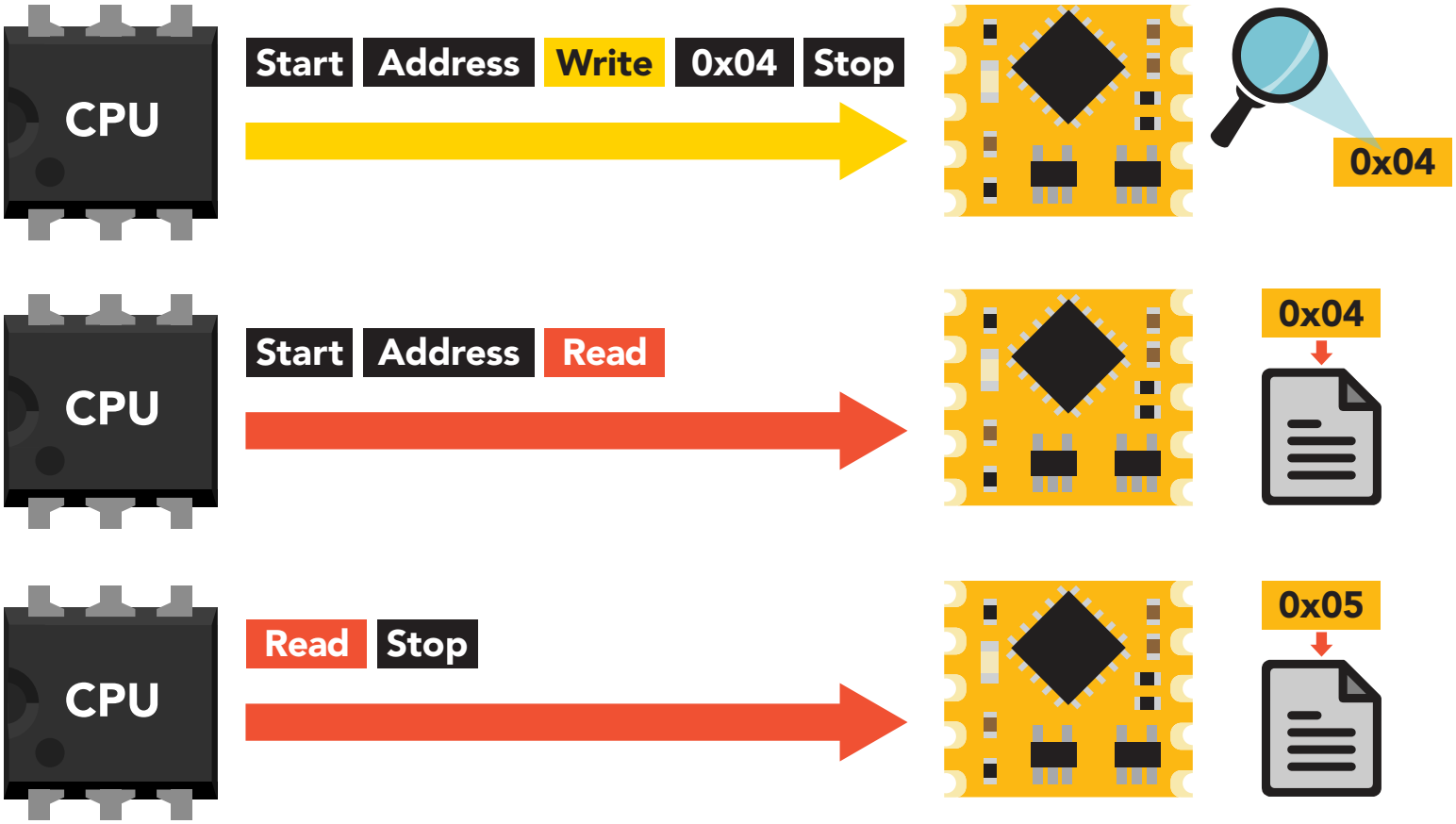
Atlas Scientific™ Environmental Robotics

Reading register values

To read one or more registers, issue a write command and transmit the register address that should be read from, followed by a stop command. Then issue a read command, the data read will be the value that is stored in that register. Issuing another read command will automatically read the value in the next register. This can go on until all registers have been read. After reading the last register, additional read commands will return 0xFF. Issuing a stop command will terminate the read event.

The default device address is **0x67**
This address can be changed.

Example
Start reading at register 0x04 and read 2 times.



Example code
reading two registers

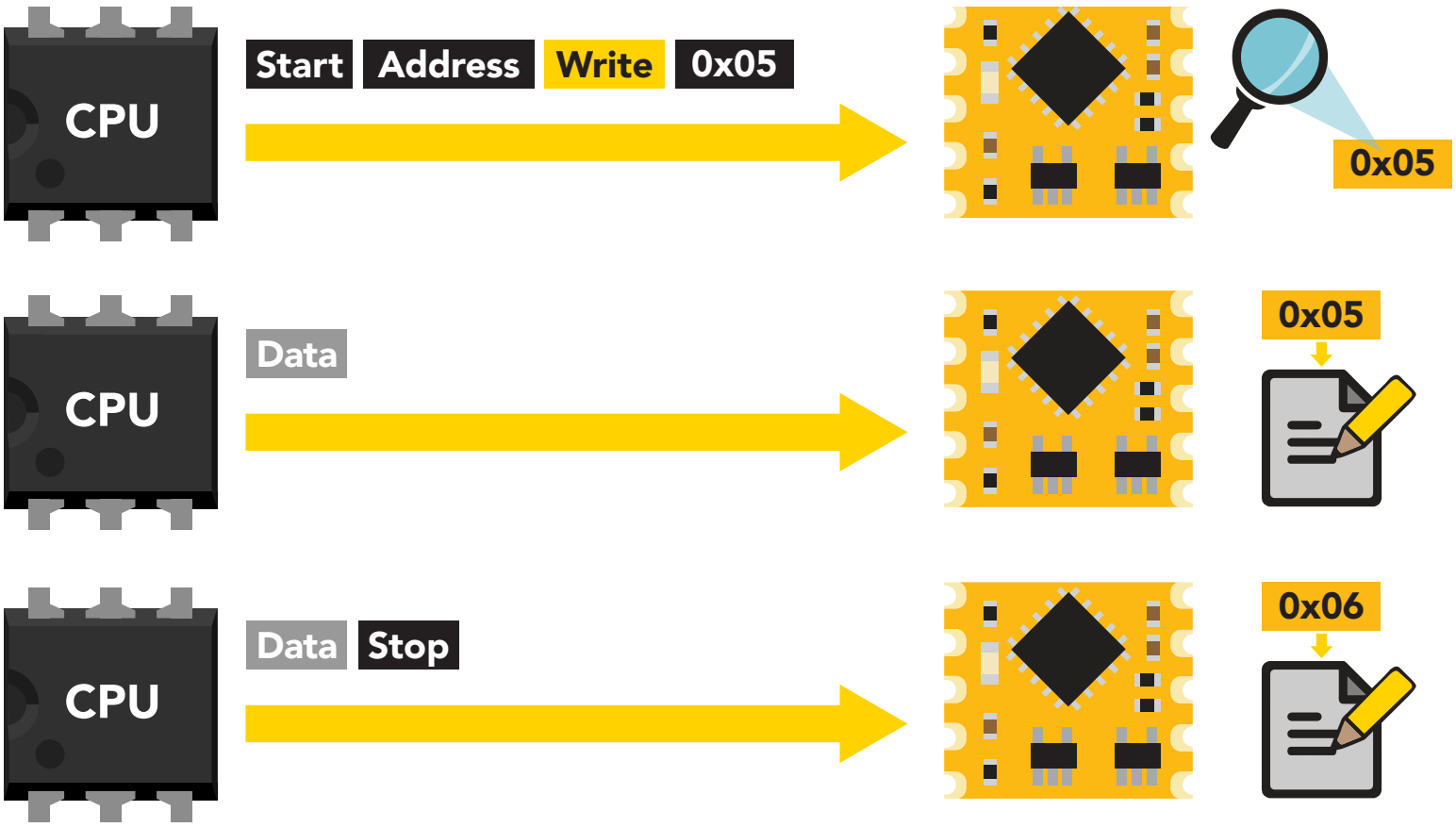
```
byte i2c_device_address=0x67;
byte reg_4, reg_5;

Wire.beginTransmission(i2c_device_address);
Wire.write(0x04);
Wire.endTransmission();

Wire.requestFrom(i2c_device_address,2);
reg_4=Wire.read();
reg_5=Wire.read();
Wire.endTransmission();
```

Writing register values

All registers can be read, but only registers marked read/write can be written to.

To write to one *(or more)* registers, issue a write command and transmit the register address that should be written to, followed by the data byte to be written. Issuing another write command will automatically write the value in the next register. This can go on until all registers have been written to. After writing to the last register, additional write commands will do nothing.

Example
Start writing at address 0x05 and write 2 values.



Example code
writing the number 1 in register 0x05 – 0x06

```
byte i2c_device_address=0x67;
byte starting_register=0x05
byte data=1;

Wire.beginTransmission(i2c_device_address);
Wire.write(starting_register);
Wire.write(data);
Wire.write(data);
Wire.endTransmission();
```


Sending floating point numbers

For ease of understanding we are calling fixed decimal numbers "floating point numbers." We are aware they are not technically floating point numbers.

It is not possible to send/receive a floating (fixed decimal) point number over the SMBus/ I^2C data protocol. Therefore, a multiplier/divider is used to remove the decimal point. Do not transmit a floating point number without property formatting the number first.

2 blocks of registers require the master to transmit a floating point number.

Pressure compensation

Temperature compensation

When transmitting a floating point number, the number must first be multiplied by 100. *(removing the decimal place)* Internally the Dissolved Oxygen OEM™ will divide the number by 100, converting it back into a floating point number.

Example

Setting the water temperature to 22.72 °C
22.72 x **100** = 2272
Transmit the number 2272 to the temperature compensation register

Setting the atmospheric pressure compensation value to 92.82 kPa
92.82 x **100** =9282
Transmit the number 9282 to the pressure compensation register

When reading back a value stored in one of these 2 register blocks the value must be divided by 100 to return it to its originally intended value.

Receiving floating point numbers

4 blocks of registers require the master receive a floating point number.

Pressure confirmation	**Temperature confirmation**
D.O. in mg/L	**D.O. in % saturation**

After receiving a value from any of these 4 register blocks, the number must be divided by 100 to convert it back into a two decimal floating point number.

Example
Reading a pressure confirmation value of 102.56 kPa
Value received = 10256
10256 / **100** = 102.56 kPa

Reading a temperature confirmation value of 99.06˚C
Value received = 9906
9906 / **100** = 99.06˚C

Reading a D.O. value of 9.56
Value received = 956
956 / **100** = 9.56 mg/L

Reading a D.O. value of 88.65% saturation
Value received = 8865
8865 / **100** = 88.65 % saturation

Registers

Device information

0x00: Device type	R
0x01: Firmware version	R

0x00 – Device type register

1 unsigned byte
Read only value = 3
3 = Dissolved Oxygen

This register contains a number indicating what type of OEM device it is.

0x01 – Firmware version register

1 unsigned byte
Read only value = 2
2 = firmware version

This register contains a number indicating the firmware version of the OEM device.

Example code
reading device type and device version registers

```
byte i2c_device_address=0x67;
byte starting_register=0x00
byte device_type;
byte version_number;

Wire.beginTransmission(i2c_device_address);
Wire.write(staring_register);
Wire.endTransmission();

Wire.requestFrom(i2c_device_address,(byte)2);
device_type = Wire.read();
version_number = Wire.read();
Wire.endTransmission();
```

0x00
0x01
0x02
0x03
0x04
0x05
0x06
0x07
0x08
0x09
0x0A
0x0B
0x0C
0x0D
0x0E
0x0F
0x10
0x11
0x12
0x13
0x14
0x15
0x16
0x17
0x18
0x19
0x1A
0x1B
0x1C
0x1D
0x1E
0x1F
0x20
0x21
0x22
0x23
0x24
0x25
0x26
0x27
0x28
0x29

Changing I²C address

0x02: SMBus/I²C address lock/unlock	R/W
0x03: SMBus/I²C address	R/W

This is a 2 step procedure

To change the I²C address, an unlock command must first be issued.

Step 1
Issue unlock command

0x02 – I²C address unlock register

1 unsigned byte
Read only value = 0 or 1
0 = **unlocked**
1 = **locked**

To unlock this register it must be written to twice.

Start	**unlock register**	**0x55**	**Stop**
Start	**unlock register**	**0xAA**	**Stop**

The two unlock commands must be sent back to back in immediate succession. No other write, or read event can occur. Once the register is unlocked it will equal 0x00 (unlocked).

To lock the register
Write any value to the register other than 0x55;
or, change the address in the Device Address Register.

Example code
address unlock

```
byte i2c_device_address=0x67;
byte unlock_register=0x02;

Wire.beginTransmission(bus_address);
Wire.write(unlock_register);
Wire.write(0x55);
Wire.endTransmission();

Wire.beginTransmission(bus_address);
Wire.write(unlock_register);
Wire.write(0xAA);
Wire.endTransmission();
```


Step 2
Change address

0x03 – I²C address register

1 unsigned byte
Default value = **0x67**
Address can be changed **0x01 – 0x7F (1–127)**

Address changes outside of the possible range **0x01 – 0x7F (1–127)** will be ignored.

After a new address has been sent to the device the Address lock/unlock register will lock and the new address will take hold. It will no longer be possible to communicate with the device using the old address.

Settings to this register are retained if the power is cut.

Example code
changing device address

```
byte i2c_device_address=0x67;
byte new_i2c_device_address=0x60;
byte address_reg=0x03;

Wire.beginTransmission(bus_address);
Wire.write(address_reg);
Wire.write(new_i2c_device_address);
Wire.endTransmission();
```

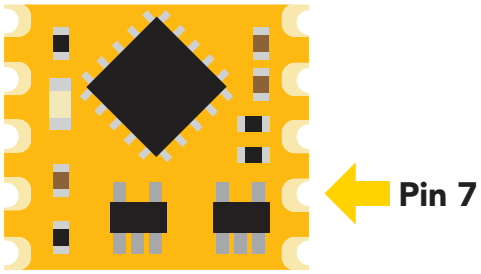

Control registers

0x04: Interrupt control	R/W
0x05: LED control	R/W
0x06: Active/hibernate	R/W
0x07: New reading available	R/W

0x04 – Interrupt control register

1 unsigned byte
Default value = 0 (disabled)

Command values
0 = disabled
2 = pin high on new reading (manually reset)
4 = pin low on new reading (manually reset)
8 = invert state on new reading (automatically reset)



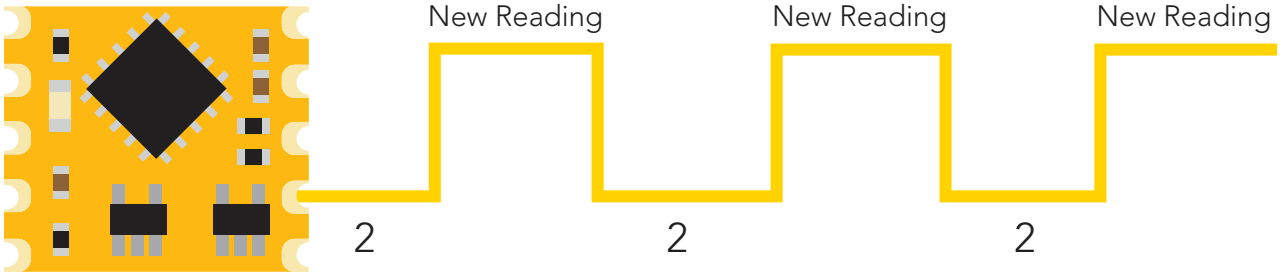
The Interrupt control register adjusts the function of pin 7 (the interrupt output pin).

Settings to this register are **not** retained if the power is cut.

Pin high on new reading

Command value = 2

By setting the interrupt control register to 2 the pin will go to a low state (0 volts). Each time a new reading is available the INT pin (pin 7) will be set and output the same voltage that is on the VCC pin.



The pin will not auto reset. 2 must be written to the interrupt control register after each transition from low to high.

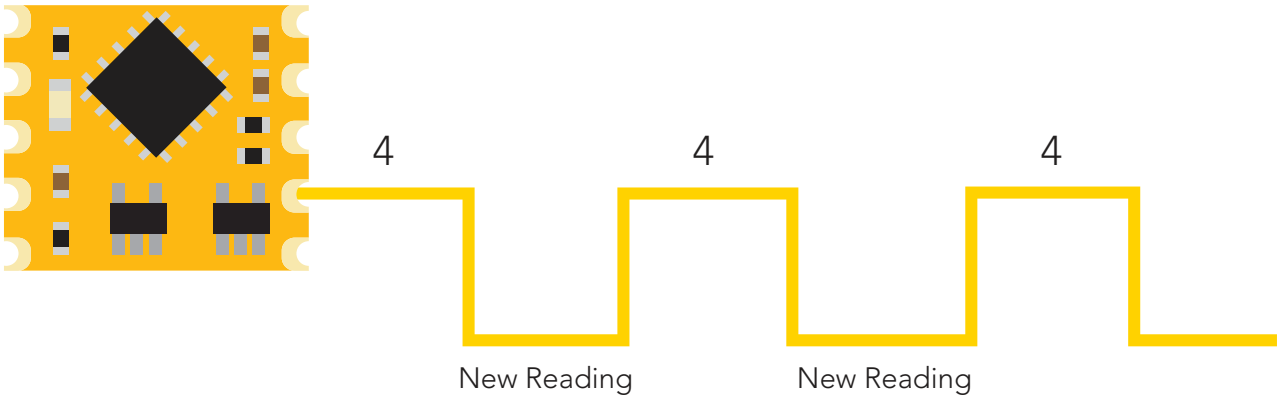
Example code
Setting pin high on new reading

```
byte i2c_device_address=0x67;
byte int_control=0x04;

Wire.beginTransmission(i2c_device_address);
Wire.write(int_control);
Wire.write(0x02);
Wire.endTransmission();
```


Pin low on new reading

Command value = 4

By setting the interrupt control register to 4 the pin will go to a high state (VCC). Each time a new reading is available the INT pin (pin 7) will be reset and the pin will be at 0 volts.



The pin will not auto set. 4 must be written to the interrupt control register after each transition from high to low.

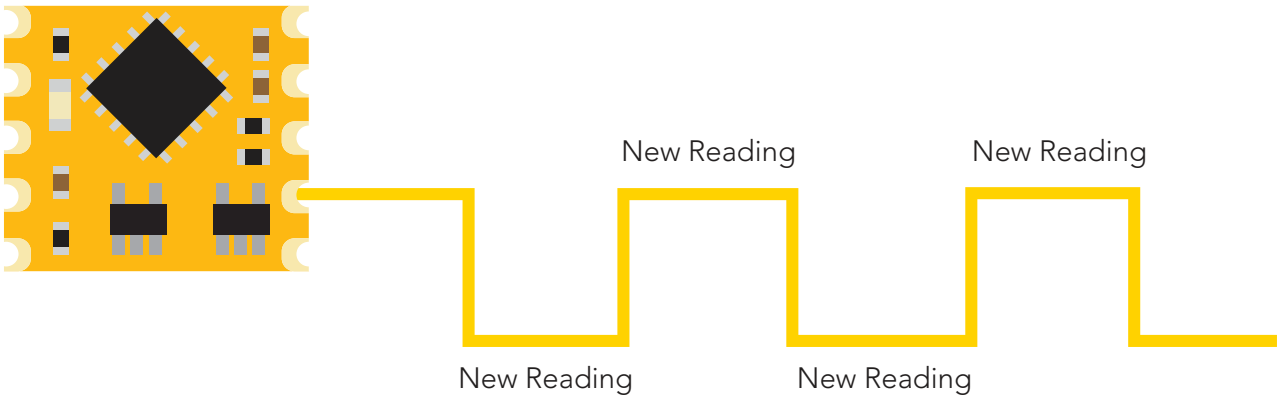
Example code
Setting pin low on new reading

```
byte I2C_device_address=0x67;
byte int_control=0x04;

Wire.beginTransmission(I2C_device_address);
Wire.write(int_control);
Wire.write(0x04);
Wire.endTransmission();
```

Invert state on new reading

Command value = 8

By setting the interrupt control register to 8 the pin will remain in whatever state it is in. Each time a new reading is available the INT pin (pin 7) will invert its state.



The pin will automatically invert its state each time a new reading is available. This setting has been specifically designed for a master device that can use an interrupt on change function.

Example code
Inverting state on new reading

```
byte i2c_device_address=0x67;
byte int_control=0x04;

Wire.beginTransmission(i2c_device_address);
Wire.write(int_control);
Wire.write(0x08);
Wire.endTransmission();
```

0x05 – LED control register

1 unsigned byte

Command values
1 = Blink each time a reading is taken
0 = Off

The LED control register adjusts the function of the on board LED. By default the LED is set to blink each time a reading is taken.

Example code
Turning off LED

```
byte i2c_device_address=0x67;
byte led_reg=0x05;

Wire.beginTransmission(i2c_device_address);
Wire.write(led_reg);
Wire.write(0x00);
Wire.endTransmission();
```

Settings to this register are **not** retained if the power is cut.

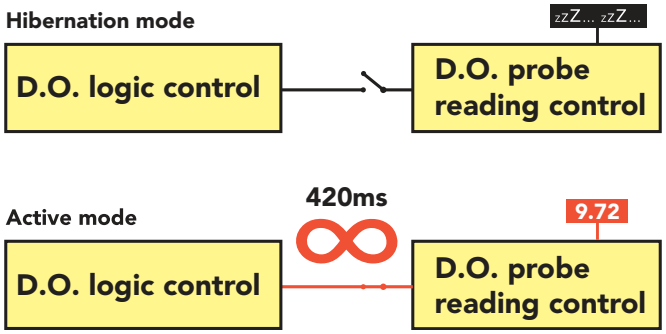
0x06 – Active/hibernate register

1 unsigned byte

To wake the device
Transmit a 0X01 to register 0x06

To hibernate the device
Transmit a 0X00 to register 0x06

This register is used to activate, or hibernate the sensing subsystem of the OEM device.



Example code
Activate Dissolved Oxygen readings

```
byte i2c_device_address=0x67;
byte active_reg=0x06;

Wire.beginTransmission(i2c_device_address);
Wire.write(active_reg);
Wire.write(0x01);
Wire.endTransmission();
```

Once the device has been woken up it will continuously take readings every 420ms. **Waking the device is the only way to take a reading. Hibernating the device is the only way to stop taking readings.**

0x07 – New reading available register

1 unsigned byte
Default value = 0 (no new reading)
New reading available = 1

Command values
0 = reset register

This register is for applications where the interrupt output pin cannot be used and continuously polling the device would be the preferred method of identifying when a new reading is available.

When the device is powered on, the New Reading Available Register will equal 0. Once the device is placed into active mode and a reading has been taken, the New Reading Available Register will move from 0 to 1.

This register will never automatically reset itself to 0. The master must reset the register back to 0 each time.

Example code
Polling new reading available register

```
byte i2c_device_address=0x67;
byte new_reading_available=0;
byte nra=0x07;

while(new_reading_available==0){
Wire.beginTransmission(i2c_device_address);
Wire.write(nra);
Wire.endTransmission();

Wire.requestFrom(i2c_device_address,(byte)1);
new_reading_available = Wire.read();
Wire.endTransmission();
delay(10);
}

if(new_reading_available==1){
call read_Dissolved Oxygen();
Wire.beginTransmission(i2c_device_address);
Wire.write(nra);
Wire.write(0x00);
Wire.endTransmission();
}
```


Calibration

0x08: Calibration	W
0x09: Calibration confirm	R

0x08 – Calibration register

1 unsigned byte

Command values
1 = clear all previous calibration
2 = calibrate to atmospheric oxygen content
3 = calibrate to 0 dissolved oxygen

After loading the register with the desired calibration type and an SMBus/I²C stop condition has been issued, calibration will occur.

The 2 register calibration block is used to perform calibration, and verify that calibration has been done.

The Dissolved Oxygen OEM™ is capable of one or two-point calibration. Once the calibration register has been set to a desired calibration type and a SMBus/I²C stop condition has been issued, calibration will occur. The calibration confirmation register is used to confirm that calibration was done.

The calibration process takes 40 ms

0x09 – Calibration confirmation register

1 unsigned byte

Read only values
0 = no calibration
1 = calibrated to atmosphere
2 = calibrated to 0 Dissolved Oxygen
3 = calibrated to both atmospheric and 0 Dissolved Oxygen

After a calibration event has been successfully carried out, the calibration confirmation register will reflect what calibration has been done.

Settings to this register are retained if the power is cut.

Salinity compensation

0x0A: Salinity compensation MSB	R/W
0x0B: Salinity compensation high byte	R/W
0x0C: Salinity compensation low byte	R/W
0x0D: Salinity compensation LSB	R/W

0x0A – 0x0D Salinity compensation registers

Unsigned long
0x0A = MSB
0x0D = LSB
Default value = 0 µs
Units = "µs" microsiemens

Salinity effects waters ability to hold oxygen. Transmitting the conductivity (in microsiemens) of the water to the salinity compensation registers will correct the dissolved oxygen readings to account for the waters salinity.

To send a new conductivity value to the salinity compensation register the value must first multiplied by 100, moved to an unsigned long and broken up into 4 bytes. Then it can be transmitted (MSB to LSB) to the salinity compensation register.

Settings to this register are **not** retained if the power is cut.

Example

Setting the register to 56,000µs
56,000 x 100 = 5,600,000
5,600,000 ➡ Unsigned long
Unsinged long = Hex (0x00, 0x55, 0x73, 0x00)

0x0A 0x0B 0x0C 0x0D

Pressure compensation

0x0E: Pressure compensation MSB	R/W
0x0F: Pressure compensation high byte	R/W
0x10: Pressure compensation low byte	R/W
0x11: Pressure compensation LSB	R/W

0x0E – 0x11 Pressure compensation registers

Unsigned long
0x0E = MSB
0x11 = LSB
Default value = 101.32 kPa
Units = "kPa" kilopascals

Pressure effects waters ability to hold oxygen. Transmitting the pressure (in kilopascals) of the atmosphere *(for high altitude measurement)* or the water pressure *(for D.O. readings at depth)* to the pressure compensation registers will correct the dissolved oxygen readings to account for the water/atmospheric pressure.

To send a new pressure value to the Pressure Compensation Register the value must first multiplied by 100, moved to an unsigned long and broken up into 4 bytes. Then it can be transmitted (MSB to LSB) to the pressure compensation register.

Pressure is always in "kPa" and can be a floating point number with no more than two decimal places.

Settings to this register are **not** retained if the power is cut.

Example

Setting the register to 135.86 kPa
135.86 x 100 = 13,586
13,586 ➡ Unsinged long
Unsinged long = Hex (0x00, 0x00, 0x35, 0x12)

0x0E	0x0F	0x10	0x11

Temperature compensation

0x12: Temperature compensation MSB	R/W
0x13: Temperature compensation high byte	R/W
0x14: Temperature compensation low byte	R/W
0x15: Temperature compensation LSB	R/W

0x12 – 0x15 Temperature compensation registers

Unsigned long
0x12 = MSB
0x15 = LSB
Default value = 20 °C
Units = °C

Temperature effects waters ability to hold oxygen. Transmitting the temperature (in °C) of the water to the temperature compensation registers will correct the dissolved oxygen readings to account for the water temperature.

To send a new temperature value to the temperature compensation register the value must first multiplied by 100, moved to an unsigned long and broken up into 4 bytes. Then it can be transmitted (MSB to LSB) to the temperature compensation register.

Settings to this register are **not** retained if the power is cut.

Example

Setting the register to 13.78°C
13.78 x 100 = 1,378
1,378 ➡ Unsigned long
Unsinged long = Hex (0x00, 0x00, 0x05, 0x62)

0x12	0x13	0x14	0x15

Salinity, pressure and temperature confirmation

0x16: Salinity confirmation MSB	R
0x17: Salinity confirmation high byte	R
0x18: Salinity confirmation low byte	R
0x19: Salinity confirmation LSB	R

0x1A: Pressure confirmation MSB	R
0x1B: Pressure confirmation high byte	R
0x1C: Pressure confirmation low byte	R
0x1D: Pressure confirmation LSB	R

0x1E: Temperature confirmation MSB	R
0x1F: Temperature confirmation high byte	R
0x20: Temperature confirmation low byte	R
0x21: Temperature confirmation LSB	R

0x16 – 0x19 Salinity confirmation registers

Unsigned long
0x16 = MSB
0x19 = LSB
Default value = 0 µs
Units = "µs" microsiemens

The value in this register is only updated when actively taking readings.

0x1A – 0x1D Pressure confirmation registers

Unsigned long
0x1A = MSB
0x1D = LSB
Default value = 101.32 kPa
Units = "kPa" kilopascals

The value in this register is only updated when actively taking readings.

0x1E – 0x21 Temperature confirmation registers

Unsigned long
0x1E = MSB
0x21 = LSB
Default value = 25 °C
Units = °C

The value in this register is only updated when actively taking readings.

This block of three 8 bit read only registers are the Salinity, pressure and temperature Confirmation registers. They are used to confirm that any one (or all) of the compensation values that were used to take the D.O. readings have been applied to the current reading. Whatever values are in these registers are the values that will be used to calculate the next D.O. reading.

Sensor data

0x22: D.O. in mg/L MSB	R
0x23: D.O. in mg/L high byte	R
0x24: D.O. in mg/L low byte	R
0x25: D.O. in mg/L LSB	R

0x26: D.O. in saturation MSB	R
0x27: D.O. in saturation high byte	R
0x28: D.O. in saturation low byte	R
0x29: D.O. in saturation LSB	R

This 8 byte read only block of registers contains the D.O. readings. The first 4 byte block is the D.O. readings in mg/L. While the second 4-byte block is the D.O. readings in percent saturation.

Each time a new reading is taken the old reading is over written. The D.O. readings are stored as a 4 byte long (4 bytes for mg/L and 4 bytes for percent saturation) After reading each of the 4 bytes (MSB to LSB) and reassembling them into a long, the value should be moved (or cast) into a float and then divide by 100.

0x22 – 0x25 D.O. in mg/L registers

Unsigned long
0x22 = MSB
0x25 = LSB
Default value = 0 mg/L
Units = mg/L

The value in this register is only updated when actively taking readings.

The last D.O. reading taken, in mg/L is stored in these four registers.

To read the value in this register, read the bytes MSB to LSB and assign them to an unsigned long, cast to a float. Divide that number by 100.

0x26 – 0x29 D.O. in % saturation registers

Unsigned long
0x26 = MSB
0x29 = LSB
Default value = 0 % sat
Units = % sat

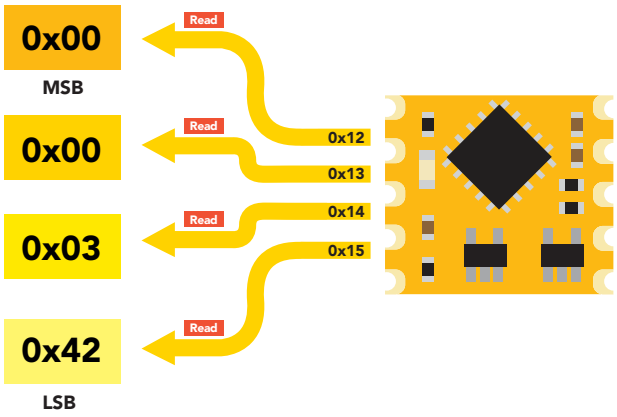
The value in this register is only updated when actively taking readings.

The last D.O. reading taken, in percent saturation is stored in these four registers.

To read the value in this register, read the bytes MSB to LSB and assign them to an unsigned long, cast to a float. Divide that number by 100.

Reading Dissolved Oxygen of 8.34 mg/L

Step 1 read 4 bytes



Step 2 read unsigned long

Step 3 cast unsigned long to a float

Step 4 divide by 100

AtlasScientific™
Environmental Robotics

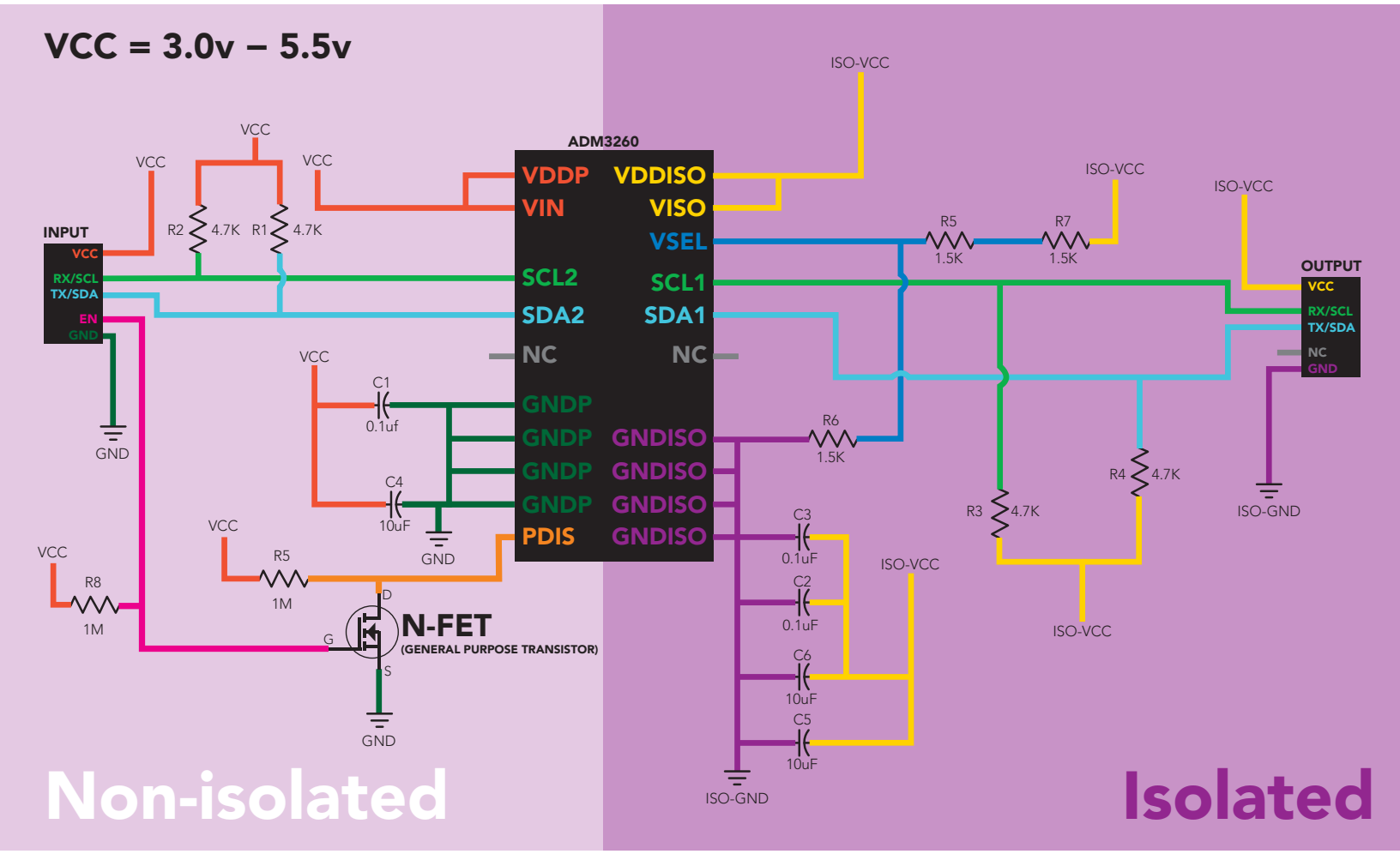
OEM electrical isolation

If the Dissolved Oxygen OEM™ Class Embedded Dissolved Oxygen Circuit is going to be used in consumer, industrial, or scientific /medical applications electrical isolation is strongly recommended. Electrically isolating the device will insure that the readings are accurate, the Dissolved Oxygen probe does not interfere with other sensors and that outside electrical noise does not affect the device.

The goal of electrically isolating the Dissolved Oxygen OEM™ circuit is to insure that the device no longer shares a common ground with the master CPU, other sensors and other devices that are can be traced back to a common ground. It is important to keep in mind that simply isolating the power and ground is not enough. Both data lines (SDA, SCL) and the INT pin must also be isolated.

This technology works by using tiny transformers to induce the voltage across an air gap. PCB layout requires special attention for EMI/EMC and RF Control, having proper ground planes and keeping the capacitors as close to the chip as possible are crucial for proper performance. The two data channels have a 4.7kΩ pull up resistor on both the isolated and non-isolated lines (R1, R2, R3, and R4) The output voltage is set using a voltage divider (R5, R6, and R,7) this produces a voltage of 3.9V regardless of your input voltage.

Isolated ground is different from non-isolated ground, these two lines should not be connected together.

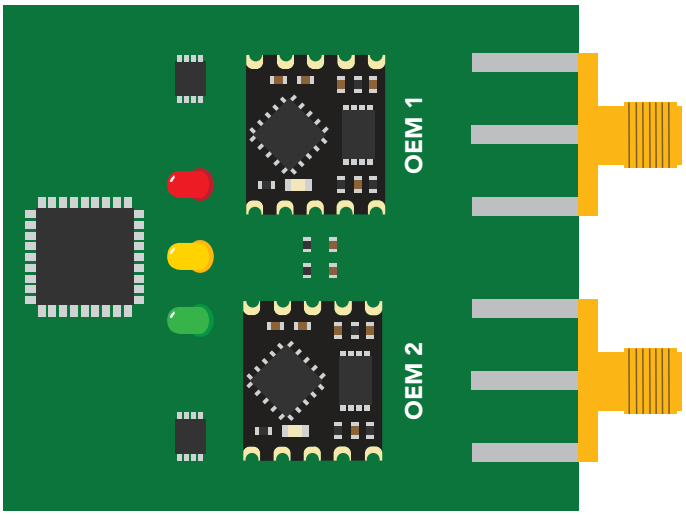


Designing your product

The Dissolved Oxygen OEM™ circuit is a sensitive device. Special care ***MUST*** be taken to ensure your Dissolved Oxygen readings are accurate.

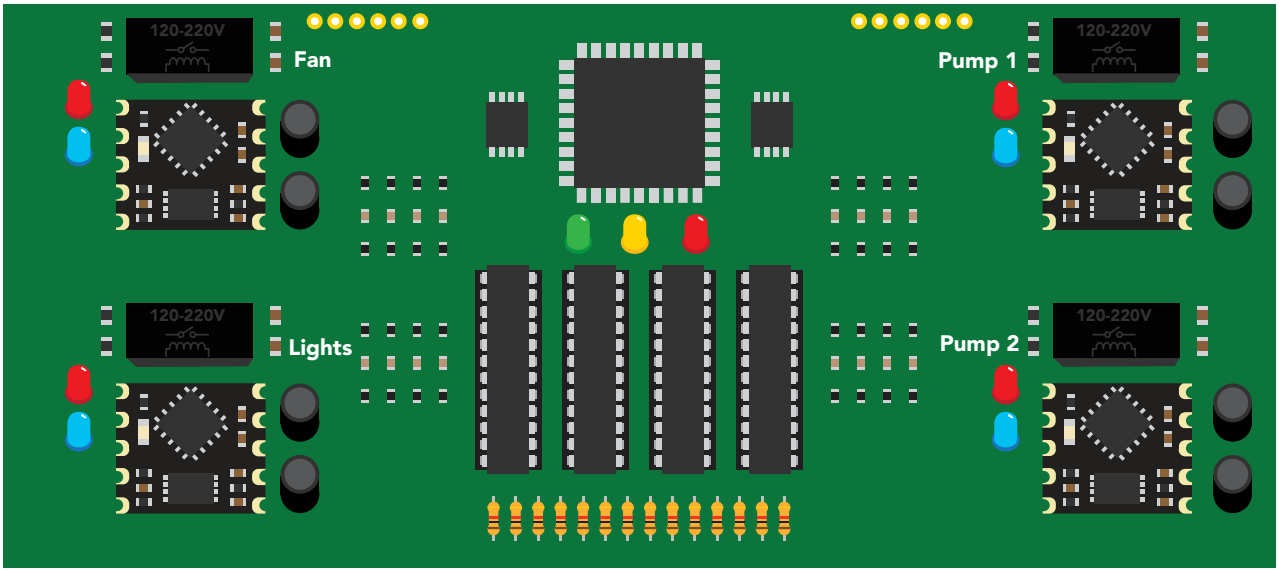
Simple design

Simple low voltage computer systems experience little to no problems during development and have no reported issues from the target customer.



Complex design

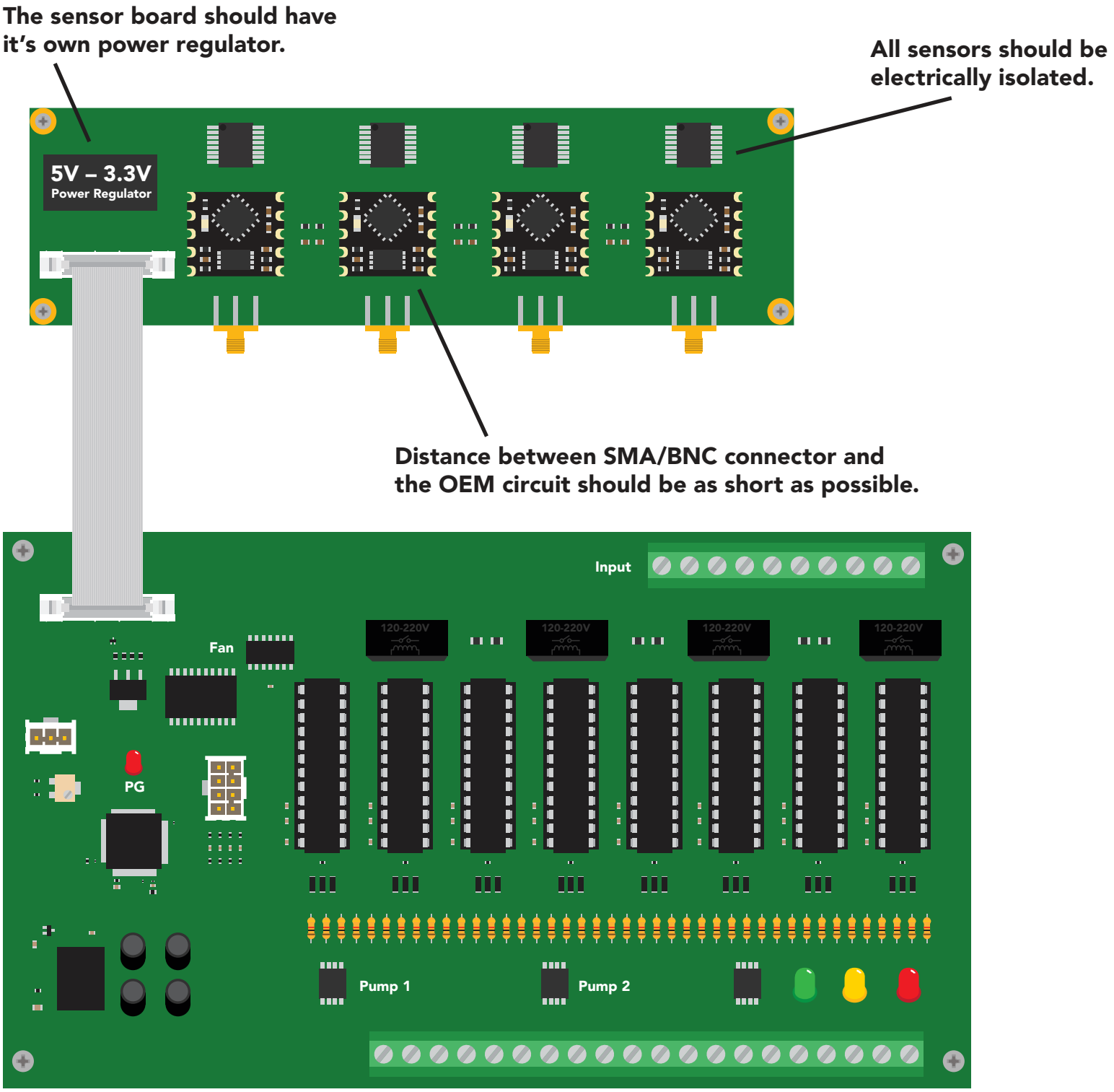
Complex computer systems with multiple voltages and switching, can lead to extended and unnecessary debugging time. Target customers can experience frequent accuracy issues.



How to add chemical sensing to a complex computer system

Placing the OEM™ circuits onto their own board is ***strongly recommended***; Not only does this help keep the design layout simple and easy to follow, it also significantly reduces debugging and development time.

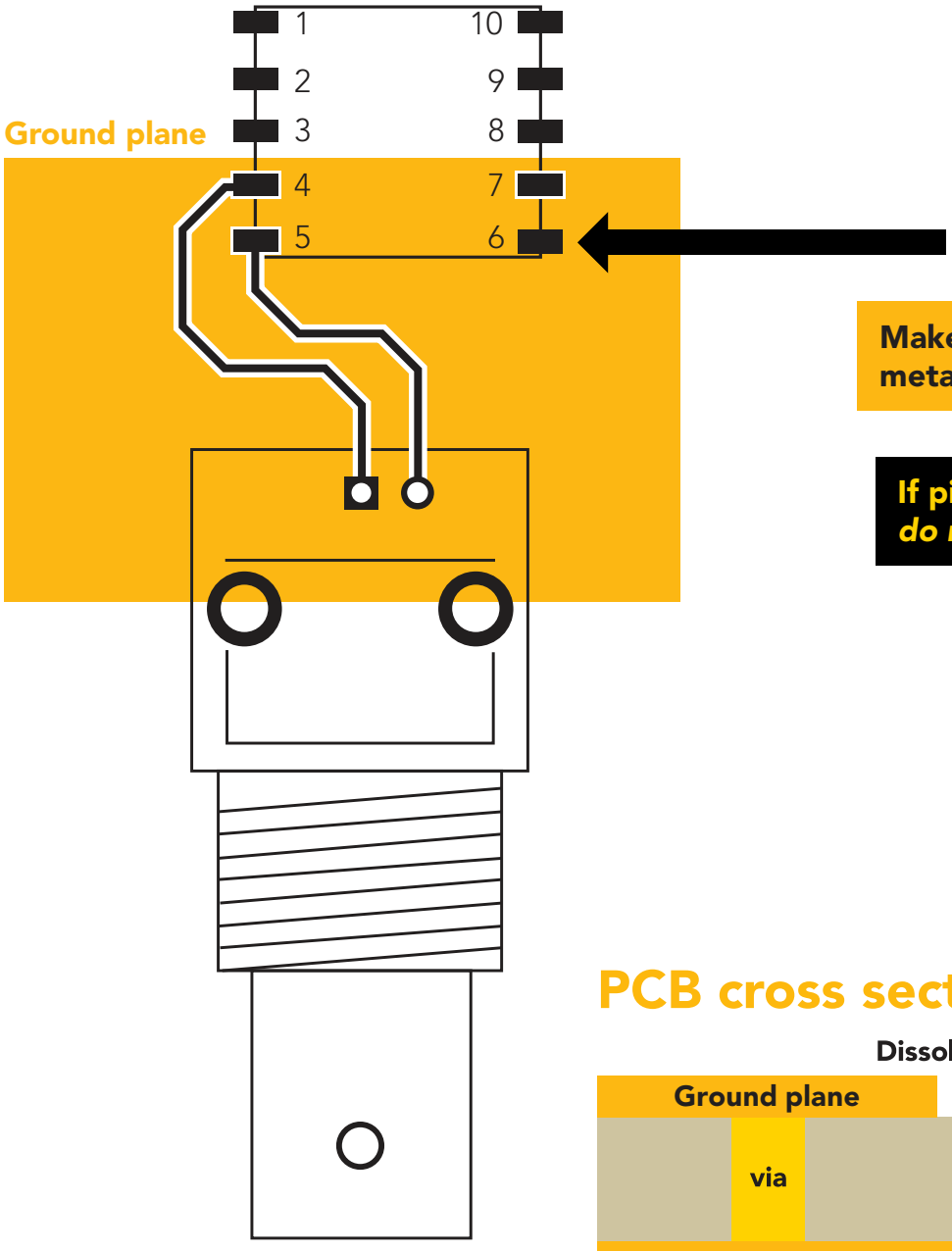
Target customers will experience accurate, stable and repeatable readings for the life of your product.



AtlasScientific™ Environmental Robotics

Designing your PCB

Create the traces as short as possible from the Dissolved Oxygen OEM™ to your probe connection. Keep the traces on your top layer, keep a distance of 1mm for any other trace. use 0.4mm trace width. Use a ground plane underneath the traces and probe connection.



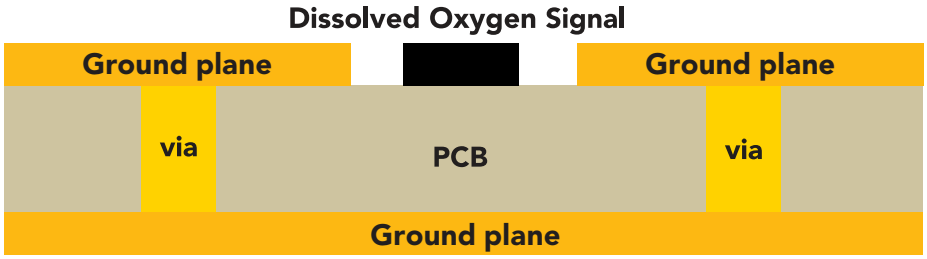
Connect pin 6 to the ground plane.

Make sure there are no vias or exposed metal underneath the D.O. OEM™ circuit.

If pin 7(INT) is unused leave it floating, *do not connect pin 7 to VCC or ground.*

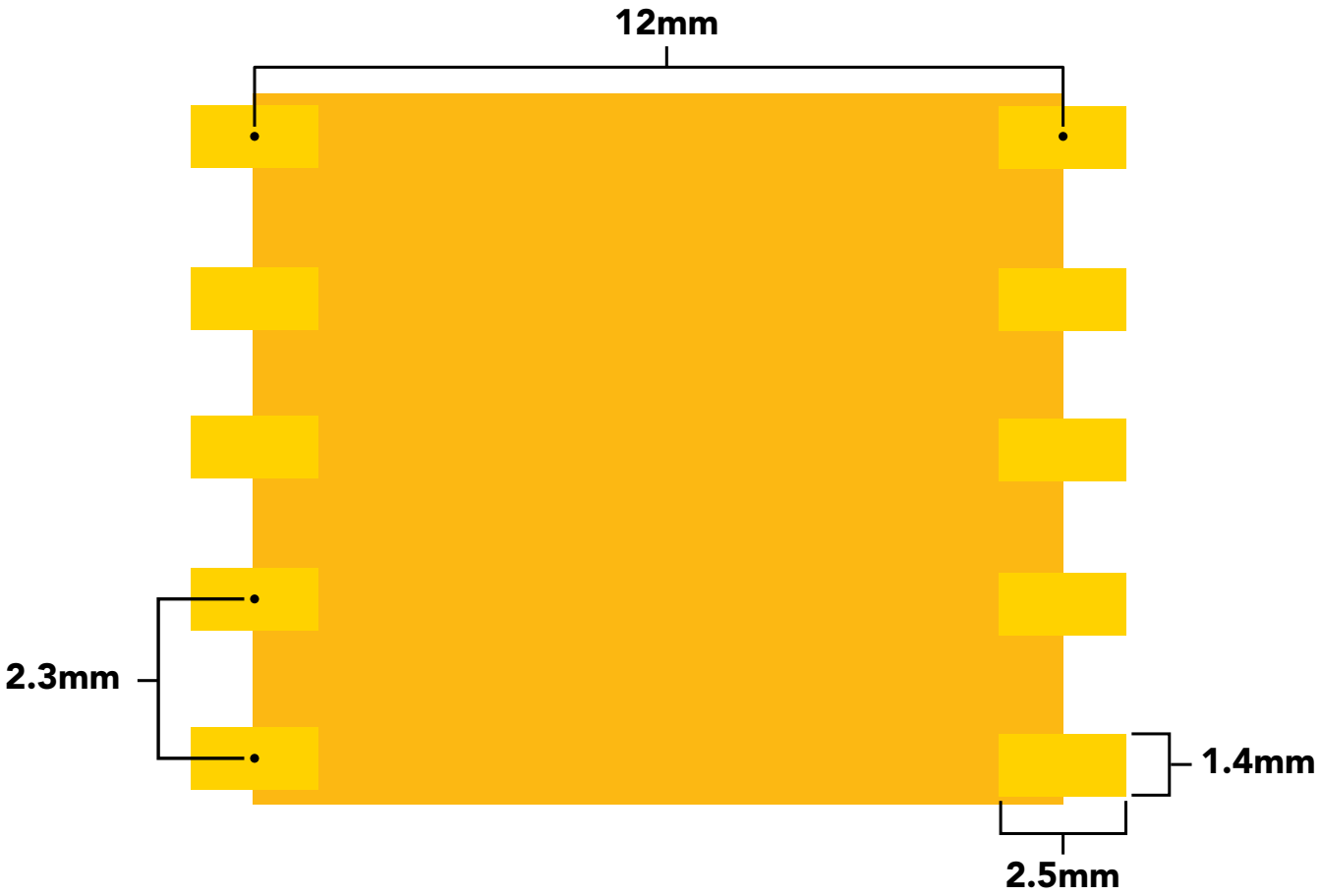
Keep the traces for both probe and probe ground as short as possible.

PCB cross section of the signal path



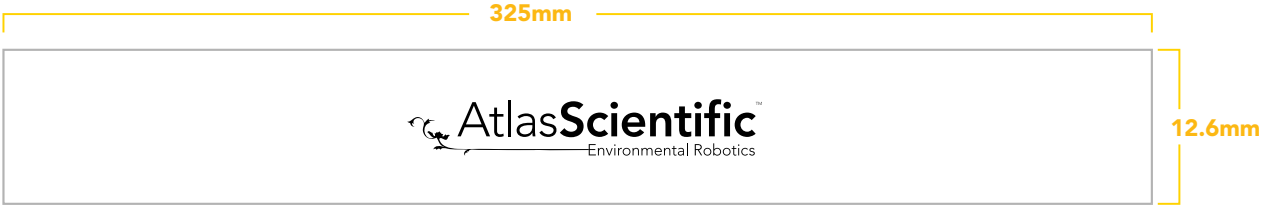
This cross section is an example of how the ground plane protects the Dissolved Oxygen signal. The ground plane should surround the Dissolved Oxygen signal, on the top layer as well as the bottom layer.

Recommended pad layout

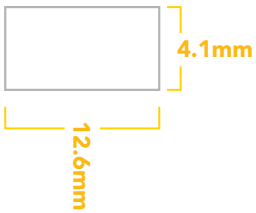


IC tube measurements

Top View



Side View



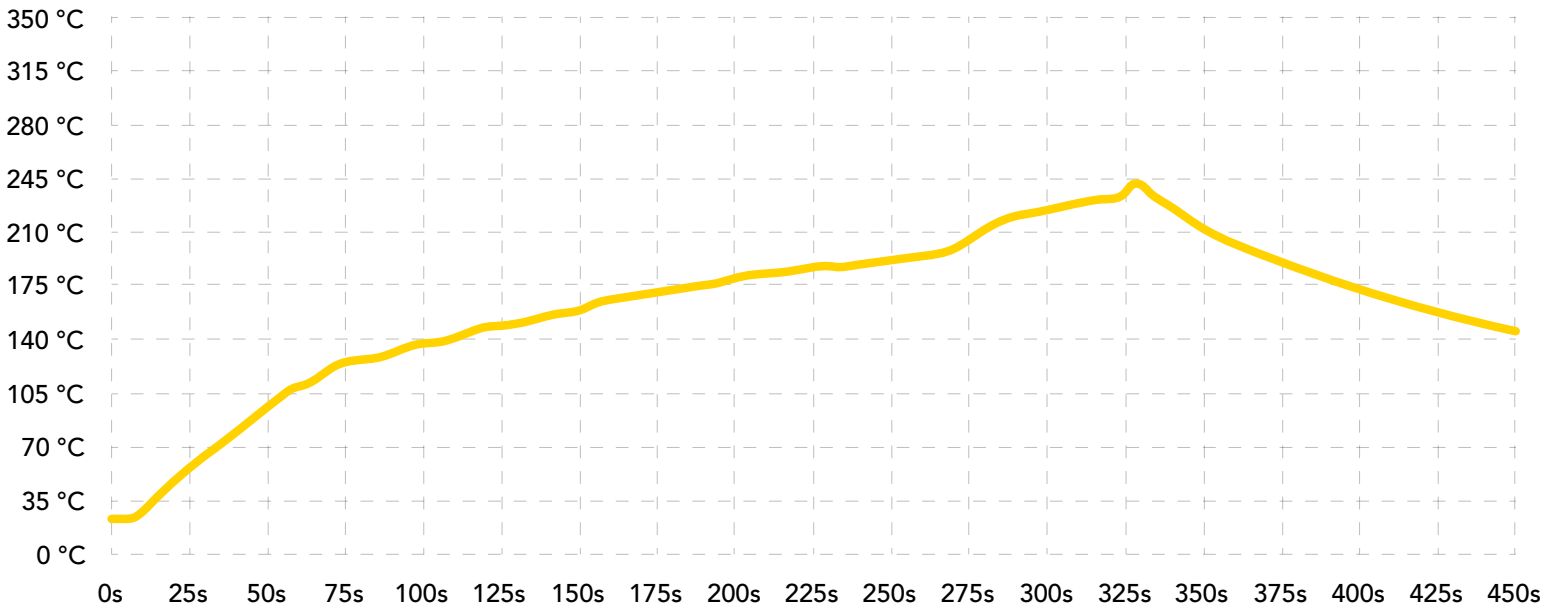
Fits 25 Dissolved Oxygen OEM™ circuits

	inside dimensions	outside dimensions
L	**325mm**	**325mm**
W	**11.6mm**	**12.6mm**
H	**3.1mm**	**4.1mm**

plastic thickness 0.5mm

AtlasScientific™ Environmental Robotics

Recommended reflow soldering profile



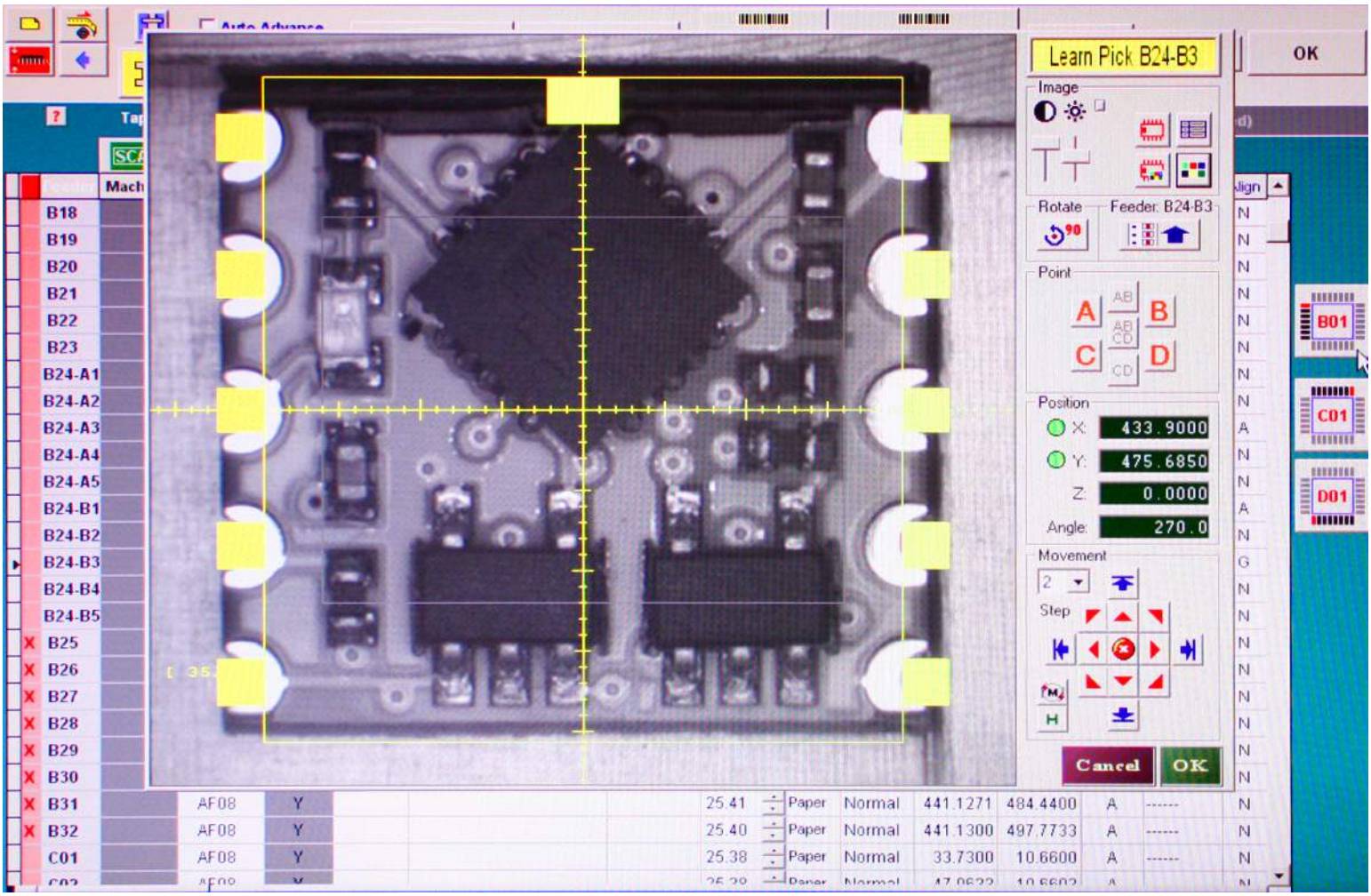
#	Temp	Sec
1	30	15
2	90	20
3	110	8
4	130	5
5	135	5
6	140	5
7	155	8
8	156	10
9	158	10
10	160	10

#	Temp	Sec
11	163	10
12	165	10
13	167	10
14	170	10
15	172	10
16	174	10
17	176	10
18	178	10
19	180	10
20	181	10

#	Temp	Sec
21	182	10
22	183	10
23	185	10
24	187	10
25	220	30
26	225	20
27	230	20
28	235	8
29	170	20
30	130	20

#	Temp	Sec
31	100	25
32	80	30
33	30	30
34	0	15

Pick and place usage



AtlasScientific™
Environmental Robotics

Datasheet change log

Datasheet V 3.4

Added new graphic on pg 3.

Datasheet V 3.3

Revised operating voltages on pages 1 & 5.

Datasheet V 3.2

Revised artwork on pg 8.

Datasheet V 3.1

Added “Designing you product” on pg 28.

Datasheet V 3.0

Changed 0x08 – calibration register from R/W to W.

Datasheet V 2.9

Added an attention page about soldered pins on page 3.

Datasheet V 2.8

Revised information about *desiging your own D.O. board* on pg. 28

Datasheet V 2.7

Revised isolation schematic on pg. 27

Datasheet V 2.6

Revised PCB cross section illustration on pg. 28

Datasheet V 2.5

Changed "Max rate" to "Response time" on cover page.

Datasheet V 2.4

Updated cover page information & firmware changes made to device.

Datasheet V 2.3

Updated firmware changes made to device.

Datasheet V 2.2

Corrected accessible registers graphic on page 7.

Datasheet V 2.1

Corrected max rate reading on cover page.

Datasheet V 2.0

Revised entire document

Firmware updates

V4.0 – Initial release (Mar 9, 2017)

V5.0 – (August 31, 2017)
• fixed a glitch where the calibration confirm byte wouldnt load on startup

V6.0 – (November 9, 2017)
• increased range for maximum reading

