**EZo-RGB™**

**Embedded Color Sensor**

**Reads**

RGB (24-bit)  
CIE (xyY)  
LUX (0 – 65535)

**Features**

- onboard LEDs  
- programmable color matching

**Connector**

5 lead data cable

**Response time**

1 reading per 400 milliseconds

**Sensing area**

15° half angle

**Cable length**

1 meter

**Water resistant/dust proof**

IP67

**Data protocol**

UART & I²C

**Default I²C address**

112 (0x70)

**Data format**

ASCII

**Operating voltage**

3.3V – 5V

This is an evolving document, check back for updates.
New Feature

The EZO-RGB™ Embedded Color Sensor is now IP67 waterproof – up to 1 meter

Strong Epoxy coating on lens

All EZO-RGB™ Embedded Color Sensors purchased after November 13th 2020, will be IP67 waterproof.
Caution

At full power the onboard LEDs are **VERY** bright. Do not look directly at the light without eye protection!

Minimum brightness = ~400 Lux
Maximum brightness = ~40,000 Lux at 5V (36,000 Lux at 3.3V)
Physical properties

Weight: 145g
Body: 316 Stainless Steel

Adjustable Wrench

Front
- x6 White LED
  - 40,000 Lux at 5V
  - 36,000 Lux at 3.3V
- Sensor

Back
- Indicator LED
  - used to show device status

Cable Length: 1m (3.2')

1/2" NPT
- 19.79mm (0.77"")
- 10.86mm (0.42"")
- 19.27mm (0.75"")

3/4" NPT
- 3mm (0.1"")

1 1/16"
27mm

Copyright © Atlas Scientific LLC
The sensor detects colored light in the red, green and blue spectrum. It is least sensitive to blue light and most sensitive to red light.
The spectrum output by the six onboard target LEDs is strongest in the blue spectrum and weakest in the red spectrum. This is the opposite of the color sensors sensitivity giving it the best possible color sensing performance.

**x6 White LED (5000K color temperature)**

The spectrum output by the six onboard target LEDs is strongest in the blue spectrum and weakest in the red spectrum. This is the opposite of the color sensors sensitivity giving it the best possible color sensing performance.

**Target LED brightness**
- Minimum ~400 Lux
- Maximum ~40,000 Lux

**Onboard LEDs output spectrum**

**120° angle of illumination**
### Power consumption

<table>
<thead>
<tr>
<th>LED</th>
<th>MAX</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON 100%</td>
<td>275 mA</td>
<td>0.40 mA</td>
</tr>
<tr>
<td>ON 1%</td>
<td>15 mA</td>
<td></td>
</tr>
<tr>
<td>OFF</td>
<td>13 mA</td>
<td></td>
</tr>
</tbody>
</table>

#### Absolute max ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage temperature</td>
<td>-65 °C</td>
<td>25 °C</td>
<td>125 °C</td>
</tr>
<tr>
<td>Operational temperature</td>
<td>-40 °C</td>
<td>85 °C</td>
<td></td>
</tr>
<tr>
<td>VCC</td>
<td>3.3V</td>
<td>5.5V</td>
<td></td>
</tr>
<tr>
<td>Pressure</td>
<td>1379kPa (200 PSI)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance testing

<table>
<thead>
<tr>
<th>Color Sample</th>
<th>Kodak™ Gray Card Plus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>2.5 cm</td>
</tr>
<tr>
<td>On-board LEDs</td>
<td>100% power</td>
</tr>
<tr>
<td>VCC</td>
<td>5V</td>
</tr>
</tbody>
</table>

The color readings were displayed using the free software on the Atlas Scientific™ website located [HERE](#).
Sensitivity

As the EZO-RGB™ color sensor is placed further away from the target object, its ability to detect color is diminished. At distances greater than 45cm most colors become varying shades of gray.
Calibration theory

The EZO-RGB™ color sensor is designed to be calibrated to a white object at the maximum brightness the object will be viewed under. In order to get the best results Atlas Scientific strongly recommends that the sensor is mounted into a fixed location. Holding the sensor in your hand during calibration will decrease performance.

1. Embed the EZO-RGB™ color sensor into its intended use location.
2. Set LED brightness to the desired level.
3. Place a white object in front of the target object and issue the calibration command “Cal”.
4. A single color reading will be taken and the device will be fully calibrated.

The RGB output has a three comma separated value, ranging from 0–255. However, It is possible to get RGB readings where one, or all of the values are greater than 255. This is because brightness is encoded in a RGB reading, if the subject being viewed is brighter than the calibrated brightness, the RGB values can go above 255. If this happens, the EZO-RGB™ Embedded Color Sensor needs to be re-calibrated for the correct brightness.
Data output

RGB
Default output

8-bit Red
8-bit Green
8-bit Blue

\{ \text{24 bits in total} \}

Color pallet
Output frequency
Output format

16,777,216 colors (24 Bit)
1 reading every 400ms
CSV string 24 bits

8-bit Red
8-bit Green
8-bit Blue

16,777,216 RGB colors

CIE 1931 color space

Human perception of color is not the same as a sensors perception of color. The CIE output is a representation of human color perception, while the RGB output is a representation of machine perception. While the two are close, they are not the same.

\[ xyY,0.373,0.463,414 \]

\[ xy = \text{coordinates} \]

\[ Y = \text{luminance} \]
Lux

Lux is a measure of light intensity as perceived by the human eye. The lux output has a comma separated identifier “Lux” followed by a single integer value from 0 – 65535. Lux readings will be effected by the sensors position.
Color matching

The EZO-RGB™ can indicate when a preset color is detected.

Place object of any color under the sensor.
Issue command “M,1”.
Color matching has been enabled.

The interrupt pin can only be used to signal a color match. It cannot be programmed to signal any other condition.

When a color match has been detected the reading will be appended with “*M” and the interrupt pin will change its state.

In order for color matching to work the EZO-RGB™ must be securely mounted and remain a fixed distance from its target.
Default state

UART mode

Baud: 9,600 Baud
Readings: continuous
Speed: 400 milliseconds
LED: on, when taking reading

Settings that are retained if power is cut:
- Automatic color matching
- Baud rate
- Calibration
- Continuous mode
- Device name
- Enable/disable parameters
- Enable/disable response codes
- LED control

Sleep mode: Settings that are NOT retained if power is cut

LED states:
- **Green**: Standby
- **Cyan**: Taking reading
- **Transmitting**

0% → 1%
Available data protocols

UART (default)

I²C

Unavailable data protocols

SPI
Analog
RS-485
Mod Bus
4–20mA
UART mode

<table>
<thead>
<tr>
<th>Settings that are retained if power is cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic color matching</td>
</tr>
<tr>
<td>Baud rate</td>
</tr>
<tr>
<td>Calibration</td>
</tr>
<tr>
<td>Continuous mode</td>
</tr>
<tr>
<td>Device name</td>
</tr>
<tr>
<td>Enable/disable parameters</td>
</tr>
<tr>
<td>Enable/disable response codes</td>
</tr>
<tr>
<td>LED control</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Settings that are <strong>NOT</strong> retained if power is cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep mode</td>
</tr>
</tbody>
</table>
UART mode

8 data bits  no parity
1 stop bit  no flow control

<table>
<thead>
<tr>
<th>Baud</th>
<th>300</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1,200</td>
</tr>
<tr>
<td></td>
<td>2,400</td>
</tr>
<tr>
<td>9,600</td>
<td>default</td>
</tr>
<tr>
<td>19,200</td>
<td></td>
</tr>
<tr>
<td>38,400</td>
<td></td>
</tr>
<tr>
<td>57,600</td>
<td></td>
</tr>
<tr>
<td>115,200</td>
<td></td>
</tr>
</tbody>
</table>

RX
Data in

TX
Data out

Vcc
3.3V – 5V

Data format

Units
RGB, LUX, & CIE

Encoding
ASCII

Format
string

Terminator
carriage return

Data type
integer & floating point

Decimal places
3

Smallest string
4 characters

Largest string
52 characters
Receiving data from device

2 parts

ASCII data string
Command

Carriage return <cr>
Terminator

Basic

9,600 baud (default)

Advanced

ASCII:

2 5 2 , 1 8 3 , 2 0 <cr>

Hex:

32 35 32 2C 31 38 33 2C 32 30 0D

Dec:

50 53 50 44 49 56 51 44 50 48 13
Sending commands to device

2 parts

Command (not case sensitive)  Carriage return <cr>

ASCII data string  Terminator

Advanced

ASCII: SIeee<cr>
Hex: 53 6C 65 65 70 0D
Dec: 83 108 101 101 112 13
Indicator LED definition

- **Green**: UART standby
- **Cyan**: Taking reading
- **Purple**: Changing I2C address
- **Red**: Command not understood
- **White**: Find

<table>
<thead>
<tr>
<th>Voltage</th>
<th>LED State</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>LED ON</td>
<td>+2.5 mA</td>
</tr>
<tr>
<td>3.3V</td>
<td>LED ON</td>
<td>+1 mA</td>
</tr>
</tbody>
</table>
## UART mode command quick reference

All commands are ASCII strings or single ASCII characters.

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
<th>Default state</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td>change baud rate</td>
<td>pg. 37</td>
<td>9,600</td>
</tr>
<tr>
<td>C</td>
<td>enable/disable continuous mode</td>
<td>pg. 26</td>
<td>enabled</td>
</tr>
<tr>
<td>Cal</td>
<td>performs calibration</td>
<td>pg. 28</td>
<td>n/a</td>
</tr>
<tr>
<td>Factory</td>
<td>enable factory reset</td>
<td>pg. 39</td>
<td>n/a</td>
</tr>
<tr>
<td>Find</td>
<td>finds device with blinking white LED</td>
<td>pg. 25</td>
<td>n/a</td>
</tr>
<tr>
<td>G</td>
<td>gamma correction</td>
<td>pg. 30</td>
<td>n/a</td>
</tr>
<tr>
<td>i</td>
<td>device information</td>
<td>pg. 33</td>
<td>n/a</td>
</tr>
<tr>
<td>iL</td>
<td>enable/disable indicator LED</td>
<td>pg. 24</td>
<td>enabled</td>
</tr>
<tr>
<td>I2C</td>
<td>change to I²C mode</td>
<td>pg. 40</td>
<td>not set</td>
</tr>
<tr>
<td>L</td>
<td>enable/disable target LED</td>
<td>pg. 23</td>
<td>enabled</td>
</tr>
<tr>
<td>M</td>
<td>automatic color matching</td>
<td>pg. 29</td>
<td>enabled</td>
</tr>
<tr>
<td>Name</td>
<td>set/show name of device</td>
<td>pg. 32</td>
<td>not set</td>
</tr>
<tr>
<td>O</td>
<td>enable/disable parameters</td>
<td>pg. 31</td>
<td>RGB</td>
</tr>
<tr>
<td>Plock</td>
<td>enable/disable protocol lock</td>
<td>pg. 38</td>
<td>n/a</td>
</tr>
<tr>
<td>R</td>
<td>returns a single reading</td>
<td>pg. 27</td>
<td>n/a</td>
</tr>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
<td>pg. 35</td>
<td>n/a</td>
</tr>
<tr>
<td>Status</td>
<td>retrieve status information</td>
<td>pg. 40</td>
<td>n/a</td>
</tr>
<tr>
<td>*OK</td>
<td>enable/disable response codes</td>
<td>pg. 34</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Target LED control

Command syntax

% represents the percentage of target LED brightness. (any number from 0–100)

- **L, %**  <cr>  set target LED brightness
- **L, %, T**  <cr>  set target LED brightness/trigger target LED only when a reading is taken (power saving)
- **L, ?**  <cr>  target LED state on/off?

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>L, 32  &lt;cr&gt;</td>
<td>*OK  &lt;cr&gt;  target LED set to 32% brightness.</td>
</tr>
<tr>
<td>L, 14, T  &lt;cr&gt;</td>
<td>*OK  &lt;cr&gt;  target LED set to 14% brightness, and will only turn on when a reading is taken.</td>
</tr>
</tbody>
</table>
| L, ?  <cr> | ?L, %, [T]  <cr>  *

<table>
<thead>
<tr>
<th>L, 0  &lt;cr&gt;</th>
<th>L, 32  &lt;cr&gt;</th>
<th>L, 100  &lt;cr&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>32%</td>
<td>100%</td>
</tr>
</tbody>
</table>
## Indicator LED control

### Command syntax

- `iL,1 <cr>`  indicator LED on
- `iL,0 <cr>`  Indicator LED off
- `iL,? <cr>`  Indicator LED state on/off?

### Example

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iL,1 &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>iL,0 &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>iL,? &lt;cr&gt;</code></td>
<td>?iL,1 &lt;cr&gt; or ?iL,0 &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

### Illustrations

- **iL,1**: Indicator LED on (Green)
- **iL,0**: Indicator LED off (Red)

---

Copyright © Atlas Scientific LLC
## Find

### Command syntax

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

### Example

<table>
<thead>
<tr>
<th>Find  &lt;cr&gt;</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td>*OK</td>
</tr>
</tbody>
</table>

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

**Command syntax**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>C,1</td>
<td>enable continuous readings once per 400ms</td>
<td>default</td>
</tr>
<tr>
<td>C,n</td>
<td>continuous readings every n x 400ms (n = 2 to 99)</td>
<td></td>
</tr>
<tr>
<td>C,0</td>
<td>disable continuous readings</td>
<td></td>
</tr>
<tr>
<td>C,?</td>
<td>continuous reading mode on/off?</td>
<td></td>
</tr>
</tbody>
</table>

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>C,1</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>R,G,B (400ms) &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>R,G,B (800ms) &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>R,G,B (1200ms) &lt;cr&gt;</td>
</tr>
<tr>
<td>C,30</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>R,G,B (12,000ms) &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>R,G,B (24,000ms) &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>R,G,B (36,000ms) &lt;cr&gt;</td>
</tr>
<tr>
<td>C,0</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>C,?</td>
<td>?C,1 &lt;cr&gt; or ?C,0 &lt;cr&gt; or ?C,30 &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>
Single reading mode

Command syntax

R <cr>  takes single reading

Example

<table>
<thead>
<tr>
<th>R &lt;cr&gt;</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R,G,B &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

Response

Green
Standby

Cyan
Taking reading

Transmitting

400ms
Calibration

Command syntax

Cal  <cr>  calibrates the EZO-RGB™

1. place white object (such as a piece of paper) in front of target
2. Issue “cal” command

Example

<table>
<thead>
<tr>
<th></th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cal</td>
<td>*OK</td>
</tr>
</tbody>
</table>

**Example:**

1. place white object (such as a piece of paper) in front of target
2. Issue “cal” command

**Response:**

*OK

---

**Uncalibrated:**

- Coordinates: 90, 172, 4

**Calibrated:**

- Coordinates: 140, 197, 64

---

Copyright © Atlas Scientific LLC
# Automatic color matching

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M,1 &lt;cr&gt;</td>
<td>Enables automatic color matching</td>
</tr>
<tr>
<td>M,0 &lt;cr&gt;</td>
<td>Disables automatic color matching</td>
</tr>
<tr>
<td>M,? &lt;cr&gt;</td>
<td>Color matching on/off?</td>
</tr>
</tbody>
</table>

## Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>M,1 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>M,0 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>M,? &lt;cr&gt;</td>
<td>?M,0 &lt;cr&gt; or ?M,1 &lt;cr&gt; *OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

![Diagram of automatic color matching](image.png)
**Gamma correction**

**Command syntax**

G,n <cr> set gamma correction

where n = a floating point number from 0.01 – 4.99

G,? <cr> gamma correction value?

The default gamma correction is 1.00 which represents no correction at all. A gamma correction factor is a floating point number from 0.01 to 4.99.

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>G,1.99</td>
<td>*OK</td>
</tr>
<tr>
<td>G,?</td>
<td>?G,1.99 *OK</td>
</tr>
</tbody>
</table>
# Enable/disable parameters from output string

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O, [parameter],[1,0]</td>
<td>enable or disable output parameter</td>
</tr>
<tr>
<td>O,?</td>
<td>enabled parameter?</td>
</tr>
</tbody>
</table>

## Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>O,RGB,1</td>
<td>*OK &lt;cr&gt; enable / disable RGB</td>
</tr>
<tr>
<td>O,LUX,1</td>
<td>*OK &lt;cr&gt; enable / disable lux</td>
</tr>
<tr>
<td>O,CIE,1</td>
<td>*OK &lt;cr&gt; enable / disable CIE</td>
</tr>
<tr>
<td>O,?</td>
<td>?,O,RGB,LUX,CIE &lt;cr&gt; if all enabled</td>
</tr>
</tbody>
</table>

## Parameters

- **RGB**: red, green, blue
- **LUX**: illuminance
- **CIE**: CIE 1931 color space

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set name</td>
<td>Name,n&lt;cr&gt;</td>
<td>Sets the device name to 'n'.</td>
</tr>
<tr>
<td>Show name</td>
<td>Name,&lt;cr&gt;</td>
<td>Displays the current device name.</td>
</tr>
</tbody>
</table>

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name,zzt&lt;cr&gt;</td>
<td>*OK&lt;cr&gt;</td>
</tr>
<tr>
<td>Name,&lt;cr&gt;</td>
<td>?Name,zzt&lt;cr&gt;</td>
</tr>
<tr>
<td>Name,&lt;cr&gt;</td>
<td>*OK&lt;cr&gt;</td>
</tr>
</tbody>
</table>

Do not use spaces in the name

Up to 16 ASCII characters

---

## Response

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name,zzt&lt;cr&gt;</td>
<td>*OK&lt;cr&gt;</td>
</tr>
<tr>
<td>Name,&lt;cr&gt;</td>
<td>?Name,zzt&lt;cr&gt;</td>
</tr>
<tr>
<td>Name,&lt;cr&gt;</td>
<td>*OK&lt;cr&gt;</td>
</tr>
</tbody>
</table>

---

*OK<cr>
Device information

Command syntax

i <cr> device information

Example

i <cr>

Response

?i,RGB,2.1 <cr>
*OK <cr>

Response breakdown

?i, RGB, 2.1

Device Firmware
## Response codes

### Command syntax

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

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R <code>&lt;cr&gt;</code></td>
<td>140,197,64 <code>&lt;cr&gt;</code> *OK <code>&lt;cr&gt;</code></td>
</tr>
<tr>
<td>*OK,0 <code>&lt;cr&gt;</code></td>
<td>no response, *OK disabled</td>
</tr>
<tr>
<td>R <code>&lt;cr&gt;</code></td>
<td>140,197,64 <code>&lt;cr&gt;</code> *OK disabled</td>
</tr>
<tr>
<td>*OK,? <code>&lt;cr&gt;</code></td>
<td>?*OK,1 <code>&lt;cr&gt;</code>  or  ?*OK,0 <code>&lt;cr&gt;</code></td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Status &lt;cr&gt;</th>
<th>voltage at Vcc pin and reason for last restart</th>
</tr>
</thead>
</table>

## Example

<table>
<thead>
<tr>
<th>Status &lt;cr&gt;</th>
<th>?Status,P,5.038 &lt;cr&gt;</th>
<th>*OK &lt;cr&gt;</th>
</tr>
</thead>
</table>

## Response breakdown

<table>
<thead>
<tr>
<th>?Status,  P, 5.038</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason for restart</td>
</tr>
</tbody>
</table>

## Restart codes

- **P**: powered off
- **S**: software reset
- **B**: brown out
- **W**: watchdog
- **U**: unknown
Sleep mode/low power

**Command syntax**

Sleep <cr>  enter sleep mode/low power

**Example**

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>*SL &lt;cr&gt;</td>
</tr>
</tbody>
</table>

**Any command**

<table>
<thead>
<tr>
<th>5V</th>
<th>MAX</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>175</td>
<td>0.40</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3.3V</th>
<th>MAX</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>138</td>
<td>0.18</td>
<td></td>
</tr>
</tbody>
</table>

Send any character or command to awaken device.

Example Response wakes up device
Change baud rate

Command syntax

Baud,n  <cr>  change baud rate

Example

| Baud,38400  <cr> | *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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plock,1 &lt;cr&gt;</td>
<td>enable Plock</td>
</tr>
<tr>
<td>Plock,0 &lt;cr&gt;</td>
<td>disable Plock (default)</td>
</tr>
<tr>
<td>Plock,? &lt;cr&gt;</td>
<td>Plock on/off?</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plock,1 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>Plock,0 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>Plock,? &lt;cr&gt;</td>
<td>?Plock,1 &lt;cr&gt; or ?Plock,0 &lt;cr&gt;</td>
</tr>
</tbody>
</table>

### Example

- **Plock,1**
  - *OK <cr>

- **I2C,100**
  - cannot change to I\(^2\)C
  - *ER <cr>

- **cannot change to I\(^2\)C**
Factory reset

Command syntax

Factory <cr> enable factory reset

Example

<table>
<thead>
<tr>
<th>Factory  &lt;cr&gt;</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>*OK &lt;cr&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Factory <cr>

(reboot)

*OK <cr>  *RS <cr>  *RE <cr>

Baud rate will not change

Clears calibration
Reset target LED brightness to 1%
Reset output to RGB
"*OK" enabled
# Change to I²C mode

## Command syntax

### Default I²C address 112 (0x70)

I²C, \(n < \text{cr}> \) sets I²C address and reboots into I²C mode

\(n = \) any number 1 – 127

## Example

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>I²C,100</td>
<td>*OK (reboot in I²C mode)</td>
</tr>
</tbody>
</table>

## Wrong example

<table>
<thead>
<tr>
<th>Wrong example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>I²C,139</td>
<td>(n \neq 127) *ER (&lt;\text{cr}&gt;)</td>
</tr>
</tbody>
</table>

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 INT
- 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 112 (0x70)

Example

Wrong Example

Disconnected RX line
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**
- Automatic color matching
- 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
I\(^2\)C mode

**I\(^2\)C address**  (0x01 – 0x7F)

112 (0x70) default

**Vcc**  3.3V – 5.5V

**Clock speed**  100 – 400 kHz

**SDA**

**SCL**

VCC

0V

---

Data format

**Units**  RGB, LUX, & CIE

**Encoding**  ASCII

**Format**  string

**Terminator**  carriage return

**Data type**  integer & floating point

**Decimal places**  3

**Smallest string**  4 characters

**Largest string**  52 characters
Sending commands to device

5 parts

- **Start**
- **I²C address**
- **Write**
- **Command (not case sensitive)**
- **Stop**

**ASCII command string**

Example

- **Start**
- **I²C address 112 (0x70)**
- **Write**
- **Sleep**
- **Stop**

Command

**Advanced**

- **SDA**
- **SCL**

Address bits

- **A6**
- **A5**
- **A4**
- **A3**
- **A2**
- **A1**
- **A0**

The entire command as ASCII with all arguments

- **W**
- **ACK**
- **First letter of command**
- **ACK**
- **Last letter of command**
- **ACK**

**W** = low

Stop
Requesting data from device

7 parts
- Start
- I²C address
- Read
- Response code
- Data string
- Null
- Stop

112 (0x70)

Response code: "413"

Terminator (Dec 0)

Advanced

252,183,20

SDA → SDA

SCL → SCL

Address bits

Data

N bytes of data

All bytes after data are Null

R = High

Start

ACK

Response code

ACK

Data

ACK

Data

ACK

Null

ACK

Null

Null

NACK

Stop

SDA

SCL

Start

ACK

Response code

ACK

Data

ACK

Data

ACK

Null

ACK

Null

Null

NACK

Stop

SDA

SCL

50 53 50 44 49 56 51 44 50 48

= 252,183,20

Dec

50 53 50 44 49 56 51 44 50 48

Dec

ASCII

Copyright © Atlas Scientific LLC
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);  
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
Indicator LED control

- **Blue**: I²C standby
- **Green**: Taking reading
- **Purple**: Changing I²C address
- **Red**: Command not understood
- **White**: Find

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>+2.5 mA</td>
</tr>
<tr>
<td>3.3V</td>
<td>+1 mA</td>
</tr>
</tbody>
</table>
# I²C mode command quick reference

All commands are ASCII strings or single ASCII characters.

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td>switch back to UART mode</td>
<td>63</td>
</tr>
<tr>
<td>Cal</td>
<td>performs custom calibration</td>
<td>53</td>
</tr>
<tr>
<td>Factory</td>
<td>enable factory reset</td>
<td>62</td>
</tr>
<tr>
<td>Find</td>
<td>finds device with blinking white LED</td>
<td>51</td>
</tr>
<tr>
<td>G</td>
<td>gamma correction</td>
<td>54</td>
</tr>
<tr>
<td>i</td>
<td>device information</td>
<td>57</td>
</tr>
<tr>
<td>iL</td>
<td>enable/disable indicator LED</td>
<td>50</td>
</tr>
<tr>
<td>I2C</td>
<td>change I²C address</td>
<td>61</td>
</tr>
<tr>
<td>L</td>
<td>enable/disable target LED</td>
<td>49</td>
</tr>
<tr>
<td>Name</td>
<td>set/show name of device</td>
<td>56</td>
</tr>
<tr>
<td>O</td>
<td>enable/disable parameters</td>
<td>55</td>
</tr>
<tr>
<td>Plock</td>
<td>enable/disable protocol lock</td>
<td>60</td>
</tr>
<tr>
<td>R</td>
<td>returns a single reading</td>
<td>52</td>
</tr>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
<td>59</td>
</tr>
<tr>
<td>Status</td>
<td>retrieve status information</td>
<td>58</td>
</tr>
</tbody>
</table>
**Target LED control**

**Command syntax**

- L,%: set target LED brightness
- L,%,T: set target LED brightness/trigger target LED only when a reading is taken (**power saving**)
- L,?: target LED state on/off?

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>L,32</td>
<td><img src="#" alt="Diagram" /></td>
</tr>
<tr>
<td>L,14,T</td>
<td><img src="#" alt="Diagram" /></td>
</tr>
<tr>
<td>L,?</td>
<td><img src="#" alt="Diagram" /></td>
</tr>
</tbody>
</table>

**Response**

- L,32
  - **Wait 300ms**
  - Dec: 0
  - ASCII: L,32
  - Null
  - Target LED set to 32% brightness.

- L,14,T
  - **Wait 300ms**
  - Dec: 0
  - ASCII: L,14,T
  - Null
  - Target LED set to 14% brightness, and will only turn on when a reading is taken.

- L,?
  - **Wait 300ms**
  - Dec: 1
  - ASCII: ?L, %, [ T ]
  - Null: 0

% represents the percentage of target LED brightness (any number from 0–100)
# Indicator LED control

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>iL,1</td>
<td>Indicator LED on</td>
<td>default</td>
</tr>
<tr>
<td>iL,0</td>
<td>Indicator LED off</td>
<td></td>
</tr>
<tr>
<td>iL,?</td>
<td>Indicator LED state on/off?</td>
<td></td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>iL,1</td>
<td><img src="image" alt="Wait 300ms" /></td>
</tr>
<tr>
<td>iL,0</td>
<td><img src="image" alt="Wait 300ms" /></td>
</tr>
<tr>
<td>iL,?</td>
<td><img src="image" alt="Wait 300ms" /></td>
</tr>
</tbody>
</table>
## Find

### Command syntax

**Find**  |  LED rapidly blinks white, used to help find device

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find</td>
<td></td>
</tr>
</tbody>
</table>

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

**300ms processing delay**
Taking reading

Command syntax

R  return 1 reading

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>1 R,G,B 0</td>
</tr>
</tbody>
</table>

300ms processing delay

Example:

- Green: Taking reading
- Wait 300ms
- Taking reading
- Transmitting
- Cyan: Standby
Calibration

Command syntax

Cal calibrates the EZO-RGB™

1. place white object (such as a piece of paper) in front of target
2. Issue “cal” command

300ms processing delay

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cal</td>
<td><img src="cal.png" alt="" /></td>
</tr>
</tbody>
</table>
Gamma correction

Command syntax

G,n  set gamma correction
    where n = a floating point number from 0.01 – 4.99

G,?  gamma correction value?

The default gamma correction is 1.00 which represents no correction at all. A gamma correction factor is a floating point number from 0.01 to 4.99.

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>G,1.99</td>
<td>1 0 Wait 300ms Dec Null</td>
</tr>
<tr>
<td>G,?</td>
<td>1 ?G,1.99 0 Wait 300ms Dec ASCII Null</td>
</tr>
</tbody>
</table>
Enable/disable parameters from output string

Command syntax

O, [parameter],[1,0]
O,?

enable or disable output parameter
enabled parameter?

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>O,RGB,1 / O,RGB,0</td>
<td><img src="image" alt="Wait 300ms" /> 1 0 enable / disable RGB</td>
</tr>
<tr>
<td>O,LUX,1 / O,LUX,0</td>
<td><img src="image" alt="Wait 300ms" /> 1 0 enable / disable lux</td>
</tr>
<tr>
<td>O,CIE,1 / O,CIE,0</td>
<td><img src="image" alt="Wait 300ms" /> 1 0 enable / disable CIE</td>
</tr>
</tbody>
</table>

O,?

Wait 300ms

1 ? O,RGB,LUX,CIE 0 if all enabled

Example

Parameterters

| RGB | red, green, blue |
| LUX | illuminance |
| CIE | CIE 1931 color space |

Followed by 1 or 0

1 enabled
0 disabled

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

Copyright © Atlas Scientific LLC
Naming device

Command syntax

<table>
<thead>
<tr>
<th>Name,n</th>
<th>set name</th>
<th>n =</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name,?</th>
<th>show name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Do not use spaces in the name

Up to 16 ASCII characters

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name,zzt</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>Name,?</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

300ms processing delay

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

Wait 300ms

Dec Null

?Name,zzt ASCII Null

Copyright © Atlas Scientific LLC
## Device information

### Command syntax

<table>
<thead>
<tr>
<th>Command Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>i, RGB, 2.1</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>i</th>
<th>Device Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wait 300ms</td>
</tr>
</tbody>
</table>

### Response

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>?i, RGB, 2.1</td>
</tr>
</tbody>
</table>

### Response breakdown

<table>
<thead>
<tr>
<th>Device</th>
<th>Firmware</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>2.1</td>
</tr>
</tbody>
</table>
## Command syntax

| Status | voltage at Vcc pin and reason for last restart |

## Example

<table>
<thead>
<tr>
<th>Status</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>?Status,</td>
<td>P, 5.038</td>
</tr>
</tbody>
</table>

## Response breakdown

```
?Status, P, 5.038
```

<table>
<thead>
<tr>
<th>Reason for restart</th>
<th>Voltage at Vcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>5.038</td>
</tr>
</tbody>
</table>

## 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.

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>no response</td>
</tr>
<tr>
<td>Any command</td>
<td>wakes up device</td>
</tr>
</tbody>
</table>

**Example**

**Response**

- **5V**
  - STANDBY: 45 mA
  - SLEEP: 3.4 mA

- **3.3V**
  - STANDBY: 42 mA
  - SLEEP: 3.0 mA

Do not read status byte after issuing sleep command.
## Protocol lock

### Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plock, 1</td>
<td>enable Plock</td>
</tr>
<tr>
<td>Plock, 0</td>
<td>disable Plock</td>
</tr>
<tr>
<td>Plock, ?</td>
<td>Plock on/off?</td>
</tr>
</tbody>
</table>

### 300ms processing delay

- Locks device to I²C mode.

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plock, 1</strong></td>
<td><img src="image" alt="Wait 300ms" /> Dec 1 Null 0</td>
</tr>
<tr>
<td><strong>Plock, 0</strong></td>
<td><img src="image" alt="Wait 300ms" /> Dec 1 Null 0</td>
</tr>
<tr>
<td><strong>Plock, ?</strong></td>
<td><img src="image" alt="Wait 300ms" /> Dec 1 ASCII ?Plock,1 Null 0</td>
</tr>
</tbody>
</table>

### Example

- **Plock, 1**: Cannot change to UART
- **Baud, 9600**: Cannot change to UART
I²C address change

Command syntax

I²C,n  sets I²C address and reboots into I²C mode

Example | Response
---|---
I²C,101 | device reboot

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 112 (0x70).

n = any number 1 – 127
Factory reset

Command syntax

Factory enable factory reset

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

I²C address will not change

Example

<table>
<thead>
<tr>
<th>Factory</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory</td>
<td>device reboot</td>
</tr>
</tbody>
</table>

Clears custom calibration
LED on
Response codes enabled

Factory (reboot)
## Change to UART mode

### Command syntax

<table>
<thead>
<tr>
<th>Baud, n</th>
<th>switch from I²C to UART</th>
</tr>
</thead>
</table>

### Example

| Baud, 9600 | reboot in UART mode |

### n =

300
1200
2400
9600
19200
38400
57600
115200

---

**Example**

- **Baud,9600**
  - **Response**: reboot in UART mode

---

**Diagram**

1. **Baud,9600**
2. Changing to UART mode
3. (reboot)
Manual switching to UART

- Disconnect ground (power off)
- Disconnect TX and RX
- Connect TX to INT
- 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 2.7
Removed proximity sensing capabilities from device.

Datasheet V 2.6
Added new feature info on pg 2.

Datasheet V 2.5
Corrected typo on pg 54.

Datasheet V 2.4
Moved Default state to pg 18.

Datasheet V 2.3
Changed the default I2C Address to 112 (0x70)

Datasheet V 2.2
Added an I²C section to the datasheet.

Datasheet V 2.1
Revised response for the sleep command in UART mode on pg 39.

Datasheet V 2.0
Revised entire datasheet
## Firmware updates

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.10</td>
<td>(November 7, 2015)</td>
<td>• Fixed sleep mode bug.</td>
</tr>
<tr>
<td>V1.15</td>
<td>(November 30, 2015)</td>
<td>• Fixed threshold bug.</td>
</tr>
<tr>
<td>V1.16</td>
<td>(February 2, 2016)</td>
<td>• Fixed bug where excessive newline characters would be output for every line.</td>
</tr>
<tr>
<td>v1.18</td>
<td>(Sept 19, 2016)</td>
<td>• Updated manufacturing process.</td>
</tr>
<tr>
<td>v1.20</td>
<td>(June 29, 2017)</td>
<td>• Issuing the I²C command will return with an error.</td>
</tr>
<tr>
<td>v2.00</td>
<td>(May 1, 2019)</td>
<td>• Added the RGB indicator LED and I²C mode, find command, C,n command</td>
</tr>
<tr>
<td>v2.10</td>
<td>(August 23, 2021)</td>
<td>• Proximity sensing capabilities removed <em>(feature was hardly ever used)</em></td>
</tr>
</tbody>
</table>
Warranty

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

The debugging phase

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

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