EZO-RGB™
Embedded Color Sensor

Reads
- RGB (24-bit)
- CIE (xyY)
- LUX (0 – 65535)
- Proximity (2 – 36cm)

Features
- programmable color matching
- proximity triggering
- onboard LEDs

Connector
- 5 lead data cable

Response time
- 400ms

Sensing area
- 15° half angle

Cable length
- 1 meter

Water resistant/dust proof
- IP65

Data protocol
- UART & I²C

Default I²C address
- 106 (0x6A)

Data format
- ASCII

Operating voltage
- 3.3V – 5V

This is an evolving document, check back for updates.
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)
Table of contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical properties</td>
<td>4</td>
</tr>
<tr>
<td>Sensor properties</td>
<td>5</td>
</tr>
<tr>
<td>Target LED properties</td>
<td>6</td>
</tr>
<tr>
<td>IR LED properties</td>
<td>7</td>
</tr>
<tr>
<td>Pin out</td>
<td>8</td>
</tr>
<tr>
<td>Performance testing</td>
<td>9</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>10</td>
</tr>
<tr>
<td>Calibration theory</td>
<td>11</td>
</tr>
<tr>
<td>Data output</td>
<td>12</td>
</tr>
<tr>
<td>CIE 1931 color space</td>
<td>12</td>
</tr>
<tr>
<td>Proximity sensing</td>
<td>13</td>
</tr>
<tr>
<td>IR LED brightness control</td>
<td>14</td>
</tr>
<tr>
<td>Lux</td>
<td>14</td>
</tr>
<tr>
<td>Proximity triggering</td>
<td>15</td>
</tr>
<tr>
<td>Color matching</td>
<td>16</td>
</tr>
<tr>
<td>Proximity and color matching</td>
<td>17</td>
</tr>
<tr>
<td>Available data protocol</td>
<td>18</td>
</tr>
</tbody>
</table>

**UART**

- Default state                              19
- UART mode                                  20
- Receiving data from device                 21
- Sending commands to device                 22
- LED color definition                       23
- **UART quick command page**                24
- Target LED control                         25
- Indicator LED control                      26
- Find                                       27
- Continuous mode                            28
- Single reading mode                        29
- Calibration                                30
- Proximity detection                        31
- Automatic color matching                   32
- Gamma correction                           33
- Enable/disable parameters                  34
- Naming device                              35
- Device information                         36
- Response codes                             37
- Reading device status                      38
- Sleep mode/low power                       39
- Change baud rate                           40
- Protocol lock                              41
- Factory reset                              42
- Change to I²C mode                         43
- Manual switching to UART                   44

**I²C**

- I²C mode                                   46
- Sending commands                           47
- Requesting data                            48
- Response codes                             49
- Processing delay                           49
- LED color definition                       50
- **I²C quick command page**                 51
- Target LED control                         52
- Indicator LED control                      53
- Find                                       54
- Taking reading                             55
- Calibration                                56
- Gamma correction                           57
- Enable/disable parameters                  58
- Device information                         59
- Reading device status                      60
- Sleep mode/low power                       61
- Protocol lock                              62
- I²C address change                         63
- Factory reset                              64
- Change to UART mode                        65
- Manual switching to UART                   66

- Datasheet change log                       67
- Firmware updates                           67
- Warranty                                   68
Physical properties

Cable Length
1m (3.2’)

50.12mm (1.97”)
3mm (0.1”)
10.86mm (0.42”)
19.79mm (0.77”)

Weight 145g

Front

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

Sensor

x3 IR LED
used for proximity

Back

Indicator LED
used to show device status

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.
Target LED properties

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.

Onboard LEDs output spectrum

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

120° angle of illumination
IR LED properties

The three IR LEDs use reflected infrared radiation to detect proximity.

Proximity sensing range ~2cm – 36cm

x3 IR LED
Wavelength = 850nm

Too close

Too far away
Pin out

Data and power cable pinout

<table>
<thead>
<tr>
<th>LED</th>
<th>RX/SCL</th>
<th>TX/SDA</th>
<th>GND</th>
<th>VCC</th>
<th>INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td>–</td>
<td>–</td>
<td>White</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Green</td>
<td>–</td>
<td>–</td>
<td>Green</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Black</td>
<td>–</td>
<td>–</td>
<td>Black</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Red</td>
<td>–</td>
<td>–</td>
<td>Red</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Blue</td>
<td>–</td>
<td>–</td>
<td>Blue</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

The interrupt pin will change its state when a color match has been detected.

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

See page 32 to enable automatic color matching in UART mode.

Power consumption

<table>
<thead>
<tr>
<th>Voltage</th>
<th>LED</th>
<th>MAX</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>ON 100%</td>
<td>275 mA</td>
<td>0.40 mA</td>
</tr>
<tr>
<td></td>
<td>ON 1%</td>
<td>15 mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>13 mA</td>
<td></td>
</tr>
<tr>
<td>3.3V</td>
<td>ON 100%</td>
<td>100 mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ON 1%</td>
<td>15 mA</td>
<td>0.14 mA</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>12 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>125 °C</td>
<td></td>
</tr>
<tr>
<td>Operational temperature</td>
<td>-40 °C</td>
<td>25 °C</td>
<td>85 °C</td>
</tr>
<tr>
<td>VCC</td>
<td>3.3V</td>
<td>3.3V</td>
<td>5.5V</td>
</tr>
<tr>
<td>Pressure</td>
<td>1379kPa (200 PSI)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance testing

Color Sample: Kodak™ Gray Card Plus
Distance: 2.5cm
On-board LEDs: 100% power
VCC: 5V

The color readings were displayed using the free software on the Atlas Scientific™ website located HERE.

Kodak™ Gray Card Plus

Color output from the EZO-RGB™
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**

8-bit color graphics

**Default output**

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

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

**Color pallet**

16,777,216 colors (24 Bit)

**Output frequency**

1 reading every 400ms

**Output format**

CSV string 24 bits

![Color palette diagram]

**CIE 1931 color space**

Human perception of color is not the same as a sensor's 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.

**Identifier**

\[ \text{x} \quad \text{y} \quad \text{Y} \]

xyY,0.373,0.463,414

xy = coordinates
Y = luminance

![CIE 1931 color space diagram]
Proximity sensing

The EZO-RGB™ uses three IR emitters to detect its proximity to another object. The intensity of the reflected IR light is used to determine if an object is in front of it. Because the IR reflectivity of materials is not uniform, the EZO-RGB™ proximity sensing capabilities should not be used as a precise distance measuring device.

The proximity output has a comma separated identifier “P” followed by a single integer value from ~250–1023. When the proximity sensor detects nothing the readings will be ~250–350.
Controlling the IR LED brightness is necessary because, not all objects have the same IR reflectivity. Some objects can have an IR reflectivity that is too intense, therefore it is necessary to lower the brightness of the IR LEDs to achieve repeatable IR proximity detection.

By lowering the IR LED brightness, IR proximity detection is now possible.

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.
Proximity triggering

The EZO-RGB™ takes a color reading only when a set proximity is met or exceeded.

Place object under the sensor.
Issue the command “P,1”.
Proximity triggering has been enabled.

Once proximity triggering has been enabled, no readings will be transmitted until an object of equal, or greater height has been detected under the EZO-RGB™.

Color readings that are taken when a proximity match has been detected will be appended with “*P”.

In order for proximity triggering to work the EZO-RGB™ must be securely mounted and remain a fixed distance from its target.
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.

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

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

Both proximity and color matching functions can be enabled simultaneously, permitting the engineer to quickly develop an object sorter with minimal coding.

In order for proximity triggering and color matching to work the EZO-RGB™ must be securely mounted and remain a fixed distance from its target.
Available data protocols

UART
(default)

I²C

Unavailable data protocols

SPI
Analog
RS-485
Mod Bus
4–20mA
Default state

UART mode

Baud

Readings

Speed

LED

9,600

continuous

400 milliseconds

on, when taking reading

Settings that are retained if power is cut

- Baud rate
- Calibration
- Continuous mode
- Device name
- Enable/disable parameters
- Enable/disable response codes
- LED control

Settings that are NOT retained if power is cut

- Sleep mode
## UART mode

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>8 data bits</strong></td>
<td><strong>no parity</strong></td>
</tr>
<tr>
<td><strong>1 stop bit</strong></td>
<td><strong>no flow control</strong></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th><strong>RX</strong></th>
<th>Data in</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>TX</strong></th>
<th>Data out</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Vcc</strong></th>
<th>3.3V – 5V</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>CPU</strong></th>
</tr>
</thead>
</table>

## Data format

<table>
<thead>
<tr>
<th><strong>Units</strong></th>
<th>RGB, LUX, CIE, and proximity</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Encoding</strong></th>
<th>ASCII</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Format</strong></th>
<th>string</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Terminator</strong></th>
<th>carriage return</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Data type</strong></th>
<th>integer &amp; floating point</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Decimal places</strong></th>
<th>3</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Smallest string</strong></th>
<th>4 characters</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Largest string</strong></th>
<th>52 characters</th>
</tr>
</thead>
</table>
Receiving data from device

2 parts

ASCII data string

Command

Carriage return <cr>

Terminator

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)
ASCII data string

Carriage return <cr>
Terminator

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

- **Green**
  - UART standby

- **Cyan**
  - Taking reading

- **Purple**
  - Changing I²C address

- **Red**
  - Command not understood

- **White**
  - Find

**LED ON**

- 5V: +2.5 mA
- 3.3V: +1 mA
# UART mode

## Command quick reference

All commands are ASCII strings or single ASCII characters.

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
<th>Default state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td>change baud rate</td>
<td>pg. 40 9,600</td>
</tr>
<tr>
<td>C</td>
<td>enable/disable continuous mode</td>
<td>pg. 28 enabled</td>
</tr>
<tr>
<td>Cal</td>
<td>performs calibration</td>
<td>pg. 30 n/a</td>
</tr>
<tr>
<td>Factory</td>
<td>enable factory reset</td>
<td>pg. 42 n/a</td>
</tr>
<tr>
<td>Find</td>
<td>finds device with blinking white LED</td>
<td>pg. 27 n/a</td>
</tr>
<tr>
<td>G</td>
<td>gamma correction</td>
<td>pg. 33 n/a</td>
</tr>
<tr>
<td>i</td>
<td>device information</td>
<td>pg. 36 n/a</td>
</tr>
<tr>
<td>iL</td>
<td>enable/disable indicator LED</td>
<td>pg. 26 enabled</td>
</tr>
<tr>
<td>I2C</td>
<td>change to I²C mode</td>
<td>pg. 43 not set</td>
</tr>
<tr>
<td>L</td>
<td>enable/disable target LED</td>
<td>pg. 25 enabled</td>
</tr>
<tr>
<td>M</td>
<td>automatic color matching</td>
<td>pg. 32 enabled</td>
</tr>
<tr>
<td>Name</td>
<td>set/show name of device</td>
<td>pg. 35 not set</td>
</tr>
<tr>
<td>O</td>
<td>enable/disable parameters</td>
<td>pg. 34 RGB</td>
</tr>
<tr>
<td>P</td>
<td>proximity detection</td>
<td>pg. 31 n/a</td>
</tr>
<tr>
<td>Plock</td>
<td>enable/disable protocol lock</td>
<td>pg. 41 n/a</td>
</tr>
<tr>
<td>R</td>
<td>returns a single reading</td>
<td>pg. 29 n/a</td>
</tr>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
<td>pg. 39 n/a</td>
</tr>
<tr>
<td>Status</td>
<td>Retrieve status information</td>
<td>pg. 38 n/a</td>
</tr>
<tr>
<td>*OK</td>
<td>enable/disable response codes</td>
<td>pg. 37 n/a</td>
</tr>
</tbody>
</table>
# Target LED control

## Command syntax

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L,% &lt;cr&gt;</td>
<td>set target LED brightness</td>
</tr>
<tr>
<td>L,%,%T &lt;cr&gt;</td>
<td>set target LED brightness/trigger target LED only when a reading is taken (power saving)</td>
</tr>
<tr>
<td>L,? &lt;cr&gt;</td>
<td>target LED state on/off?</td>
</tr>
</tbody>
</table>

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

## Example Response

<table>
<thead>
<tr>
<th>Example</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>
<tr>
<td>L,? &lt;cr&gt;</td>
<td>?L, %, [T] &lt;cr&gt; *OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

## Example

- **L,0 <cr>**
  - 0%
- **L,32 <cr>**
  - 32%
- **L,100 <cr>**
  - 100%
# Indicator LED control

**Command syntax**

- `iL,1 <cr>` indicator LED on  
  default
- `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>
</tbody>
</table>
| `iL,? <cr>` | `?iL,1 <cr>` or `?iL,0 <cr>`  
  *OK <cr> |

---

![Indicator LED images](image-url)
## Find

### Command syntax

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

---

### Example

<table>
<thead>
<tr>
<th>Find &lt;cr&gt;</th>
</tr>
</thead>
</table>

### Response

<table>
<thead>
<tr>
<th>*OK &lt;cr&gt;</th>
</tr>
</thead>
</table>

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

### Command syntax

- **C,1**  
  enable continuous readings once per 400ms  
  **default**

- **C,n**  
  continuous readings every n x 400ms (n = 2 to 99)

- **C,0**  
  disable continuous readings

- **C,?**  
  continuous reading mode on/off?

### Example

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

### Command syntax

R <cr>  takes single reading

### Example | Response
--- | ---
R <cr> | R,G,B <cr>  
 | *OK <cr>

- **Green**: Standby
- **Cyan**: Taking reading
- **Transmitting**: 400ms

Copyright © Atlas Scientific LLC
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 Response

| Cal <cr> | *OK <cr> |

Cal <cr>

uncalibrated
90, 172, 4

calibrated
140, 197, 64

Copyright © Atlas Scientific LLC
## Proximity detection

### Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>P,[1,0]</code></td>
<td>enable / disable</td>
</tr>
<tr>
<td><code>P,n</code></td>
<td>manually enable proximity detection at <code>n</code> distance</td>
</tr>
<tr>
<td><code>P,[H, M, L]</code></td>
<td>set IR LEDs brightness to high, medium or low</td>
</tr>
<tr>
<td><code>P,?</code></td>
<td>proximity state on/off?</td>
</tr>
</tbody>
</table>

**Example**

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>P,1 &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>P,800 &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>P,L &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>P,? &lt;cr&gt;</code></td>
<td>?P,0,L &lt;cr&gt;</td>
</tr>
</tbody>
</table>
### Automatic color matching

#### Command syntax

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

#### Example Response

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>M,1</td>
<td>*OK</td>
</tr>
<tr>
<td>M,0</td>
<td>*OK</td>
</tr>
<tr>
<td>M,?</td>
<td>?M,1 or ?M,0</td>
</tr>
</tbody>
</table>

---

![Diagram of a conveyor belt with colored objects and a sensor detecting color differences.](image-url)
# Gamma correction

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>G,n</strong>&lt;br&gt;set gamma correction</td>
<td>where n = a floating point number from 0.01 – 4.99</td>
</tr>
<tr>
<td><strong>G,?</strong>&lt;br&gt;gamma correction value?</td>
<td></td>
</tr>
</tbody>
</table>

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>G,1.99 &lt;cr&gt;</th>
<th>*OK &lt;cr&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>G,? &lt;cr&gt;</td>
<td>?G,1.99 &lt;cr&gt;&lt;br&gt;*OK &lt;cr&gt;</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><code>O, [parameter],[1,0]</code></td>
<td>enable or disable output parameter</td>
</tr>
<tr>
<td><code>O,?</code></td>
<td>enabled parameter?</td>
</tr>
</tbody>
</table>

## Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>O,RGB,1 / O,RGB,0</code></td>
<td>*OK &lt;cr&gt; enable / disable RGB</td>
</tr>
<tr>
<td><code>O,PROX,1 / O,PROX,0</code></td>
<td>*OK &lt;cr&gt; enable / disable proximity</td>
</tr>
<tr>
<td><code>O,LUX,1 / O,LUX,0</code></td>
<td>*OK &lt;cr&gt; enable / disable lux</td>
</tr>
<tr>
<td><code>O,CIE,1 / O,CIE,0</code></td>
<td>*OK &lt;cr&gt; enable / disable CIE</td>
</tr>
<tr>
<td><code>O,?</code></td>
<td>?,O,RGB,PROX,LUX,CIE &lt;cr&gt; if all enabled</td>
</tr>
</tbody>
</table>

## Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>red, green, blue</td>
</tr>
<tr>
<td>PROX</td>
<td>proximity</td>
</tr>
<tr>
<td>LUX</td>
<td>illuminance</td>
</tr>
<tr>
<td>CIE</td>
<td>CIE 1931 color space</td>
</tr>
</tbody>
</table>

## Followed by 1 or 0

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>enabled</td>
</tr>
<tr>
<td>0</td>
<td>disabled</td>
</tr>
</tbody>
</table>

* 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>Command</th>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name,n</td>
<td><code>&lt;cr&gt;</code></td>
<td>set name</td>
</tr>
<tr>
<td>Name,?</td>
<td><code>&lt;cr&gt;</code></td>
<td>show name</td>
</tr>
</tbody>
</table>

### Up to 16 ASCII characters

### Example

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

### Response Examples

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

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

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

- **Name,? <cr>**
  - Name,zzt <cr> *OK <cr>
### Command syntax

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

### Example

<table>
<thead>
<tr>
<th>i  &lt;cr&gt;</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>?i,RGB,1.3 &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

### Response breakdown

<table>
<thead>
<tr>
<th>?i, RGB, 1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑ Device</td>
</tr>
<tr>
<td>↑ Firmware</td>
</tr>
</tbody>
</table>
# Response codes

## Command syntax

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

## Example

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

## Other response codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*ER</td>
<td>unknown command</td>
</tr>
<tr>
<td>*OV</td>
<td>over volt (VCC&gt;=5.5V)</td>
</tr>
<tr>
<td>*UV</td>
<td>under volt (VCC&lt;=3.1V)</td>
</tr>
<tr>
<td>*RS</td>
<td>reset</td>
</tr>
<tr>
<td>*RE</td>
<td>boot up complete, ready</td>
</tr>
<tr>
<td>*SL</td>
<td>entering sleep mode</td>
</tr>
<tr>
<td>*WA</td>
<td>wake up</td>
</tr>
</tbody>
</table>

These response codes cannot be disabled
## Reading device status

### Command syntax

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

### Example

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

### Response breakdown

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

### Restart codes

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

---

38  Copyright © Atlas Scientific LLC
## Sleep mode/low power

### Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
</tr>
</tbody>
</table>

### Example

- **Sleep**
  - Send any character or command to awaken device.

### Response

- **Sleep**
  - *OK*
  - *SL*

### Any command

- *WA*
  - wakes up device

### MAX SLEEP

<table>
<thead>
<tr>
<th>Voltage</th>
<th>MAX mA</th>
<th>SLEEP mA</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>175</td>
<td>0.40</td>
</tr>
<tr>
<td>3.3V</td>
<td>138</td>
<td>0.18</td>
</tr>
</tbody>
</table>

---

Copyright © Atlas Scientific LLC
## Change baud rate

### Command syntax

- **Baud,n**  \(<\text{cr}>\)  change baud rate

### Example Response

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud,38400 (&lt;\text{cr}&gt;)</td>
<td>*OK (&lt;\text{cr}&gt;)</td>
</tr>
<tr>
<td>Baud,? (&lt;\text{cr}&gt;)</td>
<td>?Baud,38400 (&lt;\text{cr}&gt;) *OK (&lt;\text{cr}&gt;)</td>
</tr>
</tbody>
</table>

### Baud Rate Values

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

---

**Standby**

Baud,38400 \(<\text{cr}>\)

**Changing baud rate**

*OK \(<\text{cr}>\)

(reboot)

**Standby**
# Protocol lock

## Command syntax

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

## Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Plock,1 &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>Plock,0 &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>Plock,? &lt;cr&gt;</code></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 I²C
  - *ER <cr>*
- `cannot change to I²C`
# Factory reset

## Command syntax

Factory `<cr>` enable factory reset

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

* (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

---

Copyright © Atlas Scientific LLC
# Change to I²C mode

## Command syntax

- **I²C, n** <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>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>I²C,139</td>
<td>n ≠ 127  *ER &lt;cr&gt;</td>
</tr>
</tbody>
</table>

## Diagram

- **I²C,100**
  
  - **Green** *OK* <cr>
  
  - **Blue** now in I²C mode

**Default I²C address 106 (0x6A)**

---

43 Copyright © Atlas Scientific LLC
Manual switching to I²C

- Make sure Plock is set to 0
- 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 106 (0x6A)

Example

Wrong Example

Disconnect 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:
- 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^2C mode

I^2C address  \((0x01 \text{ – } 0x7F)\)

106 (0x6A) default

Vcc  \(3.3V \text{ – } 5.5V\)

Clock speed  \(100 \text{ – } 400 \text{ kHz}\)

SDA

SCL

Data format

Units  RGB, LUX, CIE, and proximity

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

Example

```
Start  106 (0x6A)  Write  Sleep  Stop
```

Advanced

```
Start  106 (0x6A)  Write  Command  Stop
```

The entire command as ASCII with all arguments:

```
SDA ← SDA
```

```
SCL ← SCL
```

**Copyright © Atlas Scientific LLC**
Requesting data from device

7 parts

Start  \(\text{I}^2\text{C address}\)  Read  Response code  Data string  Null  Stop

106 (0x6A)  1 byte "413"  Terminator (Dec 0)

252,183,20

Advanced

Requesting data from device

Advanced

CPU

SDA – SDA

SCL – SCL

Address bits

N bytes of data

All bytes after data are Null

RI – High

Start

Data

Data

Data

Null

Null

NACK

Stop

1  50  53  50  44  49  56  51  44  50  48  0 = 252,183,20

Dec

ASCII

Dec
Response codes & processing delay

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

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

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>65</td>
</tr>
<tr>
<td>Cal</td>
<td>performs custom calibration</td>
<td>56</td>
</tr>
<tr>
<td>Factory</td>
<td>enable factory reset</td>
<td>64</td>
</tr>
<tr>
<td>Find</td>
<td>finds device with blinking white LED</td>
<td>54</td>
</tr>
<tr>
<td>G</td>
<td>gamma correction</td>
<td>57</td>
</tr>
<tr>
<td>i</td>
<td>device information</td>
<td>59</td>
</tr>
<tr>
<td>iL</td>
<td>enable/disable indicator LED</td>
<td>53</td>
</tr>
<tr>
<td>I2C</td>
<td>change I²C address</td>
<td>63</td>
</tr>
<tr>
<td>L</td>
<td>enable/disable target LED</td>
<td>52</td>
</tr>
<tr>
<td>O</td>
<td>enable/disable parameters</td>
<td>58</td>
</tr>
<tr>
<td>Plock</td>
<td>enable/disable protocol lock</td>
<td>62</td>
</tr>
<tr>
<td>R</td>
<td>returns a single reading</td>
<td>55</td>
</tr>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
<td>61</td>
</tr>
<tr>
<td>Status</td>
<td>retrieve status information</td>
<td>60</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><strong>L,32</strong></td>
<td><img src="image" alt="Wait 300ms" /> <img src="image" alt="Dec 0 Null" /> target LED set to 32% brightness.</td>
</tr>
<tr>
<td><strong>L,14,T</strong></td>
<td><img src="image" alt="Wait 300ms" /> <img src="image" alt="Dec 0 Null" /> target LED set to 14% brightness, and will only turn on when a reading is taken.</td>
</tr>
<tr>
<td><strong>L,?</strong></td>
<td><img src="image" alt="Wait 300ms" /> <img src="image" alt="Dec ASCII Null" /></td>
</tr>
</tbody>
</table>
# Indicator LED control

## Command syntax

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

### Default Command Syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>ASCII</th>
<th>Decimal</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>iL,1</td>
<td>?Li,1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>iL,0</td>
<td>?Li,0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### Example Response

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

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

- **iL,?**
  - Wait 300ms
  - 1 Dec ASCII Null
  - or
  - Wait 300ms
  - 1 Dec ASCII Null

---

53 Copyright © Atlas Scientific LLC
Find

Command syntax

Find  LED rapidly blinks white, used to help find device

300ms processing delay

Example

Response

Find

Wait 300ms  1  Dec  0  Null
Taking reading

Command syntax

R  return 1 reading

300ms processing delay

Example  Response

R

Wait 300ms

1  Dec  R,G,B  0  ASCII  Null

Green
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

## Example Response

<table>
<thead>
<tr>
<th>Cal</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cal</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
</tbody>
</table>

### Example

- Place white object in front of target.
- Issue “cal” command.

### Response

- **Uncalibrated**:
  - 90, 172, 4
- **Calibrated**:
  - 140, 197, 64

300ms processing delay

*Wait 300ms*

1 Dec

0 Null
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>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>G,1.99</td>
<td><img src="https://via.placeholder.com/150" alt="Wait 300ms" /> 1 0 Dec Null</td>
</tr>
<tr>
<td>G,?</td>
<td><img src="https://via.placeholder.com/150" alt="Wait 300ms" /> 1 ?G,1.99 0 Dec ASCII Null</td>
</tr>
</tbody>
</table>
Enable/disable parameters from output string

**Command syntax**

- O, [parameter],[1,0]  
  enable or disable output parameter
- O,?  
  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</td>
</tr>
<tr>
<td>O,LUX,1 / O,LUX,0</td>
<td><img src="image" alt="Wait 300ms" /> 1 0</td>
</tr>
<tr>
<td>O,CIE,1 / O,CIE,0</td>
<td><img src="image" alt="Wait 300ms" /> 1 0</td>
</tr>
</tbody>
</table>

- O,?
  - ![Wait 300ms](image) 1 ?,0,RGB,LUX,CIE 0
  - if all enabled

**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”.

---

58  
Copyright © Atlas Scientific LLC
# Device Information

## Command Syntax

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>i</code></td>
<td><img src="image" alt="Wait 300ms" /> 1 Dec <code>?i,RGB,1.3</code> 0 Null</td>
</tr>
</tbody>
</table>

## Response Breakdown

| `?i, RGB, 1.3` |
| -------------- |----------------|
| Device         | Firmware       |
# Reading device status

## Command syntax

<table>
<thead>
<tr>
<th>Status</th>
<th>voltage at Vcc pin and reason for last restart</th>
</tr>
</thead>
</table>

### 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

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

<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>

Send any character or command to awaken device.

Example Response

- Sleep
  - 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.

Sleep Standby

Copyright © Atlas Scientific LLC
# 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>Plock,1</td>
<td>Wait 300ms 1 Dec 0 Null</td>
</tr>
<tr>
<td>Plock,0</td>
<td>Wait 300ms 1 Dec 0 Null</td>
</tr>
<tr>
<td>Plock,?</td>
<td>Wait 300ms 1 Dec ASCII 0 Null</td>
</tr>
</tbody>
</table>

**Example Response**

- Plock,1 Baud, 9600
- cannot change to UART

- Plock,1
- 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 106 (0x6A).

n = any number 1 - 127

I²C,101 (reboot)
# Factory reset

## Command syntax

Factory enable factory reset

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

<table>
<thead>
<tr>
<th>Factory</th>
<th>enable factory reset</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Factory" /></td>
<td><img src="image2" alt="Factory enable factory reset" /></td>
</tr>
</tbody>
</table>

### Example Response

- **Factory**
- **Response**: device reboot

Clears custom calibration
LED on
Response codes enabled

- **Factory**: ![Factory](image3)
- **(reboot)**: ![reboot](image4)
Change to UART mode

**Command syntax**

Baud,\(n\) switch from \(I^2C\) to UART

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud,9600</td>
<td>reboot in UART mode</td>
</tr>
</tbody>
</table>

\[ n = \begin{cases} 
300 \\
1200 \\
2400 \\
9600 \\
19200 \\
38400 \\
57600 \\
115200 
\end{cases} \]
Manual switching to UART

- Make sure Plock is set to 0
- 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
# Datasheet change log

## 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>V1.2</td>
<td>(June 29, 2017)</td>
<td>• Issuing the I²C command will return with an error.</td>
</tr>
<tr>
<td>v2.0</td>
<td>(May 1, 2019)</td>
<td>• Added the RGB indicator LED and I²C mode, find command, C,n command</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.