EZO-RTD™
Embedded Temperature Circuit

- Reads
- Range: -126.000 °C – 1254 °C
- Resolution: 0.001
- Accuracy: +/- (0.1 + 0.0017 x °C)
- Response time: 1 reading per sec
- Supported probes: Any type & brand PT-100 or PT-1000 RTD
- Calibration: Single point
- Temperature output: °C, °K, or °F
- Data protocol: UART & I²C
- Default I²C address: 102 (0x66)
- Operating voltage: 3.3V – 5.5V
- Data format: ASCII
- Onboard Data Logger: 50 Readings

Electrical Isolation not needed
This is sensitive electronic equipment. Get this device working in a solderless breadboard first. Once this device has been soldered it is no longer covered by our warranty.

This device has been designed to be soldered and can be soldered at any time. Once that decision has been made, Atlas Scientific no longer assumes responsibility for the device’s continued operation. The embedded systems engineer is now the responsible party.

STOP

SOLDERING THIS DEVICE VOIDS YOUR WARRANTY.

Get this device working in a solderless breadboard first!

Do not embed this device without testing it in a solderless breadboard!
Table of contents

Circuit dimensions 4
Power consumption 4
Absolute max ratings 4
Temperature circuit range 5
Temperature circuit accuracy 5
Atlas Scientific PT-1000 probe 6
Using other brand PT-100/PT-1000 7
Operating principle 9
Correct wiring 11
Calibration theory 12
On board data logger 13
Default state 14
Available data protocols 15

UART

UART mode 17
Receiving data from device 18
Sending commands to device 19
LED color definition 20
UART quick command page 21
LED control 22
Find 23
Continuous reading mode 24
Single reading mode 25
Calibration 26
Export calibration 27
Import calibration 28
Temperature scale 29
Enable/disable data logger 30
Memory recall 31
Memory clear 32
Naming device 33
Device information 34
Response codes 35
Reading device status 36
Sleep mode/low power 37
Change baud rate 38
Protocol lock 39
Factory reset 40
Change to I²C mode 41
Manual switching to I²C 42

I²C

I²C mode 44
Sending commands 45
Requesting data 46
Response codes 47
LED color definition 48
I²C quick command page 49
LED control 50
Find 51
Taking reading 52
Calibration 53
Export calibration 54
Import calibration 55
Temperature scale 56
Enable/disable data logger 57
Memory recall 58
Memory clear 59
Naming device 60
Device information 61
Reading device status 62
Sleep mode/low power 63
Protocol lock 64
I²C address change 65
Factory reset 66
Change to UART mode 67
Manual switching to UART 68

Circuit footprint 69
Datasheet change log 70
Warranty 72
Power consumption

<table>
<thead>
<tr>
<th></th>
<th>5V</th>
<th>3.3V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LED</td>
<td>MAX</td>
</tr>
<tr>
<td>ON</td>
<td>16 mA</td>
<td>15.4 mA</td>
</tr>
<tr>
<td>OFF</td>
<td>15.3 mA</td>
<td>15 mA</td>
</tr>
<tr>
<td>ON</td>
<td>14.3 mA</td>
<td>13.8 mA</td>
</tr>
<tr>
<td>OFF</td>
<td>14 mA</td>
<td>13.6 mA</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 (EZO™ RTD)</td>
<td>-65 °C</td>
<td>125 °C</td>
<td></td>
</tr>
<tr>
<td>Operational temperature (EZO™ RTD)</td>
<td>-40 °C</td>
<td>25 °C</td>
<td>85 °C</td>
</tr>
</tbody>
</table>
EZO™ RTD temperature sensing range

EZO™ RTD temperature sensing accuracy
To read temperatures above, or below the max cable temperature, an additional probe housing (thermowell) is needed to protect the cable.

The Atlas Scientific EZO™ RTD Temperature circuit only works with PT-100 and PT-1000 probes.
Using other brand PT-100/PT-1000

The EZO™ RTD Temperature circuit will auto-detect if the connected probe is PT-100 or PT-1000.

<table>
<thead>
<tr>
<th>Probe class</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>+/- (0.1 + 0.0017 × °C)</td>
</tr>
<tr>
<td>A</td>
<td>+/- (0.15 + 0.002 × °C)</td>
</tr>
<tr>
<td>B</td>
<td>+/- (0.3 + 0.005 × °C)</td>
</tr>
<tr>
<td>C</td>
<td>+/- (0.6 + 0.01 × °C)</td>
</tr>
</tbody>
</table>

It makes no difference which lead of the temperature probe is connected to the two probe pins.

BOTH ARE CORRECT
Any off the shelf PT-100 or PT-1000 temperature probe can be used with the Atlas Scientific EZO™ RTD Temperature circuit. The PT-100 or PT-1000 temperature probe can be a 2, 3 or 4 wire probe.
Operating principle

The Atlas Scientific EZO™ RTD Temperature circuit is a small footprint computer system that is specifically designed to be used in robotic applications where the embedded systems engineer requires accurate and precise measurements of temperature through a generic PT-100/PT-1000 temperature probe.

RTD = Resistance Temperature Detector
PT = Platinum
PT-100 = 100 Ω at 0°C
PT-1000 = 1k Ω at 0°C

Unlike any other material, platinums correlation between resistance and temperature seems to be woven into the fabric of the universe. It is for this reason, that the platinum RTD temperature sensor is the industrial standard for temperature measurement.
Power and data isolation

ELECTRICAL ISOLATION IS \textit{NOT} NEEDED.
Correct wiring

Incorrect wiring

Extended leads Sloppy setup Perfboards or Protoboards *Embedded into your device

Part # COM-104

Electrically Isolated EZO™ Carrier Board

Part # ISCCB

USB carrier board

Part # G2-USB-ISO

Part # ISCCB-2

Copyright © Atlas Scientific LLC
Calibration theory

The most important part of calibration is watching the readings during the calibration process. It’s easiest to calibrate the device in its default state (UART mode, continuous readings). Switching the device to I²C mode after calibration will not affect the stored calibration. If the device must be calibrated in I²C mode be sure to request readings continuously so you can see the output from the probe.

Calibration can be done at any value, a simple method is to calibrate the probe in boiling water.

100 °C

Atlas Scientific recommends calibration be done every three years.

Elevation and boiling point table

<table>
<thead>
<tr>
<th>Elevation in meters</th>
<th>Boiling point</th>
</tr>
</thead>
<tbody>
<tr>
<td>305</td>
<td>98.9 °C</td>
</tr>
<tr>
<td>229</td>
<td>99.2 °C</td>
</tr>
<tr>
<td>152</td>
<td>99.5 °C</td>
</tr>
<tr>
<td>76</td>
<td>99.7 °C</td>
</tr>
<tr>
<td>0</td>
<td>100 °C</td>
</tr>
<tr>
<td>-76</td>
<td>100.3 °C</td>
</tr>
<tr>
<td>-152</td>
<td>100.5 °C</td>
</tr>
</tbody>
</table>

Use purified/distilled water

For accurate calibration using different temperature values, you must use a tool called a "dry block calibrator."
On board data logger

- 50 readings
- Programmable storage interval

Minimum – 10 seconds
Maximum – 320,000 seconds

Temperature readings that are stored to the data logger will be retained even if the power is cut.

When memory becomes full, the data logger will overwrite memory location 1.
Default state

UART mode

<table>
<thead>
<tr>
<th>Feature</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td>9,600</td>
</tr>
<tr>
<td>Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>Readings</td>
<td>continuous</td>
</tr>
<tr>
<td>Speed</td>
<td>1 reading per second</td>
</tr>
<tr>
<td>With probe</td>
<td>ttt.ttt</td>
</tr>
<tr>
<td>Without probe</td>
<td>-1023.000</td>
</tr>
<tr>
<td>LED</td>
<td>on</td>
</tr>
</tbody>
</table>

UART mode

1,000 ms

- **Green**: Standby
- **Cyan**: Taking reading
- **Transmitting**
Available data protocols

UART

I²C

Unavailable data protocols

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

**Settings that are retained if power is cut**
- Baud rate
- Calibration
- Continuous mode
- Temperature scale
- Device name
- Enable/disable response codes
- Hardware switch to I2C mode
- LED control
- Protocol lock
- Software switch to I2C mode

**Settings that are NOT retained if power is cut**
- Find
- Sleep mode
UART mode

8 data bits  no parity
1 stop bit  no flow control

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

**RX**
Data in

**TX**
Data out

**Vcc**
3.3V – 5.5V

CPU

---

Data format

**Reading** temperature
**Units** °C, °K, or °F
**Encoding** ASCII
**Format** string
**Terminator** carriage return

**Data type** floating point
**Decimal places** 3
**Smallest string** 4 characters
**Largest string** 40 characters
Receiving data from device

2 parts

ASCII data string
Command

Carriage return <cr>
Terminator

Advanced

ASCII: 2 5 . 1 0 4 <cr>
Hex: 32 35 2E 31 30 34 0D
Dec: 50 53 46 49 48 52 13

9,600 baud (default)

25.104 <cr>

CPU

Receiver
Sending commands to device

2 parts

Command (not case sensitive)
ASCII data string

Carriage return <cr>
Terminator

Advanced

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

Green
UART standby

Cyan
Taking reading

Purple
Changing baud rate

Red
Command not understood

White
Find

<table>
<thead>
<tr>
<th>Voltage</th>
<th>LED ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>+0.4 mA</td>
</tr>
<tr>
<td>3.3V</td>
<td>+0.2 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>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td>change baud rate</td>
<td>pg. 38 9,600</td>
</tr>
<tr>
<td>C</td>
<td>enable/disable continuous reading</td>
<td>pg. 24 enabled</td>
</tr>
<tr>
<td>Cal</td>
<td>performs calibration</td>
<td>pg. 26 n/a</td>
</tr>
<tr>
<td>D</td>
<td>enable/disable data logger</td>
<td>pg. 30 disabled</td>
</tr>
<tr>
<td>Export</td>
<td>export calibration</td>
<td>pg. 27 n/a</td>
</tr>
<tr>
<td>Factory</td>
<td>enable factory reset</td>
<td>pg. 40 n/a</td>
</tr>
<tr>
<td>Find</td>
<td>finds device with blinking white LED</td>
<td>pg. 23 n/a</td>
</tr>
<tr>
<td>i</td>
<td>device information</td>
<td>pg. 34 n/a</td>
</tr>
<tr>
<td>I2C</td>
<td>change to I²C mode</td>
<td>pg. 41 not set</td>
</tr>
<tr>
<td>Import</td>
<td>import calibration</td>
<td>pg. 28 n/a</td>
</tr>
<tr>
<td>L</td>
<td>enable/disable LED</td>
<td>pg. 22 enabled</td>
</tr>
<tr>
<td>M</td>
<td>memory recall/clear</td>
<td>pg. 31 n/a</td>
</tr>
<tr>
<td>Name</td>
<td>set/show name of device</td>
<td>pg. 33 not set</td>
</tr>
<tr>
<td>Plock</td>
<td>enable/disable protocol lock</td>
<td>pg. 39 disabled</td>
</tr>
<tr>
<td>R</td>
<td>returns a single reading</td>
<td>pg. 25 n/a</td>
</tr>
<tr>
<td>S</td>
<td>temperature scale (°C, °K, °F)</td>
<td>pg. 29 celsius</td>
</tr>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
<td>pg. 37 n/a</td>
</tr>
<tr>
<td>Status</td>
<td>retrieve status information</td>
<td>pg. 36 n/a</td>
</tr>
<tr>
<td>*OK</td>
<td>enable/disable response codes</td>
<td>pg. 35 enable</td>
</tr>
</tbody>
</table>
# LED control

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L,1</td>
<td>LED on (default)</td>
</tr>
<tr>
<td>L,0</td>
<td>LED off</td>
</tr>
<tr>
<td>L,?</td>
<td>LED state on/off?</td>
</tr>
</tbody>
</table>

## Example

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

---

**Example:**

L,1 <cr>

*OK <cr>

L,0 <cr>

*OK <cr>

L,? <cr>

?L,1 <cr> or ?L,0 <cr>

*OK <cr>
Find

Command syntax

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

Example

Find <cr>

Response

*OK <cr>
# Continuous reading mode

## Command syntax

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

## Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C,1</strong></td>
<td>*OK&lt;br&gt;°C (1 sec)&lt;br&gt;°C (2 sec)&lt;br&gt;°C (n sec)</td>
</tr>
<tr>
<td><strong>C,30</strong></td>
<td>*OK&lt;br&gt;°C (30 sec)&lt;br&gt;°C (60 sec)&lt;br&gt;°C (90 sec)</td>
</tr>
<tr>
<td><strong>C,0</strong></td>
<td>*OK</td>
</tr>
<tr>
<td><strong>C,?</strong></td>
<td>?C,1 or ?C,0 or ?C,30&lt;br&gt;*OK</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></td>
<td>25.104 &lt;cr&gt; *OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

Green
Standby

Cyan
Taking reading

Transmitting

600 ms
The EZO™ RTD circuit uses single point calibration.

**Command syntax**

- `Cal,t <cr>`  
  t = any temperature
- `Cal,clear <cr>`  
  delete calibration data
- `Cal,? <cr>`  
  device calibrated?

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Cal,100.00 &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>Cal,clear &lt;cr&gt;</code></td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td><code>Cal,? &lt;cr&gt;</code></td>
<td>?Cal,1 &lt;cr&gt; or ?Cal,0 &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

**Example**

- `Cal,100.00 <cr>`
- `100.35 °C`
- `100.00 °C`
- `Cal,100.00 <cr>`

---

Copyright © Atlas Scientific LLC
# Export calibration

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export,? &lt;cr&gt;</td>
<td>calibration string info</td>
</tr>
<tr>
<td>Export &lt;cr&gt;</td>
<td>export calibration string from calibrated device</td>
</tr>
</tbody>
</table>

## Example

**Export,? <cr>**

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,120 &lt;cr&gt;</td>
</tr>
</tbody>
</table>

**Response breakdown**

10, 120

- # of strings to export
- # of bytes to export

Export strings can be up to 12 characters long, and is always followed by <cr>.

**Export <cr>**

<table>
<thead>
<tr>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>59 6F 75 20 61 72 &lt;cr&gt; (1 of 10)</td>
</tr>
<tr>
<td>65 20 61 20 63 6F &lt;cr&gt; (2 of 10)</td>
</tr>
<tr>
<td>(7 more)</td>
</tr>
<tr>
<td>6F 6C 20 67 75 79 &lt;cr&gt; (10 of 10)</td>
</tr>
</tbody>
</table>

**Export <cr>**

*DONE

Disabling *OK simplifies this process.

---

**Example diagram:**

- MCU to TX to RX
- MCU sends [10,120] bytes
- *DONE

---

*Copyright © Atlas Scientific LLC*
Import calibration

Command syntax

Import, n <cr>  import calibration string to new device

Example

Import, 59 6F 75 20 61 72 <cr>  (1 of 10)
Import, 65 20 61 20 63 6F <cr>  (2 of 10)
Import, 6F 6C 20 67 75 79 <cr>  (10 of 10)

Response

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

If one of the imported strings is not correctly entered, the device will not accept the import, respond with *ER and reboot.
# Temperature scale (°C, °K, °F)

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Syntax</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>S,c</td>
<td><code>&lt;cr&gt;</code> celsius</td>
<td>default</td>
</tr>
<tr>
<td>S,k</td>
<td><code>&lt;cr&gt;</code> kelvin</td>
<td></td>
</tr>
<tr>
<td>S,f</td>
<td><code>&lt;cr&gt;</code> fahrenheit</td>
<td></td>
</tr>
<tr>
<td>S,?</td>
<td><code>&lt;cr&gt;</code> temperature scale?</td>
<td></td>
</tr>
</tbody>
</table>

## Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>S,c</td>
<td>*OK <code>&lt;cr&gt;</code></td>
</tr>
<tr>
<td>S,k</td>
<td>*OK <code>&lt;cr&gt;</code></td>
</tr>
<tr>
<td>S,f</td>
<td>*OK <code>&lt;cr&gt;</code></td>
</tr>
<tr>
<td>S,?</td>
<td><code>?S,c </code>&lt;cr&gt;<code>or</code>?S,k <code>&lt;cr&gt;</code> or <code>?S,f </code>&lt;cr&gt;<code>*OK</code>&lt;cr&gt;`</td>
</tr>
</tbody>
</table>

## Diagrams

- Celsius: -126 °C to 1,254 °C
- Kelvin: 147.15 °K to 1,527.15 °K
- Fahrenheit: -194.8 °F to 2,289.2 °F
Enable/disable data logger

Command syntax

The time period (n) is in 10 second intervals and can be any value from 1 to 32,000.

D,n <cr>  n = (n x 10 seconds)
D,0 <cr>  disable  default
D,? <cr>  data logger storage interval?

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D,6 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>D,0 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>D,? &lt;cr&gt;</td>
<td>?D,6 &lt;cr&gt;</td>
</tr>
</tbody>
</table>

Example

D,6

* indicates reading has been logged

60 seconds
# Memory recall

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M &lt;cr&gt;</td>
<td>recall 1 sequential stored reading</td>
</tr>
<tr>
<td>M,all &lt;cr&gt;</td>
<td>recall all readings in a CSV string</td>
</tr>
<tr>
<td>M,? &lt;cr&gt;</td>
<td>display memory location of last stored reading</td>
</tr>
</tbody>
</table>

## Example

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>M &lt;cr&gt;</td>
<td>1,100.00 &lt;cr&gt; 2,104.00 &lt;cr&gt; *OK &lt;cr&gt;</td>
</tr>
<tr>
<td>M,all &lt;cr&gt;</td>
<td>100.00,104.00,108.00,112.00 &lt;cr&gt; Oldest Newest</td>
</tr>
<tr>
<td>M,? &lt;cr&gt;</td>
<td>?M,4 &lt;cr&gt;</td>
</tr>
</tbody>
</table>

*Disable data logger to recall memory.*

---

Copyright © Atlas Scientific LLC

Atlas Scientific
Environmental Robotics

31
## Memory clear

### Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M,clear</td>
<td>clear all stored memory</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>M,clear</td>
<td>*OK</td>
</tr>
</tbody>
</table>

*Copyright © Atlas Scientific LLC*
# Naming device

## Command syntax

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

### Diagram

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

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

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,RTD,2.01 &lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

Response breakdown

?i, RTD, 2.01

↑ Device  ↑ Firmware
## Response codes

### Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*OK,1 &lt;cr&gt;</td>
<td>enable response <strong>default</strong></td>
</tr>
<tr>
<td>*OK,0 &lt;cr&gt;</td>
<td>disable response</td>
</tr>
<tr>
<td>*OK,? &lt;cr&gt;</td>
<td>response on/off?</td>
</tr>
</tbody>
</table>

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
</table>
| R <cr>  | 25.104 <cr>  
         | *OK <cr>    |
| *OK,0 <cr> | no response, *OK disabled |
| R <cr>  | 25.104 <cr>  
         | *OK disabled |
| *OK,? <cr> | ?*OK,1 <cr> or ?*OK,0 <cr> |

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

Response breakdown

?Status, P, 5.038

↑ Reason for restart  ⬆ Voltage at Vcc

Restart codes

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

## Command syntax

Send any character or command to awaken device.

**Sleep <cr>** enter sleep mode/low power

<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

*WA <cr> wakes up device

<table>
<thead>
<tr>
<th>Voltage</th>
<th>STANDBY</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>15.40 mA</td>
<td>0.4 mA</td>
</tr>
<tr>
<td>3.3V</td>
<td>13.80 mA</td>
<td>0.09 mA</td>
</tr>
</tbody>
</table>

---

## Example

**Sleep <cr>**

- *OK <cr>
- *SL <cr>

Any command

*WA <cr> wakes up device

---

**Standby** 15.40 mA

**Sleep** 3.00 mA
Change baud rate

Command syntax

Baud, n  <cr>  change baud rate

Example

<table>
<thead>
<tr>
<th>Baud, 38400  &lt;cr&gt;</th>
<th>*OK  &lt;cr&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud, ?  &lt;cr&gt;</td>
<td>?Baud, 38400  &lt;cr&gt;  *OK  &lt;cr&gt;</td>
</tr>
</tbody>
</table>

n =

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

*OK  <cr> (reboot)
# 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  [default]</td>
</tr>
<tr>
<td>Plock,?</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</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>Plock,0</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>Plock,?</td>
<td>?Plock,1 &lt;cr&gt; or ?Plock,0 &lt;cr&gt;</td>
</tr>
</tbody>
</table>

## Example Response

- **Plock,1:** "*OK <cr>"
- **I2C,100:** "cannot change to I²C  *ER <cr>"
- **Short:** "cannot change to I²C"
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>Factory &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

(reboot)

*OK <cr>

*RS <cr>

*RE <cr>

Baud rate will not change

Clears calibration
LED on
"*OK" enabled
Clears data logger
# 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>Wrong example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C,139</td>
<td>*ER &lt;cr&gt;</td>
</tr>
</tbody>
</table>

## Response

- Green *OK <cr>
- Blue (reboot)
- Blue now in I²C mode
Manual switching to I²C

- Disconnect ground (power off)
- Disconnect TX and RX
- Connect TX to PRB
- 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

Connecting TX to PRB only works for the EZO-RTD™ and the EZO-FLO™ circuits

Manually switching to I²C will set the I²C address to 102 (0x66)

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
- Temperature scale
- Hardware switch to UART mode
- LED control
- Protocol lock
- Software switch to UART mode

Settings that are NOT retained if power is cut
- Find
- Sleep mode
**I²C mode**

**I²C address**  
(0x01 – 0x7F)  
**102 (0x66) default**

**Vcc**  
3.3V – 5.5V

**Clock speed**  
100 – 400 kHz

**SDA**

**SCL**

**Data format**

**Reading**  
temperature

**Units**  
°C, °K, or °F

**Encoding**  
ASCII

**Format**  
string

**Data type**  
floating point

**Decimal places**  
3

**Smallest string**  
4 characters

**Largest string**  
40 characters

4.7k resistor may be needed
Sending commands to device

5 parts

Start  I²C address  Write  Command (not case sensitive)  Stop

102 (0x66)  ASCII command string

Example

Start  102 (0x66)  Write  Sleep  Stop

I²C address  Command

Advanced

Address bits

The entire command as ASCII with all arguments

W = low

Start

Stop
Requesting data from device

7 parts

Start I²C address Read Response code Data string Null Stop

102 (0x66) 1 byte "25.104" Terminator (Dec 0)

Advanced

Address bits SDA SCL

A6 – A0 R ACK Response code ACK Data ACK Data Data N ACK Null ACK Null

Start

N bytes of data

All bytes after data are Null

1 50 53 46 49 48 52 0 = 25.104

Dec

ASCII

25.104

CPU

SDA (TX) (RX)

SCL

GND SDA SCL

CPU

SCL SDA

SDA

(TX) (RX)

SCL

Copyright © Atlas Scientific LLC

101
Response codes

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**

```c
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**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>no data to send</td>
</tr>
<tr>
<td>254</td>
<td>still processing, not ready</td>
</tr>
<tr>
<td>2</td>
<td>syntax error</td>
</tr>
<tr>
<td>1</td>
<td>successful request</td>
</tr>
</tbody>
</table>
LED color definition

**Blue**
- I²C standby

**Green**
- Taking reading

**Purple**
- Changing I²C address

**Red**
- Command not understood

**White**
- Find

### LED color definition

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>+0.4 mA</td>
</tr>
<tr>
<td>3.3V</td>
<td>+0.2 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>pg. 67</td>
</tr>
<tr>
<td>Cal</td>
<td>performs calibration</td>
<td>pg. 53</td>
</tr>
<tr>
<td>D</td>
<td>enable/disable data logger</td>
<td>pg. 57</td>
</tr>
<tr>
<td>Export</td>
<td>export calibration</td>
<td>pg. 54</td>
</tr>
<tr>
<td>Factory</td>
<td>enable factory reset</td>
<td>pg. 66</td>
</tr>
<tr>
<td>Find</td>
<td>finds devices with white blinking LED</td>
<td>pg. 51</td>
</tr>
<tr>
<td>i</td>
<td>device information</td>
<td>pg. 61</td>
</tr>
<tr>
<td>I2C</td>
<td>change I²C address</td>
<td>pg. 65</td>
</tr>
<tr>
<td>Import</td>
<td>import calibration</td>
<td>pg. 55</td>
</tr>
<tr>
<td>L</td>
<td>enable/disable LED</td>
<td>pg. 50</td>
</tr>
<tr>
<td>M</td>
<td>memory recall/clear</td>
<td>pg. 58</td>
</tr>
<tr>
<td>Name</td>
<td>set/show name of device</td>
<td>pg. 60</td>
</tr>
<tr>
<td>Plock</td>
<td>enable/disable protocol lock</td>
<td>pg. 64</td>
</tr>
<tr>
<td>R</td>
<td>returns a single reading</td>
<td>pg. 52</td>
</tr>
<tr>
<td>S</td>
<td>temperature scale (°C, °K, °F)</td>
<td>pg. 56</td>
</tr>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
<td>pg. 63</td>
</tr>
<tr>
<td>Status</td>
<td>retrieve status information</td>
<td>pg. 62</td>
</tr>
</tbody>
</table>
LED control

Command syntax

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

Example  Response

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>L,1</td>
<td>![Wait 300ms] 10 Dec Null</td>
</tr>
<tr>
<td>L,0</td>
<td>![Wait 300ms] 10 Dec Null</td>
</tr>
<tr>
<td>L,?</td>
<td>![Wait 300ms] 1?L,10 Dec ASCII Null or 1?L,0 Dec ASCII Null</td>
</tr>
</tbody>
</table>

300ms processing delay

Example board images:

L,1

L,0
Find

Command syntax

Find LED rapidly blinks white, used to help find device

Example | Response
--- | ---
Find <cr> |  1  0
    Dec    Null
    Wait 300ms

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

300ms processing delay
# Taking reading

## Command syntax

<table>
<thead>
<tr>
<th>R</th>
<th>return 1 reading</th>
</tr>
</thead>
</table>

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>1 25.104 0</td>
</tr>
</tbody>
</table>

### Example Diagram

1. **Green**: Taking reading
2. **Wait 600ms**
3. **Transmitting**
4. **Blue**: Standby

**Processing delay**: 600ms
# Calibration

## Command syntax

- **Cal,t**  
  \( t = \) any temperature
- **Cal,clear**  
  delete calibration data
- **Cal,?**  
  device calibrated?

## Example

### Response

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
</table>
| **Cal,t** | ![](image)  
  
  1  
  Dec  
  0  
  Null |
| **Cal,clear** | ![](image)  
  
  1  
  Dec  
  0  
  Null |
| **Cal,?** | ![](image)  
  
  1  
  Dec  
  ?Cal,1  
  ASCII  
  0  
  Null | or | ![](image)  
  
  1  
  Dec  
  ?Cal,0  
  ASCII  
  0  
  Null |

---

**EZO™ RTD circuit uses single point calibration.**

**600ms** processing delay

---

**Copyright © Atlas Scientific LLC**
# Export calibration

## Command syntax

Export: Use this command to download calibration settings

<table>
<thead>
<tr>
<th>Export,?</th>
<th>calibration string info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export</td>
<td>export calibration string from calibrated device</td>
</tr>
</tbody>
</table>

## Example

### Export,?

<table>
<thead>
<tr>
<th>Wait 300ms</th>
<th>1</th>
<th>10,120</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>ASCII</td>
<td>Null</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>10,</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>

Response breakdown: 10, 120

Export strings can be up to 12 characters long

### Export

<table>
<thead>
<tr>
<th>Wait 300ms</th>
<th>1</th>
<th>59</th>
<th>6F</th>
<th>75</th>
<th>20</th>
<th>61</th>
<th>72</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>ASCII</td>
<td>Null</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>59</td>
<td>6F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1 of 10)

<table>
<thead>
<tr>
<th>Wait 300ms</th>
<th>1</th>
<th>65</th>
<th>20</th>
<th>61</th>
<th>20</th>
<th>63</th>
<th>6F</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>ASCII</td>
<td>Null</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>65</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(2 of 10)

(7 more)

<table>
<thead>
<tr>
<th>Wait 300ms</th>
<th>1</th>
<th>6F</th>
<th>6C</th>
<th>20</th>
<th>67</th>
<th>75</th>
<th>79</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>ASCII</td>
<td>Null</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>6F</td>
<td>6C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(10 of 10)

### Export

<table>
<thead>
<tr>
<th>Wait 300ms</th>
<th>1</th>
<th>*DONE</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec</td>
<td>ASCII</td>
<td>Null</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*DONE</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

*DONE
**Import calibration**

**Command syntax**

**Import,n**

import calibration string to new device

---

**Example**

<table>
<thead>
<tr>
<th>Import</th>
<th>(sequence)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import, 59 6F 75 20 61 72</td>
<td>(1 of 10)</td>
</tr>
<tr>
<td>Import, 65 20 61 20 63 6F</td>
<td>(2 of 10)</td>
</tr>
<tr>
<td>Import, 6F 6C 20 67 75 79</td>
<td>(10 of 10)</td>
</tr>
</tbody>
</table>

**Response**

```
<table>
<thead>
<tr>
<th>Wait 300ms</th>
<th>Dec</th>
<th>Null</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```

---

*If one of the imported strings is not correctly entered, the device will not accept the import and reboot.*
# Temperature scale (°C, °K, °F)

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Scale</th>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>S,c</td>
<td>celsius</td>
<td><img src="wait_300ms.png" alt="Wait 300ms" /></td>
<td>1 Dec 0 Null</td>
</tr>
<tr>
<td>S,k</td>
<td>kelvin</td>
<td><img src="wait_300ms.png" alt="Wait 300ms" /></td>
<td>1 Dec 0 Null</td>
</tr>
<tr>
<td>S,f</td>
<td>fahrenheit</td>
<td><img src="wait_300ms.png" alt="Wait 300ms" /></td>
<td>1 Dec 0 Null</td>
</tr>
<tr>
<td>S,?</td>
<td>temperature scale?</td>
<td><img src="wait_300ms.png" alt="Wait 300ms" /></td>
<td>1 Dec ASCII Null or 1 Dec ASCII Null or 1 Dec ASCII Null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scale</th>
<th>Value 1</th>
<th>Value 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>celsius</td>
<td>-126 °C</td>
<td>1,254 °C</td>
</tr>
<tr>
<td>kelvin</td>
<td>147.15 °K</td>
<td>1,527.15 °K</td>
</tr>
<tr>
<td>fahrenheit</td>
<td>-194.8 °F</td>
<td>2,289.2 °F</td>
</tr>
</tbody>
</table>
Enable/disable data logger

Command syntax

D,n  n = (n x 10 seconds)
D,0  disable
D,?  data logger storage interval?

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D,6</td>
<td>1 0 (Wait 300ms)</td>
</tr>
<tr>
<td>D,0</td>
<td>1 0 (Wait 300ms)</td>
</tr>
<tr>
<td>D,?</td>
<td>1 ?D,6 0 (Wait 300ms)</td>
</tr>
</tbody>
</table>

Example:

- D,6 (after 60 seconds)

Response:

- 1 0 (Wait 300ms)
- 1 0 (Wait 300ms)
- 1 ?D,6 0 (Wait 300ms)

The time period (n) is in 10 second intervals and can be any value from 1 to 32,000.
Memory recall

Command syntax

M recall 1 sequential stored reading
M,? display memory location of last stored reading

Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>![Wait 300ms] 1 1,100.00 0 Dec ASCII Null</td>
</tr>
<tr>
<td>M,?</td>
<td>![Wait 300ms] 1 4,112.00 0 Dec ASCII Null</td>
</tr>
</tbody>
</table>

SDA (TX) (RX)
SCL

Atlas Scientific
Environmental Robotics

Copyright © Atlas Scientific LLC
Memory clear

**Command syntax**

M,clear  clear all stored memory

**Example**

<table>
<thead>
<tr>
<th>M,clear</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Response**

<table>
<thead>
<tr>
<th>300ms processing delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait 300ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 0 Dec Null</th>
</tr>
</thead>
</table>

M,clear

---

59 Copyright © Atlas Scientific LLC
# Naming device

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name,n</td>
<td>set name</td>
<td>Name,zzt</td>
</tr>
<tr>
<td>Name,?</td>
<td>show name</td>
<td>Name,?</td>
</tr>
</tbody>
</table>

**n =**

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

Up to 16 ASCII characters

**300ms processing delay**

Do not use spaces in the name

## Example

### Name,zzt

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name,zzt</td>
<td>![Wait 300ms] 1 0 Dec Null</td>
</tr>
</tbody>
</table>

Wait 300ms

### Name,?

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name,?</td>
<td>![Wait 300ms] 1 ?Name,zzt 0 Dec ASCII Null</td>
</tr>
</tbody>
</table>

Wait 300ms

---

Copyright © Atlas Scientific LLC
## Device information

### Command syntax

Wait 300ms

### Example

<table>
<thead>
<tr>
<th>i</th>
<th>?i, RTD, 2.01</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Dec</td>
</tr>
<tr>
<td></td>
<td>ASCII</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Null</td>
</tr>
</tbody>
</table>

### Response breakdown

- **Device**: `i`
- **RTD**: `RTD`
- **Firmware**: `2.01`
# Reading device status

## Command syntax

<table>
<thead>
<tr>
<th>Command Syntax</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>?Status, P, 5.038</code></td>
<td>Response</td>
</tr>
</tbody>
</table>

### 300ms processing delay

### Status

Voltage at Vcc pin and reason for last restart

### Example

#### Status

<table>
<thead>
<tr>
<th>Command</th>
<th>Dec</th>
<th>ASCII</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>?Status,P,5.038</code></td>
<td>0</td>
<td>Null</td>
<td>1</td>
</tr>
</tbody>
</table>

### Response breakdown

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

### Restart codes

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

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

### Command syntax

<table>
<thead>
<tr>
<th>Sleep</th>
<th>enter sleep mode/low power</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example</strong></td>
<td><strong>Response</strong></td>
</tr>
<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.**

<table>
<thead>
<tr>
<th>Voltage</th>
<th>STANDBY</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>15.40 mA</td>
<td>0.4 mA</td>
</tr>
<tr>
<td>3.3V</td>
<td>13.80 mA</td>
<td>0.09 mA</td>
</tr>
</tbody>
</table>

Do not read status byte after issuing sleep command.

---

Sleep mode/low power is a feature that allows a device to enter a low power consumption mode, typically used to conserve battery life. Commands such as `Sleep` and `STANDBY` are used to transition between active and sleep states. The diagram illustrates the transition from standby to sleep mode, with corresponding I/O pins labeled for SD (Serial Data), SCL (Serial Clock), VCC (Voltage Supply), GND (Ground), and RTD (Resistance Temperature Detector). The change in current consumption from 15.40 mA in standby to 0.4 mA in sleep mode is significant, indicating a substantial power savings.

---

Copyright © Atlas Scientific LLC
# Protocol lock

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plock,1</td>
<td>Enable Plock</td>
<td>Plock,1</td>
<td>1 Dec 0 ASCII Null</td>
</tr>
<tr>
<td>Plock,0</td>
<td>Disable Plock</td>
<td>Plock,0</td>
<td>1 Dec 0 Null</td>
</tr>
<tr>
<td>Plock,?</td>
<td>Plock on/off?</td>
<td>Plock,?</td>
<td>? Plock,1 0</td>
</tr>
</tbody>
</table>

300ms processing delay

Locks device to I^2^C mode.

### Example

#### Plock,1

- Command: Plock,1
- Example: Wait 300ms
- Response: 1 Dec 0 ASCII Null

#### Plock,0

- Command: Plock,0
- Example: Wait 300ms
- Response: 1 Dec 0 Null

#### Plock,?

- Command: Plock,?
- Example: Wait 300ms
- Response: 1 Dec ?Plock,1 0

### Baud, 9600

- Baud rate: 9600
- Cannot change to UART

### Error

- Error message: cannot change to UART

---

Copyright © Atlas Scientific LLC
**I²C address change**

**Command syntax**

I²C,n  sets I²C address and reboots into I²C mode

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>I²C,100</td>
<td>device reboot</td>
</tr>
</tbody>
</table>

**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 102 (0x66).

n = any number 1 – 127
Factory reset

Command syntax

Factory enable factory reset

Example

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

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

I²C address will not change

Clears calibration
LED on
Response codes enabled
Clears data logger

Factory

(reboot)
## Change to UART mode

### Command syntax

| Baud, n | switch from \( \text{i}^2\text{C} \) to UART |

### Example

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

\( n = \begin{bmatrix} 300 \\ 1200 \\ 2400 \\ 9600 \\ 19200 \\ 38400 \\ 57600 \\ 115200 \end{bmatrix} \)

---

---
Manual switching to UART

- Disconnect ground (power off)
- Disconnect TX and RX
- Connect TX to PRB
- 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

Connecting TX to PRB only works for the EZO-RTD™ and the EZO-FLO™ circuits
**EZO™ circuit footprint**

1. In your CAD software place an 8 position header.
2. Place a 3 position header at both top and bottom of the 8 position.
3. Delete the 8 position header. The two 3 position headers are now 17.78mm (0.7”) apart from each other.

**Specifications:**
- **2.54mm (0.1”)**
- **17.78mm (0.7”)**
- **1.05mm Ø**
Datasheet change log

Datasheet V 3.5
Added 2 wire, 3 wire, or 4 wire, wiring diagram on pg 7.

Datasheet V 3.4
Revised accuracy equation on pg 7.

Datasheet V 3.3
Moved Default state to pg 13.

Datasheet V 3.2
Revised response for the sleep command in UART mode on pg 36.

Datasheet V 3.1
Added more information on the Export calibration and Import calibration commands.

Datasheet V 3.0
Changed "Max rate" to "Response time" on cover page.

Datasheet V 2.9
Removed note from certain commands about firmware version.

Datasheet V 2.8
Added information to calibration theory on pg 9.

Datasheet V 2.7
Revised definition of response codes on pg 45.

Datasheet V 2.6
Updated calibration processing delay time on pg.51.

Datasheet V 2.5
Revised Plock pages to show default value.
Firmware updates

V1.02 – Plock (March 31, 2016)
• Added protocol lock feature “Plock”

V1.03 – EEPROM (April 26, 2016)
• Fixed bug where EEPROM would get erased if the circuit lost power 900ms into startup

V1.11 – Bug Fix (June 9, 2016)
• Fixed bug where a blank name would result in garbage output

V2.01 – Update (January 1, 2017)
• Replaced command “response” with “*OK”
• Replaced command “Serial” with “Baud”

V2.02 – Bug Fix (February 16, 2017)
• Fixed bug where calibration would not accept floating point numbers.

V2.10 – (May 9, 2017)
• Added “Find” command.
• Added “Export/import” command.
• Modified continuous mode to be able to send readings every “n” seconds.
• Sleep current is lowered.

V2.11 - Bug Fix (November 19, 2020)
• Fixed bug where the first reading after boot up could be -1024

Datasheet change log

Datasheet V 2.4

Added new commands:
"Find" pages 22 & 49.
"Export/Import calibration" pages 26 & 52.
Added new feature to continuous mode "C,n" pg 23.

Datasheet V 2.3

Added manual switching to UART information on pg. 59.

Datasheet V 2.2

Revised Baud command information on pg. 33.

Datasheet V 2.1

Revised entire datasheet.
Warranty

Atlas Scientific™ Warranties the EZO™ class RTD circuit to be free of defect during the debugging phase of device implementation, or 30 days after receiving the EZO™ class RTD circuit (which ever comes first).

The debugging phase

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

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