EZO-RTD™
Embedded Temperature Circuit

Reads

Range
-126.000 °C – 1254 °C

Resolution
0.001

Accuracy
+/- (0.10°C + 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 an evolving document, check back for updates.

Written by Jordan Press
Designed by Noah Press
STOP

SOLDERING THIS DEVICE VOIDS YOUR WARRANTY.

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.

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  Using other brand PT-100/PT-1000 7
Power consumption 4  Operating principle 8
Absolute max ratings 4  Correct wiring 10
Temperature circuit range 5  Calibration theory 11
Temperature circuit accuracy 5  On board data logger 12
Atlas Scientific PT-1000 probe 6  Available data protocols 13

UART

UART mode 15
Default state 16
Receiving data from device 17
Sending commands to device 18
LED color definition 19
**UART quick command page** 20
LED control 21
Find 22
Continuous reading mode 23
Single reading mode 24
Calibration 25
Export calibration 26
Import calibration 27
Temperature scale 28
Enable/disable data logger 29
Memory recall 30
Memory clear 31
Naming device 32
Device information 33
Response codes 34
Reading device status 35
Sleep mode/lowlower power 36
Change baud rate 37
Protocol lock 38
Factory reset 39
Change to I²C mode 40
Manual switching to I²C 41

I²C

I²C mode 43
Sending commands 44
Requesting data 45
Response codes 46
LED color definition 47
**I²C quick command page** 48
LED control 49
Find 50
Taking reading 51
Calibration 52
Export calibration 53
Import calibration 54
Temperature scale 55
Enable/disable data logger 56
Memory recall 57
Memory clear 58
Device information 59
Reading device status 60
Sleep mode/lowlower power 61
Protocol lock 62
I²C address change 63
Factory reset 64
Change to UART mode 65
Manual switching to UART 66

Circuit footprint 67
Datasheet change log 68
Warranty 70
**Power consumption**

<table>
<thead>
<tr>
<th>5V</th>
<th>LED</th>
<th>MAX</th>
<th>STANDBY</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>16 mA</td>
<td>15.4 mA</td>
<td>0.4 mA</td>
<td></td>
</tr>
<tr>
<td>OFF</td>
<td>15.3 mA</td>
<td>15 mA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3.3V</th>
<th>LED</th>
<th>MAX</th>
<th>STANDBY</th>
<th>SLEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>14.3 mA</td>
<td>13.8 mA</td>
<td>0.09 mA</td>
<td></td>
</tr>
<tr>
<td>OFF</td>
<td>14 mA</td>
<td>13.6 mA</td>
<td></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 (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>85 °C</td>
<td></td>
</tr>
<tr>
<td>VCC</td>
<td>3.3V</td>
<td>5V</td>
<td>5.5V</td>
</tr>
</tbody>
</table>
**EZO™ RTD temperature sensing range**

-126 °C 0 °C 200 °C 400 °C 600 °C 800 °C 1,000 °C 1,254 °C

**EZO™ RTD temperature sensing accuracy**

-126 °C 0 °C 200 °C 400 °C 600 °C 800 °C 1,000 °C 1,254 °C

-2 °C 0 °C 2 °C

Copyright © Atlas Scientific LLC
To read temperatures above, or below the max cable temperature, an additional probe housing (thermowell) is needed to protect the cable.

Atlas Scientific PT-1000 probe

• Accuracy +/- (0.15 + (0.002*t))
• Probe type: class A platinum, RTD
• Cable length: 81cm (32”)
• Cable material: silicone rubber
• 30mm sensing area (304 SS)
• 6mm diameter
• BNC connector
• Reaction time: 90% value in 13 seconds
• Probe output: analog
• Full sensing range -200 °C to 850 °C
• Cable max temp 125 °C
• Cable min temp -55 °C

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>+/- (0.10°C + 0.0017 x T)</td>
</tr>
<tr>
<td>A</td>
<td>+/- (0.15°C + 0.002 x T)</td>
</tr>
<tr>
<td>B</td>
<td>+/- (0.3°C + 0.005 x T)</td>
</tr>
<tr>
<td>C</td>
<td>+/- (0.6°C + 0.01 x T)</td>
</tr>
</tbody>
</table>

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

BOTH ARE CORRECT
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, platinum's 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 NOT NEEDED.
Correct wiring

- Bread board
- Extended leads
- Sloppy setup
- Perfboards or Protoboards

*Embedded into your device

Incorrect wiring

- Extended leads
- Sloppy setup
- Perfboards or Protoboards

NEVER use Perfboards or Protoboards

*Only after you are familiar with EZO™ circuits operation
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

<table>
<thead>
<tr>
<th>Minimum</th>
<th>10 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>320,000 seconds</td>
</tr>
</tbody>
</table>

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.
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
- 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

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

**RX**  
Data in

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

**Vcc**  
3.3V – 5.5V

**Data format**

<table>
<thead>
<tr>
<th>Reading</th>
<th>temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Units</td>
<td>°C, °K, or °F</td>
</tr>
<tr>
<td>Encoding</td>
<td>ASCII</td>
</tr>
<tr>
<td>Format</td>
<td>string</td>
</tr>
<tr>
<td>Terminator</td>
<td>carriage return</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data type</th>
<th>floating point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal places</td>
<td>3</td>
</tr>
<tr>
<td>Smallest string</td>
<td>4 characters</td>
</tr>
<tr>
<td>Largest string</td>
<td>40 characters</td>
</tr>
</tbody>
</table>
Default state

Mode: UART
Baud: 9,600
Temperature: °C
Readings: continuous
Speed: 1 reading per second
With probe: ttt.ttt
Without probe: -1023.000
LED: on

1,000 ms

Green
Standby

Cyan
Taking reading

Transmitting
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
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>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud</td>
<td><strong>change baud rate</strong></td>
<td><strong>9,600</strong></td>
<td><strong>37</strong></td>
</tr>
<tr>
<td>C</td>
<td><strong>enable/disable continuous reading</strong></td>
<td><strong>enabled</strong></td>
<td><strong>23</strong></td>
</tr>
<tr>
<td>Cal</td>
<td><strong>performs calibration</strong></td>
<td><strong>n/a</strong></td>
<td><strong>25</strong></td>
</tr>
<tr>
<td>D</td>
<td><strong>enable/disable data logger</strong></td>
<td><strong>disabled</strong></td>
<td><strong>29</strong></td>
</tr>
<tr>
<td>Export</td>
<td><strong>export calibration</strong></td>
<td><strong>n/a</strong></td>
<td><strong>26</strong></td>
</tr>
<tr>
<td>Factory</td>
<td><strong>enable factory reset</strong></td>
<td><strong>n/a</strong></td>
<td><strong>39</strong></td>
</tr>
<tr>
<td>Find</td>
<td><strong>finds device with blinking white LED</strong></td>
<td><strong>n/a</strong></td>
<td><strong>22</strong></td>
</tr>
<tr>
<td>i</td>
<td><strong>device information</strong></td>
<td><strong>n/a</strong></td>
<td><strong>33</strong></td>
</tr>
<tr>
<td>I2C</td>
<td><strong>change to I²C mode</strong></td>
<td><strong>not set</strong></td>
<td><strong>40</strong></td>
</tr>
<tr>
<td>Import</td>
<td><strong>import calibration</strong></td>
<td><strong>n/a</strong></td>
<td><strong>27</strong></td>
</tr>
<tr>
<td>L</td>
<td><strong>enable/disable LED</strong></td>
<td><strong>enabled</strong></td>
<td><strong>21</strong></td>
</tr>
<tr>
<td>M</td>
<td><strong>memory recall/clear</strong></td>
<td><strong>n/a</strong></td>
<td><strong>30</strong></td>
</tr>
<tr>
<td>Name</td>
<td><strong>set/show name of device</strong></td>
<td><strong>not set</strong></td>
<td><strong>32</strong></td>
</tr>
<tr>
<td>Plock</td>
<td><strong>enable/disable protocol lock</strong></td>
<td><strong>disabled</strong></td>
<td><strong>38</strong></td>
</tr>
<tr>
<td>R</td>
<td><strong>returns a single reading</strong></td>
<td><strong>n/a</strong></td>
<td><strong>24</strong></td>
</tr>
<tr>
<td>S</td>
<td><strong>temperature scale (°C, °K, °F)</strong></td>
<td><strong>celsius</strong></td>
<td><strong>28</strong></td>
</tr>
<tr>
<td>Sleep</td>
<td><strong>enter sleep mode/low power</strong></td>
<td><strong>n/a</strong></td>
<td><strong>36</strong></td>
</tr>
<tr>
<td>Status</td>
<td><strong>retrieve status information</strong></td>
<td><strong>n/a</strong></td>
<td><strong>35</strong></td>
</tr>
<tr>
<td>*OK</td>
<td><strong>enable/disable response codes</strong></td>
<td><strong>enable</strong></td>
<td><strong>34</strong></td>
</tr>
</tbody>
</table>
**LED control**

**Command syntax**

- L,1 <cr> LED on **default**
- L,0 <cr> LED off
- L,? <cr> LED state on/off?

**Example**

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

Command syntax

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

Example

Find  <cr>

Response

*OK  <cr>

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

### Command syntax

- **C,1** `<cr>` enable continuous readings once per second [default]
- **C,n** `<cr>` continuous readings every n seconds (n = 2 to 99 sec)
- **C,0** `<cr>` disable continuous readings
- **C,?** `<cr>` continuous reading mode on/off?

### Example & Response

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
</table>
| **C,1**  | *OK `<cr>`  
|          | °C (1 sec) `<cr>`                            |
|          | °C (2 sec) `<cr>`                            |
|          | °C (n sec) `<cr>`                            |
| **C,30** | *OK `<cr>`  
|          | °C (30 sec) `<cr>`                           |
|          | °C (60 sec) `<cr>`                           |
|          | °C (90 sec) `<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

<table>
<thead>
<tr>
<th>R &lt;cr&gt;</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>R &lt;cr&gt;</td>
<td>25.104 &lt;cr&gt; *OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

Example Response

Green
Standby

Cyan
Taking reading

Transmitting

600 ms
Calibration

**Command syntax**

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

**Example**

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

---

The EZO™ RTD circuit uses single point calibration.

Cal,t <cr>  \( t = \) any temperature

Cal,clear <cr> delete calibration data

Cal,? <cr> device calibrated?

Example

- **Cal,100.00 <cr>**
  - *OK <cr>*
- **Cal,clear <cr>**
  - *OK <cr>*
- **Cal,? <cr>**
  - ?Cal,1 <cr> or ?Cal,0 <cr>*OK <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> | 10,120 <cr> |
| Export <cr> | 59 6F 75 20 61 72 <cr> (1 of 10) |
| Export <cr> | 65 20 61 20 63 6F <cr> (2 of 10) |
| (7 more) | *DONE |
| Export <cr> | 6F 6C 20 67 75 79 <cr> (10 of 10) |

### 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>.

Disabling `*OK` simplifies this process.

---

<table>
<thead>
<tr>
<th>Export &lt;cr&gt;</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>RX</td>
<td>TX</td>
</tr>
<tr>
<td>[10,120]</td>
<td>10</td>
<td>2</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>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import, 59 6F 75 20 61 72 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>Import, 65 20 61 20 63 6F &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
<tr>
<td>Import, 6F 6C 20 67 75 79 &lt;cr&gt;</td>
<td>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

* If one of the imported strings is not correctly entered, the device will not accept the import, respond with *ER and reboot.

---

1. Import: Use this command to upload calibration settings to one or more devices.
## Temperature scale (°C, °K, °F)

### Command syntax

- **S,c <cr>** celsius **default**
- **S,k <cr>** kelvin
- **S,f <cr>** fahrenheit
- **S,? <cr>** temperature scale?

### Example

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

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

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

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

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>D,6</td>
<td></td>
</tr>
</tbody>
</table>

* indicates reading has been logged
# 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</td>
</tr>
<tr>
<td>M,? &lt;cr&gt;</td>
<td>?M,4 &lt;cr&gt; *OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

*Disable data logger to recall memory.*
Memory clear

Command syntax

M,clear <cr>  clear all stored memory

Example

<table>
<thead>
<tr>
<th>M,clear &lt;cr&gt;</th>
</tr>
</thead>
</table>

Response

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

Example Response

M,clear <cr>

Copyright © Atlas Scientific LLC
## Naming device

### Command syntax

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

### 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; *OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

### Response

**Example Response**

```
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  1 62 73 84 95 10 11 12 13 14 15 16
n =
```

1,000 ms

Green

Standby

Cyan

Taking reading

Transmitting

**Example Response**

```
Name,zzt <cr>
```

**Response**

```
*OK <cr>
```

---

*Copyright © Atlas Scientific LLC*
## Device information

### Command syntax

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

### Example

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

### Response breakdown

```
?i, RTD, 2.01
```

- **Device**: RTD
- **Firmware**: 2.01
# 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 (default)</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>
<tbody>
<tr>
<td>R 0.1</td>
<td>25.104 &lt;cr&gt; *OK &lt;cr&gt;</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

```
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>*OK &lt;cr&gt;</td>
</tr>
</tbody>
</table>

## Response breakdown

<table>
<thead>
<tr>
<th>?Status,</th>
<th>P, 5.038</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason for restart</td>
<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** `<cr>`  enter sleep mode/low power

### Example

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

### Response

*WA `<cr>`  wakes up device

### Power Consumption

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Standby (mA)</th>
<th>Sleep (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>15.40</td>
<td>0.4</td>
</tr>
<tr>
<td>3.3V</td>
<td>13.80</td>
<td>0.09</td>
</tr>
</tbody>
</table>

---

Copyright © Atlas Scientific LLC
# Change baud rate

## Command syntax

<table>
<thead>
<tr>
<th>Baud,n</th>
<th>change baud rate</th>
</tr>
</thead>
</table>

## Example

<table>
<thead>
<tr>
<th>Baud,38400</th>
<th>*OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baud,?</td>
<td>?Baud,38400</td>
</tr>
<tr>
<td></td>
<td>*OK</td>
</tr>
</tbody>
</table>

## Default baud rates

<table>
<thead>
<tr>
<th>Baud Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
</tr>
<tr>
<td>1200</td>
</tr>
<tr>
<td>2400</td>
</tr>
<tr>
<td>9600</td>
</tr>
<tr>
<td>19200</td>
</tr>
<tr>
<td>38400</td>
</tr>
<tr>
<td>57600</td>
</tr>
<tr>
<td>115200</td>
</tr>
</tbody>
</table>

![Diagram showing the process of changing baud rate and rebooting](image-url)
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>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>

Plock,1

*OK <cr>

cannot change to I²C

*ER <cr>

I²C,100

cannot change to I²C

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>

Factory <cr>

(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

I²C,100 <cr>

Response

*OK (reboot in I²C mode)

Wrong example

I²C,139 <cr>  n ≠ 127

Response

*ER <cr>

I²C,100

(reboot)

Green

*OK <cr>

Blue

now in I²C mode
Manual switching to I²C

- Make sure Plock is set to 0
- 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

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

Example

Wrong Example

Connecting TX to PRB only works for the EZO-RTD™ and the EZO-FLO™ circuits
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
- Find
- Sleep mode
I²C mode

I²C address  (0x01 – 0x7F)

102 (0x66) default

Vcc  3.3V – 5.5V

Clock speed  100 – 400 kHz

SDA  

SCL  

0V 0V

VCC

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

CPU

SDA  SCL

(TX) (RX)

VCC

43 Copyright © Atlas Scientific LLC
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

Sending commands to device

Advanced

Address bits | The entire command as ASCII with all arguments

Start | A6 A5 A4 A3 A2 A1 A0 W ACK | First letter of command \ACK \ACK | Last letter of command \ACK \ACK

4.7k resistor may be needed

0V 0V VCC CPU SCL SDA (TX) (RX)
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)

Response code
Read
Null
R  = High
ACK
NACK
Null
Null
Stop

Advanced

Address bits
N bytes of data
All bytes after data are Null

1 50 53 46 49 48 52 0 = 25.104

Dec
Dec

ASCII

Copyright © Atlas Scientific LLC
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**

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

delay(300);
```

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 ON**
- 5V: +0.4 mA
- 3.3V: +0.2 mA
# 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. 65</td>
</tr>
<tr>
<td>Cal</td>
<td>performs calibration</td>
<td>pg. 52</td>
</tr>
<tr>
<td>D</td>
<td>enable/disable data logger</td>
<td>pg. 56</td>
</tr>
<tr>
<td>Export</td>
<td>export calibration</td>
<td>pg. 53</td>
</tr>
<tr>
<td>Factory</td>
<td>enable factory reset</td>
<td>pg. 64</td>
</tr>
<tr>
<td>Find</td>
<td>finds devices with white blinking LED</td>
<td>pg. 50</td>
</tr>
<tr>
<td>i</td>
<td>device information</td>
<td>pg. 59</td>
</tr>
<tr>
<td>I2C</td>
<td>change I²C address</td>
<td>pg. 63</td>
</tr>
<tr>
<td>Import</td>
<td>import calibration</td>
<td>pg. 54</td>
</tr>
<tr>
<td>L</td>
<td>enable/disable LED</td>
<td>pg. 49</td>
</tr>
<tr>
<td>M</td>
<td>memory recall/clear</td>
<td>pg. 57</td>
</tr>
<tr>
<td>Plock</td>
<td>enable/disable protocol lock</td>
<td>pg. 62</td>
</tr>
<tr>
<td>R</td>
<td>returns a single reading</td>
<td>pg. 51</td>
</tr>
<tr>
<td>S</td>
<td>temperature scale (°C, °K, °F)</td>
<td>pg. 55</td>
</tr>
<tr>
<td>Sleep</td>
<td>enter sleep mode/low power</td>
<td>pg. 61</td>
</tr>
<tr>
<td>Status</td>
<td>retrieve status information</td>
<td>pg. 60</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</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 Response

<table>
<thead>
<tr>
<th>Example</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>L,1</td>
<td>![Image of LED board with L,1]</td>
</tr>
<tr>
<td>L,0</td>
<td>![Image of LED board with L,0]</td>
</tr>
<tr>
<td>L,?</td>
<td>![Image of LED board with L,?]</td>
</tr>
</tbody>
</table>

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms

or

300ms processing delay

Example:

L,1

Response:

Wait 300ms

L,0

Response:

Wait 300ms

L,?

Response:

Wait 300ms
Find

**Command syntax**

Find    LED rapidly blinks white, used to help find device

**Example**

Find `<cr>`

300ms processing delay

**Response**

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

Wait 300ms

Dec 0 Null

Send any character or command to terminate find.

Copyright © Atlas Scientific LLC
Taking reading

Command syntax

**R**  return 1 reading

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R</strong></td>
<td>![Time](Wait 600ms) 1 Dec 25.104 ASCII 0 Null</td>
</tr>
</tbody>
</table>

**Example Response**

- **600ms processing delay**
- **R** return 1 reading
- **Wait 600ms**
- **Dec** 1
- **ASCII** 25.104
- **Null** 0

**Diagram**

- **Green** Taking reading
- **Transmitting**
- **Blue** Standby

---

**Copyright © Atlas Scientific LLC**
# Calibration

## Command syntax

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cal,t</td>
<td>( t = \text{any temperature} )</td>
</tr>
<tr>
<td>Cal,clear</td>
<td>delete calibration data</td>
</tr>
<tr>
<td>Cal,?</td>
<td>device calibrated?</td>
</tr>
</tbody>
</table>

**Example Response**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cal,t</td>
<td>( 1 ) ( 0 ) Dec Null</td>
</tr>
<tr>
<td>Cal,clear</td>
<td>( 1 ) ( 0 ) Dec Null</td>
</tr>
<tr>
<td>Cal,?</td>
<td>( 1 ) ?Cal,1 0 Dec ASCII Null or ( 1 ) ?Cal,0 0 Dec ASCII Null</td>
</tr>
</tbody>
</table>

**Example**

- **Cal,t**
  - Wait 600ms

- **Cal,clear**
  - Wait 300ms

- **Cal,?**
  - Wait 300ms

---

**Diagram**

- **Cal,100.00**
  - 100.35 °C

---

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

### Command syntax

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

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Example Response</th>
<th>Response breakdown</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export,?</td>
<td>Wait 300ms</td>
<td>Dec 10,120 ASCII 0</td>
</tr>
<tr>
<td>Export</td>
<td>Wait 300ms</td>
<td>Dec 59 6F 75 20 61 72 ASCII 0</td>
</tr>
<tr>
<td>Export</td>
<td>Wait 300ms</td>
<td>Dec 65 20 61 20 63 6F ASCII 0</td>
</tr>
<tr>
<td>Export</td>
<td>Wait 300ms</td>
<td>Dec 6F 6C 20 67 75 79 ASCII 0</td>
</tr>
<tr>
<td>Export</td>
<td>Wait 300ms</td>
<td>Dec <em>DONE</em> ASCII 0</td>
</tr>
</tbody>
</table>

### Response breakdown

- **# of strings to export**: 10
- **# of bytes to export**: 120

Export strings can be up to 12 characters long.

---

*Copyright © Atlas Scientific LLC*
Import calibration

Command syntax

Import, n import calibration string to new device

Example

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

Response

* If one of the imported strings is not correctly entered, the device will not accept the import and reboot.

SDA (TX) (RX) SCL

reboot

MCU

10,120

system will reboot

Copyright © Atlas Scientific LLC
Temperature scale (°C, °K, °F)

**Command syntax**

<table>
<thead>
<tr>
<th>Command</th>
<th>Temperature Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>S,c</td>
<td>celsius (default)</td>
</tr>
<tr>
<td>S,k</td>
<td>kelvin</td>
</tr>
<tr>
<td>S,f</td>
<td>fahrenheit</td>
</tr>
<tr>
<td>S,?</td>
<td>temperature scale?</td>
</tr>
</tbody>
</table>

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
</table>
| S,c     | **Wait 300ms**
|         | 1 Dec 0 Null |
| S,k     | **Wait 300ms**
|         | 1 Dec 0 Null |
| S,f     | **Wait 300ms**
|         | 1 Dec 0 Null |
| S,?     | **Wait 300ms**
|         | 1 Dec ASCII Null | or | 1 Dec ASCII Null | or | 1 Dec ASCII Null |

**Temperature Ranges**

-126 °C to 1,254 °C (celsius)
147.15 °K to 1,527.15 °K (kelvin)
-194.8 °F to 2,289.2 °F (fahrenheit)
Enable/disable data logger

**Command syntax**

- **D,n**  \( n = (n \times 10 \text{ seconds}) \)
- **D,0**  disable
- **D,?**  data logger storage interval?

**Example**

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
</table>
| **D,6** | ![Wait 300ms](image1)  
  \( 1 \) Dec  
  \( 0 \) Null |
| **D,0** | ![Wait 300ms](image2)  
  \( 1 \) Dec  
  \( 0 \) Null |
| **D,?** | ![Wait 300ms](image3)  
  \( 1 \) Dec  
  \( ?D,6 \) ASCII  
  \( 0 \) Null |

**300ms processing delay**

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

**Example Response**

- **300ms processing delay**
  - Wait 300ms

**Example**

- **D,6** (after 60 seconds)

---

**Copyright © Atlas Scientific LLC**
## 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><strong>M</strong></td>
<td><img src="1" alt="Wait 300ms" /> 1,100.00 0</td>
</tr>
<tr>
<td><strong>M,?</strong></td>
<td><img src="1" alt="Wait 300ms" /> 4,112.00 0</td>
</tr>
</tbody>
</table>

**300ms processing delay**
Memory clear

Command syntax
M,clear  clear all stored memory

Example
M,clear

Response
Wait 300ms
1
Dec
Null

M,clear
## Device information

### Command syntax

| i | device information |

### Example

<table>
<thead>
<tr>
<th>i</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>[\text{Wait 300ms}]</td>
</tr>
</tbody>
</table>

### Response breakdown

<table>
<thead>
<tr>
<th>?i, RTD, 2.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
</tr>
</tbody>
</table>
## Reading device status

### 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, P, 5.038</td>
<td>0 (Null)</td>
</tr>
</tbody>
</table>

**Wait 300ms**

**Dec** | **ASCII** | **Null**

### Response breakdown

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

### Restart codes

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

---

**Copyright © Atlas Scientific LLC**

60
Sleep mode/low power

Command syntax

Sleep  enter sleep mode/low power

Example       Response
Sleep          no response

Any command  wakes up device

<table>
<thead>
<tr>
<th>Voltage</th>
<th>STANDBY (mA)</th>
<th>SLEEP (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5V</td>
<td>15.40</td>
<td>0.40</td>
</tr>
<tr>
<td>3.3V</td>
<td>13.80</td>
<td>0.09</td>
</tr>
</tbody>
</table>
# 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>

- **Plock,1** enables Plock, locking the device to I2C mode.
- **Plock,0** disables Plock, defaulting to UART mode.
- **Plock,?** queries the status of Plock.

### Example

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plock,1</td>
<td>1 Dec 0 Null</td>
</tr>
<tr>
<td>Plock,0</td>
<td>1 Dec 0 Null</td>
</tr>
<tr>
<td>Plock,?</td>
<td>1 Dec ASCII 0 Null</td>
</tr>
</tbody>
</table>

### Example Response

- **Plock,1**
  - **Wait 300ms**
  - 1 Dec 0 Null
- **Plock,0**
  - **Wait 300ms**
  - 1 Dec 0 Null
- **Plock,?**
  - **Wait 300ms**
  - 1 Dec ASCII 0 Null

### Example

- **Plock,1** with Baud, 9600
- **Baud, 9600**

- **Cannot change to UART**
- **Cannot change to UART**
## I²C address change

**Command syntax**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I²C, n</td>
<td>sets I²C address and reboots into I²C mode</td>
</tr>
</tbody>
</table>

### Example | Response
--- | ---
I²C, 100 | 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 102 (0x66).

### Example

$\text{I²C,100}$

---

Copyright © Atlas Scientific LLC
**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**

Factory

**Response**

device reboot

Clears calibration
LED on
Response codes enabled
Clears data logger

Factory

(reboot)
## Change to UART mode

### Command syntax

**Baud,n** switch from I²C to UART

### Example | Response
---|---
Baud,9600 | reboot in UART mode

n =

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

---

Copyright © Atlas Scientific LLC
Manual switching to UART

- Make sure Plock is set to 0
- 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
EZO™ circuit footprint

1. In your CAD software place a 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.

17.78mm (0.7”)

2.54mm (0.1”)
Datasheet change log

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.

Datasheet V 2.4
Added new commands:
"Find" pages 22 & 49.
"Export/Import calibration" pages 26 & 52.
   Added new feature to continous 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.
## Firmware updates

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.02</td>
<td>March 31, 2016</td>
<td>• Added protocol lock feature “Plock”</td>
</tr>
<tr>
<td>V1.03</td>
<td>April 26, 2016</td>
<td>• Fixed bug where EEPROM would get erased if the circuit lost power 900ms into startup</td>
</tr>
<tr>
<td>V1.11</td>
<td>June 9, 2016</td>
<td>• Fixed bug where a blank name would result in garbage output</td>
</tr>
<tr>
<td>V2.01</td>
<td>January 1, 2017</td>
<td>• Replaced command “response” with “*OK”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Replaced command “Serial” with “Baud”</td>
</tr>
<tr>
<td>V2.02</td>
<td>February 16, 2017</td>
<td>• Fixed bug where calibration would not accept floating point numbers.</td>
</tr>
<tr>
<td>V2.10</td>
<td>May 9, 2017</td>
<td>• Added &quot;Find&quot; command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added &quot;Export/import&quot; command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Modified continuous mode to be able to send readings every &quot;n&quot; seconds.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Sleep current is lowered.</td>
</tr>
</tbody>
</table>
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 embed-
dded 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.