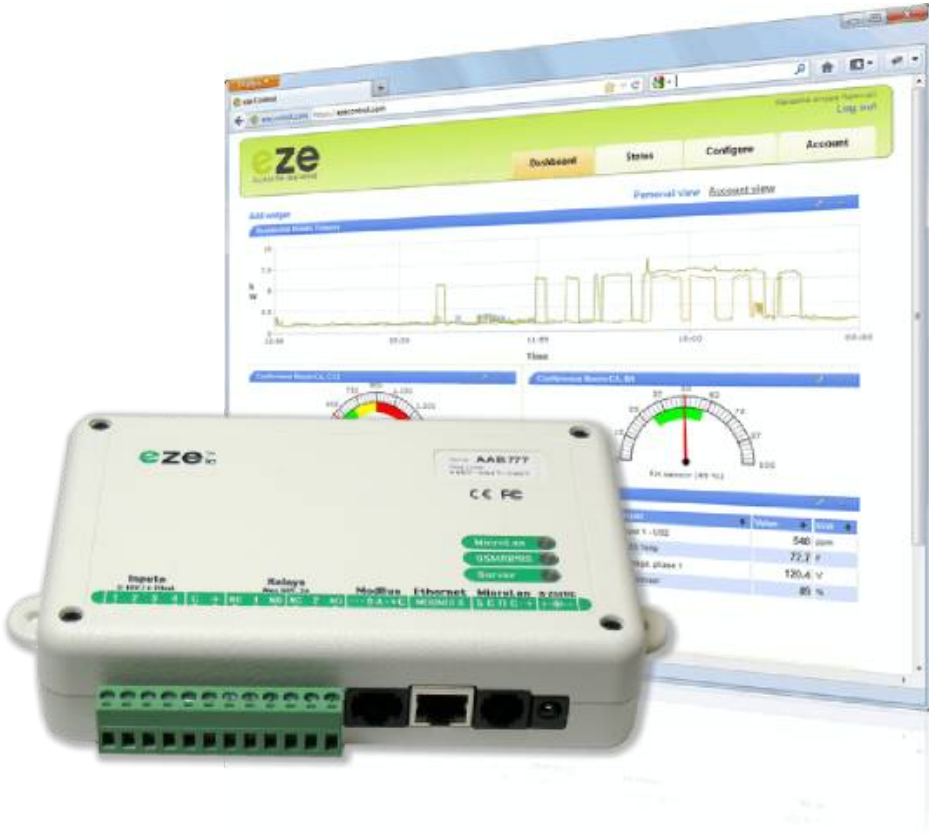


ezeio™ user manual



Manual version 170314
ezeio™ models AAC – AAF

Important information

WARNINGS



To reduce risk of fire or electric shock, do not expose this product to rain or moisture. This product is designed for use indoors and only with the supplied AC adapter.

Unplug the AC adapter before opening the cover.



The ezeio™ is a low voltage device.

Never connect high voltage to the inputs or outputs.



MicroLAN: Never use connectors or wires designed phone networks to connect MicroLAN devices.

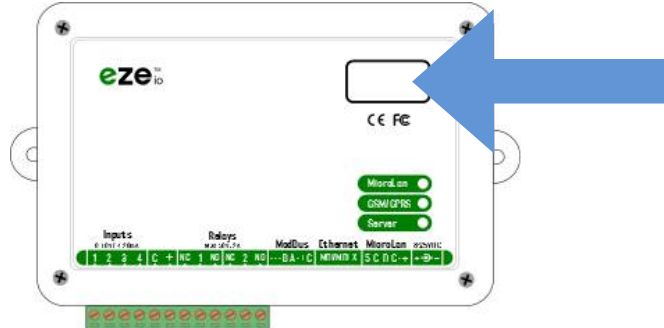
Phone connectors usually alter the polarity, and **will permanently damage MicroLAN devices**, voiding the warranty.



SIM Card & Cellular Antenna: **Always disconnect the power adapter from the ezeio™ when installing or removing the SIM card or antennas.**

Registration

This product is identified by a unique serial number and a registration code on the front of the unit.



You will need this information to communicate with the product.
When activating the unit, you will also be assigned an account number.
Make a note of this important information below

Account Number:

Serial Number:

Registration code:

Support contact information

Go to www.ezecontrol.com for support and contact information.

Table of Contents

ezeio™ user manual

Important information

WARNINGS.....	2
Registration.....	3
Support contact information.....	3

Introduction

What is the ezeio™?.....	7
Model information.....	8

Creating accounts and users

Overview.....	9
Creating a new account.....	9
Add an ezeio to an existing account.....	10
Adding users to an existing account.....	10
Moving or removing an ezeio from an account.....	10

Connections and installation

Things to consider before installing the ezeio™.....	11
ezeio™ overview.....	12
Power connection.....	13
Network connection.....	13
General purpose inputs.....	15
Inputs – Pulse, switch or resistive sensors.....	16
Inputs – External voltage sources.....	17
Inputs – Current sensors.....	18
Relay outputs.....	19
+ DC output terminal.....	19
MicroLAN.....	20
Modbus / serial port.....	22
GSM/3G/GPS module (select models only).....	25

Web interface overview

Logging in.....	29
-----------------	----

Dashboard screen

Dashboards.....	31
-----------------	----

Status screen

Live input status.....	33
Output status and control.....	33
Thermostat status.....	34
Event log.....	34
Downloading log data.....	34
Viewing graph of log data.....	35

Configure screen

ezeio™ Configuration.....	37
Resource tree.....	37

Inputs.....	38
Calibrating analog inputs.....	44
Alarm settings.....	46
Actions.....	47
Conditions.....	48
Outputs.....	49
Schedules.....	50
Timers.....	50
Thermostats / Thermostat schedules.....	51
Devices.....	52
Script (<i>premium feature</i>).....	54
System.....	55
Actions	
Action: Send message.....	60
Action: Log event.....	66
Action: Set output.....	66
Action: Set counter.....	66
Action: Increment counter.....	66
Action: Decrement counter.....	66
Action: Control thermostat.....	66
Action: ModBus coil control.....	67
Action: ModBus write register.....	67
Account screen	
Account.....	68
Personal.....	68
Users.....	69
Sending control commands	
Email.....	70
Control via SMS (cellphone texting).....	70
Control Commands.....	71
Server API	
API access and security.....	73
Live status in JSON format via REST API.....	75
Historical data access in JSON format via REST API.....	75
Controlling the ezeio™ via REST API.....	76
Spreadsheet integration.....	81
Automatic export (push).....	83
Script language	
Script introduction.....	88
Script function library	
Configuration interface functions.....	92
Calendar and time functions.....	96
Mathematical functions.....	98
Language functions.....	102
String functions.....	103

Communication functions.....	108
Library functions.....	111
System events.....	114
Specifications	
ezeio™	117
Configuration and programming.....	118
Server Communication.....	118
Warranty	
Manufacturers warranty statement.....	119
Liability disclaimer.....	119
Standards compliance	
Applicable standards.....	120

Introduction

Thank you for purchasing the ezeio™!

What is the ezeio™ ?

The ezeio™ is a complete solution for monitoring, alarming, control and automation of commercial and industrial equipment.

The ezeio™ hardware connects to sensors, meters, thermostats, VFD's and other control devices locally via a number of industry standard interfaces.

It connects securely and seamlessly via the Internet (Ethernet or Cellular) to the ezecontrol.com cloud application, where the user can access all data in real time as well as historical.

All configuration settings and programming is done via the cloud interface, allowing multiple concurrent users, automatic synchronization and secure access from anywhere without any special software or setup.

Common applications include:

- + Monitoring energy meters (electrical, water, gas)
- + M&V applications (energy saving, improvements)
- + Monitoring refrigeration systems (temperature, pressure)
- + Controlling & monitoring HVAC systems (thermostats, room sensors)
- + Construction site monitoring (cement curing, heating/cooling, alarms)
- + Automating thermal energy storage systems (*TES)
- + Technical alarm systems (fan monitors, temperature, tank levels)
- + Lighting control, monitoring and scheduling/automation
- + Sprinkler control, monitoring and scheduling/automation
- + Battery / EV charging, monitoring and control
- + Vehicle tracking, monitoring (GPS)

The ezeio™ system is designed for easy deployment in geographically spread out, multi-dicipline applications where traditionally several single-purpose systems were needed.

The ability to support different kinds of sensors, meters, actuators and applications within a single, low cost yet complete and secure system makes the ezeio™ system unique.

Model information

The ezeio™ is available in the following configurations:

Part Number	Model	Wireless sensors	Cellular GSM/3G/GPS	Cellular Compatibility
111-0010-3	ezeio-Ethernet	-	-	-
111-0020-3	ezeio-Wireless	YES	-	-
111-0011-3	ezeio-Cellular	-	YES	US
112-0011-3				EU/AU
111-0021-3	ezeio-Wireless & Cellular	YES	YES	US
112-0021-3				EU/AU

Base model

The ezeio™ connects to the Internet via standard 10/100 Ethernet and uses wired peripherals via Modbus, MicroLAN and general purpose in/outputs.

Wireless sensors

ezeio™ models equipped with a wireless transceiver module allow for communication with wireless sensors and expansion units. The wireless protocol is encrypted and only wireless devices from eze System can communicate with the ezeio™ over this network. Typical indoor range is about 50m (160ft) but depends on wall material and other environmental factors.

GSM/3G/GPS

When configured with a built-in GSM Cellular modem, the ezeio™ can communicate with the Internet via cell service. This requires service from a local cell provider and only GSM systems are supported. The ezeio™ will use the physical Ethernet path if it is available, but automatically switches to GSM if it can't communicate over Ethernet.

The GSM modem also supports GPS, so with the addition of an external antenna, the ezeio™ will have access to its position in real time.

Common features

All versions run the same software and all other features are the same.

Some features depend on the service level. All versions come with four (4) months of Basic Service, which allows for logging data from five inputs. See our web page at www.ezesys.com for all the details about service levels and the monthly cost.

Script support (page 88) can be added to any ezeio™. Please contact eze System for more information.

Creating accounts and users

Overview

To configure your ezeio™, you need to create an account on the server and associate the ezeio™ with that account.

This is important to understand to effectively manage your systems:

- (!!) Each ezeio™ is associated with a single account.
- (!!) Each user is associated with a single account.
- (!!) An account may have any number of ezeio™ units and any number of users associated with it.

Note that you can register several ezeio™ units under the same account. This allows you to access all of them from a single login. See Add an ezeio to an existing account, page 10.

All users on the same account will have access to all the ezeio units on that account. If you prefer to separate user access, simply create unique accounts for each ezeio. There is no per-account cost, but having a lot of accounts may become difficult to manage.

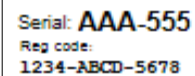
Creating a new account

Go to www.ezecontrol.com → Click **Create a new account**

Step 1:

The system will ask you for the ezeio serial number, and the registration code. These are printed on a sticker located on the front of your ezeio.

Enter them exactly as they show on the sticker.



Serial: AAA-555
Reg code:
1234-ABCD-5678

Step 2:

Enter your user info:

Name, Email, Phone & Company or Account Name

Step 3:

Enter a User Name & Password.

Click to send the verification email.

Step 4:

Open the email, click the link, and enter your password. You're done!

In addition to your login name and password, the system assigns you an account ID. You will need the account ID every time you log in. Make sure to take a note of it. It is also included in the confirmation email.

Add an ezeio to an existing account

Log in to your account and click the **Configure** tab.

In the Account section in the left column, click the **Add controller** button and enter the serial number and registration code.

The ezeio™ will be immediately added to your account.

If you receive an error, the ezeio™ may already be assigned to a different account. See below for how to remove or move an ezeio™.

Adding users to an existing account

To minimize the workload on the account owner, users register themselves.

You (the administrator or “admin”) need to provide each new user with the serial number and the registration code to one of the ezeios on your account. It doesn't matter which ezeio you use. The information is just used to link the user to the correct account.

Instruct the new user to go to the web page and click the **Create a new account** link. Then enter the ezeio serial number and registration code.

Then follow the rest of the sign-up instructions.

The process is basically the same as creating a new account. See page 9.

This will automatically link the new user with your existing account, and you will receive an email informing you that a new user has been added.

By default, new users have minimal privileges. You can log in and change the privileges for each user by going to Account → Users and click on the user in the list.

Moving or removing an ezeio from an account

As mentioned above, an ezeio can only be added to an account if it is not already assigned to an account. To remove an ezeio from an account, go to Configure → System and click **Delete Controller**.

This will remove the association with the account. The configuration of the ezeio is not changed, and any data stored with the ezeio is kept.

Creating the account and adding users does not require the ezeio to be connected to the network.

Configuration can also be done with the ezeio offline. See page 36.

Connections and installation

Things to consider before installing the ezeio™

The ezeio™ is designed for indoor use and should be installed in a dry and clean location. Do not expose the ezeio™ to rain or water, and avoid extreme temperatures. See the technical specifications for acceptable ranges.

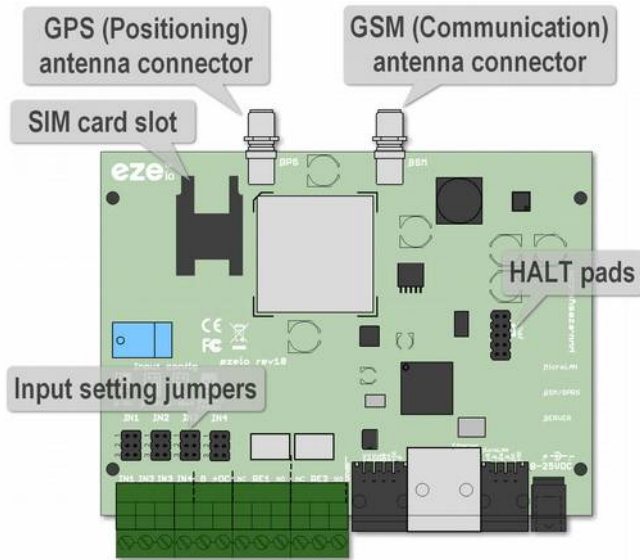
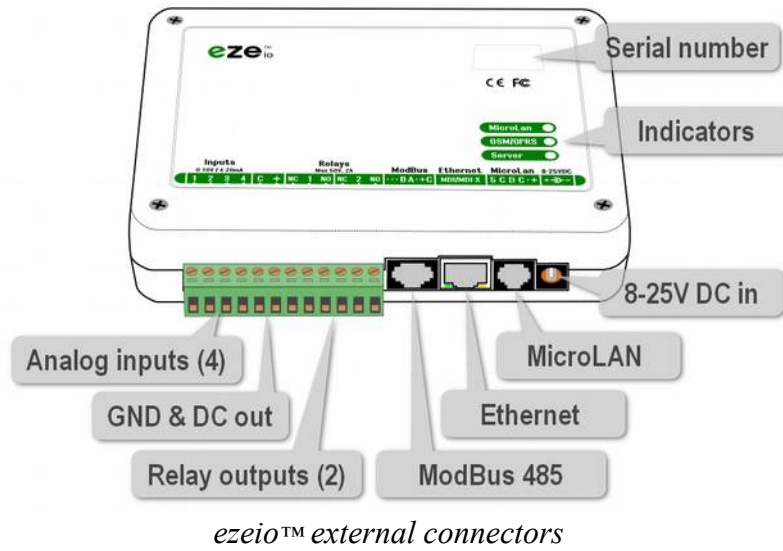
The ezeio™ is a low voltage device. Never connect high voltage to the inputs or outputs, and only use the supplied AC adapter to power the ezeio™.

Do not run wires that connects to the ezeio™ (Inputs / outputs / MicroLAN / Ethernet / Modbus / power or antenna) together with high voltage wiring. Use separate conduits whenever possible, and avoid environments with excessive RF or magnetic radiation as this may interfere or even destroy the ezeio™.

Take necessary precautions to avoid large static discharges to the ezeio™ connections.

MicroLAN: Never use connectors or wires designed for phone networks to connect MicroLAN devices. **Phone connectors** usually alter the polarity, and **will permanently damage** MicroLAN devices, voiding the warranty.

ezeio™ overview



*Internal features
(SIM and antenna connectors only on GSM model)*



*Basic communication setup
(Antennas only on GSM model)*

Power connection

Use the included AC adapter to connect to mains power.

The ezeio™ does not have an on/off switch, so as soon as the AC adapter is connected, the ezeio™ will operate.

Typical power consumption is very low (<2W), and the ezeio™ is designed to be always on.

Network connection

The ezeio™ has a standard TP 10/100 Ethernet connection. Use the included network cable to connect to a nearby Ethernet hub/switch/router that provides a link to the public Internet. If you use your own Ethernet cable, ensure the length does not exceed 30m (100ft).

There are two jacks on the ezeio™ where an Ethernet cable fits. Take care connecting your Ethernet cable to the one with a metal frame, marked "Ethernet" (NOT the Modbus jack).

All communication parameters are pre-programmed in the ezeio™, so there is nothing to set up. The ezeio™ will automatically contact the servers.

The ezeio™ automatically establishes IP information through DHCP. Ensure your network connection supports DHCP and that the DHCP server provides valid gateway and DNS information.

For static IP setup see page 14.

The green LED on the Ethernet jack lights up as soon as there is a physical connection available.

Check the SERVER LED on the ezeio™ front for connection status:

Blink pattern	Meaning
5 blinks	Looking for DHCP address information
4 blinks	IP address established Querying DNS server for server IP
3 blinks	Server address established Attempting to make contact with server
2 blinks	Communicating with server
1 blink	Server connection established and idle

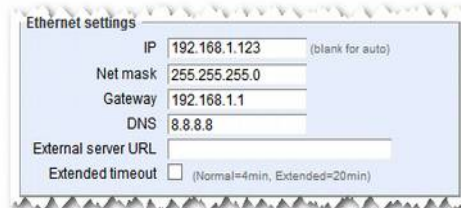
Setting a static (fixed/manual) IP

The ezeio™ by default expect a DHCP service to provide the correct network settings for the device when it is connected to the network.

In some installations, it is required to supply specific network settings manually to each device to allow it to communicate on the network.

However this change can only be made from the system settings on ezecontrol.com. Thus, the ezeio™ needs to be connected to a network that supports DHCP temporarily.

To apply manual network settings, navigate to the Configure → System screen and enter IP addresses in the Ethernet settings section (see example to the right).



Please make sure all settings are correct before applying them. Incorrect settings will cause communication to fail. As soon as the settings are synced the ezeio™ will loose contact with the servers via the DHCP network. It can now be moved to the network that requires fixed IP.

Restoring DHCP functionality

To restore DHCP support, first remove the fixed IP settings on the server by blanking out the IP field and click Save Changes. If the ezeio™ is still communicating via fixed IP, the settings will be automatically synced and applied after the next reset of the ezeio™.

If the ezeio™ is not communicating, the reset procedure is as follows:

- 1) Remove power from the ezeio
- 2) Open the enclosure and locate the holes marked HALT (see picture to the right)
- 3) Apply a jumper between the two holes and make sure it stays in place, connecting the holes.
- 4) Connect power to the ezeio. keeping the jumper in place.



The LED's will blink very fast, and the ezeio™ will attempt to use DHCP to connect to the servers.





Verify that the ezeio™ connects by checking the status on ezecontrol.com.

When the “spinner” indicator stops and shows a green dot, the new configuration is saved. Power down and remove the jumper.

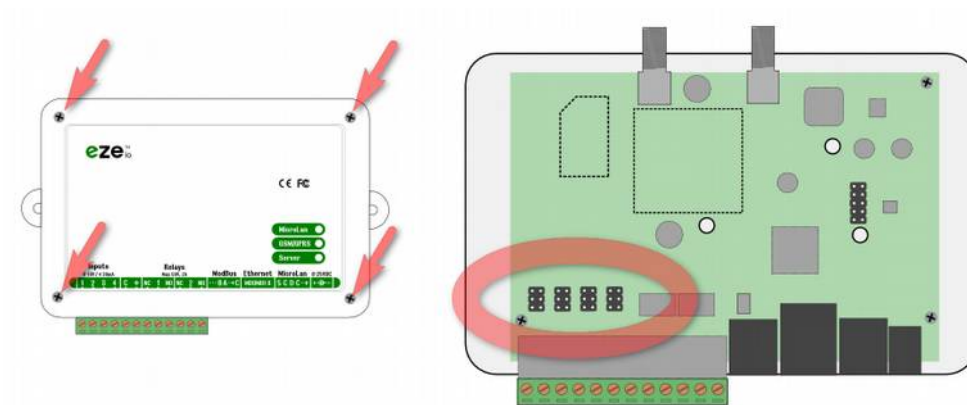
The ezeio™ is now back in default DHCP mode.

General purpose inputs


The ezeio™ has four general purpose inputs. Each input may be configured individually in one of four ways described here:

Jumper setting	Description	
	The factory default setting is Contact, Pulse or Resistive (0-50kOhm). An internal 10k resistor will hold the input to 5V.	See pg. 16
	0-5VDC Input impedance is >70kOhm. Raw reading is about 10000 at 5.0V (0.5mV per count)	See pg. 17
	0-10VDC Input impedance is >70kOhm. Raw reading is about 10000 at 10.0V (1mV per count)	
	0-30mA (suitable for 4-20mA transducers) An internal 100 Ohm resistor connects the input terminal to Common. Raw reading is about 10000 at 30mA (3uA per count)	See pg. 18

To access the input jumper settings, open the ezeio™ by removing the four black screws.

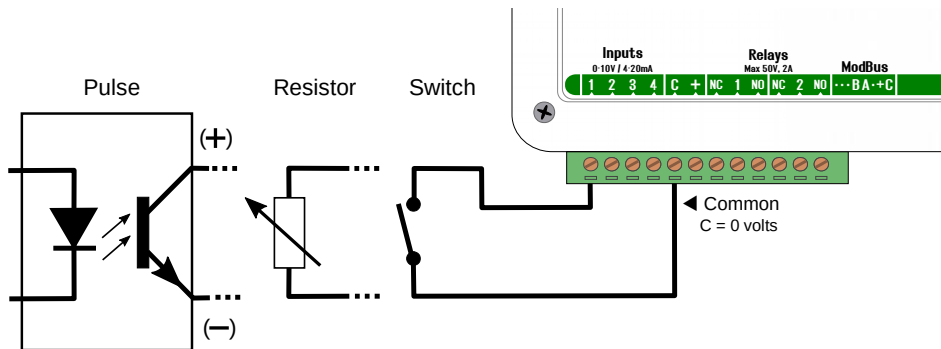


Inputs – Pulse, switch or resistive sensors

Jumper setting	Description
	Contact, Pulse or Resistive (0-50kOhm). An internal 10k resistor will hold the input to 5V. <i>This is the factory default setting.</i>

The default input configuration is suitable for connecting passive sensors, such as a switch, pulse meter or resistor.

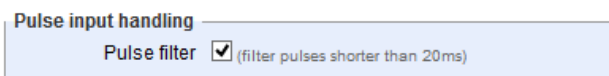
With the jumper settings configured to Pulse/resist, (as shown above), the input has a 10kΩ pull-up resistor to +5V, allowing for variable-resistance devices or switches to be connected directly to the “C” (0 volt) terminal and one of the four input terminals like this:





If connecting to a pulse meter output, please make sure the polarity matches the meter's. The C terminal on the ezeio™ should be connected to the meter's minus (-) or ground.

Three wire KYZ sensors (Form C) are read as two wire sensors (Form A); connect K to the ezeio's C-terminal, and Y to one of the ezeio™ inputs.

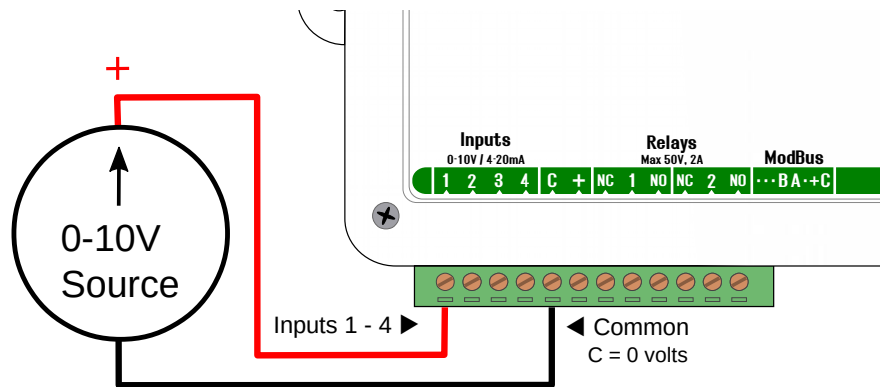
The ezeio™ input will detect pulses as short as 1ms. Some pulse outputs may have contact bounce, and requires a special setting in the software to ignore pulses that are too short. See System Settings.



Inputs – External voltage sources

Jumper setting	Description
	0-5VDC Input impedance is >70kOhm. Raw reading is about 10000 at 5.0V (0.5mV per count)
	0-10VDC Input impedance is >70kOhm. Raw reading is about 10000 at 10.0V (1mV per count)


The 0-10V and 0-5V input settings are suitable for sensors with output voltage in that respective range. Simply connect the sensor to the input between the “C” (0 volt) terminal and the input as shown here:



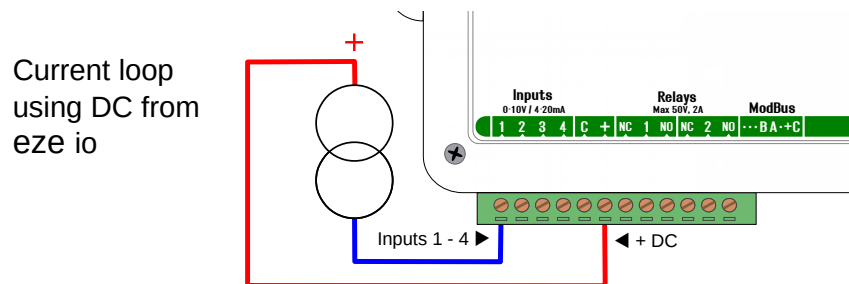
The ezeio™ is designed for low voltage connections. Never connect high voltage to the ezeio™ inputs.

A series resistor can be added to increase the range of an input. Use a 100kOhm resistor to allow measuring up to 20V. Use a 390kOhm resistor to allow measuring up to 50V. Contact eze System if you need to measure higher voltage than 50V.

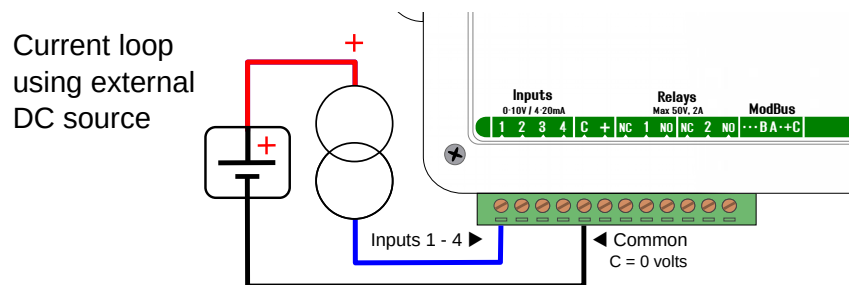
Inputs – Current sensors

Jumper setting	Description
	0-30mA (suitable for 4-20mA transducers) An internal 100 Ohm resistor connects the input terminal to Common. Raw reading is about 10000 at 30mA (3uA per count)

To use 0-30mA (or 4-20mA) sensors, after setting the input jumper as explained above, connect the current loop sensor between the +DC output terminal and the input, like this:



The +DC output provides nominally 1V less than the voltage on the DC input. The standard DC adapter shipped with the ezeio™ outputs just over 12V. If a higher voltage is required for the current loop sensor, either use a different adapter for the ezeio™, or feed the current loop from an external source, like this:



The internal current sense resistor in the ezeio™ is 100Ω, so the loop voltage drop at 20mA will be 2V ($U = R \cdot I$).

Check the data sheet for your current sensor and make sure the voltage source is at least 2V higher than the minimum voltage for your sensor.

Relay outputs

There are two relay outputs on the ezeio™. Each output has three screw terminals;

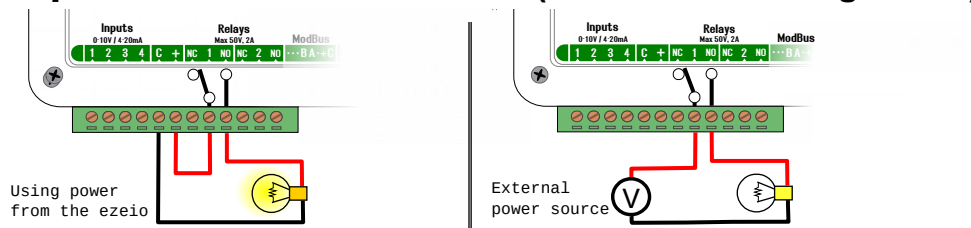
NC (Normally Closed)

RE (center)

NO (Normally Open)

The RE terminal is connected to NC when the relay is not energized, and to NO when the relay is energized.

Examples of how to connect a load (here shown as a light bulb):



The relays are rated 50V and 2A. Higher voltage or current can cause permanent damage to the relays.

The relay outputs are dry contacts. If the load is inductive, please apply appropriate protection such as spark inhibitor and/or flyback diode.

+ DC output terminal

The **+** output terminal can be used to power external sensors or relays. The voltage on this terminal is nominally 1V lower than the input voltage on the DC input jack. The + DC output can supply up to 200mA.

MicroLAN

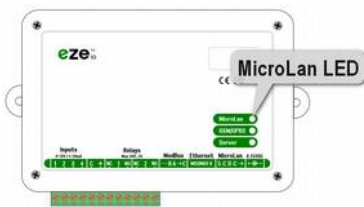
MicroLAN allows you to connect temperature and RH sensors to the ezeio™. The system automatically detects the type of sensor connected, and adds it to the configuration.

The ezeio™ supports up to 20 devices connected to the MicroLan connector.

Only MicroLAN devices supplied by eze System will work with the ezeio™. Supported sensors are automatically detected and added to the configuration.

MicroLAN indicator

The MicroLan LED indicates the status of the MicroLan device communications according to this table:



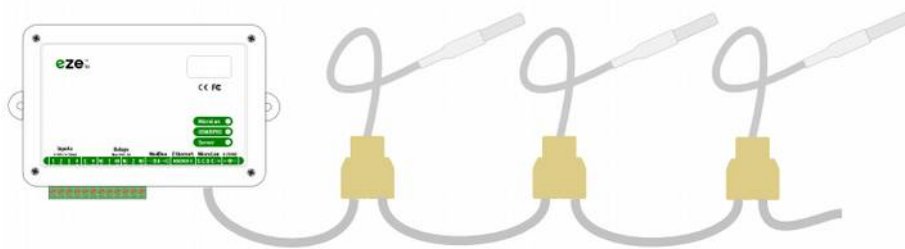
Blink pattern	Meaning
off	No devices detected and no devices expected
on	Initializing
fast flash	Searching for new devices
slow flash	Devices configured/expected, but none communicating
2-flash	Communicating, but at least one device missing
1-flash	Communicating with all devices

Connecting a MicroLAN device

To add a MicroLAN device to the ezeio™, remove power from the ezeio™, connect the new sensor and power up the unit. The ezeio™ will automatically detect the type of sensor and add it to configuration.

See ezeio™ Configuration on page 37 for more information.

To connect multiple MicroLAN devices, use MicroLAN splitters and extension cables as illustrated below.



Do not exceed 150ft (50m) total wire length on the MicroLAN network.

MicroLAN connector pinout

The MicroLAN connector looks similar to phone connectors, but note that MicroLAN is using all six conductors, while phone networks typically only use four.

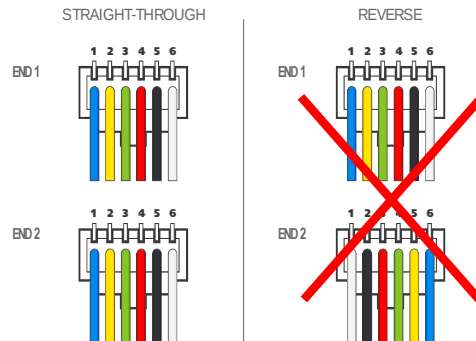
Never use connectors or wires designed for phone networks to connect MicroLAN devices. **Phone connectors** usually alter the polarity, and **will permanently damage MicroLAN devices**, voiding the warranty.



RJ12 pin	Signal	Description
1	+5V	+5V DC out, max 100mA
2	G	Signal ground
3	Data	1-wire data (bidirectional)
4	G	Signal ground
5	n/c	not connected
6	DC+	8-25V DC out, max 200mA

MicroLAN extension cables

Twisted pair cable, such as Cat3 or Cat5, AWG22-24, is recommended to ensure signal quality. Cables **must be terminated with straight through pinout** (see diagram below).



Do not exceed 50m (150ft) total wire length on the MicroLAN network.

Modbus / serial port

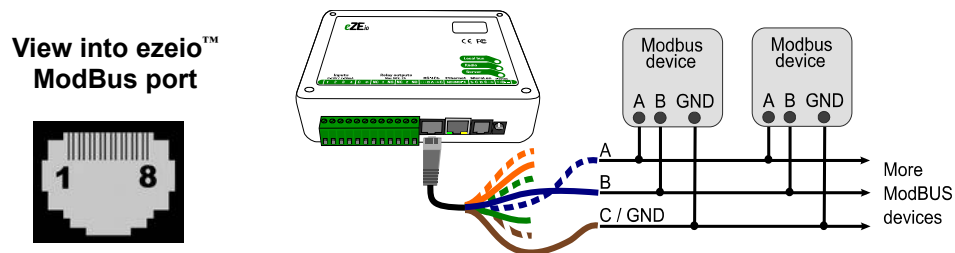
The ezeio™ has one Modbus RTU master port.

The communication default settings are **19200bsp, 8 data bits, No parity**.

The bit rate can be changed from the **Configure**→**System** screen.

Up to 20 Modbus devices (slaves) can be connected to the same Modbus RTU network, and the network can be up to 1000m (3000ft) long. Note that the wire length may in some cases be limited by the device specifications and environmental factors.

RS-485/Modbus RTU port pinout



Pin on RJ45	Common nomenclature	EIA/TIA-485 name	Description	T568A/B color
4	D1/D+	B/B'	Data 1, V1 Voltage	Blue
5	D0/D-	A/A'	Data 0, V0 Voltage	Blue/white
7	VP		8-25V DC out, max 200mA	Brown/white
8	Common	C/C'	Signal/power supply common	Brown

This pin arrangement conforms to the Modbus specification, 2W-MODBUS (see www.modbus.org). Pins not listed above are not connected.

Standard Ethernet patch cables are suitable for extending the Modbus signals. Make sure your cables conform to the T568A/B standard. For long runs (>100m/300ft) and where interference may be an issue, please consider using shielded cable and end of line resistors.

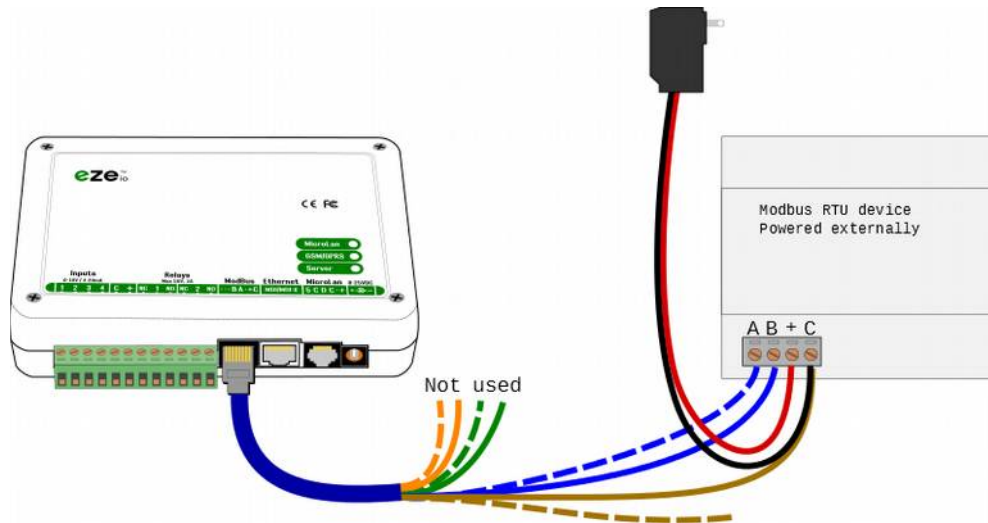
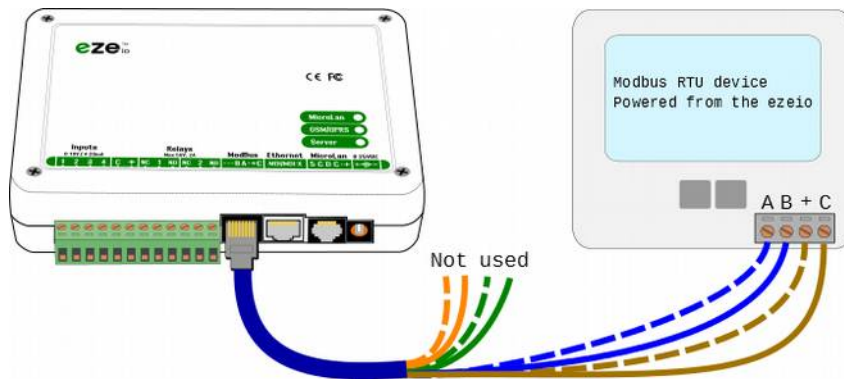
We recommend using the color scheme described in the above table.

Connecting a Modbus device

Each Modbus device will need to be configured to a unique bus address and for the same communication settings as the ezeio™. Refer to the device manual for details on how to configure the device.

In some cases the device can be powered from the ezeio™. In other cases the device needs a separate power supply. Refer to the device manual for details.

The ezeio™ needs to be configured with the device address to communicate with the new device. Refer to Configuring Modbus Devices, page 52 for details.



Always run the Common/Ground wire to all devices on the bus.

Note that some devices have other markings for A/B, such as "D0/D1" or "+/-". Do not confuse bus +/- with power terminals. Connecting power to A/B terminals will likely damage the equipment.

About Modbus

Modbus is a very common protocol used in data acquisition and control applications. Note however that just because some device supports Modbus, it doesn't mean it automatically works with any other Modbus device.

The ezeio has support for over 50 different types of devices. Contact us for the complete list and to explore adding support to devices not already on the list.

Every Modbus network requires one 'master' and one or more 'slave' devices. The master device will send out periodic questions, and the slave devices will reply to those questions. Slave devices must never transmit unless specifically requested by the master.

The ezeio is always a Modbus RTU master.

The Modbus protocol can be transported over several different physical networks. The most common are RS-485 and Ethernet.

Modbus RTU (a.k.a. Modbus RS-485) is a robust multi-drop serial network. This means that all the devices connected to the network communicate over the same pair of wires. The signal rate is typically 19200 bits per second, which is usually more than enough for the types of devices used in ezeio applications.

Modbus TCP/IP use Ethernet hardware to connect the devices. The data rate is much higher than on RS-485, but that also limits the range (typically to 100m/300ft). Ethernet uses switches to which all devices need to be connected (star topology).

While the fundamental protocol is the same over Modbus RTU and TCP/IP, the IP network requires additional consideration to have the data correctly routed and correctly addressed. In common data acquisition and control applications, Modbus TCP does not provide any functional benefit, but adds significant complexity to the setup process.

The ezeio supports Modbus RTU.

While Modbus TCP is not supported by the ezeio™, there are protocol converters for Modbus TCP to Modbus RTU. Contact eze System for details.

GSM/3G/GPS module (select models only)

ezeio™ models equipped with a GSM/3G/GPS module are capable of communicating with the Internet via mobile cell service.

The GSM/3G signal is used to communicate with the server if the Ethernet connection is not available. The switch between Ethernet and GSM is automatic.

When the Ethernet connection is available, the ezeio™ automatically communicates via the wire. If the Ethernet connection is not usable, the ezeio™ uses the cellular service to connect to the servers.

A valid SIM card with data service is required to use the GSM connection.

eze System can provide SIM and cellular service in some areas. Contact us for details.

Always disconnect power from the ezeio™ when installing or removing SIM card or antennas.
Do not power up the ezeio™ without a cellular antenna connected.

Inserting the SIM card

To insert the SIM card in the holder inside the ezeio™, remove the four screws retaining the enclosure cover, and slide in the SIM card in the holder, (as shown below).



Slide the metal latch down to release hinged lid.



Lift the hinged lid up.



Insert SIM card into the lid.
Orient cut corner top left.



Push down and lock by sliding the metal latch back up.

Attaching the GSM (communication) antenna

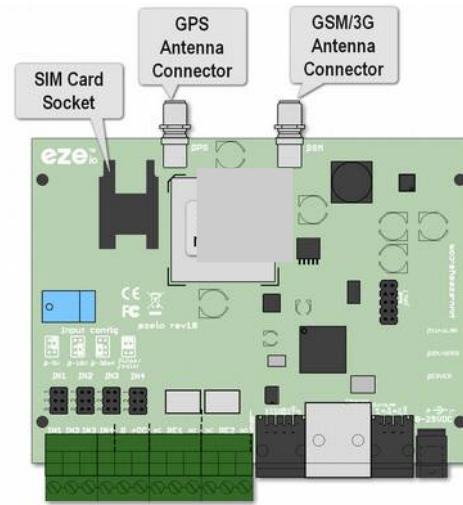
The supplied GSM antenna should be mounted on the **right**-hand antenna connector. Mount the ezeio™ with the antenna in a vertical position for best reception.

Always disconnect power from the ezeio™ when installing or removing SIM card or antennas.
Do not power up the ezeio™ without a cellular antenna connected.

GSM service

Your GSM service must allow data connectivity. The ezeio™ will only use cellular data. It does not use voice minutes or text messages.

Typical data usage for a full month is about 7-15 MB, but may vary depending on how frequently logging data is captured and other configuration parameters.



Make sure the cellular antenna is connected in the connector to the **right**. Do not use tools to tighten the antenna nut. It only needs to be finger tight.

GPS (positioning) antenna

The optional GPS antenna (purchased separate) connects to the left antenna connector. Make sure the GPS antenna has a clear view of the sky.

GSM Settings

In some cases, depending on your wireless carrier, you may need to enter the GPRS APN, GPRS Login and GPRS Password on the system configurations screen. These settings are different depending on your wireless carrier. You should have received this information with your SIM card if they are required.

Note that these settings have to be downloaded into your ezeio™ **before** the GSM will work. The ezeio™ must connect through the Ethernet port to a working network before you insert the SIM card.

GSM (Cellular) indicator

The GSM LED indicates the status of the cell radio as described in the table below.



Blink pattern	Meaning
off	GSM radio is turned off
on	Waiting for the GSM module to switch on
5 on-blink*	Attempting to initialize GSM module
4 on-blink*	GSM module requested SIM-PIN.
3 on-blink*	Module active. Waiting for GPRS network.
2 on-blink*	GPRS network ok. Establishing IP connection.
1 on-blink*	Server link dropped. Reinitializing.
Normal blinks	1-5 blinks. Reception quality (e.g. 1-5 “bars” on a cell phone)

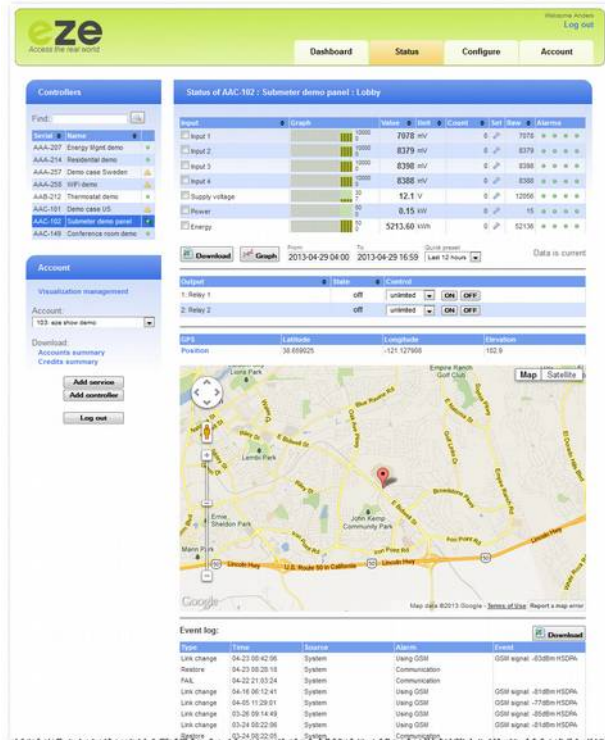
* “on-blink” refers to that the LED is on most of the time and pulses off.

GPS position

The GSM/3G transceiver module has built-in support for satellite GPS positioning, but requires a separate antenna (not included, visit our website for details)

To use the GPS feature, attach the GPS antenna to the **left** antenna connector and mount the antenna with a clear view of the sky. The ezeio™ may need to be restarted to recognize the GPS antenna. It may take a few minutes for the ezeio™ to find enough satellites and lock in an accurate position.

The GPS position will be available on the status page.



The GPS position is mapped as inputs on the ezeio™ as follows:

Input	Description
Input #7	GPS Latitude, degrees * 1 000 000
Input #8	GPS Longitude, degrees * 1 000 000
Input #9	GPS Elevation, meters * 10

Web interface overview

When the ezeio™ is online (Server LED flashing once every few seconds), the data from that ezeio™ is directly available from the web.

The web interface can be accessed even if the ezeio™ is not online, but only historical data will be available, and any changes to the configuration will be saved and committed to the ezeio™ once it's back online.

Logging in

Go to www.ezecontrol.com and log in to your account. You need your account number, user name and password.



There are four main sections of the web interface:

Dashboard – a configurable overview of all ezeio™ units on the account

Status – full live status of one ezeio™ at a time

Configure – settings for each ezeio™ unit

Account – account and user settings

Access to the individual features on the web site is controlled by the user privilege settings and the service level of the ezeio™. For example, the configuration tab is not visible for users with minimal privileges.

Access to the web page is secured with SSL. If your browser or IT policy does not allow this, the system can be accessed without encryption, but is **not recommended**, using: www.ezecontrol.com?insecure

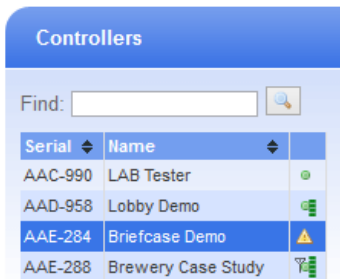
There are no restrictions on the number of simultaneous users or on using the same user login from multiple computers. Each user will be logged out after 60 min of inactivity.

Controllers Panel

The functions in next two sections of the web interface, **Status** and **Configuration**, are ezeio™ specific.

Click on **Status** or **Configuration**. Use the **Controllers** panel on the left to select the ezeio™ unit you want to view or configure.

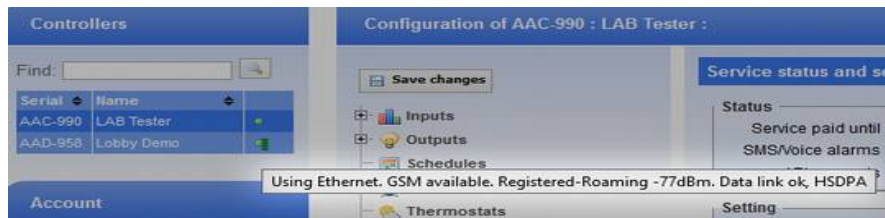
The table displays the Serial number, Name and the communication status:



Serial	Name	Status
AAC-990	LAB Tester	Green dot
AAD-958	Lobby Demo	Green dot with bar(s)
AAE-284	Briefcase Demo	Warning triangle
AAE-288	Brewery Case Study	Antenna, green dot and bar(s)

← Green dot : Online via Ethernet
← Dot with bar(s) : Online via Ethernet & Cellular link is ready
← Warning triangle : Offline
← Antenna, green dot and bar(s) : Online via cellular

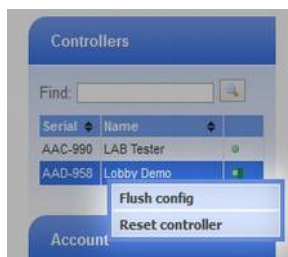
Hovering over the icon will give you addition information.



Configuration of AAC-990 : LAB Tester :

Using Ethernet. GSM available. Registered-Roaming -77dBm. Data link ok, HSDPA

Right click the ezeio™ entry to access the drop-down menu:



Flush config
Reset controller

Flush Config will download all configuration into the ezeio™ unit. This may be useful if a lot of configuration changes has been made to make sure all settings are aligned between the servers and the ezeio™.

Reset Controller will cause the ezeio™ unit to reboot. It does not change any configuration settings or logged data.

Dashboard screen

Dashboards

After logging in, the Dashboard is always shown first. Here you can graphically display data from any ezeios on the account.



There are two Dashboard views; Personal and Account.

The Personal view is only accessible by the logged in user, while the Account view is common for all users on the same account. Only users with privilege to change account information can alter the Account view.

Each Dashboard consists of configurable blocks, called “widgets”. There are many widget types to choose from, and each user may set up the widgets to his/her liking. Click on **Add widget** in the upper left of the screen and select from the options in the dialog box.

The widgets can be positioned on the Dashboard screen by dragging the blue header.

The configuration screen for each widget is accessed by clicking the small wrench-symbol in the blue header.

Status screen

The Status screen shows the live status for a single ezeio™ at a time. The available ezeios are listed in the left on the screen and their online status is shown as a green dot if online with an Ethernet connection or a warning triangle if offline.

In addition to the green dot, GSM equipped ezeios will show an antenna symbol and signal strength in green bars when they are connected by GSM.

The screenshot displays the 'Status of AAA-157 : Lighthouse demo : eZE System' interface. On the left, there are sections for 'Controllers' (with a search bar and a table listing 'AAA-157 Lighthouse demo'), 'Account' (with a welcome message and account details), and 'Download' (with a summary link). The main area shows a table of 'Input' parameters with graphs, values, units, counts, and alarm status. Below this is an 'Output' table with state and control options. At the bottom, there is an 'Event log' table with columns for Type, Time, Source, Alarm, and Event.

Input	Graph	Value	Unit	Count	Set	Raw	Alarms
<input type="checkbox"/> Motion Sensor		32	mV	2		32	
<input type="checkbox"/> Input 2		2850	mV	2		2850	
<input type="checkbox"/> Input 3		2860	mV	2		2860	
<input type="checkbox"/> Lighthouse power		0.0	kW	0		32	
<input type="checkbox"/> Supply voltage		11.9	V	0		11859	
<input checked="" type="checkbox"/> Lighthouse indoor temperature		74.3	F	0		1256	






Output	State	Control
1: Tower motor	off	unlimited <input type="button" value="ON"/> <input type="button" value="OFF"/>
2: Tower lamp	off	unlimited <input type="button" value="ON"/> <input type="button" value="OFF"/>
3: House lights	off	unlimited <input type="button" value="ON"/> <input type="button" value="OFF"/>
4: Floodlights	off	unlimited <input type="button" value="ON"/> <input type="button" value="OFF"/>

Type	Time	Source	Alarm	Event
ALARM	08-13 13:30:49	Input 1: Motion+Sensor	1: Motion+detector	1: Motion+detected
ALARM	08-13 13:07:17	Input 1: Motion+Sensor	1: Motion+detector	1: Motion+detected
ALARM	08-13 12:55:49	Input 1: Motion+Sensor	1: Motion+detector	1: Motion+detected

Even if the ezeio™ is online, it may take a few seconds for the status screen to refresh with live data.

Live input status

All configured inputs of the ezeio™ are listed in the Inputs table.

Input	Graph	Value	Unit	Count	Set	Raw	Alarms
<input type="checkbox"/> Microlan temp sensor		74.5	F	33		1258	  

The **Graph** column shows a rough bar graph of the last minutes' data with the most recent data to the right.

The **Value** and **Unit** columns show the current converted value of the sensor input.

The **Count** column shows the number of pulse counts for the input. You may alter this value manually by clicking the wrench icon next to the counter.

The count value is stored in non volatile memory every 90 seconds and automatically restored on reset.

The **Raw** column shows the value from the hardware input before converting it to a real world unit.

The **Alarms** column shows the current status of the four possible alarm settings. Hover the mouse cursor over the symbols to see their meaning.

Output status and control

Output	State	Control
1: Out 1	off	unlimited  ON OFF

The outputs of the ezeio™ can be controlled directly with the **on/off** buttons in the output table. The drop-down box allows automatic shutoff after the selected time.

Thermostat status

If the ezeio™ is connected to one or more thermostats, they will automatically be listed on the status screen. The current temperature, set points, calls, override status and schedule settings are displayed and updated every few seconds.

Thermostat	Temp	Set H/C	Call	Override	Schedule	Mode	Pgm H/C
My thermostat	76.6	52.0 / 65.0		no	Base Schedule	Std	52.0 / 65.0

The temperatures are shown in the unit used in the schedule.

The override status shows the number of minutes left on the override cycle. If the number shown is negative, that means the override is a demand-response adjust.

The wrench-symbol in the leftmost column brings up a dialog box that allow for direct control over the demand-response feature.

As of this printing, the only supported thermostat is the T-32-P, available from eze System.

Event log

At the bottom of the screen, the ezeio™ event log is shown. Any recent events are temporarily highlighted.

Downloading log data

Select the input or inputs to download data from by checking the boxes, next to the input name, in the input list, and enter the desired time span in the **From/To** boxes or select from the quick presets, drop-down menu.

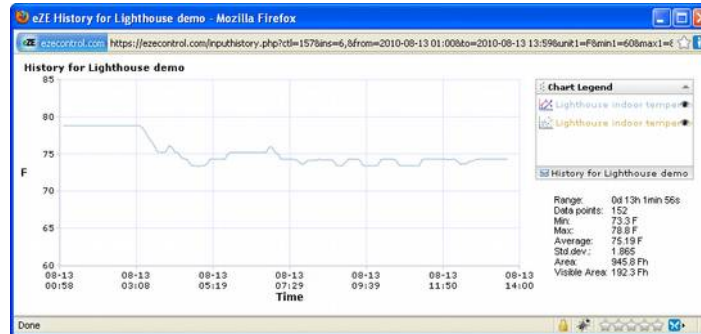
Click the **Download** button to start the download of a CSV (comma separated) file that may be opened in Excel, Calc and many other programs.

```
Time,"Power A (W)","Power A count","Power A alarm"  
2016-07-27 08:00:39,294.3,0,""  
2016-07-27 08:01:39,294.1,0,""  
2016-07-27 08:02:39,294.3,0,""  
2016-07-27 08:03:40,294.1,0,""  
2016-07-27 08:04:40,294.1,0,""  
2016-07-27 08:05:40,294.3,0,""  
2016-07-27 08:06:40,294.7,0,""  
2016-07-27 08:07:40,294.3,0,""
```

Viewing graph of log data

Select the input or inputs you want to graph by checking the boxes in the input list. Then enter the desired time span in the **From/To** boxes or select from the quick presets, drop-down menu.

Click the **Graph** button to view the graph.



The statistics to the right in the graph-window updates automatically if the graph view changes by zooming in/out.

Note that the "Area" and "Visible Area" values really only make sense if the sensor used is a power or flow sensor.

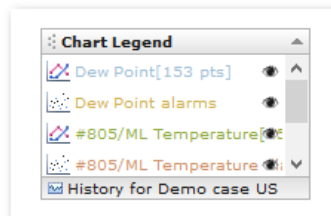
Controlling the graph

By hovering the mouse cursor over the graph, information about each sample will be shown.

The graph window allows zooming by highlighting a section with the mouse (drag from top-left to bottom right) or by using the mouse scroll wheel. Reset zoom by dragging right-to-left.

Panning is done by dragging using the right mouse button.

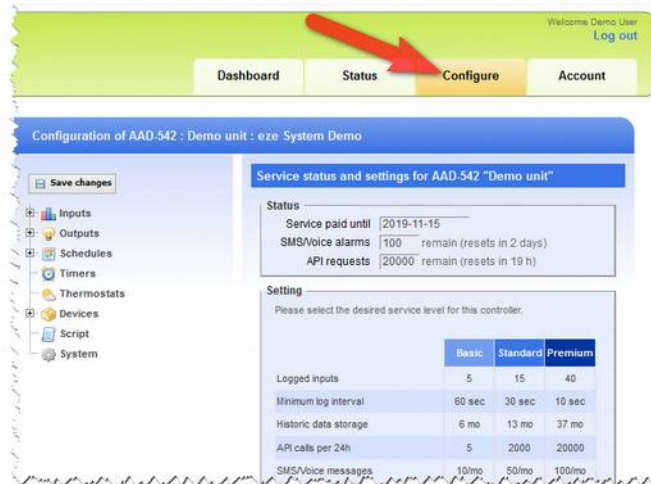
The legend allows turning on/off individual graphs by clicking on the "eye" icon in front of the name.



Configure screen

Service status & settings

With the ezeio™ selected in the left hand column, click the **Configure** tab to get to the **Service status and settings** screen:



Service status

The top section shows when the service for this ezeio™ unit will expire, the number of SMS/Voice alarms remaining this month, and the number of API requests remaining for the current 24h interval.

Service settings

If the service settings are not shown, your account is managed by your installer/reseller. Please contact the person you purchased the ezeio™ from.

In the settings box, you may select the level of service desired for this ezeio™. You may change the service level at any time, and the system will automatically pro-rate the expiration date based on the service time remaining.

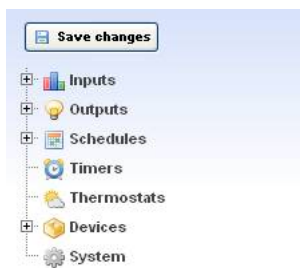
ezeio™ Configuration

The **Configure** screen allows access to all the configurable parameters of each ezeio™. Configuration can be done even if the ezeio™ is not accessible (off line). The changes are then committed as soon as the ezeio™ comes back on line.

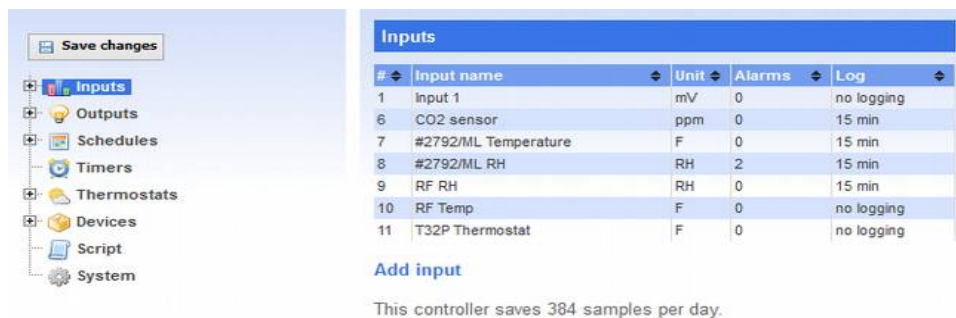
Click the **Configure** tab and then select the ezeio™ to configure in the table to the left.

Resource tree

In the center of the page, the resources of the ezeio™ are shown in a structured tree form. The tree can be expanded by clicking the plus icons.



Click the name of each object to see more information on that object.



#	Input name	Unit	Alarms	Log
1	Input 1	mV	0	no logging
6	CO2 sensor	ppm	0	15 min
7	#2792/ML Temperature	F	0	15 min
8	#2792/ML RH	RH	2	15 min
9	RF RH	RH	0	15 min
10	RF Temp	F	0	no logging
11	T32P Thermostat	F	0	no logging

[Add input](#)

This controller saves 384 samples per day.

Click on an item on the list to view or modify

Click on the **Add**-link to create new items

To commit changes, simply click the **Save changes** button. This will commit the changes to the database, and also synchronize the changes with the ezeio™. If the ezeio™ is temporarily unavailable, the changes will be transferred as soon as the ezeio™ communication is re-established.

Inputs

Each ezeio™ can support up to 40 inputs. Each input has the following settings:

Input name

A user defined name to identify the input.

Unit

The unit for the input, for example “Volt”, “kW” or “C”.

Decimals to show

The number of decimals to show when the converted value of this input is displayed. Valid range is 0 – 8.

Autoscale

If this checkbox is checked, the two following controls (Max/Min value in graphs) are ignored and the graph min/max values will be automatically chosen to fit the data. By default this is unchecked.

Max value in graphs

The maximum value on the vertical scale in graphs. Only relevant when autoscale is inactive.

Min value in graphs

The minimum value on the vertical scale in graphs. Only relevant when autoscale is inactive.

Display and Type/Conversion settings only affect the way the data is displayed. It does not change the raw stored data.

Input type

The type of this input. This defines how the raw value from the input will be converted to a user defined unit. A number of standard conversions are selectable from the drop-down list, others require manual entry of units and conversion equations. (see following page)

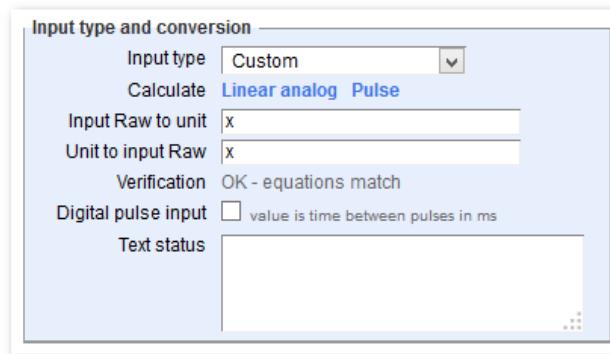
The screenshot shows the configuration window for 'Input 1'. It is divided into several sections:

- Input display settings:** Includes fields for 'Input name' (Input 1), 'Unit' (mV), 'Decimals to show' (0), 'Auto scale' (checkbox), 'Max value in graphs' (10000), and 'Min value in graphs' (0).
- Input type and conversion:** A dropdown menu for 'Input type' set to '0-10000mV'.
- Logging:** Includes 'Log interval' (5 min) and 'Import/Export' (none).
- Hardware/device setting:** 'Input location' dropdown set to 'eZE Controller, input#1'.
- Alarm setting summary:** A table with columns for 'Alarm name', 'Alarm', 'Restore', and 'Actions'. Below the table is an 'Add alarm' button.

A 'Delete this input' link is located at the bottom right of the window.

Input type – custom

If “Custom” is selected, more fields are displayed to allow customization; Raw-to-Unit, Unit-to-Raw, Digital Pulse and Text Status (See below).



Input Raw to unit

The math used to convert from the raw input value to a real world unit. The symbol “x” represents the unconverted input value in the equation.

DO NOT LEAVE THIS BOX BLANK. It has to contain valid math to provide a value to the input. If unsure, just enter “x” in this box.

The math boxes must to contain valid math expressions.
The input raw value is represented by a lower case ‘x’.

Unit to input Raw

The math used to convert from the real world unit back to the raw input value. This should be the *inverse function* of the Input Raw to Unit function.

Examples of inverse functions:

Input Raw to Unit	Unit to Input Raw
$x*10$	$x/10$
$x*5+32$	$(x-32)/5$
$5000/x$	$5000/x$

Simply speaking, if you take the output from the first function for any value of x, and plug that into the x of the second, you should end up with the value you started with.

If a “WARNING” text is displayed next to Verification, this means the second math is not the inverse function of the first. Please correct the math.

Raw reading / ADC resolution

The built-in ADC converter converts the voltage on the input to a number between 0 and 1023 (10 bit). This number is further scaled internally to 0-10000.

To calibrate the raw ADC reading to the desired unit, use a 2-point calibration and the helper feature in the web software.

See Calibrating analog inputs on page 44 for details

The tables below are average readings and can be used as a guide for calibration. This table only apply to the physical inputs on the ezeio™.

0-5V		0-10V		0-30mA		0-50kOhm *	
Input	Raw	Input	Raw	Input	Raw	Input	Raw
0.0V	0	0.0V	0	0.0mA	0	100Ω	107
2.5V	4882	5.0V	4927	4.0mA	1395	1kΩ	902
5.0V	9765	10.0V	9855	20.0mA	6975	10kΩ	4792

Note that when configured for resistance, the raw-to-Ohm relationship is not linear.

Other devices may use other scaling on the inputs.
Consult the device manual for details.

Digital pulse input

Normally the value of an input represents a voltage, current or resistance. If the sensor connected is of pulse type (such as a S0 pulse), the input can be defined as a **Digital Pulse Input** by enabling this check box.

When this box is checked, the input value will reflect the time between the two last transitions from low (<0.9V) to high (>1.1V) on the input. The time is presented as milliseconds, and range from 4 to 99999999 (~28h). If no pulses are detected, the max value will be returned.

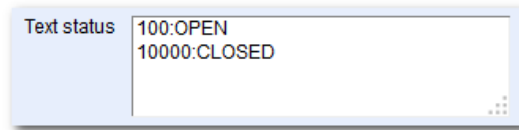
The shortest pulse that can be detected in this mode is 2ms.

When in Digital pulse input mode, the count register will automatically count up for each pulse.

The Digital Pulse input type is supported by the four inputs on the ezeio™ and the eze System “Wireless I/O Expander”.
Consult the device manual for other device types.

Text status

This allows displaying a text in addition to the value. For example, one may want to display “OPEN” and “CLOSED” based on the input value being low/high. This would be achieved like so:



Thus, if the value is 100 or less, the text “OPEN” will be displayed, and if the value is 101 up to 10000, the text “CLOSED” will be shown. Values larger than 10000 will show as blank.

Each line needs to have a number, a colon (:) and a text following the colon.

You may enter multiple text status lines. The numbers must be in increasing order. The program will search the list from top to bottom order. The first line with a number higher or equal to the current input value will be shown.

The text will be shown in the space for alarm status on the status screen, and in the widget “Live Input Alarm Table” on the dashboard.

Alert messages may also show this text by using the **#STAx#** designator in the message body (see Notes about the Message field page 61).

Log interval

This setting selects how often the value on the input is logged to the database. **The actual log value will be the average for the log interval.**

The screenshot shows two configuration panels. The top panel, titled 'Logging', contains two dropdown menus: 'Log interval' set to '5 min' and 'Import/Export' set to 'none'. The bottom panel, titled 'Hardware/device setting', contains one dropdown menu: 'Input location' set to 'eZE Controller, input#1'.

Input location

This defines the source from which the input receives its values. These can be from the ezeio™ unit, a ModBus device, a MicroLAN device or an eze System wireless expander.

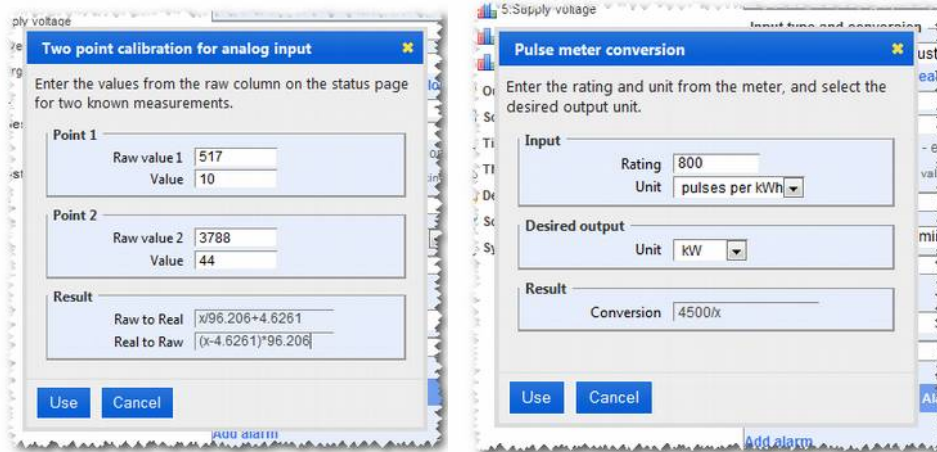
In addition to physical sensors or connected devices, input values can be software driven. The setting “Special/Software” allows script, external commands or API calls to set the value of the input.

Input locations for the ezeio™:

Source	Description
Special/Software	Script or API call
Input #1	ezeio, Input #1
Input #2	ezeio, Input #2
Input #3	ezeio, Input #3
Input #4	ezeio, Input #4
Input #5	Supply voltage (DC Power supplied to ezeio)
Input #6	CPU core temperature in Celsius (to 1 tenth degree)
Input #7	GPS Latitude, degrees * 1 000 000
Input #8	GPS Longitude, degrees * 1 000 000
Input #9	GPS Elevation, meters * 10

Input conversion helper dialogs (calibration)

The **Linear analog** and **Pulse** links are visible when the **Input type and conversion** is set to **custom**.



These will open conversion helper dialog boxes useful for calibration.

The Use key will copy the math from the dialog into the Input Raw to Unit and Unit to input Raw conversion fields. Don't forget to click Save Changes after using the Custom helper function.

Also see the section about calibration on the following page.

Calibrating analog inputs

With any type of sensor connected to an analog input, either directly to the ezeio™ or on one of the expansion modules, calibration may be necessary for accurate readings.

To calibrate the system you will need two reference points. Ideally these points should be well separated but within the expected operating range of the application.

In this example we will calibrate a current sensor, but the procedure is the same regardless of the type of sensor or unit.

1. Connect the sensor to the ezeio™ or the expansion unit and make sure we get a reading of the sensor on the status page in the **raw** column. Do not worry about setting up the correct type at this point. We will only need the raw reading.
2. Apply the first reference current to the sensor and let the sensor stabilize.



Current sensor and reference meter, 5.67A

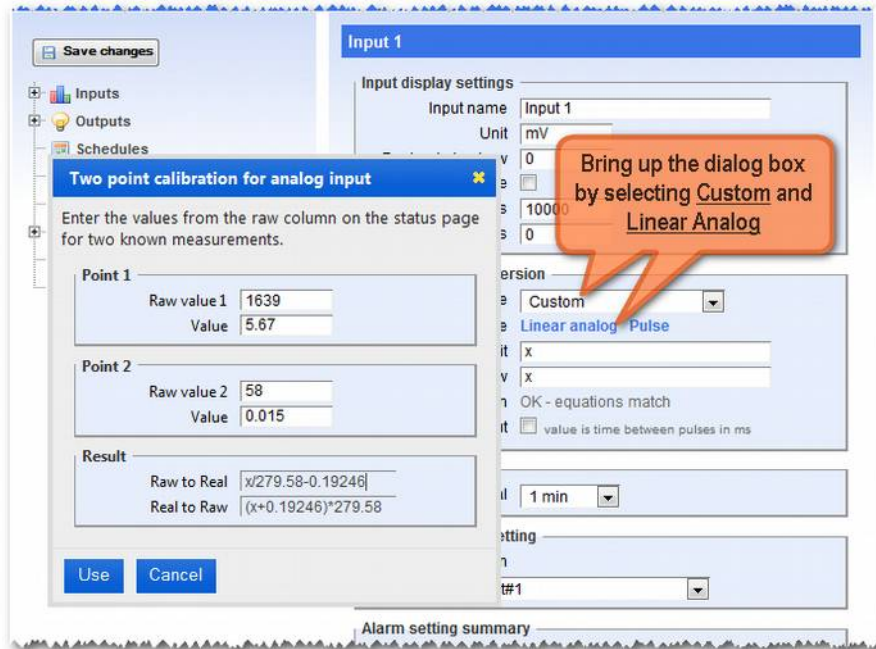
3. When stable, make note of the **raw** reading from the status page, and also what the reference current is, so in this example we got “1693 at 5.67A”

Input	Graph	Value	Unit	Count	Set	Raw	Alarms
<input type="checkbox"/> Input 1		1693	mV	0		1693	
<input type="checkbox"/> Input 2		8786	mV	0		8786	
<input type="checkbox"/> Input 3		8548	mV	0		8548	
<input type="checkbox"/> Input 4		8538	mV	0		8538	
<input type="checkbox"/> Supply voltage		14.4	V	0		14375	

Status screen, RAW column for input 1 showing “1693”

4. Apply the second reference current, let it stabilize and again note the RAW and the reference current. In our example, we read raw 58 at 0.015A.

We now have the data points we need. Navigate to the **Configure->Input** screen and select the input. In the **Input Type** box, select **Custom**. Then click the **Linear Analog** link to open the **Two point calibration** dialog:



5. In the dialog box, enter the raw and reference value currents in the Point1 and Point2 fields as seen in the image above. Click the Use button to apply the conversion to the input. Don't forget to click Save Changes to finish.

The input is now calibrated.

Alarm settings

For every input, up to four (4) alarms can be defined. Each alarm has the following settings:

Alarm name

A user defined name to identify the alarm.

Threshold for alarm

The threshold where the alarm should activate.

Alarm holdoff

Delay in seconds. The input value will need to exceed the threshold for this time interval for the alarm condition to occur.

Threshold for restore

The threshold where the alarm condition should clear.

Restore holdoff

Delay in seconds. The input value will need to exceed the threshold for this time interval for the alarm condition to clear.

If the alarm threshold is lower than the restore threshold, the alarm function is reversed. The alarm will activate when the input value is lower than the alarm threshold and restore when the input value exceeds the restore threshold.

If both thresholds are set to the same value, the alarm will never activate.

Actions

Actions describe what should happen when an alarm condition occurs or restores.

There may be up to four (4) actions on each alarm, and four (4) actions on each restore.

The settings for each action are as follows:

Action name

A user defined name for the action

Action type

Depending on the action type selected, different options are presented. See below.

The screenshot shows a configuration form for an action. It is divided into three sections: 'Action display settings', 'Action settings', and 'Conditions'.
- 'Action display settings': A text input field containing 'new alarm action'.
- 'Action settings': A dropdown menu currently set to 'no action'.
- 'Conditions': Two dropdown menus, both set to 'no condition'. Below them are two radio button options: 'both conditions are true (AND)' (which is selected) and 'either or both are true (OR)'.
A blue link labeled 'Delete this action' is located at the bottom right of the form.

List of possible actions

Action Type	Description	See Page
Send Message	Send a message to email, SMS, voice call, third party system and more	60
Log Event	Record an event in the event log	66
Set Output	Set or reset a local output	66
Set Counter	Set a counter to some specific value	66
Increment Counter	Count up a counter	66
Decrement Counter	Count down a counter	66
Control Thermostat	Adjust the set point or change operating mode of a thermostat	66
Modbus Coil Control	Directly control a coil register on a connected Modbus device	67
Modbus Write Register	Directly control a Holding register on a connected Modbus device	67

Conditions

Conditions give you the option of executing an action only if certain conditions are fulfilled. Conditions add second layer of logic to alarms and actions.

For simple alarms, the conditions should be left at **No Condition**.

When selecting the condition type, additional controls are shown depending on the condition type.

Conditions

First condition: Schedule active

Schedule: Business Hours

Second condition: Output state

Output: Relay 1

Output is: off

Do action if:

 both conditions are true (AND)

 either or both are true (OR)

[Delete this action](#)

Two conditions can be set for each action, and the logic can be either that both conditions have to be true at the same time (AND), or that at least one of them is true (OR).

The possible condition types are:

Condition	Effect	Required setting(s)
No condition	No filter. The action will always execute.	None
Input in alarm	The action will only execute if the referenced input is in alarm state.	Input reference
Input restored	The action will only execute if the referenced input is not in alarm state.	Input reference
Input less than	The action will only execute if the referenced input value is less than a given value.	Input reference Value (in 'real' unit)
Input more than	The action will only execute if the referenced input value is more than a given value.	Input reference Value (in 'real' unit)
Counter less than	The action will only execute if the referenced input counter register is less than a given value.	Input reference Counter value
Counter equal to	The action will only execute if the referenced input counter register is equal to a given value.	Input reference Counter value
Counter more than	The action will only execute if the referenced input counter register is more than a given value.	Input reference Counter value
Schedule active	The action will only execute if the referenced schedule is in active state.	Schedule reference
Schedule not active	The action will only execute if the referenced schedule is not in active state.	Schedule reference
Output state	The action will only execute if the referenced output has the given state.	Output reference Output state (on/off)

Outputs

Each ezeio™ supports up to 40 outputs. Each output has two possible states: on or off.

Outputs can be controlled by events or states.

Event controlling outputs

Event triggers include alarm actions, timer actions, schedule actions, email and text commands, or manual toggling of the buttons on the ezecontrol.com Status page.

When triggered, the ezeio™ will send a single command to the selected output. This method will control the relay state until the next command is received.

State controlling outputs

Relays can be directly controlled by conditions. By checking the **Use only conditions** box, event commands and manual controls will be disabled. The ezeio™ will continuously force the state of the output as long as the conditions selected are true (see Conditions, page 48).

The settings for each output are:

Output name

A user defined name for the output.

Output location

This defines the hardware where the output is located.

One-shot coil msg

This setting only applies to modbus connected output devices. If checked, the set command will only be sent when the output status changes. The default (unchecked) is that the modbus set command is repeated every 20 seconds.

Use only conditions

If this box is checked, the conditions will continuously be re-evaluated, and the output set accordingly. That means that any other attempts to control the output will be overruled.

Leaving this box unchecked means that the condition settings are not used. The output state is set by manual commands, alarm actions, remote commands or script.

For more details on conditions, see Conditions, page 48.

Output 1

Output display settings
Output name

Hardware/device setting
Output location
One-shot coil msg (applies to Modbus coils only)

Control conditions
Use only conditions (disables all other control)
First condition
Second condition
Turn output ON if both conditions are true (AND)
 either or both are true (OR)

[Delete this output](#)

Schedules

The ezeio™ supports up to 20 schedules.

Schedules can be used in condition logic to only cause actions during specified times, or can be used to directly trigger actions.

Each schedule can define up to four intervals, and each interval can be active on any day of the week.

To define a new schedule, click **Schedules** in the object tree, and then click **Add schedule**.

Select the days of the week when each interval should be active, and the start and stop time in 24h format (HH:MM).

If the start time is before or equal to the stop time, the interval will not be processed.

Up to four actions can be defined for when a schedule enters a defined interval, and an additional four actions can be defined for schedule interval exit.

For more information about actions, see [Actions](#), page 47.

Timers

Timers are a simplified form of schedules. Up to 20 timers are supported for each ezeio™.

When a timer reaches its defined point in time, it will run the defined actions (up to four actions per timer), and count down its recursion count.

Recurrence can be unlimited or set between 1 and 254 counts. When the counter reaches zero, the timer will not trigger any more. Timers can be set to reoccur indefinitely by entering 255 in the count field.

Leaving all drop down fields set to “- any -”, the timer will trigger every minute, until the count runs down to zero. Specify the minute, to limit triggering to once per hour. Specify the minute and hour to limit to once per day and so on.

Timers can be set from external messaging, such as email. See [Sending control commands](#), page 70 for details.

Year	- any -
Month	- any -
Day in month	- any -
Weekday	- any -
Hour	- any -
Minute	- any -
Count	255 (1-254, 255=Repeat forever)

Every minute, on the minute

Year	2025
Month	November
Day in month	12
Weekday	- any -
Hour	10
Minute	47
Count	1 (1-254, 255=Repeat forever)

Once, on Nov 12, 2025 at 10:47

Thermostats / Thermostat schedules

The ezeio™ can communicate with up to 10 thermostats connected to the Modbus network. The thermostats are controlled by thermostat schedules that are defined under the **Thermostats** menu.

Every ezeio™ supports up to four (4) thermostat schedules, and each thermostat schedule can control up to four (4) physical thermostats, with a total maximum of 10 thermostats per ezeio™.

To associate a thermostat to a thermostat schedule, first define the thermostat schedule, then find the thermostat under the **Device** menu, and select the appropriate thermostat schedule.

All thermostats that are associated with schedules will be listed on the status page automatically, where the current status of the thermostat will be shown.

Thermostat schedule 1

Thermostat schedule display settings

Schedule name:

Schedule type:

Conditions for using alternate settings

First condition: *

Second condition: *

Alternate settings if: both conditions are true (AND)
 either or both are true (OR)

*If both conditions are set to "no condition", the standard settings will apply

Other settings

Stir holdoff: minutes after last fan activity (0=off)

Stir time: minutes to run fan after holdoff

Conditions for using alternate settings

Associated thermostats will be switched to alternate mode if the conditions selected evaluate to “true”.

Other settings / Stir

The stir feature monitors the activity of each thermostat, and will ensure that the fan runs with the set interval as a minimum. If the thermostat automatically uses the fan for heating or cooling, the stir timer is reset to avoid unnecessary fan activity.

Stir holdoff sets the maximum number of minutes the fan is allowed to be off.

Stir time sets the number of minutes the fan should run if there is no other activity.

Devices

The **Devices** branch in the configuration tree lists all the defined hardware in the system.

The first item is always the ezeio™ itself.

Do not delete or rename the ezeio™ controller device.
It's required by the system.

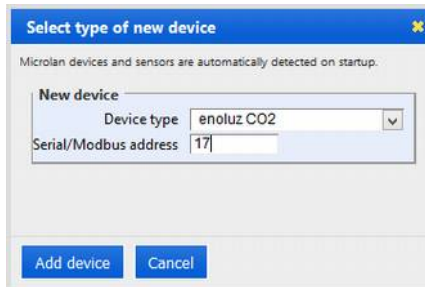
The other items are MicroLan, Wireless, and Modbus devices that the system has knowledge about.

Configuring Modbus Devices

To add a Modbus device, first ensure the bus address is unique (see the manual for the Modbus device for details on how to configure the specific device with its address and communication settings).

All devices sharing the Modbus communication line needs to be set for the same speed and format. The default is 19200 bps, 8 data bits, no parity.

Click the **Devices** root item in the configuration tree, and then click **Add Device**.



In the dialog box, select the type of device you are adding, enter its address, and click **Add Device**.

Devices listed have been tested for compatibility by eze System. Contact us to evaluate a new device for integration.

Be sure to give your new device a name so its easy to refer to in.

The device resources will now be available in the system. You can Add inputs/outputs and select the source from the **Input location** drop down menu.

Wireless devices

As of the release of this document, two wireless models of wireless expansion modules are available from eze System;

- Wireless I/O Expander (4 inputs / 2 outputs) p/n 130-0020-0
- Wireless Temp & RH (battery powered) p/n 130-0030-0

Both communicate with ezeio™ models equipped with 868/916 MHz radio modules.

A single ezeio™ can support up to nine (9) wireless devices.

Configuring wireless devices

Click the **Devices** root item in the configuration tree, and then click **Add Device**. Select **Wireless device** from the drop down menu and enter the serial number of the new device. Click **Add Device** and **Save Changes**.

The device is now added to the system and you can add inputs and outputs tied to the RF units.

Wireless Pairing

After adding the device, power up the wireless expander. The signal LED will blink slow until the communication with the ezeio™ is established, this may take up to 4 minutes. Short (1 sec) flashes indicate normal operation and communication with the ezeio™.

A paired expansion unit will only communicate with its designated ezeio™. Multiple radio networks can overlap, but each expansion unit will only communicate with a single ezeio™.

Break pairing

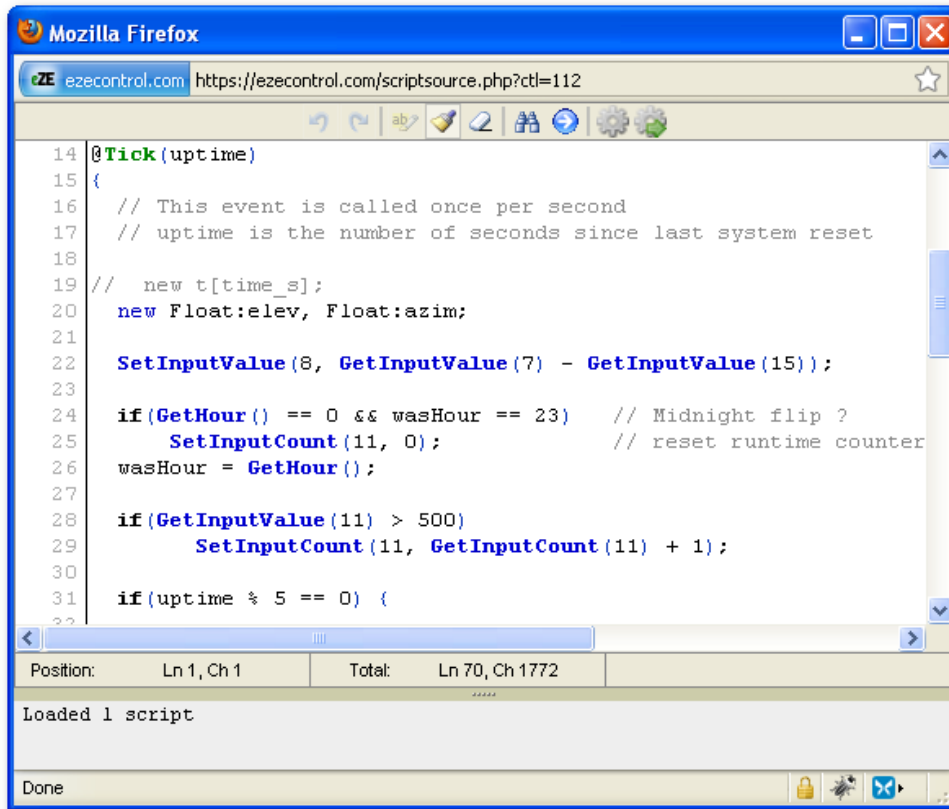
To break pairing, power cycle the Wireless expander 5 times, removing the power during the time when the LED is lit. The LED will blink rapidly on the 5th power up to indicate it is reset.

Be sure to delete the device from the previously paired ezeio™ to avoid conflict.

Script (premium feature)

Script is a premium feature that can be enabled for advanced programming of the ezeio™. Please contact eze System if you wish to use this feature.

If enabled, a “Script” option will be displayed in the resource tree and an online script editor is made available.



```
14 @Tick(uptime)
15 {
16     // This event is called once per second
17     // uptime is the number of seconds since last system reset
18
19     // new t[time_s];
20     new Float:elev, Float:azim;
21
22     SetInputValue(8, GetInputValue(7) - GetInputValue(15));
23
24     if(GetHour() == 0 && wasHour == 23) // Midnight flip ?
25         SetInputCount(11, 0); // reset runtime counter
26     wasHour = GetHour();
27
28     if(GetInputValue(11) > 500)
29         SetInputCount(11, GetInputCount(11) + 1);
30
31     if(uptime % 5 == 0) {
32
```

Position: Ln 1, Ch 1 Total: Ln 70, Ch 1772

Loaded 1 script

Done

Please refer to the section about the Script language starting on page 88.

System

The system screen has the following settings:

Controller name

A user defined name to identify the ezeio™.

Controller location

A user defined text to identify the ezeio's location.

System info address

If you enter an email address here, the system will send informational messages to this address when the ezeio™ communication with the server fails/restores, or when the ezeio™ changes from communicating over Ethernet to GSM and back. Copies of these emails are also sent to the accounts system info address (see System info address, page 68)

Time zone

The time zone where the ezeio™ is installed.

**** SECURITY ALERT ****

The fields under **Access Control Settings** enable external systems to access/control your ezeio™. To disallow any external access, leave these fields blank. If they are used, use strong passcodes and keep them safe.

Read passcode

This code is used to authorize external access to data, such as API functions and status requests. The code needs to be at least three characters long to be accepted. Only data retrieval is allowed with this code. No changes to the system are possible. To disable external access, leave this field blank.

Control passcode

This code is used to authorize external commands, such as output state changes or setting timers. The code needs to be at least four characters long to be accepted.

To disable external commands, leave this field blank.

Registration code

This code is used to authorize new users to the account, or to register the ezeio™ with a different account.

To allow a different account owner to take over the ezeio™, that user will need this code. In addition, the ezeio™ needs to be deleted from the current account in order for it to be re-registered.

Allow firmware update

If unchecked, firmware updates will not be applied to this ezeio™. This may be desirable in critical systems, where the installation has passed extensive testing. Usually we recommend leaving this checked.

Allow config update

If unchecked, no configuration changes will be downloaded to the ezeio™. Changes are still allowed on the server, but they are not synchronized.

Allow dealer access

This checkbox is only visible if the ezeio™ is serviced by one of eze System's authorized resellers/dealer. The owner of the ezeio™ may choose to allow configuration access to the dealer by checking this box.

The name of the reseller is visible to the right of the checkbox. Click the name for contact information.

Delete controller

Click **Delete controller** if you want to remove the ezeio™ from your account. The ezeio™ will be returned to an internal “pool” and be made accessible to other account holders for re-registration – provided they have the correct Registration code.

Note that all settings, and the log history for the ezeio™ will be retained even if it's deleted.

To move a ezeio™ from one account to another the ezeio™ first has to be 'deleted' in order to be available to add to the new account. Deleting an ezeio™ (**Configure**→**System**→**Delete Controller**) only removes the association with the account. It does not alter any configuration or captured data.

Ethernet settings

The settings for IP, Net mask, Gateway and DNS should normally be left blank, which will enable standard DHCP. The settings only apply to the physical Ethernet connection and if used, all four fields must have valid IP settings.

If the IP settings are incorrect, the ezeio™ will not be able to communicate with the server. To temporarily change back to DHCP, apply the HALT jumper during power up. See page 14 for details.

External server URL

This setting is used by the script function **ExtAPICall** (page 108). This function can be used to send a message from a script directly to a third party server, using html-post, json or xml formatting. The url must start with one of:

http:// or **https://**

- message parameters will be sent using http-post formatting.

json:// or **jsons://**

- message parameters formatted as a json object.

xml:// or **xmls://**

- message parameters formatted as xml.

The call will be to port 80 or port 443 (if using SSL). For verification of the data, a header “X-HASH” is added with a MD5 sum of the full submitted text concatenated with the read passcode from the ezeio™ configuration. See Verifying the validity of the data on page 84.

Phone module PIN

This should be set to the PIN code on the GSM module. If the module does not have a PIN code programmed, this field should be left blank.

SIM card PIN

This should be set to the PIN code on the GSM SIM card. If the card does not have a SIM PIN code programmed, this field should be left blank.

GPRS APN, login name , password

These fields need to be set to the APN, login and password of the GSM operator. Contact your GSM service provider for details.

An incorrect APN / login name / password will prevent the ezeio™ from communicating. Make sure you have the correct settings.

Phone init string

Additional commands to the GSM module. Usually this should be left blank.

GPRS init string

Additional commands to the GSM module. Usually this should be left blank.

Phone module PIN, SIM card PIN, GPRS settings and init strings only applies to ezeio™ models with built-in GSM transceiver.

Clone Controller

If the account has more than one ezeio™ associated, this function will allow copying all settings and data from any other ezeio™ to the current ezeio™. This will overwrite any settings and data that exists on the current ezeio™.

For Cloning to work, the current ezeio™ need a service setting equal or higher than the ezeio™ that will be cloned.

Modbus speed

This setting selects the communications speed on the Modbus interface. Possible settings are 300 bps to 57600 bps. 19200 bps is the (default).

The port always use 8 data bits, no parity.

Use slow polling

Some Modbus hardware require a delay between data exchanges. If this checkbox is active, a 50ms additional delay is added between packets.

Custom protocol

This setting will disable the Modbus functionality completely, and allow the user to write custom script functions for the communication on the serial port. See the [SerialSend](#) fuction on page 110.

Type of controller

The hardware type of ezeio™ (usually 1)

Firmware version

The ezeio™ firmware version

Last system reset

Time and date of when the ezeio™ was last reset.

Last comm reset

Time and date of when the ezeio™ last renegotiated contact with the server.

Last contact

Time and date of when the ezeio™ last communicated with the server.

Last endpoint

IP-address and port of the ezeio™ when it last communicated with the server.

Last local IP

The local IP of the ezeio™ when it last communicated with the server.

Actions

Inputs, Schedules and Timers can trigger actions.

List of possible actions:

Action Type	Description	See Page
Send Message	Send a message to email, SMS, voice call, third party system and more	60
Log Event	Record an event in the event log	66
Set Output	Set or reset a local output	66
Set Counter	Set an input counter register to some specific value	66
Increment Counter	Count up an input counter register	66
Decrement Counter	Count down an input counter register	66
Control Thermostat	Adjust the set point or change operating mode of a thermostat	66
Modbus Coil Control	Directly control a coil register on a connected Modbus device	67
Modbus Write Register	Directly control a Holding register on a connected Modbus device	67

Action: Send message

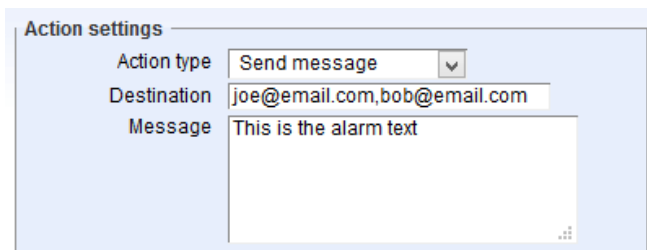
This will send a message to the defined destination.

Possible Message Types	Destination Field Format Examples	Page
Email	<code>jsmith@mycompany.com</code>	62
SMS Text	US: <code>+12125551234</code> Sweden: <code>+46707123456</code>	62
Twitter	<code>@mytwitteraccount</code> <i>*needs to be linked to twitter account.</i>	63
HTTP Post	<code>http(s)://my.server.com/script</code>	63
JSON Post	<code>json(s)://my.server.com/script</code>	63
Pushover	<code>:abc123def456abc123def456abc123xy</code>	64
Push to Speech	<code>!abc123def456abc123def456abc123xy</code>	64
Exosite	<code>exosite://[CIK]:[ResourceID]</code>	64
Voice	US: <code>12125551234</code> Outside US: <code>01161555512344</code> (AU)	64
Control API call (to another ezeio)	Same account: <code>#AAA-000</code> Different Account: <code>#AAA-000:password</code>	65

SMS recipients must begin with a plus (+) and Country Code.

Voice recipients outside the USA must begin with **011** & Country Code.

You may combine destinations by separating them with commas or semicolons, like in this example:



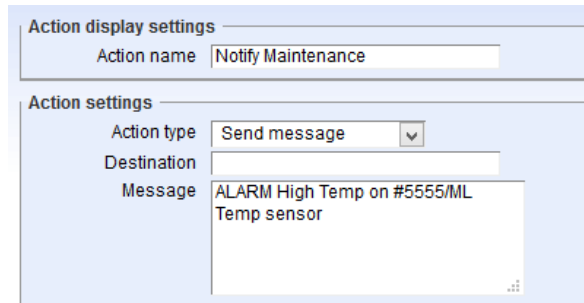
The screenshot shows a dialog box titled "Action settings". It has three main fields: "Action type" with a dropdown menu set to "Send message", "Destination" with a text input field containing "joe@email.com,bob@email.com", and "Message" with a text area containing "This is the alarm text".

It is possible to send an alarm message to email, twitter, HTTP POST and place a voice call to multiple recipients all on the same alarm.

The Destination field is limited to 200 characters.

Notes about the Message field

After selecting **Send message** or **Log event** from the **Action type** drop down menu, a default message will be generated containing the Alarm name and Input name.



The screenshot shows a configuration window with two sections. The top section, 'Action display settings', contains a text field for 'Action name' with the value 'Notify Maintenance'. The bottom section, 'Action settings', contains a dropdown menu for 'Action type' set to 'Send message', an empty text field for 'Destination', and a text area for 'Message' containing the text 'ALARM High Temp on #5555/ML Temp sensor'.

You may replace the message with your own text. The message can be up to 4000 characters long, and can contain special references to insert values from the system. References always start and end with a **#**-character.

This table lists the possible parameters:

Parameter	Description
#VALx#	Inserts the converted momentary value of input with number x.
#CNTx#	Inserts the counter register value of input number x.
#OUTx#	Inserts the current state ("on" or "off") of output number x.
#SCHx#	Inserts the current state ("active" or "inactive") of schedule x.
#STAx#	Inserts the translated input value text from the Text Status box under input settings.

Example of a message using references:

Warehouse temperature is #VAL8# and exhaust fan is #OUTx#

The input/output/schedule number can be looked up by clicking the root nodes in the resource tree.

Up to four (4) references are allowed in a single message.

Email

To send a message to an email recipient, simply enter the email address in the Destination field.

To enter multiple recipients, separate the addressees with a comma or a semi-colon.

Each email will include boilerplate information such as ezeio™ name, reason for the alarm etc. To remove the boilerplate text and only send the message text, put the message in double quotes (“”).

Please make sure that your email provider does not filter email from ezeicontrol.com. If you are missing emails, please look in your spam folders.

Controlling the email subject

When sending emails, the subject field defaults to the text “ALARM” or “RESTORE” followed by the name of the resource that caused the alarm and the alarm name.

By inserting a vertical line character “|” in the message, the subject line will be replaced with the text before the “|”, and the message body will be the text after the “|”.

Example: “This is my subject|This is the body text”

SMS Text

To send a text message, enter the recipients phone number, starting with plus (+) and the country code. Note that the country code is always required.

The message will include some boilerplate information (ezeio serial, location etc). To remove the boilerplate text and only send the message text, put the message in double quotes (“”).

Twitter

To send a message to a Twitter account, the destination should start with a '@', followed by the Twitter account name. The Twitter account needs to be linked to the ezecontrol account.

A pop-up link will be displayed when the Save Changes button is pressed. Be sure to click this link and log in to your Twitter account when asked to.

If the Twitter account name is followed by colon and one of 'name', 'url', 'location', 'description', the corresponding account setting in Twitter will be changed.

Examples:

[@mytwitteraccount](#) (update status)

[@mytwitteraccount:description](#) (update twitter account description)

Twitter account messages should not be considered reliable alarm paths.

HTTP POST

To send a HTTP POST, enter the URL of the server starting with http:// or https://.

The message will be sent with the following POST fields:

POST Field Name	Description
serial	ezeio serial number
controllername	ezeio name
controllerlocation	ezeio location
source	Source of the alarm (System/Timer/Script/Schedule/Input)
action	ALARM or RESTORE
account	Account number
accountname	Account name
time	Time of alarm (local time)
message	Text of message

JSON POST

To send a JSON POST message, enter the URL of the server starting with json:// or jsons://

The message will be sent as a json object with the same fields as in the HTTP POST above.

Pushover (<http://pushover.net>)

Pushover is a mobile app available for iOS and Android ideal for short messages. To send messages to your Pushover app, enter the destination as your Pushover User Key (32 characters, available under the app settings) starting with a **colon**. Example: **:abc123DEF456abc123DEF456abc123Xy**

See <http://pushover.net> for more information.

Push to Speech (<http://pushtospeech.appspot.com>)

Push to Speech is a mobile app available for Android that reads out loud the messages sent to it. There is no user interaction required to receive messages.

To send messages to your Push to Speech app, enter the destination as your Device Identifier (30 characters, from the app) starting with an **exclamation mark**. Example: **!abc123def456abc123def456abc123xy**

See <http://pushtospeech.appspot.com> for more information.

Exosite (<http://exosite.com>)

The ezeio™ can send messages to the Exosite web services.

Enter the destination like this: **exosite://[CIK]:[ResourceID]**

The value is set to the content of your message.

See <http://exosite.com> for more information.

Voice

To dial a phone number and send a voice message, simply enter the telephone number in the Destination field. To enter multiple recipients, separate the telephone numbers with a comma. Each number will be dialed in turn, and the subsequent number will be dialed only if the previous number does not acknowledge receipt of the call (recipients will be prompted to press five (5) to acknowledge). The number must not have any punctuation, no dashes, spaces or periods.

Calls within the US must start with 1. Calls to destinations outside the US must start with 011 immediately followed by the country code.

The voice message will include boilerplate information, such as ezeio™ name/serial, instructions on how to acknowledge etc. To send only the text in the message field, enclose the message in double quotes (e.g. “message”).

Control API call

If the destination starts with a **#**-character, the message is interpreted as a control API call to another ezeio™. Immediately following the **#**, enter the serial number of the remote ezeio™. If the remote ezeio™ is on the same account, no password is needed. If the remote ezeio™ is on a different account, enter a colon (:) followed by the control password (the control password is located system .

In the message box, enter the desired parameters separated by comma. See the REST control API section, page 68 for more information.

Example to turn on output 2 for 10 seconds on ezeio™ XYZ-987:

Destination: “**#XYZ-987:secretpass**”

Message: “**output=2, cadence=1, duration=100**”

Note that the ezeio™ that is controlled needs to have API service activated.

Action: Log event

This action will simply log the message in the ezeio™ event log together with a time stamp.

Action: Set output

This action will directly affect an output on the ezeio™ or connected expansion device. After selecting a relay, two options are available:

Cadence allows you to select the state of the output.

Cut off will act as a timer for the output. If zero is entered the output will remain in the selected state indefinitely, or until a different action changes its state.

Action: Set counter

This action will set the Count register on the referenced input to a specific value.

Action: Increment counter

This increments the Count register on the referenced input.

Action: Decrement counter

This decrements the Count register on the referenced input.

Action: Control thermostat

If T32P thermostats are connected to your ezeio™, the control thermostat action allows you to override the scheduled operation, using the options shown below.

Command	Options
System Mode	Off, Heat, Cool, Auto
Fan Mode	Off, On, Auto
Run Schedule	No setting. Cancels override if applied.
Override setting	Cool setpoint, Heat setpoint
Set Schedule Mode	Standard, Alternate

Action: ModBus coil control

This action allows you turn on or off a the coil of a selected ModBus device.

Action: ModBus write register

This action allows you to write a value to a specific write register of a selected ModBus device.

Account screen

The account screen allows access to all settings for to manage account and users. Only account administrators are allowed to change account information and edit other users.

Depending on user privileges, there are up to three tabs on the account screen:

Account - Generic account information

Personal - Own settings

Users - Settings for the other users on the same account.

Account

The settings under this tab are informational in nature. They do not change the functionality of the ezeio™ system.

Contact email

The contact person and email is referenced in emails for supporting users on the account, and is also sent an informational email when new users register.

System info address

If you enter an email address here, the system will send informational messages to this address when an ezeio™ on this account fails/restores contact with the server, or when an ezeio™ changes from communicating over Ethernet to GSM and back. Copies of these emails are also sent to the ezeio™ **system info** address, see page 55.

Account status

By checking the Accept New Users checkbox, new users can register with the account, provided they have a ezeio™ serial number and registration code of an ezeio™ that is already enrolled with the account, see Adding users to an existing account on page 10.

Personal

The personal tab allows access to information about the logged in user. The user can edit these settings if the **edit own info** privilege is set for the user.

If the user attempts to change the email address, a confirmation email will be sent to the new address. The user must retrieve a confirmation code from that email and enter it in the **Confirmation Code** field in order for the new email address to be accepted.

The Confirmation code is only required when changing email address.

Users

The **Users** tab lists all users on the account, except for the logged in user. If there are no other users registered on the account, this list will be empty.

To add users, please refer to Creating accounts and users on page 9.

To see and edit information about a specific user, click on the row in the list.

The account administrator is able to change any information about the user, except for the login name.

Passwords are stored in encrypted form in the system, and can never be retrieved or shown.

If a user has forgotten his/her password, the administrator may assign a new password, but the old password cannot be retrieved.

Log in

This check box must be checked to allow the user to log in.

Edit own info

This check box allow the user to change the information under the “**Personal**” tab.

Edit controllers

This check box allows the user access to the **Config** tab.

Remote control

Check this box to allow the user to control outputs and timers via the web. If this box is unchecked, the user will not see the **Control Passcode** on the **System** screen.

Release controllers

If this box is checked, the user can delete ezeios from the account.

Manage account

With this box checked, the user can access other users information and privilege settings.

Sending control commands

Email

Timers and outputs can easily be controlled via email. You will need:

- The serial number of the ezeio.
- The **Control passcode** (from the **Configure** → **System** screen)
- The name of the output or timer you want to control.

Create a new email and sent it to **{serial}@ezecontrol.com**

where {serial} is the serial number of the ezeio™. For example:
xyz987@ezecontrol.com

The email subject can be left blank

The first line of the email shall be the Control passcode.

The following lines shall be commands. See Control Commands, page 71.

If unsuccessful the system will reply referencing the error as communication, command syntax, or invalid ezeio ID or password.

Control via SMS (cellphone texting)

Just as the ezeio can be controlled through email, SMS (Short Message Service) can also be used.

The message should be sent to:

Region	Access number
US, Canada:	+1 916 281 9001
United Kingdom / Ireland:	+44 7937 985 875
Sweden:	+46 769 439 907
Australia / New Zealand:	+61 448 838 189
Any other region:	+44 7937 985 875

Send the message with ezeio serial number, passcode and commands separated by comma or line break.

See Control Commands, page 71 for possible commands and syntax.

Example:

SMS to **916 281-9001**: **“xyz987,mysecretpass,output fan on“**

(turn on the output named “fan”)

The system will reply with a SMS message to acknowledge the action. If unsuccessful the system will reply referencing the error.

Unlike email commands, there will be no reply message if the ezeio ID or password is invalid.

Control Commands

Using Email or Text messages (SMS), the following control commands are available:

Command	Function	Parameters
OUTPUT	Directly controls an output	Output name, State, [cutoff] If cutoff is not given, 0 is default.
COUNTER	Sets a counter register	Input name, Value
TIMER	Set/enable a timer	Timer name, Time
STATUS	Request status	No parameters
THERMOSTAT	Control a thermostat	Tstat name, Command, {setpoint} Command is one of: Heat {setpoint} Cool {setpoint} Alternate Standard Run
ADJUST	Adjust a thermostat	Degrees, [minutes] If minutes is not given, 1h is default.

The system will reply to confirm the command was understood.

Examples:

output warehouse lights on

(turn on the output named "warehouse lights")

out warehouse lights on 20s

("output" can be abbreviated, 20s means it will be on for 20 seconds)

timer sauna 18:00

(trip the timer named "sauna" at 6pm, once)

timer sauna 6:15pm

(am/pm is accepted also)

timer sauna off

(disable the timer, set counter to zero)

timer sauna 1810 x12

(colon in time is optional. Start at 6:10pm the following 12 days)

timer sauna thursday 9:00

(Start on Thursday 9am, once)

timer sauna 3-27 13:45

(MM/DD or MM-DD)

tIM SAUna FRI 11:23pM

(case doesn't matter, and weekdays can be abbreviated, 3 chars min)

Server API

API access and security

The API can be accessed either through HTTP or HTTPS. We strongly advise against accessing API features through HTTP, since the communication in this case will be sent unencrypted.

If at all possible, use HTTPS for API access.

In order to use the API features, the ezeio™ must have a service level that includes enough API calls for the specific application.

Click the Configure tab to set an appropriate service level for each ezeio.

The screenshot shows the ezeio web interface. At the top, there is a navigation bar with tabs for Dashboard, Status, Configure, and Account. The 'Configure' tab is selected. Below the navigation bar, there is a sidebar with a search field and a list of controllers. The main content area is titled 'Configuration of AAA-157 : Lighthouse demo : eZE System'. It contains a 'Service status and settings for AAA-157 "Lighthouse demo"' section. This section has a 'Status' sub-section with fields for 'Service paid until' (2014-09-01), 'SMS/Voice alarms' (100 remain (resets monthly)), and 'API requests' (20000 remain (resets every 24h)). Below this is a 'Setting' sub-section with a table for selecting a service level. The table has columns for 'Basic', 'Standard', and 'Premium'. The 'API calls per 24h' row is circled in red, showing values of 5 for Basic, 2000 for Standard, and 20000 for Premium. The 'Premium' option is selected with a radio button.

	Basic	Standard	Premium
Logged inputs	5	15	40
Minimum log interval	60 sec	30 sec	10 sec
Historic data storage	0 mo	12 mo	37 mo
API calls per 24h	5	2000	20000
SMS/Voice messages	10	100	100/mo
Cost per month	0.00	0.00	0.00

API authentication and example

All REST/JSON calls use the Digest auth (RFC 2617) method to validate credentials for accessing data. This avoids sending access credentials unencrypted even if SSL is not used, although we recommend using SSL whenever possible.

The user name is the ezeio™ serial number (e.g. 'XYZ-987'), and the password is the read-passcode for status.php and log.php calls, and the control-passcode for calls to control.php

Parameters may be passed by either the GET or POST method.

This is a simple example of how to call the log.php API from PHP code:

```
<?php
$url = "https://ezecontrol.com/api/log.php"; // API URL

$serial = "XYZ987"; // ezeio serial
$pass = "supersecret"; // Read passcode

$fields = array(
    "input" => 4, // input 4
    "from" => "2010-08-22", // start time
    "to" => "2010-08-24" // end time
);

$ch = curl_init($url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_DIGEST);
curl_setopt($ch, CURLOPT_USERPWD, "$serial:$pass");
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $fields);
$result = curl_exec($ch); // send the request
curl_close($ch);

print_r (json_decode($result)); // process the response
?>
```

The API can also be called directly from a browser like this:

```
https://XYZ987:supersecret@ezecontrol.com/api/log.php?
input=4&from=2010-08-22&to=2010-08-24
```

Live status in JSON format via REST API

This API call returns the most current status of the ezeio™, including all inputs, outputs, thermostats and the last few log events.

<https://ezecontrol.com/api/status.php>

The password needs to be the “Read passcode” from the system setting.

The following parameters are recognized:

Parameter	Range	Description
logid (optional)		The last log id from previous call (use if frequently calling the API to reduce the amount of data returned)
peek (optional)	0 (default), 1	If set to 1, the server will return the cached status rather than sending a request to the ezeio. This will speed up the request and reduce the data traffic to the ezeio (useful if ezeio is on cellular). The cache is updated roughly every minute while ezeio is on a physical connection, and every 10 minutes while on cellular.

Example:

<https://ezecontrol.com/api/status.php?logid=4821349>

Historical data access in JSON format via REST API

This API call will return historical data related to a given input:

<https://ezecontrol.com/api/log.php>

The password needs to be the “Read passcode” from the system setting .

The following parameters are required:

Parameter	Range	Description
input	1 – 40	The input number
from	YYYY-MM-DD[HH:MM]	Year, month, day, and optionally hour and minute when the data should start.
to	YYYY-MM-DD[HH:MM]	Year, month, day, and optionally hour and minute when the data should end.

All timestamps are Zulu (GMT) time.

Example:

<https://ezecontrol.com/api/log.php?input=4&from=2009-03-05&to=2009-03-08>

Controlling the ezeio™ via REST API

The following API call allows direct control of several features:

<https://ezecontrol.com/api/control.php>

The password needs to be the “Control passcode” from the system setting.

The following parameters are required in each call:

Parameter	Range	Description
output counter input timer thermostat thermostatschedule	See below	See below for required parameters for each control type

Direct output control : 'output'

Use this command to directly control the output state.

Parameter	Range	Description
output	1 – 40	The number of the output to control
cadence	0 – 7 (see below)	The cadence to apply
duration	0, 1 – 65535	Number of 1/10 th seconds to run this cadence before turning the output off again. 0 = infinite

Example, turn on output 1 for 2 seconds:

`https://ezecontrol.com/api/control.php?
output=1&cadence=1&duration=20`

Cadence	Pattern
0	Off
1	On
2	100ms on / 900ms off (0.1s pulse every second)
3	1s on / 9s off (1s pulse every 10s)
4	2s on / 58s off (2s pulse every 60s)
5	100ms on / 100ms off (5Hz blink)
6	0.5s on / 0.5s off (1Hz blink)
7	1s on / 1s off (0.5Hz blink)

Set input counter : 'counter'

Use this command to change the value in one of the counter registers.

Parameter	Range	Description
counter	1 – 40	The number of the input/counter
value	0 – 2 ³¹	The new value of the counter

Example, set the counter value on input 5 to 12345:

```
https://ezecontrol.com/api/control.php?  
counter=5&value=12345
```

Set input value: 'input'

Use this command to change the value of one raw input value.

Parameter	Range	Description
input	Min 1 character	The name of the input
value	-2 ³¹ – 2 ³¹	The new raw value of the input

Example, set the counter value on input 5 to 12345:

```
https://ezecontrol.com/api/control.php?  
input=5&value=12345
```

Set timer: 'timer'

Use this API to set up an existing timer.

Parameter	Range	Description
timer	1 – 20	The timer number to set up
year (optional)	2000 – 3000	Year. If not set, default is "any".
month (optional)	1 – 12	Month. If not set, default is "any".
day (optional)	1 – 31	Day of the month. If not set, default is "any".
weekday (optional)	0 (Monday) – 6 (Sunday)	Weekday. If not set, default is "any".
hour	0 – 23 (24h format)	Hour – required
minute	0 – 59	Minute - required
count (optional)	1 – 255	- Number of times the timer shall be executed. - Set to 255 to repeat forever. - Default is 1.

At a minimum, either hour/minute or count needs to be given for the command to be accepted.

Example, Trip timer 1 once, next Tuesday at 8:15pm

```
https://ezecontrol.com/api/control.php?timer=1&weekday=2&hour=20&minute=15
```

Example, Change the count without affecting the time setting:

```
https://ezecontrol.com/api/control.php?timer=1&count=5
```

Control thermostat: 'thermostat'

This API controls a thermostat directly

To leave a setting unchanged, just omit the parameter from the command.

Parameter	Range	Description
thermostat	1 – 31	Thermostat modbus polling address
setHeat	150-400 (1/10 Celcius) 500-950 (1/10 Farenheit)	Heating setpoint in 1/10 th degrees. (so 745 = 74.5 F) Setpoint is immediately applied, and the override timer will start automatically. When the override timer runs out, the thermostat will reset to its programmed setpoints.
setCool	150-450 (1/10 Celcius) 460-990 (1/10 Farenheit)	Cooling setpoint – see setHeat
adjHeat	0 to -150	Adjust the heat setpoint down this number of 1/10 th degrees minAdjust need to be non-zero for this command to work
adjCool	0 to 150	Adjust the cool setpoint up this number of 1/10 th degrees
minAdjust	0 (disable) to 1439	Number of minutes the system will use the adjHeat/adjCool settings before it returns to normal prorammg.
setLock	“ON”, “OFF”	If set to “ON”, the keypad is locked and will not accept any user input.
setScheduleMode	“AUTO”, “STANDARD”, “ALTERNATE”	Force the thermostat to use either the standard or the alternate settings from the controlling thermostat schedule. This overrides the condition setting in the schedule. AUTO returns to programmed state.
setSysMode	“OFF”, “HEAT”, “COOL”, “AUTO”	Set the operating mode of the thermostat.
setFanMode	“OFF”, “AUTO”, “ON”	Set the fan operating mode of the thermostat. Note that not all modes are supported by all thermostats.
newDataLatch	0, 1	If set to 1, the setHeat/setCool/setSysMode/setFanMode settings will be applied and override any other changes until latch is released.

Example; adjust +/-3 degrees for 90 minutes on thermostat on address 5

```
https://ezecontrol.com/api/control.php?  
thermostat=5&  
minAdjust=90&  
adjCool=30&  
adjHeat=-30
```

(line breaks added for clarity)

Modify thermostat schedule: 'thermostatschedule'

This API controls the settings of a thermostat schedule.

To leave a setting unchanged, just omit the parameter from the command.

Parameter	Range	Description
thermostatschedule	1 – 4	Thermostat schedule number (required)
day	1 (Monday) – 7 (Sunday)	Comma separated list of which days in the schedule that the change will apply to (required)
interval	1 (morning), 2 (day), 3 (evening), 4 (night)	Comma separated list of which intervals for each day the change will apply to (required)
stdHeat	150-400 (1/10 Celcius) 500-950 (1/10 Farenheit)	Heating setpoint in 1/10 th degrees. (so 745 = 74.5 F), standard schedule mode.
stdCool	150-450 (1/10 Celcius) 460-990 (1/10 Farenheit)	Cooling setpoint, standard schedule mode.
stdSysMode	"OFF", "HEAT", "COOL", "AUTO"	System mode, standard schedule mode.
stdFanMode	"OFF", "AUTO", "ON"	Fan mode, standard schedule mode.
stdStart	"00:00" – "23:59"	Time of day when this interval starts, standard schedule mode. Be careful not to set the interval start times to the same value, and make sure the start times are sorted.
altHeat	150-400 (1/10 Celcius) 500-950 (1/10 Farenheit)	Heating setpoint in 1/10 th degrees. (so 745 = 74.5 F), alternate schedule mode.
altCool	150-450 (1/10 Celcius) 460-990 (1/10 Farenheit)	Cooling setpoint, alternate schedule mode.
altSysMode	"OFF", "HEAT", "COOL", "AUTO"	System mode, alternate schedule mode.
altFanMode	"OFF", "AUTO", "ON"	Fan mode, alternate schedule mode.
altStart	"00:00" – "23:59"	Time of day when this interval starts, alternate schedule mode. Be careful not to set the interval start times to the same value, and make sure the start times are sorted.
OverrideMax	0 (disable) to 1439	Number of minutes the system will stay in override before it automatically resets to programmed settings.
AllowOverride	0, 1	If set to 1, manual override (at the thermostat) will be allowed. If set to 0, the thermostat keypad is locked from user input.

Example; Change the standard cooling setpoint in thermostat schedule 1, all weekdays, day and night intervals to 79.5F.

```
https://ezecontrol.com/api/control.php?  
thermostatschedule=1&  
day=1,2,3,4,5&  
interval=2,3&  
stdCool=795
```

(line breaks added for clarity)

Spreadsheet integration

Microsoft Excel® and LibreOffice Calc can be linked to the ezeio™ system so that the spreadsheet automatically updates with current data.

To use this feature, make sure the **Configure**→**System**→**Read Passcode** is set and that the service level allows API access.

Microsoft Excel®

These instructions are for Office 2013. Other version may differ.

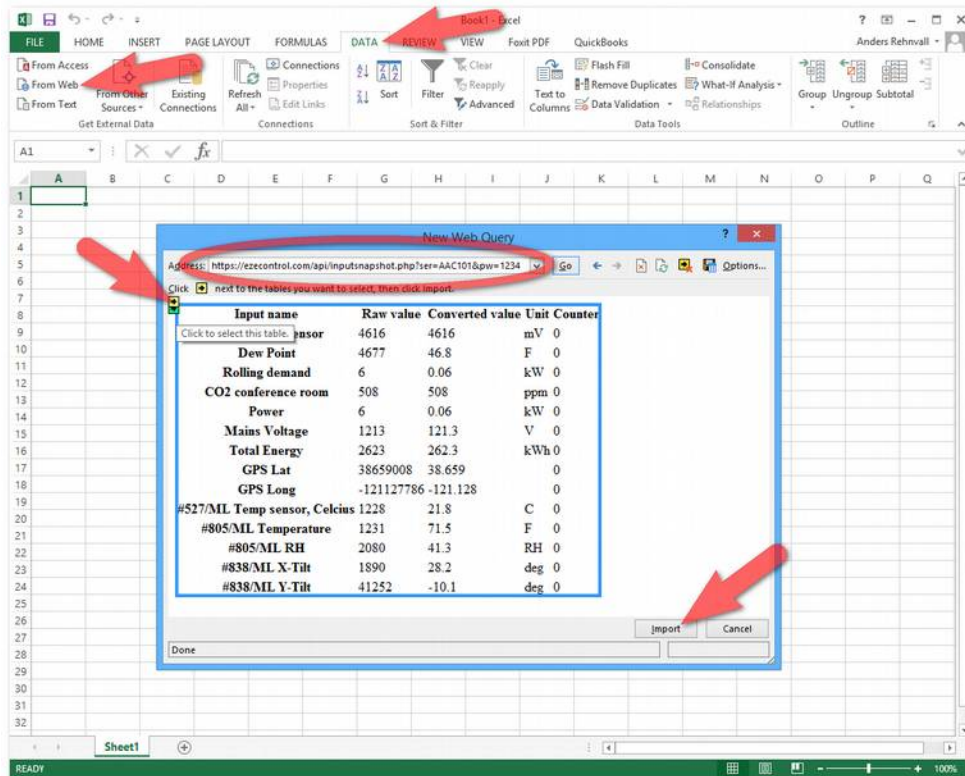
In Excel, select **Data**→**From Web**, then enter the following URL in the browser window:

<https://ezecontrol.com/api/inputsnapshot.php?ser=SERIAL&pw=PASSWORD>

(replace SERIAL and PASSWORD accordingly. The password is the **Read Passcode** from **Configure**→**System**)

Click Go. The window will populate with data from your ezeio™.

Click one of the yellow arrows to select the table, and then Import to insert it in your spreadsheet.

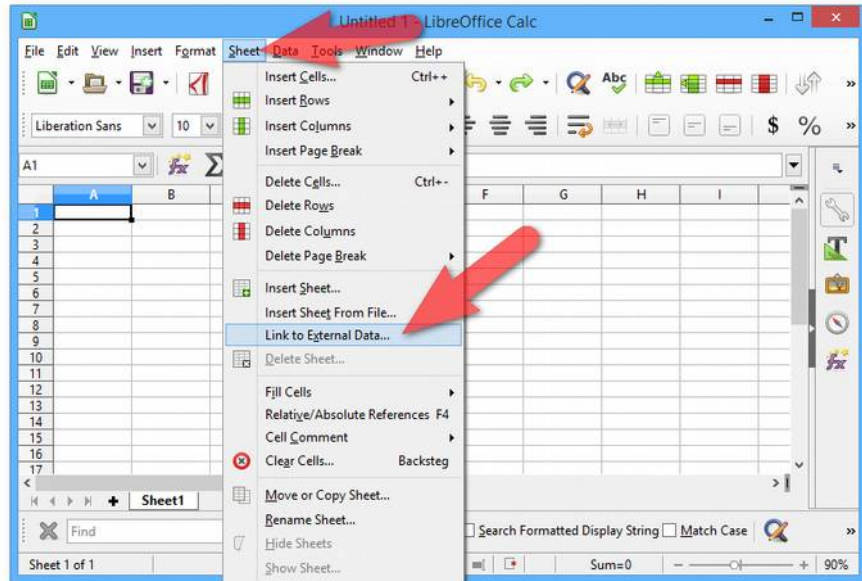


Right-click the top-left cell in the table to set up auto refresh. Refer to Excel help documents for more detail.

LibreOffice Calc

These instructions apply to version 5.1 of LibreOffice. Other versions may differ.

In Calc, select **Sheet**→**Link to External Data...**

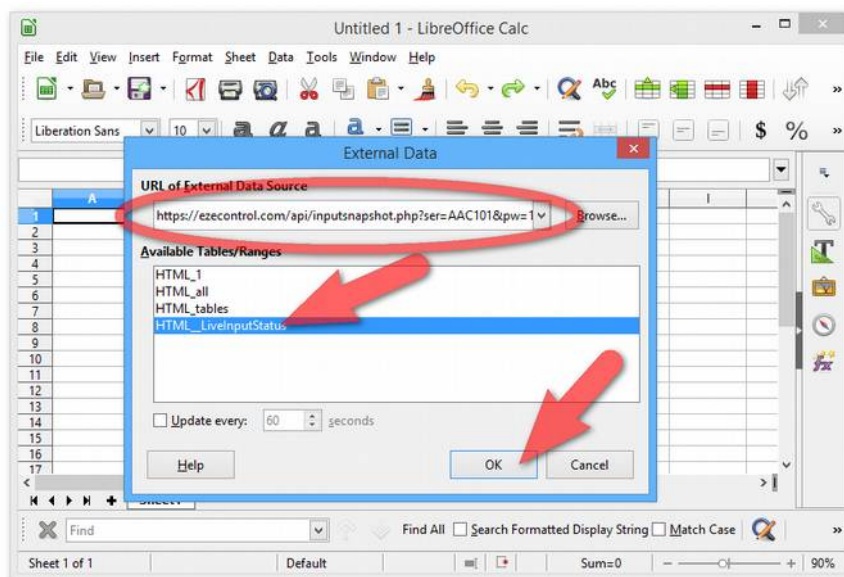


Enter the URL to your controller data in the **URL of External Data Source** field using the format below, and press Enter.

<https://ezecontrol.com/api/inputsnapshot.php?ser=SERIAL&pw=PASSWORD>

(replace SERIAL and PASSWORD accordingly. The password is the **Read Passcode** from **Configure**→**System**)

Select the **HTML_LiveInputStatus** item list and click **OK** to insert the data into your spreadsheet.

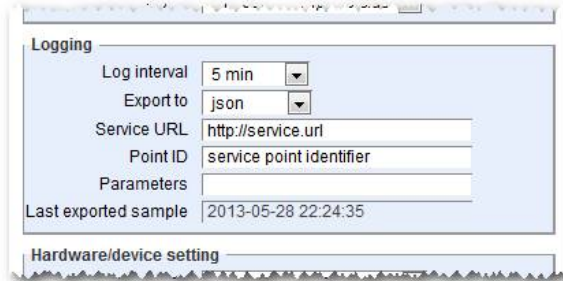


Automatic export (push)

The ezeio™ system supports pushing input log data to a third-party server. Each input can be set up to push data to a unique external server/service. There are several export protocols, most of which are vendor specific. The first protocol, “JSON”, is generic and public, and is defined here.

Exporting using JSON push

To set up JSON push export, select “JSON” in the [Export to](#) drop-down on the input setting screen. Enter the URL of the receiving server in the [Service URL](#) box. The export function supports **http** and **https** POST. In the [Point ID](#) box, enter an identifier to be used on the remote system to identify this input. This can be any text or number. The [Parameters](#) field is not used for JSON push.



Logging	
Log interval	5 min
Export to	json
Service URL	http://service.url
Point ID	service point identifier
Parameters	
Last exported sample	2013-05-28 22:24:35

The system will start exporting data as soon as the configuration is saved. The export process runs every 3 minutes and is not adjustable.

JSON push schema

The message sent to the remote server has the following structure:

```
{
  "ser" : "ezeio serial",
  "inp" : "name of the input in the ezeio config",
  "pid" : "ID of the input as set by the Point ID field",
  "unit" : "the unit of the input",
  "salt" : "random string",
  "data" : [
    {
      "time" : "timestamp for the sample in ezeio's local time",
      "raw" : "raw input value",
      "val" : "converted input value",
      "cnt" : "input's counter value"
    },
    { Additional samples, up to 200 per call }
  ]
}
```

Expected return value (acknowledgement)

The receiving service is expected to return the timestamp of the last processed sample. This will be the starting point for the next batch of data sent from the server. The expected return format is as follows:

```
{
  "result" : "timestamp of last sample"
}
```

Verifying the validity of the data

Each call from the ezeio™ system will include a HTTP header value called X-HASH. The value of this header is the MD5 sum of the full JSON data with the ezeio™ read passcode added to the end. If the receiving service has knowledge of the read passcode, it can easily verify that the data comes from a valid source by checking this hash against the received data.

An example in PHP is below.

```
<?php
define("READPASS", "verysecretpasscode");

$json = file_get_contents('php://input');

if(!isset($_SERVER["HTTP_X_HASH"]))
    die("ERROR: Missing hash");
if(md5($json . READPASS) != $_SERVER["HTTP_X_HASH"])
    die("ERROR: Hash mismatch");
if(($data = json_decode($json, true)) == NULL)
    die("ERROR: Invalid payload");

foreach($data["data"] as $sample) {
    // process the sample here
    $lastprocessed = $sample["time"];
}

// return the timestamp of the last processed sample
$r = array("result" => $lastprocessed);
print( json_encode($r) );
?>
```

For simplicity, this example has passcode hardcoded, but the user may support multiple ezeios with unique passcodes by retrieving the ezeio serial number from the JSON data before checking the hash validity.

BuildingOS export

BuildingOS by Lucid Design Group provides advanced visualization and analytics of energy usage suitable for larger organizations and kiosk type applications. More information is available at

<http://luciddesigngroup.com/buildingos>

To export data to BuildingOS, enter the following:

Field	Content	Example
Import/Export	BuildingOS export	
Service URL	Server URL	rpc.buildingdashboard.net/xmlrpc/dashboard/
Point ID	Point name	MyPointName
Parameters	User:Passcode	MYLOGIN:MYPASS

New data will be uploaded to the BuildingOS system every 180 seconds.

EnergyStar export

EnergyStar provides a free service called Portfolio Manager, designed for property managers to track and benchmark energy usage.

To use the EnergyStar system, you need to create an account at

<https://portfoliomanager.energystar.gov/pm/login.html>

Make sure the meters you want to upload to are shared and allowed to “Exchange Data”.

Find the meter ID by going to the “Edit Basic Meter Information” page. The last number in the URL is the meter ID as highlighted below:

<https://portfoliomanager.energystar.gov/pm/meter/edit/7654321>

To export data to Portfolio Manager, enter the following:

Field	Content	Example
Import/Export	EnergyStar export	
Service URL	Leave blank	
Point ID	Meter ID from PM	7654321
Parameters	Leave blank	

New data will be uploaded to Portfolio Manager once every hour.

eSight export

eSightEnergy provides a powerful software tool for data analytics, including reporting and billing functions. More information is available at <http://www.esightenergy.com>.

To export data to the eSight system, enter the following:

Field	Content	Example
Import/Export	eSight export	
Service URL	URL of service	http://test.esightenergy.com/esight/ws/
Point ID	Import point	SITENAME.POINTNAME
Parameters	Leave blank	

New data will be uploaded to the eSight system every 180 seconds.

Exosite export

Exosite is a IoT dashboard SaaS provider. It is primarily suitable for resellers that want to customize and brand the user interface for their clients. To use Exosite you have to create an account at <http://exosite.com> and create the “Devices” and “Data points” you want to use.

Each device in the Exosite system will be assigned a “CIK” code, and each data point will have a “RID” code. You will need to enter both these codes in the ezeio™ system to export data.

To export data to Exosite, enter the following:

Field	Content	Example
Import/Export	Exosite export	
Service URL	Device CIK	9770a9f7b00549c67a5a8a34e64a1c9ce17085b2
Point ID	Data point RID	94bf61241e71e7841049d624afe746c8ca2546d5
Parameters	Leave blank	

FTP export

The ezeio™ System can upload a FTP file once a day with hourly data. An example of an uploaded file is shown below

The file name is formatted like this:

YYYY-MM-DD_UPLOADHOUR_CONTROLLERID_INPUTNO.txt

```
FILE "2014-02-20_00_AAC-200_21.txt"
```

```
-----  
2014-02-19 01 METER123 13176  
2014-02-19 02 METER123 13177  
2014-02-19 03 METER123 13178  
2014-02-19 04 METER123 13178  
2014-02-19 05 METER123 13179  
2014-02-19 06 METER123 13180  
2014-02-19 07 METER123 13181  
2014-02-19 08 METER123 13182  
2014-02-19 09 METER123 13183  
2014-02-19 10 METER123 13184  
2014-02-19 11 METER123 13184  
2014-02-19 12 METER123 13185  
2014-02-19 13 METER123 13187  
2014-02-19 14 METER123 13187  
2014-02-19 15 METER123 13188  
2014-02-19 16 METER123 13189  
2014-02-19 17 METER123 13190  
2014-02-19 18 METER123 13191  
2014-02-19 19 METER123 13193  
2014-02-19 20 METER123 13194  
2014-02-19 21 METER123 13195  
2014-02-19 22 METER123 13196  
2014-02-19 23 METER123 13198  
2014-02-19 24 METER123 13199
```

To enable FTP export, enter the following:

Field	Content	Example
Import/Export	FTP 1h export	
Service URL	FTP server/path	my.ftp.server.com/directory
Point ID	Input name	METER123
Parameters	USERNAME:PASS	ftlogin:ftppass

The file will be uploaded every 24 hours after midnight local time.

Script language

Script introduction

The ezeio™ system supports the PAWN script language. PAWN has a C-like syntax and executes completely inside the ezeio™, allowing users to add custom functionality to the ezeio™.

This manual documents only the custom functions added to the language for interaction with the ezeio™ resources. We assume the reader already has a general understanding of writing computer code, and should with the help of the PAWN language guide (<http://ezesys.com/file/9>) be able to learn the specifics of the language.

Help with programming

We recognize that mastering the scripting features of the ezeio™ requires significant skill in analytics and programming. This manual and related documentation is not enough to learn how to program from scratch. You will need prior knowledge related to software development to be able to use the ezeio™ scripting efficiently.

If you have any prior experience with JavaScript, C, Java, Perl or similar languages, PAWN will look very familiar to you, and looking at our examples in this manual should get you going quickly.

eze System offers programming services on hourly basis. If you have a specific feature request, please contact us.

Capabilities

The PAWN language is powerful enough to create very complex functionality. Inputs, outputs, timers, schedules, alarm events and reporting features are available to the script through the custom functions defined in this manual, and the language can handle basic math, text strings, state machines and complex logic.

Compiled code can grow up to 64kB and use up to 6kB of RAM.

The ezeio™ executes over 100k instructions per second.

Event-driven design

Scripts written for the ezeio should be “event driven”. The ezeio™ defines a number of system events that are suitable as containers for user logic. There should be no “main loop” in the user code, as that would block the ability to process system events. Instead, design your code to react on the events, and as soon as you finish processing, return from the event call to allow the next event to be processed.

For example, let's say you want to monitor two inputs and set an output if the first input level exceeds the second. One recommended way to do this is:

```
@Tick(uptime)           // called every second
{
    if( GetInputValue(1) > GetInputValue(2) )
        SetOutput(1, 1);           // turn the output on
    else
        SetOutput(1, 0);           // turn the output off
}
```

The if-statement above will be processed once every second, and the output set according to the result of the comparison of the inputs.

Processing the whole condition will be very quick (less than a millisecond), so there will be plenty of time for other things to happen in your script.

String handling

The PAWN language defines strings as being either packed or unpacked. The ezeio™ supports only packed strings, meaning that each character is stored in a single byte.

Please refer to the PAWN documentation for further details.

Sleep-function

The **sleep()** function defined in PAWN accepts one integer parameter, and will suspend the script for the number of milliseconds given. Any system events that occurs while the script is suspended will be queued, and called in the order they occurred when the time expires. If the sleep was more than one second, only the first Tick-event will be processed. The queue can hold up to 32 events. If more events occur, they will be lost.

Please be aware of this when using the sleep() function.

Script function library

The ezeio™ support most of the language constructs defined in the PAWN language guide, including most of the functions for floating point math and the “proposed function library”.

In addition, the following functions are also supported:

Index of ezeio specific functions

Configuration interface functions

SetOutput(outputno, cadence, [cutoff]).....	92
GetOutputState(outputno).....	92
GetInputValue(inputno).....	93
SetInputValue(inputno, newvalue).....	93
GetInputCount(inputno).....	94
SetInputCount(inputno, newcount).....	94
GetInputState(inputno).....	94
GetScheduleState(scheduleno).....	95
GetSystemStatus(item).....	95

Calendar and time functions

GetSecond().....	96
GetMinute().....	96
GetHour().....	96
GetDay().....	96
GetMonth().....	96
GetYear().....	96
GetWeekday().....	96
SetTimer([timerno], timeoutms, repeat).....	97

Mathematical functions

Float:fabs(Float:value).....	98
fround(Float:value, [method]).....	98
Float:ffract(Float:value).....	98
Float:fsqrt(Float:value).....	98
Float:flog(Float:value, [Float:base]).....	98
Float:fpow(Float:value, Float:exponent).....	99
Float:fsin(Float:value).....	99
Float:fcos(Float:value).....	99
Float:ftan(Float:value).....	99
Float:fasin(Float:value).....	99
Float:facos(Float:value).....	99
Float:fatan(Float:value).....	99
Float:fatan2(Float:y, Float:x).....	99
random([max]).....	100
min(value1, value2).....	100
max(value1, value2).....	100
clamp(value, min, max).....	100

float2cell(Float:value).....	100
Float:cell2float(value).....	100
qsort(count, data[]).....	101
fqsort(count, Float:data[]).....	101
Language functions	
heapspace().....	102
numargs().....	102
getarg(argumentno, [index]).....	102
setarg(argumentno, [index], value).....	102
String functions	
tolower(character).....	103
toupper(character).....	103
strlen(string).....	103
strcpy(dest[], const source[], [maxlength]).....	103
strcmp(string1[], string2[], [ignorecase], [length]).....	104
strcat(dest[], source[], [maxlength]).....	104
strdel(string[], start, end).....	104
strfind(string[], sub[], [ignorecase], [index]).....	105
strins(dest[], src[], index, [maxlength]).....	105
strmid(dest[], source[], start, end, [maxlength]).....	105
strval(string, [index]).....	106
valstr(dest[], value).....	106
memcpy(dest[], source[], index, length, maxlength).....	106
strformat(dest[], maxlen, format[], [...]).....	107
Communication functions	
PDebug(format[], ...).....	108
ExtAPICall(wParam, lParam, format[], ...).....	108
ModbusSend(address, command, length, data[]).....	109
getThermostat(channel, address, register).....	109
setThermostat(channel, address, register, value, lock).....	110
SerialSend(length, data[]).....	110
Library functions	
GetTime(time[time_s], [UTC=false]).....	111
Linfit(Float:x[], Float:y[], ndata, &Float:a, &Float:b).....	111
SunPosition(UTCtime[time_s], Float:latitude, Float:longitude, &Float:elevation, &Float:azimuth).....	112
Float:Dewpoint(Float:RH, Float:T, [Fahrenheit=false]).....	113
Float:Enthalpy(Float:Alt, Float:RH, Float:Temp, [BTU=false]).....	113
System events	
@Tick(uptime).....	114
@Alarm(sourcetype, sourceid, alarmno).....	114
@Restore(sourcetype, sourceid, alarmno).....	114
@Timer(timerno).....	115
@ModbusReply(address, command, length, data[]).....	115
@SerialData(length, data[]).....	116

Configuration interface functions

SetOutput(outputno, cadence, [cutoff])

Use this function to directly control an output.

Parameter	Range	Description
outputno	1 – 40, required	Which output to control.
cadence	0 – 7, required	0 = off 1 = on 2 = 100ms on, 900ms off 3 = 1s on, 9s off 4 = 2s on, 59s off 5 = 100ms on, 100ms off (5Hz blink) 6 = 500ms on, 500ms off (1Hz blink) 7 = 1s on, 1s off (0.5Hz blink)
cutoff	0 – 65535, optional. Defaults to 0 (infinite)	Number of 1/10 th seconds to run the cadence. When the cutoff time expires the output will be turned off.

This function does not return a value.

Example: Turn on output 2 for 10 seconds:

```
SetOutput(2, 1, 100); // cutoff 100 is 10 seconds
```

GetOutputState(outputno)

Read the current status of an output

Parameter	Range	Description
outputno	1 – 40, required	Which output to check

The return value is 1 if the output is active (on) and 0 if the output is inactive (off).

Example: Check the status of output 8

```
if (GetOutputState(8))  
    // code to run if output is on
```

GetInputValue(inputno)

Reads the raw value from an input.

Parameter	Range	Description
inputno	1 – 40, required	Which input to read from.

The return value is the raw reading from the input. The unit depends on the type of input :

Standard 0-10V input: Return raw mV, 0=0V, 10000=10V
Current 0-30mA input: Return 29uA units, 0=0mA, 1000=2.9mA, 6820=20mA
Pulse type input: Readout is ms between pulses (see below)
MicroLAN/Modbus sensors: Depends on sensor type (see below)

Example: Read the value from input 2 and assign the value to a variable.

```
in2mV = GetInputValue (2) ;
```

Example: Read the value from a microlan temperature sensor on input 5 as 1/10th C

```
Celcius = (10*GetInputValue (5) ) /16-55 ;
```

Example: Read the value from a microlan temperature sensor on input 1 as F

```
F = (100*GetInputValue (1) ) /888-67 ;
```

Example: Read power as kW from a pulse meter with 500 pulses per kWh

```
kW = 7200/GetInputValue (8) ;
```

SetInputValue(inputno, newvalue)

Set an input raw value to the specified value

Parameter	Range	Description
inputno	1 – 40, required	Which input to set
newvalue	-2147483648 – 2147483647, required	New value

This function does not return a value.

Important: The input source must be configured as “Special/Software”.

Example: Set the value of input 12 to 3456

```
SetInputValue (12, 3456) ;
```

GetInputCount(inputno)

Reads the counter value from an input.

Parameter	Range	Description
inputno	1 – 40, required	Which input to read from.

The return value is the current counter value for the input.

Example: Read the counter value from input 2 and assign the value to a variable.

```
mycnt = GetInputCount(2) ;
```

SetInputCount(inputno, newcount)

Set an input counter counter to a new value.

Parameter	Range	Description
inputno	1 – 40, required	Which input to set
newcount	0 – 2147483647, required	New value of the counter

This function does not return a value.

Example: Set the counter of input 5 to 456

```
SetInputCount(5, 456) ;
```

GetInputState(inputno)

Reads the current alarm state from the input.

Parameter	Range	Description
inputno	1 – 40, required	Which input to read from.

The return value is a bitmap with four bits indicating the status of each of the four alarms for this input. Bit 0=first alarm, Bit 3=fourth alarm.

Example: check the state of the third alarm for input 5:

```
if(GetInputState(5) & 0x04) // 0x04 = binary 0100  
// Code to run if alarm was active
```

GetScheduleState(scheduleno)

Read the current status of a schedule

Parameter	Range	Description
scheduleno	1 – 20, required	Which schedule to read status from

The return value is 1 if the schedule is active, 0 if the schedule is inactive.

Example: Check the status of schedule 1

```
if (GetScheduleState (1) )
```

```
    // code to run if schedule is active
```

GetSystemStatus(item)

Read the current status of a system status

Parameter	Range	Description
item	See below	Identifies the system status item requested

Value	Description
sysLastHostMessage	Number of seconds since last good message from the host was received
sysGPSLat	GPS Latitude (x 1 000 000). Only valid with 3G module and GPS antenna.
sysGPSLong	GPS Longitude (x 1 000 000). Only valid with 3G module and GPS antenna.
sysGPSElev	GPS Elevation (x 1 000 000). Only valid with 3G module and GPS antenna.
sysMLOK	1 if any Microlan devices are detected. 0 if not.
sysEthOK	1 if a physical Ethernet connection is detected. 0 if not.
sysRFOK	1 if a short-range radio is detected. 0 if not.
sysGSM Avail	1 if GSM/3G modem is detected. 0 if no modem.
sysGSMSIMOK	1 if the SIM card is detected. 0 if not.
sysGSMOK	1 if the GSM modem is enrolled with the carrier network. 0 if not.
sysIP	The assigned local IP as a 32 bit word.
sysGSMActive	1 if the GSM link is being used for communication. 0 if not.
sysGSMRSSI	The RSSI (received signal strength) reported by the GSM module in dBm. -113 is really weak, -50 is perfect.
sysGSMMode	The communication mode of the GSM module. One of: 1 = GSM, 2 = GPRS, 3 = EDGE, 4 = WCDMA, 5 = HSDPA, 6 = HSUPA, 7 = HSPA
sysGMTOfs	Offset in seconds from GMT
sysEpochTime	Current time in Epoch (seconds)
sysUpTime	Number of seconds since ezeio reset last

Calendar and time functions

GetSecond()

Return the current second from the real-time clock.
This function does not have any parameters.
The return value is the current second, in the range 0-59.

GetMinute()

Return the current minute from the real-time clock.
This function does not have any parameters.
The return value is the current minute, in the range 0-59.

GetHour()

Return the current hour from the real-time clock.
This function does not have any parameters.
The return value is the current hour, in the range 0-23.

GetDay()

Return the current day of the month from the real-time clock.
This function does not have any parameters.
The return value is the current day, in the range 1-31.

GetMonth()

Return the current month from the real-time clock.
This function does not have any parameters.
The return value is the current month, in the range 1-12.

GetYear()

Return the current year from the real-time clock.
This function does not have any parameters.
The return value is the current year, in the range 2000-3000.

GetWeekday()

Return the current year from the real-time clock.
This function does not have any parameters.
The return value is the current weekday, in the range 0 (Monday) through 6 (Sunday).

SetTimer([timerno], timeoutms, repeat)

Set a millisecond timer. The timer will generate a @Timer event when the timeout is reached. Note that these timers are different from the timers in the configuration.

Parameter	Range	Description
timerno	1 – 4, optional	Which timer to set. If this parameter is omitted, the function will use the first timer that is not running.
timeoutms	0 or 1 to 2147483647, required	Number of milliseconds before generating the @Timer event. If this parameter is 0, the timer will be cancelled and the @Timer event will not be generated.
repeat	0, 1 (optional)	If set to 1, the timer will automatically reset and trip again. Defaults to 0.

Returns the timer number that was set, or 0 if no timer was set..

Example: Set timer 3 to expire in 1.5 seconds.

```
SetTimer(3, 1500);
```

Mathematical functions

Float:fabs(Float:value)

Return the absolute value of a floating point value

Parameter	Description
value	Value to return absolute value of

Returns the absolute value

Float:round(Float:value, [method])

Round a floating point value to an integer

Parameter	Description
value	The value to round
method	The rounding method to use. One of: floatround_round (default, rounds to nearest integer. 0.5 rounds up) floatround_floor (round down) floatround_ceil (round up) floatround_tozero (round down for positive values, round up of negative values)

Returns the value rounded off, as an integer.

Float:ffract(Float:value)

Return the fractional part of a number

Parameter	Description
value	The value to return the fractional part of.

Returns the fractional part of value.

Example: `ffract(3.14)` returns 0.14

Float:fsqrt(Float:value)

Return the square root of a value

Parameter	Description
value	The value to calculate the square root of

Returns the square root of the value.

Float:flog(Float:value, [Float:base])

Return the logarithm of a value

Parameter	Description
value	The value to return the logarithm of
base	Logarithmic base (optional, defaults to e, or 2.71828)

Float:fpow(Float:value, Float:exponent)

Raise a floating point value to a power

Parameter	Description
value	The value to raise
power	The exponent. May be 0 or negative.

Float:fsin(Float:value)

Return the sine of a value

Parameter	Description
value	The value to calculate sine of

Float:fcos(Float:value)

Return the cosine of a value

Parameter	Description
value	The value to calculate cosine of

Float:ftan(Float:value)

Return the tangent of a value

Parameter	Description
value	The value to calculate tangent of

Float:fasin(Float:value)

Return the reverse sine of a value

Parameter	Description
value	The value to calculate reverse sine of

Float:facos(Float:value)

Return the reverse cosine of a value

Parameter	Description
value	The value to calculate reverse cosine of

Float:fatan(Float:value)

Return the reverse tangent of a value

Parameter	Description
value	The value to calculate reverse tangent of

Float:fatan2(Float:y, Float:x)

Return the inverse circular tangent of y divided by x

Parameter	Description
y, x	coordinates

random([max])

Return a random value

Parameter	Description
Max (optional)	The max value of the random value, default to 65536.

Returns a random value between 0 and the given max.

If the max parameter is 0, the random value is between -2^{31} and $+2^{31}$.

min(value1, value2)

Return the smaller of value1 and value2.

Parameter	Description
value1, value2	The two values to compare

max(value1, value2)

Return the larger of value1 and value2.

Parameter	Description
value1, value2	The two values to compare

clamp(value, min, max)

Return the value, but no smaller than min, and no larger than max.

Parameter	Description
value	The value to clamp.
min	The smallest value to return.
max	The largest value to return.

float2cell(Float:value)

Return the Float value as a cell, using binary conversion (not converting it through its value, but just copying the bits).

This is useful in communication functions when parsing binary buffers containing float values.

Parameter	Description
value	The float value to return as a cell

Float:cell2float(value)

Return the cell value as a Float, using binary conversion (not converting it through its value, but just copying the bits).

This is useful in communication functions when parsing binary buffers containing float values.

Parameter	Description
value	The cell value to return as a Float.

qsort(count, data[])

Sorts the array in ascending order.

Parameter	Description
count	The number of values in the array (min 2, max 500)
data[]	Array of cell values to be sorted.

Returns 1 if successful. Returns 0 if there was an error

Example:

```
new a[5] = {45, 23, 89, 3, 7};  
qsort(5, a);
```

fqsort(count, Float:data[])

Sorts the array of float values in ascending order.

Parameter	Description
count	The number of values in the array (min 2, max 500)
Float:data[]	Array of float cell values to be sorted.

Returns 1 if successful. Returns 0 if there was an error

Example:

```
new Float:a[5] = {4.5, 23.3456, 0.89, 3.3, 77.7};  
fqsort(5, a);
```

Language functions

heapspace()

Return the size of the heap, in bytes.

This function does not have any parameters.

numargs()

Return the number of arguments in a function call.

This function does not have any parameters.

getarg(argumentno, [index])

Return one argument from a function call.

Parameter	Description
argumentno	The argument number to return
index (optional)	If the argument is an array, this is the index in the array (default to 0)

Returns the value of the argument.

setarg(argumentno, [index], value)

Set an argument value

Parameter	Description
argumentno	The argument number to return
index (optional)	If the argument is an array, this is the index in the array (default to 0)
value	The new value of the argument

This function does not return a value.

String functions

tolower(character)

Return the lowercase version of the character code.

Parameter	Description
character	The character to convert to lowercase

toupper(character)

Return the uppercase version of the character code.

Parameter	Description
character	The character to convert to uppercase

strlen(string)

Return the length of a string.

Parameter	Description
string	The string to compute the length of

strcpy(dest[], const source[], [maxlength])

Copy one string to a buffer.

Parameter	Description
dest	Destination buffer
source	The string that will be copied
maxlength (optional)	The max number of characters to copy (defaults to the length of the dest buffer.

strcmp(string1[], string2[], [ignorecase], [length])

Compare two strings

Parameter	Description
string1, string2	Two strings to compare
Ignorecase (optional)	If "true", case is ignored
Length (optional)	The max number of characters to compare

The return value is:

-1 if string1 comes before string2

0 if the strings are equal

1 if string1 comes after string2

strcat(dest[], source[], [maxlength])

Concatenate two strings

Parameter	Description
dest	The first part, and the destination buffer
source	The part that will be added to dest
Maxlength (optional)	The maximum length of the destination buffer (defaults to max size of dest)

Returns the length of dest after concatenation

strdel(string[], start, end)

Remove a number of characters from a string

Parameter	Description
string	The string to work on
start	The position of the first character to remove (starting at 0)
end	The position of the last character to remove. Must be equal to or larger than start.

strfind(string[], sub[], [ignorecase], [index])

Search for a substring within a string

Parameter	Description
string	The string to search in
sub	The string to search for
ignorecase (optional)	If set to true, case is ignored in the search. Defaults to false.
index (optional)	The position in string to start searching from (starting at 0), defaults to 0

strins(dest[], src[], index, [maxlength])

Insert a string into another string

Parameter	Description
dest	The buffer to work on
src	The string to insert into dest
index	The position in dest where the src buffer will be inserted
maxlength	The maximum permitted length of dest

strmid(dest[], source[], start, end, [maxlength])

Copy a section of one string to a buffer

Parameter	Description
dest	The destination buffer
src	The source string
start (optional)	The position of the first character in source to copy Defaults to 0
end (optional)	The position of the last character in source to copy (must be equal to or greater than start) Defaults to the last character in source
maxlength (optional)	The maximum size of dest, Defaults to the size of dest

strval(string, [index])

Evaluate a string and return an integer

Parameter	Description
string	The string to evaluate
index (optional)	The position in string to start evaluating from

Returns the integer value found in the string

valstr(dest[], value)

Convert an integer value to a string

Parameter	Description
dest	The destination buffer
value	The integer value to convert to text

Returns the number of characters stored in dest excluding the terminating 0

memcpy(dest[], source[], index, length, maxlength)

Copy bytes from one buffer to another

Parameter	Description
dest	The destination buffer
source	The source buffer
index	The position in the source buffer from which to start copying
length	The number of bytes to copy
maxlength (optional)	The maximum size of the dest-buffer. Defaults to the size of dest

strformat(dest[], maxlen, format[], [...])

Format a string and insert placeholders

Parameter	Description
dest	The destination buffer
maxlen	The maximum number of characters in the resulting buffer (defaults to the size of the dest buffer)
format	A string that describes the format of the result
...	The parameters for the placeholders

The format parameter is a string that may contain placeholders. The following placeholders are supported:

`%c` – a single character
`%d` – an integer
`%x` – an integer presented as lowercase hex
`%X` – an integer presented as uppercase hex
`%f` – a rational (floating point) number
`%s` – a string

Placeholders can be formatted with a number immediately following the %-sign. The number indicates the field width in characters, and will add spaces if needed. To pad with zeros instead, enter the field with preceded with a zero.

To output a percent character, enter “%%”

Example: “%5d” will output something like “ 123”. “%05d” will output something like “00123”.

Communication functions

PDebug(format[], ...)

Send a string to the debug output on the server

Parameter	Description
format	String with optional placeholders
...	Zero or more values to insert in the placeholders

This function requires a working server link. It sends a string to the debug output screen of the server. The string can be formatted according to standard C-style printf.

There is a throttling mechanism to prevent PDebug messages from saturating the communications link. If you send more than 100 PDebug, and the delay between the messages is shorter than 5 seconds, messages will be dropped.

ExtAPICall(wParam, lParam, format[], ...)

Generate an external call to a public server.

Parameter	Description
wParam	User defined 16 bit parameter
lParam	User defined 32 bit parameter
format	String with optional placeholders
...	Zero or more values to insert in the placeholders

This function requires a working server link and an active account with available API service. Each use of this function counts as one API call.

The parameters will be forwarded to a server URL defined under Configure-System-Ethernet settings-External Server URL (see page 57). The format of the data is determined by the URL type. The URL must begin with one of: http:// https:// json:// jsons:// xml:// or xmls://

The return value is 1 if the message was queued successfully, or 0 if the message buffer was full. Note that a return value of 1 does not mean the message was delivered.

ModbusSend(address, command, length, data[])

Send out a command to a Modbus device

Parameter	Description
address	The destination address of the modbus device (1-63)
command	The modbus command. One of: READ_COILS (0x01) READ_INPUTS (0x02) READ_REGISTERS (0x03) READ_INPUT_REGISTERS (0x04) WRITE_COIL (0x05) WRITE_REGISTER (0x06) READ_EXCEPTION (0x07) DIAGNOSTIC (0x08) WRITE_REGISTERS (0x10) READ_DEVICEID (0x2B)
length	Number of bytes to send after the command byte
data	The bytes to send. Refer to the modbus specification and your specific device manual for detailed information on the content of this parameter. Up to 10 bytes can be send in a single command call.

If a response is received as a result of this function, the @ModbusReply function will be called.

getThermostat(channel, address, register)

Retrieve the status of a single thermostat.

Parameter	Description
channel	The communication port where the thermostat is connected. One of: MICROLAN MODBUS LOCALRF Note – currently only MODBUS is supported.
address	The bus address of the thermostat
register	The thermostat register to retrieve. One of: setHeat Current heating setpoint (1/10th degrees) setCool Current cooling setpoint (1/10th degrees) adjHeat Heating setpoint override adjCool Cooling setpoint override minAdjust Override minutes remaining Lock 1=thermostat UI is locked, 0=unlocked ScheduleMode 0=Standard, 1=Alternate mode SysMode 0=off, 1=heat, 2=cool, 3=auto heat/cool FanMode 0=auto, 1=on Temp Ambient temperature (1/10th degrees) Fan Fan status 0=off, 1=running Call Call status: 0=off, 1=heating, 2=cooling

The return value depends on the “register” parameter as described above.

Example to read the current ambient temperature from modbus thermostat with address 8:

```
new t = getThermostat(MODBUS, 8, Temp);
```

At 78.9F, the value of t will be 789.

setThermostat(channel, address, register, value, lock)

Control a single thermostat.

Parameter	Description
channel	The communication port where the thermostat is connected. One of: MICROLAN MODBUS LOCALRF Note – currently only MODBUS is supported.
address	The bus address of the thermostat
register	The thermostat register to change. One of: setHeat Current heating setpoint (1/10th degrees) setCool Current cooling setpoint (1/10th degrees) adjHeat Heating setpoint override adjCool Cooling setpoint override minAdjust Override minutes remaining ScheduleMode 0=Standard, 1=Alternate mode SysMode 0=off, 1=heat, 2=cool, 3=auto heat/cool FanMode 0=auto, 1=on
lock	Boolean to determine if the change should lock the thermostat from user input. false = leave thermostat open to user input true = lock thermostat screen

This function do not return a value.

SerialSend(length, data[])

Send data on the serial port.

Note that the port needs to be in “Custom protocol” mode. See page 58.

Parameter	Description
length	The number of bytes to be sent
data[]	A buffer of bytes that will be sent.

The data will be sent with the bitrate configured on the system settings screen. Data is always sent as 8 databits, 2 stopbits and no parity.

Received data is handled by the **@SerialData** event handler, see page 116.

This function will return 1 if the data was buffered for transmission.

It returns 0 if the transmit buffer is full and no data was buffered.

Library functions

Library functions are declared as stock functions and will be included automatically if referenced from the user script. Using a stock function will add significantly to the size of your code, so make sure your script does not grow beyond 64k compiled.

GetTime(time[time_s], [UTC=false])

This function will fill in the supplied time[time_s] structure with current local date and time. If UTC is set to true, UTC time is returned.

Parameter	Description
time[time_s]	Structure for date and time, with the following properties: ti_year, ti_month, ti_day, ti_hour, ti_minute, ti_second, ti_wday
UTC (optional)	Flag (true or false). If set to true, the returned time is UTC time instead of local time.

Example usage:

```
new t[time_s];
GetTime(t);
PDebug("Time %d:%d:%d", t[ti_hour], t[ti_minute], t[ti_second]);
```

Linfit(Float:x[], Float:y[], ndata, &Float:a, &Float:b)

This function calculates a best-fit straight line using the least squares method

Parameter	Description
x[], y[]	Coordinated to use in calculation
ndata	How many points to expect in x[] and y[] arrays.
a	Line offset
b	Line slope

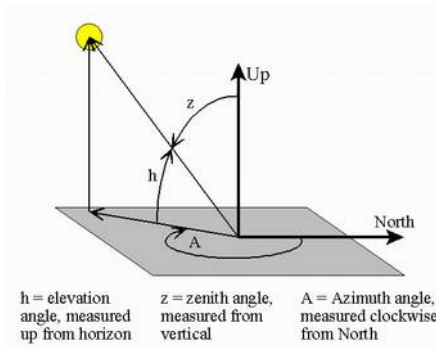
SunPosition(UTCtime[time_s], Float:latitude, Float:longitude, &Float:elevation, &Float:azimuth)

This function returns the relative position of the sun, given time and position.

Parameter	Description
UTCtime[time_s]	Structure with time and date. Note that this needs to be UTC time.
latitude	Latitude of observation point, in degrees from equator. Positive numbers are north.
longitude	Longitude of observation point, in degrees from Greenwich meridian. Positive numbers are east.
&elevation	The function will set this variable with the elevation of the sun above the horizon, in degrees. Negative numbers mean that the sun is below the equator.
&azimuth	The function will set this variable with the azimuth of the sun, in degrees. 0 is north, 90 is east, 180 is south and 270 is west.

Example usage:

```
new t[time_s];
new Float:elev, Float:azim;
GetTime(t, true); // Get UTC time
SunPosition(t, 61.191, -149.802, elev, azim); // Anchorage, AK
PDebug("Elevation:%f, Azimuth:%f", elev, azim);
```



Float:Dewpoint(Float:RH, Float:T, [Fahrenheit=false])

This function calculates the dew point.

Parameter	Description
RH	Relative humidity, between 0.00 and 1.00
Temp	Temperature, degrees Celcius (or Farenheit if Farenheit parameter is true)
Fahrenheit (optional)	false (default) input and output is in degrees Celcius. true input and output is in degrees Farenheit.

Example usage:

```
// Read temperature in F from MicroLan sensor on IN6
new Float:T = GetInputValue(6) / 8.8888 - 67.0;

// Read RH in percent from MicroLan sensor on IN7
new Float:RH = (GetInputValue(7) / 5000.0 - 0.16)/0.0062;

// Calculate Dewpoint in F
new Float:DP = Dewpoint(RH/100.0, T, 1);

// Put result in IN13
SetInputValue(13, fround(DP * 100.0));
```

Float:Enthalpy(Float:Alt, Float:RH, Float:Temp, [BTU=false])

This function calculates the energy content in moist air.

Parameter	Description
Alt	Altitude in meters
RH	Relative humidity, between 0 and 1
Temp	Temperature, degrees Celcius
BTU (optional)	false (default) returns value in SI units (kJ/kg dry air). true returns value in English units (Btu/lb dry air)

System events

@Tick(uptime)

The @Tick function is called automatically once every second.

Parameter	Description
uptime	Number of seconds since last system reset.

@Alarm(sourcetype, sourceid, alarmno)

The @Alarm function is called each time an alarm event is generated from the configuration. The function is called even if there are no actions configured, and regardless of conditions on any existing actions.

Parameter	Description
sourcetype	Indicates the source of the alarm event, and is one of: SOURCE_INPUT (1) – source is an input SOURCE_SCHEDULE (2) – source is a schedule SOURCE_TIMER (3)– source is a timer
sourceid	Indicates the source index, e.g. the input number (1-40), schedule number (1-20) or timer number (1-20)
alarmno	Indicates the alarm number (1-4)

@Restore(sourcetype, sourceid, alarmno)

The @Restore function is called each time a restore event is generated from the configuration. The function is called even if there are no actions configured, and regardless of conditions on any existing actions.

Parameter	Description
sourcetype	Indicates the source of the restore event, and is one of: SOURCE_INPUT (1) – source is an input SOURCE_SCHEDULE (2) – source is a schedule SOURCE_TIMER (3)– source is a timer
sourceid	Indicates the source index, e.g. the input number (1-40), schedule number (1-20) or timer number (1-20)
alarmno	Indicates the alarm number (1-4)

@Timer(timerno)

The @Timer function is called when a millisecond timer set with the “SetTimer” function expires. Note that the millisecond times has nothing to do with the timers in the configuration settings.

Parameter	Description
timerno	The number of the timer that expired (1-4)

@ModbusReply(address, command, length, data[])

This function is called when a reply is received from a Modbus device as a result of a call to ModbusSend.

Parameter	Description
address	The modbus device address that replied
command	The modbus command number (see ModbusSend)
length	Number of bytes in the data[] array.
data[]	A byte-array with the received data as it was received over ModBus. Note that the array starts with the byte after the command byte. The max number of bytes that can be received in one call is 20.

Example:

```
mbReadReg(deviceadr, regno, regcount)
{
    new b[4 char];
    regno = (regno%10000)-1;           // Modbus address mapping
    b{0} = (regno>>8)&0xFF;           // Build command buffer
    b{1} = regno&0xFF;
    b{2} = (regcount>>8)&0xFF;
    b{3} = regcount&0xFF;
    ModbusSend(deviceadr, READ_REGISTERS, 4, b); // Queue to send
}

@Tick(uptime)
{
    if((uptime%20)==0) { // every 20 seconds..
        mbReadReg(5, 40047, 1); // Request reg 40047 from device 5
    }
}

@ModbusReply(address, command, length, data[])
{
    new x;
    // Make sure this is a reply to the above query
    if((address==5) && (ModbusCommand:command==READ_REGISTERS)) {
        x = (data{1}<<8) | data{2}; // Extract a 16 bit value
        PDebug("Read %d", x); // Print on debug console
    }
}
```

@SerialData(length, data[])

The @SerialData function is called when a packet of bytes has been received on the serial port. A packet is considered complete when there is at least five byte-times of no data after any byte(s).

Note that this only applies when the serial port is in Custom Protocol mode (see page 58).

Parameter	Description
length	The number of bytes received
data[]	A buffer that holds the data

Specifications

ezeio™

Size	153 x 100 x 38 mm (6.0" x 3.9" x 1.5") mounting ears extend 15mm (0.6") on each side. Hole centers are 166mm (6.5") apart. Allow at least 40mm (1.6") margin for connectors
Weight	Approx 220g (0.5lb)
Power	8-25VDC, <2W average, 7W peak
Operating environment	0-50°C (32-120°F)
Hardwire inputs	4 inputs on screw terminal: 0-10V, 10mV resolution, >70kΩ impedance 0-30mA, 32uA resolution, 100Ω current sense resistor
Hardwire outputs	2 relay outputs with screw connections: Form C (1 pole switching) Max 2A, 50V load
Other connections	Ethernet, TP 10/100, RJ45 MicroLAN, RJ12 Serial RS485, RJ45-jack GSM antenna SMA (optional)
Expandability	Up to 40 sensor inputs total Up to 40 outputs total
MicroLan	Max 20 MicroLAN devices supported Active pullup on data wire 5V and raw DC provided RJ12 jack compliant to Dallas connector standard Max 50m (150ft) network length
DC output	Unregulated output, max 200mA ("+" terminals) Regulated 5V output, max 100mA ("5" terminal).
Serial	RS485/Modbus RTU, bidirectional 19200bps
GSM (optional)	GSM/GPRS/EDGE 850/900/1800/1900MHz US 3G: 850/1900MHz EU/AU 3G: 900/2100MHz GPS 16ch, active antenna

Configuration and programming

Logging	Individual logging on each input. 10s to 1h interval. Automatically communicated and stored on redundant servers. 8000 samples/channel local buffer.
Input triggers	Up to four alarms per input, each with alarm/restore thresholds and separate holdoff times. Each alarm and restore can trip up to four separate actions, such as sending messages, controlling outputs or counters.
Schedules	Up to 20 schedules, each with four intervals and flags for each day in the week. Up to four actions for each schedule on entry/exit of an interval.
Timers	Up to 20 timers, each can be set to repeat hourly, daily, weekly or monthly. Each timer can trip up to four actions.
Scripts (optional)	Up to 64kB compiled script code, with 6kB of RAM. Extensive function library with support for floating point math, string manipulation and communication functions.

Server Communication

Configuration	Automatic, DHCP
Host protocol	IP/UDP, proprietary encrypted payload
Port	Outbound port UDP 8844 Inbound port UDP 28672-32767 (random per session)
Encryption	128 bits, unique key per ezeio
Traffic	Typical 5-15MB / month (depends on usage)
Local buffer	8000 samples per channel, non-volatile

Warranty

Manufacturers warranty statement

All ezeio™ and accessories (the products) manufactured by eze System, Inc. are warranted for two years against manufacturing issues. The warranty is void if the products have been physically altered or subjected to conditions beyond the physical limits of the devices.

The company gives only the warranties specifically stated herein and waives all implied warranties, including but not limited to warranties for merchantability and for fitness for a particular use. The company's obligation for a breach of a warranty is to repair or replace the product.

Liability disclaimer

eze System is not liable for any injury or mishap sustained by the use of the product. Please consult with a qualified dealer/installer before placing the product in service. Installation and use of the product must comply with local laws and regulations. The end user of the products acknowledges risks and waives any and all claims against eze System, Inc. and any of its agents. eze System is not responsible for any applications of its products or the suitability of its products for any application. eze System only warrants that the product will log, monitor and control that device if it is properly installed, configured and if the Internet connection has been properly initiated and maintained. Company is never responsible for any losses incurred by failure of a user designed system which results in any monetary loss, damage, injury or loss of life. The company's products are not designed to be fail-safe or fool-proof and should not be used in safety critical applications.

Standards compliance

Applicable standards



Part 15 Subpart B Sections 15.107 and 15.109

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

ezeio-W and ezeio-GW models contain FCC ID: X7J-A11072401

ezeio-G and ezeio-GW models contain FCC ID: UDV-1103022011008



2004/108/EC (EMC), 73/23/EC (LVD)

This equipment meets or exceeds the requirements of the following standards: EN55022 (2010), EN55024 (2010), EN61000-3-2 (2006) +A1 +A2, EN61000-3-3 (2008).



AS/NZS CISPR 22

Tested to comply to the Australian/New Zealand requirements for information technology equipment.

California Safe Drinking Water and Toxic Enforcement Act of 1986:

WARNING: This product contains chemicals known to the State of California to cause cancer and birth defects or other reproductive harm.

Access your ezeio at:
www.ezecontrol.com

ezeio email:
{serial}@ezecontrol.com
subject doesn't matter, first line of message is password

ezeio SMS:
US: +1 916 281-9001
UK/Global: +44 7937 985 875
SE: +46 769 439 907
AU: +61 448 838 189
ezeioezeio serial, password, command

Visit the eze System, Inc. website at
www.ezesys.com

The eze trademark, ezeio and the eze system design are property of eze System, Inc.
Any other trademarks referenced are properties of their respective owner.

© eze System, Inc 2008-2017
www.ezesys.com