

杭州微碟科技有限公司

ER04 用户手册

V1.0

2016-3-16

目录

| | |
|----------------------|----|
| 简介 | 2 |
| 电气规格 | 2 |
| LED 指示说明 | 2 |
| 继电器接线说明 | 2 |
| 基本应用 | 3 |
| 感性负载保护 | 3 |
| 参数配置 | 4 |
| 默认参数 | 4 |
| ERSetting 简要说明 | 4 |
| 参数说明 | 5 |
| 断电记忆 | 5 |
| HTTP API | 5 |
| 继电器 | 5 |
| 批量获取继电器状态 | 6 |
| 控制器参数 | 8 |
| MODBUS TCP | 15 |
| 读线圈 | 15 |
| 写线圈 | 15 |
| 强制写多点线圈 | 15 |

简介

ER04 以太网 IO 控制器包括 4 路继电器输出，支持 9~35V 宽电压供电。提供点动、断电恢复功能。提供一个 Restful 的 HTTP API 接口和一个 MODBUS TCP 接口。

电气规格

工作环境温度：-20~70 度

工作环境湿度：小于 90%RH

继电器输出功率(单路)：10A 277VAC 10A 28VDC

外型尺寸：115*90*40mm

适配器电压:DC9~35V（宽电压）

网络速率：10Mbps

网络协议：IEEE802.3

LED 指示说明

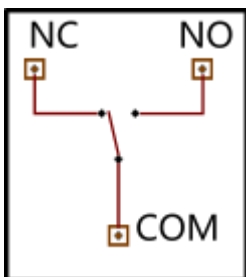
RELAY 为继电器输出指示灯，当继电器闭合相应的指示灯会亮起。

RUN 为工作指示灯，当控制器正常工作时，RUN 指示会按 1 秒的频繁闪烁。当按下 RESET 后，该指示灯会快闪，当按下 10 秒以上后，该指示灯会长亮，并完成复位。

POWER 为电源指示灯，插入电源后该指示灯会亮起。

继电器接线说明

ER04 系列控制器单个继电器包括常开、常闭二个触点，逻辑图如下所示：



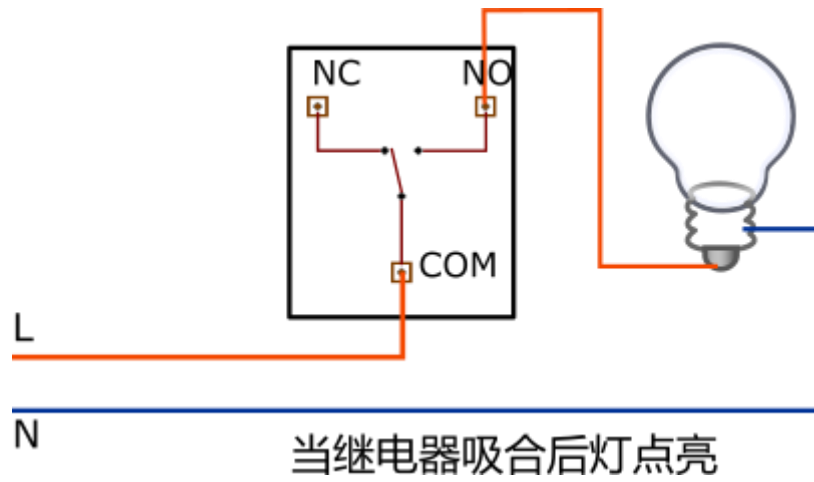
NC：常闭

NO：常开

COM：公共端

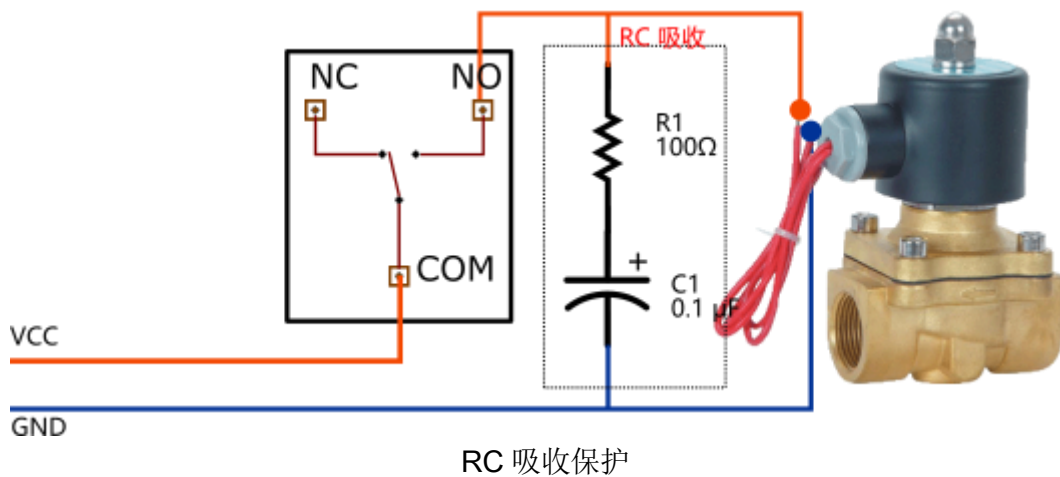
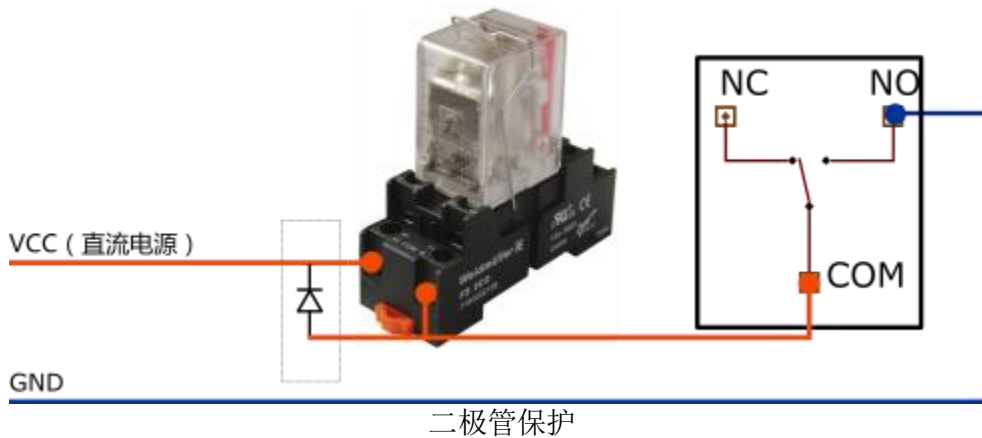
当继电器释放时，NC 和 COM 连通；当继电器吸合后，NO 和 COM 连。

基本应用



感性负载保护

当使用继电器控制交流负载时，应该添加触点保护，对于直流负载可以使用续流二极管、对于交流负载推荐使用 RC 吸收电路。如下图所示：



参数配置

控制器可以通过 ERSetting 软件进行配置，也可以通过 HTTP API 进行配置。使用 HTTP API 进行配置请参阅 HTTP API 一节。本节讲述如何使用 ERSetting 进行配置。

默认参数

ER 系列的默认 IP 为 192.168.1.25，默认密码为 123456。通过按下 RESET 按钮 10 秒可以重置控制器参数。

ERSetting 简要说明

启动 ERSetting 后在控制器 IP 处输入控制器 IP 地址，输入控制器访问密码后点击“连接”按钮。

软件会自动载入当前控制器的配置参数。

完成参数修改后点击“提交”按钮，保存成功后，软件弹出提示框。

注意：任何参数的修改必须对控制器断电重启后才能生效。

参数说明

除 IP，掩码，网关和密码等基本参数外，控制器还包括 Modbus，断电记忆等特别参数，本节具体讲述这些参数的作用。

Modbus

当勾选后 Modbus 功能会被起用，当取消勾选后 Modbus 功能会被禁止。

断电记忆

当勾选后控制器会记录断电时的继电器状态，并在供电恢复时自动将继电器状态回到断电前。

HTTP API

ER 系列的 HTTP API 是完全 Restful 形式，支持 HTTP Basic 认证，使用 GET 方法获取状态和参数，使用 PUT 或 POST 方法设置状态和参数。

ER04 的 HTTP API 参考 draft-ipso-app-framework-04，对单一资源的操作返回数据格式为 text/plain，对批量资源的参考 draft-jennings-senml-10，访问返回 JSON 格式数据。

ER 系列的默认 IP 为 192.168.1.25，HTTP Basic 认证的用户名为（并且仅有）Admin，默认密码为 123456，本节将使用这些默认参数进行讲述。

在本节中我们使用 CURL 这一命令行工具进行演示。

继电器

资源路径: /gpio/dout/(1...4)，资源 GET 为获取状态，PUT 为控制输出

url 示例: <http://192.168.1.25/gpio/dout/1>

设置第 1 路继电器闭合

```
elephant@UbuntuServerOne: ~  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/gpio/dout/1" -X PUT -d 1  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> PUT /gpio/dout/1 HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
> Content-Length: 1  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 1 out of 1 bytes  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$
```

获取第 1 路继电器状态

```
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/gpio/dout/1"  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> GET /gpio/dout/1 HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
>  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
1  
elephant@UbuntuServerOne:~$
```

批量获取继电器状态

批量获取继电器状态只支持 GET 模式，返回的数据格式为 JSON 格式。

资源路径: /gpio/dout

url 示例: <http://192.168.1.25/gpio/dout>

```
elephant@UbuntuServerOne: ~  
<  
* Closing connection 0  
1  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/gpio/dout"  
* Hostname was NOT found in DNS cache  
* Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> GET /gpio/dout HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
>  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: application/senml+json  
< Pragma: no-cache  
<  
* Closing connection 0  
{ "bn": "http://192.168.1.25/gpio/din/", "e": [{"n": "1", "bv": 1}, {"n": "2", "bv":  
1}, {"n": "3", "bv": 1}, {"n": "4", "bv": 1}] }  
elephant@UbuntuServerOne:~$
```

点动操作

资源路径: [/opt/ts/\(1...4\)](http://192.168.1.25/opt/ts/1), 该资源仅支持 POST 或 PUT 方式

url 救命: <http://192.168.1.25/opt/ts/1>

设置第 1 路继电器, 闭合 2 秒后释放


```
elephant@UbuntuServerOne: ~  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/opt/ts/1" -X PUT -d 2  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> PUT /opt/ts/1 HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
> Content-Length: 1  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 1 out of 1 bytes  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$
```

控制器参数

注意，所有对控制器参数的修改必须断电重启才能生效。

IP

资源路径: `/cfg/network/ip`

url 示例: <http://192.168.1.25/cfg/network/ip>

修改控制器 IP 为 10.10.1.2

```
elephant@UbuntuServerOne: ~
0},{ "n": "3", "bv": 0},{ "n": "4", "bv": 0}]]
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25
/cfg/network/ip" -X PUT -d "10.10.1.2"
* Hostname was NOT found in DNS cache
*   Trying 192.168.1.25...
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)
* Server auth using Basic with user 'admin'
> PUT /cfg/network/ip HTTP/1.1
> Authorization: Basic YWRtaW46MTIzNDU2
> User-Agent: curl/7.35.0
> Host: 192.168.1.25
> Accept: */*
> Content-Length: 9
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 9 out of 9 bytes
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: text/plain
< Pragma: no-cache
<
* Closing connection 0
elephant@UbuntuServerOne:~$
```

获取控制器配置的 IP 地址

```
elephant@UbuntuServerOne: ~
< Pragma: no-cache
<
* Closing connection 0
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25
/cfg/network/ip"
* Hostname was NOT found in DNS cache
*   Trying 192.168.1.25...
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)
* Server auth using Basic with user 'admin'
> GET /cfg/network/ip HTTP/1.1
> Authorization: Basic YWRtaW46MTIzNDU2
> User-Agent: curl/7.35.0
> Host: 192.168.1.25
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Content-Type: text/plain
< Pragma: no-cache
<
* Closing connection 0
10.10.1.2
elephant@UbuntuServerOne:~$
```

Netmask

资源路径: /cfg/network/netmask

url 示例: <http://192.168.1.25/cfg/network/netmask>

修改控制器的掩码地址为 255.255.0.0

```
elephant@UbuntuServerOne: ~  
10.10.1.2  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/cfg/network/netmask" -X PUT -d "255.255.0.0"  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> PUT /cfg/network/netmask HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
> Content-Length: 11  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 11 out of 11 bytes  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$
```

获取控制器配置的掩码地址

```
elephant@UbuntuServerOne: ~  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/cfg/network/netmask"  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> GET /cfg/network/netmask HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
>  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
255.255.0.0  
elephant@UbuntuServerOne:~$
```

Gateway

资源路径: /cfg/network/gw

url 示例: <http://192.168.1.25/cfg/network/gw>

修改控制器的网关地址为 10.10.1.1

```
elephant@UbuntuServerOne: ~  
255.255.0.0  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/cfg/network/gw" -X PUT -d "10.10.1.1"  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> PUT /cfg/network/gw HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
> Content-Length: 9  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 9 out of 9 bytes  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$
```

获取当前控制器配置的网关地址

```
elephant@UbuntuServerOne: ~  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/cfg/network/gw"  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> GET /cfg/network/gw HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
>  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
10.10.1.1  
elephant@UbuntuServerOne:~$
```

Modbus

资源路径: /cfg/modbus

url 示例: <http://192.168.1.25/cfg/modbus>

获取当前控制器的 modbus 配置

```
elephant@UbuntuServerOne: ~  
< Pragma: no-cache  
<  
* Closing connection 0  
10.10.1.1  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25/cfg/modbus"  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> GET /cfg/modbus HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
>  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
1  
elephant@UbuntuServerOne:~$
```

修改当前控制器的 modbus 为禁止

```
1  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25/cfg/modbus" -X PUT -d 0  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> PUT /cfg/modbus HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
> Content-Length: 1  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 1 out of 1 bytes  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$
```

rollback

资源路径: /cfg/rollback

资源参数：1 或者 0,1 表示启用断电记忆，0 表示关闭。

url 示例：<http://192.168.1.25/cfg/rollback>

获取当前控制器的 rollback 参数

```
elephant@UbuntuServerOne: ~  
< Pragma: no-cache  
<  
* Closing connection 0  
  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25/cfg/rollback"  
* Hostname was NOT found in DNS cache  
* Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> GET /cfg/rollback HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
>  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
1  
elephant@UbuntuServerOne:~$
```

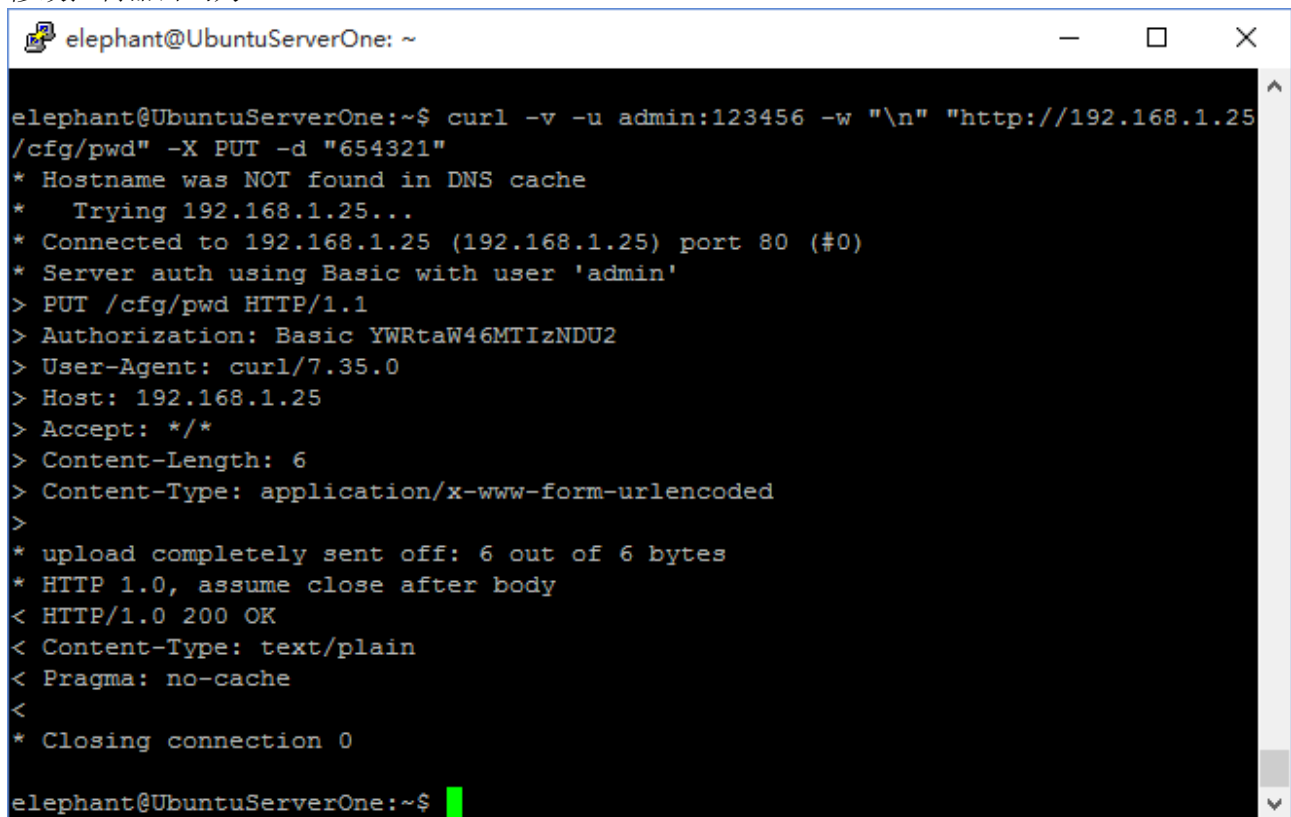
修改控制器断电记忆为关闭

```
elephant@UbuntuServerOne: ~  
1  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25/cfg/rollback" -X PUT -d 0  
* Hostname was NOT found in DNS cache  
* Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> PUT /cfg/rollback HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
> Content-Length: 1  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 1 out of 1 bytes  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$
```

密码

资源路径: /cfg/pwd

资源参数: 字符和数字组成, 最大为 6 个字符和数字的组合
修改控制器密码为 654321



```
elephant@UbuntuServerOne: ~  
elephant@UbuntuServerOne:~$ curl -v -u admin:123456 -w "\n" "http://192.168.1.25  
/cfg/pwd" -X PUT -d "654321"  
* Hostname was NOT found in DNS cache  
*   Trying 192.168.1.25...  
* Connected to 192.168.1.25 (192.168.1.25) port 80 (#0)  
* Server auth using Basic with user 'admin'  
> PUT /cfg/pwd HTTP/1.1  
> Authorization: Basic YWRtaW46MTIzNDU2  
> User-Agent: curl/7.35.0  
> Host: 192.168.1.25  
> Accept: */*  
> Content-Length: 6  
> Content-Type: application/x-www-form-urlencoded  
>  
* upload completely sent off: 6 out of 6 bytes  
* HTTP 1.0, assume close after body  
< HTTP/1.0 200 OK  
< Content-Type: text/plain  
< Pragma: no-cache  
<  
* Closing connection 0  
elephant@UbuntuServerOne:~$
```

MODBUS TCP

ER 系列支持读线圈（FC1）、写线圈（FC5）、强制写多点线圈（FC15）三个功能码。关于 MODBUS-TCP 的详细信息请参考“开放型 Modbus/TCP 规范”，本手册不再做重复描述。为方便开发者，本手册对常用的命令进行示例说明。注意，以下示例中的数据为 16 进制。

读线圈

读取四个继电器的状态

发送：00 00 00 00 00 06 01 01 00 00 00 04

返回：00 00 00 00 00 04 01 01 01 0F

返回数据中的 0F（转换成二进制为 0b00001111）表示当前四个继电器的状态，最后一位为第一个继电器。

写线圈

将继电器 1 闭合

发送：00 00 00 00 00 06 01 05 00 00 FF 00

返回：00 00 00 00 00 06 01 05 00 00 FF 00

将继电器 2 闭合

发送：00 00 00 00 00 06 01 05 00 01 FF 00

返回：00 00 00 00 00 06 01 05 00 01 FF 00

将继电器 3 释放

发送：00 00 00 00 00 06 01 05 00 02 00 00

返回：00 00 00 00 00 06 01 05 00 02 00 00

将继电器 4 释放

发送：00 00 00 00 00 06 01 05 00 03 00 00

返回：00 00 00 00 00 06 01 05 00 03 00 00

强制写多点线圈

将四个继电器全部释放

发送：00 00 00 00 00 08 01 0F 00 00 00 04 01 00

返回：00 00 00 00 00 06 01 0F 00 00 00 04

将 1 号和 4 号继电器闭合，2 号和 3 号释放

发送 : 00 00 00 00 00 08 01 0F 00 00 00 04 01 09

返回 : 00 00 00 00 00 06 01 0F 00 00 00 04

发送数据中的 09（转换成二进制为 0b00001001）表示四个继电器的状态，最后一位为第一个继电器。