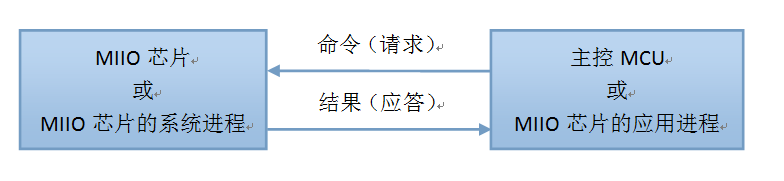
**概述**

MIIO芯片提供一套可读的串口文本命令，供外部芯片调用，一般适用于MIIO模块不是主控芯片的情况，即MIIO芯片只负责网络通讯，而不关心业务逻辑。

文本命令采用“一问一答”的方式，每条命令，无论执行成功与否，都会返回结果或错误提示。 所有文本命令，如无特殊说明，都使用小写字母。 下文中，示例里的“↑”和“↓”分别代表命令和结果。有时，也称二者为“上行”信息和“下行”信息。因为在涉及到云通信的命令里，“上”代表发往云端，“下”代表从云端发来。



一个典型的上行例子（温度传感器上报温度）：

1. 传感器探知当前温度为26度
2. 传感器调用MIIO芯片的props命令，上报温度属性：props temp 26

一个典型的下行例子（手机app打开电源开关）：

1. 手机app发出打开开关指令
2. MIIO芯片接收到app的下行消息，并翻译为文本命令：power\_on
3. 主控MCU不停的使用get\_down命令轮询MIIO芯片，得到了power\_on命令
4. 主控MCU完成电源打开的操作
5. 主控MCU调用result ”ok“，告知MIIO芯片成功完成
6. MIIO芯片回复ok，告知结果已收到

**芯片接口命令**

串口层默认配置为115200,8,1，N,N

**props**

* 参数：<prop\_name\_1> <value\_1> <prop\_name\_2> <value\_2> ...
* 返回：ok 或 error
* 说明：参数至少要有一个名值对。
* 举例：

↑props temp 21 hum 50 location "home"

↓ok

↑props location office (格式非法)

↓error

* 限制：属性名的字符个数不得多于31个（可等于31）

**event**

* 参数：<event\_name> <value\_1> <value\_2>...
* 返回：ok 或 error
* 说明：发送上行事件。
* 举例：

↑event overheat 170

↓ok

↑event button\_pressed

↓ok

↑event fault "motor stuck" "sensor error"

↓ok

↑event fault motor stuck (格式非法)

↓error

↑event fault "motor stuck"

↓ok ... (短时间内大量event）

↑event fault "motor stuck" （此时，事件队列已满）

↓error

* 限制：

1. 事件名的字符个数不得多于31个（可等于31）
2. 如果MIIO芯片断网，事件会在芯片里缓存最长10分钟，如果这段时间内网络未恢复正常，则事件会被丢弃。
3. MIIO芯片最多缓存8条事件，如果短时间内大量调用event命令，则会造成缓存列表填满，后续事件会被丢弃

**get\_down**

* 参数：无
* 返回：down <method\_name> <arg1>,<arg2>,<arg3> ...
* 说明：获取下行指令。如果有下行方法，则返回该方法，如没有，则返回none。如果对上一个方法还没有给出结果，则返回error。MCU在获得下行方法后，有1s时间处理，并用result/error命令上传结果。超过1s后，则wifi模块认为调用失败。
* 返回值说明：如果没有下行命令，则返回down none。如果有下行命令，则返回命令名、命令参数（如果有参数）。命令名和参数之间用空格隔开，多个参数之间用逗号隔开。参数可以是双引号括起的字符串，或是数字。
* 举例：

↑get\_down

↓down none

↑get\_down

↓down power\_on "light\_on",60,0,1

↑result "ok"

↓ok

↑get\_down

↓down power\_off

↑get\_down

↓error

**result**

* 参数：<value1> <value2> ...
* 返回：成功返回ok，失败返回error
* 说明：发送下行指令的执行结果，如果有多个返回值，则以逗号分开。发送成功后，返回ok。
* 举例：

↑result 123 "abc"

↓ok

**error**

* 参数：<message> <error\_code>
* 返回：成功返回ok，失败返回error
* 说明：如果下行指令执行出错，可用该命令发送错误信息。发送成功后，返回MIIO芯片回复ok。message必须是双引号括起的字符串，错误码必须是-9999 到 -5000之间的整数（含边界值）。
* 举例：

↑error "memory error" -5003

↓ok

↑error "stuck" -5001

↓ok

**json\_get\_down**

* 参数：无
* 返回：down <json\_string>
* 说明：获取Json格式的下行指令。如果有下行方法，则返回该方法，如没有，则返回none。如果对上一个方法还没有给出结果，则返回error。MCU在获得下行方法后，有1s时间处理，并用result/error命令上传结果。超过1s后，则wifi模块认为调用失败。
* 返回值说明：如果没有下行命令，则返回down none。
* 举例：

↑json\_get\_down

↓down none

↑json\_get\_down

↓down{“method":"power\_on","params":["light\_on",60,0,1],"id":123456}

↑json\_ack {"result":"ok"}

↓ok

**json\_ack**

* 参数：<json\_string>
* 返回：成功返回ok，失败返回error
* 说明：json\_ack 与json\_get\_down相对应，用来回复下行指令执行的结果。
* 举例：

↑json\_ack {"id":1,"error":{"code":-3,"message":"device timeout"}}

↓ok

↑json\_ack {"id":1,"result":1}

↓ok

**json\_send**

* 参数：<json\_string>
* 返回：ok
* 说明：用来以json格式发出上行消息。
* 举例：

↑json\_send{"method":"props","params":{"prop1":100,"prop2":"string"},"id":123}

↓ok

**log**

* 参数：<log\_name> <arg1> <arg2> <arg3> ...
* 返回：ok
* 说明：用于上报以记录为目的的数据
* 举例：

↑log pure\_water\_record "2015-5-14 19:27:31" 210 33 500

↓ok

**call**

* 参数：<method> <params>
* 返回：ok
* 说明：用于调用云端的方法，其结果以下行的return命令被get\_down获取
* 举例：

↑call get\_start\_time ↓ok

↑get\_down

↓down none ...(一段时间后）

↑get\_down

↓down result“1434446397” （成功） 或

↓down error“xxxxxxxxx” （失败）

**version**

* 参数：无
* 返回：版本号，当前为2
* 说明：串口协议（本协议）的版本
* 举例：

↑version

↓2

**net**

* 参数：无
* 返回： offline 或 local 或 cloud 或 updating 或 uap 或 unprov
* 说明：询问网络状态。返回值分别代表：连接中（或掉线）、连上路由器但未连上服务器、连上小米云服务器、固件升级中、uap模式等待连接、关闭wifi（一小时未快连）。
* 举例：

↑net

↓offline

↑net

↓local

↑net

↓cloud

**time**

* 参数：无 或 posix
* 返回：当前日期和时间，格式见举例。时间为UTC+8时间，即中国的时间。
* 举例：

↑time

↓2015-06-04 16:58:07

↑time posix

↓1434446397

**model**

* 参数：无 或 <model\_name\_string>
* 返回：无参数时，返回当前model字符串。带参数时，将芯片model设为参数字符串，并返回ok。如果参数字符串不合法，则返回error。
* 说明：合法的model字符串有3部分构成：公司名、产品类别名、产品版本。3部分以“.”连接，总长度不能超过23个字符。
* 举例：

↑model

↓xiaomi.dev.v1

↑model xiaomi.prod.v2

↓ok

↑model

↓xiaomi.prod.v2

↑model company.product\_name\_that\_is\_too\_long.v2

↓error

↑model plug

↓error

**ota\_state**

* 参数：无 或 <ota\_state\_name\_string>
* 返回：无参数时，返回当前ota\_state字符串。带参数时，将芯片ota\_state设为参数字符串，并返回ok。如果参数字符串不合法，则返回error。
* 说明：合法的model字符串有idle，表示MCU目前允许升级；busy，表示MCU不允许升级
* 举例：

↑ota\_state

↓idle

↑ota\_state busy

↓ok

↑ota\_state

↓busy

↑ota\_state downloading

↓error

注：如果当前wifi模组正在升级，即ota\_state不是idle或busy，该命令会返回ok，但是当前设置其实无效。主要不能让MCU的设置打乱wifi模组的升级状态。

**mcu\_version**

* 参数：<mcu\_version\_string>
* 返回：如果参数合法，则返回ok，非法则返回error
* 说明：上报MCU固件版本。要求必须是4位数字。MCU应该在应用固件引导后，第一时间调用该命令上报版本。另外，如果下行命令收到MIIO\_mcu\_version\_req，也应该立即上报版本。
* 举例：

↑mcu\_version 0001

↓ok

↑mcu\_version A001

↓error

↑mcu\_version 1

↓error

**reboot**

* 参数：无
* 返回：ok
* 说明：MIIO接收到该命令后，将在0.5秒内重启。
* 举例：

↑reboot

↓ok

**restore**

* 参数：无
* 返回：ok
* 说明：MIIO接收到该命令后，将清除wifi配置信息，并在0.5秒内重启。
* 举例：

↑restore

↓ok

**factory**

* 参数：无
* 返回：ok
* 说明：MIIO芯片收到该命令后，在0.5秒内重启，并进入厂测模式。该模式下，芯片会按照预设置信息连接路由器。预设路由器SSID：miio\_default，密码：0x82562647
* 举例：

↑factory

↓ok

**get\_pin\_level**

* 参数：无 或 <mask\_1> <mask\_2> <mask\_3>
* 返回：<hex\_1> <hex\_2> <hex\_3>
* 说明：获取管脚电平，返回的3个32位16进制数的每一位对应MC200的一个GPIO的电平状态，0代表低电平，1代表高电平。hex\_1的低位到高位分别对应GPIO\_0到GPIO\_31，hex\_2对应32～63，hex\_3对应64～79。参数为GPIO mask。例如，只想获得GPIO-2的电平，则参数为0x4 0x0 0x0
* 举例：

↑get\_pin\_level 0x4 0x0 0x0

↓0x00000004 0x00000000 0x00000000

**echo**

* 参数：on 或 off
* 返回：ok
* 说明：打开串口回显功能。该功能开启时，MIIO芯片的串口输出会对输入进行回显。这在使用串口工具（比如SecureCRT或Xshell）时，提供方便。

**help**

* 参数：无
* 返回：所有支持的命令和参数格式

**dl**

* 参数：uart url
* 返回：ok/error
* 说明：uart 为透传数据接口，目前版本固定为uart。 url为需要下载的url。
* 当使用uart模式时，数据传输协议为xmodem，波特率230400,传输速率约10 KB/s

**set\_low\_power**

* 参数："on"/"off"
* 返回：ok/error
* 说明：打开模组的低功耗功能。目前只有乐鑫Esp8266上有实现。

**保留的下行命令**

MIIO芯片保留了一些命令名，用来通知主控MCU特定的事件。这类命令同样可用get\_down得到，但无需result或error回复结果。**MIIO\_mcu\_version\_req**

* 表示MIIO芯片向MCU询问MCU固件版本号。MCU收到该命令后，应调用mcu\_version命令，告知自己的版本号。
* 举例：

↑get\_down

↓down MIIO\_mcu\_version\_req

↑mcu\_version 1234

↓ok

**MIIO\_model\_req**

* 表示MIIO芯片向MCU询问设备种类名（即model）。MCU收到该命令后，应该调用model命令报告种类名。
* 举例：

↑get\_down

↓down MIIO\_model\_req

↑model xiaomi.demo.v1

↓ok

**MIIO\_net\_change**

* 表示MIIO芯片的网络连接状态发生了变化。其后的参数代表了芯片最新的网络状态。网络状态请参考net命令。
* 举例：

↑get\_down

↓down MIIO\_net\_change cloud （改变LED灯的状态等等...）

[建议不要使用该命令，MIIO\_net\_change是单次失效的命令，如果MCU获取到了net\_change，但又不处理，等到它真正想获取的时候却又获取不到]