1 文档介绍

1.1 文档范围

本手册详细介绍了UC8288 WIOTA终端模块提供的AT指令集。

1.2 命令语法

1.2.1 命令格式

本手册中所有命令行必须以”AT”或”at”作为开头，以回车（）作为结尾。响应通常紧随命令之后，且通常以”<回车><换行><响应内容><回车><换行>”（<响应内容>）的形式出现。在命令介绍时，“<回车><换行>”（）通常被省略了。

1.2.2 命令类型

通常命令可以有如下表所示的四种类型中的一种或多种形式。

类型	格式	说明
测试命令	AT+<cmd>=?	用于查询设置命令或内部程序设置的参数及其取值范围
查询命令	AT+<cmd>?	用于返回参数的当前值
设置命令	AT+<cmd>=<...>	用于设置用户自定义的参数值
执行命令	AT+<cmd>	用于读取只读参数或不需要额外参数的情况

1.2.3 参数类型

命令参数虽然多种多样，但是都可以简单地归结为整数类型和字符串类型（包括不带双引号的字符串和带双引号的字符串）这两种基本的类型，如下表所示。

类型	示例
整数类型	123
字符串类型	abc
	"hellow ,world"

表2 参数类型

1.2.4 注意事项

- AT串口输入时不支持回删键(backspace)功能
- 本文档+ERROR指+CME ERROR或者+EXT ERROR

2 基础 AT命令详细说明

2.1 AT

&AT测试命令。

Command	Possible response(s)
AT	OK ERROR

2.2 AT+RST 重启

系统重启。

Command	Possible response(s)
+RST	OK ERROR

watchdog重启，执行RST返回OK后，1s后watchdog重启。

2.3 ATE 回显

AT指令回显功能。

Command	Possible response(s)
ATE<value>	OK ERROR

- <value>：默认AT回显关闭
- 0： 关闭回显
- 1： 打开回显

2.4 AT&L 查询AT列表

查询支持的AT列表。

Command	Possible response(s)
AT&L	OK ERROR

2.5 AT+UART UART0配置

UART0配置。

Command	Possible response(s)
AT+UART= <baudrate>, <databits>, <stopbits>, <parity>, <flow_control>	OK ERROR

- <baudrate>：波特率，最大支持的波特率921600.
- <databits>：有效数据长度
- <stopbits>：停止位

- <parity>：奇偶检验
- <flow_control>: 流控。不支持流控.

2.6 AT+YMODEM 进入Ymodem刷机模式

进入Ymodem串口刷机模式。

Command	Possible response(s)
AT+YMODEM	OK ERROR

2.7 系统上报

Command	Mean
+CHOOSEMODEM:D	等待2S输入'D'进入Ymodem下载模式
+SYSTEM:START	启动RT-THREAD系统

3 WITOA AT命令详细说明

3.1 AT+WIOTAVERSION 查询版本信息

查询当前wiota库的版本号、git 信息、编译生成库的时间。

Command	Possible response(s)
AT+WIOTAVERSION	+WIOTAVERSION:<VERSION> +GITINFO:<GITINFO> +TIME:<maketime> +CCEVERSION:<cceversion> OK

- WIOTAVERSION：
  当前WIOTA库版本号
- GITINFO：
  当前库的git信息
- TIME：
  当前库的生成时间
- CCEVERSION:

  CCE 版本号

3.2 AT+WIOTAINIT 初始化

初始化wiota终端的资源。

Command	Possible response(s)
+WIOTAINIT	OK ERROR

3.3 AT+WIOTALPM 低功耗

低功耗设置

Command	Possible response(s)
+WIOTALPM=<mode>,<state>	OK ERROR

- <mode>:
- 0： sleep模式。外部串口唤醒后重新启动。
- 1： Gatting模式。Wiota协议栈在没有空闲的时候进去Gatting。
- <state>:
- 0： 关闭Gatting。
- 1： 打开Gatting。

3.4 AT+WIOTARATE 传输速率配置

设置最大速率模式和级别，三种模式:
第一种基本模式，是基本速率设置，有9档mcs速率级别（包括自动mcs），详见UC_MCS_LEVEL，默认为自动mcs，设置非自动mcs时同时关闭自动速率匹配功能

在第一种模式的基础上，在系统配置中dlul_ratio为1:2时，才能打开第二种模式，打开该模式能够提高该帧结构情况下两倍速率，默认第二种模式开启状态

在第一种模式的基础上，打开第三种模式，能够提升（8*(1 << group_number)）倍单终端的速率，但是会影响网络中其他终端的上行，建议在大数据量快速传输需求时使用，默认第三种模式关闭
备注：group_number为系统配置中的参数

Command	Possible response(s)
+WIOTARATE=<rate_mode> <rate_value>	OK ERROR

- <rate_mode>：UC_DATA_RATE_MODE
- <rate_value>：当rate_mode为UC_RATE_NORMAL时，rate_value为UC_MCS_LEVEL
  当rate_mode为UC_RATE_MID时，rate_value为0或1，表示关闭或打开
  当rate_mode为UC_RATE_HIGH时，rate_value为0，表示关闭，rate_value为其他值，表示当实际发送数据量（byte）大于等于该值时才会真正开启该模式，常用建议设置rate_value为100

```
typedef enum {
  UC_RATE_NORMAL = 0,
  UC_RATE_MID,
  UC_RATE_HIGH,
}UC_DATA_RATE_MODE;

typedef enum {
  UC_MCS_LEVEL_0 = 0,
  UC_MCS_LEVEL_1,
  UC_MCS_LEVEL_2,
  UC_MCS_LEVEL_3,
```

```
    UC_MCS_LEVEL_4,
    UC_MCS_LEVEL_5,
    UC_MCS_LEVEL_6,
    UC_MCS_LEVEL_7,
    UC_MCS_AUTO = 8,
  }UC_MCS_LEVEL;
```

BT=0.3时在不同symbol length和不同MCS时，对应每帧传输的应用数据量（byte），表中0表示不支持该MCS

symbol length	mcs0	mcs1	mcs2	mcs3	mcs4	mcs5	mcs6	mcs7
128	5	7	50	64	78	0	0	0
256	5	13	20	50	106	155	190	0
512	5	13	29	40	71	134	253	295
1024	5	13	29	61	106	218	449	617

初始化协议栈时为自动速率匹配功能打开状态，调用该接口入参为0~7时，设置最大速率级别，同时关闭自动速率匹配功能，再次调用该接口入参为8（或者不是0~7）时，会打开自动速率匹配功能。重启协议栈也会恢复初始功能。
为了保证接入成功率，接入短消息暂只使用mcs0~3，由于其中需要携带user id，正常会再减去4个字节空间，实际给应用的数据量会比正常短消息少。
接入短消息的MCS还有其他限制（应用层可不关注），symbol length为128/256/512/1024时，接入短消息的MCS最高为1/2/3/3。

每帧时间长度（frameLen）的粗略计算公式：（单位微妙）

```
// dlGroupNum和ulGroupNum取值0,1,2,3，ulGroupNum即系统参数配置中的group_number
groupNum = (1 << dlGroupNum) + (1 << ulGroupNum);
symbolNum = 11 + 2 * (1 << pn_num) + 64 * groupNum;  // pn_num目前固定为1
frameLen = symbolNum * 4 * 128 * (1 << symbol_length);  // symbol_length取值为0,1,2,3
```

举例： 系统配置中group_number为0，dlul_ratio为0，symbol_length为1，则

```
groupNum = 1 + 1 = 2;
symbolNum = 15 + 128 = 143;
frameLen = 143 * 4 * 128 * 2 = 146432 us
```

在此帧结构配置情况下，如果选择MCS2，则应用数据速率为 8*20/0.146432 = 1093 bps
(计算上行数据速率时，一般不考虑第一个包即随机接入包)

- 注意
  一味提高速率，可能导致上行始终无法成功

3.5 AT+WIOTAPOW 发射功率配置

低功耗设置

Command	Possible response(s)
+WIOTAPOW=<mode>,<power>	OK ERROR

- <mode>:
- 0：设置当前发射功率。
- 1：设置最大发射功率。
- <power>：发射功率。范围-16 ~ 21db。

3.6 AT+WIOTAFREQ 锁频

设置频点，iote和ap需要设置相同频点才能同步。在初始化系统之后，在系统启动之前调用，否则无法生效。

Command	Possible response(s)
+WIOTAFREQ=<freqpint>	OK ERROR
+WIOTAFREQ?	OK ERROR

- <freqpint>:
  频点idx，范围0~200，代表频点（470M+0.2*idx)。

3.7 AT+WIOTADCXO 设置频偏

设置终端频偏。在初始化系统之后，在系统启动之前调用，否则无法生效。

Command	Possible response(s)
+WIOTADCXO=<dcxo>	OK ERROR

- <dcxo>:
- 硬件的频偏参数，输入参数是16进制。

3.8 AT+WIOTAUSERID 设置用户ID

设置终端userid。获取用户id，此id为终端唯一标识。在初始化系统之后，在系统启动之前调用，否则无法生效。
目前只支持4字节长度的user id.

Command	Possible response(s)
+WIOTAUSERID=<id0>	OK ERROR
+WIOTAUSERID?	+WIOTAUSERID:<id0> OK

- <id0>:
  获取用户id，此id为终端唯一标识。长度为4个字节。每个id是0-0XFFFFFFFF.(16进制格式输入)

3.9 AT+WIOTACONFIG 系统配置

设置系统配置。

Command	Possible response(s)
+WIOTACONFIG=<id_len>, <symbol>,<dlul>,<bt>, <group_num>,<ap_max_pow>, <spec_idx>,<systemid>, <subsystemid>	OK ERROR
+WIOTACONFIG?	+WIOTASYSTEMCONFIG:<id_len>,<symbol>,<dlul>, <bt>,<group_num>, <ap_max_pow>,spec_idx>,systemid>,<subsystemid> OK

- <id_len>: user id长度，取值0,1,2,3代表2,4,6,8字节
- <symbol>: 帧配置，取值0,1,2,3代表128,256,512,1024
- <dlul>: 帧配置，下上行比例，取值0,1代表1:1和1:2
- <bt>: 调制信号的滤波器带宽对应，BT越大，信号带宽越大，取值0,1代表1.2和0.3，BT=1.2的数据率比BT=0.3
- <group_num>: 帧配置，取值0,1,2,3代表1,2,4,8个上行group数量
- <ap_max_pow>: ap最大功率，0~34dbm，需要与AP侧配置一致
- <spec_idx>: 使用的频段序号
- <systemid>: 系统id
- <subsystemid>: 子系统id

3.10 AT+WIOTARUN 启动wiota协议栈

启动wiota系统，进入NULL状态。
启动wiota后，收到数据会主动上报，数据最长为1024字节。
格式是：+WIOTARECV:,。

Command	Possible response(s)
+WIOTARUN=<state>	OK ERROR

- <state>:
- 0： 退出协议栈，回收wiota资源
- 1： 启动协议栈，进入NULL 状态

3.11 AT+WIOTASCANFREQ 扫频

在wiota启动后扫描频点信息，可扫一组频点和全扫，返回扫频结果，执行该命令后需要在窗口工具的发送区输入长度为dataLen（dataLen只能大于或等于输入的字符串长度，不能小于否则会获取字符串失败），个数为freqNum的字符串，并点击发送。

Command	Possible response(s)
+WIOTASCANFREQ =<timeout>, <dataLen>,<freqNum> ;	+WIOTASCAFREQ:(freq,rssi,snr,is_synced) OK > ERROR
+WIOTASCANFREQ	+WIOTASCAFREQ:(freq,rssi,snr,is_synced) OK ERROR

- <timeout>：扫描超时时间，单位ms。默认超时时间是2分钟。
- <dataLen>: 发送字符串的总长度+\r\n，比如要扫描的频点为1,2,3,4,5这五个频点
  1）执行at命令AT+WIOTASCANFREQ=10000,11,5;
  2）当出现>时十秒钟内在串口工具的发送区内输入字符串1,2,3,4,5
  3）点击发送
  4）等待扫频结果返回，结果会通过串口打印出来
- <freqNum>: 频点个数
- freq：频点信息
- rssi：信号强度
- snr：信噪比
- is_synced：该频点是否能同步

3.12 AT+WIOTARADIO 无线状态

只有在wiota同步成功后才能查询wiota无线状态信息，否则数据没有任何参考意义。

Command	Possible response(s)
+WIOTARADIO?	+WIOTARADIO=<temp>,<rssi>,<<ber>,<snr>, <cur_pow>,<max_pow>,<cur_mcs> OK ERROR

无线状态数据：

- temp: 当前芯片温度
- rssi: 信号强度
- ber: 误码率
- snr: 信噪比，范围 -25dB ~ 30dB
- cur_pow: 当前发射功率，范围 -16~21dBm
- max_pow: 最大发射功率，范围 -16~21dBm
- cur_mcs: 当前数据发送速率级别，范围 0~7

3.13 AT+WIOTACONNECT wiota连接ap

断开与AP的同步连接，回到NULL状态。

Command	Possible response(s)
+WIOTACONNECT=<state>, <activetime>	OK ERROR

- <state>:
  - 0： 断开连接，wiota进入NULL状态
  - 1： wiota 连接ap，进入同步状态
- <connecttimeout>:
  - 连接保持时间,单位是秒（s）。默认是3s，最小参数值为1.参数为0，表示不修改参数。

3.14 AT+WIOTASEND wiota发送数据

断开与AP的同步连接，回到NULL状态。

Command	Possible response(s)
+WIOTASEND=<timeout>,<len>	OK > ERROR
+WIOTASEND	> data OK ERROR

- <len>：数据的长度
- <timeout>:发送超时时间，单位ms。取值范围0-65535. 0代表试用默认值（60s）。

数据透传流程：

- <len>：数据的长度
- > :运行发送数据标志。一包数据最长为310字节。数据超过最长包310将被丢掉。如果应用层需要传超过310字节的数据，建议自己先分包。
- 0x1A:透传数据结束符。 发送失败返回"ERROR"， 发送数据成功返回“OK”。

3.15 +WIOTARECV wiota数据上报

数据上报。格式是：+WIOTARECV:<type>,<len>,<data>

- <type>：上报数据类型
  - 0： 短消息
  - 1：广播消息
  - 2：OTA消息
  - 3：扫频结果
  - 4：同步异常
- <len>：上报的数据长度
- <data>：数据长度不为0时，上报的数据

3.16 +WIOTALOG wiota log设置

wiota log设置。格式是：+WIOTARECV:<type>

- <type>：上报数据类型
  - 0：关uart log
  - 1：开uart log
  - 2：uart log使用uart0，如果从uart1切换到uart0，会把uart0的波特率改为460800，此时AT的波特率也是用该值
  - 3：uart log使用uart1，如果从uart0切换到uart1，会把uart0的波特率恢复为115200

    - 4：关spi log
    - 5：开spi log
- 注意：默认状态下，uart log使用uart1，波特率460800，AT使用uart0，波特率115200，在uart log的串口切换后，需要特别注意串口工具使用的波特率是否对应，如果AT的波特率不对时，发送at cmd会直接导致at挂住！

3.17 AT+WIOTASTATS wiota统计信息

断开与AP的同步连接，回到NULL状态。

Command	Possible response(s)
+WIOTASTATS=<mode>,<type>	+WIOTASTATS=type,data OK > ERROR
+WIOTASTATS?	+WIOTASTATS=0,rach_fail,active_fail,ul_succ,dl_fail,dl_succ,bc_fail,bc_succ OK

- <mode>：UC_STATS_MODE，0: 读数据， 1: 重置数据
- <type>: UC_STATS_TYPE，需要获取的数据类型

```
typedef enum {
    UC_STATS_READ = 0,
    UC_STATS_WRITE,
}UC_STATS_MODE;

typedef enum {
    UC_STATS_TYPE_ALL = 0,
    UC_STATS_RACH_FAIL,
    UC_STATS_ACTIVE_FAIL,
    UC_STATS_UL_SUCC,
    UC_STATS_DL_FAIL,
    UC_STATS_DL_SUCC,
    UC_STATS_BC_FAIL,
    UC_STATS_BC_SUCC,
    UC_STATS_UL_SM_SUCC,
    UC_STATS_UL_SM_TOTAL,
    UC_STATS_TYPE_MAX,
}UC_STATS_TYPE;
```

- 举例：
- AT+WIOTASTATS=0,0 和 AT+WIOTASTATS? 的返回数据一样
- AT+WIOTASTATS=0,4，返回+WIOTASTATS=4,(下行失败次数)
- AT+WIOTASTATS=1,4，重置下行失败次数

3.18 AT+WIOTACRC wiota校验设置

断开与AP的同步连接，回到NULL状态。

Command	Possible response(s)
+WIOTACRC=<crc_limit>	OK > ERROR
+WIOTACRC?	+WIOTACRC=1 OK

- <crc_limit>：0: 关闭CRC校验功能， 大于1: 表示数据长度大于等于crc_limit时，才打开CRC校验功能，所以crc_limit设置为1，则可表示任意长度的数据均加CRC

4 WIOTA 测试 AT

[10:22:37.038]发→◇at+wiotainit
□
[10:22:37.051]收←◆
OK

[10:22:38.473]发→◇at+wiotafreq=135
□
[10:22:38.481]收←◆
OK

[10:22:39.894]发→◇at+wiotafreq?
□
[10:22:39.899]收←◆+WIOTAFREQ=135

OK

[10:22:40.509]发→◇at+wiotauserid=f2345678,2
□
[10:22:40.522]收←◆
OK

[10:22:41.164]发→◇at+wiotauserid?
□
[10:22:41.179]收←◆+WIOTAUSERID=0xf2345678,0x0

OK

[10:22:42.120]发→◇at+wiotadcxo=e000
□
[10:22:42.127]收←◆
OK

[10:22:43.713]发→◇at+wiotarun=1
□
[10:22:43.834]收←◆
OK

[10:22:45.067]发→◇at+wiotaconnect=1,1
□
[10:22:45.082]收←◆
OK

[10:22:47.221]发→◇at+wiotasend
□
[10:22:47.232]收←◆>

> [10:22:49.610]发→◇65661A
> □
> [10:22:50.961]收←◆
> OK

[10:23:00.537]收
←◆+WIOTARECV,0,80,!!!!!!!!!!"""""""""""""##########$$$$$$$$$%%%%%%%%%&&&&&&&&&&'''
''''''((((((((((
[10:23:01.114]收
←◆+WIOTARECV,0,80,!!!!!!!!!!"""""""""""""##########$$$$$$$$$%%%%%%%%%&&&&&&&&&&'''
''''''((((((((((
[10:23:09.957]发→◇at+wiotaconnect=0,0
□
[10:23:09.969]收←◆
OK

[10:23:11.653]发→◇at+wiotarun=0
□
[10:23:11.666]收←◆
OK