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"

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，最大理论支持的波特率为6000000（6M），RS232建议设置为115200，RS485建议设置为9600。
- <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。

序号	波特率	寄存器配置值	理论偏差(%)
1	6000000	0	0
2	3000000	1	0
3	2000000	2	0
4	1500000	3	0
5	1000000	5	0
6	921600	5	8.506944444
7	750000	7	0
8	600000	9	0
9	512000	10	6.533984375
10	500000	11	0
11	460800	12	0.16015625
12	256000	22	1.901953125
13	230400	25	0.16015625
14	128000	45	1.9015625
15	115200	51	0.159722222
16	57600	103	0.159722222
17	56000	106	0.132142857
18	38400	155	0.158854167
19	19200	311	0.15625
20	14400	415	0.159722222
21	9600	624	0
22	4800	1249	0
23	2400	2499	0
24	1200	4999	0
25	600	9999	0
26	300	19999	0
27	110	54544	0

上图为波特率支持情况，黄色的两个波特率为理论支持，但由于串口工具不支持，暂未测试，红色的两个波特率，偏差超过5%，会出现乱码情况，不建议使用，其他波特率能保证基本通信。

2.6 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

3.0 AT CMD LIST

- 3.1 WIoTa系统资源
  版本信息查询 AT+WIOTAVERSION
  初始化协议栈 AT+WIOTAINIT
  系统配置 AT+WIOTACONFIG
  启动/关闭协议栈 AT+WIOTARUN
  连接/断开AP AT+WIOTACONNECT
  用户身份标识 AT+WIOTAUSERID
  模组ID查询 AT+WIOTAMODULEID
  设置用户组播ID AT+WIOTAMULTCAST
  WIoTa设备地址 AT+WIOTADEVADDRESS
  获取世界时间 AT+WIOTAGPSINFO

- 3.2 射频相关
  发送功率设置 AT+WIOTAPOW
  通信频率设置 AT+WIOTAFREQ
  频偏DCXO设置 AT+WIOTADCXO
  扫频 AT+WIOTASCANFREQ
  无线信息 AT+WIOTARADIO
  有源晶体设置 AT+WIOTAOSC
  外部32K晶振设置 AT+WIOTAOUTERK
  获取校准参数 AT+WIOTADJRST
  是否使用温度计算dcxo值 AT+WIOTAIUTEMP
- 3.3 低功耗相关
  低功耗模式设置 AT+WIOTALPM
  pagingRX配置 AT+WIOTAPAGINGRX
  pagingRX第二个唤醒ID配置 AT+WIOTAPAGINGRXANO
  获取唤醒原因 AT+WIOTAWAKEN
  pagingTX配置 AT+WIOTAPAGINGTX
  设置扩展ID模式 AT+WIOTAPAGINGMODE
- 3.4 数据收发
  传输速率配置 AT+WIOTARATE
  数据发送配置 AT+WIOTASEND
  数据透传模式 AT+WIOTATRANS
  DTU数据模式 AT+WIOTADTUSEND
  数据接收上报 +WIOTARECV
  校验CRC设置 AT+WIOTACRC
  获取上行重发次数 AT+WIOTARESEND
- 3.5 WIOTA数据暂存和唤醒电平
  数据暂存模式设置 AT+MODE
  暂存电平脉宽设置 AT+PULSEWIDTH
  暂存数据输出 AT+STOREDATA
  唤醒输出电平 AT+WAKEOUTPIN
- 3.6 调试相关
  协议栈LOG AT+WIOTALOG
  统计信息 AT+WIOTASTATS
  指示灯配置 AT+WIOTALIGHT
  内存使用量统计 AT+WIOTACKMEM
  内存值设置 AT+WIOTAMEMADDR
- 3.7 其他
  静态数据保存 AT+WIOTASAVESTATIC
  自动管理连接标识 AT+AUTOCONNECT

3.1 WIoTa系统资源

3.1.1 查询版本信息 AT+WIOTAVERSION

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)
+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。
  - 举例：
    发送：
    AT+WIOTAVERSION?
    回显：
    +WIOTAVERSION:v3.4_iote
    +GITINFO:Wed Jan 22 10:28:31 2025
    +TIME:Jan 24 2025 15:36:49
    +CCEVERSION:d73fff
    OK

3.1.2 初始化协议栈 AT+WIOTAINIT

初始化WIoTa终端的资源。

Command	Possible response(s)
+WIOTAINIT	OK ERROR

- 举例：
  发送：
  AT+WIOTAINIT
  回显：
  OK

3.1.3 系统配置 AT+WIOTACONFIG

3.1.3.1 V2.5以前的版本支持的系统配置 AT+WIOTACONFIG

设置系统配置。

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

- <ap_max_pow>: ap最大功率，暂时0~30dbm，需要与AP侧配置一致，实际需要设置的功率加20则为输入值，更详细的解释参见3.5节AT+WIOTAPOW功率参数。
- <id_len>: user_id长度，取值0,1,2,3代表2,4,6,8字节，默认四字节，IOTE该变量需要与AP保持一致，现在只支持设置为1，即四字节。
- <symbol>: 帧配置，取值0,1,2,3代表128,256,512,1024。
- <dlul>: 帧配置，该值代表一帧里面上下行的比例，取值0,1代表1:1和1:2。
- <bt>: 该值和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BT配置为1.2和BT配置为0.3，bt_value为1时，代表使用的是低阶mcs组，即低码率传输组。bt_value为0时，代表使用的是高mcs组，即高码率传输组（symbol_length为2或者3时，bt_value不能配置为0）。
- <group_num>: 帧配置，取值0,1,2,3代表一帧里包含1,2,4,8个上行group数量。
- <spec_idx>: 使用的频段序号，默认为3，即470-510M。
- <old_v>: 默认为0，如果v2.4版本（包含）的终端与v2.3（包含）之前版本的AP通信，需要将该值设置为1。
- <bitscb>: bit加扰功能，默认为1，如果v2.4版本（包含）的终端与v2.3（包含）之前版本的AP通信，需要将该值设置为0。
- <systemid>: 系统id，每个id是0-0xFFFFFFFF，16进制格式输入，不需要0x。 预留值，必须设置，但是不起作用。
- <subsystemid>: 子系统id， 每个id是0-0xFFFFFFFF ，16进制格式输入，不需要0x。(子系统的识别码，终端IOTE如果要连接该子系统（AP），需要将config配置里的子系统ID参数配置成该ID)。如果需要在启动之后修改子系统ID，则需要先disconnect，再配置subsystemid，再重新connect。
    - 举例：
      发送：
      AT+WIOTACONFIG=20,1,1,0,1,0,3,0,1,11223344,21456981
      回显：
      OK
      发送：
      AT+WIOTACONFIG?
      回显：
      +WIOTASYSTEMCONFIG=0,1,3,0,1,0,0,3,0,1,0x11223344,0x21456981
      OK

3.1.3.2 V2.5以及以后的版本支持的系统配置 AT+WIOTACONFIG

设置系统配置。 IOTE2.5以及以后的版本主要是删除了系统ID.

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

- <ap_tx_power>: ap发送功率，暂时0~30dbm，需要与AP侧配置一致，实际需要设置的功率加20则为输入值，更详细的解释参见3.5节AT+WIOTAPOW功率参数。 v2.8版本更新名字。
- <id_len>: user_id长度，取值0,1,2,3代表2,4,6,8字节，默认四字节，IOTE该变量需要与AP保持一致，现在只支持设置为1，即四字节。
- <symbol>: 帧配置，取值0,1,2,3代表128,256,512,1024。
- <dlul>: 帧配置，该值代表一帧里面上下行的比例，取值0,1代表1:1和1:2。
- <bt>: 该值和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BT配置为1.2和BT配置为0.3，bt_value为1时，代表使用的是低阶mcs组，即低码率传输组。bt_value为0时，代表使用的是高mcs组，即高码率传输组（symbol_length为2或者3时，bt_value不能配置为0）。
- <group_num>: 帧配置，取值0,1,2,3代表一帧里包含1,2,4,8个上行group数量。
- <spec_idx>: 使用的频段序列号，默认为3，即470-510M。
- <old_v>: 默认为0，如果v2.4版本（包含）的终端与v2.3（包含）之前版本的AP通信，需要将该值设置为1。
- <bitscb>: bit加扰功能，默认为1，如果v2.4版本（包含）的终端与v2.3（包含）之前版本的AP通信，需要将该值设置为0。
- <subsystemid>: 子系统id， 每个id是0-0xFFFFFFFF ，16进制格式输入，不需要0x。(子系统的识别码，终端IOTE如果要连接该子系统（AP），需要将config配置里的子系统ID参数配置成该ID)。如果需要在启动之后修改子系统id，则需要先disconnect，再配置subsystemid，再重新connect。 V2.9版本开始，subsystem_id 的高12bit，内部默认固定为0x214, 32bit的整体默认值仍为0x21456981，举例：如果配置subsystem_id为 0x12345678，则会被内部修改为0x21445678。
    - 举例：
      发送：
      AT+WIOTACONFIG=20,1,1,0,1,0,3,0,1,21456981
      回显：
      OK
      发送：
      AT+WIOTACONFIG?
      回显：
      +WIOTASYSTEMCONFIG=0,1,3,0,1,0,0,3,0,1,0x21456981
      OK

3.1.4 启动/关闭协议栈 AT+WIOTARUN

启动wiota系统，进入空闲状态。
关闭wiota后，回收系统资源。

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

- <state>:
  0： 关闭协议栈，回收WIoTa资源。
  1： 启动协议栈，进入空闲状态。
- 举例:
  发送:
  AT+WIOTARUN=1
  回显:
  OK

3.1.5 连接/断开AP AT+WIOTACONNECT

连接或断开与AP的同步。

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

- <state>:
  0：断开连接，WIoTa进入空闲状态。
  1：WIoTa连接ap，成功后进入同步状态。
  2：快速同步，在被sync paging信号唤醒后，iote可以快速同步到ap。
  3：快速同步，并且在同步后直接强制进入连接态。
- <activetime>:
  连接保持时间，单位是秒（s）。默认设置与AP匹配，与帧长有关，具体计算参见API接口文档中的“设置终端连接时间”，最小参数值为1，当参数为0时，表示不修改参数，使用默认配置。(断开连接时填0)。

  注意查询时的返回state是另外的定义:
  0：表示未同步
  1：表示同步
  其他含义参考API文档中uc_wiota_get_state接口
  - 举例:
    发送:
    AT+WIOTACONNECT=1,0
    回显:
    OK
    发送:
    AT+WIOTACONNECT?
    回显:
    +WIOTACONNECT=1,3 (第一个参数1表示同步，0代表未同步，其他含义参考API文档中uc_wiota_get_state接口，第二个参数是当前activetime)
    OK

- 注意：如果失步，建议先执行AT+WIOTACONNECT=0,0， 再执行：AT+WIOTACONNECT=1,0进行同步操作。

3.1.6 用户身份标识 AT+WIOTAUSERID

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。
如果需要在启动之后修改，则需要先disconnect，再配置userid，再重新connect。
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。id是0x1-0xFFFFFFFF (16进制格式输入，不需要0x)。
- 举例：
  发送：
  AT+WIOTAUSERID=ae81c452
  回显：
  OK
  发送：
  AT+WIOTAUSERID?
  回显：
  +WIOTAUSERID=0xae81c452
  OK

3.1.7 模组ID查询 AT+WIOTAMODULEID

查询模组ID，v3.1版本新增接口。

Command	Possible response(s)
+WIOTAMODULEID?	+WIOTAMODULEID=<MODULEID> +WIOTAMODULEID=<MODULEID><VALID(v3.4新增)> OK

- MODULEID：
  模组ID，字符串，18个字符。
- VALID：
  模组ID是否有效，1为有效，0为无效，v3.4新增返回值。
- 举例：
  发送：
  AT+WIOTAMODULEID?
  回显：
  +MODULEID=861340000000A89AE3
  OK

v3.4版本的回显：
+MODULEID=861340000000A89AE3,1
OK

3.1.8 设置用户组播ID AT+WIOTAMULTCAST

设置终端组播id。
在初始化系统之后调用，否则无法生效。也可以在启动之后设置。
目前只支持4字节长度的multicast id，支持3个不同的组播ID。
初始版本先支持1个组播号来调试。
注意： 查询接口暂未实现。

Command	Possible response(s)
+WIOTAMULTCAST=<multid0>,<multid1>,<multid2>	OK ERROR
+WIOTAMULTCAST?	+WIOTAMULTCAST:<multid0>,<multid1>,<multid2> OK

- <multid0,multid1,multid2>:
  组播ID是0x1-0xFFFFFFFF (16进制格式输入，不需要0x)，如果设置为0，则代表取消该编号的组播ID配置。
  - 举例：
    发送：
    AT+WIOTAMULTCAST=af81c458,0,0
    回显：
    OK
    发送：
    AT+WIOTAMULTCAST? (暂不支持)
    回显：
    +WIOTAMULTCAST=0xaf81c458,0,0
    OK

3.1.9 WIoTa设备地址 AT+WIOTADEVADDRESS

作为终端设备的唯一标识，在鉴权流程中会使用到。

Command	Possible response(s)
+WIOTADEVADDRESS=<dev_address>	OK > ERROR
+WIOTADEVADDRESS?	+WIOTADEVADDRESS=0x1234ABCD OK

- <dev_address>：长度为4字节，用16进制表示，但不需要加0x前缀。
- 举例：
  发送：
  AT+WIOTADEVADDRESS=ABCD5432
  回显：

OK
发送：
AT+WIOTADEVADDRESS?
回显：
+WIOTADEVADDRESS=0xABCD5432

3.1.10 获取世界时间 AT+WIOTAGPSINFO

获取世界时间。 v3.1新增接口。

Command	Possible response(s)
+WIOTAGPSINFO=<mode>,<gpio>	OK > ERROR
+WIOTAGPSINFO?	+GPSINFO=is_valid, gps_time_s, gps_time_us, rf_cnt_us, rf_cnt_curr OK

配置参数：

- <mode>：配置是否返回中断
- <gpio>：返回中断的gpio口

返回参数：

- <is_valid>：gps时间是否有效
- <gps_time_s>：gps时间的秒
- <gps_time_us>：gps时间的微妙 （模1秒的余数）
- <rf_cnt_us>：与gps对应的rf cnt时间戳，单位us
- <rf_cnt_curr>：当前rf cnt，一般比rf_cnt_us大
- 举例：
  发送：
  AT+WIOTAGPSINFO?
  回显：
  +GPSINFO=1,1718954845786,4562,3473921,4473921
- 注意：
  拿到的gps时间，是一个过去时间点，需要进行换算得到当前时刻的gps时间，
  rf_cnt_us与gps时刻对应，rf_cnt_curr与中断时刻对应，即可计算出中断时刻对应的gps时间。

3.2 射频相关

3.2.1 发射功率配置 AT+WIOTAPOW

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 ~ 21dbm（V2.7版本更新为 -18 ~ 22 dbm)。V0.09版本及之前版本由于代码限制，不支持负数解析，如at+wiotapow=0,-10，需要写成补码形式，即at+wiotapow=0,246。V0.09版本之后的版本，实际需要设置的功率加20则为输入值，例如想要设置功率-10dbm，则 at+wiotapow=0,10；想要设置功率20dbm，则 at+wiotapow=0,40。
- 如果设置当前功率值为正常范围值，则设置成该功率，并且关闭自动功率模式；如果参数为127（at+wiotapow=0,127），则代表恢复自动功率模式。

注意：设置最大功率，是为了限制自动功率的范围，设置当前功率也不能超过最大发射功率。

- 举例：
  发送：
  AT+WIOTAPOW=0,40
  回显：
  OK

3.2.2 通信频率设置 AT+WIOTAFREQ

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。
如果需要在启动之后修改频点，则需要先disconnect，再配置频点，再重新connect。

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)。
  - 举例：
    发送：
    AT+WIOTAFREQ=115
    回显：
    OK
    发送：
    AT+WIOTAFREQ?
    回显：
    +WIOTAFREQ=115
    OK

3.2.3 频偏DCXO设置 AT+WIOTADCXO

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
+WIOTADCXO?（v3.2新增）	+WIOTADCXO=<dcxo> OK

- <dcxo>：

- 硬件的频偏参数，输入参数是16进制。有源晶体不能设置。
    - 举例：
      发送：
      AT+WIOTADCXO=20000
      回显：
      OK
      发送：
      AT+WIOTADCXO?
      回显：
      +WIOTADCXO=0x20000
      OK

3.2.4 扫频 AT+WIOTASCANFREQ

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只能等于或大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>, <mode>,<dataLen>,<freqNum> ;	+WIOTASCAFREQ:(freq,rssi,snr,is_synced,subsysid) OK > ERROR

- <timeout>：扫描超时时间，单位ms。默认超时时间是2分钟。
- <mode>：扫频模式，模式0，使用已配置的子系统id，统一扫频；模式1，需要继续输入与频点数相同个数的子系统id，与之一一对应，每个子系统id是0-0xFFFFFFFF ，16进制格式输入，不需要0x；模式2，v3.5新增，快速扫频，输入格式与模式0一样，返回结果中仅rssi有效。
- <dataLen>: 发送字符串的总长度+2（即\r\n），比如要扫描的频点为1,2,3,4,5这五个频点（V3.3版本开始，内部不再使用dataLen，可任意填写，但必须数据要\r\n结尾）。
  1）执行at命令AT+WIOTASCANFREQ=10000,0,11,5。
  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，该参数为0时为全扫，全扫默认使用模式0

返回结果参数：

- freq：频点信息。
- rssi：信号强度，范围 -128 ~0。
- snr：信噪比，范围-25 ~30。
- is_synced：该频点是否能同步。
- subsysid：子系统ID，如果mode是0，则为当前系统配置的子系统ID，如果mode是1，则为频点对应的子系统ID。
    - 举例：
      发送：
      AT+WIOTASCANFREQ=60000,0,17,4
      >
      119,115,118,120

回显（系统配置默认subsysid为0x21456981）：
OK
+WIOTASCANFREQ:
115,-83,3,1,0x21456981
120,-79,0,0,0x21456981
119,-80,0,0,0x21456981
118,-84,0,0,0x21456981
OK

或者：
AT+WIOTASCANFREQ=60000,1,42,4
>
119,115,118,120,12bc63,876ad,fa876,3b290

回显：
OK
+WIOTASCANFREQ:
115,-83,3,1,0x876ad
120,-79,0,0,0x3b290
119,-80,0,0,0x12bc63
118,-84,0,0,0xfa876
OK

3.2.5 无线状态信息 AT+WIOTARADIO

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>,<min_pow>,<max_pow>,<cur_mcs>, <max_mcs>,<frac_offset> OK ERROR

无线状态数据：

- temp: 当前芯片温度。
- rssi: 信号强度，正常均为负值，例如 - 30，单位dbm。
- ber: 误码率，暂不支持。
- snr: 信噪比，范围 -25dB ~ 30dB。
- cur_pow: 当前发射功率，范围 -16~21dBm。 (V2.7版本更新为 -18 ~ 22 dbm)
- min_pow: 最小发射功率，范围 -16~21dBm。 (V2.7版本更新为 -18 ~ 22 dbm)
- max_pow: 最大发射功率，范围 -16~21dBm。 (V2.7版本更新为 -18 ~ 22 dbm)
- cur_mcs: 当前数据发送速率级别，范围 0~7。
- max_mcs: 截止目前最大数据发送速率级别，范围 0~7。
- frac_offset: 基带同步频偏，仅供参考，可判断此时同步是否正常，-1500 ~ 1500都属于正常。
    - 举例：
      发送：
      AT+WIOTARADIO?
      回显：

+WIOTARADIO=31,-22,0,20,-16,-16,21,5,5,220
OK

3.2.6 有源晶体设置 AT+WIOTAOSC

硬件如果是有源晶体，需要设置为有源晶体。此项设置与DCXO设置互斥，如果设置了有源晶体，就不能再设置DCXO。

Command	Possible response(s)
+WIOTAOSC=<mode>	OK > ERROR
+WIOTAOSC?	+WIOTAOSC=1 OK

- <mode>：0: 设置非有源晶体， 1: 设置有源晶体。如果设置大于1，内部也处理成有源晶体。
- 举例：
  发送：
  AT+WIOTAOSC=1
  回显：
  OK

3.2.7 外部32K晶振设置 AT+WIOTAOUTERK

模组硬件如果有外部32K晶振，可根据需要设置为外部32K晶振模式。

Command	Possible response(s)
+WIOTAOUTERK=<mode>	OK > ERROR
+WIOTAOUTERK?	+WIOTAOUTERK=1 OK

- <mode>：0: 设置内部32K， 1: 设置外部32K。
- 举例：
  发送：
  AT+WIOTAOUTERK=1
  回显：
  OK
- 注意：
  暂不支持查询，可在系统启动后配置，与协议栈运行与否无关。

3.2.8 获取校准参数 AT+WIOTADJRST

获取校准参数。 v2.8新增接口。

Command	Possible response(s)
+WIOTADJRST=<mode>	+ADJUST=temp, dir, offset > ERROR
+WIOTADJRST?	+ADJUST=temp, dir, offset OK

- <mode>：0: dcxo晶体， 1: tcxo晶体。
- <temp>：温度校准偏移
- <dir>：校准偏移方向，1：正，2：负
- <offset>：校准偏移量
- 举例：
  发送：
  AT+WIOTADJRST=0
  回显：
  +ADJUST=3,1,4
- 注意：
  暂不支持查询（即AT+WIOTADJRST？）

3.2.9 是否使用温度计算dcxo值 AT+WIOTAIUTEMP

设置是否使用温度计算dcxo。dcxo模组，每次同步完成或者同步失败，都会重新计算初始dcxo值，一种是使用默认校准dcxo值，是量产时在室内温度下测出的dcxo值，一种是根据温度曲线和当前温度计算出当前的dcxo值，但是读取温度每次会耗时16ms，如不需要根据温度计算dcxo，可通过该接口关闭。（v3.0新增接口）

Command	Possible response(s)
+WIOTAIUTEMP=<is_use>	OK ERROR
+WIOTAIUTEMP?	+WIOTAIUTEMP=<is_use> OK

- <is_use>:
  是否使用温度计算dcxo的标志。
- 举例：
  发送：
  AT+WIOTAIUTEMP=1
  回显：
  OK
  发送：
  AT+WIOTAIUTEMP?
  回显：
  +IUTEMP=1
  OK

3.3 低功耗相关

3.3.1 低功耗模式 AT+WIOTALPM

不同的低功耗模式设置。

Command	Possible response(s)
+WIOTALPM=<mode>,<value>,<value2>	OK ERROR

- <mode>:
- 0： 系统进入sleep模式。如果需要串口等外部唤醒源，则value设为1，如果不需要外部唤醒，则value设为0；如果需要定时唤醒，则需要先设置clock闹钟，再进入sleep； value2为是否降低32K时钟频率，可进一步降低0.3uA的sleep电流，但会导致sleep期间定时不准。
- 1： paging rx模式，value为是否降低32K时钟频率，同上，value2为最大检测次数，如果达到最大次数仍未检测到信号，基带会强制唤醒系统，如value2为0，则不会强制唤醒；终端进入paging rx模式，直到被AP侧的paging tx信号唤醒，则上电重启整个系统，进入前需要使用at+wiotapagingrx 设置相关配置。（v2.9版本，增加value2的最大次数功能，之前版本填0并且无意义）不能打开外部唤醒寄存器，只能通过拉低spi cs引脚来强制唤醒！
- 2： Gating模式。value为协议栈gating开关标志，value2为基带gating开关标志，开启后，WIoTa协议栈/基带在空闲的时候自动进入Gating，可降低运行时功耗。
- 3： clock闹钟设置，value单位为秒，value2设为0即可（无实际意义），设置闹钟后，会定时给系统一个rtc中断，如果在sleep态则可唤醒系统；
- 4： 降频，降低系统主频，value为UC_FREQ_DIV_MODE_E，value2设为0即可（无实际意义）。
- 5： 降压，降低系统电压，value为0，默认电压（1.82v）；1，降压（1.56v），value2设为0即可（无实际意义）。
- 6：sync paging模式，value为times（间隔次数），value2为最大检测次数，每隔(times+1) * 帧长，基带醒来检测一次信号，未检测到则计数加一继续休眠，如果检测到信号或者达到最大次数，则唤醒系统。需要配置外部32K晶振！不能打开外部唤醒寄存器，只能通过拉低spi cs引脚来强制唤醒！
- 7：paging timing模式，value为period（计数周期，单位微秒us），value2为最大计数次数，每隔period基带醒来计数加一，达到最大次数时唤醒系统。不能打开外部唤醒寄存器，只能通过拉低spi cs引脚来强制唤醒！
- 8：设置外部唤醒，如果需要串口等外部唤醒源，则value设为1，如果不需要外部唤醒，则value设为0。
- 9： paging tx信号发送，发送一段时间的信号，用来唤醒进入了paging rx模式的ap，发送前需要使用at+wiotapagingtx 设置相关配置。v3.1新增接口。
- 10： 设置进入paging rx或者sync paging时候的dcdc电压，用于降低功耗。v3.5新增接口。

注意：在需要用到32K时钟定时时，不建议降低32K时钟频率。

```
  enum at_wiota_lpm
  {
    AT_WIOTA_SLEEP = 0,
    AT_WIOTA_PAGING_RX, // 1, enter paging mode(system is sleep)
    AT_WIOTA_GATING,    // 2
    AT_WIOTA_CLOCK,     // 3
    AT_WIOTA_FREQ_DIV,  // 4
    AT_WIOTA_VOL_MODE,  // 5
    AT_WIOTA_SYNC_PAGING, // 6, sync paging, need sync ok, then enter
    AT_WIOTA_PAGING_TIMINT, // 7, sync paging, need sync ok, then enter
    AT_WIOTA_EX_WK,     // 8
    AT_WIOTA_PAGING_TX,     // 9
    AT_WIOTA_PAGING_VOL,    // 10, v3.5新增，set dcdc vol when enter paging rx or
sync paging
```

```
  AT_WIOTA_LPM_MAX,
};

typedef enum {
  FREQ_DIV_MODE_1 = 0,  // default: 96M
  FREQ_DIV_MODE_2 = 1,  // 48M
  FREQ_DIV_MODE_4 = 2,  // 24M
  FREQ_DIV_MODE_6 = 3,  // 16M
  FREQ_DIV_MODE_8 = 4,  // 12M
  FREQ_DIV_MODE_10 = 5, // 9.6M
  FREQ_DIV_MODE_12 = 6, // 8M
  FREQ_DIV_MODE_14 = 7, // 48/7 M
  FREQ_DIV_MODE_16 = 8, // 6M
  FREQ_DIV_MODE_MAX,
} UC_FREQ_DIV_MODE_E;

typedef enum {
  VOL_MODE_CLOSE = 0,  // 1.82v
  VOL_MODE_OPEN = 1,   // 1.56v
  VOL_MODE_TEMP_MAX,
} UC_VOL_MODE_E;
```

- 举例：
  发送：
  AT+WIOTALPM=1,0,0
  回显：
  OK

3.3.2 pagingRX配置 AT+WIOTAPAGINGRX

3.3.2.1 V2.5之前版本支持的pagingRX配置

设置paging rx接收检测模式的相关配置。

Command	Possible response(s)
+WIOTAPAGINGRX=<freq>, <spec_idx>,<band>,<symbol>, <nlen>,<utimes>,<thres>, <awaken_id>,<detect_period>	OK > ERROR

- <freq>：频点。
- <spec_idx>：频段。
- <band>：带宽。暂时只支持200K，即band为1。
- <symbol>：symbol length。
- <nlen>：检测头配置，1,2,3,4，默认值4。
- <utimes>：检测头配置，1,2,3，默认值2。
- <thres>：threshold检测门限，3~15，默认值10。增大该值，漏检率增大，虚警率减小。（虚警率即对噪声的敏感程度，漏检率即对唤醒信号的敏感程度）
- <awaken_id>：唤醒ID，根据symbol length不同，最大值不同，当symbol length为[0,1,2,3]时，唤醒ID最大值限制分别为[41,82,168,339]（可等于，最小值为0）。
- <detect_period>：接收端检测周期（单位ms，最大值44000），每隔该时间，基带会自动单独起来检测一次信号，如果检测到信号，则唤醒整个系统，如果没有则继续sleep，该时间越长，整体

功耗越低，相应的发送端想要唤醒接收端时则需要发送更长的时间。

- 举例：
  发送：
  AT+WIOTAPAGINGRX=57,3,1,0,4,2,10,23,1000
  回显：
  OK

3.3.2.2 V2.5以及之后版本支持的pagingRX配置

设置paging rx接收检测模式的相关配置。

Command	Possible response(s)
+WIOTAPAGINGRX=<freq>, <spec_idx>,<band>,<symbol>, <awaken_id>,<detect_period>, <nlen>,<utimes>,<thres>, <extra_flag>,<extra_period>	OK > ERROR

- <freq>：频点。
- <spec_idx>：频段。
- <band>：带宽。暂时只支持200K，即band为1。
- <symbol>：symbol length。
- <awaken_id>：唤醒ID，根据symbol length不同，最大值不同，当symbol length为[0,1,2,3]时，唤醒ID最大值限制分别为[41,82,168,339]（可等于，最小值为0）。
- <detect_period>：接收端检测周期（单位ms，最大值44000），每隔该时间，基带会自动单独起来检测一次信号，如果检测到信号，则唤醒整个系统，如果没有则继续sleep，该时间越长，整体功耗越低，相应的发送端想要唤醒接收端时则需要发送更长的时间。
- <nlen>：检测头配置，1,2,3,4，默认值4。
- <utimes>：检测头配置，1,2,3，默认值2。
- <thres>：threshold检测门限，3~15，默认值10。增大该值，漏检率增大，虚警率减小。（虚警率即对噪声的敏感程度，漏检率即对唤醒信号的敏感程度）
- <extra_flag>：物理层检测到唤醒信号后，自动继续休眠的功能flag配置，设为1则开启该功能。
- <extra_period>：物理层检测到唤醒信号后，自动继续休眠的时长配置，单位ms，如果 extra_period 小于等于 （detect_period + 10）ms，则继续休眠 detect_period 时长，否则继续休眠 extra_period 时长。(PS: v2.7及之前版本，extra_period均不能小于等于 （detect_period + 10）ms，v2.8版本已修复)
- 举例：
  发送：
  AT+WIOTAPAGINGRX=57,3,1,0,23,1000,4,2,10,1,5000
  回显：
  OK

3.3.3 pagingRX第二个唤醒ID配置 AT+WIOTAPAGINGRXANO

设置paging rx的第二个唤醒ID的配置。v2.9版本新增

Command	Possible response(s)
+WIOTAPAGINGRXANO= <period_multiple>, <awaken_id_another>	OK > ERROR

- <period_multiple>：第二个唤醒id的检测周期只能是第一个唤醒id的检测周期的倍数，该参数即为倍数，当倍数为0时，表示不检测第二个唤醒id，当倍数为1时，周期与第一个唤醒id相同，以此类推，倍数的最大值限制为255，一般不建议设置太大。
- <awaken_id_another>：第二个唤醒id，范围与第一个一样，不建议两个awaken id相同，当period_multiple不为0时才有效。
- 举例：
  发送：
  AT+WIOTAPAGINGRXANO=3,32
  回显：
  OK

3.3.4 获取唤醒原因 AT+WIOTAWAKEN

获取唤醒原因。 v2.8新增接口。

Command	Possible response(s)
+WIOTAWAKEN?	+WIOTAWAKEN=awaken_cause, is_cs_awawen, pg_awaken_cause, detected_times, detect_idx OK

- <awaken_cause>：唤醒原因
- <is_cs_awawen>：是否SPI CS唤醒
- <pg_awaken_cause>：paging唤醒原因，只有唤醒原因awaken_cause为AWAKENED_CAUSE_PAGING，此域才有效。
- <detected_times>：当前paging检测次数，有效性同上。
- <detect_idx>：检测到awaken id的index，0或者1，表示第一个或者第二个唤醒id。（v2.9新增）

参数对应表详见API文档中接口：
uc_wiota_get_awakened_cause
uc_wiota_get_paging_awaken_cause

- 举例：
  发送：
  AT+WIOTAWAKEN?
  回显：
  +WIOTAWAKEN=2,0,4,40,0
- 注意：
  无。

3.3.5 pagingTX配置 AT+WIOTAPAGINGTX

设置paging tx信号的相关配置。 v3.1新增接口。

Command	Possible response(s)
+WIOTAPAGINGTX=<freq>, <spec_idx>,<band>,<symbol>, <awaken_id>,<send_time>	OK > ERROR
+WIOTAPAGINGTX?	+WIOTAPAGINGTX=150,3,1,1,30,3000 > OK

- <freq>：频点。

- <spec_idx>：频段。
- <band>：带宽。暂时只支持200K，即band为1。
- <symbol>：symbol length。
- <awaken_id>：唤醒ID，根据symbol length不同，最大值不同，为[41,82,168,339]（可等于，最小值为0）。
- <send_time>：每次信号发送时长（单位ms）。
- 举例：
  发送：
  AT+WIOTAPAGINGTX=57,3,1,0,23,1000
  回显：
  OK

3.3.6 设置扩展ID模式 AT+WIOTAPAGINGMODE

（v3.1新增接口）扩展ID模式开启后，唤醒ID范围增大，根据symbol length不同，最大值限制为[1023, 4095,16383, 65535]（可等于，最小值为0），另外扩展ID模式，第二个唤醒ID仅支持与第一个唤醒ID相同周期，进休眠不能32K时钟降频，paging tx的send time必须与paging rx的detect time相同！

Command	Possible response(s)
+WIOTAPAGINGMODE= <rx_mode>,<tx_mode>	OK ERROR
+WIOTAPAGINGMODE?	+WIOTAPAGINGMODE=<rx_mode>,<tx_mode> OK

- <rx_mode>:
  是否使用rx扩展ID模式的标志。
- <tx_mode>:
  是否使用tx扩展ID模式的标志。一般rx，tx都配置成相同的。
- 举例：
  发送：
  AT+WIOTAPAGINGMODE=1,1
  回显：
  OK
  发送：
  AT+WIOTAPAGINGMODE?
  回显：
  +WIOTAPAGINGMODE=1,1
  OK

3.4 数据收发

3.4.1 传输速率配置 AT+WIOTARATE

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>：枚举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，对应每个子帧帧传输的应用数据量（byte）
(备注：下表中为单播数据包的数据量，如果是普通广播包，下表每项减2，如果是OTA包，下表每项减1)

symbol length	mcs0	mcs1	mcs2	mcs3	mcs4	mcs5	mcs6	mcs7
128	6	8	51	65	79	不支持	不支持	不支持
256	6	14	21	51	107	156	191	不支持
512	6	14	30	41	72	135	254	296
1024	6	14	30	62	107	210	450	618

1024	6	14	30	62	107	219	450	618
symbol length	**mcs0**	**mcs1**	**mcs2**	**mcs3**	**mcs4**	**mcs5**	**mcs6**	**mcs7**

初始化协议栈时默认打开自动速率匹配功能，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UC_MCS_AUTO（或者不是0~7）时，会打开自动速率匹配功能。
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最高分别为1/2/3/3。

每帧时间长度（frameLen）的粗略计算表格（单位微妙，该表格并不绝对准确）：
计算公式暂不公开，如需要可使用接口uc_wiota_get_frame_len获取（v0.13版本及之后提供该接口）

dlul_ratio	group_number	symbol_length	frameLen(us)
0	0	0	73216
0	0	1	146432
0	0	2	292864
0	0	3	585728
0	1	0	138752
0	1	1	277504
0	1	2	555008
0	2	0	269824
0	2	1	539648
0	3	0	531968
1	0	0	105984
1	0	0	211968
1	0	0	423936
1	0	0	208576
1	0	0	408576
1	0	0	400896

举例： 系统配置中group_number为0，dlul_ratio为0，symbol_length为1，则frameLen为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*21/0.146432 = 1147 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。
- 举例：
  发送：
  AT+WIOTARATE=0,3

回显：
OK

3.4.2 数据发送配置 AT+WIOTASEND

数据发送。

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

指定数据长度发送流程：

- <len>：数据的长度，最大310字节。
- <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）。
- 发送失败返回"ERROR"， 发送数据成功返回“OK”。

无数据长度的数据发送流程：

- 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。
- 在每读到一个字符之后等待10s，等待后续数据输入。
- 默认发送超时时间为60s。
    - 举例：
      发送：
      AT+WIOTASEND=5000,12
      >
      1234567890（备注：最后串口隐藏有\r\n，为2个字节)
      回显：
      SEND SUCC
      OK

注意：

1. 数据发送需要2条指令，每条指令之后都有工具自动增加的\r\n；
2. 如果工具支持也可以合并为AT+WIOTASEND=5000,12\r\n1234567890

3.4.3 数据透传模式 AT+WIOTATRANS

进入数据透传模式，在发送任意长度（不超过310字节）数据后不会立即退出。

Command	Possible response(s)
+WIOTATRANS=<timeout>,<endflag>	Enter transmission mode > Leave transmission mode OK
+WIOTATRANS	Enter transmission mode > Leave transmission mode OK

指定结束标记的数据发送流程：

- <timeout>: 发送超时时间，单位ms。取值范围0-65535，0代表使用默认值（60s），超出会被0xFFFF 取模。
- <endflag>: 指定退出数据透传模式标记，标记可为任意可见字符，长度最少为1个字符，最长不超过8个字符，超出时只取前8个字符，后面的舍弃。
- Enter transmission mode >: 进入透传模式。
- Leave transmission mode: 退出透传模式。
- 发送失败返回"SEND FAIL"， 发送数据成功返回“SEND SUCC”。
- 不指定结束标记的数据发送流程，采用默认结束标记"EOF"。
- 默认发送超时时间为60s
    - 举例：
      发送：
      AT+WIOTATRANS=5000,AA
      Enter transmission mode >
      123456789012
      回显：
      SEND SUCC
      发送：
      123456789012AA
      回显：
      SEND SUCC
      Leave transmission mode
      OK

注: 与普通AT命令类似，"\r\n"作为发送标记，发送内容不包含"\r\n"，同时发送内容也不包含结束标记，只有实际消息的内容。

3.4.4 接收数据上报 +WIOTARECV

接收数据上报。

Command	Possible response(s)
无	+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：组播ID0消息。
  4：组播ID1消息。
  5：组播ID2消息。
  6：扫频结果。
  7：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 。
  - 举例：
    回显：
    +WIOTARECV,0,12,123456789012
    OK
    回显：
    +WIOTARECV,0,12,1234567890 （有时候是这样，可能是AP发送的数据是带了\r\n的，所以显示不出来，实际传输的还是12个字符）
    OK

3.4.5 WIoTa校验设置 AT+WIOTACRC

CRC校验设置。

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

- <crc_limit>：0: 关闭CRC校验功能， 大于等于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。
- 举例：
  发送：
  AT+WIOTACRC=100
  回显：
  OK

3.4.6 获取上行重发次数 AT+WIOTARESEND

设置上行数据包重发次数。 v2.8新增接口。协议分包后，每包发送时，默认重发尝试次数为4，达到该次数后，认为发送失败，停止发送并上报给上层。 可配置该发送次数。

Command	Possible response(s)
+WIOTARESEND=<times>	OK > ERROR
+WIOTARESEND?	+WIOTARESEND=times OK

- <times>：发送总次数，默认值为4。可配置最小值为0，表示失败一次后就不重发。最大不建议超过6次。
- 举例：
  发送：
  AT+WIOTARESEND=3
  回显：
  OK
- 注意：
  暂不支持查询（即AT+WIOTARESEND?）

3.5 WIOTA数据暂存和唤醒电平

应用于上位机进入休眠，不能接收通过AT上报的WIOTA消息时，IoTE提供数据暂存功能，并在收到WIOTA消息时 `GPIO2` 输出电平信号，默认无数据时低电平，发生暂存数据时输出高电平，电平脉宽可配。GPIO及电平脉宽可通过静态数据配置。

3.5.1 数据暂存模式设置 AT+MODE

设置WIOTA数据暂存模式，默认通过AT输出。

Command	Possible response(s)
+MODE=<mode>	OK > ERROR
+MODE?	+MODE=1 OK

- <mode>：0: 直接通过AT串口输出接收到的WIOTA消息， 1: 进入数据暂存模式；最多存储 `5` 条WIOTA消息。
- 举例：
  发送：
  AT+MODE=1
  回显：
  OK

3.5.2 暂存电平脉宽设置 AT+PULSEWIDTH

设置暂存数据时电平的脉宽，在设置WIOTA数据暂存模式后生效。

Command	Possible response(s)
+PULSEWIDTH=<pin>,<width>	OK > ERROR
+PULSEWIDTH?	+PULSEWIDTH=50 OK

- <pin>：0~18，0对应GPIO0，1对应GPIO1，以此类推，默认2，即GPIO2；
- <width>：脉宽值，即电平保持时间(5~255)，单位：毫秒ms，建议上位机采取下降沿判断该电平信号。
- 举例：
  发送：
  AT+PULSEWIDTH=2,50
  回显：
  OK

3.5.3 暂存数据输出 AT+STOREDATA

把暂存的数据按接收时间的顺序`全部`输出。

Command	Possible response(s)
+STOREDATA?	+WIOTARECV=<type>,<len>,<data> OK

返回数据格式与3.16 接收数据上报描述一致。

注：命令执行后暂存的数据将被全部清空，因此该命令多次执行后将没有数据输出。

3.5.4 唤醒输出电平 AT+WAKEOUTPIN

应用于终端进入休眠后唤醒时，通过GPIO输出电平信号，通知上位机，默认为`GPIO7`，当被唤醒时输出高电平，并保持设定的脉宽后输出低电平。GPIO及电平脉宽可通过静态数据配置。

设置唤醒输出的GPIO引脚以及脉宽保持时间。

Command	Possible response(s)
+WAKEOUTPIN=<pin>,<width>	OK > ERROR
+WAKEOUTPIN?	+PIN=7 +WIDTH=50 OK

- <pin>：0~18，0对应GPIO0，1对应GPIO1，以此类推，默认7，即GPIO7；
- <width>：脉宽值，即电平保持时间(1~255)，单位：毫秒ms，建议上位机采取下降沿判断该电平信号。
- 举例：
  发送：

AT+WAKEOUTPIN=7,50
回显：
OK

3.6 调试相关

3.6.1 WIoTa log设置 AT+WIOTALOG

WIoTa log设置。

Command	Possible response(s)
+WIOTALOG=<type>	OK

- <type>：LOG类型
  0：关rv uart log，即协议栈串口LOG。
  1：开rv uart log。
  2：rv 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：rv 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rv spi log，即协议栈SPI LOG，需要通过另外的TRACE工具抓取。
  5：开rv spi log。
- 注意：默认状态下，rv uart log使用uart1，波特率460800，AT使用uart0，波特率115200，在rv 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！
  - 举例：
    发送：
    AT+WIOTALOG=2
    回显：
    B （由于此时AT的波特率已经切换成460800，但是工具还是原来的115200波特率接收，所以字符都是乱码，此时OK显示成了B，此时需要将工具波特率改为460800，才能正常显示rv uart log和使用AT命令）
    发送：
    AT+WIOTALOG=3 （在上一步基础上，先把工具波特率改为460800，然后再发该AT）
    回显：
    鶘x黿纗<□\0€x\0 （由于此时AT的波特率已经切换成115200，但是工具还是原来的460800波特率接收，所以字符都是乱码，此时OK显示成了乱码，乱码有可能不同，此时需要将工具波特率改为115200，才能正常使用AT命令）

3.6.2 WIoTa统计信息 AT+WIOTASTATS

WIoTa统计信息。

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。

3.6.3 指示灯设置 AT+WIOTALIGHT

开关指示灯，在二次开发版本中，可关闭指示灯，即停止协议栈对相应GPIO（2/3/7/16/17）的操作，避免冲突。（v3.2版本中移除了该功能！）

Command	Possible response(s)
+WIOTALIGHT=<mode>	OK > ERROR

- <mode>：0: 关闭指示灯， 1: 打开指示灯。
- 举例:
  发送:
  AT+WIOTALIGHT=1
  回显:
  OK

3.6.4 内存使用量统计 AT+WIOTACKMEM

内存使用量统计

Command	Possible response(s)
+WIOTACKMEM?	+MEMORY=total,used,maxused OK

- 举例：
  发送：
  AT+WIOTACKMEM?
  回显：
  +MEMORY=42028,9832,10764

3.6.5 内存值设置 AT+WIOTAMEMADDR

V3.5版本新增，设置任意内存地址的值，万能接口，调试备用，请勿随意调用。

Command	Possible response(s)
+WIOTAMEMADDR=<addr>, <value>	OK > ERROR

- <addr>：需要写入的地址。16进制数，不需要0x开头。
- <value>：需要写入的值。16进制数，不需要0x开头。
- 举例：
  发送：
  AT+WIOTAMEMADDR=391000,12ab
  回显：
  OK

3.7 其他

3.7.1 保存静态数据 AT+WIOTASAVESTATIC

在协议栈非RUN状态下执行用户静态数据保存。

Command	Possible response(s)
+WIOTSAVESTATIC	OK

- 举例：
  发送：
  AT+WIOTASAVESTATIC
  回显：
  OK

3.7.2 自动管理连接标识 AT+AUTOCONNECT

配置静态数据中自动管理连接标识。修改成功后IOTE会自动重启。IOTE V2.7版本开始支持。

Command	Possible response(s)
+AUTOCONNECT=<value>	OK ERROR

<value>：

- 0：设置WIOTA自动管理连接标识为0，表示开机默认不启动扫频连接AP功能；
- 1：设置WIOTA自动管理连接标识为1，表示开机默认启动扫频连接AP功能。

4 WIOTA 测试 AT

[16:16:16.485]发→◇at+wiotainit
□
[16:16:16.520]收←◆OK

[16:16:26.203]发→◇at+wiotafreq=115
□
[16:16:26.216]收←◆OK

[16:16:35.942]发→◇at+wiotauserid=63c8b54c
□
[16:16:35.959]收←◆OK

[16:16:42.244]发→◇at+wiotafreq?
□
[16:16:42.247]收←◆+WIOTAFREQ=115
OK

[16:16:44.580]发→◇at+wiotauserid?
□
[16:16:44.586]收←◆+WIOTAUSERID=0x63c8b54c
OK

[16:17:22.244]发→◇at+wiotaconfig=44,1,1,0,1,0,3,0,1,21456981
□
[16:17:22.261]收←◆OK

[16:17:25.763]发→◇at+wiotarun=1
□
[16:17:25.797]收←◆OK

[16:17:27.164]发→◇at+wiotaconnect=1,0
□
[16:17:27.173]收←◆OK

[16:17:29.516]发→◇at+wiotaconnect?
□
[16:17:29.528]收←◆+WIOTACONNECT=1,3
OK

[16:19:50.836]发→◇at+wiotasend=5000,22
12345012345678901234
□
[16:19:50.847]收←◆>
[16:19:51.182]收←◆SEND SUCC
OK

[16:20:28.045]收←◆+WIOTARECV,0,12,1234567890

[16:20:38.327]发→◇at+wiotaconnect=0,0

□

[16:20:38.349]收←◆OK

[16:20:39.404]发→◇at+wiotarun=0

□

[16:20:40.044]收←◆OK