


UCON 设备间 BLE 可靠传输约定

V2.3.0

1. 关于蓝牙连接

连接分两种场景：新添加遥控器、配置已经添加过的遥控器。

- (1) 新添加遥控器，是在点击  之后，进行蓝牙扫描，能扫描到附近已经打开的蓝牙设备，得到他们的名称和 Mac 地址，**此时不应在界面上显示已经配置过的设备**。所以需要扫描到的蓝牙设备列表进行过滤，规律规则是：只有名称为“UCON”并且 Mac 地址不存在于已配置过的 UCON 设备 Mac 地址列表（也就是侧滑菜单的遥控器 Remote_Instance 列表）中的设备，才能被显示。**如果有多个设备，默认连接扫描到的第一个。**
- (2) 在侧滑菜单中点击之前配置过的设备进行连接，直接根据 Remote_Instance 列表中保存的被点击设备的 Mac 地址进行直接连接即可。

Android 手机请注意需要在蓝牙设备 scan 结束之后再发起 GATT 连接，部分 Android 手机如果在 scan 过程中试图连接蓝牙从机，会导致异常断开，请一定要注意。这可能是 Android 4.X 的蓝牙协议栈引起的，5.0 上没有这个问题。

现在在 APP 和遥控器之间打通了两个 GATT Characteristic(后面简称 CHAR) 分别是 CHAR1(UUID 高 4 字节是 :0x0000fff1) ,和 CHAR2(UUID 高 4 字节是 :0x0000fff2) 。请在连接过程中把这两个 CHAR 配置成允许 **notify**。

其中 CHAR1 用于 APP 发送应用层数据给遥控器，并且接收发送过程中的 ACK。

CHAR2 用于遥控器发送数据给 APP，同时 APP 在收到分片之后在 CHAR2 上向遥控器回复 ACK。

2. APP 向遥控器发送数据

传输层作用是分包以及在 BLE 上实现可靠传输。

APP 发送给遥控器的用户数据总长度暂定不能超过 512 字节，否则遥控器侧无法连续保存。

传输时将数据分为 20 字节一个小片 (Fragment)，每个分片采用统一数据格式：

2字节总长度	1字节index	16字节用户数据
--------	----------	----------

图 1-1

总长度表示本次要传输的用户数据的总长度，比如传送从云端下载的码表时，每个码表均在 100~200 字节之间，总长度应为不含总长度头和 index 头部的其余用户数据总长。

传输时每次用户层传输数据的第一个分片填写总长度字段，后续分片可以不填写（也可以填写，但接收方不作解析）。Index 序号每次在接受到接收方确认时进行更改，更改规则下面会提到。

传输层的确认数据格式：

1字节数据类型	1字节响应类型	16字节响应数据
---------	---------	----------

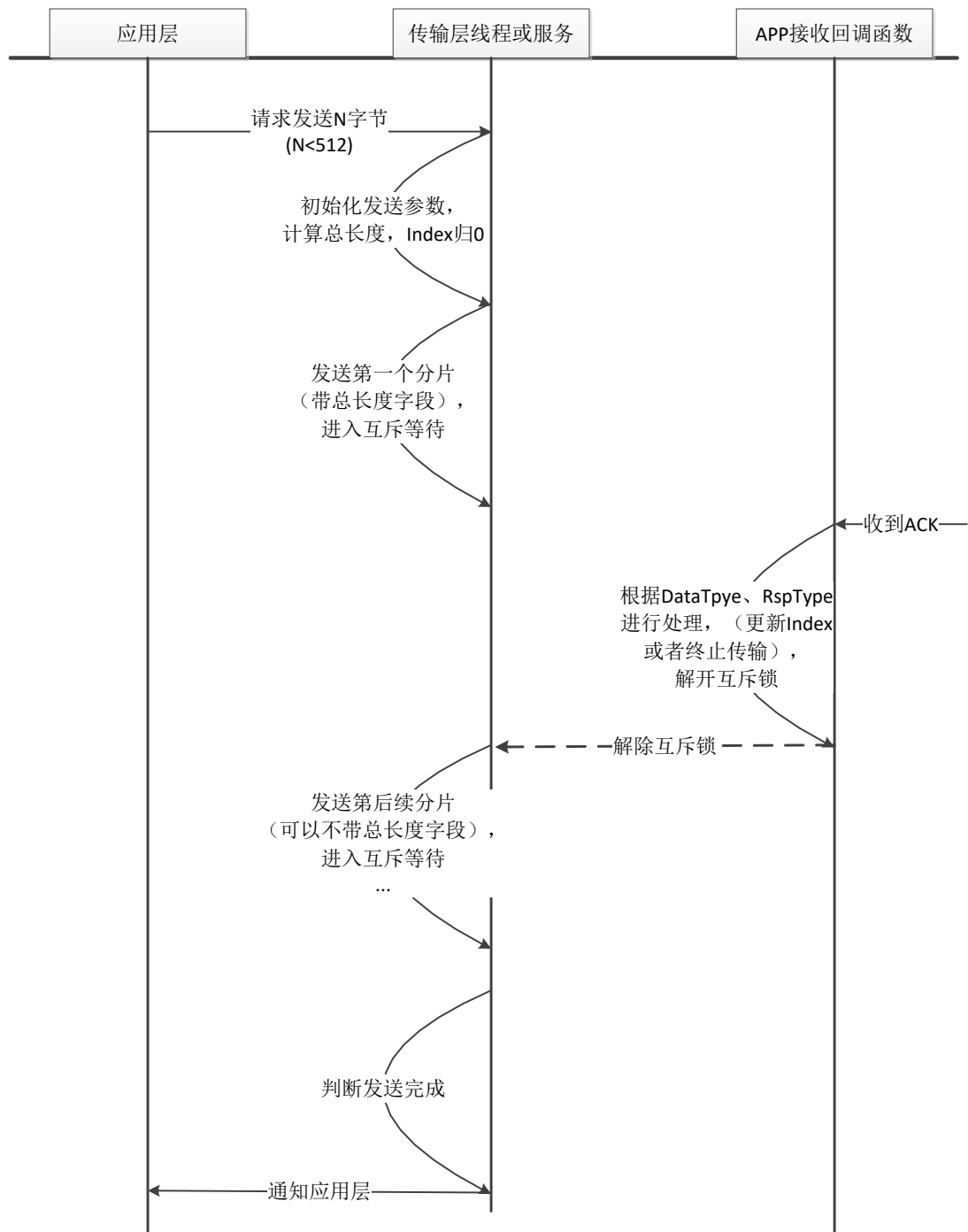
图 1-2

此数据在分片传输过程中由 APP 接收到，**遥控器会确保在分片传输过程中在 CHAR1 上没有用户层数据发送给 APP（就算有，也是从 CHAR2 发送）**，以保证此传输机制正常运行。

其中：

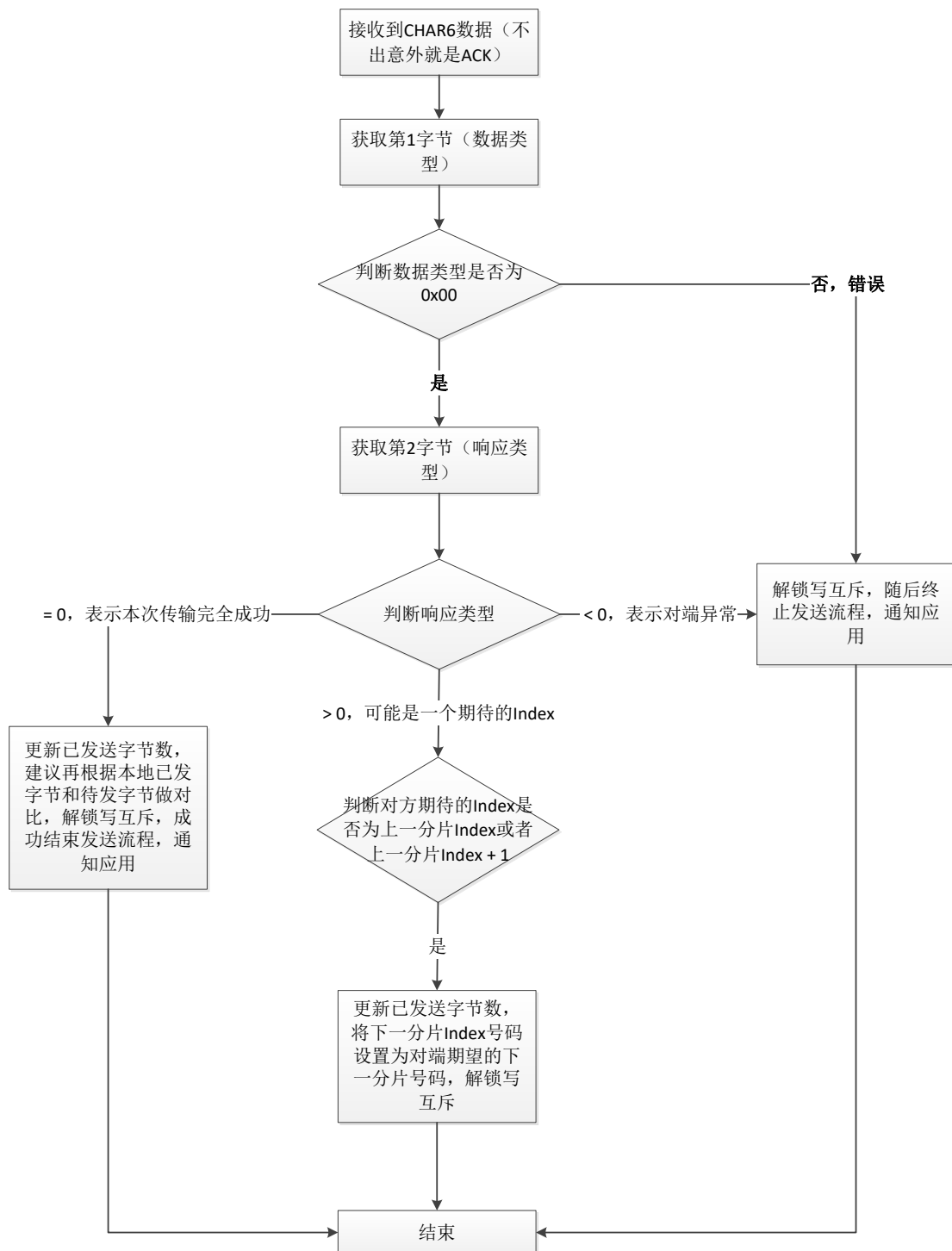
- 1 字节数据类型应为：0x00，否则 APP 视为无效确认信息。
- 1 字节响应类型可能为：
 - 0x00，表示本次用户层数据传输全部完成
 - 0x01，表示本次分片传输完成，在响应数据部分为对端期望的下一个 Index 号
 - 0x02，表示对端错误，APP 侧应终止用户层传输
- 如果当响应类型为 0x01 时，在响应数据的第一字节存放对端期望的下一个 Index 号(分片编号)，APP 端应更新下一次要传输的数据相对总用户数据的偏移，以及下一个分片的 Index 字段

APP 发送流程和互斥时序如下图



一些需要注意的地方：

- (1) ACK 非常短，是不分片的，所以为了区分 APP 收到的数据是分片 ACK，还是遥控器主动发送的用户层数据，我们用了两个 CHAR 来进行通信，因为 APP 通过 CHAR1 向遥控器发送数据，所以 CHAR1 上收到的数据完全当成 ACK 来处理。而 CHAR2 上收到的数据完全当成遥控器主动发送的用户数据来处理。
- (2) 处理 ACK 的时候不需要组包。
- (3) 发送线程解除互斥锁的时候有两种可能：
 - a) 互斥等待超时：如果是互斥超时，那么有可能是 APP 的这一个 20 字节分片数据在传输过程中丢掉了，也可能是对端的 ACK 帧在传回的过程中丢掉了。这种情况下，重复发送上一分片（使用上一分片的 index，和上一分片的数据），并启动一个重传计数，目前测试程序中是重试 3 次，如果 3 次都因为重传超时解除的互斥锁，那就判定为蓝牙链路问题，主动结束发送流程。如果下次重传成功，则将重传计数归零。
 - b) 收到对端的 ACK 解除互斥锁：此时要解析图 1-2 的 ACK 帧。解析流程如下：



(4) 几种失步的情况和避免原理：

- a) 某分片在发送至对端过程中因为 BLE 弱连接而丢失。此时遥控器方收不到任何分片，APP 的发送进入写互斥，等待一段时间后超时，并开始重传上一分片。
- b) 遥控器收到序号为 Index 的分片，但发给 APP 的带有期望序号是 Index+1 的 ACK 在传输过程中被丢掉，同理，APP 侧进入写互斥超时，此时按 APP 的逻辑应当重发上一分片，该重发的分片被遥控器接收之后，遥控器对比 Index 并不是期望的 Index+1，故丢弃该分片，并重发期望序号是 Index+1 的 ACK 给 APP。
- c) 遥控器在接收处理时发生异常，直接丢弃本次的分片，并以当前 Index 作为期待的下一分片

Index 通过 ACK 告知 APP，APP 收到之后重发一次分片。

- d) 蓝牙断开，在 APP 和遥控器侧都会产生一个断开事件，双方收到该事件后终止发送操作，并复位所有跟分片重组相关的变量。

3. 遥控器或者底座蓝牙模块向 App 发送数据

从单片机设备向 APP 传输数据，采用 GATT2 通道上传数据，上传场景请参考章节 4.应用场景协议。上行数据在 APP 端无需组包，直接按照如下应用层协议解析即可：

1字节数据类型	1字节响应类型	16字节用户数据
---------	---------	----------

其中：

- 1 字节数据类型应为：0x01(以区分作为 ACK 时的 0x00)，否则 APP 视为无效确认信息。
- 1 字节响应类型请具体参考第 4 章节，应用场景协议。
- 16 字节用户数据也请参考第 4 章节，应用场景协议。

再次声明：从 GATT2 上接收数据，需要首先将 GATT2 配置成可 notify 类型，且目前 GATT2 上通过回调函数等从单片机设备侧接收到的数据报文无需组包、直接解析即可。

4. 应用场景协议（红色标识的表示暂未实现）

（1）App 连接上遥控器之后

App → Remote: 0x0000(hello CMD) + remote instance LCD code ,LCD code 不足 160 字节使用 0x00 填充

0x00	0x00	160 Bytes Instance Name (LCD code)
------	------	------------------------------------

Remote → App: 0x0100 + 10 字节版本号，例如 V1.0.0，空白部分用 0x00 填充。

0x01	0x00	10 Bytes Version Number (eg. V1.0.0)
------	------	---

（2）传输红外码表

App → Remote: 0x0001(download CMD) + remote number + binary length + binaries

0x00	0x01	1 Byte Remote Number	1 Byte Category	2 Bytes Binary Length	2 Bytes Binary Checksum	N Bytes Binaries
------	------	----------------------	-----------------	-----------------------	-------------------------	------------------

Remote → App: 0x0101(download RSP)

0x01	0x01	1 Byte ACK
------	------	------------

应用层返回值：1 字节，0x00 代表本次传输成功（包括遥控器侧数据校验）；0x01 代表本次传输失败

（3）确认码表传输

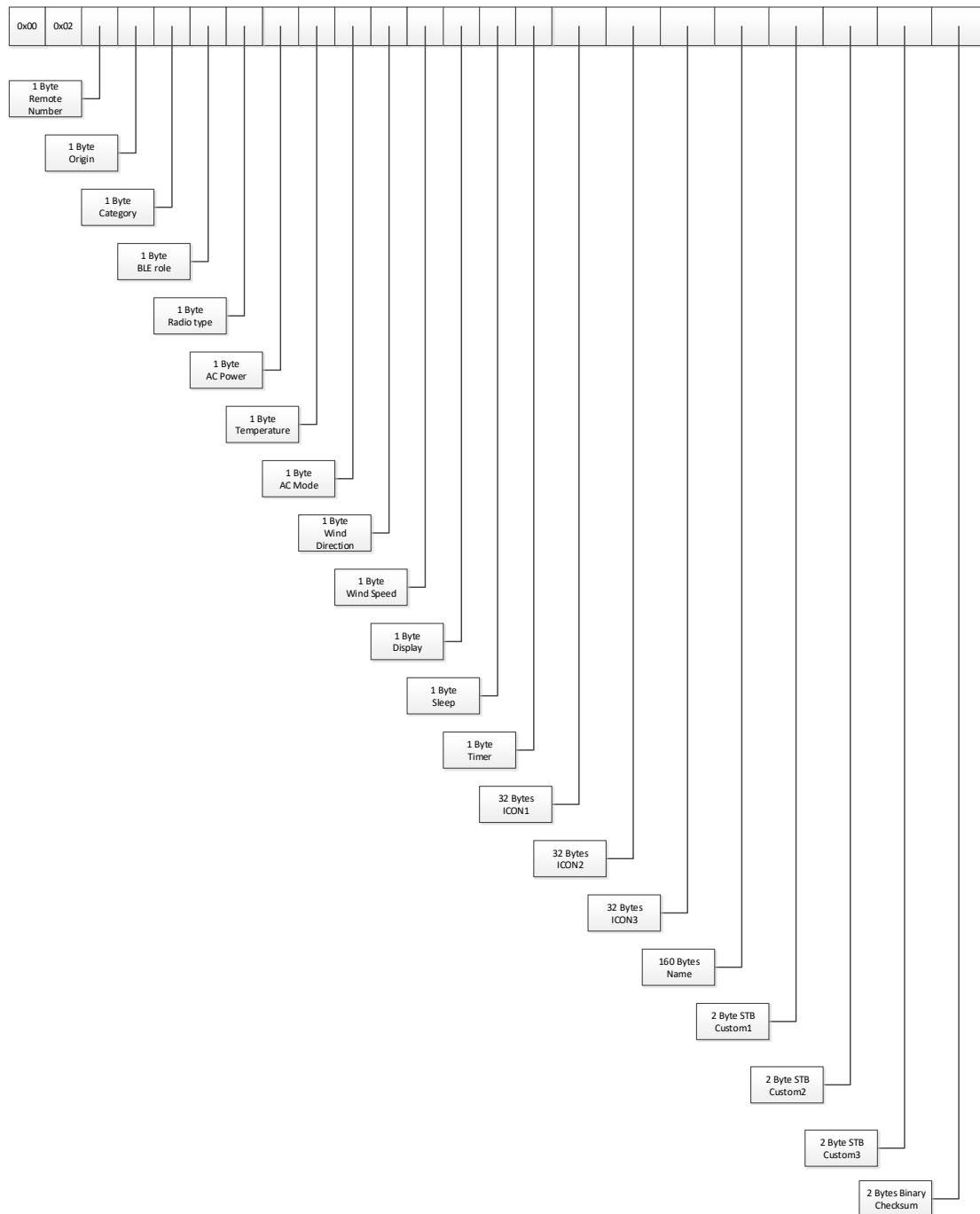
App → Remote: 0x0002(download confirm CMD) + remote number + origin + category + BLE role + radio type + AC power + temperature + AC mode + Wind direction + Wind speed + AC display + AC sleep + AC timer + ICON1 + ICON2 + ICON3 + Name + STB custom1 + STB custom2 + STB custom3，Name 字段的 LCD code 不足 160 字节使用 0x00 填充

（STB custom 为机顶盒自定义频道，在下载码表的机顶盒页面可输入，在协议字段中格式为 BYTE 数字，而并非 LCD 点阵码）

协议中各参数如下表：

Remote Number	遥控器上的模式序号，从 0 – 7 一共支持 8 种家电模式，APP 根据当前云端的遥控器序号决定接下来的一个序号是几。例如，已经配置的序号是 1、2、3，那么接下来的序号是 4；如果已经配置的序号是 1、2、4、5，那么接下来的序号是 3，以此类推。
Origin	遥控器当前模式的来由，分为码表下载和自定义学习两种，这里应该是码表下载，默认为 0x01。
Category	家电种类序号，在 APP 的选择家电种类一页中由用户选定，即数据库中 Category 的 ID 号：空调为 0x01，电视机为 0x02，机顶盒为 0x03，网络盒子为 0x04，IPTV 为 0x05，DVD 为 0x06，风扇为 0x07，投影仪为 0x08，音响为 0x09，灯为 0x0A，蓝牙为 0x18。
BLE Role	蓝牙模式，0x00 表示从机，0x01 表示主机。
Radio Type	红外模式是 0x00，蓝牙模式是 0x01，目前全是红外码表，所以固定为 0x00
AC Power	0x01 – 表示空调关闭状态，遥控器下次按下时为打开
Temperature	0x08 – 24 摄氏度
Mode	0x00 – 空调功能，默认为自动“冬暖夏凉”？
Wind Direction	0x00 – 没有摆风
Wind Speed	0x00 – 低风速
Display	0x00 – 正常数显
Sleep	0x00 – 非睡眠（静音）模式
Timer	0x00 – 没有定时关闭

上述绿色表格部分为空调相关的字段，**但是无论当前配置的是不是空调，均请按照默认值进行设置。**



Remote → App: 0x0102(download confirm RSP)

0x01	0x02	1 Byte ACK
------	------	------------

应用层返回值：1 字节，0x00 代表本次传输成功（包括遥控器侧数据校验）；0x01 代表本次传输失败

（4）命令遥控器进入学习模式

App → Remote: 0x0003(start DIY CMD) + remote number

0x00	0x03	1 Byte Remote Number
------	------	----------------------

Remote → App: 0x0103 (start DIY response)

0x01	0x03
------	------

(5) 命令遥控器保存学习模式

App → Remote: 0x0004(save DIY CMD) + remote number + remote name ,Name 字段的 LCD code
不足 160 字节使用 0x00 填充

0x00	0x04	1 Byte Remote Number	160 Bytes LCD code	32 Bytes ICON1	32 Bytes ICON2	32 Bytes ICON3
------	------	----------------------	--------------------	----------------	----------------	----------------

Remote → App: 0x0104 (save DIY response)

0x01	0x04
------	------

(6) 寻找遥控器

App → Remote: 0x0005 (find request)

0x00	0x05
------	------

Remote → App: 0x0105 (find response)

0x01	0x05
------	------

(7) 进入遥控器配置状态

App → Remote: 0x0006 (remote to configure + remote number (当前欲配置的家电遥控器位置，范围是 0~7，因为一共只支持 8 种不同家电))

0x00 0x06 remoteNumber (1 字节)

Remote → App: 0x0106

(8) 结束遥控器配置状态

App → Remote: 0x0007 (remote to release + remote number (当前正在配置的家电遥控器位置，范围是 0~7，因为一共只支持 8 种不同家电))

Remote → App: 0x0107

0x00 0x07 remoteNumber(1 字节)

(9) 恢复出厂设置

App → Remote: 0x0008

Remote → App: 0x0108

(10) 查询版本号

App → Remote : 0x0009

Remote → App: 0x0109 + 10 字节版本号，例如 V1.0.0，空白部分用 0x00 填充。

(11) 开始 OAD

App → Remote: 0x000A

Remote → App: 0x010A

(12) App 配置底座命令

App → Remote: 0x0040 + SecurityType + mobileID + channel + passkey + length of Router SSID + RouterSSID

0x00	0x40	1 Byte SecurityType	32 Bytes MobileID	2 Bytes CHANNEL	16 Bytes Router Passkey	1 Byte Router SSID Name length	N Bytes Router SSID
------	------	------------------------	-------------------	-----------------	----------------------------	-----------------------------------	------------------------

Remote → App: 0x0140 + PDSN , 此命令通过 GATT2 进行接收, 不组包, 0x01 代表从 Center 到 App , 0x40 代表上述下行命令 0x0040 的响应, 后面紧接着 16 字节为底座的 PDSN, 为字符串类型, 例如 “P0000000000000001” 。

0x01	0x40	16 Bytes UCON Center PDSN
------	------	---------------------------

各个字段含义解释：

SecurityType	家庭路由器的鉴权模式，由 APP 探测得到： 0 – 无密码 1 – WEP OPEN 2 – WEP SHARED 3 – WPA 4 – WPA2 5 – WPA, WPA2 MIXED 6 – EAP TLS
MobileID	手机的唯一标识码，只要能通过手机平台接口获取到手机唯一码，并做 32 字节 MD5 散列即可
Channel	通过热点检测获取（应该为 WIFI 频段），据我们实测数据，2.4G WIFI 热点此参数为 6，默认为 0
Router Passkey	家庭路由器的密码
Router SSID Name Length	家庭路由器名字长度，例如 “TPLINK” 长度为 6 字节
Router SSID	家庭路由器热点名，例如 “TPLINK”

新的方案（仅限 Android 1.6.2 以上版本，不包括 1.6.2，其余版本及 Apple 客户端 V1.6.0 以下版本，均沿用 0x0040 命令方案）：

App → Remote: 0x0041 + SecurityType + mobileID + channel + passkey + length of Router SSID + RouterSSID

Remote → App: 0x0141 + ERROR_CODE or PDSN

ERROR_CODE 定义如下：

Progress0 – AP found	0 (ASCII 字符 ‘0’ ,下同)
Progress0 – AP not found	1
Progress1 – WLAN Connected	2
Progress1 – WLAN Connect failed	3
Progress2 – Registered	4
Progress2 – Register failed	5

(13) 遥控器纯键码表下载

App → Remote: 0x000C + Remote Number + Origin + KeyCount + [KeyIndex + Length per key + KeyCode] + ...

0x00	0x0C	1 Byte Remote Number	1 Byte Origin	1 Byte Key Count	1 Byte Key1_index	2 Bytes Key1_length	N Bytes Key1_content
------	------	-------------------------	---------------	------------------	----------------------	------------------------	-------------------------	-------

Remote → App: 0x010C

协议中各个参数如下表所示：

Remote Number	遥控器上的模式序号，从 0 – 7 一共支持 8 种家电模式，APP 根据当前云端的遥控器序号决定接下来的一个序号是几。例如，已经配置的序号是 1、2、3，那么接下来的序号是 4；如果已经配置的序号是 1、2、4、5，那么接下来的序号是 3，以此类推。
Origin	遥控器当前模式的来由，分为码表下载和自定义学习两种，这里应该是键码码表下载，默认为 0x02。
Key Count	1 字节 Key Count 表示本次传输了多少个键码。
Key Index	遥控器面板按键固有编号
Key Length	键码长度
Key Content	键码内容
备注：Key Index，Key Length，Key Content 为三元组，出现次数和 Key Count 相同	

(14) 删除家电遥控器

App → Remote: 0x000B + Remote Number

Remote Number 的含义和前面的协议一致，即用升序排序插入的方式产生的遥控器空位编号 (0~7)，同时也是数据库 Remote 表的 remote_number 字段的值。

(15) 获取遥控器设置参数

App → Remote: 0x000C

Remote → App: 0x010C + status array

0x01	0x0C	2 Byte Vibration Sensibility	1 Byte Welcome logo Status	4 Bytes Screen-ON time
------	------	------------------------------	----------------------------	------------------------

(16) 设置，振动开关灵敏度

App → Remote: 0x000D + Vibration Sensitive

Vibration Sensitive 占 2 字节，用数字表示，由 APP 确定不连续区间 [100—500]

(17) 设置，遥控器是否显示欢迎 LOGO

App → Remote: 0x000E + Welcome LOGO Switch

Welcome LOGO Switch 占 1 字节，0 表示禁用（不显示），1 表示显示，默认是显示

(18) 调整家用电器显示顺序

App → Remote: 0x000F + Remote Order

(19) 设置，亮屏幕时间

App → Remote: 0x0010 + Screen on time (ms)

(20) 底座和 APP 向遥控器同步 UNIX 时间

APP → Remote: 0x0012（来自 APP）/ 0x0042（来自底座）+ RTC sync

0x00	0x12 0x42	4 Bytes RTC
------	--------------	-------------

(21) 底座和 APP 向遥控器读取用户日活数据

APP → Remote：在 GATT Char3 上进行一次读取操作

Remote → APP: 0x0113（目标 APP）/ 0x0143（目标底座）+ 16 Bytes UAD

0x01	0x13 0x43	4 Bytes UDA0	4 Bytes UDA1	4 Bytes UDA2	4 Bytes UDA3
------	--------------	--------------	--------------	--------------	--------------

其中第一个包表示 16 字节的遥控器 PDSN 序号；

(22) 获取遥控器版本号，不挂断

App → Remote：0x0014

Remote → App: 0x0114 + 10 字节版本号，例如 V1.0.0，空白部分用 0x00 填充。

(23) 下载蓝牙主机码表

App → Remote：0x0015，遥控器需要首先切换到 BLE Peripheral 模式

0x00	0x15	1 Byte Remote Number	1 Byte Category	1 Byte Mac or Name (TAG104)	6 Bytes Mac address	1 Byte Name length	20 Byte peripheral name	2 Bytes Binary Length	2 Bytes Binary Checksum	N Bytes Binaries
------	------	----------------------	-----------------	-----------------------------	---------------------	--------------------	-------------------------	-----------------------	-------------------------	------------------

Remote → App：0x0115

0x01	0x15	1 Byte ACK
------	------	------------