蓝牙终端-通信模组BLE通信协议

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 日期 | 修改记录 | 作者 |
| V1.0 | 2020.7.20 | Draft | Lane |
| V0.1 | 2020-08-11 | 修改 | Allen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 概述

本协议为APP与蓝牙模组之间BLE通讯的规范。 蓝牙终端的连接示意图如下图所示：



蓝牙终端通信连接示意图

# 协议定义

命令采取主从通信模式，蓝牙终端为主，BLE模组为从，通信方式采用请求-应答方式，所有命令都由蓝牙终端发起请求，蓝牙模组应答。

**UUID列表：**

Service UUID:

0x14839AC4-7D7E-415C-9A42-167340CF2339

0x14839AC4-7D7E-415C-9A42-167340CF2339

Command Characteristic UUID:

0x8B00ACE7-EB0B-49B0-BBE9-9AEE0A26E1A3

Notify Characteristic UUID:

0X0734594A-A8E7-4B1A-A6B1-CD5243059A57

## 帧格式如下

蓝牙终端和BLE模组之间的数据通讯的是以帧单位的。每帧的数据最长长度为256 bytes。

帧格式定义:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **LSB** |  | | | **MSB** |
| **Head** | Payload | | | Tail |
| 0x7E | **CMD/RSP** | **Data Length** | **Data** | 0xFF |
| 1 octet | 1 octet | 1 octet | Variable | 1 octet |

1. 所有数据域以小端格式表示，即低字节先发送，高字节后发送。
2. Header： 0x7E，表示一帧数据的开始，后面是被传数据。
3. Tail： 0xFF，表示一帧数据的结束。
4. CMD/RSP：命令或者响应。
5. Data Length：传输的Data长度。
6. Data：传输的数据。
7. 在发送方，如果Payload存在如下字节，必须要做转码处理，接收方接收到数据之后，必须做相反的转码处理。

|  |  |
| --- | --- |
| 字符 | 转码 |
| 0x7E | 0x8C 0x81 |
| 0xFF | 0x8C 0x00 |
| 0x8C | 0x8C 0x73 |

# 透传命令

## BlePassthrough Cmd

BLE模组接收到所有的命令（文件传输命令除外），都必须要透明转发到MCU，MCU返回应答，模组必须透明转发到蓝牙终端。

命令格式定义如下表所示：

Request定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Cmd | UINT8 | 0xXX | 命令码 |
| 1 | Data Length | UINT8 | N | 数据长度 |
| 2 | data | UINT8[n] | - | 数据 |

Response定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Cmd | UINT8 | 0xXX | 命令码 |
| 1 | Data Length | UINT8 | N | 数据长度 |
| 2 | data | UINT8[n] | - | 数据 |

# 文件传输命令

## FwUpdate Start (0x20)

文件传输开始请求

Request定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Head | UINT8 | 0x7E | 包头 |
| 1 | Cmd | UINT8 | 0x20 | 命令码 |
| 2 | Length | UINT8 | 0x0D | 数据包长度，不包含本身和Cmd字节 |
| 3 | Target | UINT8 |  | 升级目标：  1：智能中控固件。  2：按键板固件。  其他值：保留 |
| 4 | Reserved | UINT8 |  | 保留 |
| 5-8 | File Length | UINT32 |  | 文件长度 |
| 9 | MainVer | UINT8 |  | 固件主版本号 |
| 10 | SubVer | UINT8 |  | 固件子版本号 |
| 11 | MinorVer | UINT8 |  | 固件修订版本号 |
| 12-15 | BuildNum | UINT32 |  | 固件Build版本号 |
| 16 | Tail | UINT8 | 0xFF | 包尾 |

Response定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Head | UINT8 | 0x7E | 包头 |
| 1 | Cmd | UINT8 | 0x20 | 命令码 |
| 2 | Length | UINT8 | 0x03 | 数据包长度，不包含本身和Cmd字节 |
| 3 | Result | UINT8 |  | 参见附录1，错误码的定义 |
| 4 | Tail | UINT8 | 0xFF | 包尾 |

## FwUpdate (0x21)

文件数据传输请求。

Request定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Head | UINT8 | 0x7E | 包头 |
| 1 | Cmd | UINT8 | 0x21 | 命令码 |
| 2 | Length | UINT8 | 132 | 数据包长度，不包含本身和Cmd字节 |
| 3-6 | offset | UINT32 |  | 偏移 |
| 7-134 | File Data | UINT8[128] |  | 文件数据 |
| 135 | Tail | UINT8 | 0xFF | 包尾 |

Response定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Head | UINT8 | 0x7E | 包头 |
| 1 | Cmd | UINT8 | 0x21 | 命令码 |
| 2 | Data Length | UINT8 | 0x01 | 数据包长度，不包含本身和Cmd字节 |
| 3 | Result | UINT8 |  | 参见附录1，错误码的定义 |
| 4 | Tail | UINT8 | 0xFF | 包尾 |

## FwUpdate Done (0x22)

文件传输结束请求。

Request定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Head | UINT8 | 0x7E | 包头 |
| 1 | Cmd | UINT8 | 0x22 | 命令码 |
| 2 | Length | UINT8 | 0x04 | 数据包长度，不包含本身和Cmd字节 |
| 3-6 | CRC | UINT32 |  | 检验和 |
| 7 | Tail | UINT8 | 0xFF | 包尾 |

Response定义：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Name | Type | Value | Descriptor |
| 0 | Head | UINT8 | 0x7E | 包头 |
| 1 | Cmd | UINT8 | 0x22 | 命令码 |
| 2 | Data Length | UINT8 | 0x01 | 数据包长度，不包含本身和Cmd字节 |
| 3 | Result | UINT8 |  | 参见附录1，错误码的定义 |
| 4 | Tail | UINT8 | 0xFF | 包尾 |

## APPROM OTA文件格式定义

APPROM的文件格式定义如下：文件的前段数据是纯的固件数据，文件将会被逻辑划分成若干个块，每个块的大小是128字节对齐，文件尾部不完整块用00h填充，文件的最后一个块包含文件的CRC，文件有效长度信息，版本信息等。



说明：

1. BIN DATA CRC: 使用CRC校验算法对BIN DATA计算出一个CRC值, 不包含补位的数据。
2. BIN DATA LEN: BIN DATA的长度，不包含补位的数据。
3. Firwware version: 固件的版本号。
4. FILE INFO CRC: 使用CRC校验算法对文件最后的一个区块字节，从BIN DATA CRC位置开始，长度为128 - 4，计算出的CRC值，用于校验最后的区块是否是有效的内容。
5. CRC算法如下：

Uint16\_t crc16\_compute(const uint8\_t \* p\_data, uint32\_t size, const uint16\_t \* p\_crc)

{

uint32\_t i;

uin电池仓6\_t crc = (p\_crc == NULL) ? 0xffff : \*p\_crc;

for (i = 0; i < size; i++)

{

crc = (unsigned char)(crc >> 8) | (crc << 8);

crc ^= p\_data[i];

crc ^= (unsigned char)(crc & 0xff) >> 4;

crc ^= (crc << 8) << 4;

crc ^= ((crc & 0xff) << 4) << 1;

}

return crc;

}

# 附录1：Firmware Version定义

MCU Firmware version adopts **GNU** style(Including 4 parts):

Major\_Version\_Number**.**Minor\_Version\_Number**.**Revision\_Number**.**Build\_Number

* Main Version Number (**主版号，1字节**)

从1开始，当项目在进行重大修改或局部修正较多，而导致项目整体发生全局变

化时，主版本号加1.

* Minor Version Number (**子版本号，1字节**)

当项目在原有的基础上增加了部分功能时，主版本号不变，子版本号加1，修正

版本号复位成0.

* Revision Number **(修正版本号，1字节)**

当项目进行了局部修改或bug修正时，主版本号和子版本号都不变，修正版本号

加1.

* Build Number **(编译版本号，4字节)**

Build Number是不断递增的。

# 附录2：Hardware Version定义

Hardware adopts the below version style(Including 2 parts):

Device\_Typer. Major\_Version\_Number