耳机连接机制设计文档

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文档名称 |  | 版本号 | 1.00.00 | | |
| 文档编号 |  | | |
| 文档类别 | 使用说明 | 文档阶段 | 初稿 | | |
| 项目名称 | TWS | 作者 | 何继胜 | | |
| 承担部门 | 软件研发部 | 批准 |  | | |
| 文档日期 | 2020-04-26 | 使用范围 | 公司内部 | 页数 | 9 |

目录

[1 文档版本 3](#_Toc38730461)

[2 总体流程 3](#_Toc38730462)

[3 功能键 3](#_Toc38730463)

[4 打开充电盒 4](#_Toc38730464)

[5 取出充电盒 5](#_Toc38730465)

[6 佩戴耳机 5](#_Toc38730466)

[7 摘下耳机 5](#_Toc38730467)

[8 放入充电盒 6](#_Toc38730468)

[9 关闭充电盒 6](#_Toc38730469)

[10 低功耗 7](#_Toc38730470)

[11 升级 7](#_Toc38730471)

[12 状态定义 7](#_Toc38730472)

[13 核心流程 8](#_Toc38730473)

# 文档版本

|  |  |  |
| --- | --- | --- |
| 日期 | 版本号 | 简要描述 |
| 2020-04-25 | 1.00.00 | 初稿 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 总体流程

每个操作，都会触发同步左右耳机状态的事件。



# 功能键

用途：

* 配对：按大约3秒左右，白色呼吸灯闪烁。
* 恢复出厂设置：按大约10秒左右，换色呼吸灯常亮。

**注意：**

* 需要充电盒打开的时候，按键事件才有效。
* 一副耳机定义：两只耳机相会配对使用
* 非一副耳机定义：左右耳机不是出厂配对的状态；从其他充电盒中取出的耳机，放入当前充电盒中。
* 在恢复出厂设置之后，打开充电盒，自动进入配对状态。

## 短按/3秒

* 单只耳机：该耳机进入配对模式
* 一副耳机：右耳机进入配对模式
* 两只耳机，非一副：进入恢复出厂设置模式，重新配置成一对

## 长按/10秒

* 单只耳机：恢复出厂设置，进入单耳模式
* 两只耳机（一副耳机/非一副）：恢复出厂设置，进入双耳模式

## 配对



## 恢复出厂设置

# 打开充电盒

|  |  |
| --- | --- |
| 触发事件 | 事件用途 |
| OPEN\_CASE | 通知耳机，可以主动连接手机。前提：需要与另一只耳机通讯。 |

**相关Task：**

充电盒🡪PLC🡪PhyState🡪SM

充电盒通知PLC时，可能有延时。

**外围传感器工作：**

|  |  |
| --- | --- |
| 传感器 | 是否工作 |
| 接近光 | OFF |
| Apollo | OFF |
| 敲击 | OFF |

**注意：**如果耳机处于低功耗状态，充电盒在打开的时候，会唤醒耳机。

**事件处理流程：**



状态描述:

* 主动连接手机：与手机连接经典蓝牙，开启BLE广播（功能码，GAIA）。
* 进入配对状态：开启配对，在手机经典蓝牙界面，可以找到蓝牙设备，开启BLE广播（功能码：PAIR）。
* 进入空闲状态：手机发现不了耳机，不会主动连接手机，不开启BLE广播。

相关状态图：A🡪B，F🡪E，H🡪G，D🡪C

配对状态会持续60s。60s以后，如果还没有与手机配对，进入空闲模式。如果希望再次进入配对模式，可以【关闭充电盒，再打开充电】或是【按功能键3s】，重新进入配对模式。

配对

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 打开充电盒 | 配对状态 | 配对成功 | 配对连接 |
| 左耳机 | 0X2008 | 0X2008 | 0X2008 | 0X2008 |
| 右耳机 | 0X2008 | 0X0800 | 0X2008 | 0X2008 |

主动连接

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 打开充电盒 | 配对连接 | 播放音乐/通话 |  |
| 左耳机 | 0X2008 | 0X2008 | 0X2010 |  |
| 右耳机 | 0X2008 | 0X2008 | 0X2010 |  |

# 取出充电盒

|  |  |
| --- | --- |
| 触发事件 | 事件用途 |
| OUT\_CASE | 通知耳机，位置发生变化 |

**相关Task：**PLC🡪PhyState🡪SM

**外围传感器工作：**

|  |  |
| --- | --- |
| 传感器 | 是否工作 |
| 接近光 | ON |
| Apollo | OFF |
| 敲击 | OFF |

**事件处理流程：**



相关状态图：B🡪C，B🡪E，C🡪G，E🡪G

# 佩戴耳机

|  |  |
| --- | --- |
| 触发事件 | 事件用途 |
| IN\_EAR | 通知耳机，位置发生变化 |

**相关Task：**

Proximity🡪PhyState🡪SM

**外围传感器工作：**

|  |  |
| --- | --- |
| 传感器 | 是否工作 |
| 接近光 | ON |
| Apollo | ON |
| 敲击 | ON |

**事件处理流程：**

是否与手机断开连接

是否在0min-5min之间 -> 耳机自己在处理重连

是否在5min-50min之间 -> 佩戴上，触发耳机重连手机

# 摘下耳机

|  |  |
| --- | --- |
| 触发事件 | 事件用途 |
| OUT\_EAR | 通知耳机，位置发生变化 |

**相关Task：**

Proximity🡪PhyState🡪SM

**外围传感器工作：**

|  |  |
| --- | --- |
| 传感器 | 是否工作 |
| 接近光 | ON |
| Apollo | OFF |
| 敲击 | OFF |

**事件处理流程：**

无特殊操作。

# 放入充电盒

|  |  |
| --- | --- |
| 触发事件 | 事件用途 |
| IN\_CASE | 通知耳机，位置发生变化 |

相关Task：

PLC🡪PhyState🡪SM

外围传感器工作：

|  |  |
| --- | --- |
| 传感器 | 是否工作 |
| 接近光 | OFF |
| Apollo | OFF |
| 敲击 | OFF |

相关状态图：C🡪B，E🡪B，G🡪C，G🡪E

# 关闭充电盒

|  |  |
| --- | --- |
| 触发事件 | 事件用途 |
| CLOSE\_CASE | 断开与手机的连接 |

相关Task：

PLC🡪PhyState🡪SM

外围传感器工作：

|  |  |
| --- | --- |
| 传感器 | 是否工作 |
| 接近光 | OFF |
| Apollo | OFF |
| 敲击 | OFF |

相关状态图：B🡪A，C🡪D，G🡪H，E🡪F

# 低功耗

# 升级

打开充电盒，耳机主动连接手机，耳机发送进入dfu指令。进入升级状态，断开所有连接，开启BLE广播。如果在指定的时间内没有连接上，取消dfu。

如果在dfu模式下，打开充电盒，不主动连接手机。

# 状态定义

|  |
| --- |
| typedef enum sm\_application\_sub\_states {  APP\_SUBSTATE\_TERMINATING = 0x0001, /\*!< Preparing to shutdown (e.g. disconnecting links) \*/  APP\_SUBSTATE\_SOPORIFIC = 0x0002, /\*!< Allowing sleep \*/  APP\_SUBSTATE\_SOPORIFIC\_TERMINATING = 0x0004, /\*!< Preparing to sleep (e.g. disconnecting links) \*/  APP\_SUBSTATE\_IDLE = 0x0008, /\*!< Audio is inactive \*/  APP\_SUBSTATE\_BUSY = 0x0010, /\*!< Audio is active \*/  APP\_SUBSTATE\_DFU = 0x0020, /\*!< Upgrading firmware \*/   APP\_SUBSTATE\_INITIALISING = 0x0040, /\*!< App module and library initialisation in progress. \*/  APP\_SUBSTATE\_DFU\_CHECK = 0x0080, /\*!< Interim state, to see if DFU is in progress. \*/  APP\_SUBSTATE\_FACTORY\_RESET = 0x0100, /\*!< Resetting the earbud to factory defaults. \*/  APP\_SUBSTATE\_STARTUP = 0x0200, /\*!< Startup, syncing with peer. \*/   APP\_SUBSTATE\_PEER\_PAIRING = 0x0400, /\*!< Pairing with peer earbud \*/  APP\_SUBSTATE\_HANDSET\_PAIRING = 0x0800, /\*!< Pairing with a handset \*/  APP\_SUBSTATE\_DISCONNECTING = 0x1000, /\*!< Handing over handset connection \*/   APP\_END\_OF\_SUBSTATES = APP\_SUBSTATE\_DISCONNECTING, /\*!< The last substate \*/  APP\_SUBSTATE\_MASK = ((APP\_END\_OF\_SUBSTATES << 1) - 1), /\*!< Bitmask to retrieve substate from full state \*/ } appSubState;  typedef enum sm\_application\_states {  /\*!< Initial state before state machine is running. \*/  APP\_STATE\_NULL = 0x0000,  APP\_STATE\_INITIALISING = APP\_SUBSTATE\_INITIALISING,  APP\_STATE\_DFU\_CHECK = APP\_SUBSTATE\_DFU\_CHECK,  APP\_STATE\_FACTORY\_RESET = APP\_SUBSTATE\_FACTORY\_RESET,  APP\_STATE\_STARTUP = APP\_SUBSTATE\_STARTUP,  APP\_STATE\_PEER\_PAIRING = APP\_SUBSTATE\_PEER\_PAIRING,  APP\_STATE\_HANDSET\_PAIRING = APP\_SUBSTATE\_HANDSET\_PAIRING,  APP\_STATE\_TERMINATING = APP\_SUBSTATE\_TERMINATING,   /\*! Earbud is in the case, parent state \*/  APP\_STATE\_IN\_CASE = APP\_END\_OF\_SUBSTATES<<1, /// 0X200N  APP\_STATE\_IN\_CASE\_TERMINATING = APP\_STATE\_IN\_CASE + APP\_SUBSTATE\_TERMINATING,  APP\_STATE\_IN\_CASE\_IDLE = APP\_STATE\_IN\_CASE + APP\_SUBSTATE\_IDLE,  APP\_STATE\_IN\_CASE\_DFU = APP\_STATE\_IN\_CASE + APP\_SUBSTATE\_DFU,  APP\_STATE\_IN\_CASE\_DISCONNECTING = APP\_STATE\_IN\_CASE + APP\_SUBSTATE\_DISCONNECTING,   /\*!< Earbud is out of the case, parent state \*/  APP\_STATE\_OUT\_OF\_CASE = APP\_STATE\_IN\_CASE<<1, /// 0X400N  APP\_STATE\_OUT\_OF\_CASE\_TERMINATING = APP\_STATE\_OUT\_OF\_CASE + APP\_SUBSTATE\_TERMINATING,  APP\_STATE\_OUT\_OF\_CASE\_SOPORIFIC = APP\_STATE\_OUT\_OF\_CASE + APP\_SUBSTATE\_SOPORIFIC,  APP\_STATE\_OUT\_OF\_CASE\_SOPORIFIC\_TERMINATING = APP\_STATE\_OUT\_OF\_CASE + APP\_SUBSTATE\_SOPORIFIC\_TERMINATING,  APP\_STATE\_OUT\_OF\_CASE\_IDLE = APP\_STATE\_OUT\_OF\_CASE + APP\_SUBSTATE\_IDLE,  APP\_STATE\_OUT\_OF\_CASE\_BUSY = APP\_STATE\_OUT\_OF\_CASE + APP\_SUBSTATE\_BUSY,  APP\_STATE\_OUT\_OF\_CASE\_DISCONNECTING = APP\_STATE\_OUT\_OF\_CASE + APP\_SUBSTATE\_DISCONNECTING,   /\*!< Earbud in in ear, parent state \*/  APP\_STATE\_IN\_EAR = APP\_STATE\_OUT\_OF\_CASE<<1, /// 0X800N  APP\_STATE\_IN\_EAR\_TERMINATING = APP\_STATE\_IN\_EAR + APP\_SUBSTATE\_TERMINATING,  APP\_STATE\_IN\_EAR\_IDLE = APP\_STATE\_IN\_EAR + APP\_SUBSTATE\_IDLE,  APP\_STATE\_IN\_EAR\_BUSY = APP\_STATE\_IN\_EAR + APP\_SUBSTATE\_BUSY,  APP\_STATE\_IN\_EAR\_DISCONNECTING = APP\_STATE\_IN\_EAR + APP\_SUBSTATE\_DISCONNECTING, } appState; |

# 核心流程

* 同步左右耳机状态，再决定做相应的操作。