

SYD8801 OTA 设备端使用说明

一、简介

SYD8801 设备端使用 A、B 区的方式储存代码，即当前程序是在存储在 A 区，OTA 将新程序写入 B 区，然后重启系统，程序从 B 区开始执行，故中途断开连接或者中断 OTA 不会造成设备“变砖”。A、B 区随着 OTA 的次数相互切换。

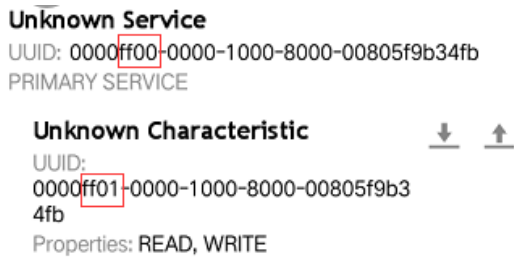
设备端大部分功能已经在 ota.h、ota.c 实现，只需实现相应的服务特性读写，以及调用相关 API 即可。

二、OTA 升级，设备端需要条件

1、实现服务以及对应的特性读写

<1>服务 UUID 为 FF00

<2>特性 UUID 为 FF01 权限可读、可写



2、keil 工程包含 ota.c，ota.h，并调用相关函数

3、OTA 之前，最好能将蓝牙的连接间隔变小、以便提升 OTA 的速度。

三、OTA 实现

第一、实现设备端与 APP 数据接收与返回

```

main.c lib.h ota.c ota.h
986 void ble_evt_callback(struct gap_ble_evt *p_evt)
987 {
988     if(p_evt->evt_code == GAP_EVT_ADV_END)
989     {
990     }
991     else if(p_evt->evt_code == GAP_EVT_CONNECTED) //连接事件
992     {
993     }
994     else if(p_evt->evt_code == GAP_EVT_DISCONNECTED) //断连事件
995     {
996     }
997     else if(p_evt->evt_code == GAP_EVT_ATT_HANDLE_CONFIGURE)
998     {
999     }
1000     else if(p_evt->evt_code == GAP_EVT_ATT_WRITE)
1001     {
1002     #ifdef _OTA_
1003     if(p_evt->evt.att_write_evt.uuid == BLE_SERVICE_UUID_OTA_READ_WRITE)
1004     {
1005         update_latency_mode=0;
1006         ota_cmd(p_evt->evt.att_write_evt.data, p_evt->evt.att_write_evt.sz);
1007     }else
1008     #endif
1009     {
1010         ble_gatt_write(p_evt->evt.att_write_evt);
1011     }
1012     }
1013     else if(p_evt->evt_code == GAP_EVT_ATT_READ)
1014     {
1015     #ifdef _OTA_
1016     if(p_evt->evt.att_read_evt.uuid == BLE_SERVICE_UUID_OTA_READ_WRITE)
1017     {
1018         uint8_t sz=0;
1019         uint8_t rsp[sizeof(struct gap_ble_evt)]=0;
1020         ota_rsp(rsp, &sz);
1021         SetGATTReadResp(sz, rsp);
1022     } else
1023     #endif
1024     {
1025         ble_gatt_read(p_evt->evt.att_read_evt);
1026     }
1027     }
1028     // #ifdef CONFIG_MARCH_STATE
1029     // march state.state =MARCH_STATE_ATTREAD | MARCH_STATE_NOTIFY;
1030     //

```

蓝牙协议栈回调

接收APP发来的OTA数据

在OTA.C

APP的OTA读请求返回处理

第二、main.c 实现 ota 状态管理函数——ota_manage

```

main.c lib.h ota.c ota.h
1677 }
1678 #endif
1679 //oled_close_down_init(0);
1680 }
1681 }
1682 }
1683 }
1684 void ota_manage(void){
1685     #ifdef _OTA_
1686     if(ota_state){
1687         uint8_t ota_callback[2]={0};
1688         switch(ota_state){
1689             case 1 :
1690                 #ifdef _DEBUG_
1691                 dbg_printf("OTA start\r\n");
1692                 #endif
1693                 break;
1694             case 2 :
1695                 #ifdef _DEBUG_
1696                 dbg_printf("OTA ing\r\n");
1697                 #endif //oled_close_down_init(255);
1698                 break;
1699             case 3 :
1700                 ota_state=0;
1701                 #ifdef _DEBUG_
1702                 dbg_printf("OTA finish\r\n");
1703                 #endif
1704                 SystemReset();
1705                 delay_ms(400);
1706                 #ifdef _DEBUG_
1707                 dbg_printf("Reset failed \r\n");
1708                 #endif
1709                 break;
1710             default :
1711                 break;
1712         }
1713     }
1714     #endif
1715 }
1716

```

OTA完成必须软复位或手动复位

第三、在 main 函数的 while(1)调用 ota_manage 函数

```
main.c lib.h ota.c ota.h
1943 wdt_enable(0x400); // 256 * 8 ms = 2*60ms = 2 s
1944 #endif
1945 __enable_irq();
1946
1947 motor_shock_time=10;
1948
1949 while(1)
1950 {
1951     ble_sched_execute();
1952
1953     if(timer1s_inting){ //1s定时器
1954
1955         ota_manage();
1956
1957         Timer_Evt_Handle_1s();
1958
1959         #ifdef _WDT_
1960         wdt_clear();
1961         #endif
1962
1963         #ifdef USER_32K_CLOCK_RCOSC
1964         if(SYD 1S EVT&SYD 1S 185S)
1965         }
```

调用这个OTA管理函数，不一定是1s调用一次，但是最好能在while（1）中

名称	日期	撰写人	版本
SYD8801 OTA 设备端使用说明.PDF	2018 年 3 月 20 日	Bihu	0.1