SYDTEK OTA 通信协议说明

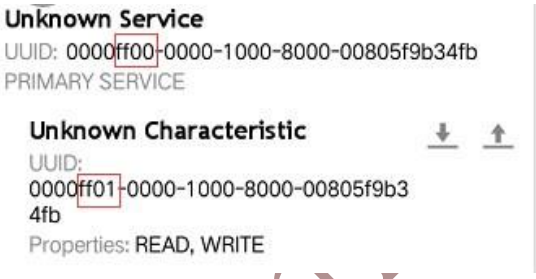
SYDTEK的OTA协议适用于SYDTEK公司旗下所有系列的芯片，这里为了方便说明，就用SYD8801来作为例子。

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。

二、OTA 升级需要条件

1、设备端实现 OTA 接收协议
2、APP 端实现了 OTA 的发送协议
3、设备端蓝牙的 Profile 含有服务 0xFF00，和该服务下有一个可读写的特性 0xFF01



三、OTA 过程

OTA 一共分三步，分别对应三个命令。每一步都是 APP 往 UUID 为 0xFF01 的特性里面写数据，或者读取状态值（为提高 OTA 速度，可以不读取状态值）
1、CMD_FW_ERASE（0x16）。发送擦除命令，擦除程序存储 B 扇区（不是当前程序 A 扇区）
2、CMD_FW_WRITE（0x17）。发送新程序，设备端写入另一扇区
3、CMD_FW_UPGRADE（0x18）发送更新命令，设备进行复位，程序执行 OTA 程序

四、步骤详解

第一步：CMD_FW_ERASE

1、APP 连设备，获取到 UUID 为 0xFF01 的特性，并往该特性写入 0x16、0x00（两个字节），即：
data[0] = 0x16 (opcode，固定值)
data[1] = 0x00 (parameters length，固定值)
2、成功发送后，可读取改特性 0xFF01 的值，返回格式如下：
data[0] = 0x0E (event code，固定值)
data[1] = 0x02 (event length，固定值)
data[2] = 0x16 (opcode，固定值)
data[3] = 0x00 (status，0x00 表示成功，其余值失败)

第二步：CMD_FW_ WRITE

1、APP 获取到 OTA 的 bin 文件，分包往 UUID 为 0xFF01 特性发送。每一包的格式如下
data[0] = 0x17 (opcode，固定值)

data[1] = 0x13 (parameters length，固定值)
data[2] = 0x?? (fw offset 0)
data[3] = 0x?? (fw offset 1)
data[4] = 0x?? (fw size)
data[5~21] = 0x?? (fw data)
解析:
<1>、fw offset 0、fw offset 1 两个字节表示第几包，即第（fw offset 1*256+ fw offset 0）包
<2>、fw size：表示该包（data[5~21]）具有几个有效的字节数
<3>、fw data：bin 文件拆包数据
2、每成功发送一包，可读取改特性 0xFF01 的值，返回格式如下：
data[0] = 0x0E (event code，固定值)
data[1] = 0x02 (event length，固定值)
data[2] = 0x17 (opcod，固定值 e)
data[3] = 0x00 (status，0x00 表示成功，其余值失败)

第三步：CMD_FW_ UPGRADE

1、当 APP 执行完前面两步，就执行第三步，即往 UUID 为 0xFF01 特性写入如下命令
data[0] = 0x18 (opcode，固定值)
data[1] = 0x04 (parameters length，固定值)
data[2] = 0x?? (fw total size 0)
data[3] = 0x?? (fw total size 1)
data[4] = 0x?? (fw checksum 0)
data[5] = 0x?? (fw checksum 1)
解析:
<1>、fw total size 0、fw total size 1 两个字节表示总发送了多少包，即总（fw total size 1*256+ fw total size 0）包
<2>、fw checksum 0、fw checksum 1 两个字节表示 bin 文件按照字节求和，再取低 16 位 checksum。fw checksum 0 =（checksum& 0x00FF），fw checksum 0 =（（checksum&0xFF00）>>8）
2、成功发送后，可读取改特性 0xFF01 的值，返回格式如下：
data[0] = 0x0E (event code，固定值)
data[1] = 0x02 (event length，固定值)
data[2] = 0x18 (opcode，固定值)
data[3] = 0x00 (status，0x00 表示成功，其余值失败)

名称	日期	撰写人	版本
SYD8801 OTA 通信协议说明.PDF	2018 年 3 月 20 日	Bihu	0.1

Version 2.0版本协议补充说明：

上文中所述都是针对Version 1.0版本的协议，为了弥补Version 1.0版本在空升代码量只能够达到65536Byte的局限性，这里开出Version 2.0的版本。

不能够升级大于65536Byte的代码主要是因为CMD_FW_UPGRADE命令中的” sz”时候“uint16_t”类型的，所以这里增加一个新的命令CMD_FW_UPGRADEV20替换原来的CMD_FW_UPGRADE命令，当执行到上文提到的第三步的时候APP按照如下格式发送命令：

data[0] = 0x18 (opcode，固定值)

data[1] = 0x04 (parameters length，固定值)

data[2] = 0x?? (fw total size 0)

data[3] = 0x?? (fw total size 1)

data[4] = 0x?? (fw total size 2)

data[5] = 0x?? (fw total size 3)

data[6] = 0x?? (fw checksum 0)

data[7] = 0x?? (fw checksum 1)

Version 3.0版本协议补充说明：

Version 2.0虽然解决了OTA空间大小的问题，但是因为他是基于Version 1.0的，所以OTA的速度依旧达不到理想的效果，特别是在SYD8821和SYD8811中代码量至少达到了248KB以上，这时候如果还用原来协议OTA的速度的体验不是很好。

这里改掉CMD_FW_WRITE的协议，原来的一个CMD_FW_WRITE命令只是发送了15个byte的数据，这里改成了一个命令发送20个byte，同时这里采用分段发送，每个段都会进行校验，如果检验不通过会重发当前段！具体修改如下：

1. 在发出上面的第一步发送了擦除命令后，这里不进入CMD_FW_WRITE流程，而是进入进入了一个新的写机制！这里发送一个新的命令CMD_FW_WRITE_START,其格式如下：

```
WriteData[0] = CMD_FW_WRITE_START;
WriteData[1] = 0x13;
WriteData[2] = (byte)(Address & 0x000000FF);
WriteData[3] = (byte)((Address & 0x0000FF00)>>8);
WriteData[4] = (byte)((Address & 0x00FF0000)>>16);
WriteData[5] = (byte)((Address & 0xFF000000)>>24);
WriteData[6] = (byte)(size & 0x000000FF);
WriteData[7] = (byte)((size & 0x0000FF00)>>8);
WriteData[8] = (byte)(check& 0x000000FF);
WriteData[9] = (byte)((check & 0x0000FF00)>>8);
```

其中Address代表的是段序列，在官方推出的demo中，每段有5120个数据。Size代表该段中有多少数据，在官方demo中，该值为5120，check代表该段的校验和值。

2. 当发送了CMD_FW_WRITE_START命令后将发送CMD_FW_WRITE_START命令中size指定大小的数据，一直到整个文件发送完成即该段数据全部发送完成。
3. 当该段数据发送完后将发起一次对设备端的读操作，读回的数据包含了设备端计算出来的校验和值。当设备端的检验和等于APP本地的检验和一样则代表当前段的数据发送正确，这里将发送新段落的数据；当设备端的检验和等于APP本地的检验和不一样则代表当前段落数据发送失败，则重发当前段落的数据！