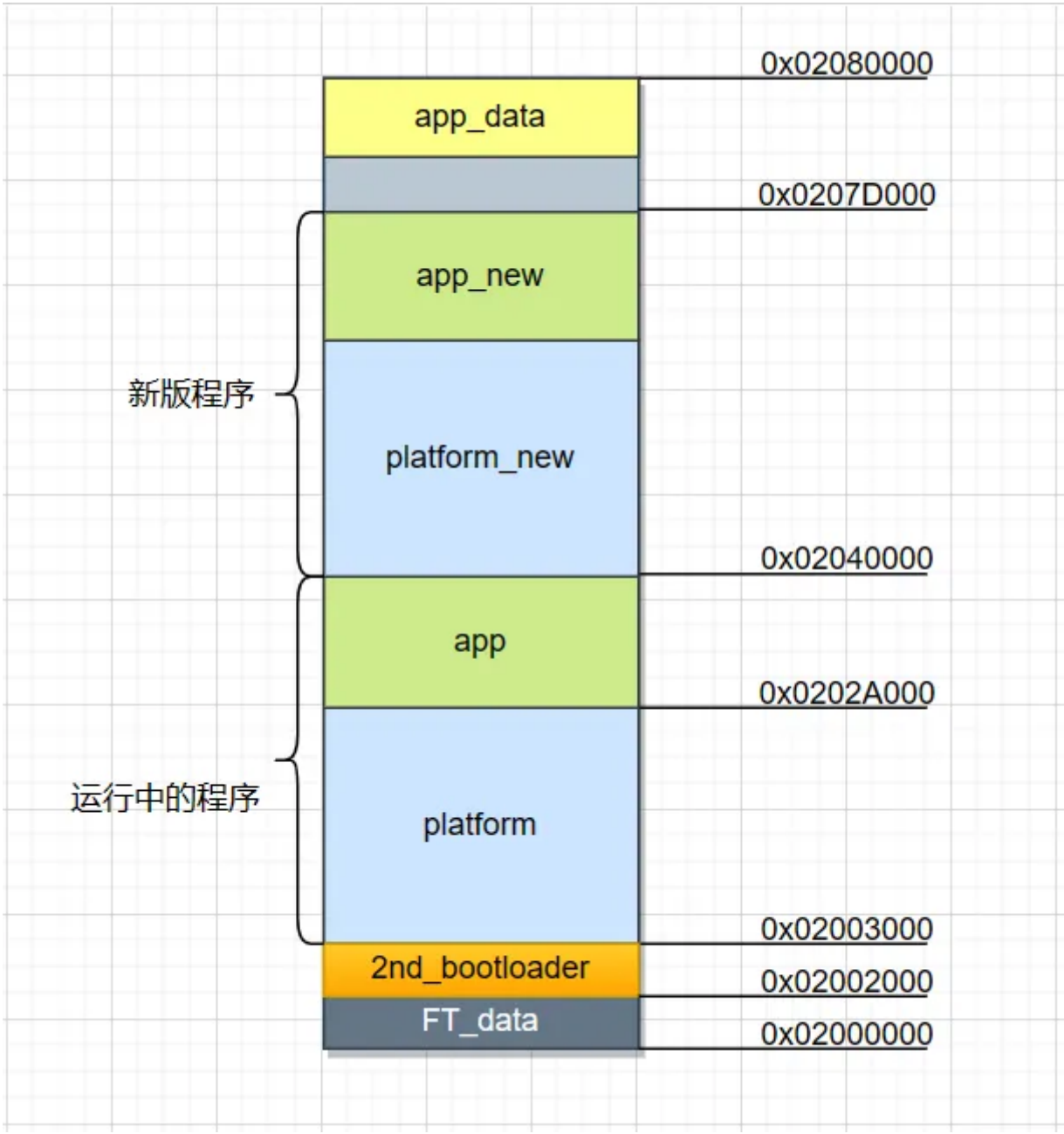
二级Bootloader+IAP方案Flash分布

二级Bootloader+IAP方案Flash分布

一步升级

使用双bank的方式，留出一份大小和待更新区域一样的独立存储区域，由二级bootloader将新程序覆盖旧版本程序来完成升级

一个典型的支持双bank升级的升级时flash结构要能同时容纳图中的几个部分：FT_data、2nd_bootloader、platform、app、platform_new、app_new、app_data

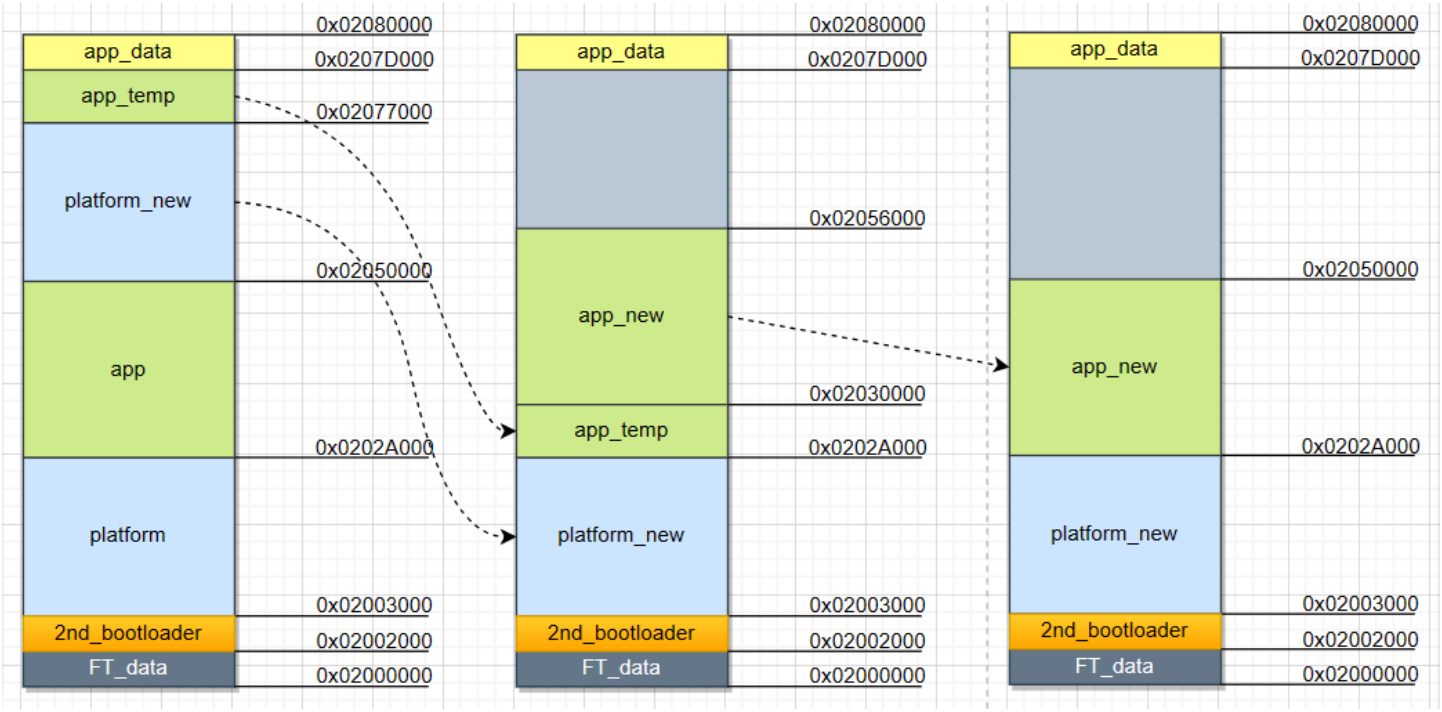


名称	描述	大小
FT_data + 2ndbootloader	二级bootloader以及FT数据存放区域	3个sector，大小固定
platform	正在使用中的platform存储的区域	39个sector，大小基本固定，除非platform有很大的改动
app	正在使用中的app存储的区域	大小不确定
platform_new	新版本platform的临时存储区域	39个sector，大小基本固定，除非platform有很大的改动，当只升级APP时，这一块不会用到
app_new	新版本app的临时存储区域	大小不确定
app_data	应用保存数据的区域（蓝牙配对信息等）	大小不确定

- 由于916 flash一共是128个sector，每个sector的大小为4096Byte，通过简单的计算可以得到，app、app_new、app_data 这三块区域的总大小不能超过50个sector，当 app_data 用到4个sector时， app+app_new 就不能超过46个sector了，如果app和app_new大小相当，那么 app 的大小需要限制在23个sector内，这种情况下 app 的大小就不能超过**94208Byte**，超过了会有可能导致没有足够的flash空间来存储新版程序，从而导致无法升级

分两步升级

当APP的体积很大，超出了满足单步升级的最大限制，并且难以缩减体积时，对于这种情况，可以采用分两步升级的方式，过程如下：



1. 升级新版platform和一份仅有升级功能的APP（先称作app_temp，因为仅有升级功能，所以程序体积会比较小）

2. 升级新版的APP

经过初步评估，仅包含USB_IAP升级功能的 app_temp 的大小在0x5EE0Byte左右，需要6个sector

名称	描述	大小
FT_data + 2ndbootloader	二级bootloader以及FT数据存放区域	3个sector，大小固定
platform	正在使用中的platform存储的区域	39个sector，大小基本固定，除非platform有很大的改动
app	正在使用中的app存储的区域	大小不确定
platform_new	新版本platform的临时存储区域	39个sector，大小基本固定，除非platform有很大的改动，当只升级APP时，这一块不会用到
app_temp	仅包含升级功能	6个sector，大小基本固定
app_data	应用保存数据的区域（蓝牙配对信息等）	大小不确定

由于flash一共是128个sector，通过简单的计算可以得到，app、app_data 这两块区域的总大小不能超过41个sector，当 app_data 用到4个sector时， app 就不能超过37个sector了，这种情况下app的大小就不能超过**151552Byte**，超过了会有可能导致没有足够的flash空间来存储新版程序，从而导致无法升级。

相比于单步升级，分两步升级可以允许APP体积更大

- 单步升级，估算app体积不能超过**94208Byte**
- 分两步升级，估算app体积不能超过**151552Byte**

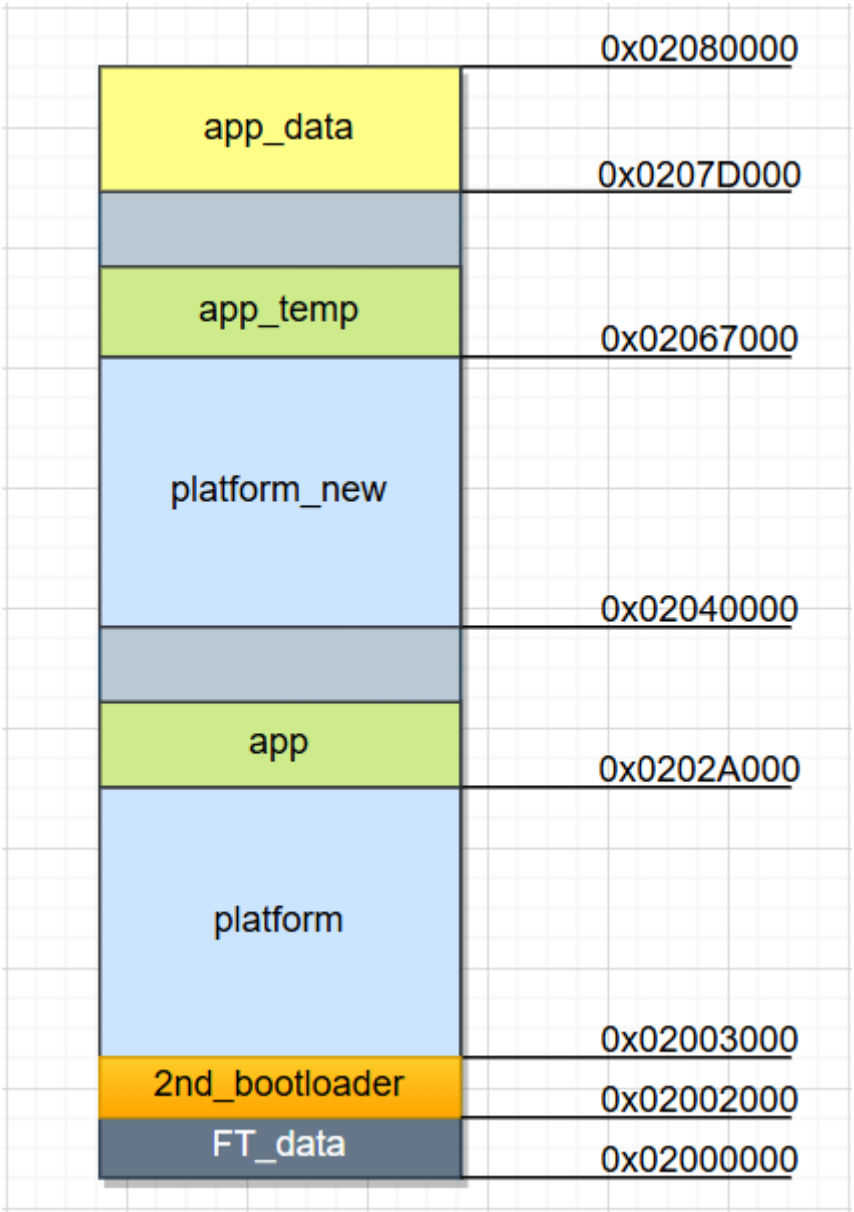
分两步升级的缺点很明显

1. 升级时需要分两步，整体更复杂
2. 需要额外维护一个仅包含USB_IAP升级功能的app

*航晟键盘项目boot + app的总大小为：**130244Byte**，目前来看，采用单步升级的方式就会出现问题，但是却可以采用分两步升级的方式来实现IAP升级。*

固定大小的双bank升级

这种升级方式是一步升级的其中一种情况，这种升级方式的flash分布更为固定，如下图所示：



相比于更一般的一步升级，固定的双bank升级将flash减去底部3个sector（FT_data + 2nd_bootloader的区域）再减去顶部3个sector（应用保存数据的区域）再将flash对半分开，一半存放正在运行的程序，另一半存放ota时的新版程序。

名称	描述	大小
FT_data + 2ndbootloader	二级bootloader以及FT数据存放区域	3个sector，大小固定
platform	正在使用中的platform存储的区域	39个sector，大小基本固定，除非platform有很大的改动
app	正在使用中的app存储的区域	大小不确定
platform_new	新版本platform的临时存储区域	39个sector，大小基本固定，除非platform有很大的改动，当只升级APP时，这一块不会用到
app_new	新版本app的临时存储区域	大小不确定

app_data	应用保存数据的区域（蓝牙配对信息等）	3个sector，大小固定

由于flash一共是128个sector，通过简单的计算可以得到，app和app_new区域的大小都不能超过22个sector（(128 - 3 - 39 - 39 - 3) / 2），这种情况下app的大小就不能超过90112**Byte**，超过了会有可能导致没有足够的flash空间来存储新版程序，从而导致无法升级。

相比于更一般的单步升级

1. app大小会限制的稍微更死一点
2. 新版程序固定存放在0x02040000~0x0207D000的位置
3. 相对更稳定安全

应用二级bootloader

参考文档

为 ING916 添加二级 Bootloader

ING916 二级Bootloader 调试记录

步骤简述

1. 保证工程使用COPY的SDK
2. 使用脚本修改platform.bin
3. 修改IROM适应地址变化，修改烧录地址
4. 烧录二级bootloader以及platform+app

添加IAP软件模块

IAP升级协议

IAP协议基于航晟键盘项目的协议，并在其基础上修改部分内容以适配二级bootloader的方式

IAP通讯协议

原始协议在二级Bootloader结构上应用时的问题

原本的协议是基于platform+boot+app的结构进行的，并且只会升级APP，APP的起始地址是一个固定值不能变，现在升级的架构变更为2ndbootloader+platform+app，并且允许升级platform，因为platform的大小可能会根据SDK版本变化，这会导致APP的起始地址并不固定，旧版程序不能确定新版程序的APP的起始地址，协议中目前没有字段可以表明新版APP的目标地址。

协议改动说明

升级协议不变，IAP_BIN_HEADER的尾部追加几个字段，前面的内容格式保持不变

- bin的load_address（需要开发人员在bin生成工具上设置该项目）
- bin文件的原始大小（merge bin的文件大小）
- 对header部分校验的CRC

序号	数据项	长度	偏移地址	数据类型	说明
11	**load_address**	4	106	HEX	新版程序的加载地址，对齐到sector，升级platform+app时固定为0x02003000
12	**bin_size**	4	110	HEX	原始bin文件的大小，升级platform+app时为merge后的bin文件大小
13	**填充字段**	12	114	HEX	预留字段，将header填充，默认值0xFF
14	**CRC**	2	126	HEX	对header的校验，校验范围：[0, 126)，校验算法一致(crc16_modbus)

IAP程序模块需要配置一个目前留给IAP升级存放新版程序的数据区域（存放platform_new和app_new的区域），称作IAP storage，由两个宏来表示

- IAP_STORAGE_ADDR：例如：0x02041000，需要对齐到扇区
- IAP_STORAGE_SIZE：例如：0x0003B000

升级流程的部分变化

1. 进行header部分的crc校验
2. 如果load_address不在0x02003000~0x02040000范围内或没有对齐到sector就返回错误：非法的load_address
3. 如果bin_size的值超过了IAP storage的总大小就返回错误：非法的bin_size
4. 后续的bin数据会先保存在IAP storage（从IAP_STORAGE_ADDR开始存）
5. 如果有设置加密就先进行就地解密（in-place decryption），由于数据存储在Flash中，执行就地解密可能会受到Flash特性的限制。
6. 创建升级元数据
    - meta_block.src设置为IAP_STORAGE_ADDR
    - meta_block.dest设置为load_address

    - meta_block.size设置为header中的bin_size字段值
7. platform_reset()，二级bootloader会将新版程序从IAP storage搬运到load_address，并启动

其他

- 不启用加密的情况下header中的load_address和size字段和bin数据一样会被窃取。header中的字段通过header中的新添加的crc来一定程度上防止篡改
- 启用加密的情况下header中的load_address和size字段不会被窃取，但是可能被篡改，程序端会对size和load_address的值做有效性的验证，但是还是有很小的可能，值被篡改并且通过了有效性验证。

其他

若采用两步升级方式，需要提供

- platform_newapp_temp 合并的bin文件，进行第一步升级

1. app_new 单独的bin文件，进行第二步升级

若有需要，可以让IAP_tool来自动处理这两步逻辑，用户只需要提供三个原始bin文件以及相关配置