关于 STM32 的 IAP 总结

最近有项目要用到 IAP 的功能,于是调试了下 STM32 的 IAP,可能因为个人水平的原因吧,也颇费了一般周折
现在返回头来想,其实还是蛮简单的.
整个过程按照如下步骤:

1. 解锁
2. 判断是否保护,有保护的话要先关闭保护
3. 擦除
4. 编程
5. 复位进入应用程序区

关于解锁:看资料的时候说的神乎其神,有个读/编程控制器叫"FPEC
有几个寄存器,专门负责 Flash 的,对这几个寄存器以一定得顺序访问并设置即可成功解锁 Flash,至于怎么访问,谁先谁后,数据手册上写的头晕,直接来个快刀斩乱麻 Flash_UnLock()函数封装了这一系列的操作,有一点要注意,如果你是自己操作寄存器的话,如果操作的方法或者顺序不对都会造成 Flash 的锁定,之后的所有操作都会返回一个错误,直到下次启动后才能正常操作

关于保护,为了保护用户数据不被无意修改或者恶意读取,STM32 提供了对芯片 FLASH 的写读等一系列的保护,加密方式是按照每 4 页为一个单位,也就是说,如果你想加密的话,你至少要加密 4 页,也就至少 4K 的空间,至于高密的 STM32 是否就是 8K 了?这个我没仔细去看!还待以后仔细查看?

关于擦除,擦除也是很简单,但是只能一页一页的擦除,ST 公司也提供了一个函数,至于这个函数后面的输入地址参数,经过试验发现只要这个地址落在这个页里,就是擦除这个页,不知道这样理解对不对,还需要验证???

```
FLASHStatus = FLASH_ErasePage(Address );
```

关于编程,STM32 编程一次只能以半字(16 位)的方式编程,库提供了两个函数

```
FLASH_Status FLASH_ProgramWord(u32 Address, u32 Data)   编程一个字
FLASH_Status FLASH_ProgramHalfWord(u32 Address, u16 Data)   编程半个字
```

在实际编程虽然你调用的一个字编程的时候内部操作仍然是按照半字的方式编程
另外还有个最最重要的一点:还要注意大小端的问题,有些你认为可移植性很好的代码,其实并不一定,用位移组合成一个32位整型,然后当做参数来编程,由于大小端的模式,刚好第一个字节在最后边了,最后一个字节在前面了,导致了AN2557下载我的代码可以使用,我自己的下载我自己的代码竟然不能使用,很是郁闷了一阵子,读出整个Flash的内容就很容易看出来不同了,这个没有想到也着实该死,后来用指针强指,不但效率高了,程序也方便了,汗一个!

为什么有些人说STM32的IAP其实一个半成品,不成熟的呢????这个还需以后了解!

关于应用程序还有些要说，在写应用程序的时候，只要记住，你的应用程序放在哪个地方，记得要在工程设置里设置一下，我用的是 MDK，这点比 IAR 方便一些，指定你的代码从哪个地址加载，然后还要指定你的代码区的 Size，也就是你芯片 flash 的总 size 减去 IAP 程序的 Size 后得到的，这个设置的地方在 project Options for Target 的 IROM1 这个位置指定，还有一个地方，在 Linker 页面有个项也要指定

R/O Base 设置包含 **RO** 输出段的常量和代码域地址。地址必须为字对齐。如果没有指定，缺省的 **RO** 地址为 0x8000。这里也要设置成应用程序启动的地址 R/O 基地址含有初始化程序的入口地址。

最最重要的一点，中断向量表的映射，因为应用程序从其它地方启动了，向量表同样也要增加偏移，不然中断入口找不到可是个麻烦事

```
NVIC_SetVectorTable(NVIC_VectTab_FLASH, 0x2000);
```

这个函数就是做这个的，第二个参数是相对的，不是绝对的，也就是基地址+第二个参数

其它好像也没什么了

整理：Fei.Mu mu.blank@gmail.com 09/07/24