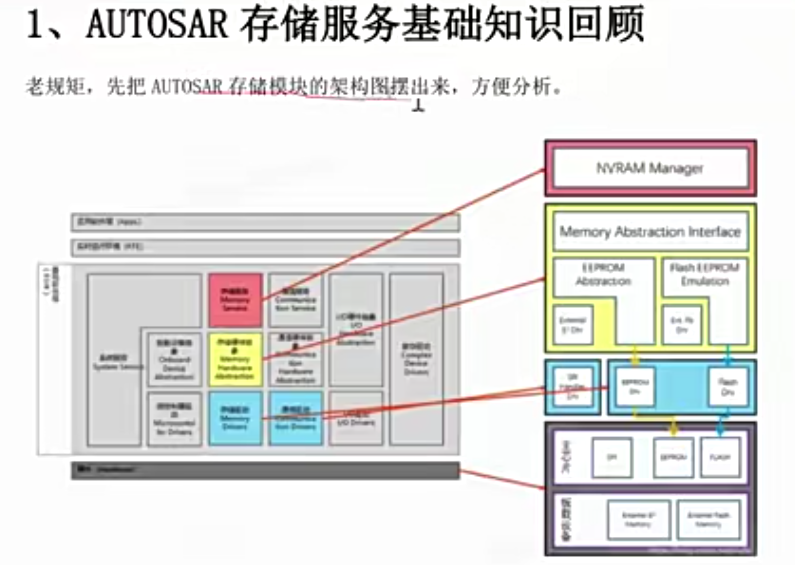
# Autosar存储服务之Fee换页策略

## AutoSAR存储服务基础知识回顾

老规矩，先把AUTOSAR存储模块的架构图摆出来，方便分析



存储基础知识请参考上一篇文章：

在TC397芯片部署AUTOSAR，存储模块软件架构分层依次是：

**NVM->MemIf->Fee->FLS->FLASH(TC397上使用的方案)**

NVM是存储服务层，也就是存储模块最上层，该层是对外提供的接口。

MemIf是接口层，区分下面是Fee还是Ea。

Fee是Flash模拟EEPROM的抽象层。

Fls是flash驱动。在TC397中有一块DFLASH，AutoSAR规范中是通过DFLASH模拟EEPROM的形式将数据存到DFLASH中的。

**Page**：DFLASH最小编程单元，大小为8byte，意味着即使只想写入1个字节的数据，也会将对应的8字节数据重新写入。这就要求Fee软件层需要先读取出相应Page的数据，修改其中的1个字节，再将8byte写入。

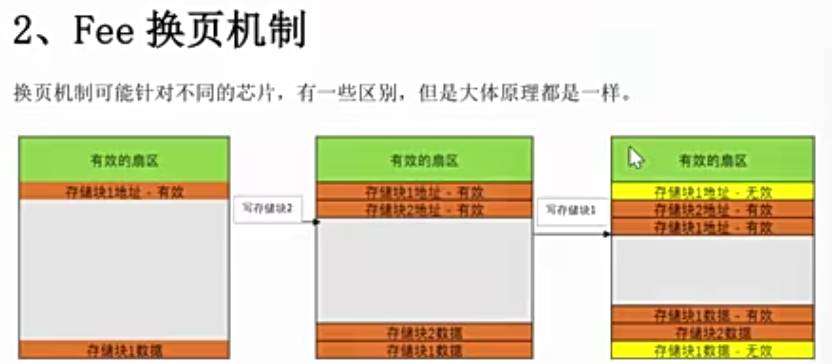
**Logical Section**:逻辑扇区，擦除的最小单元，由若干个Page构成，配置成2k，就相当于一次擦除最少擦除2k大小

**Physical Section**:物理扇区，由多个逻辑扇区组成。

**Data Flash**:可以由多个物理扇区组成，我们也称物理扇区为BANK，所需要写入的数据会最终写入到物理扇区中，一个物理扇区写满了就需要新的物理扇区，但是物理扇区是有限的，不能一直有新的BANK，针对同一个数据如果多次写入，其实对于程序而言只有最后一次是有效的，因此如何擦除无用数据并循环使用物理扇区就是解决这个问题的关键，这就是**FEE换页机制**。

## Fee换页机制

换页机制可能针对不同的芯片，有一些区别，但是大体原理都是一样。



第一次写入BLOCK1，Block 1地址状态为有效

第一次写入BLOCK2，Block 2地址状态为有效

第二次写入BLOCK1，将第一次写入的Block 1的地址状态置为无效，同时第二次写入的地址是有效

当APP请求读取BLOCK1的数据时，只会读取地址有效的数据。

**因此，如果针对同一个BLOCK数据多次写入，并不是将原来的内容覆盖，而是在扇区中写入新的数据，而将老的数据置为无效。**

**为什么对存在记录的数据不能采用覆盖原来数据的方式呢？**

如果采用覆盖的方式替换原来的旧有数据（针对同一BLOCK），“覆盖”这个动作可以理解为先擦除原有数据再在原地址写入新数据，这种操作从理论上是可行的。但实际上却又很多问题

正如上面我们介绍过，擦除的最小单元是逻辑扇区，写入的最小单元是Page，逻辑扇区要比PAGE大很多，如果用“覆盖的”方式意味着这个逻辑扇区中只能存在这一个BLOCK（避免擦除时将其他BLOCK DATA擦除掉，这么做会造成大量的空间浪费。）

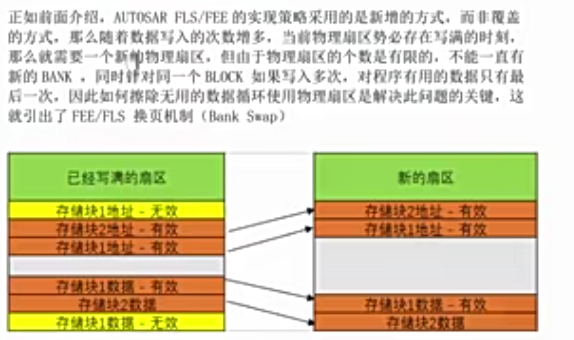
另外，Data Flash是有擦除寿命的，频繁的擦除是会减少D-Flash的使用寿命。

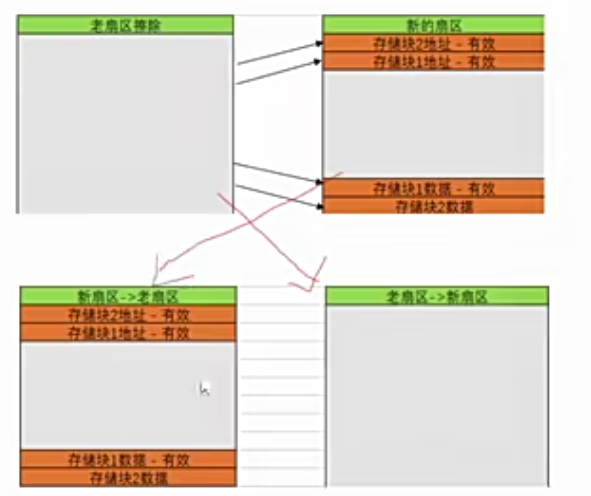
擦除是最耗时的，每次BLOCK写入都要擦除原有数据再写入的话，会让效率很低，而新增写入不需要每次擦除（会同一擦除），每次只需要直接写入，再整个BANK写满后进行换页的时候再同一擦除。

鉴于此，AUTOSAR FLS/FEE的实现策略并未采用这种“覆盖”的方式

## 换页机制（BANK SWAP）

正如签名介绍，AUTOSAR FLS/FEE的实现策略采用的时新增的方式，而非覆盖的方式，那么随着数据写入的次数增多，当前物理扇区势必存在写满的时刻，那么就需要一个新的物理扇区，但由于物理扇区的个数是有限的，不能一直有新的BANK，同时针对同一个BLOCK如果写入多次，对程序有用的数据只有最后一次。因此如何擦除无用的数据循环使用物理扇区是解决问题的关键，这就引出了FEE/FLS换页机制（Bank Swap）





**将所有Valid的数据拷贝到新的BANK**

**将原来Bank整体擦除**

**将擦除的BANK作为新的Bank供下一次换页使用**

**Note：这是2个BANK的换页策略，理论上可以有N个BANK，不过换页机制大体相近**

## 换页时BLOCK写入请求

可能有人会问如果在换页过程中，有新的BLOCK写的请求该如何处理，首先，针对BLOCK请求分为2种类型，immediate/Normal,对于immediate的写入请求，需要打断擦除，立刻将新的数据写入到New Bank中（有的FEE Driver可能会写入到“第3块BANK”），原则上对于immediate Write Request数据是不需要等待换页完成的。对于Normal的写入请求，则需要等待换页完成才可以写入，因此可能出现Block写入的时间会比较长的情况，在设计的时候需要考虑到此种情况。

Note：在NVM BLOCK配置中如果Priority为0，则此BLOCK的请求为Immediate request反之为normal request.

## 如何设计BANK的大小

正如前面的介绍，BANK换页的工作原理，可以有如下总线

|  |  |  |
| --- | --- | --- |
| 单个BANK大小 | 优点 | 缺点 |
| 大 | 换页次数少，D-FLASH使用寿命长 | 换页时间长 |
| 小 | 换页次数频繁，D-FLASH使用寿命短 | 换页时间短 |

NOTE:一般D-FLSAH的会规定最多擦除次数

**那么我们应该如何设计BANK的大小呢？**

1. 我们的ECU都是有**设计使用时长**的，因此首选BANK的大小需要满足ECU的设计使用时长的，假设，BANK>30K就可以满足
2. 在这里我们需要估算出最多可能有**多少数据写入**，每隔一段时间又会有多少增量数据，进而推算出我们**换页的频率**
3. 在满足上一步的前提下，在根据ECU的特性，估算ECU所能接受的最大换页时长，正如前面说过的，对于**NORMAL Request**需要等待换页完成，因此写入时长会被拉长。假设，BANK<60K就可以满足
4. 综上单个BANK（更严谨的说法是单次擦除的所有BANK总大小，因此当我们存在多个BANK时，换页可能发生在写满N个BANK才会触发）的大小就是在30K-60K之间就可以。
5. 算出了BANK的大小，再结合整体D-FLASH的大小，我们就能估算出大概需要几个BANK了

## 总结

当FEE作为AUTOSAR存储模块的核心软件层，这一层需要考虑到FLASH使用寿命，BLOCK稳定写入、换页时长、启动换页阈值、BANK大小等关键功能设计。AUTOSAR作为车载软件的标准，在很多模块设计上都会考虑到实际应用场景，满足车载软件的高标准要求。