

引言

许多应用都需要用 EEPROM（电可擦可编程只读存储器）来存储非易失性数据。低成本的目的使得 STM32F101xx 和 STM32F103xx 器件没有使用 EEPROM。而是使用了嵌入式闪存来实现 EEPROM 模拟。

这篇应用笔记解释了外部 EEPROM 和嵌入式闪存的区别，并且它还描述了一种软件方法，这个方法是为了使用 STM32F101xx 和 STM32F103xx 器件的片上闪存来模拟 EEPROM。

这篇文档也致力于一些针对模拟 EEPROM 数据存储的嵌入式方面，这些是假定读者知道的。

术语

中等密度器件是指那些闪存存储范围在 32 和 128Kb 之间的 STM32F101xx 和 STM32F103xx 微控制器。

高密度器件是指那些闪存存储范围在 256 和 512Kb 之间的 STM32F101xx 和 STM32F103xx 微控制器。

目录

- 1 嵌入式闪存 vs EEPROM:主要区别
 - 1.1 写访问时间的区别
 - 1.2 写方法的区别
 - 1.3 擦除时间的区别
- 2 模拟 EEPROM 的实现
 - 2.1 原理
 - 2.1.1 应用举例
 - 2.1.2 EEPROM 软件描述
- 3 嵌入式应用方面
 - 3.1 数据间隔尺寸管理
 - 3.1.1 一个字一个字的编程
 - 3.1.2 一个字节一个字节的编程
 - 3.2 平均抹写存储区块：闪存持续时间的改善
 - 3.2.1 平均抹写存储区块实现的举例
 - 3.3 功率损耗下的页数据头恢复
 - 3.4 可擦写性能
- 4 出版历史

表格清单

表 1. 内嵌式闪存和 EEPROM 之间的区别

表 2. 状态组合和采取的行动

表 3. 模拟 EEPROM 中储存变量的最大值（10000 个周期）

表 4. 文档修正历史

图清单

图 1. Page0 和 Page1 之间的页状态切换

图 2. EEPROM 变量格式化

图 3. 数据更新流程

图 4. 写变量流程图

图 5. 四页的页交换计划（平均抹写存储区块）

1 嵌入式闪存 VS EEPROM:主要区别

电可擦可编程只读存储器（EEPROM）是许多嵌入式应用程序的关键组成部分，这些应用程序要求非易失性存储数据在运行的时候一个字或者一个字节的更新。

另一方面，微控制器在那些系统中每次使用的时候更多的是基于嵌入式闪存。为了消除组件，节省硅的空间和减少系统花费，STM32F10xxx 是用闪存来代替 EEPROM 去同步代码和存储数据。

不像闪存，外部 EEPROM 在重写数据的时候不需要一个擦除动作去挪出空间。所以一些特殊的软件管理是需要用嵌入式闪存来存储代码的。

很显然仿真软件方案取决于许多因素，包括 EEPROM 的可靠性，使用的闪存的构架，还有产品的需求。

嵌入式闪存和外部串口 EEPROM 的主要区别是使用相同闪存技术的任何通用微控制器（不是特别指 STM32F10xxx 系列产品）。主要区别在表 1 中概括起来了。

表 1. 内嵌式闪存和 EEPROM 之间的区别

特性	外部 EEPROM	模仿 EEPROM 使用片上 Flash memory
写时间	— 几个 ms — 随机字节：5 到 10ms — 页：每个字 100us（每页 5 到 10ms）	字程序时间：20us
擦除时间	N/A	页/块擦除时间：20ms
写方法	— 只要开始，便不再依靠 CPU — 仅仅需要适当的供应	一旦开始，便依靠 CPU：一个 CPU 复位将会停止写动作，即使电源停留在规格内
读访问	— 串口：100us — 随机字：92us — 页：22.5us 每个字节	— 并口：100ns — 非常少的 CPU 周期每个字 — 访问时间：35ns

写/擦除周期	— 从 10 千周到 1000 千周	— 从 10 千周到 100 千周（片上闪存页的使用相当于增加了写周期的次数）见 3.4 节：可擦写性能
--------	--------------------	--

1.1 写访问时间的区别

闪存有一个很短的写访问时间，相比较与外部串口 EEPROM，关键的参数可以更快的被存储到模拟 EEPROM 中去，然后改善数据存储。

1.2 写方法的区别

外部 EEPROM 和模拟 EEPROM 在嵌入式应用方面的一个主要区别就是写方法。

- **外部独立 EEPROM：**一旦被 CPU 启动，一个字的写动作便不能被 CPU 重启给打断。只有断电可以打断写过程，所以在独立 EEPROM 内，适当大小的退偶电容可以获得完整的写动作过程。
- **使用嵌入式闪存模拟 EEPROM：**一旦被 CPU 启动，一个字的写动作可以被断电和 CPU 重启打断。系统设计者应该分析这个差异，明白他们应用程序上的可能影响并且决定一个合适的处理方法。

1.3 擦除时间的区别

擦除时间的区别是独立 EEPROM 和使用嵌入式闪存模拟 EEPROM 之间的另一个主要区别。不像闪存，在写入它之前，EEPROM 不需要擦除动作去挪出空间，这就意味着一些形式的软件管理是需要存储数据在闪存中的。另外，当闪存中擦除块操作需要几毫秒的时候，掉电和一些其他的伪造事件可能打断擦除操作（例如重启），设计闪存管理软件的时候这点应该考虑到。设计一个完整的闪存管理软件，对于闪存擦除过程的深入理解是必不可少的。

2 模拟 EEPROM 的实现

2.1 原理

考虑到闪存的局限性和产品需求，模拟 EEPROM 有多种多样的执行方式。下面详细介绍的方法需要至少两页闪存，对于被分配的相同大小的非易失数据。一页是初始化擦写，并且支持一个字节一个字节的编程；另一页是当前页需要被垃圾收集的时候用来准备接管的。每页占用了第一个 16 位半字的头字段表明了页状态。

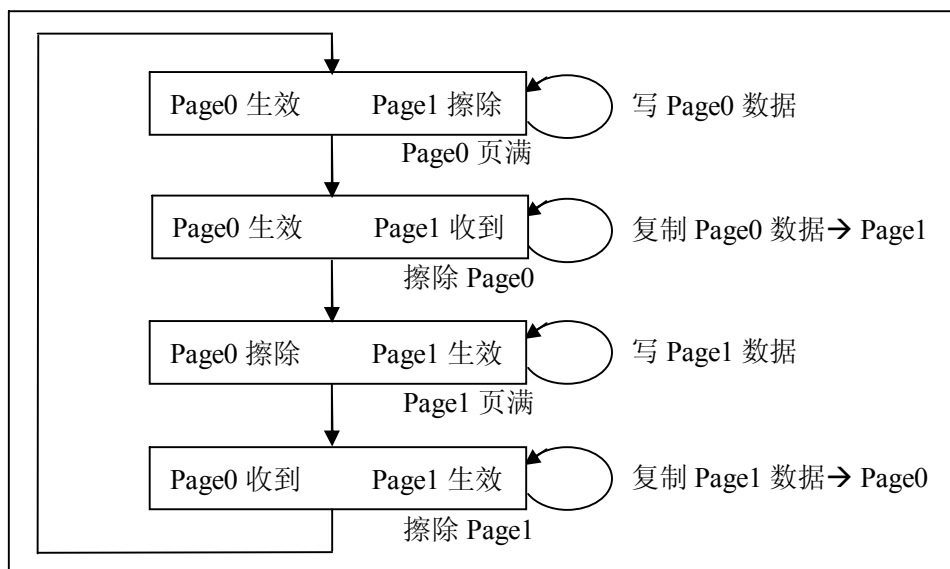
头字段的信息包含了每页的基地址和这页的状态。

每页有三种可能的状态：

- **擦除：**页是空的。
- **接受数据：**正在从其他满的页接受数据。
- **有效页：**这页包含有效数据并且这个状态时不会改变的，直到所有的有效数据全部被转移到擦除页。

图 1 展示了和其他页之间页状态是怎么改变的。

图 1. Page0 和 Page1 之间的页状态切换

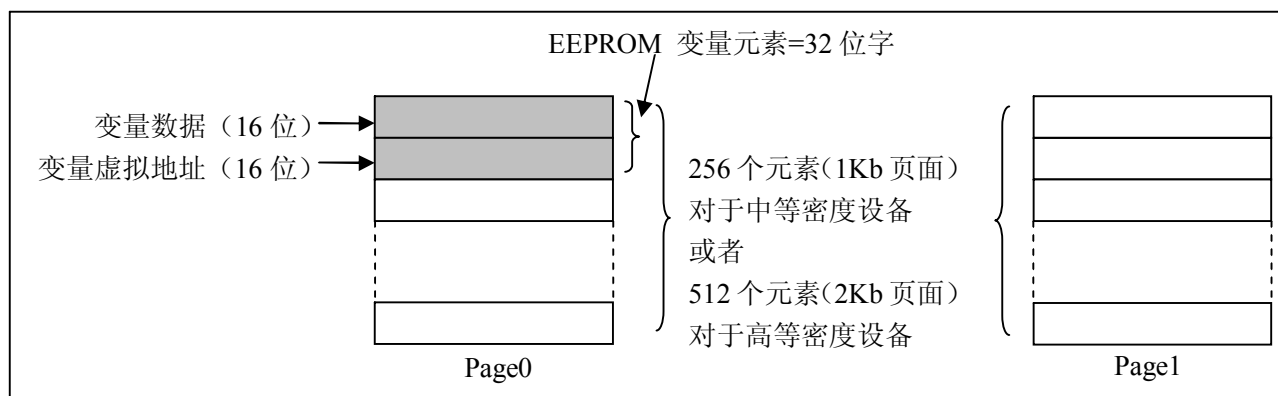


通常，在使用这个方式的时候，使用者预先不知道变量更新的频率。

这篇文章中描述软件和编译器使用两页闪存去模拟 EEPROM。

为了后续检索和更新，每一个变量元素都被定义为一个虚拟地址和一个存储在闪存中的值（在编译器软件中，虚拟地址和数据都是 16 位长）。当数据被修改的时候，与早先虚拟地址有关的数据被存储在一个新的闪存位置。数据检索返回修改的数据在闪存最后的位置。

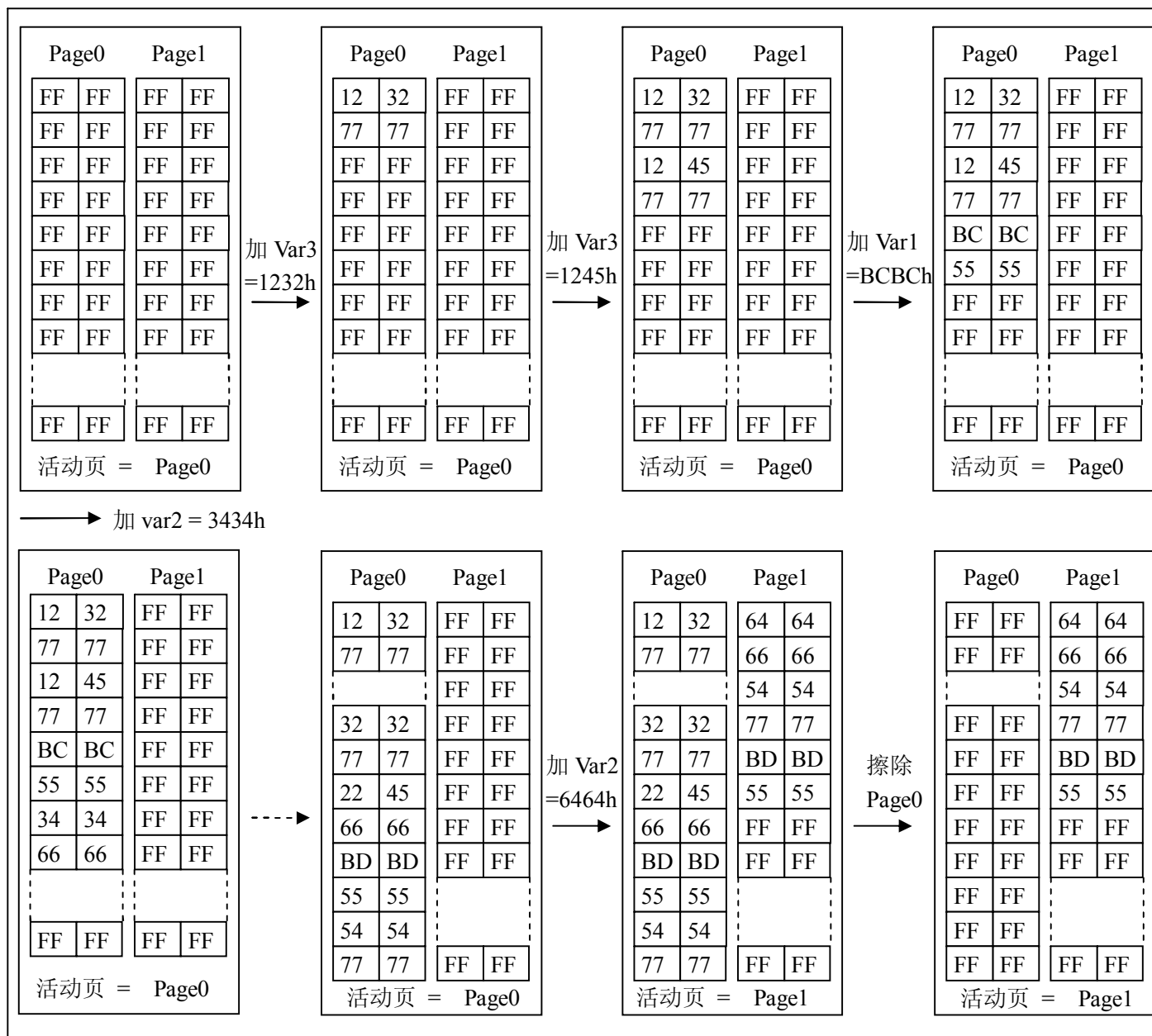
图 2. EEPROM 变量格式化



2.1.1 应用举例

下面的例子展示了三个 EEPROM 变量（Var1，Var2 和 Var3）同下面虚拟地址的软件管理： Var1: 5555h, Var1: 6666h, Var1: 7777h

图 3.数据更新流程



2.1.1 EEPROM 软件描述

本节描述了使用意法半导体公司提供 STM32F10xxx 闪存器件模拟 EEPROM 驱动程序的实现。

一个演示程序的例子也会被提供，用来论证和测试 EEPROM 模拟驱动程序，这里使用的三个变量 Var1, Var2 和 Var3 是表格 VirtAddVarTab 中定义的，VirtAddVarTab 表格在软件 main.c 文件里面声明的。

除了闪存的库文件，这个工程包含了三个源文件：

- eeprom.c: 它包含了下面工程常规用到的 c 代码：
EE_Init()
EE_Format()
EE_FindValidPage()
EE_VerifyPageFullWriteVariable()
EE_ReadVariable()
EE_PageTransfer()
EE_WriteVariable()
- eeprom.h: 它包含了常规的原型和一些声明。
- main.c: 这个应用程序是一个使用上面描述的常规的例子，为的是写入和读取 EEPROM。

用户 API 定义

函数的设置在 eeprom.c 文件内，然后用于模拟 EEPROM，描述如下：

- EE_Init()
- 扇区擦除或转移和数据更新的时候发生功率损耗，可能会发生扇区头讹误。在这种情况下，EE_Init()函数将会尝试修复数据库到一个良好的状态。每一次断电后访问数据库之前，这个函数都会被调用。它不接受任何参数。这个过程在表 2 中有描述。
- EE_Format()
这个函数擦除 page0 和 page1 并且写有效页头到 page0。
- EE_FindValidPage()
这个函数读取所有的页数据头并且返回有效页页码。这个返回的参数表明了有效页是否在寻求写入或读取操作 (READ_FROM_VALID_PAGE or WRITE_IN_VALID_PAGE)。
- EE_VerifyPageFullWriteVariable()
必须更新或者建立一个变量的第一个实例，它使这个写过程生效。它表现为找到活动页的第一个空位置，从最后开始，并写进去已经传送过的虚拟地址和变量的数据。在活动页满的情况下，返回页满值。这段程序用到了一下参数：
 - 虚拟地址：可能是任何三个已经声明的变量的虚拟地址 (Var1, Var2 或者 Var3)
 - 数据：将要存储变量的值

这个函数成功的话返回闪存完成，如果没有足够的内存区更新变量，就返回页满，或者闪存错误代码，用来指示操作失败（擦除或编程）。

- **EE_ReadVariable()**

这个函数返回的是虚拟地址所对应的数据，虚拟地址是作为一个参数被传送的。只有最后更新的被读取。这个函数进入的时候是一个循环，在这个循环内读变量目录知道最后一个。如果变量没有出现，读状态变量将返回 1，否则就重置表明变量已经被发现并且变量值被返回到 Read_data 变量。

- **EE_PageTransfer()**

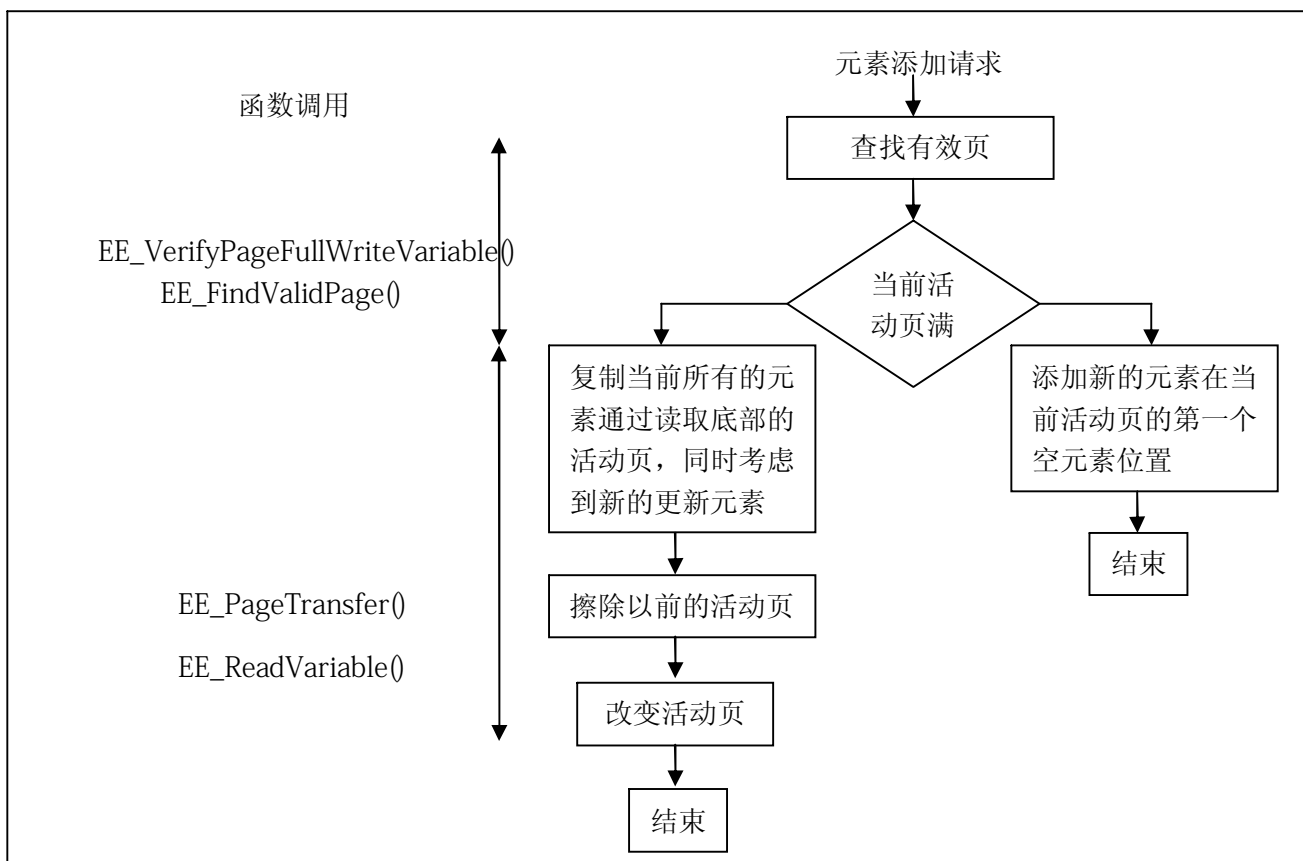
它转移最近期的数据（上次更新的变量），这些数据从一个满的页转移到一个空的里面。起初，他决定活动页，活动页就是要转移这页的数据的那一页。新页的数据头字段将被定义和写入（新页状态是 RECEIVE_DATA，它是在接收数据的过程中被给出的）。当数据传送完毕的时候，新页的数据头就是 VALID_PAGE，旧页将被擦除并且它的数据头回变成 ERASED。

- **EE_WriteVariable(..)**

当使用者更新变量的应用时，这个函数会被调用。它使用 EE_VerifyPageFullWriteVariable() 和 EE_PageTransfer() 常规，这些常规前面描述过。

图 4 展示了 EEPROM 更新数据变量入口点的过程

图 4.写变量流程图



主要特点：

- 用户配置的模拟 EEPROM 大小
- 增强型闪存的持续时间：页只有在满的时候才会被擦除
- 非易失性数据可以不用经常更新
- 编程或擦除的时候暂停检修时可以的

3 嵌入式应用方面

本节给出了许多建议，关于怎样克服软件限制在嵌入式应用方面并且实现不同应用的这种需要。

3.1 数据间隔尺寸管理

模拟 EEPROM 可以用在一些嵌入式应用当中，这些应用当中要求存储的非易失性数据更新的时候是以一个字节，半字或者一个字间隔尺寸的形式。它通常取决于用户的要求和闪存的体系结构，例如储存数据的长度，写访问等等。

STM32F10xxx 片上闪存允许 16 位，半字编程，数据不管怎样被编程，都可以一个字节或一个字的被编程通过使用一些软件技术。

3.1.1 一个字一个字的编程

闪存驱动提供一个函数，这个函数将 32 位的数据 VarData 写到闪存地址 VarAddress: FLASH_ProgramWord(VarAddress, VarData)。

通过这个函数，一个完整的字可以被写入一个特殊的嵌入式闪存位置。

3.1.2 一个字节一个字节的编程

字节方式写提供给用户存储更多变量数据的可能性。性能方面不关怎样可能会减少。

使用 FLASH_ProgramHalfWord()函数，虚拟地址和数据都可以以一个半字的形式被写入。

3.2 平均抹写存储区块：闪存持续时间的改善

STM32F10xxx 的片上闪存，每一页都可以被可靠地编程或擦除 10000 次左右。

对于集中写超过两页（3 或者 4）的模拟 EEPROM，推荐使用一种平均读写的算法，用来监控和分配这也写循环的次数。

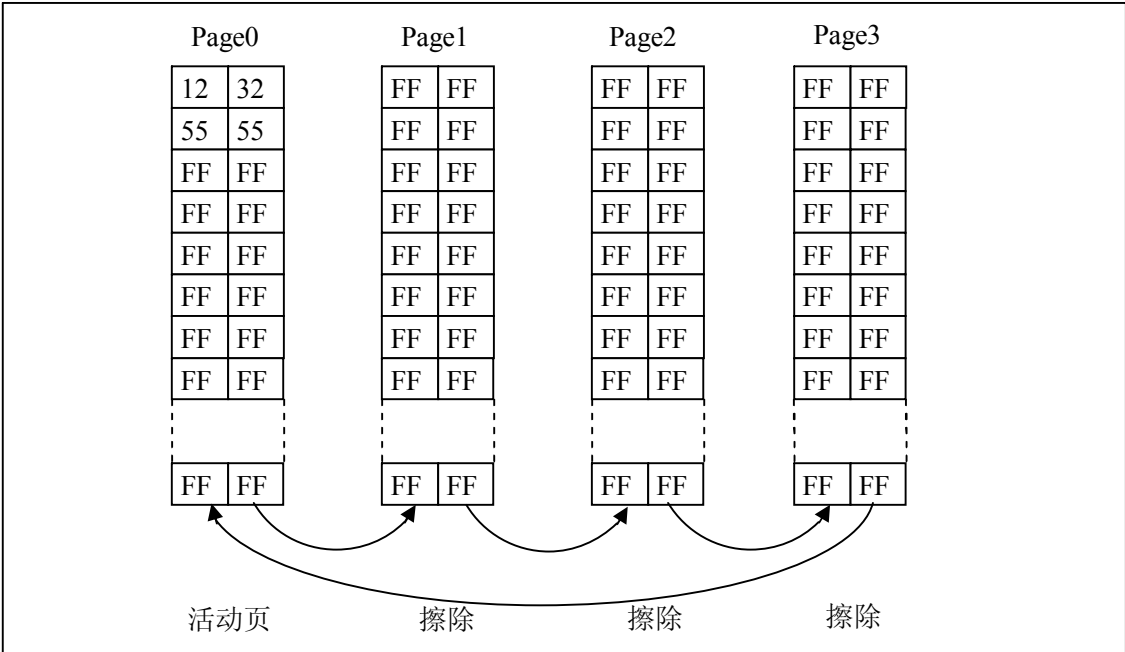
当没有使用平均读写算法时，这也就不能以同样速度被使用。有长时间存储的数据的页不能容忍一样多的写周期，相对于那些包含有频繁更新数据的页。对于每一个扇区而言，平均读写算法保证了相等有效的写周期的实现。

3.2.1 平均抹写存储区块实现的举例

在这个例子当中，为了提高模拟 EEPROM 的容量，有四页将会被使用（Page0, Page1, Page2 和 Page3）。

平均算法像下面这样被实施：当 page n 满的时候，元件转变到 page n+1, Page n 垃圾收集后并被擦除，当轮到 Page3 满的时候，元件返回到 Page0, Page3 垃圾收集后并被擦除并且以此类推下去（参见图 5）

图 5. 四页的页交换计划（平均抹写存储区块）



在软件当中，平均读写算法可以在函数 EE_FindValidPage()中被实现。

3.3 功率损耗下的页数据头恢复

变量更新，页擦除或转移期间功率损耗的话，数据或页数据头讹误可能会发生。

为了检测这种讹误并且恢复它，执行 EE_Init()常规。电源一关掉，这个常规就会立马被调用。这个常规的原则在备注中有记述。这个常规用也状态来检查完整性，如果需要的话并进行修复。

功率损耗之后，EE_Init()常规将会被用来检查页数据头状态。有就中可能的状态组合，其中第三种是无效的。表 2 展示了基于页配置在上电后应该采取的措施。

表 2. 状态组合和采取的行动

Page1	Page0		
	擦除	受到数据	有效页
擦除	无效状态 擦除所有页 并且格式化 page0	擦除 Page1 并且标记 Page0 为有效页	将 page0 当做有效页 使用并擦除 page1
收到数据	擦除 Page0 并且标记 Page1 为有效页	无效状态 擦除所有页并 格式化 page0	将 page0 当做有效页 使用&将 page0 中最 后更新的变量转移到 page1&标记 page1 为 有效页&擦除 page0
有效页	把 page1 当做有效页 使用并且擦除 page0	将 page1 当做有效页 使用&将 page1 中最 后更新的变量转移到 page0&标记 page0 为 有效页&擦除 page1	无效状态 擦除所有页并 格式化 page0

3.4 可擦写性能

一个编程或擦除周期包含一个或更多写访问和一个页擦除操作。
当使用 EEPROM 技术的时候，每一个字节可以被编程或擦除在一个有限的次数内，典型范围是 10000 到 100000。
但是，在嵌入式闪存当中，最小的擦除尺寸是一页，一页的编程或擦除周期次数就是可能擦除周期的次数。STM32F10xxx 的电气特性保证了每页 10000 个编程或擦除周期。因此模拟 EEPROM 的寿命被最经常写的参数的更新速率所限制。

循环的能力与用户想要处理的数据的数量和大小有关。
在这个例子中，两页（1Kb 的中等密度器件或 2Kb 的高密度器件）被使用并且用十六位的数据编程。每一个变量都对应一个 16 位虚拟地址。也就是说，每一个变量占用一个字的存储空间。1Kb（对于中等密度器件）或 2Kb（对于高密度器件）乘以闪存可以忍耐的周期数 10000，模拟闪存一个页面的生命周期内，就有可以存储总共 10000Kb（对于中等密度器件）或 20000Kb（对于高密度器件）的数据的能力。因此，在仿真过程中，模拟 EEPROM 提供的两个页面可以存储 20000Kb（对于中等密度器件）或 40000Kb（对于高密度器件）的数据。如果超过两个页面被使用，这个数量也会相应的增加。当知道所要储存变量数据的宽度的时候，就可以计算出在模拟 EEPROM 的整个生命周期中，总共能够存储的变量的数量。

通过变量的虚拟地址和数据大小，表 3 给出了一个关于模拟 EEPROM 可以存储变量的数量的想法。

表 3.模拟 EEPROM 中储存变量的最大值（10000 个周期）

变量大小	2 x 1Kb	2 x 2Kb
8 位变量（8 位虚拟地址）	10 000 x ($2^{10} - 2$)	10 000 x ($2^{11} - 2$)
16 位变量（16 位虚拟地址）	5000 x ($2^{10} - 4$)	5000 x ($2^{11} - 4$)
32 位变量（32 位虚拟地址）	2500 x ($2^{10} - 8$)	2500 x ($2^{11} - 8$)

1. 最大的变量数目不包括他们相对应的虚拟地址。
2. 两个页面可以写入变量的最大数目减去这个值的数值对应着页面顶部的页状态，取决于变量的粒度，去保护队列，一些空字节会被加进页状态里。这个字节也从最大数里面减去。

4.出版历史

表 4.文档修正历史

日期	版次	变化
2007 年 10 月 5 日	1	首次发行
2008 年 6 月 24 日	2	文档更新后也使用与高密度的 STM32F10xxx 微控制器。 小文本的改变。 写/擦除周期添加到了表 1：内嵌式闪存和 EEPROM 之间的区别。 EE_Init()函数在第 9 也被添加到用户自定义下。 第十页修改了图 4.写变量流程图。在第十页的主要功能也更新了。 更新了 3.3 节：功率损耗下的页数据头恢复。 更新了表 2.状态组合和采取的行动。 更新了表 3. 模拟 EEPROM 中储存变量的最大值（10000 个周期），增加了备注。