life.augmented

AN4061
应用笔记

STM32F0xx 微控制器中的 EEPROM 仿真

简介

在工业应用中经常使用 EEPROM（电可擦除可编程只读存储器）来存储可更新的数据。EEPROM 是用在复杂系统（例如计算机）和其它电子器件中的一种永久（非易失）存储器存储系统，它可以在电源故障时存储和保留少量数据。

为了降低成本，可以使用特定的软件算法用片上 Flash 替代外部 EEPROM。

此应用笔记将介绍使用 STM32F0xx 器件的片上 Flash 通过仿真 EEPROM 机制来取代独立 EEPROM 的软件解决方案。

此仿真至少要使用 Flash 中的两个扇区。EEPROM 仿真代码会在页面填满后在页面之间交换数据，而且此过程对用户是透明的。

此应用笔记随附的 EEPROM 仿真驱动程序满足以下要求：

- 提供简单 API 的轻量级实现，这种 API 由初始化、读写数据和降低存储器占用量三种功能构成。
- 简单且可轻松更新的代码模型。
- 对用户透明的清除和内部数据管理。
- 后台页擦除。
- 至少需要使用两个 Flash 页，如果需要进行耗损均衡，则需要更多。

目录

表格索引

图片索引

1 外部 EEPROM 与仿真 EEPROM 之间的主要差异

EEPROM 是许多需要非易失性数据存储的嵌入式应用的关键组件，它在运行期间以字节或字为粒度进行更新。

这些系统中使用的微控制器通常是基于嵌入式 Flash 存储器的。为了避免使用这些组件、节约 PCB 空间并降低系统成本，可使用 STM32F0xx Flash 代替外部 EEPOM,模拟代码和数据的存储。

但是与 Flash 不同的是，外部 EEPROM 在重写数据之前并不需要执行擦除操作来释放空间。要将数据存储到嵌入式 Flash 中，需要执行特殊的软件管理。

这种仿真软件机制由许多因素决定，包括 EEPROM 可靠性、所使用的 Flash 的架构以及产品要求等。

对于使用相同 Flash 技术的任何微控制器（不局限于 STM32F0xx 系列产品），嵌入式 Flash 和外部串行 EEPROM 之间的主要差异完全一致。*表 1* 中汇总了这些主要差异。

表 1. 外部 EEPROM 与仿真 EEPROM 之间的差异

特性	外部 EEPROM （例如，M24C64： I²C 串行访问 EEPROM）	仿真 EEPROM （使用片上 Flash）
写时间	– 5 ms 内的随机字节写。 字编程时间 = 20 ms – 5 ms 内的页（32 字节）写入。 字编程时间 = 625 µs	半字编程时间：124 µs 到 26 ms (1)
擦除时间	N/A	页擦除时间：20 ms 到 40 ms (2)
写方法	– 启动之后即与 CPU 无关 – 只需要正确供电	启动之后即与 CPU 相关。 如果写操作因软件重置而中断，EEPROM 仿真算法会停止，但是当前的 Flash 写操作不会因软件重置而中断。 可作为半字（16 位）或全字（32 位）访问。
读访问	– 串行：100 µs – 随机字：92 µs – 页：每字节 22.5 µs	并行：（48 MHz 时）半字访问时间为 3.8 µs 到 110 µs (2)
写/擦除循环次数	100 万次写循环	每页 1 万次循环。使用多个片上 Flash 页等同于增加写循环次数。请参见*第 3.4 节：循环性能和页分配*。

1. 有关更多详细信息，请参见*第 2.5 章：EEPROM 仿真时间*。
2. 有关更多详细信息，请参见 STM32F051xx 数据手册中的“内存特性”。

1.1 写访问时间上的差异

由于 Flash 的写访问时间较短，所以对于一些关键参数，在仿真 EEPROM 中的存储速度要比在外部串行 EEPROM 中更快，从而可以改善数据存储。

1.2 擦除时间上的差异

擦除时间方面的差异是独立 EEPROM 与使用嵌入式 Flash 的仿真 EEPROM 之间的另一个重大差异。与 Flash 不同，EEPROM 在写之前不需要执行擦除操作来释放空间。这就意味着必须执行某种形式的软件管理，才能将数据存储到 Flash 中。此外，由于 Flash 中的块擦除过程不需要太长时间，所以在设计 Flash 管理软件时，应注意考虑可能会中断擦除过程的电源关闭和其它一些意外事件（例如复位）。要设计强大的 Flash 内存管理软件，必须透彻了解 Flash 擦除过程。

注： *即使软件重置，也不会中断正在对 STM32F0xx 嵌入式 Flash 执行的页擦除或批量擦除操作。*

1.3 写方法上的相似之处

外部 EEPROM 与具有 STM32F0xx 嵌入式 Flash 的仿真 EEPROM 之间的一个相似之处是写方法。

- 独立外部 EEPROM：在被 CPU 启动之后，字写入不能被软件复位中断。只有供电故障才会中断写过程，因此正确设置去耦电容的大小可以保护独立 EEPROM 中的整个写过程。
- 使用嵌入式 Flash 仿真的 EEPROM：由 CPU 启动后，写过程可由电源故障中断。即使软件复位，也不会中断正在对 STM32F0xx 嵌入式 Flash 执行的字写入操作。EEPROM 算法会停止，但是当前的 Flash 字写入操作不会因软件复位而中断。

2 实现 EEPROM 仿真

2.1 原理

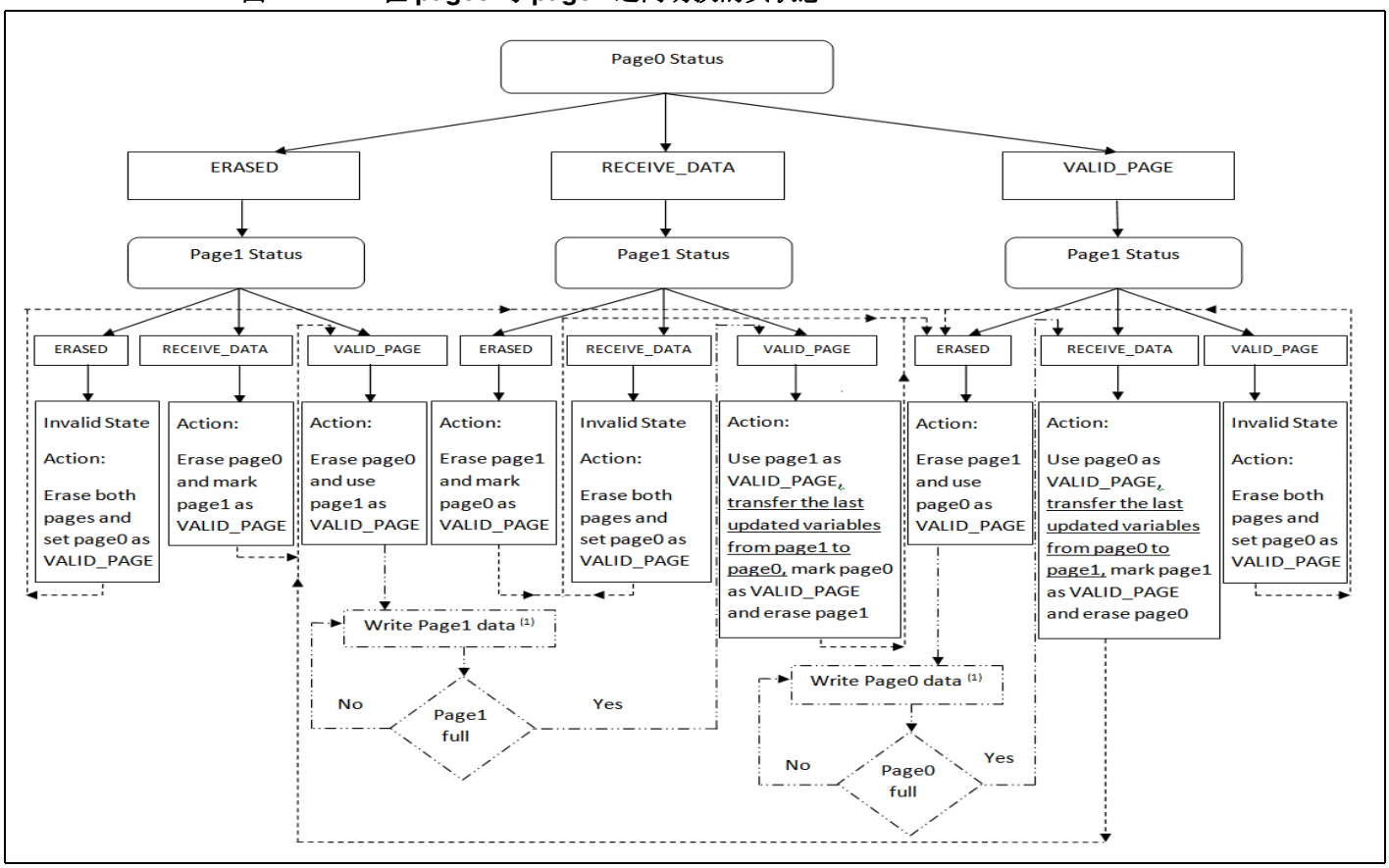
EEPROM 仿真可以通过多种方式实现，但要注意 Flash 限制和产品要求。下面所详述的方法要求为非易失性数据分配至少两个相同大小的 Flash 页：一个在开始时擦除，另一个在需要对前一页执行垃圾回收时接管工作。占用每页前半个字（16 位）的头字段指示页的状态。在本文档的其余部分，这些页分别被称为 Page0 和 Page1。

每个页都有三个可能的状态：

- **ERASED**：页为空。
- **RECEIVE_DATA**：页正在从另一个满页接收数据。
- **VALID_PAGE**：页中包含有效数据，并且在将所有有效数据完全传输到已擦除页之前，此状态不会改变。

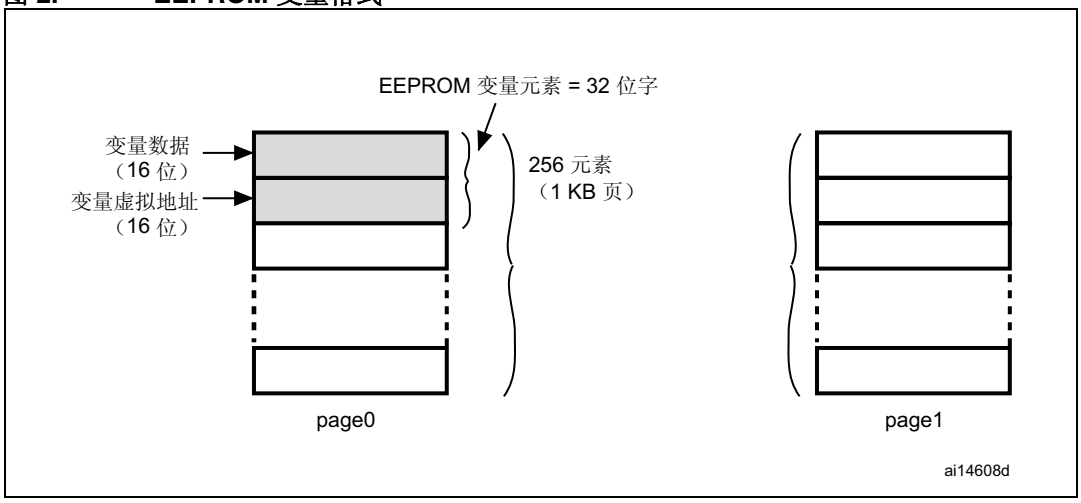
图 1 显示页的状态如何改变。

图 1. 在 page0 与 page1 之间切换的头状态



每个变量元素都由一个虚拟地址和值来定义，它们将被存储在 Flash 中，用于执行后续检索或更新（在实施的软件中，虚拟地址和数据的长度均为 16 位）。如果修改了数据，与之前虚拟地址相关联的已修改数据将会存储到新的 Flash 位置。数据检索会返回最新的数据值。

图 2. EEPROM 变量格式

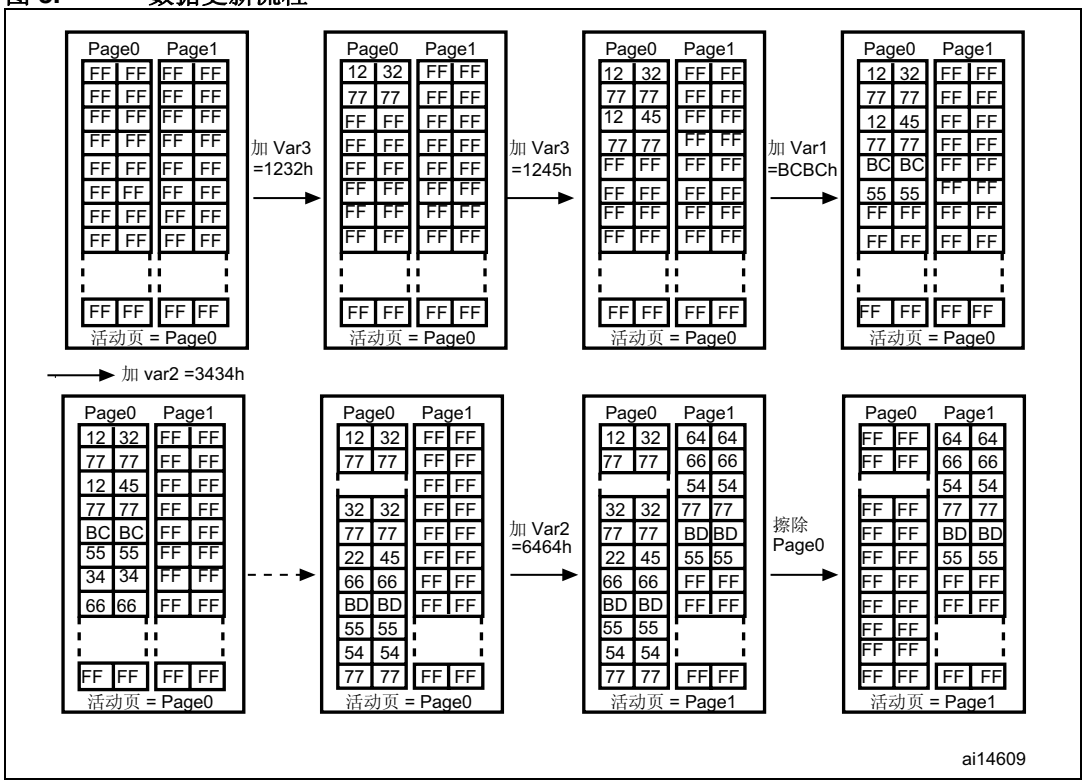


2.2 使用实例：应用示例

以下示例显示的是对具有以下虚拟地址的三个 EEPROM 变量（Var1、Var2 和 Var3）的软件管理：

Var1 的虚拟地址	5555h
Var2 虚拟地址	6666h
Var3 虚拟地址	7777h

图 3. 数据更新流程



2.3 EEPROM 仿真软件说明

本节介绍使用 STMicroelectronics 提供的 STM32F0xx Flash 驱动程序针对 EEPROM 仿真而执行的驱动程序。

此外还提供了一个样本演示程序，用于使用三个变量 (Var1、Var2 和 Var3) 来演示和测试 EEPROM 仿真驱动程序，而这三个变量是在软件的 *main.c* 文件所声明的 *VirtAddVarTab[]* 表中定义的。

此项目除了 Flash 库源文件之外，还包含三个源文件：

- ***eeprom.c***：其中包含 EEPROM 仿真固件函数：

```
EE_Init()
EE_Format()
EE_FindValidPage()
EE_VerifyPageFullWriteVariable()
EE_ReadVariable()
EE_PageTransfer()
EE_WriteVariable()
```

- ***eeprom.h***：其中包含函数原型和一些声明。可使用这个文件来改编下列参数，以满足应用要求：
  - 要使用的数据变量的个数（默认值：3）
- ***main.c***：这个应用程序是使用所描述的例程对 EEPROM 执行读写操作的示例。

用户 API 定义

下表中描述了 *eeprom.c* 文件中包含的函数集，这些函数将用于 EEPROM 仿真：

表 2. API 定义

函数名称	说明
EE_Init()	如果在数据更新或页擦除/传输期间断电，页头可能会损坏。在这种情况下，EE_Init() 函数会尝试将仿真 EEPROM 恢复到某个已知的正常状态。每次断电之后，在访问仿真 EEPROM 之前，都应调用此函数。它不接受任何参数。
EE_Format()	此函数用于擦除 page0 和 page1，并将 VALID_PAGE 头写入到 page0。
EE_FindValidPage()	此函数对两个页头都会进行读取，并返回有效的页编号。传递的参数将指示有效页是在试图执行写入还是读取操作（READ_FROM_VALID_PAGE 或 WRITE_IN_VALID_PAGE）。
EE_VerifyPageFullWriteVariable()	用于实施必须更新或创建变量的首个实例的写过程。这包括在活动页上从结尾开始查找第一个空白位置，然后使用传递的虚拟地址和变量数据填充它。如果活动页已满，将返回值 PAGE_FULL。此例程使用以下参数： 虚拟地址：可以是三个声明变量的虚拟地址中的任何一个（Var1、Var2 或 Var3） 数据：要存储的变量的值 如果成功，此函数将返回 FLASH_COMPLETE，如果变量更新所需的内存不足，则返回 PAGE_FULL，或者返回 Flash 错误码来指示操作故障（擦除或编程）。
EE_ReadVariable()	此函数返回与作为参数传递的虚拟地址所对应的数据。仅读取最后一次更新结果。此函数将进入一个循环，从中循环读取变量项，直到最后一个为止。如果找不到变量的任何具体值，将返回 ReadStatus 变量，且变量值为“1”，否则会将其重置，以指示找到了该变量且变量值在 Read_data 变量中返回。
EE_PageTransfer()	它会将所有变量（具有关联的虚拟地址的数据）的最新值从当前页传输到新的活动页。在开始时，它会确定活动页，也就是要从中传输数据的页。定义和写入新的页头字段（如果正在接收数据，则新页的状态为 RECEIVE_DATA）。如果数据传输已完成，新页头的状态将为 VALID_PAGE，旧页将被擦除，其标头状态变为 ERASED。
EE_WriteVariable()	此函数供用户应用程序调用来更新变量。它使用前面介绍过的 EE_VerifyPageFullWriteVariable() 和 EE_PageTransfer() 例程。

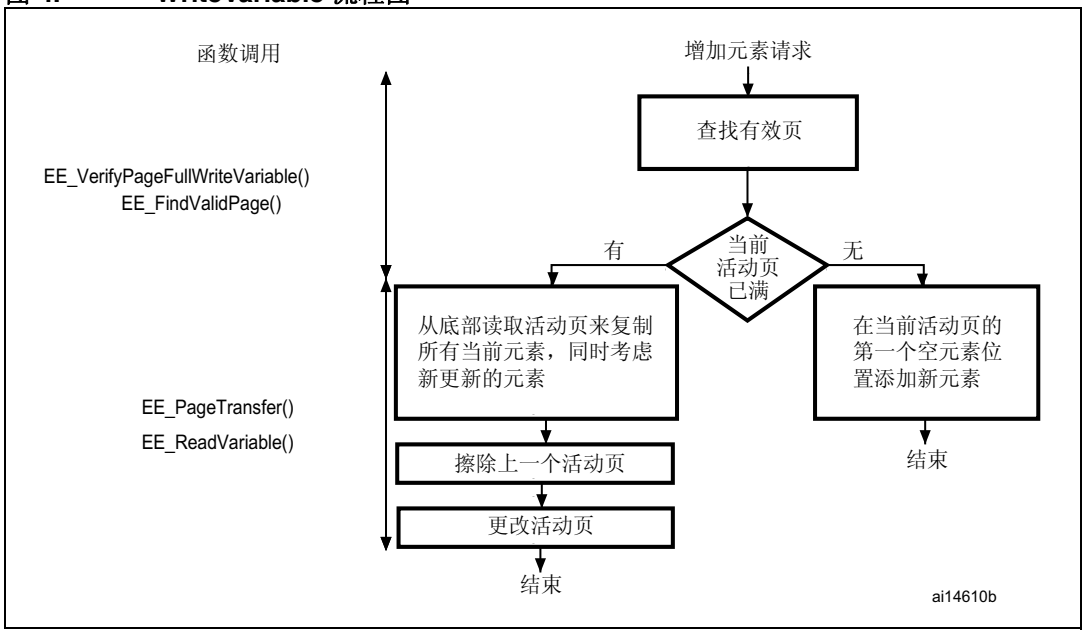
注： *下列函数可用于访问仿真 EEPROM：*

- *EE_Init()*
- *EE_ReadVariable()*
- *EE_WriteVariable()*

这些函数用在此应用笔记所随附的应用程序代码中。

图 4 显示了 EEPROM 中变量项的更新过程。

图 4. WriteVariable 流程图



关键特性

- 用户可配置的仿真 EEPROM 大小
- 增加了 Flash 的可擦写次数：页只有在已满时才会被擦除
- 可以减少非易失性数据变量的更新次数
- 可以在执行编程/擦除期间处理中断

2.4 EEPROM 仿真内存占用量

表 3 通过 Flash 大小和 RAM 大小详细说明了 EEPROM 仿真驱动程序的内存占用量。

下面的表和图已使用 IAR EWARM v6.30.7 工具的 High Size 优化级别进行了确认。

表 3. EEPROM 仿真机制的内存占用量

机制	所需的最低(1)代码大小（字节）	
	Flash	SRAM
EEPROM 仿真软件机制	1824	1046

1. 基于三个 32 位变量（16 位用于地址，16 位用于数据）。所用的 SRAM 内存会根据所用的变量数而增加。

2.5 EEPROM 仿真时间

本节将基于两个 16 KB 的 EEPROM 页大小来介绍与 EEPROM 仿真驱动程序关联的时间参数。

所有时间测量的执行条件：

- STM32F051RBT6
- 系统时钟为 48 MHz，并已使能 Flash 预取和高速缓存功能
- 从 Flash 执行代码
- 在室温下

表 4 列出了 EEPROM 的时值。

表 4. 系统时钟为 48 MHz 的 EEPROM 仿真时间

操作	EEPROM 仿真时间		
	最小值 (µs)	典型值 (ms)	最大值 (µs)
EEPROM 中的典型(1)变量(2)写操作	124	-	219
EEPROM 中具有页交换(3)的变量写操作	-	26	-
EEPROM(4) 的变量写操作	3.8	-	110
EEPROM 的第一次初始化(5)	-	52.11	-
典型 EEPROM 初始化(6)	-	26	-

1. 写入但没有页交换。最小值是指在 Flash 页的开始部分某个变量的写操作，而最大值是指在 Flash 页的结束部分的写操作。最小值与最大值之间的差异是由于查找空闲 Flash 地址来存储新数据所花费的时间不同所致。
2. 所用变量大小为 32 位（16 位用于虚拟地址，16 位用于数据）。
3. 当有效页变满时页交换完成。这包括将各个变量最后存储的数据传输到另一个空闲页并擦除满页。
4. 最小值是指 Flash 页中存储的第一个变量的读操作，而最大值是指最后一个变量 -1 的读操作。最小值与最大值之间的差值是查找最后存储的变量数据所花费的时间。
5. 当 EEPROM 机制第一次运行或当出现无效状态时（请参见 *表 1：在 page0 与 page1 之间切换的头状态* 获取详细信息），将会擦除这两页，并将用于存储的页标记为 VALID_PAGE。
6. 典型 EEPROM 初始化在存在有效页（即 EEPROM 至少已初始化一次）时执行。在典型 EEPROM 初始化期间，将擦除两页中的一页（请参见 *表 1：在 page0 与 page1 之间切换的头状态* 获取详细信息）。

3 嵌入式应用的相关信息

本节提供有关如何克服嵌入式应用中的软件限制以及如何满足不同应用需求的建议。

3.1 数据粒度管理

如果嵌入式应用需要使用半字或字粒度进行更新的非易失性数据存储，则可在其中使用仿真 EEPROM。这通常由用户要求和 Flash 架构（例如存储的数据长度、写访问等）来决定。

STM32F0xx 片上 Flash 允许 16 位或字编程：

表 5. Flash 编程函数

数据粒度	函数名称
按字（32 位）	FLASH_ProgramWord
按半字（16 位）	FLASH_ProgramHalfWord

3.2 耗损均衡：增加 Flash 可擦写次数

在 STM32F0xx 片上 Flash 中，每页能够可靠编程或擦除的次数大约为 1 万次。

对于在仿真 EEPROM 中使用两页以上的写密集型应用，建议实施耗损均衡算法来监控和分配页之间的写循环次数。

如果不使用耗损均衡算法，则不能以相同的比率使用这些页。对于存放长期数据的页，其写循环次数要小于存放经常更新的数据的页。耗损均衡算法可以确保每个页的所有可用写循环都有相等的使用机会。

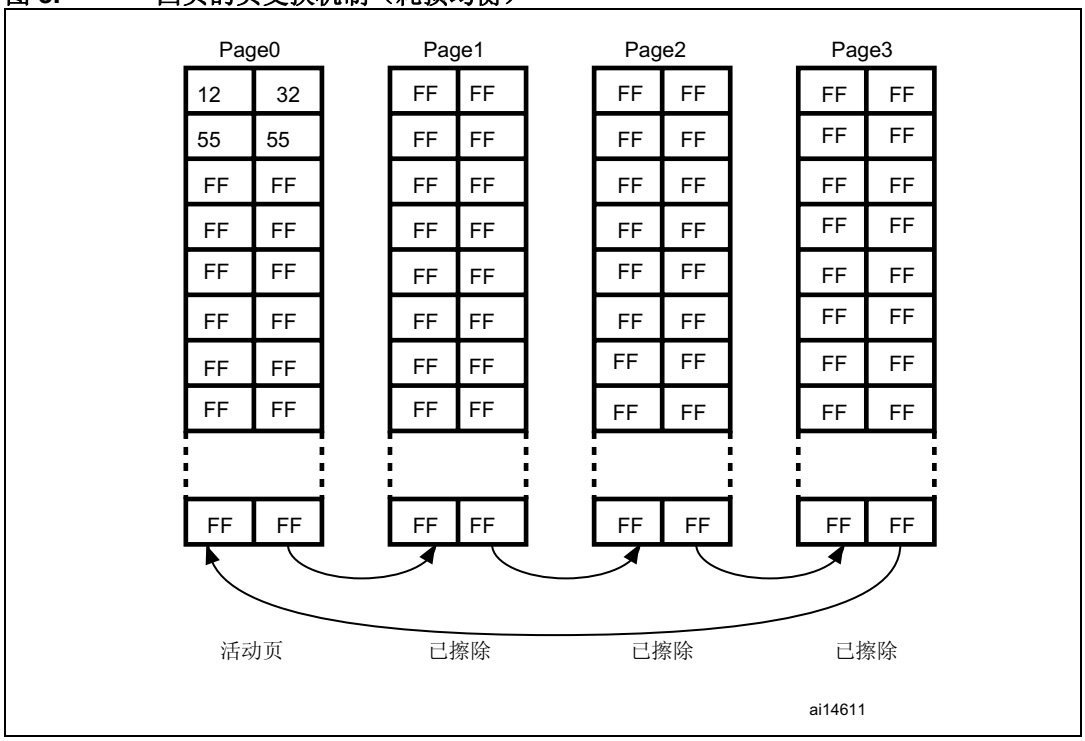
注： *STM32F0xx Flash 中的主内存块将按照表 3：EEPROM 仿真机制的内存占用量中的说明进行分割。*

3.2.1 耗损均衡实现示例

在本示例中，为了增强仿真 EEPROM 的功能，将使用四个页（Page0、Page1、Page2 和 Page3）。

将按照下面的方法实施耗损均衡算法：如果页 *n* 已满，器件将切换到页 *n+1*。对页 *n* 执行垃圾回收，然后执行擦除。当轮到 Page3 变满时，器件将返回 Page0，并对 Page3 执行垃圾回收和擦除，以此类推（请参见*图 5*）。

图 5. 四页的页交换机制（耗损均衡）



在软件中，可使用 EE_FindValidPage() 函数来实施耗损均衡算法（请参见*表 2*）。

3.3 断电时的页头恢复

在执行变量更新、页擦除或传输期间如果断电，可能会损坏数据或页头。

要检测此类损坏并加以恢复，可执行 EE_Init() 例程。应在上电之后立即调用它。在本应用笔记中将介绍此例程的原理。此例程使用页状态检查完整性，并在必要时执行修复。

在断电之后，将使用 EE_Init() 例程检查页头状态。共有九种可能的状态组合，其中三种无效。*图 1：在 page0 与 page1 之间切换的头状态*显示了上电后应根据页状态采取的操作。

3.4 循环性能和页分配

3.4.1 循环性能

编程/擦除循环由一个或多个写访问和一个页擦除操作构成。

如果使用 EEPROM 技术，则每个字节的编程和擦除次数会受到限制，范围通常为 1 万到 10 万次。

但是在嵌入式 Flash 中，最小擦除大小为页，而应用到页的编程/擦除循环数即为可执行的擦除循环数。STM32F0xx 的电气特性保证了可对每页执行 1 万次编程/擦除循环。因此，仿真 EEPROM 的最大使用寿命受最频繁写入参数的更新频率限制。

循环性能取决于用户希望处理的数据量/大小。在本示例中，使用了两页（大小为 1 KB），并通过 16 位数据进行了编程。每个变量都与一个 16 位虚拟地址对应。即每个变量占用一个字的存储空间。一页可存储的字节数是 1 KB 乘以 Flash 的耐久性 1 万次循环，前提是仿真 EEPROM 内存中一页的生存期的数据存储总容量为 10000 KB。因此，如果在仿真过程中使用两页，则仿真 EEPROM 中可存储 20000 KB。如果使用的页超过两个，则该数字会相应倍增。

由于了解了所存储变量的数据宽度，所以可以计算仿真 EEPROM 区在其使用寿命内可存储的变量总数。

3.4.2 Flash 页分配

EEPROM 仿真应用所需的页大小和页数量可根据系统生存期内写入的数据量进行选择。

例如，在系统生存期内，可使用具有 10 KB 擦除循环的 1 KB 页来最多写入 10 MB 的数据。

可按照以下方法计算可用变量空间：

FreeVarSpace = (PageSize) / (VariableTotalSize) - [NbVar+1]

其中：

- **PageSize**：以字节为单位的页大小（例如 1 KB）
- **NbVar**：正在使用的变量数（例如 10 个变量）
- **VariableTotalSize**：用于存储变量（地址和数据）的字节数
  - VariableTotalSize = 8，用于 32 位变量（32 位数据 + 32 位虚拟地址）
  - VariableTotalSize = 4，用于 16 位变量（16 位数据 + 16 位虚拟地址）

通过以下方法可以计算出应用生存期内保证可预测 Flash 操作所需的页数估算值：

所需页数：

NbPages = NbWrites / (PageEraseCycles * FreeVarSpace)

其中：

- **NbPages** = 所需页数
- **NbWrites** = 变量总写入数
- **PageEraseCycles** = 页的擦除循环次数

注： *如果变量是 16 位，则每个变量都占用 32 位（16 位数据加 16 位虚拟地址），这意味着每次写入新数据时，各个变量分别使用 4 字节的 Flash。每个 1 KB 页在变满之前可执行 256 次变量写入。*

注： *这个计算结果是较为保守的估算值。*

使用实例示例

设计一个更新 **20** 个不同变量的应用，每 **2** 分钟更新一次，共更新 **10** 年：

- NbVar = **20**
- NbWrites = **10** * 365 * 24 * (60/**2**) * NbVar = ~5200 万次写入

- 如果变量包含 16 位虚拟地址和 16 位数据，则保证所有这些数据的非易失性存储所需的页数为：
  - 1 KB 大小的 21 页
- 如果变量包含 32 位虚拟地址和 32 位数据，则保证所有这些数据的非易失性存储所需的页数为：
  - 1 KB 大小的 41 页

表 6. 应用设计

变量大小	写入数 (NbWrites)	要写入的数据总量（以字节为单位）	页大小 (KB)	页擦除循环次数	所需页数	所用页数
16 位	52 560 000	210 240 000	1	10000	20.5	21
32 位	52 560 000	420 480 000	1	10000	41	41

3.5 实时注意事项

所提供的 EEPROM 仿真固件要从内部 Flash 运行，所以如果执行的操作需要 Flash 擦除或编程（EEPROM 初始化、变量更新或页擦除），则对 Flash 的访问将会终止。应用代码也会因此不再执行，对中断也不再进行处理。

许多应用可以接受此行为，但是对于具有实时约束的应用，用户需要从内部 RAM 运行关键进程。

在这种情况下：

1. 在内部 RAM 中重定位向量表。
2. 从内部 RAM 执行所有关键代码和中断服务例程。编译器将提供一个关键字，将函数声明为 RAM 函数；在系统启动时，函数将从 Flash 复制到 RAM，就像任何初始化的变量一样。请务必注意以下内容：对于 RAM 函数，所有使用的变量和调用的函数都应位于 RAM 内。

4 版本历史

表 7. 文档版本历史

日期	版本	变更
2012 年 05 月 11 日	1	初始版本。

请仔细阅读：

中文翻译仅为方便阅读之目的。该翻译也许不是对本文档最新版本的翻译，如有任何不同，以最新版本的英文原版文档为准。

本文档中信息的提供仅与ST产品有关。意法半导体公司及其子公司（“ST”）保留随时对本文档及本文所述产品与服务进行变更、更正、修改或改进的权利，恕不另行通知。

所有ST产品均根据ST的销售条款出售。

买方自行负责对本文所述ST产品和服务的选择和使用，ST概不承担与选择或使用本文所述ST产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为 ST 授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在ST的销售条款中另有说明，否则，ST对ST产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

意法半导体的产品不得应用于武器。此外，意法半导体产品也不是为下列用途而设计并不得应用于下列用途：（A）对安全性有特别要求的应用，例如，生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）汽车应用或汽车环境，且/或（D）航天应用或航天环境。如果意法半导体产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向意法半导体发出了书面通知，采购商仍将独自承担因此而导致的任何风险，意法半导体的产品设计规格明确指定的汽车、汽车安全或医疗工业领域专用产品除外。根据相关政府主管部门的规定，ESCC、QML或JAN正式认证产品适用于航天应用。

经销的ST产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致ST针对本文所述ST产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大ST的任何责任。

ST和ST徽标是ST在各个国家或地区的商标或注册商标。

本文档中的信息取代之前提供的所有信息。

ST徽标是意法半导体公司的注册商标。其他所有名称是其各自所有者的财产。

意法半导体集团公司

澳大利亚 - 比利时 - 巴西 - 加拿大 - 中国 - 捷克共和国 - 芬兰 - 法国 - 德国 - 中国香港 - 印度 - 以色列 - 意大利 - 日本 - 马来西亚 - 马耳他 - 摩洛哥 – 菲律宾 – 新加坡 - 西班牙 - 瑞典 - 瑞士 - 英国 - 美国

www.st.com