AN3390
应用笔记

STM32F2xx 微控制器中的 EEPROM 模拟

前言

EEPROM （电可擦除可编程只读存储器）通常用于工业应用领域，用于存储可更新数据。EEPROM 是一种用于复杂系统 （如计算机）和其他电子设备的永久性 （非易失的）存储系统，用于在断电时存储和保留少量数据。

为了降低成本，外部 EEPROM 可以用 STM32F2xx 产品所具有的下述特性之一代替：

- 片上 4 KB 备份 SRAM
- 片上 Flash，并采用专门软件算法

STM32F2xx 具有 4 KB 的备份 SRAM，当主要的 VDD 电源断电时，该 SRAM 可由 VBAT 电源供电。

当有 VBAT （通常在电池供电应用中）时该备份 SRAM 可作为内部 EEPROM （无任何额外软件）使用，并具有以 CPU 频率进行高速存取的优势。

而当备份 SRAM 另作他用且 / 或该应用不用 VBAT 供电时，片上 Flash（采用专门软件算法）可以用于模拟 EEPROM 存储器使用。

本应用笔记对通过利用 STM32F2xx 产品的片上 Flash 模拟 EEPROM 机制来代替独立 EEPROM 的软件方案进行了说明。

至少使用两个 Flash 扇区才能实现该模拟。 EEPROM 模拟代码在两个扇区 （当它们被填充时）之间交换数据，这在某种程度上是对用户透明的。

本应用笔记所提供的 EEPROM 模拟驱动器满足以下要求：

- 轻量级实现，具有一个简单的包含了三个函数 （用于初始化、读数据和写数据）的 API，并减少了封装。
- 简单易用的可更新代码模型
- 清理和内部数据管理对用户透明
- 后台扇区擦除
- 至少要使用两个 Flash 扇区，对于损耗均衡则需要使用更多 （Flash 扇区）

模拟 EEPROM 的大小是弹性可变的，它受扇区大小的限制和约束，并需考虑最大 EEPROM 空间大小。

目录

表格索引

图片索引

1 外部 EEPROM 和模拟 EEPROM 之间的主要区别

EEPROM 是许多嵌入式应用（需要能够进行非易失性数据存储，且运行时间内以字节或字的颗粒度进行更新）的关键元件。

用于这些系统的微控制器通常基于嵌入式 Flash。为了减少所用元件、节省 PCB 空间、降低系统成本，可以用 STM32F2xx Flash 代替外部 EEPROM 来进行同步编码和数据存储。

与 Flash 不同，数据可被重写前，外部 EEPROM 不需要擦除操作来释放空间。需要专门的软件管理来将数据存入嵌入式 Flash。

仿真软件方案取决于多种因素，包括 EEPROM 可靠性、所用 Flash 结构和产品需求。

嵌入式 Flash 和外部串行 EEPROM 之间的主要区别对于任何使用同样 Flash 技术的微控制器（并非针对 STM32F2xx 系列产品）都是相同的。主要区别概括如*表 1*。

表 1. 外部 EEPROM 和模拟 EEPROM 之间的区别

特性	外部 EEPROM （例如， M24C64：I²C 串行存取 EEPROM）	利用片上 Flash 模拟的 EEPROM	利用片上备份 SRAM 内存模拟的 EEPROM (1)
写时间	– 随机字节写操作在 5 ms 之内。全字编程时间 = 20 ms – 页 （32 字节）写操作在 5 ms 之内。全字编程时间 = 625 µs	半字编程时间：28 µs 至 251 ms (2)	0 等待状态下的 CPU 速度
擦除时间	N/A	扇区 （大页）擦除时间：1s 至 3 s （取决于扇区大小）	NA
写方式	– 启动后，不依赖于 CPU – 仅需要适当的供电	启动后，依赖于 CPU 如果一个写操作被 CPU 复位中断， EEPROM 模拟算法就会停止，但是当前的 Flash 写操作不会被软件复位中断。 可按字节 （8 位）、半字 （16 位）或全字 （32 位）访问。	可按字节 （8 位）、半字 （16 位）或全字 （32 位）访问。 写操作被软件复位中断。

表 1. 外部 EEPROM 和模拟 EEPROM 之间的区别 （续）

特性	外部 EEPROM （例如， M24C64：I²C 串行存取 EEPROM）	利用片上 Flash 模拟的 EEPROM	利用片上备份 SRAM 内存模拟的 EEPROM (1)
读访问	– 串行：100 us – 随机字：92 µs – 页：每字节 22.5 µs	并行：（以 120 MHz 速率）半字访问时间为 0.44 µs 至 332 µs(2)	1 等待状态下的 CPU 速度
写 / 擦除周期	100 万次写周期	每扇区 （大页） 10 千周。使用多个片上 Flash 页等效于增加写周期的次数。请参见*第 3.4 章节：可擦写性能和存储页分配*。	有 VBAT 时，无限制

1. 关于备份 SRAM 使用的更多信息，请参考 *STM32F2xx 参考手册*（RM0033）中的 “ 电池备份域 ” 。
2. 有关详细信息，请参见*第 2.5 节：EEPROM 模拟时序*。

1.1 写访问时间的不同

由于 Flash 的写访问时间更短，重要参数可以更快地存入模拟 EEPROM （比外部串行 EEPROM 更快），因此能够提高数据存储能力。

1.2 擦除时间的不同

擦除时间的不同是独立 EEPROM 和使用嵌入式 Flash 模拟的 EEPROM 之间的另一个重要区别。与 Flash 不同，在向其写入数据前， EEPROM 不需要擦除操作来释放空间。这意味着需要某种形式的软件管理来将数据存入 Flash。此外，由于 Flash 中的块擦除过程不需要花费很多时间，因此设计 Flash 管理软件时，应当考虑断电和其他可能中断擦除过程的伪事件 （例如复位）。为了设计出稳健的 Flash 管理软件，有必要彻底了解 Flash 擦除过程。

注： CPU 复位过程中， STM32F2xx 嵌入式 Flash 上正在进行的扇区擦除或批量擦除操作不会被中断。

1.3 写方式的相似之处

外部 EEPROM 和采用 STM32F2xx 嵌入式 Flash 模拟的 EEPROM 之间的相同点之一是其写方式。

- 独立外部 EEPROM：由 CPU 启动时，全字写操作不会被 CPU 复位而中断。只有电源故障会中断该写过程，因此适当地按大小排列去耦电容可以保护独立 EEPROM 中的完整写过程。

- 利用嵌入式 Flash 模拟的 EEPROM：由 CPU 启动时，写过程可被电源故障中断。CPU 复位过程中， STM32F2xx 嵌入式 Flash 上正在进行的全字写操作不会被中断。 EEPROM 算法停止，但是当前的 Flash 全字写操作不会被 CPU 复位中断。

2 实现 EEPROM 模拟

2.1 原理

考虑到 Flash 限制和产品需求，存在多种方式实现 EEPROM 模拟。下面所描述的方式需要至少两个同样大小的 Flash 扇区（分配给非易失性数据）：一个扇区首先被擦除，并提供逐字节的编程；当前扇区需要被回收时，另一个扇区准备好接收（该扇区）。占据了每个扇区第一个半字（16 位）的头字段标志了该扇区状态。将每个扇区作为一个存储页，在本文档下述部分中称作 Page0 和 Page1。

头字段位于每页的基址，提供了该存储页的状态信息。

每页有三个可能状态：

- **ERASED**：该页为空。
- **RECEIVE_DATA**：该页正在从其他满页中接收数据。
- **VALID_PAGE**：该页容纳有效数据，直到所有有效数据传输到被擦除页，这个状态才会变化。

图 1 显示了页状态是如何变化的。

图 1. page0 和 page1 之间的头状态转换

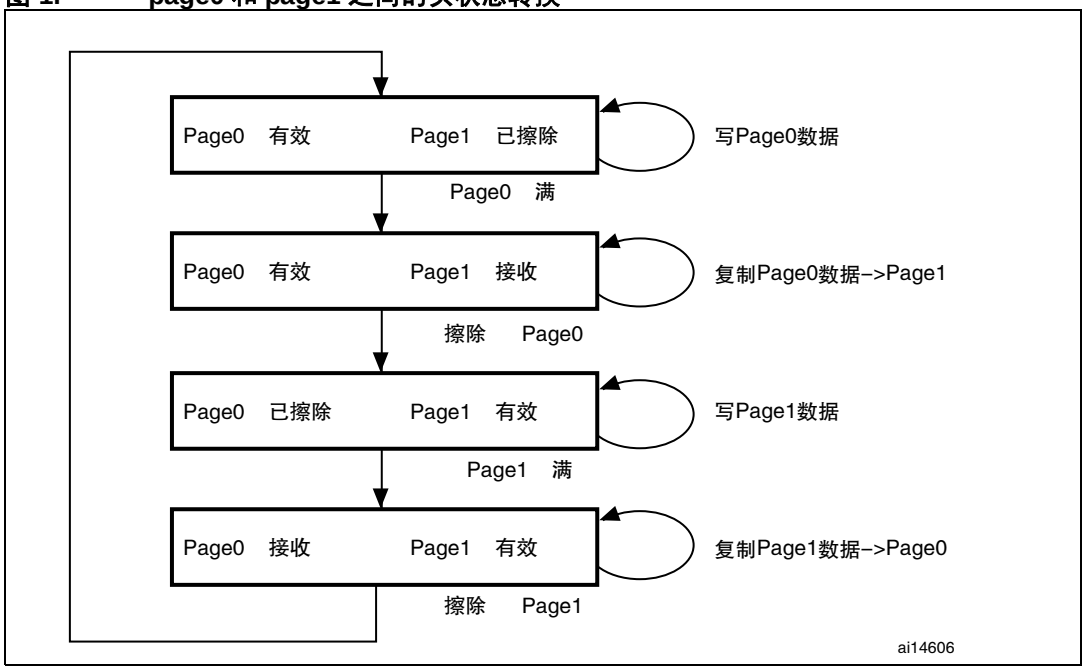


表 2. 模拟页的可能状态和相应行为

Page1	Page0		
	ERASED	RECEIVE_DATA	VALID_PAGE
ERASED	**无效状态** **行为：**擦除两个页并格式化 page0	**行为：**擦除 Page1 并将 Page0 标记为 VALID_PAGE	**行为：**将 page0 作为有效页使用并擦除 page1
RECEIVE_DATA	**行为：**擦除 Page0 并将 Page1 标记为 VALID_PAGE	**无效状态** **行为：**擦除两个页并格式化 page0	**行为：**将 page0 作为有效页使用 & 将最新更新的变量从 page0 传输到 page1 & 将 page1 标记为有效 & 擦除 page0
VALID_PAGE	**行为：**将 page1 作为有效页使用并擦除 page0	**行为：**将 page1 作为有效页使用 & 将最新更新的变量从 page1 传输到 page0 & 将 page0 标记为有效 & 擦除 page1	**无效状态** **行为：**擦除两个页并格式化 page0

通常，使用这种方式时，用户不必提前知道变量更新频率。

本文档中所描述的软件和实现使用了两个 16 KB 的 Flash 扇区（扇区 2 和扇区 3）来模拟 EEPROM。

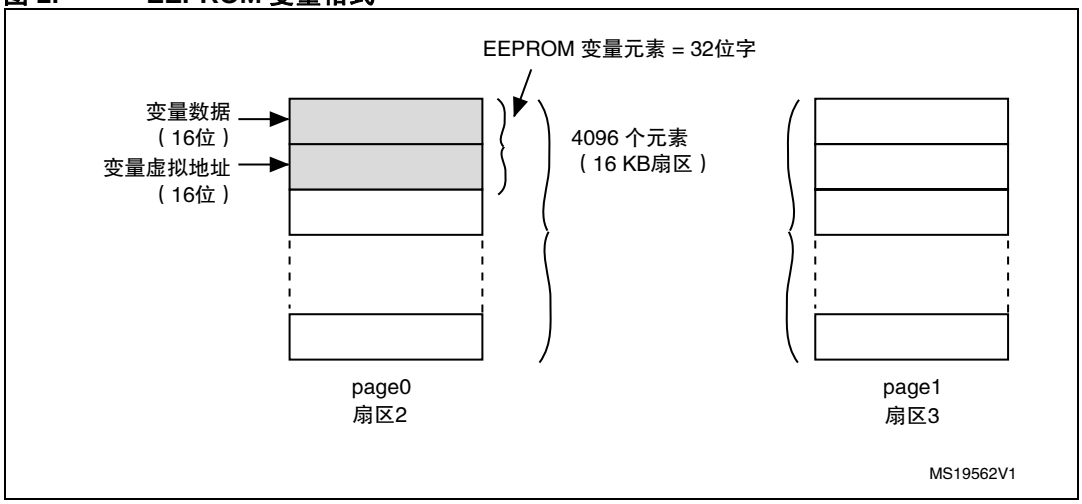
注： *选择扇区 2 和扇区 3 是由于这两个扇区相比于 STM32F2xx Flash 的其他扇区来说空间较小（STM32F2xx Flash 的主要内存块在表 3: STM32F2xx Flash 扇区中有单独描述）。根据应用和用户需要，也可以使用大的扇区。*

表 3. STM32F2xx Flash 扇区

名称	扇区大小
扇区 0-3	16 KB
扇区 4	64 KB
扇区 5-11	128 KB

每个可变参数通过虚拟地址和要被存入 Flash 的值来定义，以便随后检索或更新（在所实现的软件中虚拟地址和数据都是长 16 位）。当数据被修改时，关联到先前虚拟地址的修改数据会被存入新的 Flash 位置。数据检索返回最新的数据值。

图 2. EEPROM 变量格式

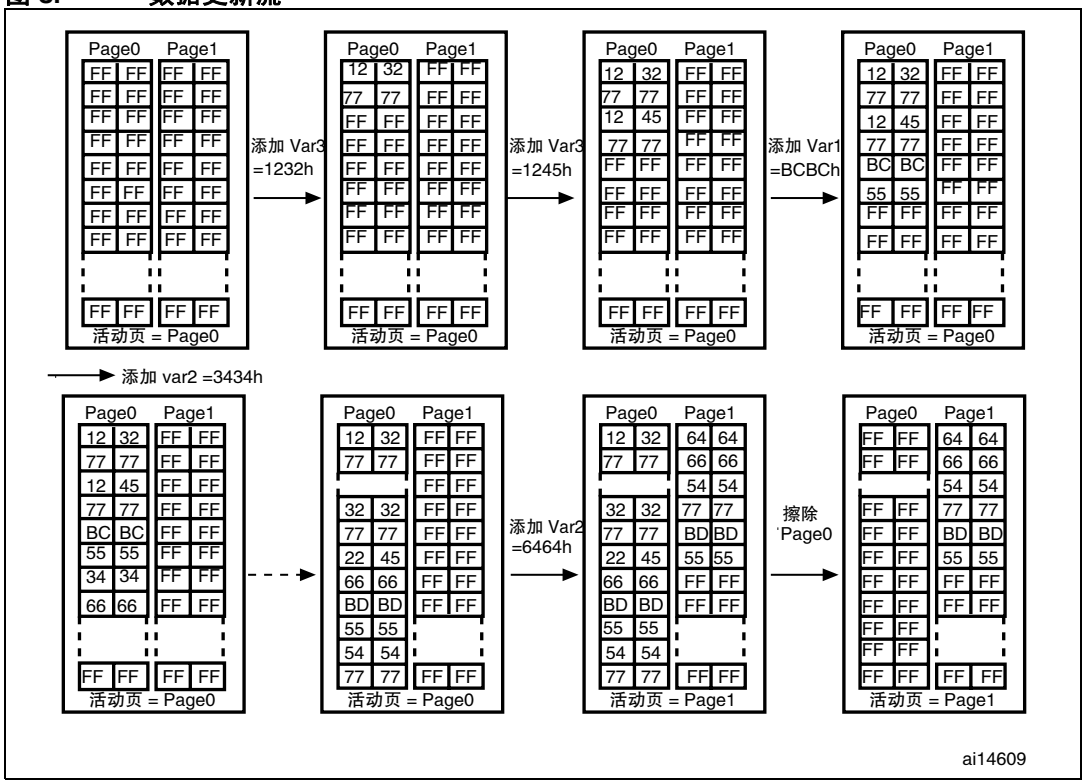


2.2 用例：应用实例

下面的例子显示了三个采用如下虚拟地址的 EEPROM 变量 （Var1、 Var2 和 Var3）的软件管理：

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

图 3. 数据更新流



2.3 EEPROM 模拟软件说明

本节对实现 EEPROM 模拟的驱动（利用意法半导体提供的 STM32F2xx Flash 的驱动）进行了说明。

（本应用笔记）提供了实例演示程序，利用 *VirtAddVarTab[]* 表 （在软件 *main.c* 文件中声明的）中定义的三个变量 Var1、 Var2 和 Var3，来演示和测试 EEPROM 模拟驱动器。

除了 Flash 库源文件之外，项目中还包含了三个源文件：

- ***eeprom.c***：包含了 EEPROM 模拟固件函数：

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

- ***eeprom.h***：包含了函数原型和一些声明。您可以对本文件中的下述参数进行改写，以便适应您的应用需求：
  - 所用 Flash 扇区 （默认：扇区 2 和扇区 3）

– 器件电压范围 （默认：范围 4， 2.7V 至 3.6V）。

EEPROM 模拟固件中， FLASH_ProgramHalfWord() 函数用于为内存编程。该函数仅当器件电压在 2.1V 至 3.6V 范围内时可用。因此，如果您的应用中器件电压范围为 1.8V 至 2.1 V，就必须使用 FLASH_ProgramByte() 函数来替代 （上述函数），并且相应地改写固件以适应此函数。

– 所用的数据变量数目 （默认：3）

- ***main.c***: 该应用程序是一个示例，使用所述程序向 EEPROM 写入或从 EEPROM 读取数据。

用户 API 定义

eeprom.c 文件包含的函数集 （用于进行 EEPROM 模拟）如下表所示：

表 4. API 定义

函数名称	说明
EE_Init()	在数据更新或扇区擦除 / 传输过程中，失去供电可能导致扇区头损坏。这种情况下， EE_Init() 函数会尝试将模拟 EEPROM 恢复到一个已知的良好状态。每次电源关闭后，应当在访问模拟 EEPROM 前调用该函数。它不需要参数。该过程如 *表 2: 模拟页的可能状态和相应行为第 9 页*中所述。
EE_Format()	该函数擦除 page0 和 page1，并向 page0 中写入 VALID_PAGE 头。
EE_FindValidPage()	该函数读取 （两个）页头并返回有效页编号。传递参数指示是否搜索到有效页，以便进行写或读操作 （READ_FROM_VALID_PAGE 或 WRITE_IN_VALID_PAGE）。
EE_VerifyPageFullWriteVariable()	它执行写过程，必须更新或者创建一个变量的第一个实例。它从末尾开始寻找活动页的第一个空位置，并用传递的虚拟地址和变量数据将其填充。当活动页满时，返回 PAGE_FULL 值。这个程序使用下面的参数： 虚拟地址：可能是三个已声明变量的虚拟地址 （Var1、 Var2 或 Var3）中的任意一个 数据：要存储的变量值 该函数成功时将返回 FLASH_COMPLETE，用于变量更新的内存不足时返回 PAGE_FULL，或返回一个 Flash 错误码来指示操作失败 （擦除或编程）。
EE_ReadVariable()	该函数返回一个数据，此数据对应于变量传递的虚拟地址。仅读取最新更新。函数会进入一个循环，在该循环中，它一直读取进入的变量，直到读完最后一个。如果没有发现变量， ReadStatus 变量会返回 “1”，否则 （发现了变量），它将会被重置来指示变量已被找到，找到的变量值将会在 Read_data 变量中返回。

表 4. API 定义 （续）

函数名称	说明
EE_PageTransfer()	它将所有变量的最新值 （关联到虚拟地址的数据）从当前页传输到新的活动页。开始时，该函数首先确定数据将要从中传输出去的活动页。定义并写入新存储页的头字段 （新的页状态为 RECEIVE_DATA，假定它正在接收数据过程中）。当数据传输完成时，新页的头为 VALID_PAGE，原来的存储页被擦除，其头变为 ERASED。
EE_WriteVariable()	用户应用调用该函数来更新变量。它使用了 EE_VerifyPageFullWriteVariable() 和 EE_PageTransfer() 程序 （二者已有描述）。

注： *可用下面的函数来访问模拟 EEPROM：*

- EE_Init()

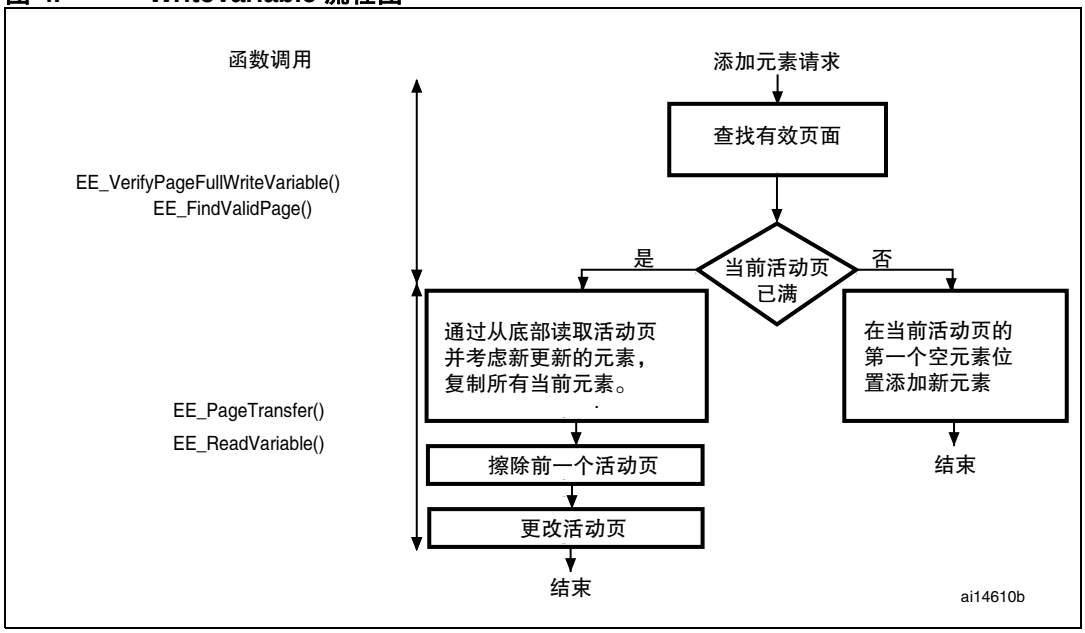
- EE_ReadVariable()

- EE_WriteVariable()

这些函数可用于本应用笔记的应用代码中。

图 4 表示更新 EEPROM 的变量入口的过程。

图 4. WriteVariable 流程图



主要特性：

- 用户配置的模拟 EEPROM 大小
- 增加 Flash 可擦写次数：仅当页满时才进行擦除
- 可以偶尔更新非易失性数据
- 编程 / 擦除过程中可进行中断服务

2.4 EEPROM 模拟内存占用

表 5 详细描述了 EEPROM 模拟驱动器在 Flash 空间和 RAM 空间方面的占用情况。

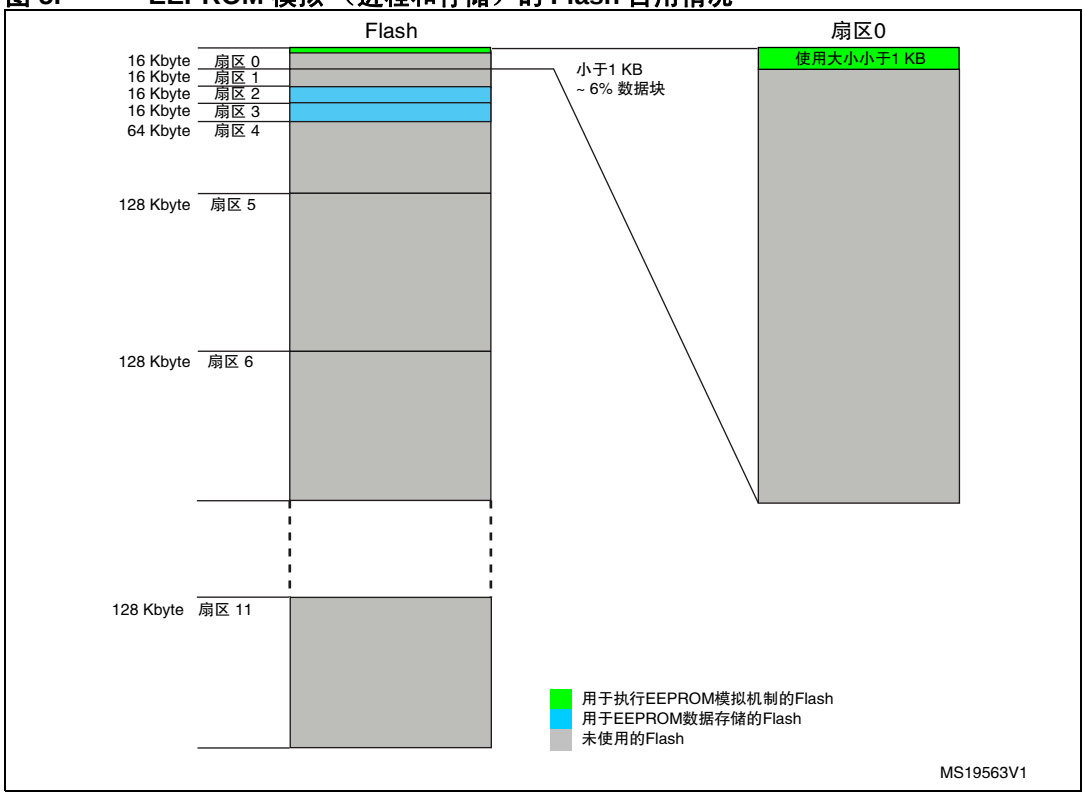
下面的表格和图是在使用大尺寸优化级别的 IAR EWARM 6.10 工具情况下得到的。

表 5. EEPROM 模拟机制的内存占用情况

原理	所需最小(1) 代码空间 （字节）	
	闪存	SRAM
EEPROM 模拟软件原理	984	6

1. 基于 32 位变量 （地址占 16 位，数据占 16 位）。所用 SRAM 内存随着所使用的变量数量 （增加）而增加。

图 5. EEPROM 模拟 （进程和存储）的 Flash 占用情况



2.5 EEPROM 模拟时序

本节对两个 16 KB EEPROM 页空间上的 EEPROM 模拟驱动器的时序参数进行说明。

进行所有的时序测量：

- STM32F207IGH6 版本 Y
- 电压范围 3 （2.7 V 至 3.6 V）
- 120 MHz 的系统时钟，启用 Flash 预取和高速缓存功能
- 由 Flash 执行
- 室温下

表 6 列出了 EEPROM 的时序值。

表 6. 采用 120 MHz 系统时钟的 EEPROM 模拟时序

操作	EEPROM 模拟时序		
	最小值（µs）	典型值（ms）	最大值（µs）
EEPROM 中典型的(1) 变量(2) 写操作	28	-	330
EEPROM 中存在页交换(3) 的变量写操作	-	251	-
由 EEPROM 进行的变量读操作 (4)	0.44	-	332
EEPROM 首次初始化 (5)	-	510	-
典型 EEPROM 初始化(6)	-	241	-

1. 写数据，无页交换。最小值指在 Flash 扇区开始处进行的变量写操作，最大值指在 Flash 扇区末尾处进行的写操作。最大值和最小值之间的差别来自于寻找空的 Flash 地址存储新数据所花费的时间。
2. 所用变量空间为 32 位 （虚拟地址占 16 位，数据占 16 位）
3. 当有效页满时，进行页交换。该过程包括将每个变量最后存储的数据传输到另一个空白页，并擦除原来的满页。
4. 最小值指对存储在 Flash 扇区中的第一个变量进行读操作的时间，最大值指对最后一个变量 -1 进行读操作的时间。最大值和最小值之间的差别来自于寻找存储的最后一个变量数据所花费的时间。
5. 首次或以无效状态运行该 EEPROM 机制（详细信息请参考*表 2: 模拟页的可能状态和相应行为第 9 页*）时，这两个页都被擦除，用于存储数据的页被标记为 VALID_PAGE。
6. 当有效页存在时 （EEPROM 已进行过至少一次初始化）可实现典型 EEPROM 初始化。典型 EEPROM 初始化过程中，两个页中的一个会被擦除 （详细信息请参考*表 2: 模拟页的可能状态和相应行为第 9 页*）。

3 嵌入式应用方面

本节在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要等方面提供了建议。

3.1 数据粒度管理

模拟 EEPROM 可用于嵌入式应用，这些应用可对数据 （需要以字节、半字或全字的颗粒度进行更新）进行非易失性存储。通常取决于用户需求或 Flash 结构 （例如，存储数据长度、写访问）。

STM32F2xx 片上 Flash 允许进行 8 位、 16 位或全字编程 （取决于所用电压范围，如*表 7* 中所描述）。

表 7. Flash 程序函数

数据粒度	函数名称	功能电压范围
全字形式 （32 位）	FLASH_ProgramWord	从 2.7 V 到 3.6 V。
半字形式 （16 位）	FLASH_ProgramHalfWord	从 2.1 V 到 3.6 V。
字节形式 （8 位）(1)	FLASH_ProgramByte	所有的器件电源电压范围均可适用。

1. 逐字节的编程：以字节形式写数据，为存储更多数据变量提供了可能。使用 FLASH_ProgramByte() 函数时，可能反而降低性能。我们建议您使用 FLASH_ProgramHalfWord() 函数。虚拟地址和数据均可同时以半字形式写入。

3.2 损耗均衡：提高 Flash 可擦写次数

STM32F2xx 片上 Flash 中，每个扇区可编程或擦除约 1 万次。

对于模拟 EEPROM 中使用超过 2 页（3 或 4）的写密集型应用，建议利用损耗均衡算法来监控和分配存储页中写周期的数量。
不使用损耗均衡算法时，这些存储页的使用频率不同。长期活跃数据的存储页不能支持像频繁更新数据的存储页那样多的写周期。损耗均衡算法保证了每个扇区都能均等使用其全部可用的写周期。

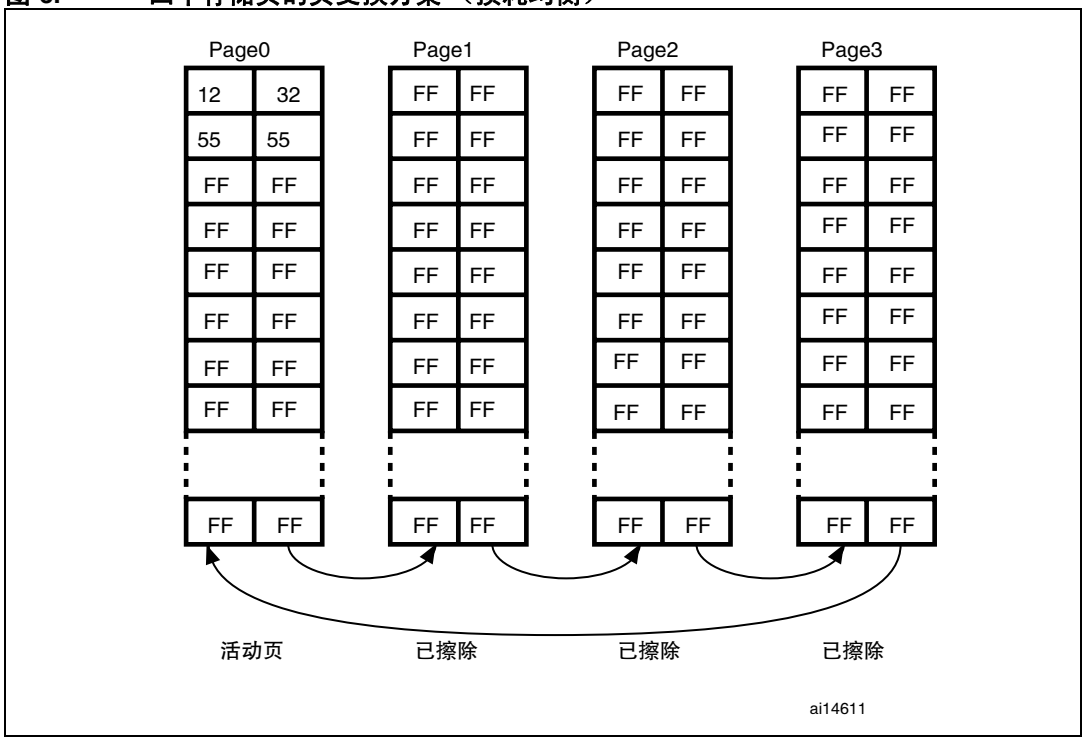
注： *STM32F2xx Flash 中的主存储器块在表 3: STM32F2xx Flash 扇区中有单独描述。*

3.2.1 损耗均衡实现示例

为了提高模拟 EEPROM 容量，本示例中将会使用四个扇区 （sector5 为 Page0， sector6 为 Page1， sector7 为 Page2， sector8 为 Page3）。

损耗均衡算法实现如下：当 page *n* 满时，设备转换为 page *n+1*。 Page *n* 被回收然后擦除。当轮到 Page3 满时，设备返回到 Page0，回收 Page3 然后将其擦除，以此类推（参考*图 6*）。

图 6. 四个存储页的页交换方案 （损耗均衡）



该软件中，可用 EE_FindValidPage() 函数 （参考*表 4*）实现损耗均衡算法。

3.3 失去供电情况下的页头恢复

变量更新、页擦除或传输过程中失去供电，可能发生数据或页头损坏。

为了检测这种损坏并从中恢复，需要执行 *EE_Init()* 程序。该程序应在断电后立即被调用。本应用笔记中描述了该程序原理。该程序利用页状态来检测 （该页的）完整性，必要时执行修复。

失去供电后，使用 *EE_Init()* 程序检测页头状态。共有 9 种可能的状态组合，其中 3 种是无效的。*表 2: 模拟页的可能状态和相应行为第 9 页*显示了基于断电时的页状态所应采取的措施。

3.4 可擦写性能和存储页分配

3.4.1 可擦写性能

编程 / 擦除周期包含了一个或更多的写访问和一个页擦除操作。

使用 EEPROM 技术时，每个字节可进行有限次数的编程或擦除，通常在 1 万次至 10 万次范围内。

但是在嵌入式 Flash 中，最小擦除空间单位为扇区，一个扇区的编程 / 擦除次数是其可擦除的数量。 STM32F2xx 产品的电特性保证每个扇区可进行 1 万次编程 / 擦除周期。因此模拟 EEPROM 的最大寿命受限于最频繁写入的参数更新率。

可擦写性能取决于用户希望处理的数据量 / 规模。

本例中使用了两个扇区 （16 KB 的），采用 16 位数据编程。每个变量对应于一个 16 位的虚拟地址。即，每个变量占据了一个全字的存储空间。对于模拟 EEPROM 内存的一个存储页来说，其寿命内一个扇区具有共计 160000 KB （16 KB 乘以 1 万次的 Flash 可擦写次数）的数据存储容量。因此，假如模拟过程中使用了两个存储页，则模拟 EEPROM 可存储 320 000 KB 数据。如果使用了超过两个存储页，该数值相应地成倍增加 （160000 KB 乘以页数）。

知道了所存储变量的数据宽度，就可以计算出能够存入模拟 EEPROM 区域 （在其寿命内）的变量总数。

3.4.2 Flash 页分配

EEPROM 模拟应用所需要的扇区大小和扇区数量可根据系统寿命内写入数据量进行选择。

例如，具有 10KB 擦除周期的 16 KB 存储页可在系统寿命内写入最多 160 MB 数据。128 KB 存储页可在系统寿命内写入最多 1280 MB 数据。

空闲变量空间可计算如下：

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

其中：

- **Page size**：页大小单位为字节 （例如， 16 KB）
- **NbVar**：所用变量数 （例如， 10 个变量）
- **VariableTotalSize**：存储一个变量 （地址和数据）所用的字节数
  - 对于 32 位变量， VariableTotalSize = 8，它具有 （32 位数据 + 32 位虚拟地址）
  - 对于 16 位变量， VariableTotalSize = 4，它具有 （16 位数据 + 16 位虚拟地址）
  - 对于 8 位变量， VariableTotalSize = 2，它具有 （8 位数据 + 8 位虚拟地址）

保证应用寿命内预期的 Flash 操作所需要的存储页可估计如下：

所需页数：

NbPages = NbWrites / (PageEraseCycles * FreeVarSpace)

其中：

- **NbPages** = 所需页数
- **NbWrites** = 写入变量总数
- **PageEraseCycles** = 一页的擦除周期次数

注： *如果变量为 16 位，则每个变量占据 32 位空间 （16 位存放数据， 16 位存放虚拟地址），这意味着每个变量在进行每次新数据写入时需要使用 4 字节的 Flash。每个 16 KB 的 （或 128 KB 的）存储页在其写满前可支持 4096 （或 32768）个变量写入。*

注： *这是略保守的估计值。*

使用示例的情况

设计一个应用，使其更新 **20** 个不同的变量，每隔 **2** 分钟更新一次，共 **10** 年：

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

- 如果变量为8位数据并采用8位虚拟地址，保证所有数据实现非易失性存储所需要的页数为：
  - 两个 16 KB 的页或者
  - 两个 128 KB 的页
- 如果变量为16位数据并采用16位虚拟地址，保证所有数据实现非易失性存储所需要的页数为：
  - 两个 16 KB 的页或者
  - 两个 128 KB 的页
- 如果变量为32位数据并采用32位虚拟地址，保证所有数据实现非易失性存储所需要的页数为：
  - 三个 16 KB 的页或者
  - 两个 128 KB 的页

表 8. 应用设计

变量大小	写入次数 (NbWrites)	要写入的数据总量（以字节计）	页大小 （KB）	页擦除周期	所需页数	所用页数 (1)
8 位	52 560 000	105 120 000	16	10000	0.6	2
			128		0.1	
16 位	52 560 000	210 240 000	16	10000	1.3	2
			128		0.2	
32 位	52 560 000	420 480 000	16	10000	2.6	3
			128		0.3	2

1. 模拟 EEPROM 最少需要两个存储页

3.5 实时性考虑

EEPROM 模拟固件所提供的实现由内部 Flash 运行，因此在操作需要进行 Flash 擦除或编程（EEPROM 初始化、变量更新或页擦除）的过程中，将会停止访问 Flash 。因此 （这时）应用代码不被执行且不进行中断服务。

这种行为对许多应用来说是可接受的，但是对于有实时性限制的应用来说，您就需要从内部 RAM 来运行关键流程。

在这种情况下：

1. 重置内部 RAM 中的向量表。
2. 从内部 RAM 上执行所有的关键代码和中断服务程序。编译器向作为 RAM 函数的声明函数提供关键字；系统启动时，将该函数从 Flash 拷贝到 RAM 中，同任意初始化变量操作一样。需要注意的是，对于一个 RAM 函数，所有使用的变量和调用的函数都应当在 RAM 中。

4 修订历史

表 9. 文档修订历史

日期	修订	变更
2011 年 6 月 6 日	1	初始版本。
2011 年 11 月 07 日	2	更新了*第 1 章节：外部 EEPROM 和模拟 EEPROM 之间的主要区别*标题，和*表 1* 中的写入时间、擦除时间、读访问时间和注释。 更新了*第 1.2 章节：擦除时间的不同* 和*第 1.3 章节：写方式的相似之处*。 增加*表 2: 模拟页的可能状态和相应行为*。 修改了*表 4: API 定义*中的 `EE_PageTransfer()` 说明。 更新了*表 6: 采用 120 MHz 系统时钟的 EEPROM 模拟时序*。 增加*第 3.5 章节：实时性考虑*。

重要通知 - 请仔细阅读

意法半导体公司及其子公司 （“ST”）保留随时对 ST 产品和 / 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于 ST 产品的最新信息。 ST 产品的销售依照订单确认时的相关 ST 销售条款。

买方自行负责对 ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的 ST 产品如有不同于此处提供的信息的规定，将导致 ST 针对该产品授予的任何保证失效。

ST 和 ST 徽标是 ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。