提高 MSP430G 系列单片机的 Flash 擦写寿命的方法

MSP430 产品技术支持

摘要

在嵌入式设计中，许多应用设计都需要使用 EEPROM 存储非易失性数据，由于成本原因，某些单片机在芯片内部并没有集成 EEPROM。MSP430G 系列处理器是 TI 推出的低成本 16 位处理器，在 MSP430G 系列单片机中并不具备 EEPROM。为了存储非易失性数据，MSP430G 系列处理器在芯片内部划分出了 256 字节的 Flash 空间作为信息 Flash，可用于存储非易失性数据，但是由于 Flash 与 EEPROM 在擦写寿命上存在一定差距，所以在实际应用中，这种应用方式并不能够满足所有客户的需求。本应用笔记介绍了使用代码区域 Flash 来模拟 EEPROM，通过一定的软件处理算法，可以大大增加数据存储周期的一种方法。本文给出了实现上述功能的软件流程。

1. 嵌入式 Flash 存储介质与 EEPROM 的主要特性对比

电可擦除和编程只读存储器（EEPROM）是在绝大多数嵌入式应用中都会使用到的用于保存非易失性数据的关键器件，用于在程序运行期间保存数据。Flash 闪存（Flash Memory，简称为"Flash"）是一种非易失性（Non-Volatile）存储器，广泛应用于各种嵌入式处理器中，用于存储程序代码。

由于硬件成本原因，在许多嵌入式处理器中并没有集成 EEPROM 模块，通常我们可以采用在片内 Flash 存储器中保存非易失性数据的应用方式来达到使用要求。对一些普通的应用场合，这种使用方式可以满足要求。

表一 EEPROM 与 Flash 对比分析

特性	EEPROM	MSP430G 系列 Flash
写时间	几个ms 随机字节写： 5 到10 ms 页写: 100μs每字 (5 to 10 ms 每页)	字节写：30 个 Flash 操作时钟周期，典型数据 70us
擦除时间	N/A	页擦除：4819 个 Flash 操作时钟周期，典型数据 10ms 全部擦除：10593 个 Flash 操作时钟周期，典型数据 20ms
擦写方法	一旦启动写动作，不依赖 CPU，但需要持续的电源供给	需要芯片内部执行升压操作
读取访问方式	连续方式: 大概100μs 随机字方式: 大概92μs 页方式: 22.5μs 每字节	N/A
擦除次数	10 万次以上，典型参数 100 万次	1 万次以上，典型参数 10 万次

1.1 写访问时间

由于 EEPROM 和 Flash 的工作特性不同，所以写访问时间也不相同。Flash 具有更短的写访问时间，所以更适用于对存储速度有要求的场合。

1.2 写方法

外置 EEPROM 和采用 Flash 模拟 EEPROM 的最大不同之处在于写的方法。

EEPROM：对 EEPROM 的写操作不需要额外的操作，只需要提供电源供给；但是一旦启动写操作流程后，写操作不能够被打断。所以需要外接电容器等措施来保证在芯片掉电时能够维持供电，保证完成数据操作。

Flash 模拟 EEPROM：当芯片上电后，写操作可以被电源掉电和芯片复位打断。和 EEPROM 相比，需要应用设计者增加相关的处理来应对可能存在的异常。

1.3 擦写时间

EEPROM 和采用 Flash 模拟 EEPROM 在擦除时间上存在很大的差异。

与 Flash 不同，EEPROM 在进行写操作之前不要擦除操作。由于 Flash 需要几个毫秒时间进行擦除操作，所以如果在进行擦除操作的过程中出现电源掉电的情况，需要软件做相关的保护处理。为了设计一个健壮的 Flash 存储器的管理软件，需要深入的了解和掌握 Flash 存储器的擦除过程特性。

2. 增加 Flash 模拟 EEPROM 擦写寿命的方法

可以根据用户的需求采用不同的方法实现 Flash 存储器模拟 EEPROM。

2.1 虚拟地址加数据方案

通常需要两个页以上的 Flash 空间来模拟 EEPROM。上电后，初始化代码先查找出有效页，同时将另外一个页初始化为擦除状况，以提供字节写的能力，并用作备份和随时准备执行写入操作。需要存储 EEPROM 的变量数据首先写入有效页，当有效页写满后，需将所有数据的最后状态保存到备份页，并切换到备份页进行操作。每一页的第一个字节通常用来指示该页的状态。

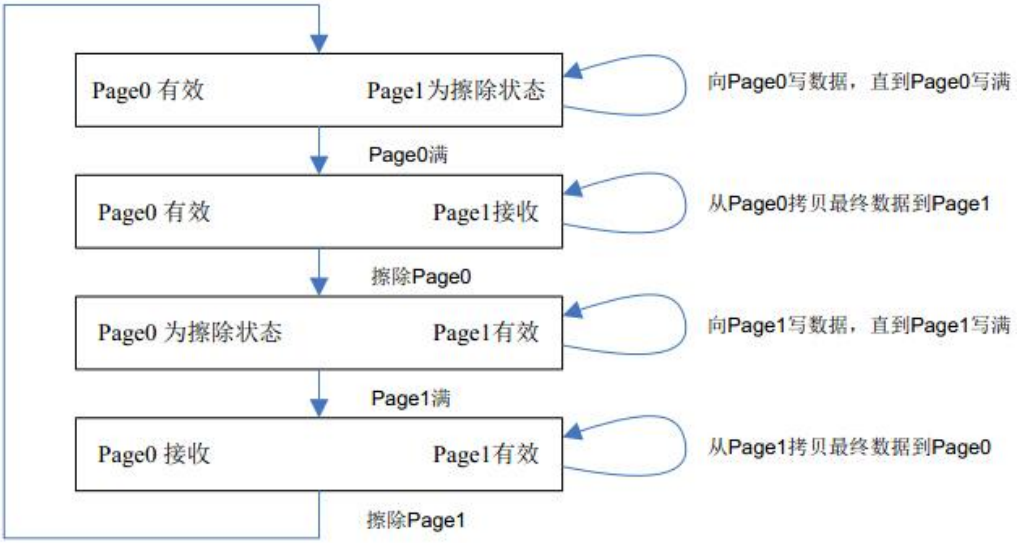
每个页存在 3 种可能状态：

擦除态：该页是空的。

已写满数据状态：该页已经写满数据，准备切换到下一个页进行操作。

有效页状态：该页包含着有效数据并且标示状态尚未改变，所有的有效数据全部拷贝到了已经擦除的页。

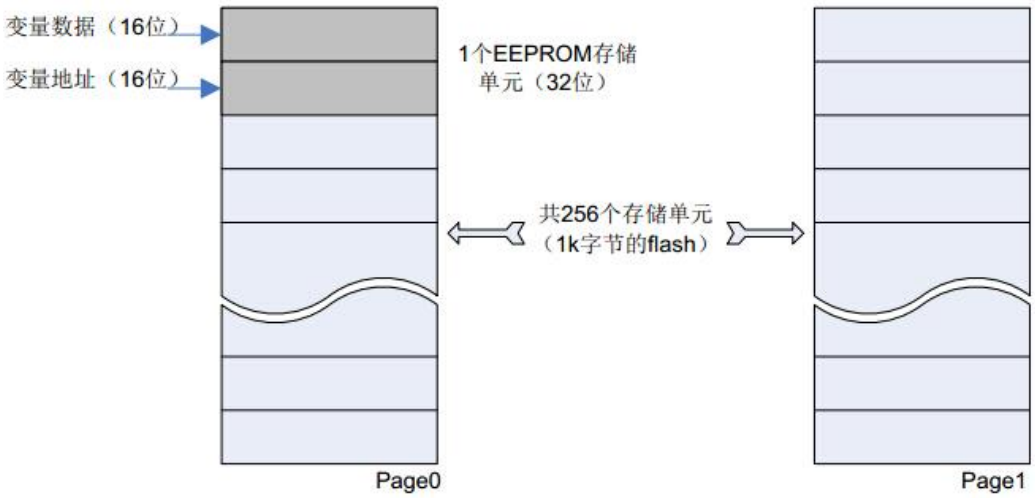
下图以采用两个页模拟 EEPROM 的方式为例，描述了页状态字的在页 0 和页 1 之间的切换过程。



图一 页状态字的在页 0 和页 1 之间的切换

采用这种方式，用户不知道数据刷新的频率。

下面的图例以采用两个页模拟 EEPROM 的应用方式为例进行描述。为了方便获取模拟 EEPROM 数据和更新数据内容，每个存储变量元素都在 Flash 里定义了一个操作单元，在该操作单元中对每个存储变量元素都分配一个虚拟操作地址，即一个 EEPROM 操作单元包含一个虚拟地址单元和一个数据单元。当需要修改数据单元内容时，新的数据内容和之前分配的虚拟地址一同写入一个新的模拟 EEPROM 存储器单元中，同时返回最新修改的数据内容。EEPROM 存储单元格式描述如图二。



图二 EEPROM 存储单元格式

使用虚拟地址加数据的方案总结如下。

• 为每一个目标存储变量分配一个虚拟地址，该虚拟地址需一同存入 Flash 中。当读取存储变量内容时，需根据该变量的虚拟地址搜索虚拟 EEPROM 并返回最后更新的内容。

• 在软件处理上，需要记录下一次写入的物理目的地址；在每一次执行写入操作后，根据 EEPROM 存储单元大小（操作粒度），将目的操作指针自动累加。

• 当一个页（Page）写满后，需要将所有变量的 EEPROM 数据拷贝到下一个页，再执行该页的擦除操作。

• 在嵌入式软件处理上需加入合适的校验机制，保证写入数据的正确性并监测 Flash 是否已经失效。

2.2 划分子页方案

在 Flash 中划分出至少 2 个页（Page）用作模拟 EEPROM，根据应用需求将需写入 EEPROM 进行保存的变量数据划分成一个定长的数组（子页），例如 16 个字节或者 32 字节，将页划分成若干子页后，需对 Flash 中的所有子页按照地址顺序进行逐次编号。每个子页的第一个字节通常用来指示该子页的状态，子页状态可以为：空、已写入或者失效。

在芯片上电初始化时，首先查找出第一个尚未写入数据的子页，并进行标识，在进行写 EEPROM 操作时，应用程序需将待写入 EEPROM 子页的所有数据按照事先约定好的顺序整理好，再一次性将所有变量数据写入空的子页中，最后将模拟 EEPROM 的操作指针指向下一个空闲的子页，等待下一次写入。待将一个页的数据写满后，再进行一次擦除操作。需要处理好指向子页的指针的跳转。

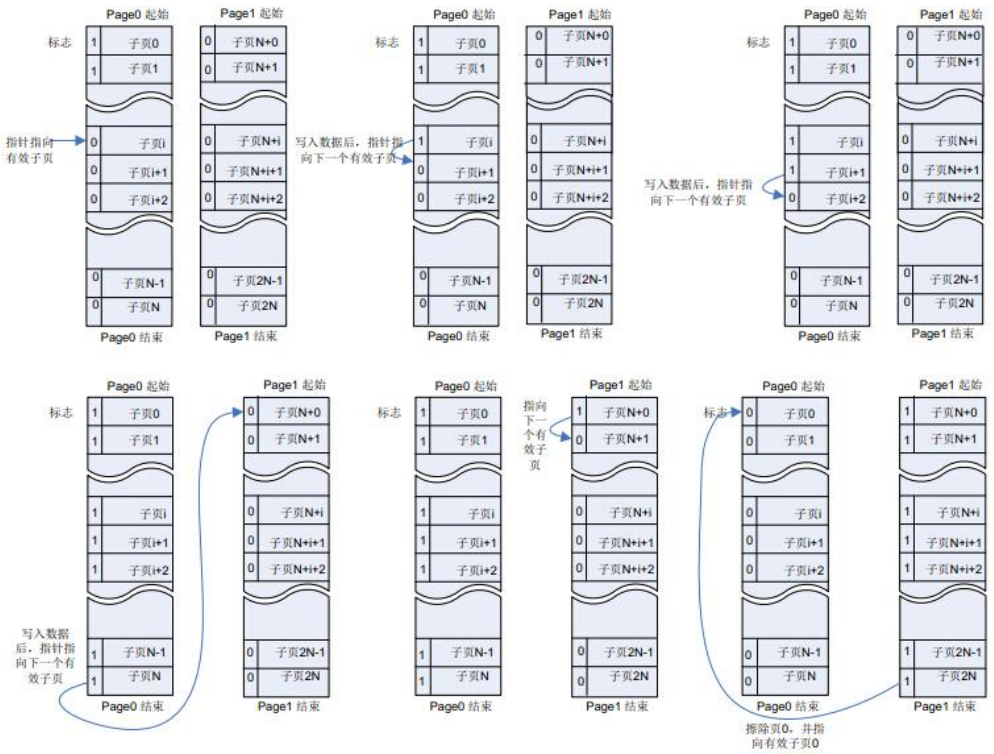
每个页存在 3 种可能状态：

擦除态：该页是空的。

已写满数据状态：该页已经写满数据。

有效页状态：该页包含着有效数据并且该页尚未写满，仍可向子页写入数据。

图三介绍了使用子页的方式实现 Flash 模拟 EEPROM 的数据处理方法。



图三 使用子页的方式模拟 EEPROM 存储单元

2.2.1 软件描述

在软件实现上，为了便于软件处理，建议定义一些关键宏定义和结构体，指定 Flash 模拟 EEPROM 的起始、结束地址、页的大小、子页的大小、每个页的子页数目等参数，同时将需要操作的参数封装起来，便于软件操作和管理，不建议定义许多离散的标志变量。

```
#define CODE_FLASH_EEPROM_STR_ADD 0xF800    // Start address
#define CODE_FLASH_EEPROM_END_ADD 0xFBFF    // End address
#define SEGMENT_SIZE              512       // Page size
#define SUB_SEGMENT_SIZE          16        // Sub-page size
#define SUB_NUM_PER_SEGMENT      (SEGMENT_SIZE)/(SUB_SEGMENT_SIZE)    // Sub-page numbers of each page
#define SUB_SEGMENT_NUMBER       (CODE_FLASH_EEPROM_END_ADD - CODE_FLASH_EEPROM_STR_ADD)
                                 /SUB_SEGMENT_SIZE                    // Total numbers of sub-segment

typedef struct
{
    u8  EnFlag;         // Aged flag
    u8  WrtFlag;        // Write enable flag
}FLASH_WRT_PARA;

typedef struct
{
    u8  *ptr_dst;     // point to corrent code flash address
    u8  *ptr_next;    // point to next code flash address
    u8  SubSegIdx;    // sub-segment index of each page
    u8  SegIdx;       // segment(page) index
    u8  Index;        // sub-segment index in total
}TARGET_WRT_ADDR;

FLASH_WRT_PARA      FlashWrtPara[SUB_SEGMENT_NUMBER];   // each sub-segment flag
TARGET_WRT_ADDR     FlashWrtAddr;        // writing parameter
```

在软件操作上，Flash 模拟 EEPROM 模块需要提供几个 API 接口给应用程序调用。

• 通过 typedef 关键字定义设备类型，typedef unsigned char u8;

• ChkFstPowerOnInfo（）用于检测芯片是否为第一次上电并初始化 EEPROM 参数到内存，原型如下。

Void ChkFstPowerOnInfo(void);

• FlashWrite（）用于写 Flash，传递的形参包括指向待写入数据的指针，待写入数据在子页中的起始字节编号，写入数据的长度，原型如下。

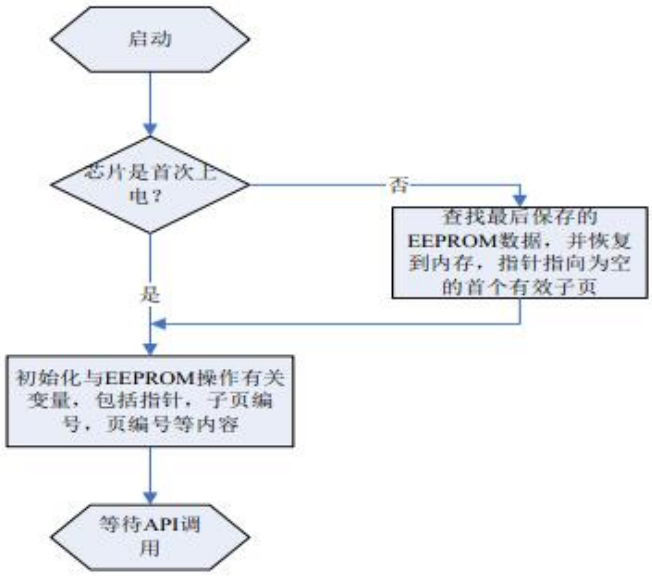
void FlashWrite(u8 *array, u8 startNum, u8 length);

• FlashErase（）用于擦除 Flash，传递的形参是子页的编号，在擦除函数中需要根据子页的编号判断是否需要执行页的擦除操作，原型如下。

void FlashErase(u8 seg_sn);

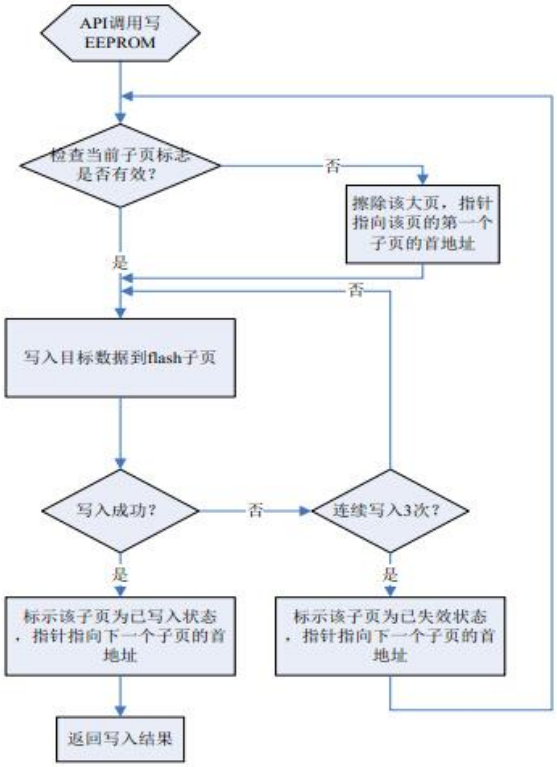
2.2.2 软件流程图

软件启动后，初始化模拟 EEPROM 流程图描述如下。



图四 初始化流程图

调用 API，向模拟 EEPROM 写入数据的软件流程如图五所示。在软件处理中，要特别注意目标指针的切换和保证写入数据的正确性，在代码空间允许的情况下，可以增加一些校验算法来保证。



图五 写入模拟 EEPROM 数据流程

采用划分子页的方案总结如下。

• 每次写入模拟 EEPROM 的数据长度为定长，即为子页的长度。

• 软件需要定义一个存储变量结构体，用于刷新和同步模拟 EEPROM 内容。在将数据写入模拟 EEPROM 之前，程序员需要按照约定的数据格式，在内存中将所有的目标存储变量进行整理。

• 在软件处理上，需要计算当前写入和下一次写入的物理地址；在每一次执行写入操作后，根据子页长度大小，将指向子页的目的操作指针自动累加。

• 待一个页（Page）写满后，需要将最后更新的模拟 EEPROM 数据拷贝到下一个页，再对写满页执行一次擦除操作。

• 在嵌入式软件处理上需加入合适的校验机制，保证写入数据的正确性并监测用于模拟 EEPROM 功能的 Flash 子页是否已经失效。

2.3 两种方案的对比分析

两种方案的对比分析见表二。

表二 两种方案的对比分析

	虚拟地址加数据的方案	划分子页的方案
优点	对所有存储变量进行了虚拟地址预分配，完全模拟了 EEPROM 的地址加变量数据的访问方式，易于理解并且操作简便。	对所有存储变量进行了封装，通过由模拟 EEPROM 驱动模块提供的 API 接口进行整体操作，操作简便。 存储空间利用率高。
缺点	由于为每个存储变量分配了虚拟地址，在有限 Flash 资源前提下，存储空间利用率低，理论利用率低于 50%。	每次数据保存，都需要对整个子页进行写操作，效率较低。 在每次将数据保存到模拟 EEPROM 之前，需要应用程序将待写入的变量数据结构体进行整理，增加软件开销。
总结	两种方案都可以提高 Flash 的擦写寿命，用户可以结合自己的应用设计进行方案选择： 在有限资源前提下，如需要更大容量的数据存储空间，建议选择划分子页的方式； 在实际应用中，可以根据不同的需求，将存储变量进行分类：将可能频繁变化和需要保存的非易失性数据存储到 Flash 模拟 EEPROM（code Flash）中，将不会经常改变的非易失性数据存储到信息 Flash（information Flash）中，从而增加 Flash 模拟 EEPROM 模块的利用率，更加灵活的实现数据保存。	

3. 实际的嵌入式应用

根据软件需要，建议采用字节（8bit）做为操作的最小粒度，适用性会更广泛。

3.1 Flash 存储器擦写寿命的提升

对于 MSP430G 系列的 Flash 存储器，可以保证至少 10000 次的编程和擦除寿命。如图六所示。

Flash Memory

over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	V_{CC}	MIN	TYP	MAX	UNIT
$V_{CC(PGM/ERASE)}$	Program and erase supply voltage			2.2		3.6	V
f_{FTG}	Flash timing generator frequency			257		476	kHz
I_{PGM}	Supply current from V_{CC} during program		2.2 V/3.6 V		1	5	mA
I_{ERASE}	Supply current from V_{CC} during erase		2.2 V/3.6 V		1	7	mA
t_{CPT}	Cumulative program time(1)		2.2 V/3.6 V			10	ms
$t_{CMErase}$	Cumulative mass erase time		2.2 V/3.6 V	20			ms
	Program/erase endurance			10^4	10^5		cycles

图六 MSP430G 系列单片机 Flash 编程和擦除寿命

采用划分小页结合至少分配 2 个大页的操作方式，则可以大大增加 Flash 模拟 EEPROM 的擦写寿命。例如，对于 MSP430G 系列单片机，如果将每个小页的尺寸划分为 16 字节，采用 2 个大页（每页 512 字节）作为模拟 EEPROM 使用，则可以提供 64 个操作子页（(512/16)x2=64），可以保证至少 640000 次的擦写寿命。

3.2 掉电时的异常处理

如果正在进行 Flash 数据存储时发生掉电，数据可能会保存不成功，存在异常。为了增强健壮性，在软件处理上，需要考虑设备异常掉电等可能会导致 Flash 擦写失败的情况。

在软件处理中，当成功保存 Flash 数据后，再写入该子页的状态标志。单片机上电后，用户程序将查找最后一次写入的子页，再将该子页的数据内容并恢复到内存中的数据结构中。

4. 系统可靠性设计

4.1 时钟源的选择

由于驱动 Flash 的时钟源（ACLK、MCLK、SMCLK）和时钟频率可以设定，为了保证在将数据写入模拟 EEPROM 时的可靠性，建议在将 Flash 的时钟频率降低后，再对其进行操作。例如将 Flash 的时钟频率降低到 1MHz 后，进行写入操作。需要注意，在降低了时钟频率后，若此时钟源也是定时器（Timer）的时钟源，则可能会影响到定时器的定时准确性，需要软件上做好处理。

4.2 代码在 RAM 中运行

由于向 Flash 写入数据操作是通过执行 Flash 中程序代码，对 Flash 进行擦除和编程操作。由于对 Flash 的编程需要 mcu 内部执行一个升压操作，所以如果有足够的内存空间，建议将编程、擦除等关键代码拷贝到 RAM 中运行，可以使用关键字__ramfunc 指定，如下图七所示。


__ramfunc

Syntax

Follows the generic syntax rules for object attributes, see Object attributes.

Description

The __ramfunc keyword makes a function execute in RAM. Two code segments will be created: one for the RAM execution, and one for the ROM initialization.

Functions declared __ramfunc are by default stored in the CODE_I segment.

Example

```
__ramfunc int FlashPage(char * data, char * page);
```

图七 使用关键字__ramfunc 将程序指定到 Ram 中运行

5. 总结

本文从软件方面，以及安全性方面探讨了使用 MSP430G 系列单片机在使用 Flash 模拟 EEPROM 方面的应用，提供了两种不同的方式供选择。两种方式都可以大幅度提高模拟 EEPROM 的编写、擦除寿命，并且满足高可靠性的应用设计，用户可以结合具体的应用进行选择。

参考文档

1. MSP430x2xx family user’s guide （SLAU144）

2. MSP430G2x53 datasheet （SLAS735）