SPC1168 Flash 存储器寿命延长使用指南

概述

Spintrol 提供了 flash 模拟 eeprom 算法，可将数据存储单元寿命延长，亦即最具体能延长多少倍寿命，取决于使用 flash 中多少 sector 来模拟 eeprom。

目录

图片列表

版本历史

版本	日期	作者	状态	变更
A/0	2023-06-21	CanChai	Outdated	首次发布。
C/0	2024-04-18	Jiali Zhou	Released	修改排版格式。

术语或缩写

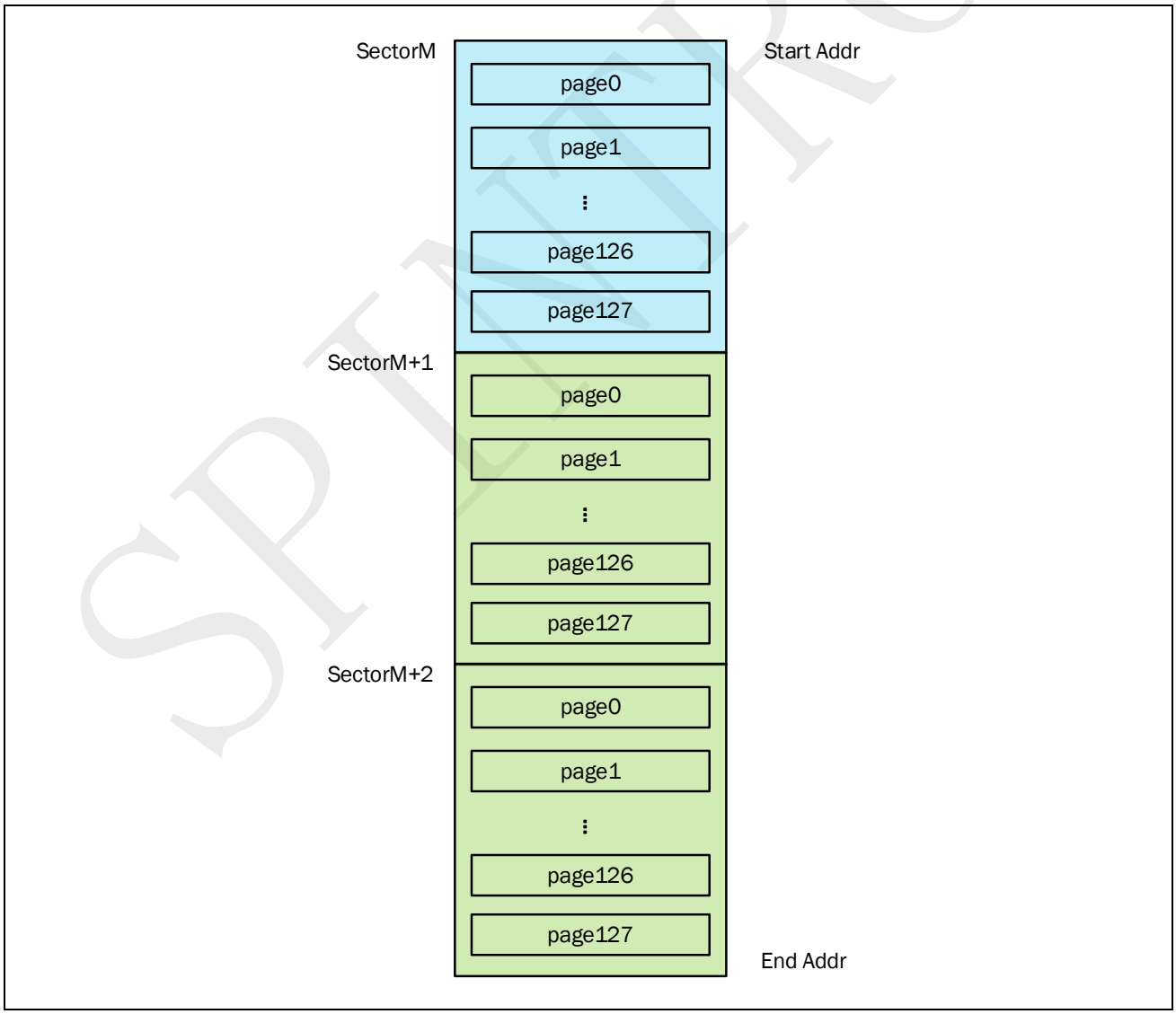
术语或缩写	描述
/	/

1 引言

SPC11X8/SPD11X8 内部集成了 Flash，数据空间最大可达 128KB，每个 Sector(512 byte)可经历 100K 次的擦写，在工程中，有时候需要存储一些实时变量，存储频率非常高，每次存储时都需要将数据所在的扇区擦除一次，然后才能将新的数据写入，若不对这种行为加以特殊处理，经常性的擦除 flash，很快会将 flash 擦坏。为此， Spintrol 提供了 flash 模拟 eeprom 算法，可将数据存储单元寿命延长，亦即最具体能延长多少倍寿命，取决于使用 flash 中多少 sector 来模拟 eeprom。

为方便理解，以下篇幅将以一个示例来阐明本方法的具体做法，为简化说明，并假设存储数据不超过一个 sector 大小(512Bytes)，且用来模拟 eeprom 的 sector 数量小于 128 个 sector，另外，在以下示例说明中，称 sector 的 4 个 bytes 为一个 page。

图 1-1: 轮序存储表示图



如上图，假设使用 3 个 sector 用来模拟 eeprom，以延长 flash 的使用寿命，则在 Spintrol 算法中，会自动测算出：将使用 sectorM 作为索引区，sectorM+1 以及 sectorM+2 将作为数据存储区（由于假设每次存储的数据大小小于一个 sector，所以数据区能够存储 2 次数据）。当第一次存储数据时，算法会将数据存储在 sectorM+1 扇区，并将 sectorM 的 page0 写入一个特定的标记数据（称为魔数），以此表明 sectorM+1 这个扇区已经被写过一次数据，是一个脏扇区；当第二次有数据写入时，算法首先逐个 page 读取 sectorM 扇区的数据，查看哪个 page 的数据不是魔数，假如 pageX 里的数据不是魔数，则将数据写入 sectorM+X 的扇区。在第三次写入数据时，由于数据区全部都是脏扇区，所以算法将会擦除所有索引区以及数据区，然后开始新的一个循环写入数据。

从以上的例子中，可知，采用了算法之后，原本需要每次写入数据之前，都需要擦除 flash 对应的扇区，但采用算法之后，每写入 2 次数据，才会擦一次 flash，也即提高了 flash 的使用寿命。

Spintrol 所提供之 Data manage 代码不保证数据写入前后的正确性，若使用者有需求，请自行加入 checksum 机制确保数据之正确性。

在以上的例子中，对每次写入的数据以及用来模拟 eeprom 的 sector 数量进行了限制，这只是为了能够使说明得到简化，实际的 Spintrol 算法，会按照用户传入的参数，自行计算出各种所需的参数，对每次写入的数据以及用来模拟 eeprom 的 sector 数量并无限制。

2 调用 API 进行存储

在 Spintrol 的 SDK 中能够找到名为《Flash_EEPROM_Emulation》的示例代码，使来演示使用 flash 模拟 eeprom，以便增加 flash 的使用寿命。

2.1 算法数据结构初始化

```
ErrorStatus EEPROM_Init(uint32_t u32StartAddr,uint32_t u32EndAddr, uint32_t u32UserDataSize);
```

u32StartAddr：用来模拟 eeprom 的起始地址，这个地址需要按照 sector 大小进行对齐，也即需要按照 0x200 的大小对齐。
u32EndAddr：用来模拟 eeprom 的结束地址，这个地址需要按照 sector 大小进行对齐，也即需要按照 0x200 的大小对齐。
u32UserDataSize：需要保存的数据结构的大小。

EEPROM_Init 函数即为算法的数据结构初始化函数，使用 flash 模拟 eeprom 之前，首先一定要调用此函数进行算法数据结构的初始化。

2.2 Data Manage 存储功能

```
ErrorStatus EEPROM_Write(struct UserData_t* UserData);
```

UserData：需要存储的数据结构。

每次需要向模拟的 eeprom 写入数据时，只需要调用此函数进行数据写入即可。

2.3 Data Manage 读取功能

```
void EEPROM_Read(struct UserData_t* UserData);
```

UserData：用来保存读取结果的指针。

需要读取最近一次写入的数据时，调用此函数即可。