来自网络的FLASH掉电总结

原因：

由于FLASH的擦除和写入的特性，如果在擦除中掉电或者写入时掉电，有可能会出现失败的情况，下次再上电读取就会出错；

擦除的时候掉电，不能保证擦除完全，但是已经擦除的部分，肯定是0xff了。写的时候掉电，已经写入的就是正确的，电平掉到最低极限工作电压附近时，写入的就无法保证了，地址无法保证，写入的值也无法保证。没有写到的，肯定还是0xff。

解决方式：

# 硬件方式解决—掉电保护机制

* 必须硬件电路上有相应的掉电警告中断，而且必须有足够的时间供软件响应。
* 楼上还有一位兄弟说的对;掉电保存和上电恢复;需要以下:  
  1)快速检测掉电信号;(太敏感了也不好,常常喊狼来了);  
  2)要有足够的时间让你停下来保存现场;  
  3)上电后可靠的复位;  
  这些的关键是掉电信号\电源\复位信号三个怎么配合的事;软件怎么写应该大家都会;关键在硬件上;
* 掉电保护不是重新擦写那么简单，首先要有一个机制能知道发生了掉电事件，还要知道什么地方的数据是可能出错的，如果是新建文件还好一点，一般就是数据作废，如果是改写一个文件时发生掉电事故的话，还要有恢复机制，将数据恢复成改写以前的数据。由于flash的先擦后写的特性，不会在原来位置重新写数据，一般都是把原位置的数据标记为无效，再在新位置写入数据，对于文件来说有一个存储位置的关系，这不同于文件系统的逻辑地址，每个逻辑地址要有对应的物理地址，要有一套逻辑-物理地址的转换映射关系在里面。flash空间总是有限的，还要有垃圾回收的机制，系统要将无效数据的存储空间回收。而nand的擦单元是写单元的2的n次幂（如32，64，128）倍，如果一个擦除单元内有有效数据的话，还要将有效的数据搬移到其它地方，要对相应的文件的地址映射进行更新维护。擦写均衡的实现也是一件比较复杂的事情。。。。 所以我们看到的U盘大小总是小于标定的大小，因为它内部要保留一定区间用来实现FTL的算法，保存FTL层的数据。
* 如果是这么个要求让我重新设计一下硬件的话会有如下几个部分：  
  1.电源管理芯片，检测掉电事件的发生；  
  2.储能元件，在掉电发生后能满足执行掉电保护的操作即有足够的时间存储一些标志位和必要的数据；  
  然后结合前面大家提到的软件掉电保护方法来实现。
* 一个电源检测芯片，一个储能元件（比如大点的电容）结合软件判断以及提前保存现场就能很好的实现掉电保护了  
  还是硬件上做些改动吧，成本又没多少，想纯软件实现感觉比较难。

# 软件方式解决

* 要改写某扇区内容，

1. 先把该扇区内容备份到一个特殊的扇区；
2. 在某个地方写一个特殊的标记；
3. 完成该扇区的改写操作，清除标记；
4. 系统每次上电或复位，都去检查下标记，就跟windows下的“磁盘扫描程序”一样；即使最坏情况，也可以恢复改写之前的数据；

* 这个过程“备份->置标记->写->清标记”的办法，是个可行的办法，也是目前比较通行的办法。  
  但是要注意：这个标记必须是单独占用一个无用的扇区（没有实际有用数据的扇区），保证对标记的操作不会影响到实际数据。  
  二是这只能保证数据能恢复到之前的状态，不能确保数据结构的完整性。譬如你有一个数据结构有16字节（它们之间是有逻辑关系的），跨两个扇区保存，现在要全面更新这16个字节的数据，如果发生某个扇区没有成功更新（即恢复到之前的数据），就会发生新旧数据同时存在的情况，程序利用这些数据来操作，就有可能发生错误。所以，还需进一步的机制来保证“整体数据结构”的完整性，譬如某个文件的一部分数据坏掉了，那么就把整个文件无效掉。