



(12) 发明专利申请

(10) 申请公布号 CN 113488098 A

(43) 申请公布日 2021. 10. 08

(21) 申请号 202110820057.6

(22) 申请日 2021.07.20

(71) 申请人 南京冷火电子科技有限公司  
地址 210000 江苏省南京市建邺区云锦路  
45号730室

(72) 发明人 柴天新

(74) 专利代理机构 南京中擎科智知识产权代理  
事务所(普通合伙) 32549  
代理人 黄智明

(51) Int. Cl.  
G11C 16/34 (2006.01)  
G11C 16/16 (2006.01)

权利要求书1页 说明书3页 附图1页

(54) 发明名称

基于FALSH存储器的计数算法

(57) 摘要

本发明公开了基于FALSH存储器的计数算法,涉及存储器的技术领域,旨在解决现有存储设备对整扇区的擦除,导致了使用寿命减少的问题。其技术方案要点是对n个扇区的预留给字节从低位至高位依次置0,当第1至i个扇区(A<sub>1</sub>+A<sub>2</sub>+...+A<sub>i</sub>)个字节均置全0,第i+1个扇区A<sub>i+1</sub>个字节中的第1个字节最低位置0,同时对第1至i个扇区进行擦除处理,从第1个扇区中的第一个字节最低位重新开始置0;当第1至i个扇区的A<sub>1</sub>、A<sub>2</sub>、...、A<sub>i</sub>个字节再次均置全0后,将第i+1个扇区的A<sub>i+1</sub>个字节中第1个字节置0位的高一位置0;重复进行擦除和置位操作,直至n个扇区的字节均置全0。本发明达到了延长存储器使用寿命的效果。

脉冲	u8_Second_Eight_Bit_Num								u8_First_Eight_Bit_Num							
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	
0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	
1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	
1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	
1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	
1	1	1	1	1	1	1	1	0	1	1	1	0	0	0	0	
1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	
1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	
1	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	

1. 一种基于FALSH存储器的计数算法,其特征在于,包括以下步骤:

在存储器中选取n个扇区,在n个所述扇区中分别预留 $A_1$ 、 $A_2$ 、 $A_3$ 、……、 $A_n$ 个字节,对n个扇区进行擦除处理,令 $A_1$ 、 $A_2$ 、 $A_3$ …… $A_n$ 个字节置为全1;

对n个扇区的预留字节从低位至高位依次置0,当第1至i个扇区( $A_1+A_2+\dots+A_i$ )个字节均置全0,第i+1个扇区 $A_{i+1}$ 个字节中的第1个字节最低位置0,同时对第1至i个扇区进行擦除处理,令第1至i个扇区的 $A_1$ 、 $A_2$ 、……、 $A_i$ 个字节置为全1,从第1个扇区中的第一个字节最低位重新开始置0;

当第1至i个扇区的 $A_1$ 、 $A_2$ 、……、 $A_i$ 个字节再次均置全0后,将第i+1个扇区的 $A_{i+1}$ 个字节中第1个字节置0位的高一位置0;

重复进行擦除和置位操作,直至n个扇区的字节均置全0。

2. 根据权利要求1所述的基于FALSH存储器的计数算法,其特征在于:所述存储器的使用寿命扩展为原使用寿命的 $\prod_{i=1}^n A_i \times 8^n$ 倍。

3. 根据权利要求2所述的基于FALSH存储器的计数算法,其特征在于:所述擦除和置位操作在存储器接收到脉冲信号后进行。

4. 根据权利要求3所述的基于FALSH存储器的计数算法,其特征在于:当所述存储器接收到脉冲信号,触发置位操作,或者同时触发擦除和置位操作。

5. 根据权利要求4所述的基于FALSH存储器的计数算法,其特征在于:当所述存储器接收到脉冲信号,若第1至i个扇区( $A_1+A_2+\dots+A_i$ )个字节均置全0,则同时触发第1至i个扇区的擦除处理和第i+1个扇区 $A_{i+1}$ 个字节中的第1个字节最低位置0操作,其余情况接收到脉冲信号仅触发置位操作。

6. 根据权利要求5所述的基于FALSH存储器的计数算法,其特征在于:当n个扇区中 $A_1$ 、 $A_2$ 、 $A_3$ …… $A_n$ 个字节均置全0时,对n个扇区进行擦除处理,自第1个扇区 $A_1$ 个字节中第1个字节的最低位开始置0。

## 基于FALSH存储器的计数算法

### 技术领域

[0001] 本发明涉及存储器的技术领域,尤其是涉及一种基于FALSH存储器的计数算法。

### 背景技术

[0002] 在嵌入式领域,FLASH存储芯片有非常广泛的应用,硬件设计和驱动程序简单,相对于铁电(FRAM)存储器而言成本较低,可靠性好。因此在低成本的测控设备中有普遍应用,但是FALSH存储芯片都有使用寿命的限制,一般为100万次。在嵌入式设备中,会涉及到频繁数据的存储,比如电表或者雨量计脉冲计数和累加、投币机的脉冲计数和累加等,该计数总量在应用中往往会远远大于100万次,并且对于存储数据的可靠性以及准确性有很高要求。

[0003] FLASH的编程原理都是只能将1写为0,而不能将0写为1。所以在FLASH编程之前,必须将对应的扇区擦除,而擦除的过程就是把所有位都写为1的过程,将扇区内的所有字节变为0xFF,然后对FALSH进行最基本的读、写和擦除操作。正是因为需要整扇区进行擦除,导致了FLASH存储器使用寿命的减少,该问题有待解决。

### 发明内容

[0004] 本发明的目的是提供一种基于FALSH存储器的计数算法,其达到了延长存储器使用寿命的效果。

[0005] 本发明的上述发明目的是通过以下技术方案得以实现的:

[0006] 一种基于FALSH存储器的计数算法,包括以下步骤:

[0007] 在存储器中选取n个扇区,在n个所述扇区中分别预留 $A_1$ 、 $A_2$ 、 $A_3$ 、……、 $A_n$ 个字节,对n个扇区进行擦除处理,令 $A_1$ 、 $A_2$ 、 $A_3$ …… $A_n$ 个字节置为全1;

[0008] 对n个扇区的预留字节从低位至高位依次置0,当第1至i个扇区( $A_1+A_2+\dots+A_i$ )个字节均置全0,第i+1个扇区 $A_{i+1}$ 个字节中的第1个字节最低位置0,同时对第1至i个扇区进行擦除处理,令第1至i个扇区的 $A_1$ 、 $A_2$ 、……、 $A_i$ 个字节置为全1,从第1个扇区中的第一个字节最低位重新开始置0;

[0009] 当第1至i个扇区的 $A_1$ 、 $A_2$ 、……、 $A_i$ 个字节再次均置全0后,将第i+1个扇区的 $A_{i+1}$ 个字节中第1个字节置0位的高一位置0;

[0010] 重复进行擦除和置位操作,直至n个扇区的字节均置全0。

[0011] 本发明进一步设置为:所述存储器的使用寿命扩展为原使用寿命的 $\prod_{i=1}^n A_i \times 8^n$ 倍。

[0012] 本发明进一步设置为:所述擦除和置位操作在存储器接收到脉冲信号后进行。

[0013] 本发明进一步设置为:当所述存储器接收到脉冲信号,触发置位操作,或者同时触发擦除和置位操作。

[0014] 本发明进一步设置为:当所述存储器接收到脉冲信号,若第1至i个扇区( $A_1+A_2+\dots+A_i$ )个字节均置全0,则同时触发第1至i个扇区的擦除处理和第i+1个扇区 $A_{i+1}$ 个字节

中的第1个字节最低位置0操作,其余情况接收到脉冲信号仅触发置位操作。

[0015] 本发明进一步设置为:当n个扇区中 $A_1$ 、 $A_2$ 、 $A_3$ …… $A_n$ 个字节均置全0时,对n个扇区进行擦除处理,自第1个扇区 $A_1$ 个字节中第1个字节的最低位开始置0。

[0016] 综上所述,本发明的有益技术效果为:

[0017] 通过对扇区预留字节的有序管理,降低了存储器对扇区进行全面擦除的频次,大大延长了存储器的使用寿命,实现了嵌入式产品生命周期内存储器擦写次数的全部覆盖。

## 附图说明

[0018] 图1是本发明实施例一的整体结构示意图。

## 具体实施方式

[0019] 参照图1,本发明公开了一种基于FLASH存储器的计数算法,包括以下步骤:

[0020] 在存储器中选取n个扇区,在n个扇区中分别预留 $A_1$ 、 $A_2$ 、 $A_3$ 、……、 $A_n$ 个字节,对n个扇区进行擦除处理,令 $A_1$ 、 $A_2$ 、 $A_3$ …… $A_n$ 个字节置为全1;

[0021] 对n个扇区的预留字节从低位至高位依次置0,当第1至i个扇区( $A_1+A_2+\dots+A_i$ )个字节均置全0,第i+1个扇区 $A_{i+1}$ 个字节中的第1个字节最低位置0,同时对第1至i个扇区进行擦除处理,令第1至i个扇区的 $A_1$ 、 $A_2$ 、……、 $A_i$ 个字节置为全1,从第1个扇区中的第一个字节最低位重新开始置0;

[0022] 当第1至i个扇区的 $A_1$ 、 $A_2$ 、……、 $A_i$ 个字节再次均置全0后,将第i+1个扇区的 $A_{i+1}$ 个字节中第1个字节置0位的高一位置0;

[0023] 重复进行擦除和置位操作,直至n个扇区的字节均置全0。

[0024] 存储器的使用寿命扩展为原使用寿命的 $\prod_{i=1}^n A_i \times 8^n$ 倍。

[0025] 擦除和置位操作在存储器接收到脉冲信号后进行。

[0026] 当存储器接收到脉冲信号,触发置位操作,或者同时触发擦除和置位操作。

[0027] 当存储器接收到脉冲信号,若第1至i个扇区( $A_1+A_2+\dots+A_i$ )个字节均置全0,则同时触发第1至i个扇区的擦除处理和第i+1个扇区 $A_{i+1}$ 个字节中的第1个字节最低位置0操作,其余情况接收到脉冲信号仅触发置位操作。

[0028] 当n个扇区中 $A_1$ 、 $A_2$ 、 $A_3$ …… $A_n$ 个字节均置全0时,对n个扇区进行擦除处理,自第1个扇区 $A_1$ 个字节中第1个字节的最低位开始置0。

[0029] 本实施例的具体实施方式为:

[0030] 1.在第一扇区预留一个8bit的变量u8\_First\_Eight\_Bit\_Num,预先擦除第一扇区,置位u8\_First\_Eight\_Bit\_Num=0xFF;

[0031] 2.在第二扇区预留一个8bit的变量u8\_Second\_Eight\_Bit\_Num,预先擦除第二扇区,置位u8\_Second\_Eight\_Bit\_Num=0xFF;

[0032] 3.当嵌入式设备外围电路检测到脉冲后,开始计数:

[0033] 第一个脉冲到来,u8\_First\_Eight\_Bit\_Num=1111 1110;

[0034] 第二个脉冲到来,u8\_First\_Eight\_Bit\_Num=1111 1100;

[0035] 第三个脉冲到来,u8\_First\_Eight\_Bit\_Num=1111 1000;

[0036] 第四个脉冲到来,  $u8\_First\_Eight\_Bit\_Num=1111\ 0000$ ;

[0037] 第五个脉冲到来,  $u8\_First\_Eight\_Bit\_Num=1110\ 0000$ ;

[0038] 第六个脉冲到来,  $u8\_First\_Eight\_Bit\_Num=1100\ 0000$ ;

[0039] 第七个脉冲到来,  $u8\_First\_Eight\_Bit\_Num=1000\ 0000$ ;

[0040] 第八个脉冲到来,  $u8\_First\_Eight\_Bit\_Num=0000\ 0000$ ;

[0041] 4. 当位于第一扇区的变量  $u8\_First\_Eight\_Bit\_Num=00000000$  时, 则触发第二扇区的变量  $u8\_Second\_Eight\_Bit\_Num=1111\ 1110$ , 以此类推, 当  $u8\_Second\_Eight\_Bit\_Num=0000\ 0000$  时, 共计数  $8*8=64$  个脉冲。

[0042] 以此总结: 仅仅使用两个FALSH扇区, 每个扇区仅占用1个字节空间的情况下, 使得FLASH使用寿命扩展到了  $8*8=64$  倍, 即计数总量扩展为  $64*100万=6400$  万次。那么如果两个扇区, 每个扇区预留32bit的变量, 则使用寿命扩展到了  $32*32=1024$  倍, 即FLASH计数总量扩展为  $1024*100万=102400$  万次。

[0043] 同理, 根据不同FALSH型号和扇区多少, 依据软件的依托, 理论上可以实现嵌入式产品生命周期内FALSH擦写次数的全部覆盖。

[0044] 本具体实施方式的实施例均为本发明的较佳实施例, 并非依此限制本发明的保护范围, 故: 凡依本发明的结构、形状、原理所做的等效变化, 均应涵盖于本发明的保护范围之内。

脉冲	u8_Second_Eight_Bit_Num								u8_First_Eight_Bit_Num							
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
1	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

图1