

1 概述

用户在使用 EEPROM 偶尔会出现数据丢失的情况，尤其在上下电或电源抖动过程中，会出现不确定的干扰。为了防止故障发生，建议用户按规定条件使用，避免在异常条件下操作。

关于数据丢失的原因，需根据实际使用情况来分析。

2 注意事项

在具体应用中，请注意以下事项：

- 1) VCC 电源确定是在 1.8V~5.5V 范围内，VCC 脚附近加一个 0.1uF 电容。
- 2) SCL 和 SDA 外部上拉，上拉电阻范围 1K~10K。
- 3) 若应用板上干扰较大，在 SCL 和 SDA 线上串几十欧姆的电阻来去毛刺。
- 4) 若要在 SCL 和 SDA 线加到地电容，确认应用板上的 IIC 总线电容 $\leq 400\mu\text{F}$ 。
- 5) 注意 SDA、SCL、VCC、GND 的 PCB 走线布局，若 PCB 板 IIC 总线长度超过 10cm，那么走线排列最好是：



或者在 SDA 和 SCL 走线周围用 GND 包围。

- 6) WP 接系统高复位输出，在系统复位时保护数据不被改写。
- 7) 上电后稍作延时，待电压稳定时对 EEPROM 操作。
- 8) 写入 EEPROM 后，采用应答查询方式判断是否写入结束。
- 9) 必要时，对写入数据后做一次读出校验。
- 10) 做好数据的冗余备份，发生局部数据丢失后可执行恢复。

3 应答查询流程

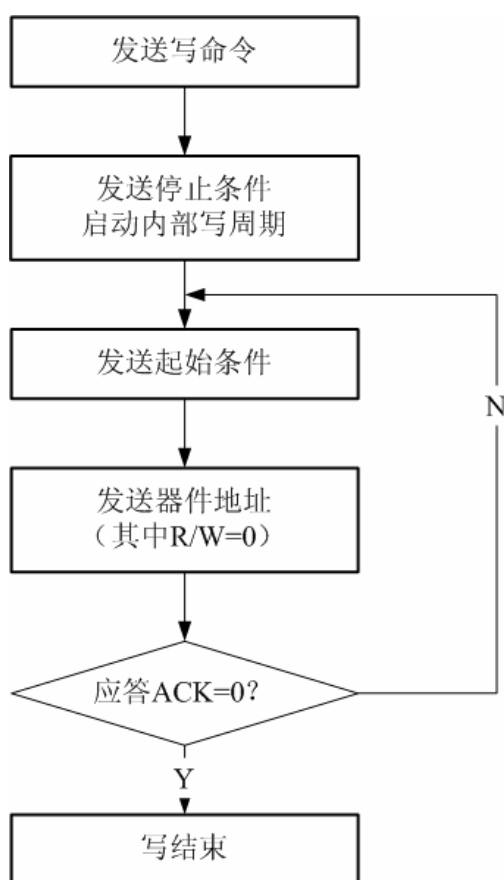
虽然写数据后做软件延时，等待数据写入，然后再操作 EEPROM，也不影响功能。但是，为了更安全地使用 EEPROM，仍推荐采用写数据后应答查询方式。

这种方式的好处是：

- 1) 通过应答查询判断是否写入结束，可以保证数据写入，并防止后续写出错。
- 2) 可以节省软件的等待时间。
- 3) 易于兼容和替换同类型的芯片而无需关心延时。

应答查询流程如下所示：





以 CW24C32/64 为例的程序：

```

#define ACK          0      // 应答位
#define WRONG       1
#define SUCCESS     0
#define E2_I2C_SLAVE_ADR 0xA0 // EEPROM 地址
#define E2_BUF_SIZE 8
  
```

```

unsigned char E2BufIn[E2_BUF_SIZE]; // 开辟缓冲区
unsigned int SubAddr;                // 子地址
  
```

// 写 EEPROM 测试函数

```
void E2Test(void)
```

```
{
```

```
    unsigned char Count;        // 字节计数
```

```
    .....
```

```
    Count = 5;
```



August, 2008

Rev1.0

www.chipwinner.com

24 系列 EEPROM 应用注意事项

```
// 写 EEPROM
E2BufIn[0] = E2_I2C_SLAVE_ADR; // 器件地址
E2BufIn[1] = SubAddr >> 8; // 子地址高字节
E2BufIn[2] = SubAddr & 0xFF; // 子地址低字节
E2BufIn[3] = 0xAA; // 第 1 个字节内容 0xAA
E2BufIn[4] = 0xBB; // 第 2 个字节内容 0xBB
if (I2CMasterDataTx(Count, E2BufIn) == WRONG)
{
    PutString(" Write Error! "); // 写失败
}
PutString(" Write OK! "); // 写成功
.....
}

// I2C 发送数据子函数
unsigned char I2CMasterDataTx(unsigned char n, unsigned char *buffer)
{
    unsigned char i;

    // 写命令（器件地址和数据）
    I2C_Start(); // 发送起始条件
    for (i=0; i<n; i++)
    {
        if(I2C_Send_Byte(buffer[i]) != ACK) // 发送数据
            return WRONG; // 非 ACK，返回写出错
    }
    I2C_Stop(); // 发送停止条件

    // 应答查询，判是否写完成
    I2C_Start(); // 发送起始条件
    while (I2C_Send_Byte(buffer[0]) != ACK) // 发送数据（器件地址）
        ; // 非 ACK，内部写周期，等待
    I2C_Stop(); // 发送停止条件
    return SUCCESS; // 写结束
}
```

