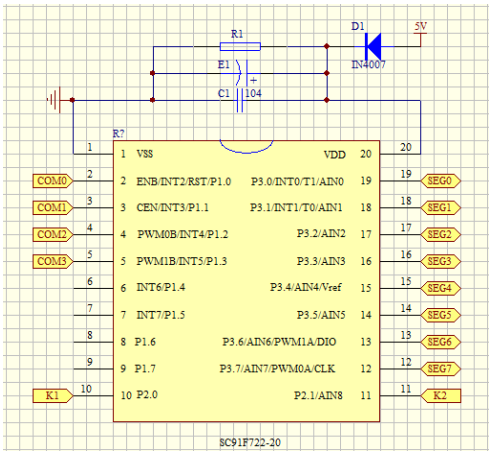
目录

赛元 SC91F722/721/733/832/831 等 MCU 内部集成的 256/512Byte SRAM，具有掉电时 RAM 数据可以保存的功能：即在无法满足 IC 正常工作电压的一定范围内，RAM 仍然可以保存着上一次 IC 正常工作掉电前的数据。因此非常适合电压力锅等产品使用。

1.应用原理图



RAM 掉电保存所需的元器件：1 个 IN4007 二极管 D1、1 个放电电阻 R1、1 个电解电容 E1、1 个滤波电容 C1

2.实现原理

放电时间：

t = RC*Ln[（V1-V0）/（V1-Vt）]
= RC*Ln[（0-0.7）/（0-0.4）]
= RC*Ln（7/4）
= 0.5596 RC

说明： V0 为电容上的初始电压值
V1 为电容可充到或放到的电压值
Vt 为 t 时刻电容上的电压值

RAM 不可保存电压约在 0.4V 左右，电容电压从 5V 掉到 0.7V 的时间很快，掉电可保存的时间主要是电容放电的电压值从 0.7V 掉到 0.4V 左右所耗的时间。如果把 R=20M,C=220uF,代入公式得 t=2462 秒=41 分。此理论时间跟实际可保存时间会有一定的偏差，实际可保存时间可能会稍有偏短，主要与产品所处环境的湿度、温度有关。

由于 VCC 的电压掉到 0.4V 左右，可能会有部分 RAM 优先不可保存，为了防止在这个临界点上电时因部分 RAM 不可保存而导致程序出错，以 C 语言为例，建议按以下做法处理，以保证程序可靠：

A： 建立 keil 工程时，请加载 STARTUP.A51 文件，并在此文件里修改
IDATALEN EQU 00H
目的：上电时 IC 不对片内的 RAM 清零。

STARTUP.A51

```
;  at processor reset can be defined:
;
; <o> IDATALEN: IDATA memory size <0x0-0x100>
;     <i> Note: The absolute start-address of IDATA m
;     <i>       The IDATA space overlaps physically t
IDATALEN        EQU     00H
```

B： 假设用户需要保存的原始数据区为 20 个 Byte，假设定义为：ram1~ ram20
并定义 20Byte 的用户原始数据备份区假设为：
ram 1_backup~ram20_backup；
目的：用于校验，保证用户原始数据可靠。

C： 用户不需要保存的数据，请在定义时全部初始化数据；
目的：防止在 RAM 可保存的电压临界点部分 RAM 不可保存，导致程序出错。

D： 上电时首先检验原始数据区与备份区是否相同，相同则继续执行下一步程序；不相同则初始化原始数据区，并同时修改备份区数据，使原始数据区的内容与备份区的数据区的相同；
目的：保证用户原始数据可靠。

E： 主程序如果要修改原始数据区的某个 RAM，假设修改 ram1 的值为 50；则在修改 ram1=50 的下一条语句，修改 ram1_backup=ram1。
目的：保持两个区域的数据一致性。

3.举例

程序功能实现：上电首先检验原始要保存的数据是否与备份区数据相同，不相同则初始化原始和备份区数据，相同则继续往下执行；通过按键修改 count 的值，并用 4 位数据码显示 count 的值，修改原始保存的数据的同时修改备份区数据，以实现 RAM 掉电保存功能。

```
#include"SC91F722_C.H"
//显示的 IO
#define   com0      P10
#define   com1      P11
#define   com2      P12
#define   com3      P13
#define   seg0      P30
#define   seg1      P31
#define   seg2      P32
#define   seg3      P33
#define   seg4      P34
#define   seg5      P35
#define   seg6      P36
#define   seg7      P37

//按键的 IO
#define   key_add   P20
#define   key_sub   P21

//按键值
#define   KEY_ADD   1
#define   KEY_SUB   2

//定义 CODE 型数据，用于显示
const unsigned char code table[]=
{0x3F,0x06,0x5b,0x4F,0x66,0x6d,0x7d,0x07,0x7f,0x6f,0x77,0x7c,0x39,0x5E,0x79,0x71};

//定义不需要掉电保存的数据，定义时初始化数据
unsigned char com_n=0,key_number=0,key_number_old=0,keybounce=0;
unsigned char TusCounter=0;
bit sys_500us=0,key_flag=0;
unsigned char ge=0,shi=0,bai=0,qian=0;

//定义要掉电保存的原始数据，不初始化数据
unsigned int count;

//定义备分区数据，不初始化数据
unsigned int idata count_backup;
//ram_verify():校验原始数据与备份数据，相同则继续进行，不相同则初始化原始数据和备份数据；
void ram_verify()
{
    if(count==count_backup) ;
```

```
    else                                        //校验不相同时，初始化保存的数据
    {
            count=0;count_backup=count;
    }
}
//display():4COM 数码管显示
void display()
{
    qian=count/1000;
    bai=count%1000/100;
    shi=count%100/10;
    ge=count%10;
    switch(com_n)
    {
            case 0:com3=1;com2=1;com1=1;com0=1;P3=table[qian];com3=0;break;
            case 1:com3=1;com2=1;com1=1;com0=1;P3=table[bai]; com2=0;break;
            case 2:com3=1;com2=1;com1=1;com0=1;P3=table[shi]; com1=0;break;
            case 3:com3=1;com2=1;com1=1;com0=1;P3=table[ge];  com0=0;break;
            default:com_n=0;break;
    }
    com_n++;
    if(com_n>3)
    com_n = 0;
}
//keyscan():按键按下，修改 count，紧接着同时修改备份 count_backup;
void keyscan()
{
    key_number = 0;
    if(!key_add) key_number = KEY_ADD;
    if(!key_sub) key_number = KEY_SUB;
    if(key_number)
    {
            if(key_number == key_number_old)
            {
                    if(!key_flag)
                    {
                            if(++keybounce>20)
                            {
                                    keybounce = 0;
                                    key_flag=1;
                                    if(key_number == KEY_ADD)
                                    {
                                            count++;
                                            if(count>=10000)
                                                    count=0;
                                            count_backup=count;
                                            //修改原始数据，紧接着就同时修改备份数据
                                    }
                                    if(key_number == KEY_SUB)
                                    {
                                            if(count==0) count=9999;
                                            else     count--;
                                            count_backup=count;
                                            //修改原始数据，紧接着就同时修改备份数据
                                    }
                            }
                    }
```

```
            }
            else
            {
                keybounce = 0;
                key_number_old = key_number;
                if(key_flag) key_flag=0;
            }
    }
    else
    {
            keybounce = 0;
            key_number_old = 0;
    }
}
void timer0_init(void)
{
    TMCON=0x01;                         //Fosc/4
    TMOD=0x02;                          //模式 2，8 位自动装载 16M,50us
    TL0=(256-200);                      //低位值
    TH0=(256-200);                      //高位值
    TR0=0;                              //禁止定时器 0 计数
    ET0=1;                              //允许定时器 0 中断
    TR0=1;                              //允许定时器 0 计数
}
void timer0()interrupt 1                 //定时器 0 中断服务程序
{
    TusCounter++;
    if(TusCounter>=10)                  //500us;
    {
            TusCounter=0;
            sys_500us=1;                //每 500us 置 1
    }
}
void main()
{
    RSTCFG=0X24;                        // 设置 P10 为普通 IO 功能，设置 LVR=2.45V
    P1CFG0=0X55;                        //设置 IO 模式
    P1CFG1=0X55;
    P2CFG0=0X00;
    P3CFG0=0X55;
    P3CFG1=0X55;
    ram_verify();                       //上电校验原始保存的数据与备份区数据是否相同
    timer0_init();                      //timer0 初始化
    EA=1;                               //打开总中断
    while(1)
    {
            if(sys_500us)
            {
                sys_500us=0;
                keyscan();              //扫描按键
                display();              //显示
            }
    }
}
```