SC8P171XE 应用手册

增强型8位CMOS单片机

Rev. 1.00

目录

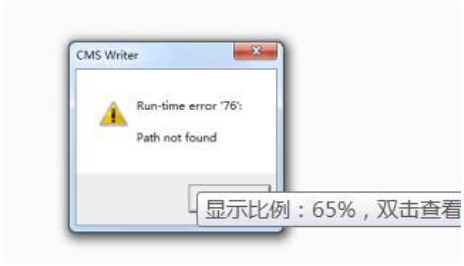
1. 产品概述

1.1 烧录器使用问题

1.1.1 打开文件

烧录软件必须打开.SCX文件，不能打开.BIN文件；

1.1.2 安装问题



或

如果打开绿色版本后弹出上面的报警信息，请安装一个安装版本，然后再打开 WWW.SCMCU.COM 上下载的绿色版本使用。

1.1.3 连接问题

芯片按照说明书的管脚定义连接到烧录器小板的对应标识口线上

1.1.4 电源问题

烧录器必须接 20V-DC 电源，9V-DC 电源会导致芯片不能烧写。

1.2 在线仿真

1.2.1 连接问题

按照上图所示连接仿真小板及仿真器，用线将对应管脚连接到实际电路中

1.2.2 使用问题

1. 仿真小板的PB4口不能仿真输入状态，芯片无此问题；
2. 仿真使用外部电源有较大的几率不能正常工作，建议使用内部电源仿真；
3. 目前可以用171XD的仿真小板仿真171XE，二者之间的区别参见下页；171XE的PWM2~4功能及内部基准做AD参考的功能，LVD功能暂时无法仿真

1.3 SC8P171XD 与 SC8P171XE 的使用区别

1.3.1 功能比较

功能模块	SC8P171xD	SC8P171xE
LVR	2.1/2.5/3.5	1.8/2.0/2.6/3.0
SLEEP_LVREN	只有 ENABLE	可选,选择 DISABLE 的话休眠电流较小
LVD 模块	无	有，可选 2.2V/2.4V/2.7V/3.0V/3.3V/3.7V/4.0V/4.3V
WDT 溢出时间	无分频时为 18ms，不同芯片偏差较大	无分频时为 16ms
AD 模块	参考电压为 VDD；	参考电压可选 2V,2.4V 和 VDD 转换时间可选，通过 ADCON1 的 BIT3 可选快速时钟 程序兼容 171XD 需 ADCON1 除 Bit7 外均清零
PWM 模块	2 路，PWM0/PWM1；	5 路，PWM0-PWM4， PWM0-PWM3 共周期，PWM4 独立周期
休眠唤醒等待时间	内振模式：等待 1024 个时钟； 低速模式：等待 8 个时钟。	内振模式：等待 128 个时钟； 低速模式：等待 4 个时钟。
低速 RC 振荡	精度较差，常温下 VDD=5V:32KHz； VDD=3V:16KHz。	精度较好，常温下，VDD=2V~5V:32KHz；
输出口灌电流典型值	35mA@5V	50mA@5V
振荡周期	8M	8M/16M
寄存器		新增： PWMD2L,PWMD3L,PWMD4L,PWMD23H,PWM4TL,PWMCON1,LVDCON,PIE2,PIR2 修改： ADCON1

1.3.2 程序兼容方面

如果考虑程序兼容，需要重点关注以下内容：

序号	内容
1	ADC 的操作，当 ADCON1 寄存器的 BIT3~BIT0=0 时，则与 SC8P171XD 兼容；
2	内部低速振荡 32KHz，SC8P171XD 的 32K 振荡电压特性较差，不同的电源电压下的 WDT 溢出时间，SC8P171XD 和 SC8P171XE 区别较大，需要注意；
3	在 SC8P171XD 中操作 PWM 的高低占空比寄存器没有先后顺序要求，在 SC8P171XE 中,操作 PWM0,PWM1 的高低占空比寄存器也没有先后顺序要求，但操作 PWM2-PWM4 时要求先写高位占空比寄存器，再写低八位寄存器。

2. 中央处理器（CPU）

2.1 中断向量

所有中断向量地址为0004H。一旦有中断响应，程序计数器PC的当前值就会存入堆栈缓存器并跳转到0004H开始执行中断服务程序。

中断服务程序有一个特殊的定义方法:void interrupt ISR(void);

其中的函数名“ISR”可以改成任意合法的字母或数字组合，但其入口参数和返回参数类型必须是“void”型，亦即没有入口参数和返回参数，且中间必须有一个关键词“interrupt”。中断函数可以被放置在源程序的任意位置。因为已有关键词“interrupt”声明，SCMCU C在最后进行代码连接时会自动将其定位到0x0004中断入口处，实现中断服务响应。编译器也会自动实现中断函数的返回指令“retfie”。

2.2 程序状态寄存器（STATUS）

程序状态寄存器 STATUS(03H)

03H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	----	----	RP0	TO	PD	Z	DC	C
读写	----	----	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	----	0	1	1	X	X	X

建议不用 TO 和 PD 标志位判断芯片复位的原因；

2.3 预分频器（OPTION_REG）

预分频器 OPTION_REG(81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	1	1	1	1	0	1	1

必须注意 OPTION_REG 在 WDT 和 TIMER0 间切换时需要按照一定的顺序处理，参见下列程序：

```
asm("clrwdt");
TMR0 = 0;
OPTION_REG |= 0x0F;          //预分频器数值设为最大
OPTION_REG = 0X0A;           //需要的预分频数值给看门狗 16ms*4=64ms
asm("clrwdt");
```

2.4 看门狗计数器（WDT）

在所有复位后，WDT 溢出周期 144ms，假如你需要改变的 WDT 周期，可以设置 OPTION_REG 寄存器。WDT 的最低溢出周期为 16mS。

看门狗定时器控制寄存器 WDTCON(88H)

105H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	----	----	----	----	----	----	----	SWDTEN
R/W	----	----	----	----	----	----	----	R/W
复位值	----	----	----	----	----	----	----	0

注：如果 CONFIG 中 WDT 配置位为 1，则 WDT 始终被使能，而与 SWDTEN 控制位的状态无关。如果 CONFIG 中 WDT 配置位为 0，则可以使用 SWDTEN 控制位使能或禁止 WDT。

WDT 溢出周期受电压影响较大，不建议用 WDT 做定时

3. 系统时钟

3.1 振荡器控制寄存器

振荡器控制寄存器 OSCCON(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	---
R/W	---	R/W	R/W	R/W	---	---	---	---
复位值	---	1	0	1	---	---	---	---
注：F_{OSC} 为内部振荡器频率 8MHz；F_{SYS} 为系统工作频率。								

当 IRCF(2~0)= 000　FSYS = 32kHz（LFINTOSC）,此时频率受电压影响较大，不建议用该频率做有计时要求的场合；

4. 休眠模式

4.1 注意事项

1.芯片在执行STOP指令后进入休眠模式，汇编指令为STOP，C编译时语句为asm(“SLEEP”)，在C中写STOP有可能不编译。

2.在休眠模式下，为了尽量降低电流消耗，所有I/O引脚都应该保持为VDD或GND，没有外部电路从I/O引脚消耗电流。为了避免输入引脚悬空而引入开关电流，应在外部将高阻输入的I/O引脚拉为高电平或低电平。为了将电流消耗降至最低，还应考虑芯片内部上拉电阻的影响。

3.在休眠模式下，需要将打开的可能造成电流消耗的模块关闭，比如AD、中断等。

4.当MCU从休眠态被唤醒时，需要等待一个振荡稳定时间（Reset Time），这个时间标称值为1024/Fosc。为了减少电流消耗，可在这个时候将振荡切换到31KHz上。

5.休眠唤醒从下一条指令继续运行。如果GIE位被置1（允许），器件执行STOP指令之后的指令，然后跳转到中断地址 （0004h）处执行代码。如果不想执行STOP指令之后的指令，用户应该在STOP指令后面放置一条NOP指令。

6.休眠前可以把WDT关闭，芯片唤醒后先清WDT计数器后再打开WDT。建议除休眠外，运行其余程序均要打开WDT；必须注意如果需要软件开关WDT，必须在芯片配置中将WDT设为DISABLE.

7.休眠时如果有口线做输入，口线输入电压尽量接近VDD或地，否则有可能会导致芯片休眠电流加大；如果做AD输入用，ADCON0必须清零，但对应的IO口需要通过ANSEL,ANSELH设为模拟输入通道

8.如果用到PORTA,PORTB口唤醒，在执行休眠指令前，需要读取PORTA,PORTB的状态；用PORTB唤醒，必需将PORTB中断允许位（RBIE)置1，口线对应的IOCB位也要置1.如果不使用中断可以将总中断标志位GIE清零；

9.用PORTA唤醒，必需将PORTA中断允许位（RAIE)置1，口线对应的IOCA位也要置1.此外外设中断允许位（PEIE）必须打开；

10.如果使用电平中断唤醒，建议在中断程序中加入读端口状态程序，避免程序反复进入中断程序；

11.看门狗把芯片从休眠唤醒，程序会从休眠指令的下一条指令继续运行而不会复位。看门狗的溢出时间受电压影响较大，不建议用作计时场合；

4.2 范例程序

```
INTCON = 0;
OPTION_REG = 0;
TRISA = 0B00000000;          //关闭所有输出
PORTA = 0B00000000;
WPUA   = 0B00000000;
TRISB = 0B00001000;
PORTB = 0B00000000;
WPUB = 0B00001000;

IOCB = 0B00001000;           //允许 RB3 的 IO 口电平变化中断
RBIE = 1;                    //允许 PORTB 电平变化中断
GIE = 1;      //GIE = 0 时，唤醒后执行 SLEEP 后程序;GIE = 1 时，唤醒后跳至中断服务

ADCON0 = 0;                  //关闭所有模块
PWMCON = 0;
OSCCON = 0X70;               //配置振荡为 8M,
SWDTEN = 0;                  //关闭 WDT
PORTB;                       //读 PORTB 值并锁存
```

```
asm("clrwdt");
asm("sleep");                    //进入休眠模式
asm("nop");
Init_System();
```

5. I/O 端口

5.1 I/O 口使用注意事项

1.若在 I/O 口线串入较长的连接线，请在靠近芯片 I/O 的地方加上限流电阻以增强 MCU 抗 EMC 能力。
2.如选择 IO 做数字 I/O 输入，必须将 ANSEL,ANSELH 对应的位清零；
3.上电将口线变为输出态时，建议先赋值需要的输出电平，然后改为输出口，再赋值输出电平，可以避免口线输出不需要的脉冲；

5.2 范例 程序

```
INTCON = 0;
OPTION_REG = 0;
ANSEL = 0;                  //PORTA口做数字I/O
ANSELH = 0;                 //PORTB口做数字I/O
PORTA = 0B00000000;         //先赋值输出电平
TRISA = 0B00000000;         //关闭所有输出
PORTA = 0B00000000;
WPUA  = 0B00000000;

PORTB = 0B00000000;
TRISB = 0B00001000;         //RB3做输入，其余口线做输出
PORTB = 0B00000000;
WPUB = 0B00001000;          //RB3打开上拉电阻
```

6. 中断

6.1 中断处理范例程序

```
void interrupt ISR(void)                    //中断服务程序，所有中断程序的入口
{
  if (T0IE && T0IF)                         //判 TMR0 中断
  {
      T0IF = 0;   //清除 TMR0 中断标志      //在此加入 TMR0 中断服务
  }
  if (TMR1IE && TMR1IF)                     //判 TMR1 中断
  {
      TMR1IF = 0; //清除 TMR1 中断标志      //在此加入 TMR1 中断服务
  }
}  //中断结束并返回芯片
```

6.2 外部中断

6.2.1 外部中断使用注意事项

INT 外部中断相关寄存器包括：OPTION_REG、INTCON。

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	1	1	1	1	0	1	1

Bit6 INTEDG: 触发中断的边沿选择位。
1= INT 引脚上升沿触发中断。
0= INT 引脚下降沿触发中断。

6.2.2 外部中断范例程序

```
void main()
{
    asm("nop");
    asm("clrwdt");

    OSCCON = 0x70;          //8MHZ,内部振荡器用作系统时钟
    OPTION_REG = 0;
    ANSEL = 0;
    TRISA = 0x20;           //RA5 设为输入
    WPUA = 0x20;            //RA5 开启上拉电阻
//--------------------------------------
    INTEDG = 0;             //INT 引脚下降沿触发
    INTCON = 0x90;          //允许所有未被屏蔽的中断、禁止外设中断，使能 INT 外部中断
//--------------------------------------
```

```
    while(1)
    {
        asm("clrwdt");
    }
}

/***********************************************
函数名称：INT_Isr
函数功能：INT 中断服务
入口参数：无
出口参数：无
备注：
***********************************************/
void interrupt INT_Isr()
{
    if(INTF)
    {
        INTF = 0;                //清中断标志

    }
}
```

7. 定时计数器 TIMER0

7.1 使用注意事项

1.用作定时/计数器时其相关寄存器包括 TMR0、OPTION_REG；当使用 TIMER0 中断时，还涉及寄存器 INTCON，在使用时需要操作相关的位。

2.若要让 TIMER0 获得 1:1 的预分频比，可将寄存器 OPTION_REG 中的 PSA 位置 1，将分频器分配给 WDT。

3.使用 TIMER0 中断时，当 TMR0 溢出产生中断时，进入中断服务并产生复位重新计数，在重新赋初值前 TMR0 已有计数，故重新赋初值时应补偿该计数值。

4.请注意 TIMER0 时钟输入使用的是 4 分频的系统时钟频率。

5.将预分频器分配给 TIMER0 或 WDT 后，在切换预分频比时可能会产生无意的器件复位。要将预分频器从分配给 TIMER0 改为分配给 WDT 模块时或从 WDT 到 TIMER0 都需要按照一定指令顺序，

7.2 范例程序

```
#include <SC.h>

void main(void)
{
        asm("nop");
        asm("clrwdt");
        OSCCON = 0X70;                  //8M
        TRISA = 0;
        TRISB = 0;                      //PORTB 口作输出口
        OPTION_REG = 0B00000000; //Timer0 使用内部时钟，预分频为 1:2
        TMR0 = 6;                       //设定 Timer 初始值，计时时间为（256-6）*4*2/8M=250uS
        T0IF = 0;
        T0IE = 1;                       //使能 Timer0 溢出中断
        GIE = 1;                        //开启总中断
        while(1)
        {
            asm("clrwdt");
        }
}

void interrupt Timer0_Isr(void)
{
        if(T0IF)
        {
            TMR0 += 6;              //TMR0 不能自动赋值，+6 是避免计时误差
            T0IF = 0;
```

```
            PORTB ^= 0XFF;
        }
    }
```

8. 定时计数器 TIMER2

8.1 使用注意事项

1.用作定时/计数器时其相关寄存器包括 PR2、TMR2、T2CON；当使用 TIMER2 中断时，还涉及寄存器 INTCON、PIE1、PIR1，在使用时需要操作相关的位。

2.请注意 TIMER2 时钟输入使用的是 4 分频的系统时钟频率。

3.TIMER2 溢出标志位，时间控制不仅存在预分频还存在后分频。

8.2 范例程序

```
#include <sc.h>
void main(void)
{
    asm("nop");
    asm("clrwdt");
    OSCCON = 0X70;              //8M
    TRISA = 0;
    TRISB = 0;                  //PORTB 口作输出口

    PR2 = 250;                  //设定 Timer 初始值，定时周期是 250*4/8M=125uS
    TMR2IF = 0;
    TMR2IE = 1;                 //使能 Timer2 溢出中断
    T2CON = 0B00000100;         //开启 Timer2,设置 TMR2 的分频比为 1:1
    INTCON = 0XC0;              //开启总中断
    while(1)
    {
        asm("clrwdt");
    }
}

void interrupt Timer2_Isr(void)
{
    if(TMR2IF)
    {
        TMR2IF = 0;
        PORTB ^= 0XFF;
    }
}
```

9. 模数转换（ADC）

9.1 ADC 注意事项

1.ADC 模块的相关寄存器包括：控制寄存器 ADCON0、ADCON1 和数据寄存器 ADRESH、ADRESL。以及模拟选择寄存器 ANSEL,ANSELH.

2.选择较慢的时钟频率有利于提高 AD 的精度。在 8M 主频时不能选用 Fsys/8 的分频；

3.开启模块指令（ADON 置 1）不能够与开始模数转换指令（GO/DONE 置 1）同时进行，即不能够在同一指令中将上述两个位置 1。

4.切换 AD 通道后，需延时 2Tad 个时钟才能开始模数转换指令；

5.选择左对齐时，AD 结果为 12 位；选择右对齐时，AD 结果为 10 位。

6.具有 AD 功能的 IO 口输入高于 VCC 或负压时，会导致 AD 检测不准。所以避免类似电阻降压过零引入具有 AD 模块的管脚。

7.AD 模块中通道 15 输出是固定电压 1.2V 对应的 AD。测试精度±1.5%。一般用来倒推芯片 VDD 电压。

8. 如果AD模块采用内部基准做AD参考，在AD参考电压从外部VDD转为内部基准参考的时候必须延时100US以上；从内部基准转为外部VDD做ADC参考，必须延时10US以上；

9. 如果最低工作电压低于2.5V时使用AD模块，AD的转换时钟必须选择FSYS/128或更慢时钟，否则，AD精度在低压下会受较大影响；

9.2 范例程序

```
#include <sc.h>

volatile unsigned int    adresult;
volatile unsigned long adsum;
volatile unsigned int admin,admax;
volatile unsigned char adtimes;

void ADC_Sample(unsigned char adch);
void DelayXms(unsigned char x);

#define _DEBUG              //调试程序用

/************************************************************
函数名称：AD_Sample
函数功能：AD 检测
入口参数：adch - 检测通道
出口参数：无
备      注：采样通道需自行设置为模拟口
          采样 10 次,取中间八次的平均值为采样结果存于 adresult 中
************************************************************/
void ADC_Sample(unsigned char adch)
{
    volatile unsigned int ad_temp;
```

```
ADCON1 = 0;                          //左对齐,AD 值取 12 位
ADCON0 = 0X41 | (adch << 2);         //16 分频，8M 主频必须选 16 分频或以上
asm("nop");
asm("nop");
asm("nop");
asm("nop");
GODONE = 1;                          //开始转换

unsigned char i = 0;
while(GODONE)
{
    if(0 == (--i))
        return;
}

ad_temp=(ADRESH<<4)+(ADRESL>>4);     //计算 12 位 AD 值

if(0 == admax)
{
    admax = ad_temp;
    admin = ad_temp;
}
else if(ad_temp > admax)
    admax = ad_temp;                 //AD 采样最大值
else if(ad_temp < admin)
    admin = ad_temp;                 //AD 采样最小值

adsum += ad_temp;
if(++adtimes >= 10)
{
    adsum -= admax;
    if(adsum >= admin)    adsum -= admin;
        else adsum = 0;

    adresult = adsum >> 3;           //8 次平均值作为最终结果

    adsum = 0;
    admin = 0;
    admax = 0;
    adtimes = 0;
}
```

```
}


//ADC 单次采样
unsigned char ADC_Result(unsigned char adch)
{
    ADCON1 = 0;                         //左对齐
    ADCON0 = 0X41 | (adch << 2);        //16 分频
    asm("nop");
    asm("nop");
    asm("nop");
    asm("nop");
    GODONE = 1;                         //开始转换

    unsigned char i = 0;
    while(GODONE)
    {
        if(0 == (--i))
            return 0;                   //转换超时
    }
    return ADRESH;
}
/************************************************************
函数名称：DelayXms
函数功能：毫秒级非精准延时
入口参数：x - 延时时间
出口参数：
备    注：
************************************************************/
void DelayXms(unsigned char x)
{
    unsigned char i,j;
    for(i=x;i>0;i--)
        for(j=153;j>0;j--);
}

/************************************************************
main 主函数
************************************************************/
void main()
{
    asm("nop");
```

```
    asm("clrwdt");
    INTCON = 0;                 //禁止中断
    OSCCON = 0X70;                  //配置振荡为 8M

    while(1)
    {
        asm("clrwdt");
        DelayXms(1);

#ifdef _DEBUG
        ANSELH = 0X20;      //RB5 选择模拟输入
        TRISB = 0X20;       //RB5 设为输入
        ADC_Sample(13);

        unsigned char result;

        result = ADC_Result(15);        //测试内部基准源
        if (result > 50) RB4 = 0;
            else RB4 =1;
#endif
    }
}
```

10. PWM 模块

10.1 注意事项

1.PWM 模式相关寄存器包括：控制寄存器 PWMCON、PWM 周期寄存器 PWMTL,PWMTH、PWM 占空比寄存器 PWMD0L,PWMD1L,PWMD01H。

2.PWM 模式下 PWMx 引脚应配置为输出。

3.PWM 模式下，注意合理设置脉冲宽度及周期值。当脉冲宽度大于周期值时，输出引脚将保持不变。

4.PWM 频率越高，这样分辨率位数也会越小。

5.设置多个 PWM 占空比时，需确保原先配置完成的 PWM 输出使能位不变；

6.PWM 的占空比设置为 0 的时候仍会有脉冲，如要输出低电平，需关闭 PWMxEN，对应口线直接输出低电平；

7.PWM 的最低输出脉宽是 1/Fosc；

8.可通过 CONFIG 配置选择 PWM 输出的口线；

- RB1&RB0　　PWM1=RB1，PWM0=RB0
- RB1&RB3　　PWM1=RB1，PWM0=RB3
- RB4&RB3　　PWM1=RB4，PWM0=RB3
- RA2&RA1　　PWM1=RA2，PWM0=RA1
- RB6&RA6　　PWM1=RB6，PWM0=RA6
- RB6&RB5　　PWM1=RB6，PWM0=RB5

9.PWM2,PWM3,PWM4 的口线管脚确定，在 RA3~RA5 口；

10.PWM4 的周期设置必须先设置周期高位，在设置周期低位，否则有可能 PWM 周期不是设置的周期；

11.PWM2,PWM3,PWM4 的占空比设置必须先设置占空比高位，再设置占空比低位，否则有可能 PWM 占空比不是设置的占空比；

12.PWM0,PWM1 的周期和占空比不用按照顺序修改；

10.2 范例程序

```
#include <SC.h>                    //调用单片机的头文件

void main(void)                    //主函数,单片机开机后就是从这个函数开始运行
{
    unsigned int delaycnt = 0;
    unsigned char pwmbuf = 0;

    asm("nop");
    asm("clrwdt");
    OSCCON = 0X70;                 //8M

    TRISA = 0B00000110;            //选择的 RA1,RA2 口设为输入
                                   //以下是对 PWM 功能初始化
    PWMCON = 0B00000100;           //PWM 选择时钟源为 Fosc/1,PWM0 反向
    PWMTL = 0XFF;
    PWMTH = 01;                    //周期选择为 1FF，则周期为（511+1）*1/8M=64uS，周期的时钟分
```

频在 PWMCON 选择

```
    PWMD0L = 0X1F;          //脉宽为（31+1）*1/8M=4uS,则占空比为 1/16
    PWMD1L = 0X7F;          //脉宽为（127+1）*1/8M=16uS,则占空比为 1/4
    PWMD01H = 0X00;         //配置 PWM1,PWM2 的占空比，该值不能超过周期，否者为 100%输出

    PWMIF = 0;                  //清零 PWMIF 中断标志位
    PWMCON = 0B00000111;        //PWM 选择时钟源为 Fosc/1,PWM0 反向，允许 PWM0,PWM1 使能
    TRISA = 0B00000000;         //选择的 PA1,PA2 口设为输出

    while(1)                    //死循环,单片机初始化后,将一直运行这个死循环
    {
        asm("clrwdt");
        if(++delaycnt > 1000)  //延时 1000 个软件周期，非精确定时
        {
            delaycnt = 0;           //清零以备下次重新计数
            pwmbuf++;               //占空比加一
            PWMD0L = pwmbuf;   //写入占空比
        }
    }
}
```