KF8F312 DEMO

程序使用说明
（第二版）

上海芯旺微电子有限公司
2015.10

目录

实验一、I/O 口作为输入输出实验 5
1.1 程序声明 5
1.2 硬件说明 5
1.3 程序设计说明 6
1.4.1 C 程序代码 6
1.4.2 汇编程序代码 8
实验二、P0 口电平变化中断实验 12
2.1 程序声明 12
2.2 硬件说明 12
2.3 程序设计说明 12
2.4.1 C 程序代码 13
2.4.2 汇编程序代码 16
实验三、数码管进行 0~9999 的计数实验 22
3.1 程序声明 22
3.2 硬件说明 23
3.3 程序设计说明 24
3.4.1 C 程序代码 25
3.4.2 汇编程序代码 28
实验四、T0 定时器中断模式实验 34
4.1 程序声明 34
4.2 硬件说明 34
4.3 程序设计说明 34
4.4.1 C 程序代码 35
4.4.2 汇编程序代码 38
实验五、T0 计数器中断模式实验 45
5.1 程序声明 45
5.2 硬件说明 45
5.3 程序设计说明 46
5.4.1 C 程序代码 47
5.4.2 汇编程序代码 49
实验六　T1 定时器门控位启动实验 56
6.1 程序声明 56
6.2 硬件说明 56
6.3 程序设计说明 56
6.4.1 C 程序代码 57
6.4.2 汇编程序代码 60
实验七、T2 定时中断模式实验 66
7.1 程序声明 66
7.2 硬件说明 66
7.3 程序设计说明 67
7.4.1 C 程序代码 67
7.4.2 汇编程序代码 70

实验八、ADC 转换采样实验 77
8.1 程序声明 77
8.2 硬件说明 77
8.3 程序设计说明 79
8.4.1 C 程序代码 81
8.4.2 汇编程序代码 85
实验九、T2 启动 AD 转换采样 93
9.1 程序声明 93
9.2 硬件说明 93
9.3 程序设计说明 93
9.4.1 C 程序代码 94
9.4.2 汇编程序代码 98
10.1 程序声明 106
10.2 硬件说明 106
10.3 程序设计说明 107
10.4 .1 C 程序代码 108
10.4.2 汇编程序代码 111
实验十一、PWM3 单输出模式实验 116
11.1 程序声明 116
11.2 硬件说明 117
11.3 程序设计说明 117
11.4.1 C 程序代码 121
11.4.2 汇编程序代码 123
实验十二、PWM3 带死区半桥输出模式实验 128
12.1 程序声明 128
12.2 硬件说明 128
12.3 程序设计说明 128
12.4.1 C 程序代码 130
12.4.2 汇编程序代码 132
实验十三、PWM3 全桥输出模式实验 135
13.1 程序声明 135
13.2 硬件说明 135
13.3 程序设计说明 136
13.4.1 C 程序代码 139
13.4.2 汇编程序代码 141
实验十四、模拟比较器模块实验 146
14.1 程序声明 146
14.2 硬件说明 146
14.3 程序设计说明 147
14.4.1 C 程序代码 148
14.4.2 汇编程序代码 149
实验十五、系统时钟输出和内部参考电压输出 153
15.1 程序声明 153
15.2 硬件说明 153

15.3 程序设计说明 153
15.4.1 C 程序代码 155
15.4.2 汇编程序代码 156
实验十六、串口通信(全双工异步模式)实验 160
16.1 程序声明 160
16.2 硬件说明 160
16.3 程序设计说明 161
16.4.1 C 程序代码 167
16.4.2 汇编程序代码 171
实验十七、看门狗唤醒休眠实验 178
17.1 程序声明 178
17.2 硬件说明 178
17.3 程序设计说明 179
17.4.1 C 程序代码 181
17.4.2 汇编程序代码 182
实验十八、 读写 BLOCK EEPROM 实验 185
18.1 程序声明 185
18.2 硬件说明 185
18.3 程序设计说明 186
18.4.1 C 程序代码 188
18.4.2 汇编程序代码 198
实验十九、INT1 中断实验 208
19.1 程序声明 208
19.2 硬件说明 209
19.3 程序设计说明 209
19.4.1 C 程序代码 209
19.4.2 汇编程序代码 211

实验一、I/O口作为输入输出实验

1.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：I/O口作为输入输出实验

项 目 名：01-Input_Output_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：I/O作为输入输出控制功能时，当有按键按下，则改变相应LED显示状态。S3控制LED1，S4控制LED2。

1.2 硬件说明

D3发光二极管的正极接地，负极接KF8F312单片机的P16端口，所以要点亮D3发光二极管，只需配置P16端口为输出口，并让其输出低电平即可。连接按键的端口配置为输入状态，当检测到按键按下时分别改变相应LED的显示状态。 J13、J14要跳上跳线帽，如图。

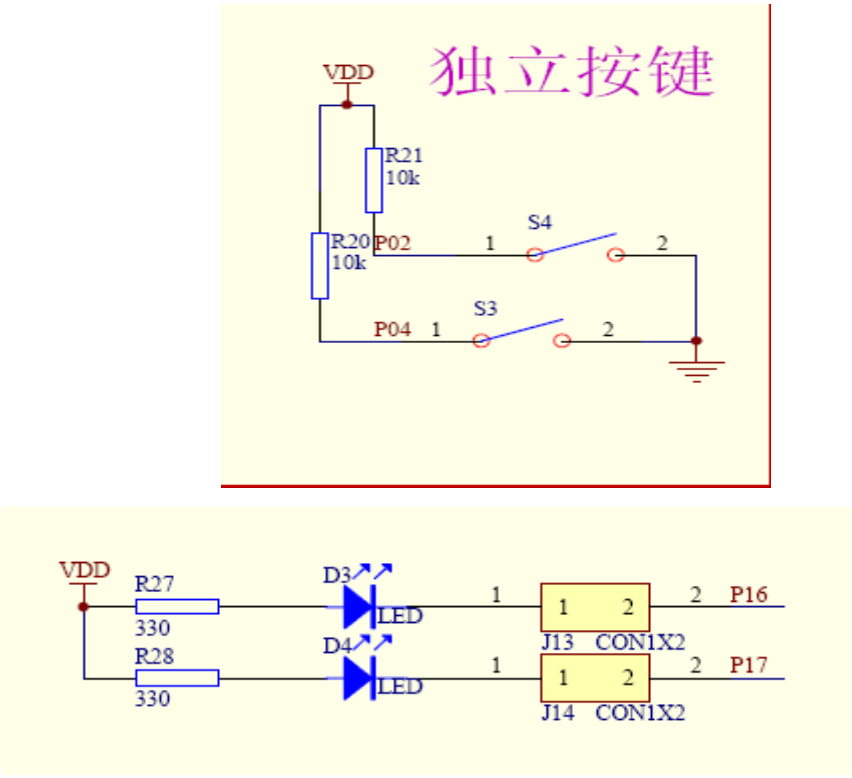


图1.1：LED模块原理图

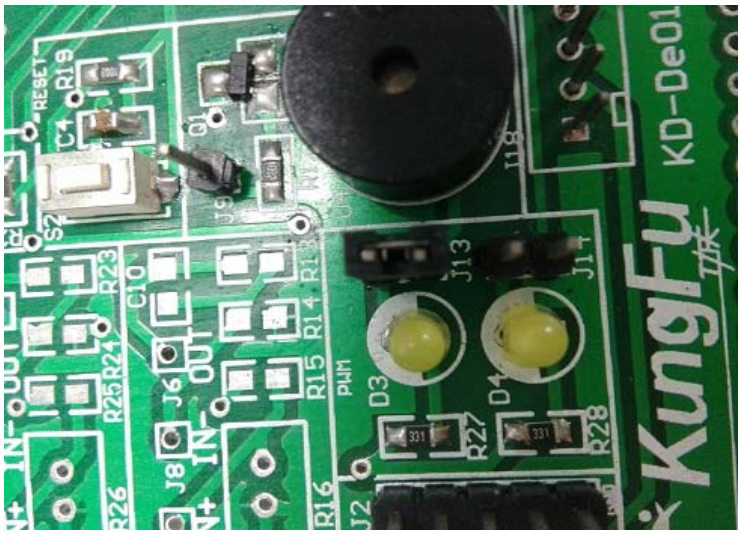


图1.2：跳线帽接线图

1.3 程序设计说明

程序设计的重点为把单片机的P16口配置为输出口，然后让其输出低电平。根据芯片手册，把方向控制寄存器TR1的相应位写入0，即可把对应的IO口配置为输出口， 把状态寄存器P1的相应位置0，即可让其输出低电平。

1.4. 1 C程序代码

```
#include<KF8F312.h>

/*********************宏定义*******************************/
#define uchar unsigned char
#define uint  unsigned int
#define LED1 P16                                        // 对应Demo板上的D3
#define LED2 P17                                        // 对应Demo板上的D4
#define S3     P04
#define S4  P02
/*********************宏定义结束***************************/

/************************
 * 函数名     : init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun()
{
    OSCCTL = 0x60;                          //设置系统时钟为8M
```

```
    /*********端口初始化***********/
    TR0 = 0x1c;                                 //设置P03端口只能设置为输入,按
键接口P02、P04需配置为输入口
    TR1 = 0x00;                                 //设置P1端口为输出
    TR2 = 0x00;                                 //设置P2端口为输出

    P0 = 0;
    P1 = 0;
    P2 = 0XF0;                                  //关闭数码管位选端
}
/************************
 * 函数名     : Delay_ms
 * 函数功能：长时间延时
 * 入口参数：延时时长
 * 返回         : 无
 ************************/
void Delay_ms(uchar ms_data)
{
    uchar i;
    while(ms_data--)
    {
        i = 200;                // 根据晶振的选择设定合适的值
        while(i--);
    }
}

/************************
 * 函数名     : main
 * 函数功能：主函数
 * 入口参数：
 * 返回         :
 ************************/
void main()
{
    Init_fun();                 // 端口初始化

    while (1)
    {
        if (!S3)
        {
            Delay_ms(10);       // 消抖延时
            if (!S3)
            {
                LED1 = !LED1;   // LED1状态取反
```

```
                while (!S3);    // 等待按键释放
            }
        }
        if (!S4)
        {
            Delay_ms(10);       // 消抖延时
            if (!S4)
            {
                LED2 = !LED2;   // LED2状态取反
                while (!S4);    // 等待按键释放
            }
        }
    }
}

/************************
 * 函数名     : int_fun
 * 函数功能: __interrupt
 * 入口参数:
 * 返回         :
 ************************/
void int_fun() __interrupt
{

}
```

1.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

DELAY_NUM0              .EQU          0x80      ;延时用的临时变量
DELAY_NUM1              .EQU          0x81      ;延时用的临时变量
PSW_TEMP            .EQU          0X85      ;PSW的临时保存变量
PCH_TEMP            .EQU          0X86      ;PCH的临时保存变量

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT
INTERRUPT
;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
     SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
```

```
    MOVPCH_TEMP, R1
;/*********************************中断处理程序
**********************************/



;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET

MAIN
    CALL INIT_ALL                ;初始化寄存器
    CALL INIT_RAM                ;初始化RAM

LOOP                             ; S3
    JNB P0,P04
    JMP LOOP3
    CALL DELAY_MS                ; 消抖延时
    JNB P0,P04
    JMP LOOP3
    MOV R0,#0X40                 ; LED1状态取反
    XOR P1,R0
LOOP2
    JB P0,P04                    ; 等待按键释放
    JMP LOOP2

LOOP3                            ; S4
    JNB P0,P02
    JMP LOOP
    CALL DELAY_MS                ; 消抖延时
    JNB P0,P02
    JMP LOOP
    MOV R0,#0X80                 ; LED1状态取反
    XOR P1,R0
LOOP4
    JB P0,P02                    ; 等待按键释放
    JMP LOOP4
```

```
    JMP LOOP
    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0x1c         ;配置按键P02 P04 IO口未输入模式
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00         ;设置P2端口为输出,数码管位选设置为输入，关闭
数码管显示
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0XF0         ;关闭数码管位选端
    MOV P2,R0
    CRET

;/*************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 根据数据使用情况，默认RAM数值
;* 入口参数: 无
```

```
;* 返    回: 无
;*******************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作


    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET
;*******************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*******************************************************************
*****************
DELAY_MS
    MOV    R0 ,#0xC8
    MOV    DELAY_NUM0, R0
LP0
    CWDT
    MOV    R0 ,#0XC8
    MOV    DELAY_NUM1, R0
LP1
```

```
DECJZ  DELAY_NUM1       ;DELAY_NUM2自减1，若为0，则跳过
JMP    LP1
DECJZ  DELAY_NUM0       ;DELAY_NUM1自减1，若为0，则跳过
JMP    LP0
CRET
.END
```

实验二、P0口电平变化中断实验

2.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：P0口电平变化中断实验

项 目 名：02-IO_Change_Interrupt_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：配置P02口电平变化中断，在中断处理函数中改变LED的亮灭状态，使得LED闪烁。

2.2 硬件说明

使用杜邦线将单片机的P04口连接到单片机的P02口，改变P04口的输出电平，使得P02口的电平发生变化，从而产生中断，在中断处理程序中改变LED的亮灭状态，从而使得LED闪烁，接线图如图2.1所示。

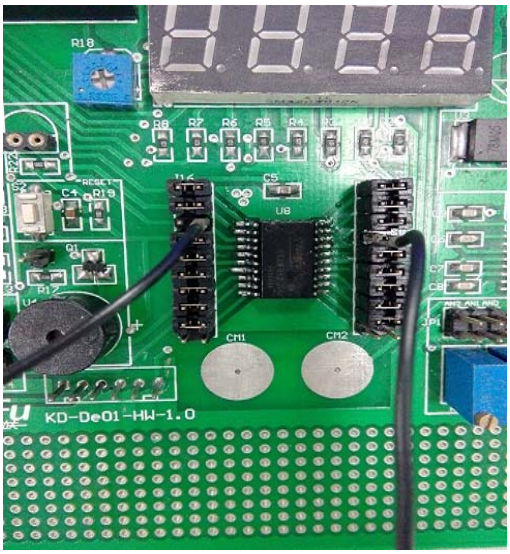


图2.1　P02和P04连接图

2.3 程序设计说明

P0口每个引脚都具有电平变化中断功能，当引脚的当前电平与上次读P0寄存器时的电平不匹配时将产生电平变化中断。IOCL为电平变化中断控制寄存器，将IOCL某位置1将开启对应引脚的电平变化中断功能，如果该引脚电平发生变化，不管电平变化中断是否使能，电平变化中断标志位(P0IF)都会置1，如果全局中断使能位(AIE)和电平变化中断使能位(P0IE)都已置1，则会响应中断进入中断服务子程序。除此之外，还需注意一下两点：

1、 使用P0口电平中断功能，首先将相应引脚设置为数字输入模式，如果将某引脚设置为输出或者设置为模拟输入口时将会自动禁止该引脚的电平变化中断功能。

2、 P0口各引脚的电平变化中断共用一个中断使能位P0IE和一个中断响应标志位P0IF，所以同一时间只能判断一个端口的电平变化中断功能。

3、 使用电平变化中断功能，当中断产生并进入中断处理函数时，必须读取当前P0口电平，如在中断处理函数中加上 MOV P0。

4、 除电平变化中断功能外，P0口还带有内部上拉功能（除P03口外），可通过上拉功能控制寄存器PUR和OPTR寄存器中的PUPH来配置上拉功能是否打开。同时只有将引脚设置为数字输入口时才可开启上拉电阻功能。例如,使能P02口的上拉功能需作如下配置

```
ANS2 = 0;     // 寄存器 ANSEL 配置为数字口
TRO2 = 1;     // 寄存器TRO 配置为输入口
PUPH = 0;     //  寄存器OPTR 使能上拉功能总使能位
PUR2 = 1;     // 寄存器PUR 使能P02上拉功能
```

2.4.1 C 程序代码

```
#include<kf8f312.h>
/*********************宏定义*******************************/
#define uchar unsigned char
#define uint  unsigned int

/*********************宏定义结束***************************/
/*********************全局变量*****************************/
uchar const Arr_num[] = {0X3F,
                         0X06,
                         0X5B,
                         0X4F,
```

```
                    0X66,
                    0X6D,
                    0X7D,
                    0X07,
                    0X7F,
                    0X6F
};                                          //共阴接法
uchar Unit, Decade, Hundred, Thousand;      //定义个十百千位
/*********************全局变量结束************************/
/******************函数声明*******************************/
void Init_fun();
void Display();
void Delay_200us();
/******************函数声明结束****************************/

/************************
 * 函数名     : init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun()
{
    OSCCTL = 0x60;                          //设置为8M
    /*********端口初始化***********/
    TR0 = 0x08;                               //设置P03端口为输入
    TR1 = 0x00;                               //设置P1端口为输出
    TR2 = 0x00;                               //设置P2端口为输出
    P0 = 0;
    P1 = 0;
    P2 = 0xF0;                              //关闭数码管位选端
}

/************************
 * 函数名     : display
 * 函数功能：数码管显示
 * 入口参数：显示数字
 * 返回         : 无
 ************************/
void Display()
 {

    uint i;                                 // 控制显示时长，确定自增的速度
    for(i = 0; i < 500; i++)
```

```
    {
        P2 = 0XE0;                          //打开数码管的 个位
        P1 = Arr_num[Unit];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影

        P2 = 0XD0;                          //打开数码管的 十位
        P1 = Arr_num[Decade];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影

        P2 = 0XB0;                          //打开数码管的 百位
        P1 = Arr_num[Hundred];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影

        P2 = 0X70;                          //打开数码管的 千位
        P1 = Arr_num[Thousand];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影
    }
}

/************************
 * 函数名     : Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         : 无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}
/************************
 * 函数名     : main
 * 函数功能：程序入口主函数
 * 入口参数：无
 * 返回         : 无
 ************************/
```

```c
void main()
{
    Unit = Decade = Hundred = Thousand = 0;
    Init_fun();
    while(1)
    {
        Unit++;                             //个位加1
        if(Unit == 10)                       //如果个位的值加到10，那么个位归零，
十位加1，以下以此类推
        {
            Unit = 0;
            Decade++;
            if(Decade == 10)
            {
                Decade = 0;
                Hundred++;
                if(Hundred == 10)
                {
                    Hundred = 0;
                    Thousand++;
                    if(Thousand == 10)
                    {
                        Thousand = 0;
                    }
                }
            }
        }
        Display();                          //显示函数调用
    }
}

//中断函数
void int_fun() __interrupt
{

}
```

2.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"
;;;;;;;;;;;;;;;;通用寄存器;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DELAY_NUM1      .EQU   0X81       ;延时用的临时变量
DELAY_NUM2      .EQU   0X82       ;延时用的临时变量
PSW_TEMP     .EQU   0X85       ;PSW的临时保存变量
PCH_TEMP     .EQU   0X86       ;PCH的临时保存变量
```

```
UNIT          .EQU   0X88       ;数码管个位
DECADE        .EQU   0X89       ;数码管十位
HUNDRED          .EQU   0X8A       ;数码管百位
THOUSAND      .EQU   0X8B       ;数码管千位
DIS_DEL          .EQU   0X8C       ;DIS_DEL是用来延时数值的递增用的，
数值越大，那么递增的速度就越慢
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    .ORG 0X0000
     NOP
     JMP MAIN                    ;入口
    .ORG 0X0004
     JMP INTERRUPT

;*****************************************************************
*****************
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返     回: R0，数码管显示的编码值
;* 注   意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不
同页造成查表出错
;*****************************************************************
*****************
DIG_DATA
    ADD PCL,R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
    RRET R0 ,#0X7D      ;6
    RRET R0 ,#0X07      ;7
    RRET R0 ,#0X7F      ;8
    RRET R0 ,#0X6F      ;9
    RRET R0 ,#0X77      ;A
    RRET R0 ,#0X7C      ;B
    RRET R0 ,#0X39      ;C
    RRET R0 ,#0X5E      ;D
    RRET R0 ,#0X79      ;E
    RRET R0 ,#0X71      ;F
    RRET R0 ,#0X00      ;空

;;;;;;;;;;;;;;;中断部分;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
INTERRUPT
;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1
    ;---------------------------------
    ;中断处理
    ;...
    ;...
    ;...
    ;---------------------------------
;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;;;;;;;;;;;;;;中断部分结束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;;;;;;;;;;;;;;入口函数,主程序;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN
    CALL INIT_ALL          ;初始化寄存器
     CALL INIT_RAM           ;初始化RAM
LOOP
    INC UNIT             ;个位数字的递增
    MOV R0 ,UNIT
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR UNIT

    INC DECADE              ;十位数字的递增
    MOV R0 ,DECADE
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR DECADE
```

```
    INC HUNDRED             ;百位数字的递增
    MOV R0 ,HUNDRED
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR HUNDRED

    INC THOUSAND          ;千位数字的递增
    MOV R0 ,THOUSAND
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR THOUSAND

C_DIS
    MOV R0,#0x33
    MOV DIS_DEL,R0          ;DIS_DEL是用来延时数值的递增用的，数值越
大，那么递增的速度就越慢
    CALL LED_DISPLAY        ;数码管显示当前数值
    DECJZ DIS_DEL
    JMP $-2                 ;调整地址以当前地址减2的位置

    JMP LOOP

    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
```

```
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    ;***端口初始化********
    MOV R0 ,#0x08
    MOV TR0, R0          ;设置P0端口为输入
    MOV R0 ,#0x00
    MOV TR1, R0          ;设置P1端口为输出
    MOV R0 ,#0X00
    MOV TR2, R0          ;设置P2端口为输出

    CLR P0
    CLR P1
    MOV R0 ,#0XF0        ;关闭数码管位选端
    MOV P2, R0
    CRET

;*************************************************************
*****************
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位 ，
每个数的范围 0~f
;* 返    回: 无
;*************************************************************
*****************
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0          ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0          ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
```

```
    CLR P1          ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0         ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1          ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0         ;打开数码管的 千位
    MOV R2 , THOUSAND
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1          ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0
    CRET

;*************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数，时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
DELAY
    MOV R0,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
```

```
    CRET                ;子程序返回

;/*****************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X20    ;操作个数
    CALL    INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X01    ;操作个数
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .end
```

实验三、数码管进行0~9999的计数实验

3.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：数码管进行0~9999的计数实验

项 目 名：03-Number_count_TEST

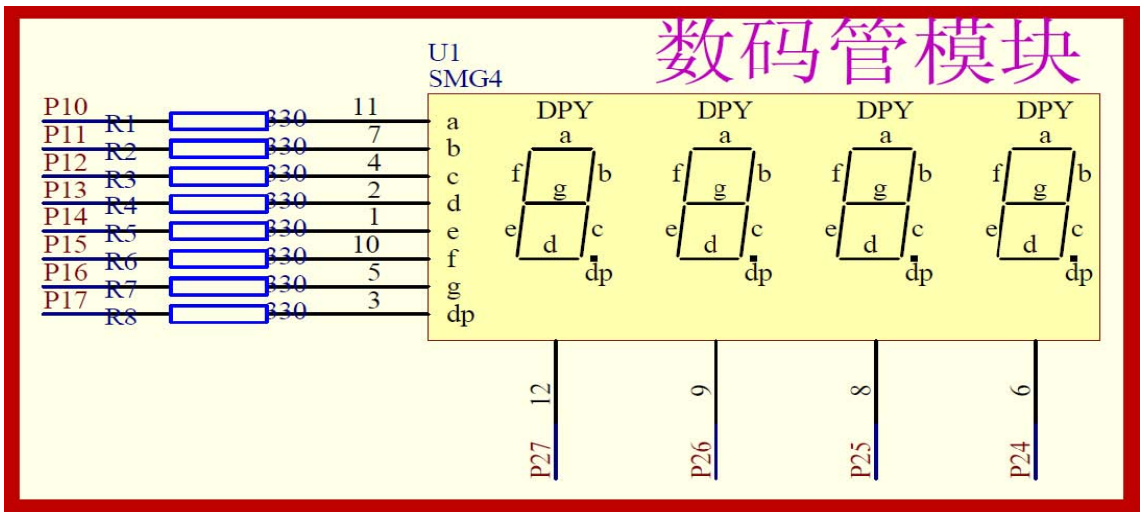
开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

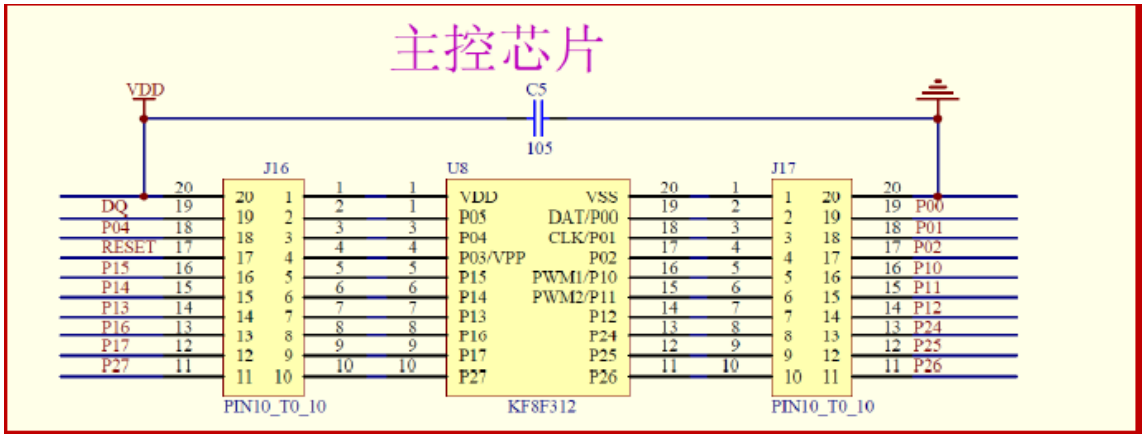
功能简述：数码管进行0~9999的计数，到达9999后，又从0开始。

3.2 硬件说明

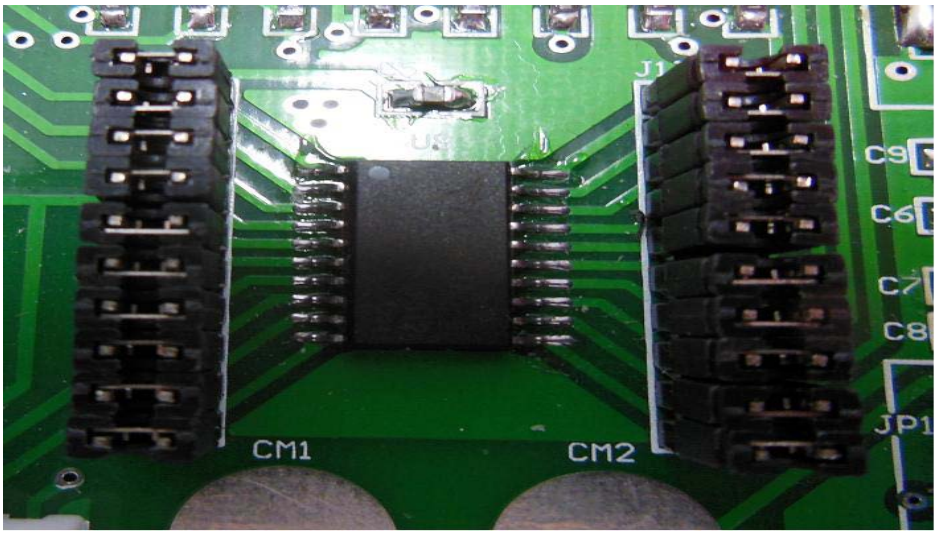
KF8F312单片机的P1端口P10-P17连接七段数码管的a,b,c,d,e,f,g,dp。P24-P27分别接显示位（个位，十位，百位，千位）的公共端，该数码管为共阴数码管,所以若要点亮数码管，首先将该位的显示公共端置低电平，然后将要显示的那一段置高电平。如：要在个位上显示0，需将P24置低电平，P10-P15置高电平，P16，P17置低电平。原理图如图1，图2所示。单片机的P10~P17跳上跳线,P24~P27跳上跳线，所需跳线如图3所示。



（图1：数码管模块原理图）



（图2：KF8F312单片机原理图）



（图3：需要连接的跳线）

3.3 程序设计说明

由于驱动数码管使用到了P1、P2端口，所以首先需将P1、P2端口方向设置为输出。由于个十百千位共用了P1端口，所以只能分时显示各位，先显示第一位，再显示第二位，依次循环，由于人眼的视觉停留，快速的切换显示位便无法察觉到。数值计数累加先加个位，如果个位的值加到10，那么个位归零，十位加1，以此类推。加到10000的时候将每一位清零。数码管显示流程图如图4所示，主函数运行流程图如图5所示。

综上，该程序主要包含以下几块：

① 器件初始化子函数

② 数码管显示子函数

③ 主函数（数值累加循环显示）

3.4.1 C程序代码

```
#include<kf8f312.h>
/*********************宏定义*******************************/
#define uchar unsigned char
#define uint  unsigned int

/*********************宏定义结束***************************/
/*********************全局变量*****************************/
uchar const Arr_num[] = {0X3F,
                         0X06,
                         0X5B,
                         0X4F,
                         0X66,
                         0X6D,
                         0X7D,
                         0X07,
                         0X7F,
                         0X6F
};                                          //共阴接法
uchar Unit, Decade, Hundred, Thousand;       //定义个十百千位
/*********************全局变量结束*************************/
/*******************函数声明*******************************/
void Init_fun();
void Display();
void Delay_200us();
/*******************函数声明结束***************************/

/************************
 * 函数名     : init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun()
{
    OSCCTL = 0x60;                          //设置为8M
    /*********端口初始化***********/
    TR0 = 0x08;                               //设置P03端口为输入
    TR1 = 0x00;                               //设置P1端口为输出
    TR2 = 0x00;                               //设置P2端口为输出
    P0 = 0;
```

```
    P1 = 0;
    P2 = 0xF0;                          //关闭数码管位选端
}

/************************
 * 函数名     : display
 * 函数功能：数码管显示
 * 入口参数：显示数字
 * 返回         : 无
 ************************/
void Display()
  {

    uint i;                             // 控制显示时长，确定自增的速度
    for(i = 0; i < 500; i++)
     {
        P2 = 0XE0;                      //打开数码管的 个位
        P1 = Arr_num[Unit];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                      //消影

        P2 = 0XD0;                      //打开数码管的 十位
        P1 = Arr_num[Decade];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                      //消影

        P2 = 0XB0;                      //打开数码管的 百位
        P1 = Arr_num[Hundred];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                      //消影

        P2 = 0X70;                      //打开数码管的 千位
        P1 = Arr_num[Thousand];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                      //消影
    }
}

/************************
 * 函数名     : Delay_200us
```

```
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         ：无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}
/************************
 * 函数名     ：main
 * 函数功能：程序入口主函数
 * 入口参数：无
 * 返回         ：无
 ************************/
void main()
{
    Unit = Decade = Hundred = Thousand = 0;
    Init_fun();
    while(1)
    {
        Unit++;                             //个位加1
        if(Unit == 10)                       //如果个位的值加到10，那么个位归零，
十位加1，以下以此类推
        {
            Unit = 0;
            Decade++;
            if(Decade == 10)
            {
                Decade = 0;
                Hundred++;
                if(Hundred == 10)
                {
                    Hundred = 0;
                    Thousand++;
                    if(Thousand == 10)
                    {
                        Thousand = 0;
                    }
                }
            }
        }
        Display();                          //显示函数调用
    }
```

```
}

//中断函数
void int_fun() __interrupt
{

}
```

3.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"
;;;;;;;;;;;;;;;;通用寄存器;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DELAY_NUM1      .EQU   0X81      ;延时用的临时变量
DELAY_NUM2      .EQU   0X82      ;延时用的临时变量
PSW_TEMP     .EQU   0X85      ;PSW的临时保存变量
PCH_TEMP     .EQU   0X86      ;PCH的临时保存变量
UNIT         .EQU   0X88      ;数码管个位
DECADE       .EQU   0X89      ;数码管十位
HUNDRED         .EQU   0X8A      ;数码管百位
THOUSAND     .EQU   0X8B      ;数码管千位
DIS_DEL         .EQU   0X8C      ;DIS_DEL是用来延时数值的递增用的，
数值越大，那么递增的速度就越慢
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    .ORG 0X0000
     NOP
     JMP MAIN                    ;入口
    .ORG 0X0004
     JMP INTERRUPT

;*******************************************************************
*****************
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返    回: R0，数码管显示的编码值
;* 注  意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不
同页造成查表出错
;*******************************************************************
*****************
DIG_DATA
    ADD PCL,R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
```

```
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
    RRET R0 ,#0X7D      ;6
    RRET R0 ,#0X07      ;7
    RRET R0 ,#0X7F      ;8
    RRET R0 ,#0X6F      ;9
    RRET R0 ,#0X77      ;A
    RRET R0 ,#0X7C      ;B
    RRET R0 ,#0X39      ;C
    RRET R0 ,#0X5E      ;D
    RRET R0 ,#0X79      ;E
    RRET R0 ,#0X71      ;F
    RRET R0 ,#0X00      ;空

;;;;;;;;;;;;;;;中断部分;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERRUPT
;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1
    ;--------------------------------
    ;中断处理
    ;...
    ;...
    ;...
    ;--------------------------------
;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;;;;;;;;;;;;;;中断部分结束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;;;;;;;;;;;;;;入口函数,主程序;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN
    CALL INIT_ALL          ;初始化寄存器
```

```
    CALL INIT_RAM           ;初始化RAM
LOOP
    INC UNIT            ;个位数字的递增
    MOV R0 ,UNIT
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR UNIT

    INC DECADE              ;十位数字的递增
    MOV R0 ,DECADE
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR DECADE

    INC HUNDRED             ;百位数字的递增
    MOV R0 ,HUNDRED
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR HUNDRED

    INC THOUSAND        ;千位数字的递增
    MOV R0 ,THOUSAND
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR THOUSAND

C_DIS
    MOV R0,#0x33
    MOV DIS_DEL,R0          ;DIS_DEL是用来延时数值的递增用的，数值越
大，那么递增的速度就越慢
    CALL LED_DISPLAY        ;数码管显示当前数值
    DECJZ DIS_DEL
    JMP $-2                 ;调整地址以当前地址减2的位置

    JMP LOOP

    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
```

```
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*******************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    ;***端口初始化********
    MOV R0 ,#0x08
    MOV TR0, R0          ;设置P0端口为输入
    MOV R0 ,#0x00
    MOV TR1, R0          ;设置P1端口为输出
    MOV R0 ,#0X00
    MOV TR2, R0          ;设置P2端口为输出

    CLR P0
    CLR P1
    MOV R0 ,#0XF0        ;关闭数码管位选端
    MOV P2, R0
    CRET

;*******************************************************************
*****************
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位  ,
每个数的范围 0~f
;* 返    回: 无
;*******************************************************************
*****************
LED_DISPLAY
    MOV R0 , #0XE0
```

```
    MOV P2 , R0          ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0          ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0          ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0          ;打开数码管的 千位
    MOV R2 , THOUSAND
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消隐
    MOV R0 ,#0XF0
    MOV P2 , R0
    CRET

;*************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数，时间为1ms
```

```
;* 入口参数: 无
;* 返    回: 无
;*******************************************************************
*****************
DELAY
    MOV R0,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                ;子程序返回

;/*****************************************************************
******************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*******************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X20    ;操作个数
    CALL    INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X01    ;操作个数
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
```

```
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .end
```

实验四、T0定时器中断模式实验

4.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：T0定时器中断模式实验

项 目 名：04-T0_Timer_Interrupt_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：T0定时中断模式的使用，使得数码管从0~9999计数，1秒增加1，当加到 9999时，又从0开始计数。

4.2 硬件说明

本实验运用到两个模块，一个是单片机内部功能模块，定时器0（T0），另一个是数码管显示模块。数码管具体特性在此之前已经讲解过，在此不作敷述，T0属于内部模块，不需进行接线操作。

4.3 程序设计说明

T0是一个8位的定时/计数器，当T0寄存器值加到255时，再加1，则会产生溢出，T0寄存器的值返回到0开始重新计数。

使用T0定时器，只需配置OPTR一个寄存器，其各个位如图4.1所示。在此重点讲解定时器0的分频器使用，例如我们采用8M频率，T0工作在定时模式下为每个机器周期加1，则定时器每次加1的时间是（1/8M）*4=0.5us。如果此时将预分频器分频比设置成64分频，那么定时器每次加1的时间就是0.5us*64=32us。如果我们需要一个5MS的定时，那么我们需要156个这样的计数。此时我们即可将T0寄存器初始值设为100，当T0数值加到256时正好是156个计数，这样便会产生一

个5MS的定时。200个5MS这样的定时就是1S的时间。当T0用于定时中断功能时，关于OPTR寄存器的说明如下：

1、 配置T0CS = 0，使其工作于定时模式。

2、 PS<2：0>位用来设置T0预分频器分频比，如需要进行64分频，则PS0 = 1，PS1 = 0，PS2 = 1。

其它位按照上电默认配置即可。

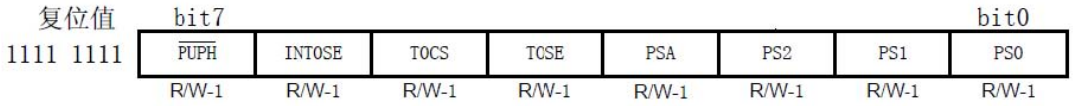


图4.1：OPTR选择寄存器

4.4 .1 C程序代码

```
#include<KF8F312.h>

/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int

/*********************宏定义结束*************************/

/*********************全局变量***************************/
uchar const Arr_num[] = {0X3F,
                         0X06,
                         0X5B,
                         0X4F,
                         0X66,
                         0X6D,
                         0X7D,
                         0X07,
                         0X7F,
                         0X6F
};                                      //共阴接法,数码管显示笔形码
uchar Unit, Decade, Hundred, Thousand;   //定义个十百千位
uchar time_1s;                          //定时计数器
/*********************全局变量结束***********************/


/***********************
```

```
 * 函数名     ：Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         ：无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}

/************************
 * 函数名     ：Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         ：无
 ************************/

void Init_fun(void)
{
    OSCCTL = 0x60;            // 设置为8M

    /*****端口初始化********/
    TR0 = 0x08;                 //设置P03端口为输入
    TR1 = 0x00;                 //设置P1端口为输出
    TR2 = 0x00;                 //设置P2端口为输出

    P0 = 0;
    P1 = 0;
    P2 = 0;

    /****初始化定时器0*****/
    OPTR = 0x05;             //定时器0,分频为1:64，现在晶振为8M，一条指令为0.5us（1/(8M/4)=0.5us ），则经过32us，计数1
    T0 = 0x64;               //设定定时5MS,  0X64=100D  即计数156后为256溢出产生中断，而156*32=4992 约为5000us=5ms
    INTCTL = 0XA0;            //使能总中断AIE、使能T0中断T0IE、清T0中断标志位T0IF
}

void main()
{
    uchar i;
    Unit = 0;
```

```
    Decade = 0;
    Hundred = 0;
    Thousand = 0;
    time_1s = 0;
    Init_fun();                    //初始化
    while(1)
    {
      for(i = 0; i < 100; i++)
      {
       P2 = 0XE0;            //打开数码管的 个位
       P1 = Arr_num[Unit];
       Delay_200us();
       P1 = 0;
       P2 = 0XF0;            //消影

       P2 = 0XD0;            //打开数码管的 十位
       P1 = Arr_num[Decade];
       Delay_200us();
       P1 = 0;
       P2 = 0XF0;            //消影

       P2 = 0XB0;            //打开数码管的 百位
       P1 = Arr_num[Hundred];
       Delay_200us();
       P1 = 0;
       P2 = 0XF0;            //消影

       P2 = 0X70;            //打开数码管的 千位
       P1 = Arr_num[Thousand];
       Delay_200us();
       P1 = 0;
       P2 = 0XF0;            //消影
      }
    }
}
//中断函数
void int_fun() __interrupt
{
    if(T0IF)
    {
       T0IF = 0;                          //清零中断标志
       T0 = 0x64;                         //T0重新赋值
       if(time_1s++ >= 200)                //判断是否达到1s记时   5ms*200
= 1s
```

```
        {
            time_1s = 0;                    //清零计数器
            Unit++;                         //个位相加
            if(Unit == 10)
            {
                Unit = 0;
                Decade++;                   //十位相加
                if(Decade == 10)
                {
                    Decade = 0;
                    Hundred++;              //百位相加
                    if(Hundred == 10)
                    {
                        Hundred = 0;
                        Thousand++;             //千位相加
                        if(Thousand == 10)
                        {
                            Thousand = 0;
                        }
                    }
                }
            }
        }
    }
}
```

4.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"
;;;;;;;;;;;;;;;;通用寄存器;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DELAY_NUM1      .EQU   0X81      ;延时用的临时变量
DELAY_NUM2      .EQU   0X82      ;延时用的临时变量
PSW_TEMP     .EQU   0X83      ;PSW的临时保存变量
PCH_TEMP     .EQU   0X84      ;PCH的临时保存变量
UNIT         .EQU   0X85      ;数码管个位
DECADE       .EQU   0X86      ;数码管十位
HUNDRED         .EQU   0X87      ;数码管百位
THOUSAND     .EQU   0X88      ;数码管千位
TIME_1S         .EQU   0X89      ;累计计数1s
FLAG_1S         .EQU   0X8A      ;计数1s的标志位

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    .ORG 0X0000
     NOP
     JMP MAIN                   ;入口
```

```
    .ORG 0X0004
    JMP INTERRUPT

;*****************************************************************************
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返      回: R0，数码管显示的编码值
;* 注   意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不同页造成查表出错
;*****************************************************************************
DIG_DATA
    ADD PCL, R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
    RRET R0 ,#0X7D      ;6
    RRET R0 ,#0X07      ;7
    RRET R0 ,#0X7F      ;8
    RRET R0 ,#0X6F      ;9
    RRET R0 ,#0X77      ;A
    RRET R0 ,#0X7C      ;B
    RRET R0 ,#0X39      ;C
    RRET R0 ,#0X5E      ;D
    RRET R0 ,#0X79      ;E
    RRET R0 ,#0X71      ;F
    RRET R0 ,#0X00      ;空

;;;;;;;;;;;;;;;中断部分;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERRUPT
;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1
    ;---------------------------------
    ;中断处理
```

```
    JB INTCTL,T0IF
    JMP INT_END
    CLR INTCTL,T0IF

    MOV R1 ,#0X64
    MOV T0, R1                    ;设定定时5MS
    INC TIME_1S
    MOV R1 ,#0XC8                 ;累计 计数200个5ms 即1S
    XOR R1 ,TIME_1S
    JB PSW,Z
    JMP INT_END
    CLR  TIME_1S
    MOV R1 ,#0X55
    MOV FLAG_1S, R1
    ;...
    ;...
    ;...
    ;---------------------------------
;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;;;;;;;;;;;;;;中断部分结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;;;;;;;;;;;;入口函数,主程
序;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN
    CALL INIT_ALL              ;初始化寄存器
    CALL INIT_RAM               ;初始化RAM

TIME_1S_LOOP
    CALL LED_DISPLAY           ;数码管显示当前数值
    MOV R0 ,#0X55
    XOR R0 ,FLAG_1S
    JB  PSW,Z
    JMP TIME_1S_LOOP
    ;; 时间增加操作
    CLR FLAG_1S
```

```
    INC UNIT              ;个位数字的递增
    MOV R0 ,UNIT
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR UNIT

    INC DECADE            ;十位数字的递增
    MOV R0 ,DECADE
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR DECADE

    INC HUNDRED           ;百位数字的递增
    MOV R0 ,HUNDRED
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR HUNDRED

    INC THOUSAND          ;千位数字的递增
    MOV R0 ,THOUSAND
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR THOUSAND

C_DIS
    JMP TIME_1S_LOOP
    CRET
;;;;;;;;;;;;;主程序结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
```

```
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0 ,#0xFF
    MOV TR0, R0          ;设置P0端口为输入
    MOV R0 ,#0x00
    MOV TR1, R0          ;设置P1端口为输出
    MOV R0 ,#0X0F
    MOV TR2, R0          ;设置P2端口的4,5,6,7为输出，0,1,2,3,为输入

    CLR P0
    CLR P1
    CLR P2

    ;初始化定时器0
    MOV R0 ,#0X84
    MOV OPTR, R0     ;定时器0,分频为1:32，现在晶振为4M，一条指令为1us（1/(4M/4)=1us ），则经过32us，计数1
    MOV R0 ,#0X64
    MOV T0, R0           ;设定定时5MS,  0X64=100D  即计数156后为256溢出产生中断，而156*32=4992 约为5000us=5ms
    MOV R0,#0Xa0
    MOV INTCTL,R0        ;使能总中断AIE、使能T0中断T0IE、清T0中断标志位T0IF
    CRET


;*******************************************************************
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位  ，每个数的范围 0~f
;* 返    回: 无
```

```
;*************************************************************
*****************
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0          ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0          ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0          ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0          ;打开数码管的 千位
    MOV R2 , THOUSAND
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0
    CRET
```

```
;*****************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
DELAY
    MOV R0,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                ;子程序返回

;/*****************************************************************
******************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作
```

```
    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .end
```

实验五、T0计数器中断模式实验

5.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：T0计数器中断模式实验

项 目 名：05-T0_Count_Interrupt_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：计数器0的使用，使用P04口输出一定个数高低电平到P02（T0计数器外部时钟引脚）做计数输入，可设置计数器为上升沿或者下降沿触发，并通过数码管显示脉冲累计个数。

5.2 硬件说明

本实验需要通过改变P04口的电平产生脉冲，使用杜邦线将P04和P02（T0计数器外部时钟引脚）端口进行连接，并使能T0计数器对P02端口的输入脉冲数进行计数，接线图如图5.1所示。

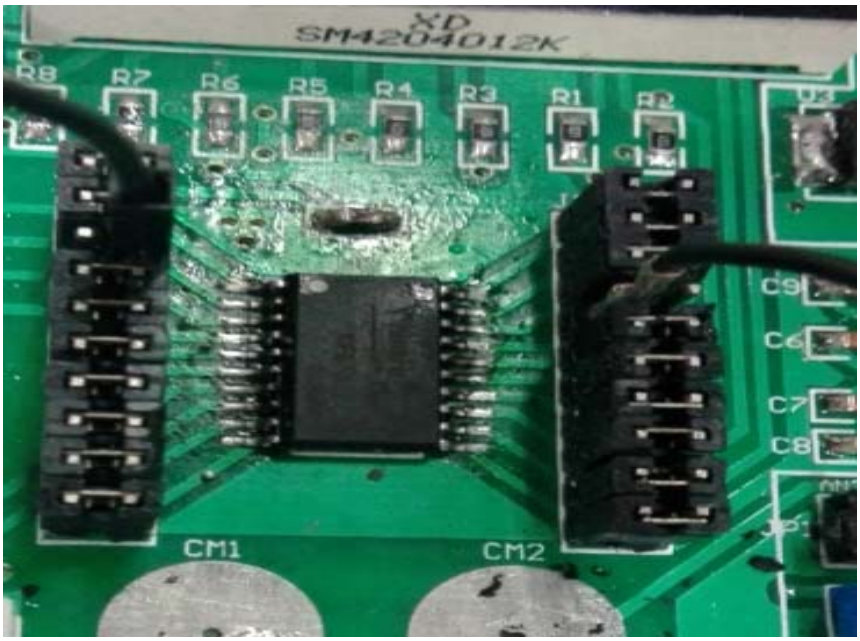


图5.1 P04和P02连接图

5.3 程序设计说明

单片机提供的T0是一个8位的定时/计数器，当T0寄存器值加到255时，再加1，则会产生溢出，T0寄存器的值返回到0开始重新计数。T0可以作为定时器使用，也可以作为计数器使用。两者的唯一区别是T0递增的条件不同。当作为定时器时，T0为一个机器周期加1，机器周期由系统时钟和T0预分频器决定（此时T0预分频器已经分配给T0使用）。当作为计数器时，T0模块在T0CK（P02）引脚信号的每一次上升沿(T0SE = 0)或下降沿(T0SE = 1)递增计数。所以定时器和计数器的本质都是计数器 ，计数器只是单纯的累计次数，而定时器则是在计数的基础上确定时间。

本实验中，通过P04口输出高低电平控制P02口电平， P02口电平每变化一次（可设置为上升沿或者下降沿触发）则计数器计数值加1，最终通过数码管显示累计脉冲数值。

使用T0计数器中断功能在配置时需要注意以下几点：

1、 P02口需设置为数字输入口。

2、 定时器模式和计数器模式使用同一个中断使能位（T0IE）和同一个中断标志位（T0IF）。

3、 关于预分频器，以下降沿触发为例，当设置预分频器用于WDT时（PSA = 1），P02口每一次下降沿T0累计加1，当设置预分频器用于T0且分频比为2分频时，

则P02每两次下降沿T0累计加1。

5.4 .1 C程序代码

```
#include<KF8F312.h>

/*********************宏定义*****************************/
#define uchar  unsigned char
#define uint   unsigned int
/*********************宏定义结束*************************/
/*********************全局变量*************************/
uchar const Arr_num[] = {0X3F,
                         0X06,
                         0X5B,
                         0X4F,
                         0X66,
                         0X6D,
                         0X7D,
                         0X07,
                         0X7F,
                         0X6F
};                                          //共阴接法
uchar Unit, Decade, Hundred, Thousand;       //定义个十百千位

// 宏定义需要计数脉冲个数,但必须是0-255的数据，8位计数器
#define Count_Timers    255
/************************
 * 函数名     : Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         : 无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}
/************************
 * 函数名     : Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun(void)
{
```

```
    OSCCTL = 0x60;              // 设置为8M

    /*****端口初始化********/
    TR0 = 0x0C;                   //设置P03端口为输入 P02作为T0计数器时钟输入口，需配置为输入
    TR1 = 0x00;                   //设置P1端口为输出
    TR2 = 0x00;                   //设置P2端口为输出

    P0 = 0;
    P1 = Arr_num[8];          //上电显示值
    P2 = 0x00;

    /****初始化定时器0*****/
    OPTR = 0x38;  //0x28|0x38  //允许P0口使用上拉功能，使能T0计数模式，下降沿触发  将预分频器用于WDT
    ANSEL = 0;                  //P02口配置为数字口
    T0 = 0x00;                 //设定计数初始值为0x00
}
/************************
 * 函数名     : display
 * 函数功能：数码管显示
 * 入口参数：显示数字
 * 返回         : 无
 ************************/
void Display()
  {

    uchar i;
    for(i = 0; i < 100; i++)
     {
        P2 = 0XE0;                          //打开数码管的 个位
        P1 = Arr_num[Unit];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影

        P2 = 0XD0;                          //打开数码管的 十位
        P1 = Arr_num[Decade];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影

        P2 = 0XB0;                          //打开数码管的 百位
        P1 = Arr_num[Hundred];
```

```
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影

        P2 = 0X70;                          //打开数码管的 千位
        P1 = Arr_num[Thousand];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;                          //消影
    }
}

void main()
{
    uint i;
    Init_fun();
    // 外部脉冲次数发送
    for(i=0;i<Count_Timers;i++)
    {
        P04 = 1;
        Delay_200us();
        P04 = 0;
        Delay_200us();
    }
    Unit=T0;
    Hundred=Unit/100;          //提取脉冲计数值百位
    Unit=Unit%100;
    Decade=Unit/10;           //提取脉冲计数值十位
    Unit=Unit%10;             //提取脉冲计数值个位

    Thousand=0;               //计数值最多255，所以千位赋值为0
    while(1)
    {
        Display();            //调用显示函数
    }

}
//中断函数
void int_fun() __interrupt
{

}
```

5.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"
.define      Count_Timers   0x0F ;设定最大15个，不做除法运算，可个位查表即可,按16进制写值

DELAY_NUM1      .EQU   0X81      ;延时用的临时变量
DELAY_NUM2      .EQU   0X82      ;延时用的临时变量
PSW_TEMP     .EQU   0X83      ;PSW的临时保存变量
PCH_TEMP     .EQU   0X84      ;PCH的临时保存变量

UNIT         .EQU   0X85      ;数码管个位
DECADE       .EQU   0X86      ;数码管十位
HUNDRED         .EQU   0X87      ;数码管百位
THOUSAND     .EQU   0X88      ;数码管千位
Count_Timer     .EQU   0X89      ;数码管千位

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT
;*****************************************************************
*****************
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返    回: R0，数码管显示的编码值
;* 注  意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不同页造成查表出错
;*****************************************************************
*****************
DIG_DATA
    ADD PCL, R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
    RRET R0 ,#0X7D      ;6
    RRET R0 ,#0X07      ;7
    RRET R0 ,#0X7F      ;8
    RRET R0 ,#0X6F      ;9
    RRET R0 ,#0X77      ;A
    RRET R0 ,#0X7C      ;B
```

```
    RRET R0 ,#0X39      ;C
    RRET R0 ,#0X5E      ;D
    RRET R0 ,#0X79      ;E
    RRET R0 ,#0X71      ;F
    RRET R0 ,#0X40      ;-

INTERRUPT
;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1
    ;--------------------------------
    ;中断处理

    ;--------------------------------
;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;*************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
DELAY
    MOV R0,#0x12
    MOV DELAY_NUM1, R0
DLP0
    MOV R0,#0X12
    MOV DELAY_NUM2, R0
DLP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP DLP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP DLP0
    CRET                ;子程序返回
```

```
;*************************************************************
*****************
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位 ，
每个数的范围 0~f
;* 返    回: 无
;*************************************************************
*****************
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0          ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0          ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0          ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0          ;打开数码管的 千位
    MOV R2 , THOUSAND
```

```
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1          ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0
    CALL DELAY
    CRET
;*************************************************************
*****************
;* 函 数 名: MAIN
;* 函数功能:
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
MAIN
    CALL INIT_RAM              ;初始化RAM
    CALL INIT_ALL             ;初始化寄存器

    MOV R0,#(Count_Timers)
    MOV Count_Timer,R0
CLOCK_OUT
    SET P0,P04
    CALL DELAY

    CLR P0,P04
    CALL DELAY

    DECJZ Count_Timer
    JMP CLOCK_OUT

    MOV R0,T0
    MOV UNIT,R0
    MOV R0,#0x0F
    AND UNIT,R0               ;防溢出数量的运行异常处理

    CLR DECADE                ;最多计数15个，所以 十位、百位、千位均
清零
    CLR HUNDRED
    CLR THOUSAND

LOOP
    CALL LED_DISPLAY
```

```
    JMP LOOP

    CRET
;*************************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0 ,#0x0C
    MOV TR0, R0          ;设置P03端口为输入 P02作为T0计数器时钟输入
口，需配置为输入
    MOV R0 ,#0x00
    MOV TR1, R0          ;设置P1端口为输出
    MOV R0 ,#0X00
    MOV TR2, R0          ;设置P2端口为输出

    CLR P0
    MOV R0 ,#0X3F
    MOV P1, R0           ;默认LED显示为0
    CLR P2

;   MOV R0 ,#0x28        ;允许P0口使用上拉功能，使能T0计数模式，上升
沿触发  将预分频器用于WDT
    MOV R0 ,#0x38        ;允许P0口使用上拉功能，使能T0计数模式，下降
沿触发  将预分频器用于WDT
    MOV OPTR, R0
    CLR ANSEL        ;P02口配置为数字口
```

```
    MOV R0 ,#0x00
    MOV T0, R0          ;设定计数初始值为0x00，当外部时钟引脚P02有
一个上升沿信号时加1
    MOV R0,#0X00
    MOV INTCTL,R0       ;不使用中断

    CRET
;/*************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************/
INIT_RAM
    CLR   R2            ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR   PSW ,RP0
    MOV   R0 ,#0X80     ;起始地址0x80
    MOV   R1 ,#0X20     ;操作个数
    CALL    INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET   PSW ,RP0
    MOV   R0 ,#0X80     ;起始地址0x80
    MOV   R1 ,#0X01     ;操作个数
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR   PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .END
```

实验六　T1定时器门控位启动实验

6.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：T1定时器门控位启动实验

项 目 名：06-T1_Timer_Gating_Start_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述:门控使能位启动T1定时器，使用芯片外部引脚T1G（P04）控制T1启动，如果T1G引脚为低电平，启动 T1，数码管显示数值开始递增，为高电平或者悬空，关闭T1，数码管显示数值停止递增。

6.2 硬件说明

单片机上电时，P04口为高电平，T1不启动，数码管显示0000，用杜邦线将P04与GND连接到一起时，定时器T1开始工作，数码管每隔1S加1，当用杜邦线将P04与VCC连接或者悬空时定时器T1停止工作，数码管显示数值停止累加。单片机的P10~P17跳上跳线,P24~P27跳上跳线（数码管显示）。

6.3 程序设计说明

T1是一个16位的定时/计数器,T1的低8位在寄存器T1L中,高8位在寄存器T1H中，当T1计数值达到65535后，T1的值再加1就会产生溢出，将T1中断标志位置1。T1用于定时和计数模式时其用法和T0基本一样,本实验着重讲解T1门控启动的使用方法。当配置T1GC = 1时，则使能了门控启动功能，所谓门控启动，其实就是通过外部引脚的电平变化来控制T1是否启动的一种方式。例如，当T1GC = 0时，只要配置T1ON = 1即可启动T1，但是当T1GC = 1时，必须当T1ON = 1并且P04端口电平为低时T1才启动，否则T1不启动。所以根据实验现象，刚上电时P04口电平为高，所以数码管显示数值不递增，当使用杜邦线将P04端口和GND连接时T1启动，数码管显示数值开始递增就是这个原因。

在配置过程中中需注意一下几点：

1、 T1属于外部中断，使用其中断功能时，需使能INTCTL寄存器的外部中断总使能位，即PUIE = 1。

2、 T1使用门控启动功能时，P04配置为输入口，并且需使能上拉功能。

3、 门控位是在T1CTL寄存器的T1ON = 1的情况下才起作用。

4、 使用门控启动T1可以较方便地测试出T1GC（P04）引脚为低电平的时间。

6.4 .1 C程序代码

```
#include<KF8F312.h>

/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int

/*********************宏定义结束*************************/

/*********************全局变量*************************/
uchar const Arr_num[] = {0X3F,
                         0X06,
                         0X5B,
                         0X4F,
                         0X66,
                         0X6D,
                         0X7D,
                         0X07,
                         0X7F,
                         0X6F
};                                      //共阴接法,数码管显示笔形码
uchar Unit, Decade, Hundred, Thousand;   //定义个十百千位
uchar time_1s;                           //定时计数器
/*********************全局变量结束***********************/


/************************
 * 函数名     : Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回       : 无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}

void Init_fun(void)
{
```

```
    OSCCTL = 0x60;              // 设置为8M

    /*****端口初始化********/
    TR0 = 0x18;                   //设置P03端口为输入  P04为输入口
    TR1 = 0x00;                   //设置P1端口为输出
    TR2 = 0x00;                   //设置P2端口为输出

    P0 = 0x10;
    P1 = 0;
    P2 = 0;

    ANSEL = 0;                 // 配置为数字口
    PUR = 0X10;                // 使能P04上拉位ANS4
    OPTR = 0;                  // 使能上拉总使能位PUPH

    /****初始化定时器0*****/
    T1CTL = 0x50;             //使能门控位，2分频，定时模式
    T1H = 0xEC;                   //设定定时5ms,8M时钟、T12分频  则每隔1us计
数器加1，0xFFFF - 0XEC77 = 0X1388 = 5000us 即5ms产生一次中断
    T1L = 0x77;
    EIE1 = 0X01;               //使能T1中断 T1IE
    EIF1 = 0;                  //清T1的标志位T1IF
    T1CTL = 0X41;              //使能T1启动位T1ON,使能T1门控位T1GC
    INTCTL = 0Xc0;             //使能总中断AIE、使能外部中断位PUIE  T1属于外
部中断，所以需使能外部中断
}
void main()
{
    uchar i;
    Init_fun();                   //初始化
    while(1)
    {
      for(i = 0; i < 100; i++)
      {
       P2 = 0XE0;             //打开数码管的 个位
       P1 = Arr_num[Unit];
       Delay_200us();
       P1 = 0;
       P2 = 0XF0;             //消隐

       P2 = 0XD0;             //打开数码管的 十位
       P1 = Arr_num[Decade];
       Delay_200us();
       P1 = 0;
```

```
        P2 = 0XF0;            //消隐

        P2 = 0XB0;            //打开数码管的 百位
        P1 = Arr_num[Hundred];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐

        P2 = 0X70;            //打开数码管的 千位
        P1 = Arr_num[Thousand];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐
      }
    }
}
//中断函数
void int_fun() __interrupt
{
    if(T1IF)
    {
        T1IF = 0;                           //清零中断标志
        T1H = 0xEC;
        T1L = 0x77;                            //T0重新赋值
        if(time_1s++ >= 200)                //判断是否达到1s记时   5ms*200
= 1s
        {
            time_1s = 0;                    //清零计数器
            Unit++;                         //个位相加
            if(Unit == 10)
            {
                Unit = 0;
                Decade++;                    //十位相加
                if(Decade == 10)
                {
                    Decade = 0;
                    Hundred++;              //百位相加
                    if(Hundred == 10)
                    {
                        Hundred = 0;
                        Thousand++;              //千位相加
                        if(Thousand == 10)
                        {
                            Thousand = 0;
```

```
                    }
                }
            }
        }
    }
}
```

6.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

DELAY_NUM1          .EQU        0x80        ;延时用的临时变量
DELAY_NUM2          .EQU        0x81        ;延时用的临时变量
PSW_TEMP            .EQU        0X83        ;PSW的临时保存变量
PCH_TEMP            .EQU        0X84        ;PCH的临时保存变量
UNIT                .EQU        0X85        ;数码管个位
DECADE              .EQU        0X86        ;数码管十位
HUNDRED                 .EQU        0X87        ;数码管百位
THOUSAND            .EQU        0X88        ;数码管千位
TIME_1S                 .EQU        0X89        ;累计计数1s

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

;*********************************************************************
**********
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返    回: R0，数码管显示的编码值
;* 注  意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不同页造成查表
出错
;*********************************************************************
**********
DIG_DATA
    ADD PCL, R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
```

```
    RRET R0 ,#0X7D        ;6
    RRET R0 ,#0X07        ;7
    RRET R0 ,#0X7F        ;8
    RRET R0 ,#0X6F        ;9
    RRET R0 ,#0X77        ;A
    RRET R0 ,#0X7C        ;B
    RRET R0 ,#0X39        ;C
    RRET R0 ,#0X5E        ;D
    RRET R0 ,#0X79        ;E
    RRET R0 ,#0X71        ;F
    RRET R0 ,#0X00        ;空

INTERRUPT
;保护现场作用
    SWAPR R1 ,PSW
    MOV PSW_TEMP, R1
    MOV R1 , PCH
    MOV PCH_TEMP, R1
    ;---------------------------------
    ;中断处理
    JB EIF1,T1IF
    JMP INT_END
    CLR EIF1,T1IF

    MOV R1 ,#0xEC
    MOV T1H, R1
    MOV R1 ,#0x77
    MOV T1L, R1
    INC TIME_1S
    MOV R1 ,#0XC8                ;累计 计数200个5ms 即1S
    XOR R1 ,TIME_1S
    JB PSW,Z
    JMP INT_END
    CLR  TIME_1S

    INC UNIT                    ;个位数字的递增
    MOV R1 ,UNIT
    XOR R1 ,#0X0A
    JB PSW ,Z
    JMP INT_END
    CLR UNIT

    INC DECADE                  ;十位数字的递增
    MOV R1 ,DECADE
```

```
    XOR R1 ,#0X0A
    JB PSW ,Z
    JMP INT_END
    CLR DECADE

    INC HUNDRED                    ;百位数字的递增
    MOV R1 ,HUNDRED
    XOR R1 ,#0X0A
    JB PSW ,Z
    JMP INT_END
    CLR HUNDRED

    INC THOUSAND                ;千位数字的递增
    MOV R1 ,THOUSAND
    XOR R1 ,#0X0A
    JB PSW ,Z
    JMP INT_END
    CLR THOUSAND

    ;---------------------------------
;恢复现场作用

INT_END
    MOV R1 , PCH_TEMP
    MOV PCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET

MAIN
    CALL INIT_ALL              ;初始化寄存器
    CALL INIT_RAM               ;初始化RAM

LOOP
    CALL LED_DISPLAY
    JMP LOOP


;*********************************************************************
**********
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位  ，每个数的范
围 0~f
```

```
;* 返    回: 无
;*********************************************************************
**********
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0           ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1            ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0           ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1            ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0           ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1            ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0           ;打开数码管的 千位
    MOV R2 , THOUSAND
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1            ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0
    CRET
```

```
;*********************************************************************
**********
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********
INIT_ALL
        ;初始化晶振频率
    MOV R0 ,#0X60
    MOV OSCCTL, R0          ;设置为8M
    CALL #0xFFF
    MOV OSCCAL0, R0         ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0         ;晶振校准1
    NOP
    NOP
    ;端口初始化
    MOV R0 ,#0x18
    MOV TR0, R0             ;设置P03端口为输入  P04为输入口
    MOV R0 ,#0x00
    MOV TR1, R0             ;设置P1端口为输出
    MOV R0 ,#0X00
    MOV TR2, R0             ;设置P2端口为输出

    MOV R0 ,#0x01
    MOV P0, R0          ;开始时给P10赋高电平
    CLR P1
    CLR P2

    CLR ANSEL           ;配置为数字口
    CLR OPTR            ;使能上拉总使能位
    MOV R0,#0X10
    MOV PUR,R0          ;使能P04上拉

    MOV R0,#0X50        ;使能门控位，2分频，定时模式
    MOV T1CTL,R0        ;设定定时5ms,8M时钟、T12分频  则每隔1us计数器加1,
0xFFFF - 0XEC77 = 0X1388 = 5000us 即5ms产生一次中断
    MOV R0,#0xEC
    MOV T1H,R0
```

```
    MOV R0,#0x77
    MOV T1L,R0

    CLR EIF1            ;清T1的标志位T1IF
    MOV R0,#0X01
    MOV EIE1,R0         ;使能T1中断 T1IE
    MOV R0,#0X41
    MOV T1CTL,R0        ;使能T1启动位T1ON,使能T1门控位T1GC
    MOV R0,#0Xc0
    MOV INTCTL,R0       ;使能总中断AIE、使能外部中断位PUIE  T1属于外部中断，所以需使能外部中断
    CRET

;*********************************************************************
**********
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********
DELAY
    MOV R0,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2      ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1      ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                  ;子程序返回

;/*********************************************************************
***********
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********/
INIT_RAM
    CLR    PSW ,RP0       ;清除0区的RAM
```

```
_START_CLR
    MOV     R2 ,#0XFF
    MOV     R1 ,#0X80       ;起始地址0x80
    JMP     _CLR_JUDGE
_CLR_PRO
    XOR     R0 ,R2
    CLR     R0
    ST      [R1],R0
    INC     R1
_CLR_JUDGE
    MOV     R0 ,R1
    XOR     R0 ,R2          ;R2xor R0----R2
    JB      PSW,Z
    JMP     _CLR_PRO            ;RESULT !=0, Z=0
    JB      PSW ,RP0
    JMP     _CLR_CHG_BANK
    JMP     _INIT_END

_CLR_CHG_BANK
    SET     PSW ,RP0            ;清除1区的RAM
    JMP     _START_CLR
_INIT_END
    CLR     PSW ,RP0
    CRET

    .END
```

实验七、T2定时中断模式实验

7.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：T2定时中断模式实验

项 目 名：07-T2_Timer_Interrupt_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：T2定时中断模式的使用，使得数码管从0~9999计数，1秒增1，当到9999时，又从0开始计数。

7.2 硬件说明

本实验运用到两个模块，一个是单片机内部功能模块，定时器2（T2），另

一个是数码管显示模块，数码管显示模块用到P1,P2两组IO端口。数码管具体特性在此之前已经讲解过，在此不作敷述。

7.3 程序设计说明

T2与T0/1有所不同，T2是一个8位的定时器，没有外部计数时钟输入脚。当T2用作定时器中断功能时，使用到了两个预分频器，分别为T2分频器1和T2分频器2。和T0作为定时器功能时一样，T2为一个机器周期加1，如当前系统时钟为8M，T2分频器1分频比为4分频，则可计算出一个机器周期 = ((1/8M)* 4) = 0.5us，因分频器1为4分频，所以T2每隔0.5*4 = 2us加1。当T2寄存器与PP3相等时，T2自动清0，发出相等信号给分频器2，分频器2递增。当分频器2分频比设置为1:1时，每次寄存器T2与PP3相等，将会使T2中断标志位T2IF置1；当其设置为1:2时，寄存器T2与PP3相等累计两次才会使T2IF置1，以此类推。例如，设置PP3 = 0xFA，T2每2us加1，分频器2分频比设置为1:10，则产生一次中断所经历的时间T = 2us*0xFA*10 = 5ms。

在使用T2定时器中断时应注意以下几点：

1、 T2属于外部中断，使用中断时应使能INCTL寄存器的外部中断总使能位，即PUIE = 1；

2、 T2中断标志位T2IF置1是由T2计数值和PP3比较并且相等产生的，而不是由T2计数值溢出产生。

T2的原理框图如图7.1所示。

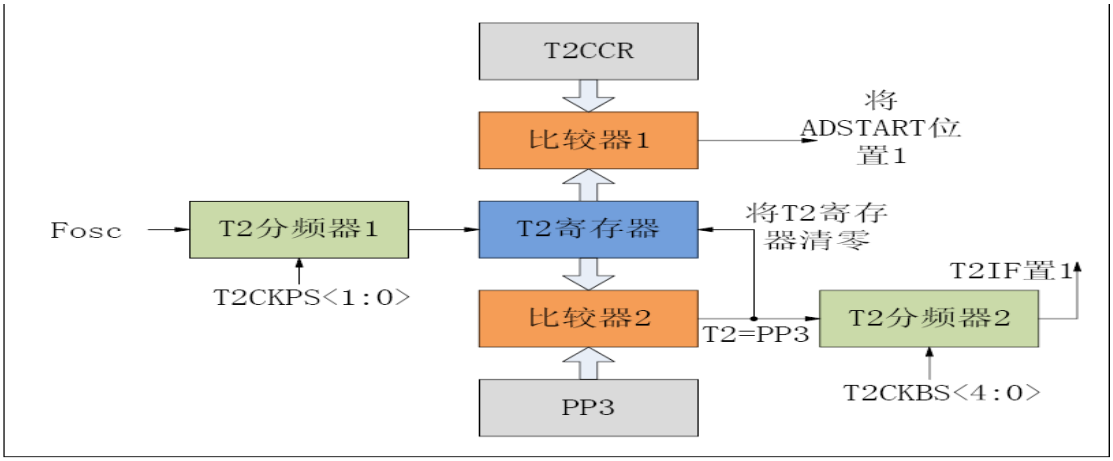


图7.1

7.4 .1 C程序代码

```c
#include<KF8F312.h>

/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int

/*********************宏定义结束*************************/

/*********************全局变量*************************/
uchar const Arr_num[] = {0X3F,
                         0X06,
                         0X5B,
                         0X4F,
                         0X66,
                         0X6D,
                         0X7D,
                         0X07,
                         0X7F,
                         0X6F
};                                      //共阴接法,数码管显示笔形码
uchar Unit, Decade, Hundred, Thousand;   //定义个十百千位
uchar time_1s;                           //定时计数器
/*********************全局变量结束*************************/


/************************
 * 函数名     : Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         : 无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}

void Init_fun(void)
{
    OSCCTL = 0x60;            // 设置为8M

    /*****端口初始化********/
    TR0 = 0x08;                 //设置P03端口为输入
    TR1 = 0x00;                 //设置P1端口为输出
```

```
    TR2 = 0x00;                 //设置P2端口为输出

    P0 = 0;
    P1 = 0;
    P2 = 0;

    /****初始化定时器2*****/
    T2 = 0x00;      // 清T2计数器
    PP3 = 0XFA;     // T2增加到0xFA时即自动清零 并进入中断

    // 预分频器2 10分频   禁止T2  预分频器1 4分频   T2每2us加1，  增加到250时分频器2加1 ，
    // 分频器2递增到10时中断标志位T2IF置1 ，即5ms进入一次中断  启动T2  T2ON = 1
    T2CTL = 0X4D;
    EIE1 = 0X02;  // 使能T2中断 T2IE = 1
    INTCTL = 0XC0;  // 使能总中断AIE 使能外部中断 PUIE
}
void main()
{
    uchar i;
    Init_fun();                 //初始化
    while(1)
    {
       for(i = 0; i < 100; i++)
       {
        P2 = 0XE0;          //打开数码管的 个位
        P1 = Arr_num[Unit];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;          //消隐

        P2 = 0XD0;          //打开数码管的 十位
        P1 = Arr_num[Decade];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;          //消隐

        P2 = 0XB0;          //打开数码管的 百位
        P1 = Arr_num[Hundred];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;          //消隐

        P2 = 0X70;          //打开数码管的 千位
```

```
        P1 = Arr_num[Thousand];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐
       }
    }
}
//中断函数
void int_fun() __interrupt
{
    if(T2IF)
    {
        T2IF = 0;                           //清零中断标志
        if(time_1s++ >= 200)                //判断是否达到1s记时   5ms*200
= 1s
        {
            time_1s = 0;                    //清零计数器
            Unit++;                         //个位相加
            if(Unit == 10)
            {
                Unit = 0;
                Decade++;                    //十位相加
                if(Decade == 10)
                {
                    Decade = 0;
                    Hundred++;              //百位相加
                    if(Hundred == 10)
                    {
                        Hundred = 0;
                        Thousand++;             //千位相加
                        if(Thousand == 10)
                        {
                            Thousand = 0;
                        }
                    }
                }
            }
        }
    }
}
```

7.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"
;;;;;;;;;;;;;;;;通用寄存器;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
DELAY_NUM1      .EQU    0X81        ;延时用的临时变量
DELAY_NUM2      .EQU    0X82        ;延时用的临时变量
PSW_TEMP        .EQU    0X83        ;PSW的临时保存变量
PCH_TEMP        .EQU    0X84        ;PCH的临时保存变量
UNIT            .EQU    0X85        ;数码管个位
DECADE          .EQU    0X86        ;数码管十位
HUNDRED             .EQU    0X87        ;数码管百位
THOUSAND        .EQU    0X88        ;数码管千位
TIME_1S             .EQU    0X89        ;累计计数1s
FLAG_1S             .EQU    0X8A        ;计数1s的标志位

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

;*********************************************************************
**********
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返    回: R0，数码管显示的编码值
;* 注  意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不同页造成查表
出错
;*********************************************************************
**********
DIG_DATA
    ADD PCL, R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
    RRET R0 ,#0X7D      ;6
    RRET R0 ,#0X07      ;7
    RRET R0 ,#0X7F      ;8
    RRET R0 ,#0X6F      ;9
    RRET R0 ,#0X77      ;A
    RRET R0 ,#0X7C      ;B
    RRET R0 ,#0X39      ;C
```

```
    RRET R0 ,#0X5E        ;D
    RRET R0 ,#0X79        ;E
    RRET R0 ,#0X71        ;F
    RRET R0 ,#0X00        ;空

INTERRUPT

;保护现场作用
    SWAPR R1 ,PSW
    MOV PSW_TEMP, R1
    MOV R1 , PCH
    MOV PCH_TEMP, R1

;/*********************************中断处理程序
**********************************/
    ;中断处理
    JB EIF1,T2IF
    JMP INT_END
    CLR EIF1,T2IF

    INC TIME_1S
    MOV R1 ,#0XC8                ;累计 计数200个5ms 即1S
    XOR R1 ,TIME_1S
    JB PSW,Z
    JMP INT_END
    CLR  TIME_1S
    MOV R1 ,#0X55
    MOV FLAG_1S, R1

;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOV R1 , PCH_TEMP
    MOV PCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET

MAIN
    CALL INIT_ALL              ;初始化寄存器
    CALL INIT_RAM              ;初始化RAM
```

```
TIME_1S_LOOP
    CALL LED_DISPLAY          ;数码管显示当前数值
    MOV R0 ,#0X55
    XOR R0 ,FLAG_1S
    JB  PSW,Z
    JMP TIME_1S_LOOP

LOOP
    CLR FLAG_1S
    INC UNIT                  ;个位数字的递增
    MOV R0 ,UNIT
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR UNIT

    INC DECADE                ;十位数字的递增
    MOV R0 ,DECADE
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR DECADE

    INC HUNDRED                   ;百位数字的递增
    MOV R0 ,HUNDRED
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR HUNDRED

    INC THOUSAND              ;千位数字的递增
    MOV R0 ,THOUSAND
    XOR R0 ,#0X0A
    JB PSW ,Z
    JMP C_DIS
    CLR THOUSAND

C_DIS
    CALL LED_DISPLAY          ;数码管显示当前数值
    JMP TIME_1S_LOOP

;*********************************************************************
**********
;* 函 数 名: LED_DISPLAY
```

```
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位  ， 每个数的范围 0~f
;* 返    回: 无
;*********************************************************************
**********
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0           ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0           ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0           ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0           ;打开数码管的 千位
    MOV R2 , THOUSAND
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
```

```
    MOV R0 ,#0XF0
    MOV P2 , R0
    CRET

;*********************************************************************
**********
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********
DELAY
    MOV R0,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2      ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1      ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                  ;子程序返回

;*********************************************************************
**********
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********
INIT_ALL
    ;初始化晶振频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0         ;设置为4M
    CALL #0xFFF
    MOV OSCCAL0, R0        ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0        ;晶振校准1
    NOP
```

```
    NOP
    MOV R0,#0X08
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    CLR P2

;   // 预分频器2 10分频   禁止T2  预分频器1 4分频   T2每2us加1，  增加到250时分
频器2加1 ，
;   // 分频器2递增到10时中断标志位T2IF置1 ，即5ms进入一次中断  启动T2  T2ON = 1
    MOV R0,#0X4D
    MOV T2CTL,R0
    CLR T2      ;清T2计数器
    MOV R0,#0XFA ;T2增加到0xFA时即自动清零 并进入中断
    MOV PP3,R0
    MOV R0,#0X02 ;使能T2中断 T2IE = 1
    MOV EIE1,R0
    MOV R0,#0XC0 ;使能总中断AIE 使能外部中断 PUIE
    MOV INTCTL,R0
    CRET

;/*********************************************************************
***********
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********/
INIT_RAM
    CLR    PSW ,RP0       ;清除0区的RAM
_START_CLR
    MOV    R2 ,#0XFF
    MOV    R1 ,#0X80      ;起始地址0x80
    JMP    _CLR_JUDGE
_CLR_PRO
    XOR    R0 ,R2
    CLR    R0
    ST     [R1],R0
```

```
    INC     R1
_CLR_JUDGE
    MOV     R0 ,R1
    XOR     R0 ,R2          ;R2xor R0----R2
    JB      PSW,Z
    JMP     _CLR_PRO            ;RESULT !=0, Z=0
    JB      PSW ,RP0
    JMP     _CLR_CHG_BANK
    JMP     _INIT_END

_CLR_CHG_BANK
    SET     PSW ,RP0            ;清除1区的RAM
    JMP     _START_CLR
_INIT_END
    CLR     PSW ,RP0
    CRET

    .END
```

实验八、ADC转换采样实验

8.1 程序声明

KF8F系列单片机开发板演示程序

标 题: ADC转换采样

项 目 名: 08-ADC_TEST

开发环境: ChipON IDE

使用芯片: KF8F312

作 者:上海芯旺微电子技术有限公司

功能简述: ADC转换采样，数码管显示采样得到的值(16进制显示)

8.2 硬件说明

R10，R11，R12分别为10K的电位器，使用杜邦线将任意一个电位器的引出端接到单片机的AN2（P02）即可，接线图如图。通过改变电位器的阻值使得AN2端口的电压值发生改变，并通过数码管显示相应的转换结果。显示模块的接口已经详细讲解过，在此不作敷述。ADC模块原理图如图8.1所示。

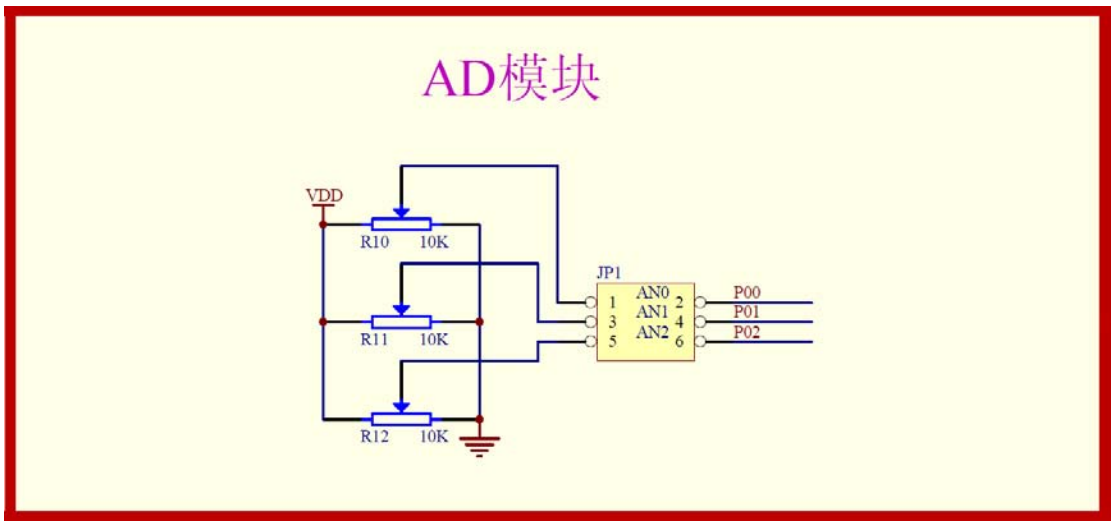


图 8.1 ADC 模块原理图

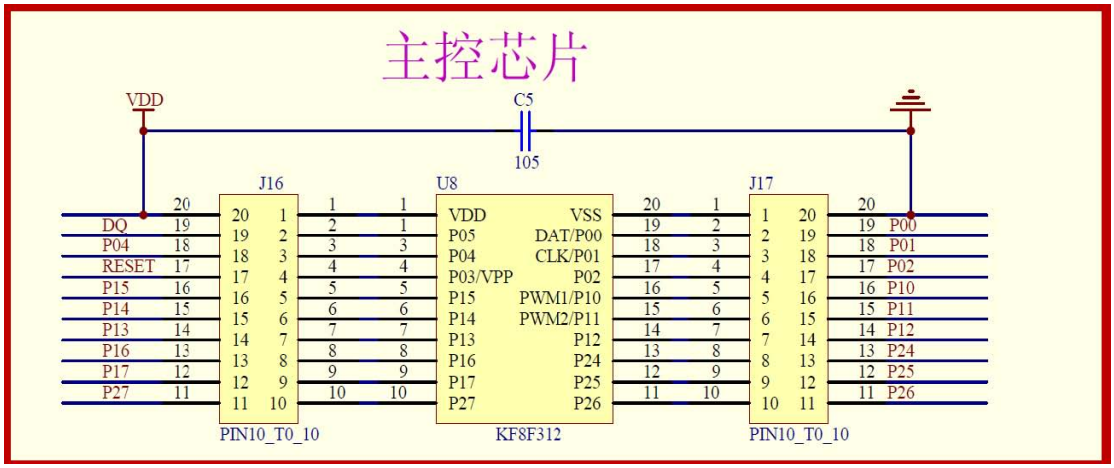


图 8.2：主控芯片原理图

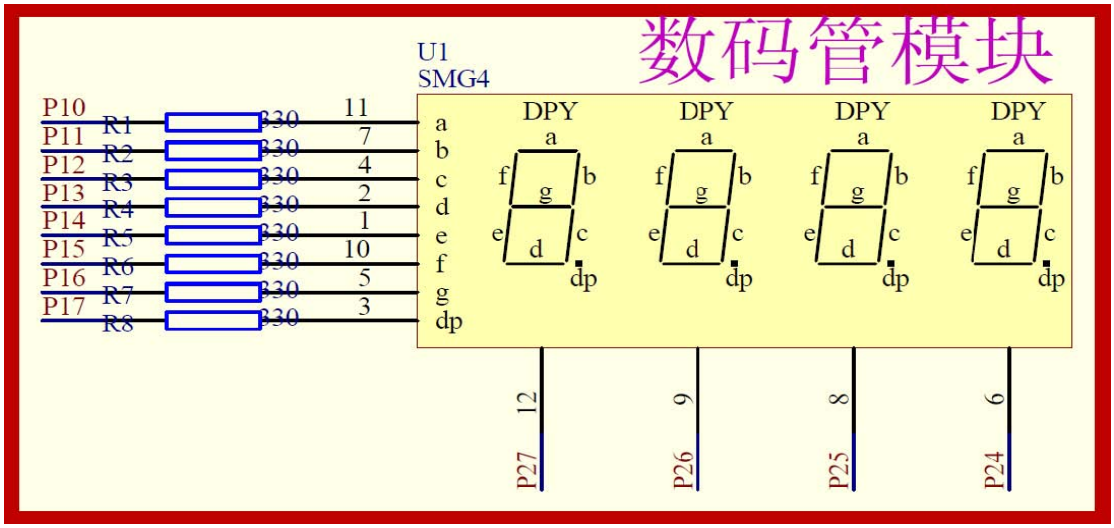


图 8.3：数码管原理图

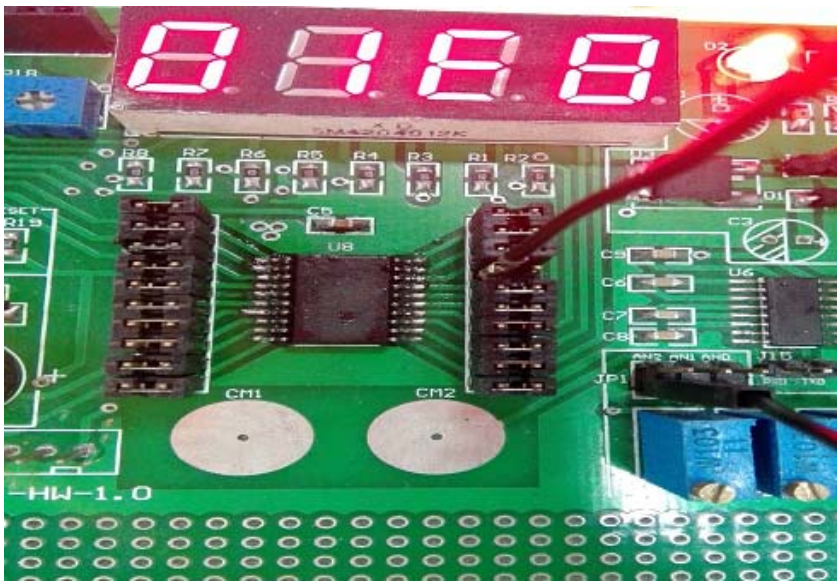


图 8.6：P02 和电位器引出端接线图

8.3 程序设计说明

本实验主要对通道2进行模拟量的采集，并将转换后的数字量显示在数码管上。在此重点讲解ADC的转换过程。KF8F312拥有12路模拟输入通道和1个内部输入通道(VREOUT)，这些通道被多路转换到同一采样保持电路。采样保持电路的输出与转换器的输入相连接。转换器通过逐次逼近法将模拟输入信号转换为二进制值，并将转换结果存放到10位寄存器中。由于所有的通道转换都是输送到同一采样电路，所以KF8F312无法进行同时多路ADC采样，只能一次进行一路采样。

使用AD转换模块需要对ADCCTL0和ADCCTL1两个寄存器进行配置，下面分别对这两个寄存器的配置进行说明：

ADCCTL0寄存器的各个位如下

	bit7							bit0
复位值	ADLR	T2CCRON	CHS3	CHS2	CHS1	CHS0	START	ADEN
0000 0000	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

ADLR = 1 转换结果右对齐

ADLR = 0 转换结果左对齐

结果对齐方式设置其实就是设置转换后得到的10位二进制结果存放在ADCDATAH和ADCDATAL的方式，具体如图8.4所示。

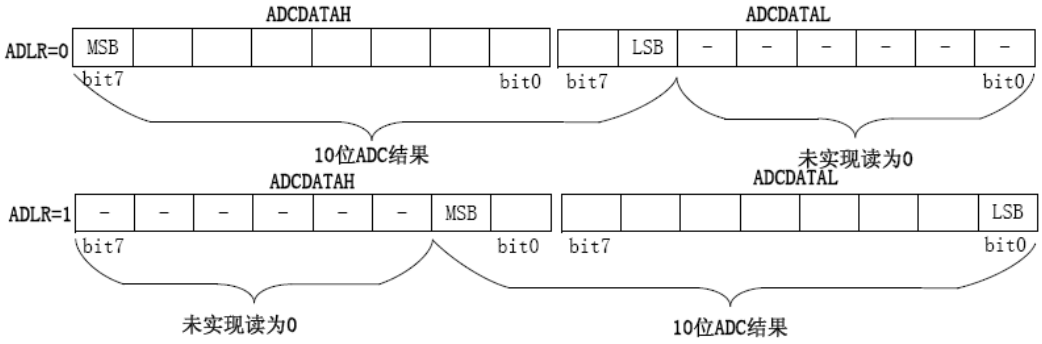


图8.4 AD转换结果对齐方式

T2CCRON :A/D转换结果输出格式选择位

T2CCRON = 1 使能T2启动AD转换, AD转换模块可以使用T2定时器通过定时的方式启动转换，具体使用方法在实验九中有专门讲解，在此不做赘述。

T2CCRON = 0 禁止T2启动AD转换,需要在软件中使

CHS<3:0> :模拟通道选择位，如本实验中模拟信号由AN2模拟通道输入，则CHS<3:0> = 0B0010。

START：AD转换状态位

START = 1，启动AD转换，转换结束后硬件清0。START = 1启动AD转换，本实验就是采用这种方式。

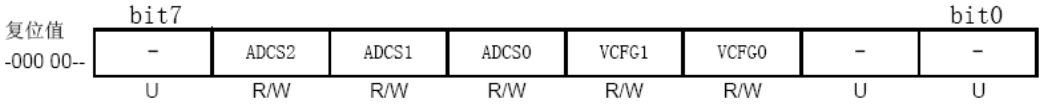
START = 0，AD转换结束或未进行

ADEN: A/D转换使能位

ADEN = 1 使能AD转换,使用AD转换时需对其进行使能

ADEN = 0 关闭AD转换

ADCCTL1寄存器的各个位如下



ADCS<2:0>: A/D转换时钟选择位 本实验使用8分频，所以ADCS<2:0> = 0B001。

VCFG<1:0>: A/D转换参考电压选择位 本实验使用VDD作为参考电压，所以VCFG<1:0> = 0B00。

关于参考电压，如实验采用VDD = 5V作为参考电压，AD为10位转化精度，则当转换输出的10二进制数全部为1时表示的电压值就是参考电压值，即5V。例如，在本实验中从AN2模拟通道输入的电压值为2.0V,则转化后的数值应该为2/5*0x3FF

= 0x199(数码管显示的是十六进制数值)。所以，当模拟通道输入的电压值较小时，减小参考电压的值可以提高分辨率。

至此，以完成对ADCCTL0和ADCCTL1两个寄存器的配置，除此之外，还需注意一下几点：

1、 AD转换中，相应的模拟输入通道需配置为模拟输入状态。

2、 关于转换时钟的选择，完成一次AD转换需要11个AD转换周期，为了保证转换数据地正确性，应AD转换周期的典型值为1us，即完成一次AD转换需11us。如本实验中系统时钟频率为8M，AD转换频率为8分频，则转换周期为1/（8M/8）= 1us。

为了让采样结果更加稳定，本实验进行连续8次采样再取平均值，这样结果更加稳定，如果对稳定性能要求较高，可自行适当增加连续采样的次数。

8.4.1 C 程序代码

```
#include<kf8f312.h>
/***************宏定义**********************/
#define uchar unsigned char
#define uint  unsigned int

/******************宏定义结束*****************/
/**************定义全局变量（注：为节省内存空间，全局变量尽量不要赋初值）
*********/
uchar const Arr_num[] = {
                  0X3F,      //     ;0
                  0X06,      //     ;1
                  0X5B,      //     ;2
                  0X4F,      //     ;3
                  0X66,      //     ;4
                  0X6D,      //     ;5
                  0X7D,      //     ;6
                  0X07,      //     ;7
                  0X7F,      //     ;8
                  0X6F,      //     ;9
                  0X77,      //     ;A
                  0X7C,      //     ;B
                  0X39,      //     ;C
                  0X5E,      //     ;D
```

```
                0X79,      //      ;E
                0X71,      //      ;F
                0X00       //      ;空
};                                  //定义数码管可以显示的0——f,共阴接法
uchar Unit, Decade, Hundred, Thousand;    //定义数码管的个十百千位
/**********************全局变量定义结束
************************************/
/*****************函数声明(注：中断和main函数不需声
明)************************/
void Init_fun();
void Delay_200us();
uint Adc_fun(void);
uint Adc_read(void);
void Display();
/********************函数声明结束**********************************/

/************************
 * 函数名     ：main
 * 函数功能：程序入口主函数
 * 入口参数：无
 * 返回         ：无
 ************************/

void main()
{
    uint Dis_adc = 0 ;                  //采样接收变量
    Init_fun();                         //初始化
    while(1)
    {
        Dis_adc = Adc_read();          //将ADC采样返回值赋予Dis_adc
        Unit = Dis_adc&0x0f;           //将Dis_adc的最低4位赋予数码管的个位
        Decade = Dis_adc&0xf0;
        Decade >>= 4;                 //将Dis_adc的第7-4位赋予数码管的十位
        Dis_adc >>= 8;                //右移8位，取Dis_adc的高8位
        Hundred = Dis_adc&0x0f;        //将此时的Dis_adc的最低4位赋予数码管
的百位
        Thousand = Dis_adc&0xf0;
        Thousand >>= 4;               //将此时的Dis_adc的第7-4位赋予数码管的
千位
        Display();                    //显示

    }
}
```

```
/************************
 * 函数名     ：Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         ：无
 ************************/

void Init_fun()
{
    OSCCTL = 0x60;                    //设置为4M
    /*****端口初始化*****/
    TR0 = 0x0C;                         //设置P03端口为输入  P02为AN2模拟通
道 配置为输入口
    TR1 = 0x00;                         //设置P1端口为输出
    TR2 = 0x00;                         //设置P2端口为输出
    P0 = 0;
    P1 = 0;
    P2 = 0;

    /****初始化AD寄存器****/
    ANSEL = 0x04;                   //设置通道2为模拟口
    ADCCTL0 = 0x89;                     //右对齐，通道2， AD使能打开
    ADCCTL1 = 0x10;                     //8分频，VDD作为参考电压

}
/************************
 * 函数名     ：Adc_fun
 * 函数功能：ADC函数，模数转换
 * 入口参数：无
 * 返回         ：返回ADC采样值
 ************************/
uint Adc_fun(void)
{
       uint Adc_num = 0 ;             //ADC转换缓冲变量
       START = 1;                   //启动ADC
       while(START);                 //等待转换结束
/************************************结果右对齐
*****************************************/
       Adc_num = ADCDATAH&0x03;   //将高位加进去
       Adc_num <<= 8;                  //左移动8位高位归位
       Adc_num |= ADCDATAL;       //将低位加进去
/************************************结果右对齐
*****************************************/
       return Adc_num;                 //返回转换值
```

```
}
/************************
 * 函数名     ：Adc_read
 * 函数功能：ADC求均值函数，adc读取函数
 * 入口参数：无
 * 返回         ：返回ADC采样8次均值
 ************************/
uint Adc_read(void)
{
    uint Adc_sum = 0 ;                //adc采样累加变量
    uchar i = 0;
    for(i = 0; i < 8; i++)
    {
        Adc_sum += Adc_fun();          //累加八次采样值
    }
    Adc_sum >>= 3;                    //右移动3位  除8求均值
    return Adc_sum;
}

/************************
 * 函数名     ：Display
 * 函数功能：数码管显示
 * 入口参数：显示数字
 * 返回         ：无
 ************************/
void Display()
  {

    uchar i;
    for(i = 0; i < 50 ; i++)
     {

        P2 = 0XE0;            //打开数码管的 个位
        P1 = Arr_num[Unit];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐

        P2 = 0XD0;            //打开数码管的 十位
        P1 = Arr_num[Decade];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐
```

```
        P2 = 0XB0;            //打开数码管的 百位
        P1 = Arr_num[Hundred];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐

        P2 = 0X70;            //打开数码管的 千位
        P1 = Arr_num[Thousand];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐
        }
}


/************************
 * 函数名     : Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         : 无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}

//中断函数
void int_fun() __interrupt
{
}
```

8.4.2 汇编程序代码

```
    .INCLUDE "KF8F312.INC"
;;;;;;;;;;;;;;;;通用寄存器;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DELAY_NUM1     .EQU    0X81        ;延时用的临时变量
DELAY_NUM2     .EQU    0X82        ;延时用的临时变量
PSW_TEMP       .EQU    0X85        ;PSW的临时保存变量
PCH_TEMP       .EQU    0X86        ;PCH的临时保存变量
UNIT           .EQU    0X88        ;数码管个位
DECADE         .EQU    0X89        ;数码管十位
```

```
HUNDRED          .EQU   0X8A       ;数码管百位
THOUSAND      .EQU   0X8B       ;数码管千位
AD_H          .EQU   0X8C       ;采样得到的高位
AD_L          .EQU   0X8D       ;采样得到的低位
AD_TEMP          .EQU   0X8E       ;AD采样的临时变量
DIS_DEL          .EQU   0X8F       ;减缓采样速度用的变量
AD_H_SUM      .EQU   0X90       ;存放AD采样8次的高位
AD_L_SUM      .EQU   0X91       ;存放AD采样8次的低位
AD_8_T        .EQU   0X92       ;8次采样电压，求平均计数

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    .ORG 0X0000
     NOP
     JMP MAIN                    ;入口
    .ORG 0X0004
    JMP INTERRUPT
;*********************************************************************
**********
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返    回: R0，数码管显示的编码值
;* 注  意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不同页造成查表
出错
;*********************************************************************
**********
DIG_DATA
    ADD PCL, R2
    RRET R0 ,#0X3F        ;0
    RRET R0 ,#0X06        ;1
    RRET R0 ,#0X5B        ;2
    RRET R0 ,#0X4F        ;3
    RRET R0 ,#0X66        ;4
    RRET R0 ,#0X6D        ;5
    RRET R0 ,#0X7D        ;6
    RRET R0 ,#0X07        ;7
    RRET R0 ,#0X7F        ;8
    RRET R0 ,#0X6F        ;9
    RRET R0 ,#0X77        ;A
    RRET R0 ,#0X7C        ;B
    RRET R0 ,#0X39        ;C
    RRET R0 ,#0X5E        ;D
    RRET R0 ,#0X79        ;E
    RRET R0 ,#0X71        ;F
```

```
    RRET R0 ,#0X00          ;空

;;;;;;;;;;;;;;;中断部分;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERRUPT
;保护现场作用
    SWAPR R1 ,PSW
    MOV PSW_TEMP, R1
    MOV R1 , PCH
    MOV PCH_TEMP, R1
    ;---------------------------------
    ;中断处理
    ;...
    ;...
    ;...
    ;---------------------------------
;恢复现场作用
INT_END
    MOV R1 , PCH_TEMP
    MOV PCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;;;;;;;;;;;;;;中断部分结束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;入口函数,主程序;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN
    CALL INIT_ALL               ;初始化寄存器
    CALL INIT_RAM               ;初始化RAM
LOOP
    CLR AD_H_SUM
    CLR AD_L_SUM
    CLR AD_8_T

ADC_LOOP_8                      ;AD采样8次求平均
    CALL ADC_FUN
    MOV R0 ,AD_L
    ADD AD_L_SUM, R0            ;累加AD采样的低位
    JNB PSW ,CY
    INC AD_H_SUM
    MOV R0 ,AD_H                ;累加AD采样的高位
    ADD AD_H_SUM, R0
    INC AD_8_T
    MOV R0 ,AD_8_T
    XOR R0 ,#0X08               ;是否采样了8次进行判断
```

```
    JB  PSW,Z
    JMP ADC_LOOP_8

    CLR PSW,CY                  ;以下进行3次右移，即除以8 ，取平均
    RRC AD_H_SUM
    RRC AD_L_SUM
    CLR PSW,CY
    RRC AD_H_SUM
    RRC AD_L_SUM
    CLR PSW,CY
    RRC AD_H_SUM
    RRC AD_L_SUM

    MOV R0 ,AD_H_SUM
    MOV AD_H, R0
    MOV R0 ,AD_L_SUM
    MOV AD_L, R0

    MOV R0 ,AD_H
    RRC R0
    RRC R0
    RRC R0
    RRC R0                      ;右移4位
    AND R0 , #0X0F
    MOV THOUSAND, R0            ;数码管千位的值

    MOV R0 ,AD_H
    AND R0 ,#0X0F
    MOV HUNDRED, R0                 ;数码管百位的值

    MOV R0 ,AD_L
    RRC R0
    RRC R0
    RRC R0
    RRC R0                      ;右移4位
    AND R0 , #0X0F
    MOV DECADE, R0                  ;数码管十位的值

    MOV R0 ,AD_L
    AND R0 ,#0X0F
    MOV UNIT, R0                ;数码管个位的值

C_DIS
    CALL LED_DISPLAY            ;数码管显示当前数值
```

```
    INC DIS_DEL                 ;DIS_DEL是用来延时AD采样速度用的
    MOV R0 ,DIS_DEL
    XOR R0 ,#0X55
    JB PSW ,Z
    JMP C_DIS
    CLR DIS_DEL
    JMP LOOP

;;;;;;;;;;;;;;主程序结束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;*********************************************************************
**********
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********
INIT_ALL
    ;初始化晶振频率
    MOV R0 ,#0X60
    MOV OSCCTL, R0          ;设置为8M
    CALL #0xFFF
    MOV OSCCAL0, R0         ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0         ;晶振校准1
    NOP
    NOP
    ;端口初始化
    MOV R0 ,#0x0C
    MOV TR0, R0             ;设置P03端口为输入  P02为AN2模拟通道 配置为输入口
    MOV R0 ,#0x00
    MOV TR1, R0             ;设置P1端口为输出
    MOV R0 ,#0x00
    MOV TR2, R0             ;设置P2端口为输出

    CLR P0
    CLR P1
    CLR P2
```

```
    ;初始化AD寄存器组
    MOV R0 ,#0x04
    MOV ANSEL, R0      ;设置通道2为模拟口
    MOV R0 ,#0x89      ;右对齐，通道2， AD使能打开
    MOV ADCCTL0, R0
    MOV R0 ,#0x10      ;8分频，VDD作为参考电压
    MOV ADCCTL1, R0

    CRET

;*******************************************************************
**********
;* 函 数 名: ADC_FUN
;* 函数功能: ADC函数，模数转换
;* 入口参数: 无
;* 返    回: ADCDATAH采样值的高位，ADCDATAL采样值的低位
;*******************************************************************
**********
ADC_FUN
    SET ADCCTL0,START
ADC_LOOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    JNB ADCCTL0,START
    JMP ADC_LOOP
    MOV R0 ,ADCDATAH
    MOV AD_H, R0       ;10位AD转换结果的高2位
    MOV R0 ,ADCDATAL
    MOV AD_L, R0       ;10位AD转换结果的低8位
    CRET

;*******************************************************************
**********
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位  ，每个数的范
围 0~f
;* 返    回: 无
```

```
;*********************************************************************
**********
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0           ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0           ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0           ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0           ;打开数码管的 千位
    MOV R2 , THOUSAND
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1              ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0
    CRET
```

```
;*********************************************************************
**********
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********
DELAY
    MOV R0 ,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0 ,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2      ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1      ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                  ;子程序返回

;/*********************************************************************
***********
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*********************************************************************
**********/
INIT_RAM
    CLR    PSW ,RP0       ;清除0区的RAM
_START_CLR
    MOV    R2 ,#0XFF
    MOV    R1 ,#0X80      ;起始地址0x80
    JMP    _CLR_JUDGE
_CLR_PRO
    XOR    R0 ,R2
    CLR    R0
    ST     [R1],R0
    INC    R1
_CLR_JUDGE
    MOV    R0 ,R1
    XOR    R0 ,R2         ;R2xor R0----R2
    JB     PSW,Z
```

```
    JMP     _CLR_PRO            ;RESULT !=0, Z=0
    JB      PSW ,RP0
    JMP     _CLR_CHG_BANK
    JMP     _INIT_END

_CLR_CHG_BANK
    SET     PSW ,RP0            ;清除1区的RAM
    JMP     _START_CLR
_INIT_END
    CLR     PSW ,RP0
    CRET

.end
```

实验九、T2 启动 AD 转换采样

9.1 程序声明

KF8F系列单片机开发板演示程序

标 题: ADC转换采样

项 目 名: 9-T2_Start_ADC_TEST

开发环境: ChipON IDE

使用芯片: KF8F312

作 者:上海芯旺微电子技术有限公司

功能简述: 使用 T2 定时器每隔 100us 启动 AD 转换一次，并通过数码管显示转换结果。

9.2 硬件说明

本实验硬件连接和实验 8 的一样，都是对 AN2 通道进行 AD 转换采样，所以按照实验 8 的硬件进行连接即可。

9.3 程序设计说明

当对 ADCCTL0 寄存器的T2CCRON 位置1时，即可使能T2触发AD启动。在实验

8中，是通过在程序中对START位进行置1从而启动AD转换，而本实验是通过T2定时中断启动AD转换，其转换原理图如图9.1所示。

T2定时计数器配置为每隔2us加1，当T2的值和T2CCR的值相等时，如果对应的控制位置1（即T2CCRON = 1），则会发出信号，使ADCCTL0的START位置1，从而启动AD转换。本实验中 T2CCR = 0x32，所以可以推算出AD转换每隔100us进行一次。

程序配置过程中需注意以下几点：

1：使用T2启动AD转换，每转换完成一次，T2CCRON需重新置1才能进行连续转换，否则只能转换一次。

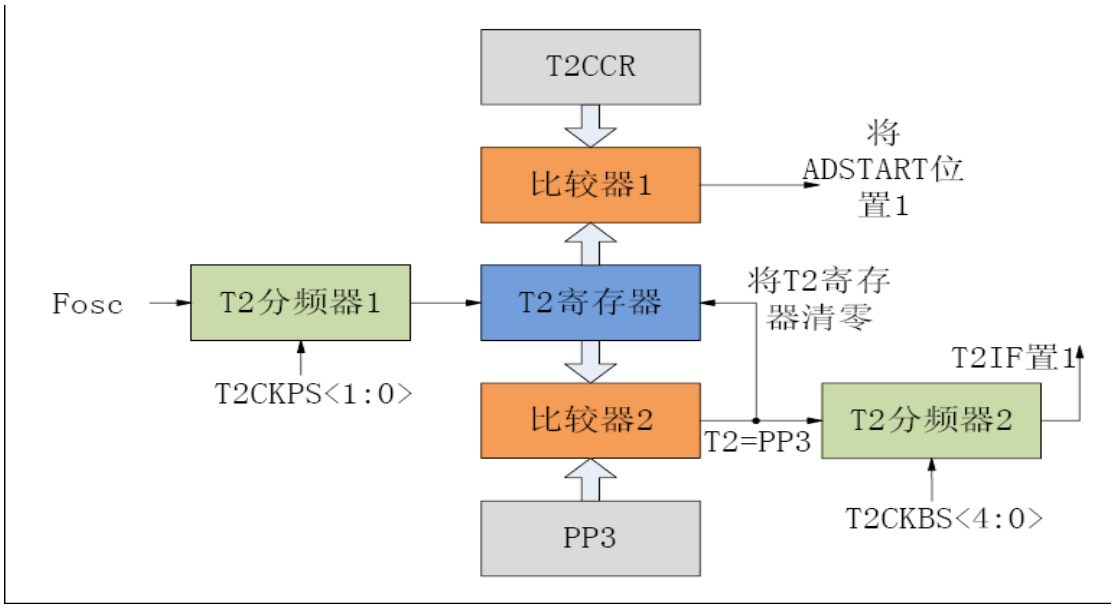


图 9.1

9.4 .1 C 程序代码

```
#include<KF8F312.h>

/***************宏定义*********************/
#define uchar unsigned char
#define uint  unsigned int

/******************宏定义结束*****************/

/**************定义全局变量（注：为节省内存空间，全局变量尽量不要赋初值）
```

```
*********/
uchar const Arr_num[] = {
                  0X3F,      //      ;0
                  0X06,      //      ;1
                  0X5B,      //      ;2
                  0X4F,      //      ;3
                  0X66,      //      ;4
                  0X6D,      //      ;5
                  0X7D,      //      ;6
                  0X07,      //      ;7
                  0X7F,      //      ;8
                  0X6F,      //      ;9
                  0X77,      //      ;A
                  0X7C,      //      ;B
                  0X39,      //      ;C
                  0X5E,      //      ;D
                  0X79,      //      ;E
                  0X71,      //      ;F
                  0X00       //      ;空
};                                    //定义数码管可以显示的0——f,共阴接法
uchar Unit, Decade, Hundred, Thousand;    //定义数码管的个十百千位
uchar flag;
uint Adc_sum = 0,dis_Num = 0,dis_Num_buf;              //adc采样累加变量
uchar Num = 0;
uint number = 0;
/**********************全局变量定义结束
************************************/

/************************
 * 函数名     : Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/

void Init_fun()
{
    OSCCTL = 0x60;                   //设置为8M

    /*****端口初始化*****/
    TR0 = 0x0C;                        //设置P03端口为输入  P02为AN2模拟通
道 配置为输入口
    TR1 = 0x00;                        //设置P1端口为输出
    TR2 = 0x00;                        //设置P2端口为输出
```

```
    P0 = 0;
    P1 = 0;
    P2 = 0;

    /****初始化AD寄存器****/
    ANSEL = 0x04;                  //设置通道2为模拟口
    ADCCTL0 = 0xC9;                    //右对齐，通道2， AD使能打开  使能T2触
发AD启动
    ADCCTL1 = 0x10;                    //8分频，VDD作为参考电压

    /****初始化T2寄存器****/
    T2CCR = 0X32;    // xxxus进行一次转换
    PP3 = 0XFF;        // T2每次分频功能下计数到值
    T2 = 0;

    EIF1 = 0;       // 清T2标志位 T2IF
    T2CTL = 0x7F;    // 启动T2  T2ON = 1  T2分频器1和分频器2都是16分频
    EIE1 = 0X40;     // 打开AD中断
    INTCTL = 0XC0;   // 打开外部中断 PUIE 使能总中断 AIE
}
/************************
 * 函数名     : Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         : 无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}

/************************
 * 函数名     : Display
 * 函数功能：数码管显示
 * 入口参数：显示数字
 * 返回         : 无
 ************************/
void Display()
  {
    uint i;
    for(i = 0; i < 500 ; i++)
     {
        P2 = 0XE0;            //打开数码管的 个位
```

```
        P1 = Arr_num[Unit];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐

        P2 = 0XD0;            //打开数码管的 十位
        P1 = Arr_num[Decade];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐

        P2 = 0XB0;            //打开数码管的 百位
        P1 = Arr_num[Hundred];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐

        P2 = 0X70;            //打开数码管的 千位
        P1 = Arr_num[Thousand];
        Delay_200us();
        P1 = 0;
        P2 = 0XF0;            //消隐
        }
}

void main()
{
    Init_fun();

    while(1)
    {
        AIE=0;
        dis_Num_buf=dis_Num;
        AIE=1;
        Unit = dis_Num_buf&0x0f;           //将Dis_adc的最低4位赋予数码管的
个位
        Decade = dis_Num_buf&0xf0;
        Decade >>= 4;                     //将Dis_adc的第7-4位赋予数码管的十
位
        dis_Num_buf >>= 8;                //右移8位，取Dis_adc的高8位
        Hundred = dis_Num_buf&0x0f;        //将此时的Dis_adc的最低4位赋予数
码管的百位
        Thousand = dis_Num_buf&0xf0;
        Thousand >>= 4;                   //将此时的Dis_adc的第7-4位赋予数码
```

```
管的千位
        Display();                          //显示
    }
}
//中断函数
void int_fun() __interrupt
{
    if (ADIF)                           // 转换完成后产生中断
    {
        ADIF = 0;
        number = ADCDATAH&0x03;
        number <<= 8;
        number |= ADCDATAL&0xFE;
        Adc_sum +=  number;
        Num++;
        if (Num == 8)
        {
            Num = 0;
            dis_Num = (Adc_sum+4)>> 3;
            Adc_sum = 0;
        }
        T2CCRON = 1;  //每次转换结束必须再次开启才可继续T2触发
    }
}
```

9.4.2 汇编程序代码

```
    .INCLUDE "KF8F312.INC"
;;;;;;;;;;;;;;;;通用寄存器;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
DELAY_NUM1      .EQU   0X81      ;延时用的临时变量
DELAY_NUM2      .EQU   0X82      ;延时用的临时变量
PSW_TEMP     .EQU   0X85      ;PSW的临时保存变量
PCH_TEMP     .EQU   0X86      ;PCH的临时保存变量
UNIT         .EQU   0X88      ;数码管个位
DECADE       .EQU   0X89      ;数码管十位
HUNDRED         .EQU   0X8A      ;数码管百位
THOUSAND     .EQU   0X8B      ;数码管千位
AD_H         .EQU   0X8C      ;采样得到的高位
AD_L         .EQU   0X8D      ;采样得到的低位
AD_TEMP         .EQU   0X8E      ;AD采样的临时变量
DIS_DEL         .EQU   0X8F      ;减缓采样速度用的变量
AD_H_SUM     .EQU   0X90      ;存放AD采样8次的高位
AD_L_SUM     .EQU   0X91      ;存放AD采样8次的低位
```

```
AD_8_T        .EQU   0X92      ;8次采样电压，求平均计数
RESULT_H      .EQU   0X93      ;采样8次平均得到的高位
RESULT_L      .EQU   0X94      ;采样8次平均得到的低位

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
    .ORG 0X0000
     NOP
     JMP MAIN                    ;入口
    .ORG 0X0004
    JMP INTERRUPT
    ;***********************************************************
*******************
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返    回: R0，数码管显示的编码值
;* 注  意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不
同页造成查表出错
;***********************************************************
****************
DIG_DATA
    ADD PCL, R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
    RRET R0 ,#0X7D      ;6
    RRET R0 ,#0X07      ;7
    RRET R0 ,#0X7F      ;8
    RRET R0 ,#0X6F      ;9
    RRET R0 ,#0X77      ;A
    RRET R0 ,#0X7C      ;B
    RRET R0 ,#0X39      ;C
    RRET R0 ,#0X5E      ;D
    RRET R0 ,#0X79      ;E
    RRET R0 ,#0X71      ;F
    RRET R0 ,#0X00      ;空

;;;;;;;;;;;;;;;;中断部
分;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERRUPT
```

```
;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1
    ;---------------------------------
    ;中断处理
    JB EIF1,ADIF
    JMP INT_END
    CLR EIF1,ADIF
    SET ADCCTL0,T2CCRON        ;AD转化完成，T2CCRON自动清零
ADC_LOOP_8
    MOV R1 ,ADCDATAH
    MOV AD_H, R1     ;10位AD转换结果的高2位
    MOV R1 ,ADCDATAL
    MOV AD_L, R1     ;10位AD转换结果的低8位

     MOV R1 ,AD_L
    ADD AD_L_SUM, R1           ;累加AD采样的低位
    JNB PSW ,CY
    INC AD_H_SUM
    MOV R1 ,AD_H             ;累加AD采样的高位
    ADD AD_H_SUM, R1
    INC AD_8_T
    MOV R1 ,AD_8_T
    XOR R1 ,#0X08              ;是否采样了8次进行判断
    JB  PSW,Z
    JMP INT_END

    CLR PSW,CY                 ;以下进行3次右移，即除以8 ， 取平均
    RRC AD_H_SUM
    RRC AD_L_SUM
    CLR PSW,CY
    RRC AD_H_SUM
    RRC AD_L_SUM
    CLR PSW,CY
    RRC AD_H_SUM
    RRC AD_L_SUM

    MOV R1 ,AD_H_SUM
    MOV RESULT_H, R1
    MOV R1 ,AD_L_SUM
    MOV RESULT_L, R1
```

```
    CLR AD_H_SUM
    CLR AD_L_SUM
    CLR AD_8_T

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;;;;;;;;;;;;;;中断部分结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;入口函数,主程
序;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN
    CALL INIT_RAM                ;初始化RAM
    CALL INIT_ALL               ;初始化寄存器
LOOP
    MOV R0 ,RESULT_H
    RRC R0
    RRC R0
    RRC R0
    RRC R0                  ;右移4位
    AND R0 , #0X0F
    MOV THOUSAND, R0            ;数码管千位的值

    MOV R0 ,RESULT_H
    AND R0 ,#0X0F
    MOV HUNDRED, R0             ;数码管百位的值

    MOV R0 ,RESULT_L
    RRC R0
    RRC R0
    RRC R0
    RRC R0                  ;右移4位
    AND R0 , #0X0F
    MOV DECADE, R0              ;数码管十位的值

    MOV R0 ,RESULT_L
    AND R0 ,#0X0F
    MOV UNIT, R0            ;数码管个位的值
```

```
C_DIS
    CALL LED_DISPLAY          ;数码管显示当前数值
    INC DIS_DEL               ;DIS_DEL是用来延时AD采样速度用的
    MOV R0 ,DIS_DEL
    XOR R0 ,#0X55
    JB PSW ,Z
    JMP C_DIS
    CLR DIS_DEL
    JMP LOOP
    CRET
;;;;;;;;;;;;;;主程序结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;***************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返      回: 无
;***************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0       ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0       ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0        ;设置为8M
    ;端口初始化
    MOV R0 ,#0x0C
    MOV TR0, R0         ;设置P03端口为输入  P02为AN2模拟通道 配置为
输入口
    MOV R0 ,#0x00
    MOV TR1, R0         ;设置P1端口为输出
    MOV R0 ,#0x00
    MOV TR2, R0         ;设置P2端口为输出
```

```
    CLR P0
    CLR P1
    CLR P2

    ;初始化AD寄存器组
    MOV R0 ,#0x04
    MOV ANSEL, R0       ;设置通道2为模拟口
    MOVR0 ,#0xC9    ;右对齐，通道2， AD使能打开
    MOV ADCCTL0, R0
    MOV R0 ,#0x10       ;8分频，VDD作为参考电压
    MOV ADCCTL1, R0

    MOV R0 ,#0X32       ;xxus进行一次转换
    MOV T2CCR, R0
    MOV R0 ,#0XFF
    MOV PP3, R0
    CLR T2

    CLR EIF1            ;清T2标志位 T2IF
    MOV R0 ,#0x7F       ;T2分频器1为16分频,分频器2为16分频
    MOV T2CTL, R0
    MOV R0 ,#0X40       ;打开AD中断
    MOV EIE1, R0
    MOV R0 ,#0XC0       ;打开外部中断 PUIE 使能总中断 AIE
    MOV INTCTL, R0

    CRET

;*************************************************************
*****************
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位 ,
每个数的范围 0~f
;* 返    回: 无
;*************************************************************
*****************
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0         ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
```

```
    CALL DELAY
    CLR P1          ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 ,#0XD0
    MOV P2 , R0         ;打开数码管的 十位
    MOV R2 , DECADE
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1          ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0XB0
    MOV P2 , R0         ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1          ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    MOV R0 , #0X70
    MOV P2 , R0         ;打开数码管的 千位
    MOV R2 , THOUSAND
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1          ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0
    CRET

;*************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
```

```
DELAY
    MOV R0 ,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0 ,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                ;子程序返回

;/*************************************************************
******************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X20    ;操作个数
    CALL    INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X01    ;操作个数
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
```

```
DECJZ R1
JMP  INIT_RAM_0
CRET

.end
```

实验十、PWM1/2输出实验

10.1 程序声明

KF8F系列单片机开发板演示程序

标 题：PWM1/2 输出实验

项 目 名：10-PWM1_PWM2_TEST

开发环境：ChipON IDE

使用芯片：KF8F312

作 者:上海芯旺微电子技术有限公司

功能简述：两路的PWM实验，在主程序中改变占空比，使得一路LED灯实验亮暗变化，一路LED亮度不变作为参考亮度。

10.2 硬件说明

KF8F312单片机具有2路8位的PWM模块PWM1/PWM2和1路10位增强型PWM3模块，本实验使用了PWM1和PWM2进行模数转化。PWM1/PWM2分别复用了IO引脚P10，P11，那么只需将J13，J14的1脚端通过杜邦线与P10，P11连接即可，接线图如图10.2所示。

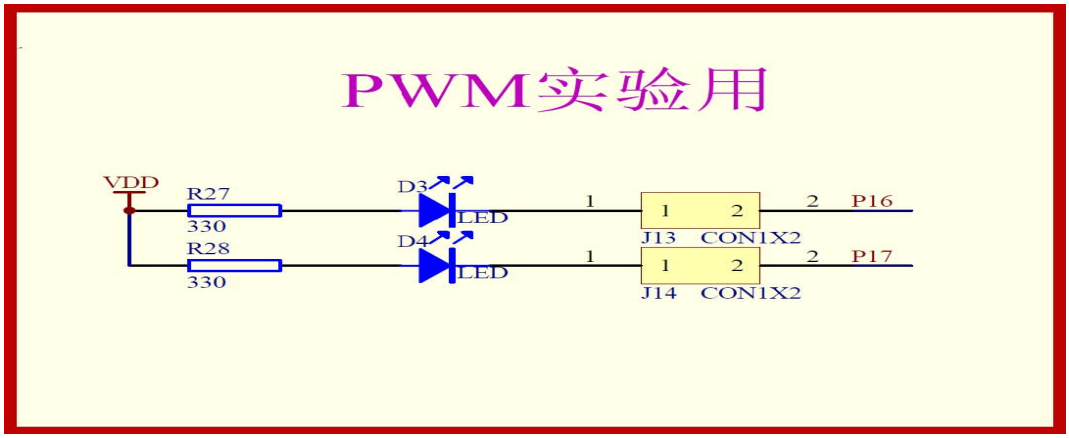


图 10.1

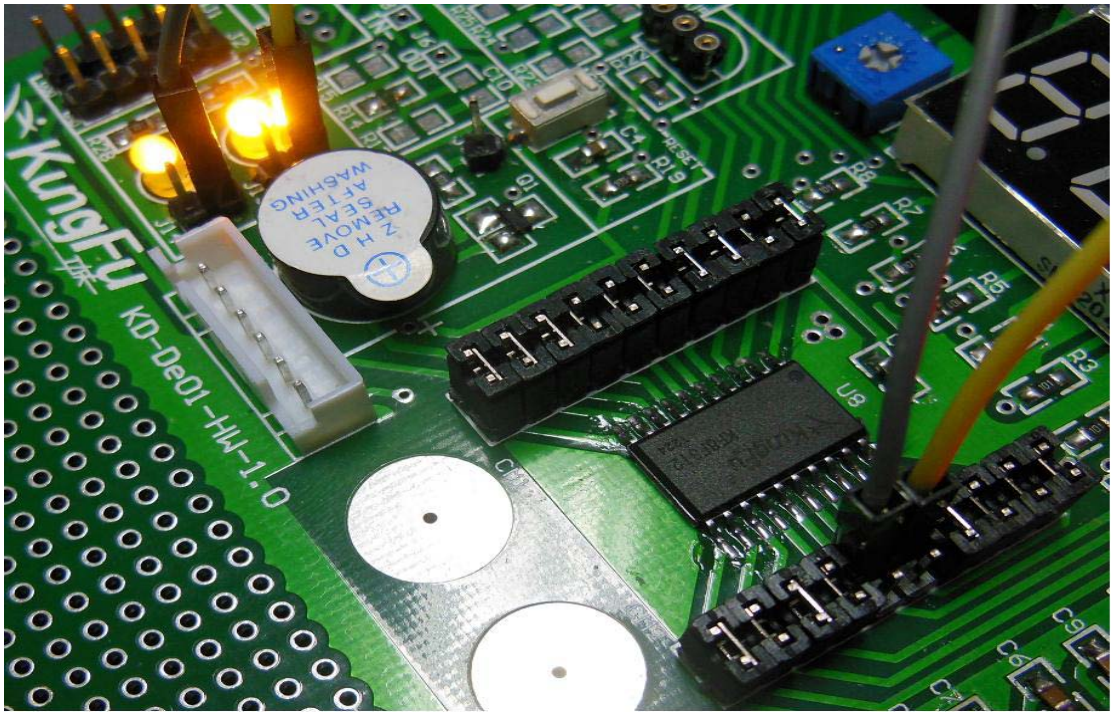


图 10.2：需要连接的跳线图

10.3 程序设计说明

本实验主要是通过PWM输出方波脉冲改变LED灯的变化亮度，启动PWM后，在对应的PWM1(或PWM2)引脚输出PWM脉冲。PWM脉冲的频率和占空比通过PP1(或PP2)和PWM1L(或PWM2L)设置。PP1为PWM1模块的周期寄存器，PWM1L为PWM1模块占空比设置寄存器，当T1L/H与PP1/2匹配时，会触发相应的中断标志位T1IF和PWM2IF。如果使能T1IE或者PWM2IE，则会触发中断（AIE、PUIE置1）。PP2/PWM2L同理。改变PP1和PWM1L的值可产生不同的PWM1周期和PWM1占空比。PWM2模块的工作原理和PWM1模块完全一致。需要注意的是，使用PWM时需要将定时器1配置给PWM做定时用，其中T1L、T1IE和T1IF分配给PWM1,T1H分配给PWM2。例如启动PWM1后，当T1L计数值和PP1相等时，P1.0引脚被置1，此时T1L被清0，重新开始计数，当T1L的计数值和PWM1L相等时,P1.0引脚清0,如下图10.3所示。用户可以参阅《KF8F312数据手册》PWM章节。

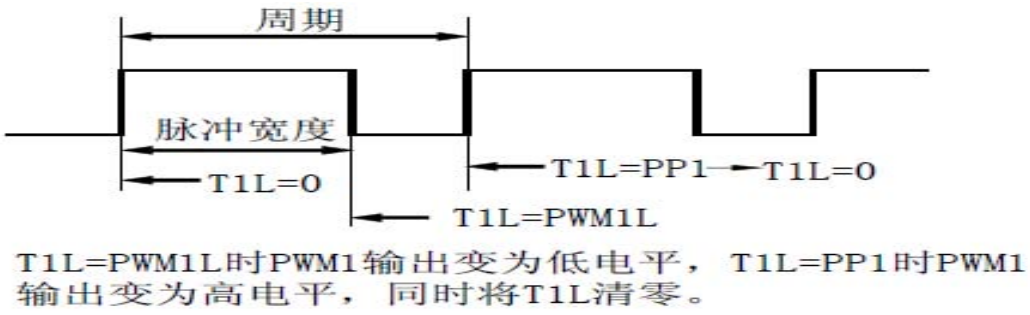


图 10.3：PWM1/2 输出波形图

PWM 周期通过 PP1/PP2 寄存器计算，计算公式为：

PWM 周期=(PPx+1)*4*Tosc*(T1 预分频比) (x=1、2)

PWM 的脉冲宽度可通过 PWM1L/PWM2L 寄存器计算得出，计算公式如下：

脉冲宽度=PWMxL*4*Tosc*(T1 预分频比) (x=1、2)

PWM 的占空比等于脉冲宽度与周期的比值。那么 PWM 的占空比计算公式为：

占空比=脉冲宽度/PWM 周期=PWMxL/(PPx+1) (X=1、2)

PWM 还涉及到分辨率的概念，PWM 分辨率即单个周期内的占空比个数，KF8F312 的 PWM1/PWM2 的分辨率最高为 8 位，可以通过 PP1/PP2 寄存器来设置，例如，PP1 为最大值 255，那么 PWM1 的分辨率就为 8 位的。

综上：PWM 的设置可以参照以下步骤：

1、 赋 PP1 或 PP2 寄存器的初值以设置 PWM1 或 PWM2 的 PWM 周期。

2、 赋 PWM1L 或 PWM2L 寄存器的初值以设置 PWM1 或 PWM2 的占空比。

3、 配置并启动定时器/计数器 T1:

- 配置 T1CTL 寄存器的 T1CKS1 和 T1CKS0 以选择 T1 的预分频比；
- 将 T1L/H 清 0；
- 将 T1CTL 寄存器的 T1ON 位置 1 以启动 T1；

4、 将 PWMCTL 寄存器的 PWM1ON 或 PWM2ON 置 1 以启动 PWM1 或 PWM2。

5、 将 TR10 或 TR11 清 0 使能引脚 P1.0/PWM1 或 P1.1/PWM2 的输出驱动器。

10.4 .1 C 程序代码

```
#include<kf8f312.h>

/*********************宏定义*****************************/
#define uchar unsigned char
```

```c
#define uint  unsigned int

uchar   Control_Way;
/*********************宏定义结束************************/

/******************函数声明*****************************/
void Init_fun();
void Delay_ms(uint ms_data);
/******************函数声明结束*************************/

/************************
 * 函数名     : main
 * 函数功能：程序入口主函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void main()
{
    Init_fun();              //初始化函数

    PWM1L = 0;             //初始化调光灯亮度
    PWM2L = 0xF0;          //参考灯的固定占空比下亮度，值从0-255
    Control_Way=1;             //默认调光方向增亮
    PWMCTL = 0x03;             //使能PWM1,PWM2

    while(1)
    {
        Delay_ms(30);
        //修改占空比，实现亮度变化
        if(Control_Way)
        {
            if(PWM1L<254)      //有效可区分占空阀上限
            {
                PWM1L+=2;
            }
            else
            {
                Control_Way=0;
                Delay_ms(1000);   //亮度翻转点暂停一定时间
            }
        }
        else
        {
            if(PWM1L>128)      //有效可区分占空阀下限
```

```
            {
                PWM1L-=2;
            }
            else
            {
                Control_Way=1;
                Delay_ms(1000);   //亮度翻转点暂停一定时间
            }
        }
    }
}
/************************
 * 函数名     : Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun()
{
    OSCCTL = 0x50;          //设置为4M
    /***端口初始化****/
    TR0 = 0x08;               //设置P0端口为输入
    TR1 = 0x00;               //设置P1端口为输出
    TR2 = 0x00;               //设置P2端口为输出
    P0 = 0;
    P1 = 0;
    P2 = 0xF0;
    /***PWM相关寄存器初始化****/
    T1CTL = 0X01;
    T1L = 0;
    T1H = 0;
    PP1 = 0xff;               //PWM1的周期寄存器 ， 分辨率为255
    PP2 = 0xff;               //PWM2的周期寄存器 ， 分辨率为255

    PWMCTL = 0;
}
/************************
 * 函数名     : Delay_ms
 * 函数功能：长时间延时
 * 入口参数：延时基数 uchar ms_data
 * 返回         : 无
 ************************/
void Delay_ms(uint ms_data)
{
```

```
    uchar i;
    while(ms_data--)
    {
        i = 124;
        while(i--);
    }
}
/************************
 * 函数名     : int_fun
 * 函数功能: __interrupt
 * 入口参数:
 * 返回         :
 ************************/
void int_fun() __interrupt
{

}
```

10.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

DELAY_NUM0              .EQU        0X80      ;PWM占空比计数
DELAY_NUM1              .EQU        0x81      ;延时用的临时变量
DELAY_NUM2              .EQU        0x82      ;延时用的临时变量
PSW_TEMP            .EQU        0X85      ;PSW的临时保存变量
PCH_TEMP            .EQU        0X86      ;PCH的临时保存变量
Control_Way             .EQU        0X87      ;PCH的临时保存变量
    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

INTERRUPT
;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
**********************************/
```

```
    JB EIF1,PWM2IF
    JMP INT_END
    CLR EIF1,PWM2IF

;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;*************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: DELAY_NUM0
;* 返    回: 无
;*************************************************************
*****************
DELAY
    MOV R0 ,#0x12
    MOV DELAY_NUM1, R0
LP0
    MOV R0 ,#0X12
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0

    DECJZ DELAY_NUM0
    JMP DELAY

    CRET                ;子程序返回

;/*********************************主函数程序
**********************************/

MAIN
    CALL INIT_ALL               ;初始化寄存器
```

```
    CALL INIT_RAM                ;初始化RAM

    CLR PWM1L                    ;设置PWM1的占空比
    MOV R0,#0XF0
    MOV PWM2L,R0                 ;设置PWM2的占空比
    MOV R0,#0x03                 ;使能PWM1,PWM2
    MOV PWMCTL,R0

    SET Control_Way,0

LOOP
    ;; delay 30ms
    MOV R0,#0x1E
    MOV DELAY_NUM0,R0
    CALL DELAY

    JB Control_Way,0
    JMP PWM_DEC
;当没有增加到设定大时，每次增加2个单位，当达到时切换保持一定时间后切换方向
PWM_INC
    MOV R0,PWM1L
    SUB R0,#0xFD
    JB PSW ,CY
    JMP Way_INC_Switch
    ;// 累加
    INC PWM1L
    INC PWM1L
    JMP PWM_Deal_End
    ;//延迟并换向
Way_INC_Switch
    CLR Control_Way,0
    ;; delay 1000ms
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
    CALL DELAY
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
    CALL DELAY
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
    CALL DELAY
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
```

```
    CALL DELAY
    ;;
    JMP PWM_Deal_End
;当没有减小到设定大时，每次减小2个单位，当达到时切换保持一定时间后切换方向
PWM_DEC
    MOV R0,PWM1L
    SUB R0,#0x80
    JNB PSW,CY
    JMP Way_DEC_Switch
    DEC PWM1L
    DEC PWM1L
    JMP PWM_Deal_End
Way_DEC_Switch
    SET Control_Way,0
    ;; delay 1000ms
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
    CALL DELAY
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
    CALL DELAY
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
    CALL DELAY
    MOV R0,#0xFA
    MOV DELAY_NUM0,R0
    CALL DELAY
PWM_Deal_End
    JMP LOOP
    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
```

```
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X08
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0     ;数码管模块关闭
    MOV P2,R0

    MOV R0,#0X01
    MOV T1CTL,R0
    CLR T1L
    CLR T1H
    MOV R0,#0XFF      ;PWM1的周期寄存器 ， 分辨率为255
    MOV PP1,R0
    MOV R0,#0XFF      ;PWM2的周期寄存器 ， 分辨率为255
    MOV PP2,R0

    CLR PWMCTL

    CRET

;/*************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
****************/
INIT_RAM
    CLR    R2        ;传递默认RAM值
```

```
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .END
```

实验十一、PWM3 单输出模式实验

11.1 程序声明

KF8F 系列单片机开发板演示程序

标　　题：PWM3 单输出模式实验

项 目 名：11-PWM3_Single_Output_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：设置PWM3单输出模式，使P3A、P3B、P3C和P3D四个端口同时输出同一个PWM信号，控制同一路LED从亮到灭的变化过程。

11.2 硬件说明

使用杜邦线把单片机的 P3A、P3B、P3C 和 P3D 端口分别与 D3 LED 相连，即可观察相应的实验现象，如 P3A 和 D3 连接如图 11.7 所示。

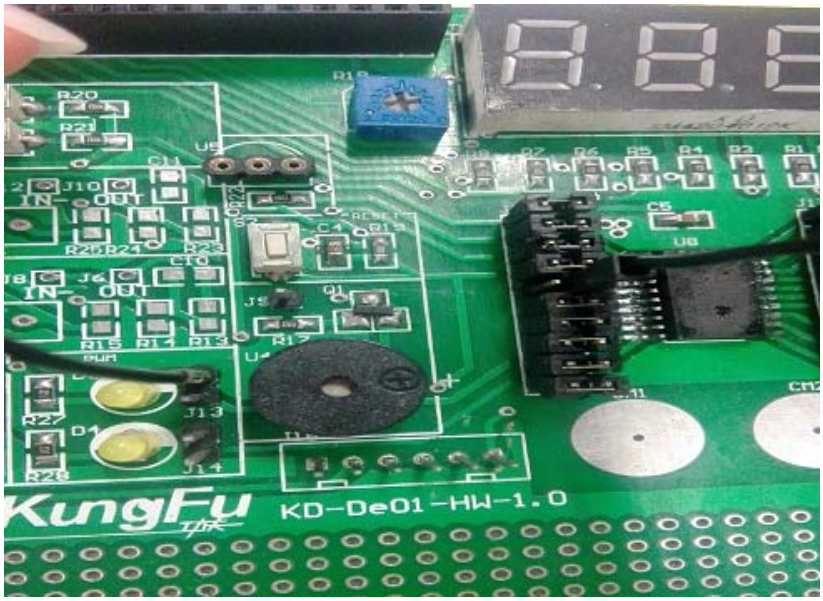


图 11.7　P3A 和 D3 连接图

11.3 程序设计说明

PWM3与PWM1/2不同：带有死区控制延时功能，占空比设置寄存器为10位，PWM3最多可在4个不同的引脚(P3A、P3B、P3C和P3D)输出PWM信号，分辨率最高10位；PWM3有4种输出模式：单输出、半桥输出、全桥正向输出模式和全桥反向输出模式，通过寄存器PWM3CTL0中的P3M<1:0>位选择4种输出模式之一。通过将寄存器PWM3CTL0中的P3M<1:0>位设置为00，选择单输出模式，在此模式下，默认从P3A引脚输出PWM信号，P3B、P3C和P3D引脚为通用端口引脚。使用单模式输出时，通过设置寄存器PATRCTL的STREN<D:A>位，可以使得P3A、P3B、P3C和P3D同时输出四路相同的PWM信号。

如图11.1，P3A一路设置为PWM输出和P3A和P3B两路同时设置为PWM输出的示例，其它设置情况与此类似。

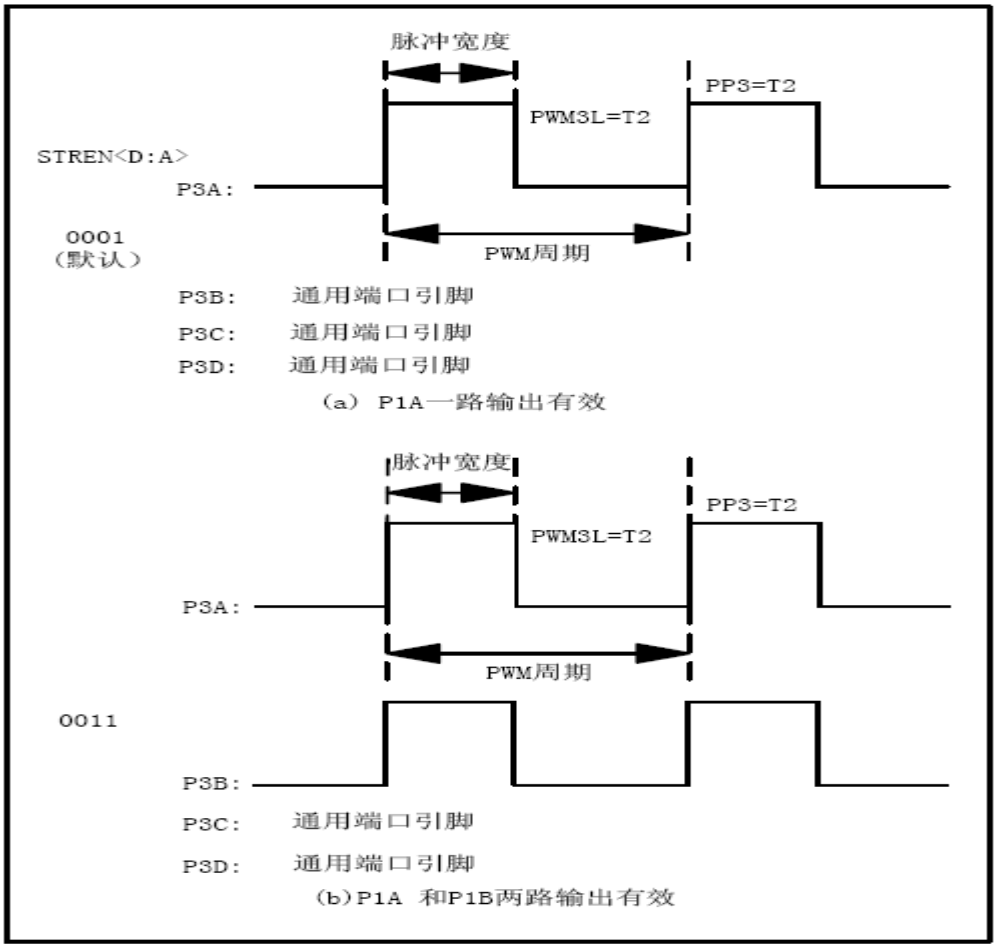


图 11.1 PWM3的输出示例

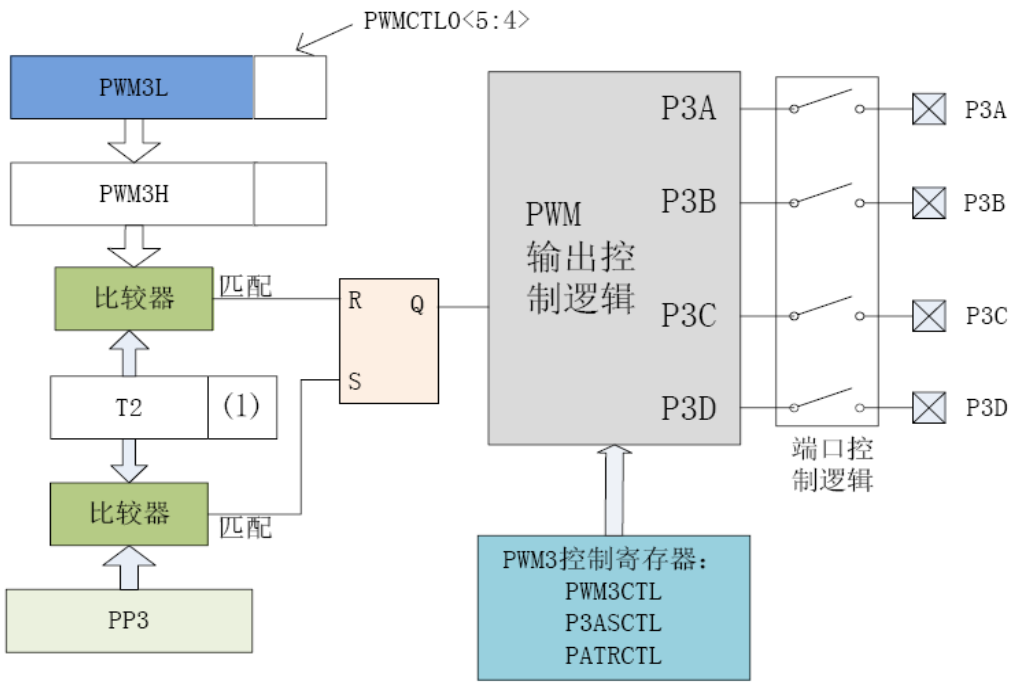


图11.2 PWM3原理框图

PWM3周期的计算：PWM3的周期通过8位的寄存器PP3 (地址:52H)进行设置，其值可设置为0～255，PWM3的周期通过如下公式进行计算。

★ 式8.5: PWM周期=(PP3+1)・4・Tosc・(T2预分频比)

PWM3占空比的计算：PWM3占空比设置寄存器为10位，通过寄存器PWM3L(地址:55H)和PWM3CTL0的PDT<1:0>位进行设置，PWM3L为占空比的高8位，PDT<1:0>为低两位。脉冲宽度和占空比通过如下两个公式计算:

★ 式8.6: 脉冲宽度=(PWM3L:PWM3CTL0<5:4>)・Tosc・(T2预分频比)

★ 式8.7: $$占空比=\frac{脉冲宽度}{PWM周期}=\frac{PWM3L:PWM3CTL0<5:4>}{4(PP3+1)}$$

单输出模式程序配置过程需注意以下几点：

1、 配置P3A、P3B、P3C和P3D端口输出PWM波形时，需要把P3A、P3B、P3C和P3D四个端口设置为数字输出口，即将相应的TRX寄存器的位置0。

2、 关于PWM3M<1:0> 配置有效电平的说明，如PWM3M<1:0> = 00，则配置P3A、P3B、P3C和P3D均为高电平有效，此时如果PWM3输出波形周期为200us，占空比为60%，则一个周期波形中高电平为120us，低电平为80us，如图11.3所示。如PWM3M<1:0> = 11，则配置P3A、P3B、P3C和P3D均为低电平有效，同样的情况，则低电平为120us，高电平为80us，如图11.4所示。

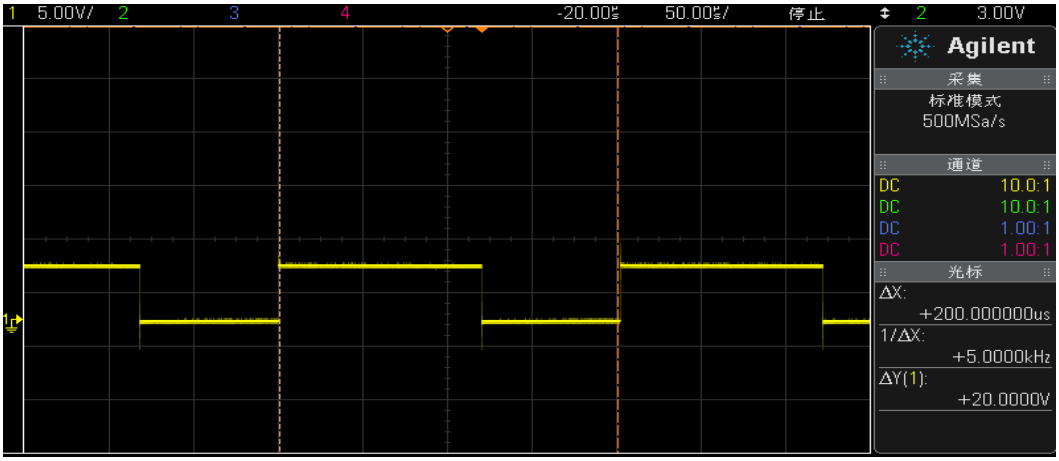


图 11.3 PWM3M<1:0> = 00

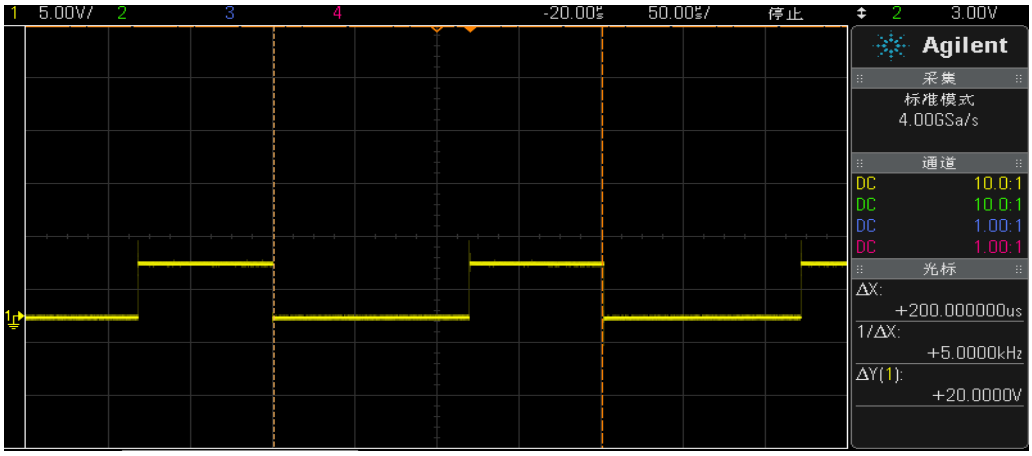


图 11.4 PWM3M<1:0> = 11

3、 PWM3的脉冲宽度计算公式中相比PWM1/2的脉冲宽度计算公式少乘了一个4，其它是一样的。

4、 关于转向同步位STRSYNC的操作，通过寄存器PATRCTL中的STRSYNC位进行设置引脚输出切换时是否与指令同步。

STRSYNC=1时，对应引脚P3x输出的PWM信号在STRENx置1后且PWM3输出信号周期结束时输出，如图11.5所示，黄色波形信号是等待绿色波形信号周期结束时才输出的；

STRSYNC=0时，对应引脚P3x输出的PWM信号在STRENx置1后立即输出，如图11.6所示，软件中一旦STRENx置1，则黄色波形信号立即输出，不一定等到绿色波形信号周期结束。

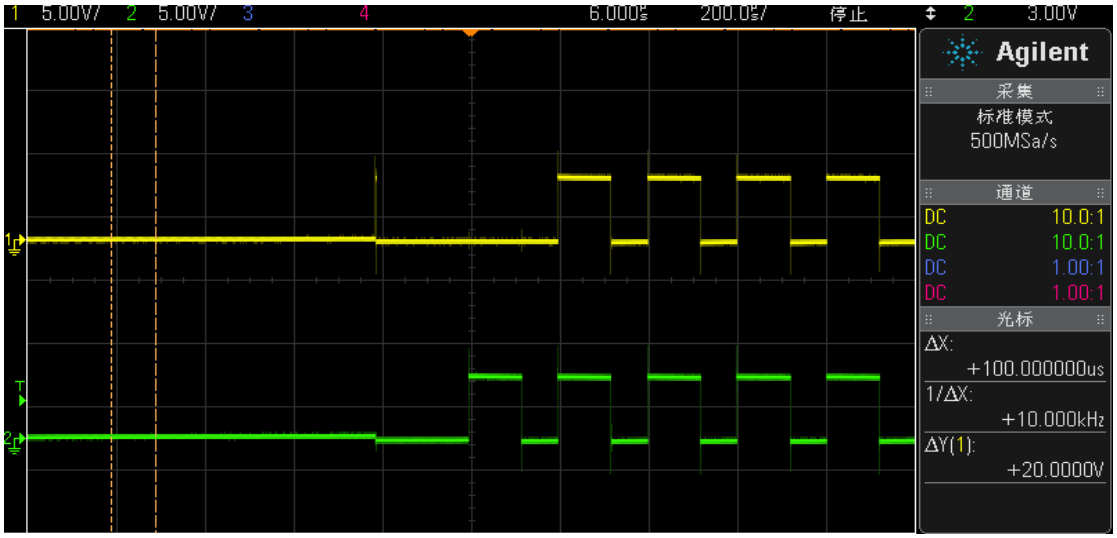


图 11.5 STRSYNC = 1

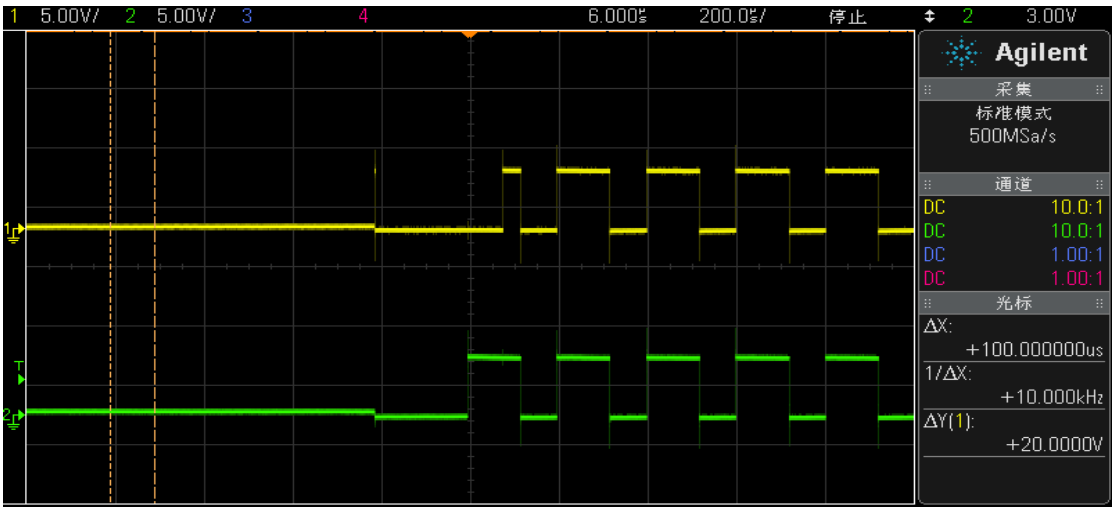


图 11.6 STRSYNC = 0

11.4 .1 C 程序代码

```
#include<KF8F312.h>

/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int

/*********************宏定义结束*************************/

/************************
 * 函数名     : Delay_ms
 * 函数功能：长时间延时
 * 入口参数：延时基数 uchar ms_data
 * 返回       : 无
 ************************/
void Delay_ms(uchar ms_data)
{
    uchar i;
    while(ms_data--)
    {
        i = 124;
        while(i--);
    }
}
/************************
 * 函数名     : Init_fun
```

```
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         ：无
 ************************/
void Init_fun()
{
    OSCCTL = 0X60;      // 8M时钟

    TR0 = 0X08;         // P03口必须设置为输入口
    TR1 = 0X00;
    TR2 = 0X00;

    P0 = 0x00;
    P1 = 0x3C;
    P2 = 0xF0;

//==========PWM3 int================//
    PWM3CTL1=0x00;    // 不开启自动使能，死区仅对半桥有效。
    P3ASCTL=0x00; // 禁止自动关闭源，默认关闭状态为1。
    PATRCTL=0x1F; // 开启转向同步，默认P3A输出，0x11 P3A 0x12 P3B  0x14 P3C
0x18 P3D
    PWM3CTL0=0x00;  // 单输出模式，暂不开启，默认有效状态高电平。
    // 占空比初始化 25%
    PWM3L=0x3F;
    PDT1=1;         // IN PWM3CTL0 bit 5
    PDT0=1;         // IN PWM3CTL0 bit 4
    PP3=0xFF;       // 4倍数值构成PWM周期最小单位
    // PWM使能
    T2=0;
    T2CTL=0x04;         // 设定计数采用1:1，启动T2计数
    P3ON1=1;        // IN PWM3CTL0 bit 3
    P3ON0=1;        // IN PWM3CTL0 bit 2
    //=======PWM3 int end============//
}
/************************
 * 函数名     ：main
 * 函数功能：主函数
 * 入口参数：无
 * 返回         ：无
 ************************/
void main()
{
    Init_fun();
```

```
    while (1)
    {
        Delay_ms(50);
        if(PWM3L<0xEF)
            PWM3L++;
        else
            PWM3L=0x00;
    }
}
/************************
 * 函数名     : int_fun
 * 函数功能: __interrupt
 * 入口参数: 无
 * 返回         : 无
 ************************/
void int_fun() __interrupt
{

}
```

11.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

NumL            .EQU        0x80      ;占空比计数低位
NumH            .EQU        0x81      ;占空比计数高位
DELAY_NUM1          .EQU        0X82      ;延时用的临时变量
DELAY_NUM2          .EQU        0X83      ;延时用的临时变量
PSW_TEMP        .EQU        0X85      ;PSW的临时保存变量
PCH_TEMP        .EQU        0X86      ;PCH的临时保存变量

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

INTERRUPT
;;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1
```

```
;/*********************************中断处理程序
**********************************/



;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET

;/*********************************主函数程序
**********************************/
MAIN
    CALL INIT_ALL                ;初始化寄存器
    CALL INIT_RAM                ;初始化RAM
LOOP
    ;//占空数+1
    INC NumL
    JNB PSW,Z
    INC NumH
    ;// 0x3FF 后回0
    MOV R0,#0x03
    XOR R0,NumH
    JB PSW,Z
    JMP SETPWM
    MOV R0,#0xFF
    XOR R0,NumL
    JB PSW,Z
    JMP SETPWM
    CLR NumL
    CLR NumH
SETPWM
    ;;//修改占空比,计算高位
    MOV R0,NumL
    RRC R0
    RRC R0
    JB  NumH,0
    CLR R0,6
```

```
    JNB NumH,0
    SET R0,6
    JB  NumH,1
    CLR R0,7
    JNB NumH,1
    SET R0,7
    ;;占空比赋值
    MOV PWM3L,R0

    JB  NumL,1
    CLR PWM3CTL1,PDT1
    JNB NumL,1
    SET PWM3CTL1,PDT1
    JB  NumL,0
    CLR PWM3CTL1,PDT0
    JNB NumL,0
    SET PWM3CTL1,PDT0
    ;;延迟间隔
    CALL DELAY
    JMP LOOP
    CRET
;*****************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X08         ;P03口必须设置为输入口
```

```
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0
    MOV P2,R0        ;LED模块不显示

    MOV R0,#0x1F
    MOV PATRCTL,R0       ;开启转向同步，默认P3A输出，0x11 P3A 0x12 P3B
0x14 P3C 0x18 P3D
    CLR T2               ;清T2计数器
    MOV R0,#0XFF
    MOV PP3,R0           ;设置PWM3周期

    MOV R0,#0X05
    MOV T2CTL,R0         ;启动T2，分频器1为4分频
    MOV R0,#0X0C
    MOV PWM3CTL0,R0      ;单输出模式     打开PWM3   输出高电平有效

    CRET

;/*************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************/
INIT_RAM
    CLR    R2        ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X20    ;操作个数
    CALL    INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X01    ;操作个数
```

```
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

;*****************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
DELAY
    MOV R0 ,#0x36
    MOV DELAY_NUM1, R0
LP0
    MOV R0 ,#0X36
    MOV DELAY_NUM2, R0
LP1
    DECJZ DELAY_NUM2       ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1       ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                ;子程序返回

    .END
```

实验十二、PWM3 带死区半桥输出模式实验

12.1 程序声明

KF8F 系列单片机开发板演示程序

标　　题: PWM3 带死区半桥输出模式实验

项 目 名: 12-Half_Bridges_With_Dead_Band_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述: 设置 PWM3 为半桥输出模式，带死区延时功能。PWM3 输出波形周期为 200us，占空比为 50%，死区延时为 7.5us。P3A 和 P3B 被配置为调制输出，PWM 输出信号在 P3A 引脚上输出，而互补的 PWM 输出信号在 P3B 引脚上输出。

12.2 硬件说明

本实验使用到了PWM3的半桥输出模式，只需使用示波器观察P3A和P3B端口的波形即可。

12.3 程序设计说明

通过将寄存器PWM3CTL0的P3M<1:0>位设置为10，把PWM3设置为半桥输出模式。此模式的周期和占空比设置方式和单输出模式一样。在此模式下，P3A和P3B被配置为调制输出，来驱动推挽式负载，P3C和P3D被配置为通用端口。PWM输出信号在P3A引脚上输出，而互补的PWM输出信号在P3B引脚上输出。当未使用死区延时功能时，PWM3半桥输出的波形如图12.1，周期为200us　占空比为50%，黄色为P3A端口波形，绿色为P3B端口波形。

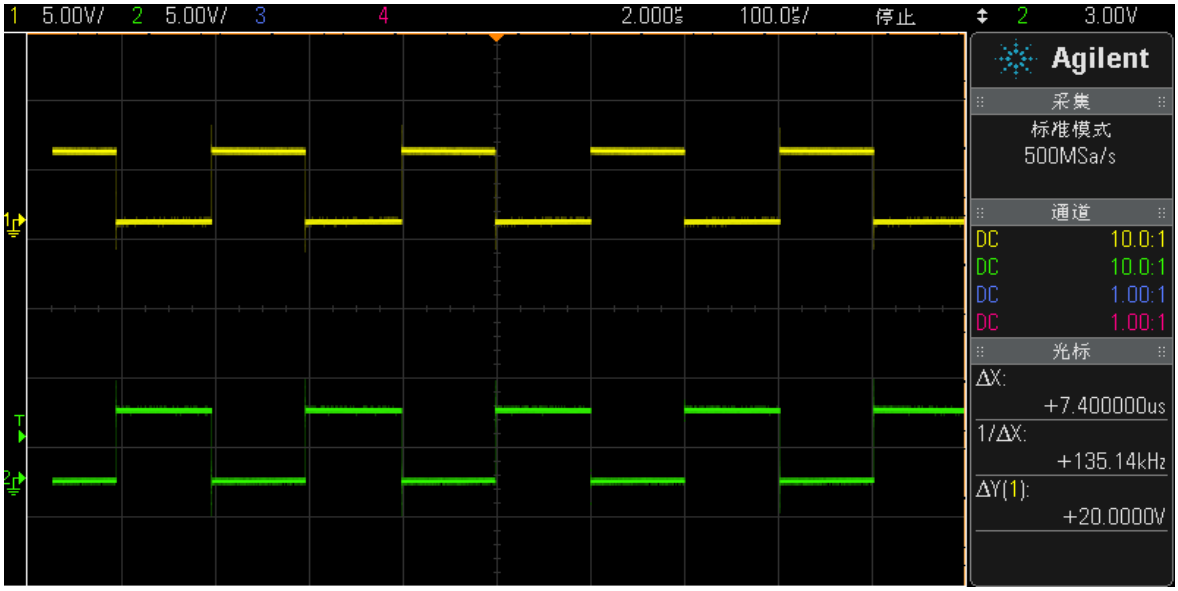


图 12.1　不带死区延时功能

半桥输出模式具有可编程的死区延时功能，由于外部电路种的开关管等元件导通和截止时间存在差异，可用来防止在半桥驱动电路中产生直通电流，损坏相关电路。PWM3CTL1 寄存器中PDC<6:0> 位的值用来设置死区延时时间，延时时间计数公式如下：

延时时间 = PDC<6:0>•Tosc * 4

如图12.2所示，P3A和P3B一直以 PWM 频率调制两个开关管，通常开关管的截止比导通需要更多的时间。如果QA和QB同时导通，两个管子可能会在一段很短的时间内都处于导通状态，在这很短的时间内，将会产生很大的电流流过两个管子，从而可能导致电路损坏。为了避免开关期间产生这种具有破坏性的直通电流，可使其中一个管子关闭后再打开另一个管子。在半桥输出模式下，使用一个可编程死区延时模块，来避免产生的直通电流破坏电路。由图可知，该延时在PWM3信号从非有效电平到有效电平转换时发生。

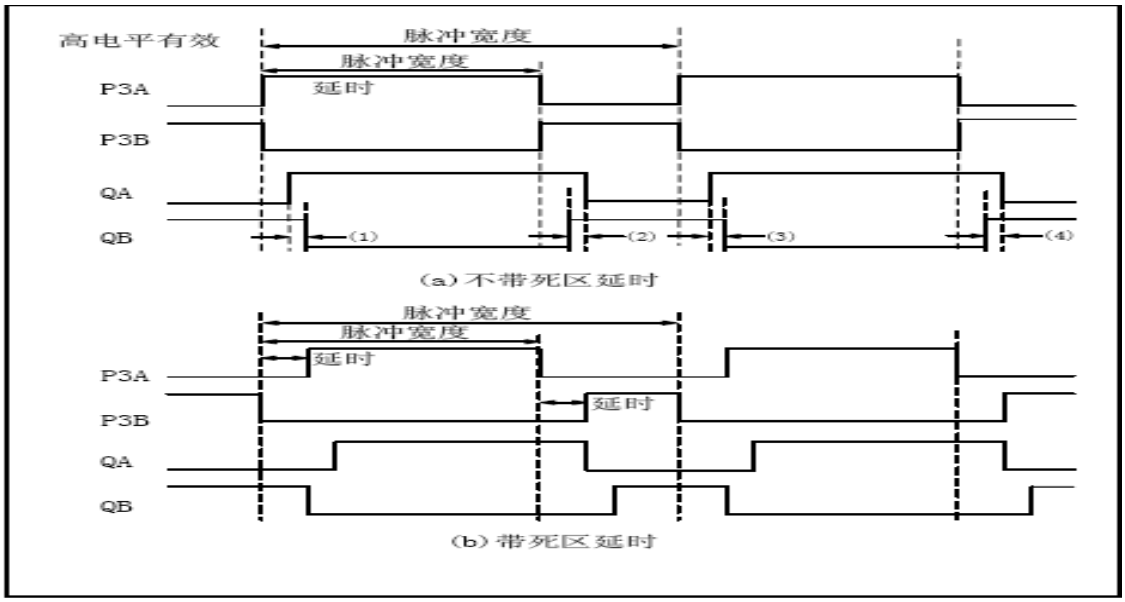
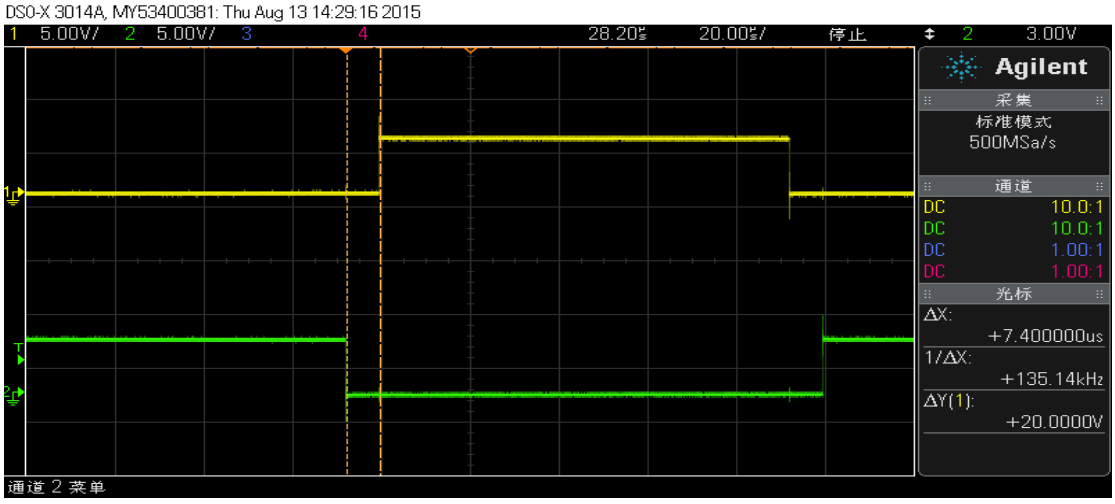


图 12.2

例如，当前系统时钟频率为 8M，PWM3CTL1 = 0x0F。Tosc = 1/8M = 0.125us，根据公式可算出延时时间 = 15*0.125us*4 = 7.5us。根据示波器波形可看出延时为 7.4us，在误差允许范围内约等于 7.5us。



12.4 .1 C 程序代码

```
#include<KF8F312.h>

/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int
```

```
/*********************宏定义结束************************/
/************************
 * 函数名     : Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/

void Init_fun()
{
    OSCCTL = 0X60;      // 8M时钟

    TR0 = 0X08;         // P03口必须设置为输入口
    TR1 = 0X00;         // 设置P3A、P3B为输出模式
    TR2 = 0X00;

    P0 = 0x00;
    P1 = 0x3C;
    P2 = 0x00;

    PWM3CTL0 = 0X8c; // 设定为半桥输出  启动PWM3  各引脚高电平有效  占空比低两位全部赋0
    // 模式设定
    PWM3CTL1=0x0F;    // 不开启自动使能，死区时间设定为0x0f   即当前延时T = 15*(1/(8M/4(T2分频比))) = 15*500ns = 7.5us
    P3ASCTL=0x00; // 禁止自动关闭源，默认关闭状态为1。
    PATRCTL=0x10; // 开启转向同步。
    PWM3CTL0=0x80;  // 半桥输出模式，暂不开启，默认有效状态高电平。
    // 占空比初始化 50%
    PWM3L=0x32;
    PDT1=0;           // IN PWM3CTL0 bit 5
    PDT0=0;           // IN PWM3CTL0 bit 4
    PP3=0x64;      // 4倍数值构成PWM周期最小单位,周期为200us
    // PWM使能
    T2=0;
    T2CTL=0x05;       // 启动T2，分频器1为4分频
    P3ON1=1;       // IN PWM3CTL0 bit 3
    P3ON0=1;       // IN PWM3CTL0 bit 2
//=======PWM3 int end============//
}
void main()
{
    Init_fun();
```

```
    while (1);
}


//中断函数
void int_fun() __interrupt
{

}
```

12.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

PSW_TEMP            .EQU        0X85      ;PSW的临时保存变量
PCH_TEMP            .EQU        0X86      ;PCH的临时保存变量

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

INTERRUPT
;;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
*********************************/



;/*********************************中断处理程序
*********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
```

```
    MOV PSW, R1
    IRET

;/*********************************主函数程序
**********************************/
MAIN
    CALL INIT_ALL                ;初始化寄存器
    CALL INIT_RAM                ;初始化RAM

LOOP
    JMP LOOP
    CRET
;*****************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X08         ;P03口必须设置为输入口
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0
```

```
    MOV P2,R0

    MOV R0,#0X64
    MOV PP3,R0              ; 设置PWM3周期为200us
    MOV R0,#0X32
    MOV PWM3L,R0            ;占空比50%
    MOV R0,#0X05
    MOV T2CTL,R0            ;启动T2，分频器1为4分频
    MOV R0,#0X8c
    MOV PWM3CTL0,R0         ;设定为半桥输出  启动PWM3  各引脚高电平有
效   占空比低两位全部赋0
    MOV R0,#0X0F
    MOV PWM3CTL1,R0         ;死区时间设定为0x0f   即当前延时T =
15*(1/(8M/4(T2分频比))) = 15*500ns = 7.5us

    CRET

;/*************************************************************
******************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
```

```
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .END
```

实验十三、PWM3 全桥输出模式实验

13.1 程序声明

KF8F 系列单片机开发板演示程序

标　　题：PWM3 全桥输出模式实验

项 目 名：13-PWM3_Full_Bridge_Output_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：设置PWM3为全桥正向输出模式，使能自动关断和自动重启功能，设定INT0为关断源，当INT0（P02）为低电平时，自动关断功能自行启动，该功能启动后禁止PWM3输出，并将P3A、P3B、P3C和P3D四个引脚输出电平置于其预定义的状态。当INT0（P02）为高电平时，自动重启功能启动，PWM3又回到正常输出模式。

13.2 硬件说明

本实验使用到了 PWM3 的全桥正向输出模式，使用示波器观察 P3A、P3B、P3C 和 P3D 端口的波形，当 INT0（P02）与 GND 断开连接时，PWM3 工作在正常输出模式，当使用杜邦线将 INT0（P02）与 GND 连接时，自动关闭功能启动，各端口输出电平为其预定义状态，当 INT0（P02）与 GND 再次断开时，自启动功能启动，各端口恢复到正常输出。

13.3 程序设计说明

全桥输出模式有全桥正向输出模式和全桥反向输出模式两种。通过将寄存器PWM3CTL0的P3M<1:0>设置为01，把PWM3设置为全桥正向输出模式；将其设置为11，把PWM3设置为全桥反向输出模式。

在全桥输出模式下，P3A、P3B、P3C和P3D四个引脚都用作输出，各个端口在不同模式下的电平输出入表13.1所示。高电平有效时，引脚信号示例如图13.1所示，低电平有效时，引脚信号示例如图13.2所示。

	P3A	P3B	P3C	P3D
全桥正向输出模式	有效电平	无效电平	无效电平	PWM调制信号
全桥反相输出模式	无效电平	PWM调制信号	有效电平	无效电平

表 13.1

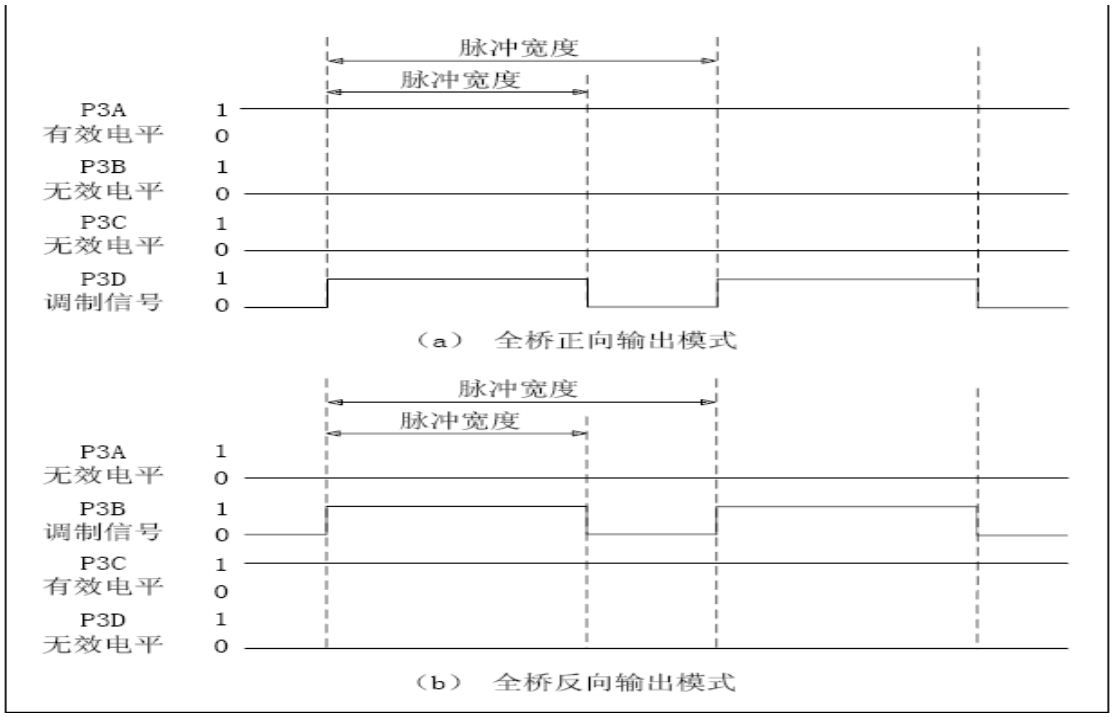


图13.1（高电平有效）

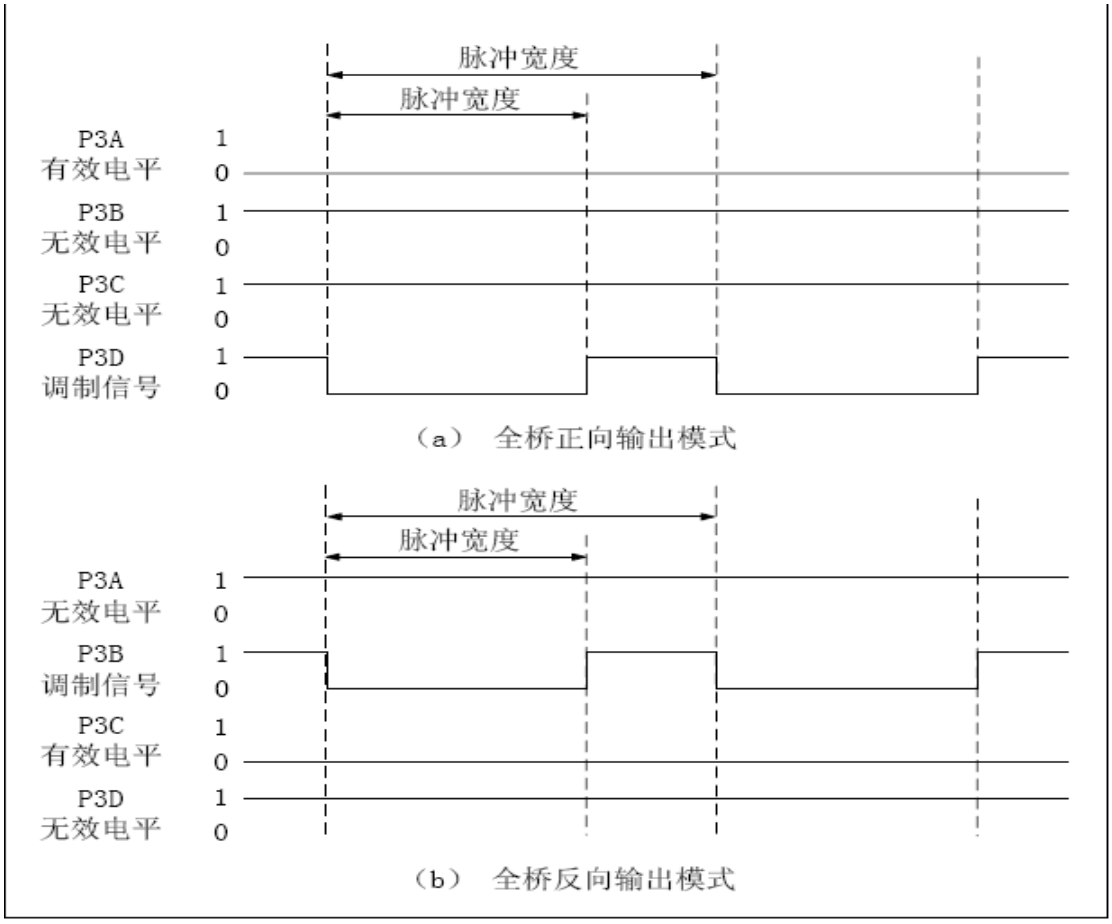


图13.2（低电平有效）

关于自动关断，PWM3模块具有自动关断功能，使能自动关断功能后，在外部关断事件发生时，该功能自动禁止PWM输出，然后将P3A、P3B、P3C和P3D四个引脚输出电平置于其预定义的状态。此模式用于防止PWM破坏应用电路。

自动关断模式具有4个关断源:INT0引脚的逻辑低电平、比较器1输出高电平、比较器2输出高电平和在软件中直接将P3ASE位置1。关断源触发关断的信号是高电平或低电平，而不是上升沿或下降沿，只要关断源的关断电平存在，自动关断状态将保持。

关于自动重启，即一旦清除自动关断条件就自动重启PWM。通过将PWM3CTL1寄存器中的PRSEN位置1使能自动重启。

如果使能自动重启，只要自动关断条件有效，P3ASE位就将保持置1。当清除自动关闭条件时，将通过硬件将P3ASE位清0，并且将恢复常规操作。

在本实验中，PWM3工作在全桥正向输出模式下，P3ASCTL = 0x44，即设定自动关断源为INT0，当INT0引脚为低电平时，使能自动关断功能，同时设定关闭状

态下P3A、P3C端口为高电平，P3B、P3D端口为低电平。PWM3CTL1 = 0x80，使能自动重启功能。PWM正常输出情况下，P3A(黄色)和P3D（绿色）端口波形如图13.3所示。当使用杜邦线将INT0(P02)和GND连接后，即自动关断功能启动，此时P3A(黄色)和P3D（绿色）端口波形如图13.4所示。当断开INT0（P02）和GND的连接后，关闭事件消失，因使能了自动重启功能，此时P3A(黄色)和P3D（绿色）端口波形又回到图13.3，即进入正常输出模式。

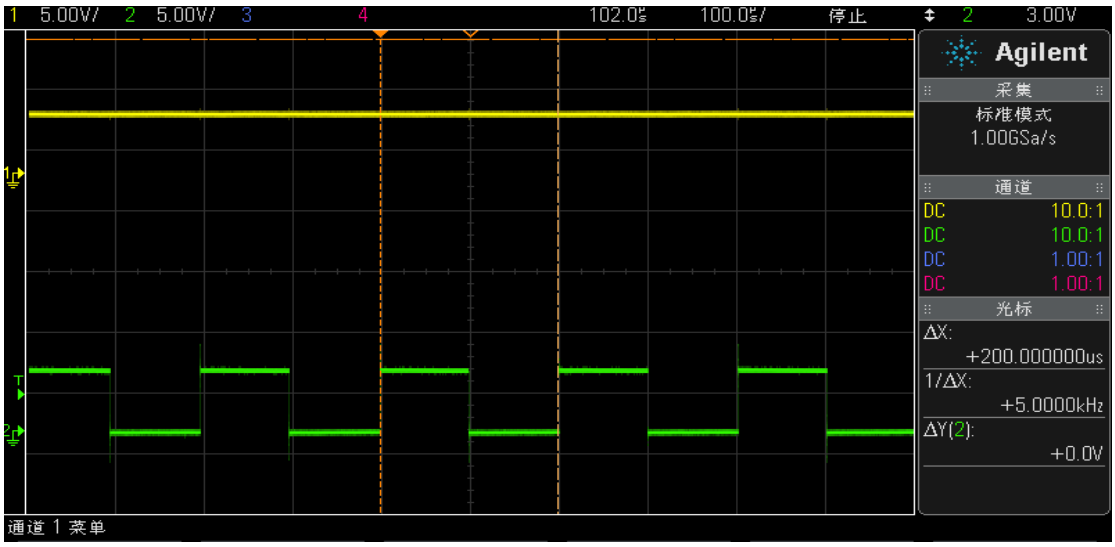


图 13.3　PWM3 正常工作波形输出（P3A 黄色 P3D 绿色）

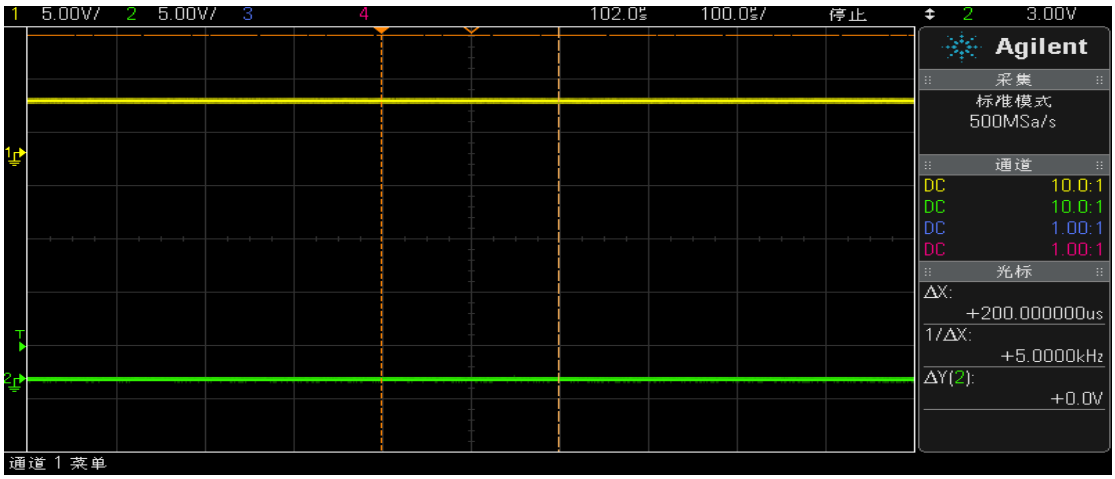


图 13.4　自动关断后波形输出（P3A 黄色 P3D 绿色）

在配置 PWM3 全桥输出时需注意以下几点：

1、在使用全桥输出模式时，需将P3A、P3B、P3C和P3D引脚对应的方向控制位TR1x清0，设置为输出，将对应ANSEx位清0，设置为数字I/O口。

2、全桥输出模式下没有死区延时功能。通常在此模式中，任何时间只调制一对输出，因此不会导致电路产生直通电流，所以不需要死区延时。

3、INT0（P02）需使能其上拉功能。

13.4 .1　C 程序代码

```
#include<KF8F312.h>

/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int

/*********************宏定义结束*************************/
unsigned char Full_Bridge_Way;  //全桥正向反向标记 1 正向   0 反向
unsigned char Full_PWM_Bak;     //占空比高位备份，全桥转向防直通措施
/************************
 * 函数名     : Delay_ms
 * 函数功能：长时间延时
 * 入口参数：延时基数 uchar ms_data
 * 返回         : 无
 ************************/
void Delay_ms(uchar ms_data)
{
    uchar i;
    while(ms_data--)
    {
        i = 220;
        while(i--);
    }
}
/************************
 * 函数名     : Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun()
{
    OSCCTL = 0X60;      // 8M时钟

    TR0 = 0X0C;         // P03口必须设置为输入口
    TR1 = 0X00;
```

```
    TR2 = 0X00;

    P0 = 0x00;
    P1 = 0x00;
    P2 = 0xF0;

    PUPH = 0;          // 使能总上拉 PUPH
    PUR = 0X04;         // 使能P02上拉

    // 模式设定
        PWM3CTL0=0x40;  // 全桥正向输出模式，暂不开启，默认有效状态高电平。
        PWM3CTL1=0x80;    // 开启自动使能，死区仅对半桥有效
        P3ASCTL = 0x44; // 选择自动关闭源 INT0（P02）为低电平    关闭状态下P3A、
P3C端口为高电平    P3B、P3D端口为低电平
        PATRCTL=0x10; // 开启转向同步。
        Full_Bridge_Way=1;
    // 占空比初始化50%
        PWM3L=0x32;
        PDT1=0;           // IN PWM3CTL0 bit 5
        PDT0=0;           // IN PWM3CTL0 bit 4
        PP3=0x64;      // 4倍数值构成PWM周期最小单位,设置PWM3周期为200us
    // PWM使能
        T2=0;
        T2CTL = 0X05;       // 启动T2，分频器1为4分频
        P3ON1=1;       // IN PWM3CTL0 bit 3
        P3ON0=1;       // IN PWM3CTL0 bit 2
//=======PWM3 int end============//
}
void main()
{
    Init_fun();
    while (1)
    {
        Delay_ms(50);
        //全桥转向,需防止接近百分百占空比下的，开关管导通时间小于截止时间时的直通，
-
        //如果占空比运行模式均小于一定值，可以不考虑直通现象,详细介绍见数据手册。
        if(Full_Bridge_Way)
        {
            Full_PWM_Bak=PWM3L;
            PWM3L=0x3F;              // 暂时降低占空比
            Delay_ms(1);          // 延后1个周期的时间，使占空比生效
            PWM3CTL0|=0x80;
            Full_Bridge_Way=0;
```

```
            PWM3L=Full_PWM_Bak;       // 占空比还原
        }
        else
        {
            Full_PWM_Bak=PWM3L;
            PWM3L=0x3F;               // 暂时降低占空比
            Delay_ms(1);          // 延后1个周期的时间，使占空比生效
            PWM3CTL0&=0x7F;
            Full_Bridge_Way=1;
            PWM3L=Full_PWM_Bak;       // 占空比还原
        }
    }
}
//中断函数
void int_fun() __interrupt
{


}
```

13.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

PSW_TEMP            .EQU        0X85      ;PSW的临时保存变量
PCH_TEMP            .EQU        0X86      ;PCH的临时保存变量
DELAY_NUM1              .EQU        0X82      ;延时用的临时变量
DELAY_NUM2              .EQU        0X83      ;延时用的临时变量

Full_Bridge_Way         .EQU        0X80      ;全桥正向反向标记 1
正向   0 反向
Full_PWM_Bak        .EQU        0X81      ;占空比高位备份，全桥转
向防直通措施

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

INTERRUPT

;;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要
保护，常规需要保护的内容有PSW PCH R0 R1
```

```
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
**********************************/



;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET

;/*********************************主函数程序
**********************************/
MAIN
    CALL INIT_ALL                ;初始化寄存器
    CALL INIT_RAM                ;初始化RAM

LOOP
    ;//全桥转向,需防止接近百分百占空比下的,开关管导通时间小于截止时间时的直通，-
    ;//如果占空比运行模式均小于一定值，可以不考虑直通现象,详细介绍见数据手册。
    JB Full_Bridge_Way,0
    JMPFull_Bridge_Way_0
Full_Bridge_Way_1
    MOVR0, PWM3L
    MOVFull_PWM_Bak, R0        ;// 备份部分当前占空比

    MOVR0,# 0x3f
    MOVPWM3L, R0             ;// 暂时降低占空比
    CALL  DELAY              ;// 等待生效

    SETPWM3CTL0, 7           ;// 换向
    CLRFull_Bridge_Way
```

```
    MOVR0, Full_PWM_Bak
    MOVPWM3L, R0            ;// 还原占空比

    JMPDELAY_DO
Full_Bridge_Way_0
    MOVR0, PWM3L
    MOVFull_PWM_Bak, R0

    MOVR0,# 0x3f
    MOVPWM3L, R0            ;// 暂时降低占空比
    CALL   DELAY            ;// 延后1个周期的时间，使占空比生效

    CLRPWM3CTL0, 7              ;// 换向
    SETFull_Bridge_Way, 0

    MOVR0, Full_PWM_Bak
    MOVPWM3L, R0            ;// 还原占空比


DELAY_DO
    MOV R7,#0x5A
    CALL DELAY
    DECJZ R7
    JMP $-2

    JMP LOOP
    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
```

```
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X0C
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0        ;LED模块关闭
    MOV P2,R0

    CLR ANSEL              ;P02 配置为数字口 ANS2 = 0
    CLR OPTR,7
    MOV R0,#0X04           ;使能P02上拉
    MOV PUR,R0

    MOV R0,#0X64
    MOV PP3,R0             ;设置PWM3周期为200us
    MOV R0,#0X32
    MOV PWM3L,R0           ;占空比50%
    MOV R0,#0X05
    MOV T2CTL,R0           ;启动T2，分频器1为4分频
    MOV R0,#0x4c
    MOV PWM3CTL0,R0        ;设定为全桥正向输出  启动PWM3 各引脚高电
平有效
    SET Full_Bridge_Way,0
    MOV R0,#0x80
    MOV PWM3CTL1,R0        ;使能自动重启
    MOV R0,#0x44
    MOV P3ASCTL,R0         ;选择自动关闭源 INT0（P02）为低电平   关
闭状态下P3A、P3C端口为高电平    P3B、P3D端口为低电平
    CRET

;/*************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
```

```
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET
;*****************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
DELAY
    MOV R0 ,#0x36
    MOV DELAY_NUM1, R0
LP0
    MOV R0 ,#0X36
    MOV DELAY_NUM2, R0
```

```
LP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP LP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP LP0
    CRET                    ;子程序返回

    .END
```

实验十四、模拟比较器模块实验

14.1 程序声明

KF8F 系列单片机开发板演示程序

标　　题：模拟比较器模块实验

项 目 名：14-CMP1_CMP2_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：设置模拟比较器 1，P00 作为正端输入，P01 作为负端输入，通过滑动变阻器改变 P00 和 P01 端口电平，从而改变比较器输出端电平，当正端输入大于负端输入时产生相应中断。

14.2 硬件说明

本实验使用了 P00 和 P01 端口作为比较器的输入端，所以需将其连接到滑动变阻器。将单片机的 P00 和 P01 端口的跳上跳线帽，将 JP1 上的 AN0 和 AN1 跳上。D4 连接到了单片机的 P17 端口，所以单片机的 P17 端口需跳上跳线帽，同时 J14 也需跳上跳线帽，如图 14.2 所示。

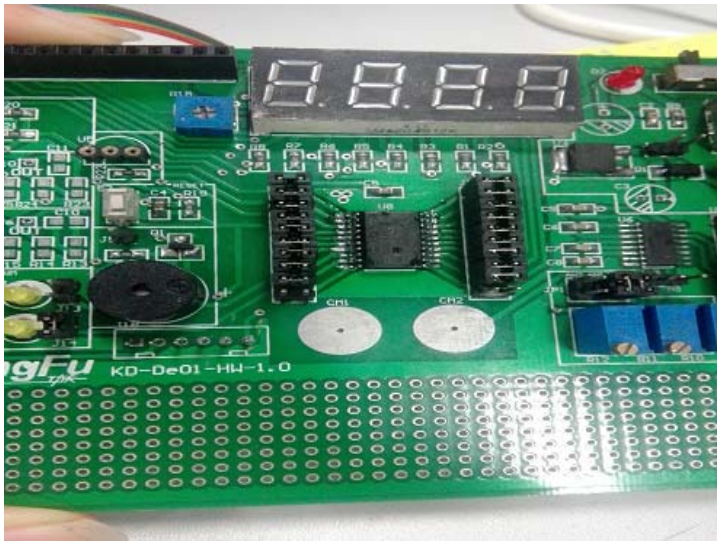


图 14.2　接线图

14.3 程序设计说明

KF8F312含有2路模拟比较器，分别是CMP1和CMP2。其中模拟比较器的正端输入为IO端口，负端输入可选择IO端口、1.9V的VREOUT或内部的1/2VDD。 其原理框图如图14.1

本实验使用了CMP1，CMCTL1 = 0X01，选择P01作为比较器负端输入。其正负端口的电平通过改变滑动变阻器的阻值来改变，当C1IN+ > C1IN- ，比较器输出端输出高电平，否则输出低电平。当比较器的争端输入大于负端输入时，将产生相应的中断标志，在中断处理函数中，改变LED灯的显示状态，所以观察到的实验现象是当改变滑动变阻器的阻值，可以改变LED的显示状态。

程序配置过程需注意一下几点：

1、比较器中需要作为输入的I/O端口如果有模拟输入功能则设为模拟输入口，否则只用设为输入口，需要作为输出的I/O端口只用设为输出口；

2、比较器中断属于外部中断，使用时需使能外部中断，即 PUIE = 1；

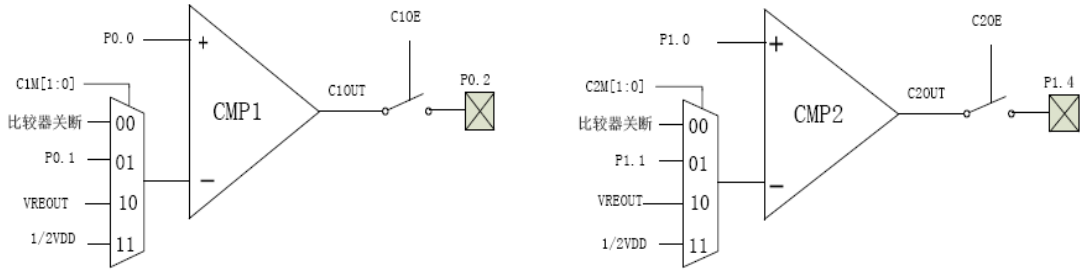


图 14.1　模拟比较器原理框图

14.4 .1 C 程序代码

```
#include<KF8F312.h>
/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int

#define LED P17

/*********************宏定义结束*************************/

/************************
 * 函数名     : Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun()
{
    OSCCTL = 0X60;    // 配置8M时钟
    TR0 = 0X0B;       // p03 复位引脚 设为输入          P00、P01分别为比较器的
正端输入和负端输入 设置为输入 P02可选比较结果硬件输出
    TR1 = 0x00;
    TR2 = 0x00;

    ANSEL = 0X03;     // 比较器的两个输入端设置为模拟口 AN0 AN1

    P0 = 0x00;
    P1 = 0x00;
    P2 = 0xF0;
}

void delay (void)
{
    unsigned char i = 0;
    unsigned char j = 0;

    for ( i = 100; i > 0;i--)
        for (j = 100;j > 0;j--);
}
```

```c
void main()
{
    Init_fun();
    delay();        // 初始化完毕后作适当延时
    C1OUT=1;       // IN CMCTL0 bit 4  +>-=1
    C1OE=1;           // IN CMCTL0 bit 2  P02输出
    CMCTL1 = 0X01;  // P00 为比较器正端输入，选定P01为比较器负端输入

    C1IE = 1;      // 使能比较器1中断  EIE1 bit 3
    C1IF = 0;       // 清中断标志 EIF1 bit 3
    INTCTL = 0XC0;  // 使能外部中断 PUIE  使能总中断 AIE
    while (1);
}
//中断函数
void int_fun() __interrupt
{
    if (C1IF)
    {
        C1IF = 0;
        LED = !LED;             // 改变LED灯状态
    }
}
```

14.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

DELAY_NUM0              .EQU        0x80      ;延时用的临时变量
DELAY_NUM1              .EQU        0x81      ;延时用的临时变量
PSW_TEMP            .EQU        0X85      ;PSW的临时保存变量
PCH_TEMP            .EQU        0X86      ;PCH的临时保存变量

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

INTERRUPT
;;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要
保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
```

```
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
**********************************/
    JB EIF1,C1IF
    JMP INT_END
    CLR EIF1,C1IF

    MOV R1,#0X80              ; P17取反
    XOR P1,R1

;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET


;/*********************************主函数程序
**********************************/
MAIN
    CALL INIT_ALL                ;初始化寄存器
    CALL INIT_RAM                ;初始化RAM

    CALL DELAY_MS

    MOV R0,#0X04                 ;启动比较器1 C1OE 使能其输出到对应
引脚
    MOV CMCTL0,R0
    MOV R0,#0X01                 ;P00 为比较器正端输入，P01为比较器
负端输入
    MOV CMCTL1,R0

    MOV R0,#0X08                 ; 使能比较器1中断 C1IE
    MOV EIE1,R0
    CLR EIF1                     ; 清中断标志  C1IF
    MOV R0,#0XC0                 ; 使能比较器1中断 C1IE
    MOV INTCTL,R0
```

```
LOOP


    JMP LOOP
    CRET
;*******************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*******************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X0B         ;p03 复位引脚 设为输入      P00、P01分别为比
较器的正端输入和负端输入 设置为输入
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    MOV R0,#0X03         ;比较器的两个输入端设置为模拟口
    MOV ANSEL,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0
    MOV P2,R0

    CRET
```

```
;/*****************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL    INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

;*****************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
```

```
DELAY_MS
    MOV    R0 ,#0xC8
    MOV    DELAY_NUM0, R0
LP0
    CWDT
    MOV    R0 ,#0XC8
    MOV    DELAY_NUM1, R0
LP1
    DECJZ  DELAY_NUM1       ;DELAY_NUM2自减1，若为0，则跳过
    JMP    LP1
    DECJZ  DELAY_NUM0       ;DELAY_NUM1自减1，若为0，则跳过
    JMP    LP0
    CRET

    .END
```

实验十五、系统时钟输出和内部参考电压输出

15.1 程序声明

KF8F 系列单片机开发板演示程序

标　　题：系统时钟输出和内部参考电压输出

项 目 名：15-Output_System_Clock_And_Vreout_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：配置单片机 CLKOUT 引脚输出系统时钟和 VREOUT 引脚输出内部参考电压（LDO）。

15.2 硬件说明

本实验只需使用示波器观察单片机的 CLKOUT 引脚和 VREOUT 引脚的电平即可，由于 CLKOUT 引脚和 VREOUT 引脚在单片机上共用 P04 脚，所以实验应分开做，同一时间只能输出一个。

15.3 程序设计说明

振荡周期又叫时钟周期，是振荡器振荡频率的倒数。本芯片中一个机器周期

等于四个时钟周期。OSCCTL为系统频率控制寄存器，CKOEN = 1即可通过CLKOUT引脚输出当前系统时钟。如OSCCTL = 0XE0;即配置当前系统时钟为8M，并使能通过CLKOUT引脚输出，在此需注意的是CLKOUT引脚输出波形的频率为系统时钟4分频后的频率，如图15.1。

KF8F312内部有一个参考电压模块，使能该功能后，通过引脚P0.4/VREOUT可输出稳定的1.9V参考电压，将VREEN(VRECTL1.1)位置1将打开参考电压模块，此时的1.9V参考电压可供芯片内部使用(如用作AD参考电压和比较器参考电压)，再将VREOE(VRECTL1.3)位置1可使能内部1.9V参考电压输出，相应的引脚输出1.9V参考电压，如图15.2。

用户如果要用到内部1.9V参考电压，需先读出7FBH/7FAH地址的参考电压校准值，送到VRECAL1/VRECAL2寄存器，然后根据需要设置VRECTL1中的VREEN和VREOE位。

程序配置过程需注意一下几点：

1、CLKOUT引脚输出波形的频率为当前系统时钟频率的4分之一，如当前系统时钟为8M，使能输出后CLKOUT引脚输出的波形频率为2M。

2、对于参考电压，如果只是供内部使用（比如用作AD参考电压和比较器参考电压），不需要将寄存器VRECTL1的VREOE（VRECTL1.3）位置1。

3、当通过VREOUT引脚输出内部参考电压时，需要配置VREOUT引脚为输入口。

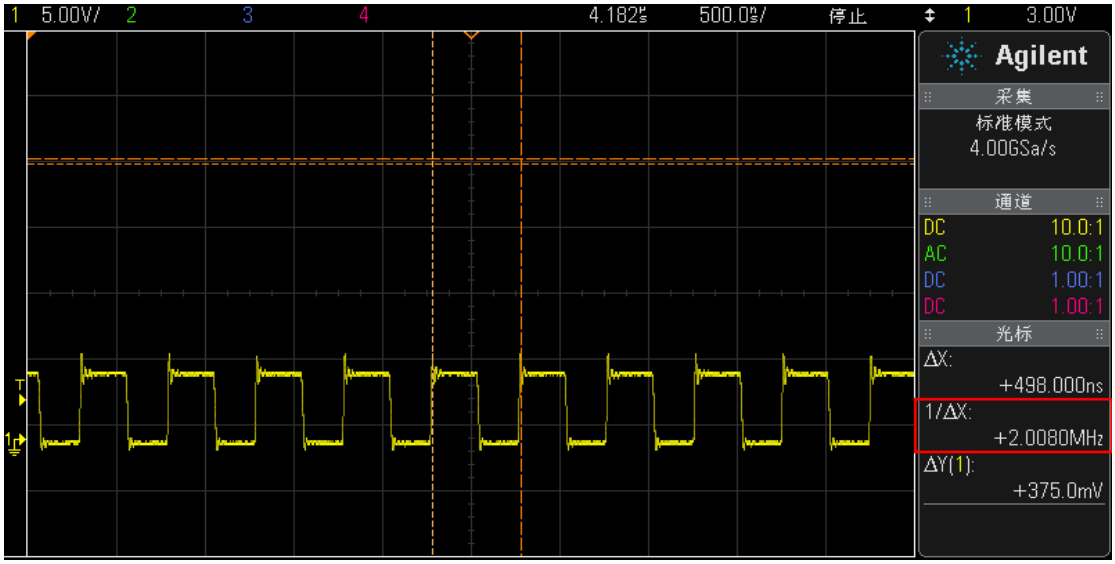


图15.1 CLKOUT引脚输出频率为2M

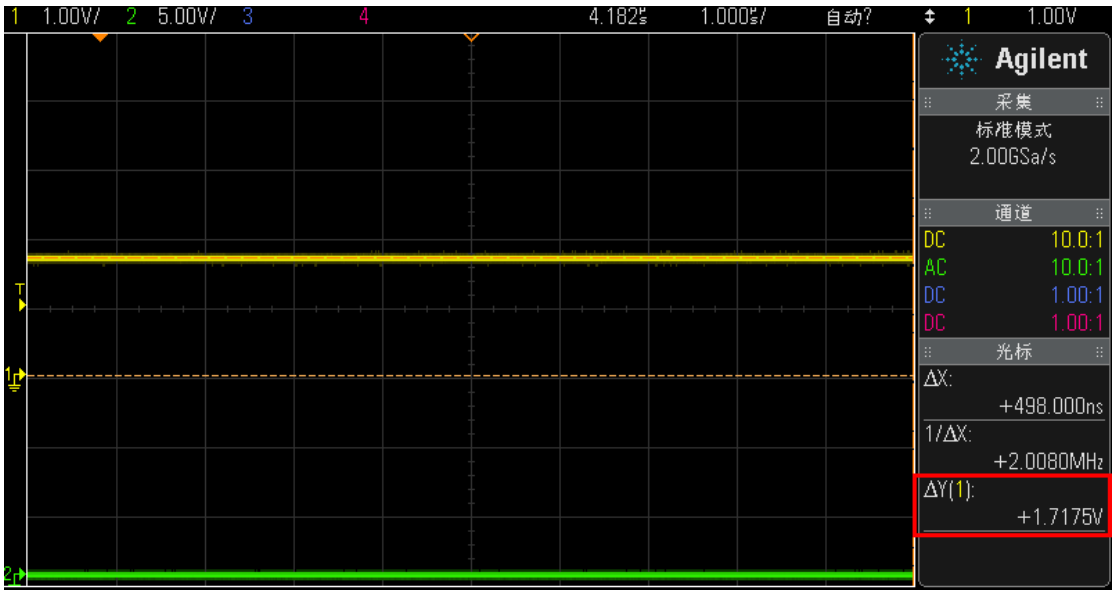


图15.2 VREOUT引脚输出1.7V的内部参考电压

15.4.1 C程序代码

```
#include<KF8F312.h>

/************************
 * 函数名     : Init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回          : 无
 ************************/
void Init_fun()
{
#if 1   //  1 时钟输出   0 参考电压输出
    /*****************************系统时钟输出配置
*********************************/
    OSCCTL = 0XE0;    // 配置8M时钟  使能CLKOUT引脚输出时钟为2M
    TR0 = 0X08;       // p03 复位引脚 设为输入
    TR1 = 0;
    TR2 = 0;

    P0 = 0;
    P1 = 0;
    P2 = 0xF0;
    /*****************************系统时钟输出配置
*********************************/
#else
```

```
    /*****************************LDO输出配置
************************************/
    OSCCTL = 0X60;
    TR0 = 0X18;       // p03 复位引脚 设为输入  参考电压输出引脚P04需配置为输入
    TR1 = 0;
    TR2 = 0;

    P0 = 0;
    P1 = 0;
    P2 = 0xF0;
    __asm             // 内部参考电压校准
    CALL 0XFFD
    MOV VRECAL,R0
    __endasm;

    PCTL = 0XC0;      // 使能参考电压 VREEN 使能参考电压输出 VREOE
    /*****************************LDO输出配置
************************************/
#endif
}

void main()
{
    Init_fun();
    while (1);
}


//中断函数
void int_fun() __interrupt
{


}
```

15.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

PSW_TEMP            .EQU          0X85       ;PSW的临时保存变量
PCH_TEMP            .EQU          0X86       ;PCH的临时保存变量

    .ORG 0X0000
    NOP
```

```
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

INTERRUPT

;;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
     SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
**********************************/



;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
MAIN
    CALL INIT_ALL                ;初始化寄存器
    CALL INIT_RAM                ;初始化RAM

LOOP
    JMP LOOP
    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
```

```
.if 1        ;// 1  选择时钟输出  0 选择参考电压输出
;/*****************************系统时钟输出配置
*********************************/
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0xE0
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X08
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0
    MOV P2,R0
;/*****************************系统时钟输出配置
*********************************/
.else
;/*****************************LDO输出配置
************************************/
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
```

```
    ;端口初始化
    MOV R0,#0X18          ;p03 复位引脚 设为输入  参考电压输出引脚P04
需配置为输入
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0
    MOV P2,R0
    ;初始化参考电压校准
    CALL 0XFFD            ;内部参考电压校准
    MOV VRECAL,R0

    MOV R0,#0XC0          ;使能参考电压 VREEN 使能参考电压输出 VREOE
    MOV PCTL,R0

;/*****************************LDO输出配置
************************************/
.endif
    CRET

;/*****************************************************************
******************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
```

```
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .END
```

实验十六、串口通信(全双工异步模式)实验

16.1 程序声明

KF8F 系列单片机开发板演示程序

标　　题：串口通信(全双工异步模式)实验

项 目 名：18-Usart_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：配置芯片串口的接收和发送功能，串口发送端隔一定时间向外发送一个数值，且数值逐一递增。接收端则把接收到的数据通过数码管的个位和百位进行显示（十六进制），所以在本实验中可以将 demo 板通过串口线和 PC 机相连，通过 PC 机上的串口助手可以接收到单片机发送的数据，同时通过串口助手发送的数据也可以在 demo 板的数码管上显示。同时也可以使用杜邦线将单片机的发送端和接收端直接相连，实现串口的自收发功能。

16.2 硬件说明

本实验使用开发板的串口和计算机的串口进行通信，单片机的高电平为 5V，

低电平为 0V，但是计算机的串口为 RS-232C 电平，高电平为-12V，低电平为 12V，所以当计算机和单片机之间要进行通信时，需要接电平转换芯片，原理框图如图 16.1 所示。在此需要使用杜邦线将 J15 的 RXD 与单片机的 RXD 引脚连接，J15 的 TXD 与单片机的 TXD 引脚相连，同时使用串口线连接单片机的串口和计算机的 USB 口，在计算机上使用串口助手对数据进行收发即可，接线图如图 16.2。

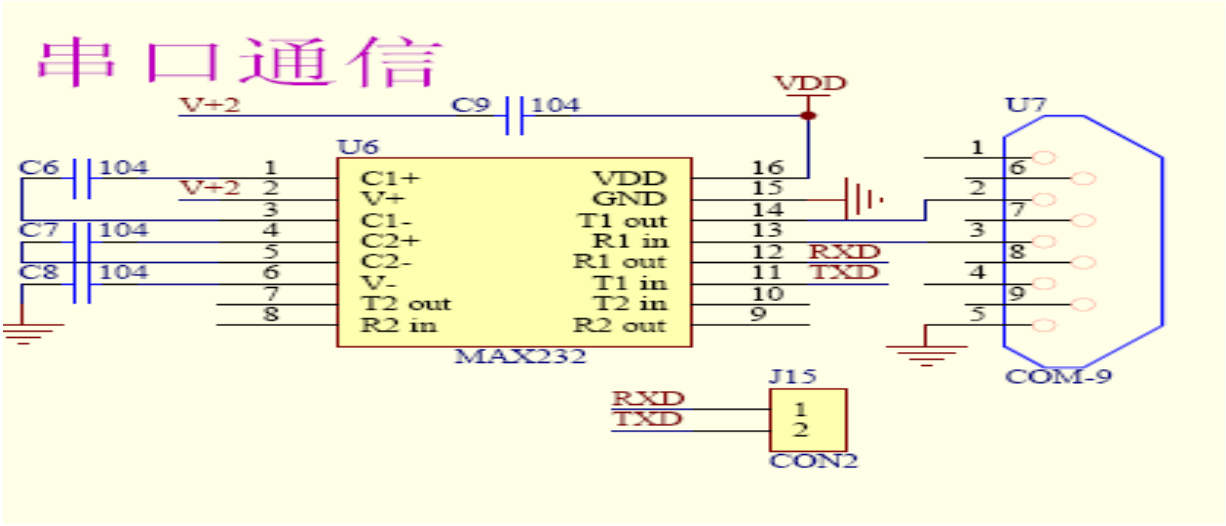


图 16.1

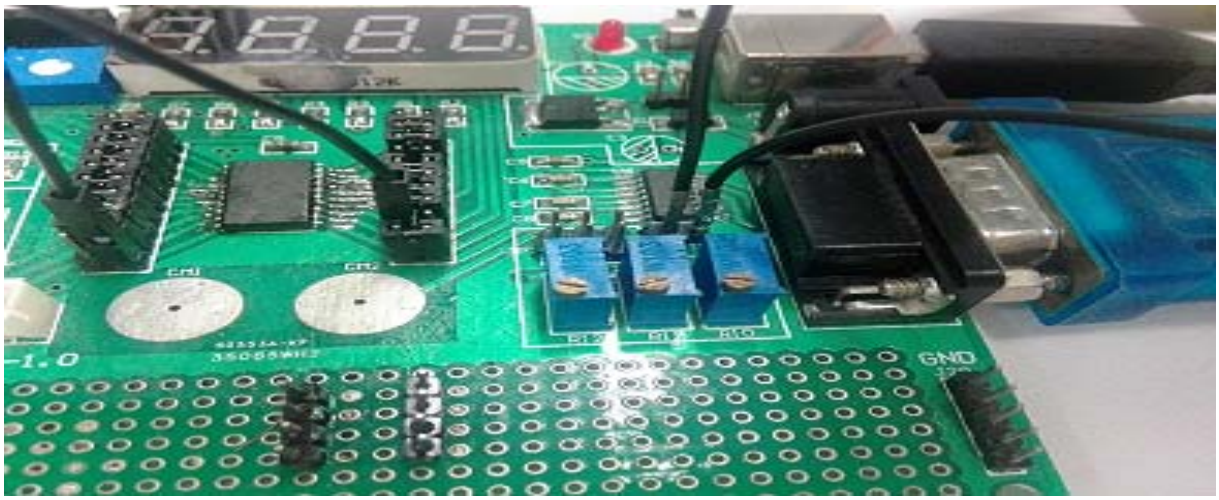


图 16.2 接线图

16.3 程序设计说明

USART 称通用全双工/半双工收发器，这是一个串口通信的I/O外设，也可作为串行通信接口。它可被配置为与个人计算机等外设通信的全双工异步系统。也可以被配置为与外设或其它单片机通信的半双工同步系统，与之通信的单片机通常不具有产生波特率的内部时钟，它需要主控同步器件提供外部时钟信号。它有

如下功能特点：全双工异步发送和接收、RS485检测、双字节输入缓冲器、单字节输出缓冲器、可将字符长度编程为8位或9位、输入缓冲溢出错误检测、接收到字符的帧错误检测、半双工同步主控/从动模式和半双工同步模式下可编程时钟极性。

USART模块还可实现如下附加功能，从而使其成为局域互联网络总线系统的理想选择：自动波特率检测、校准和13位间隔字符发送。

其原理框图如图16.3所示

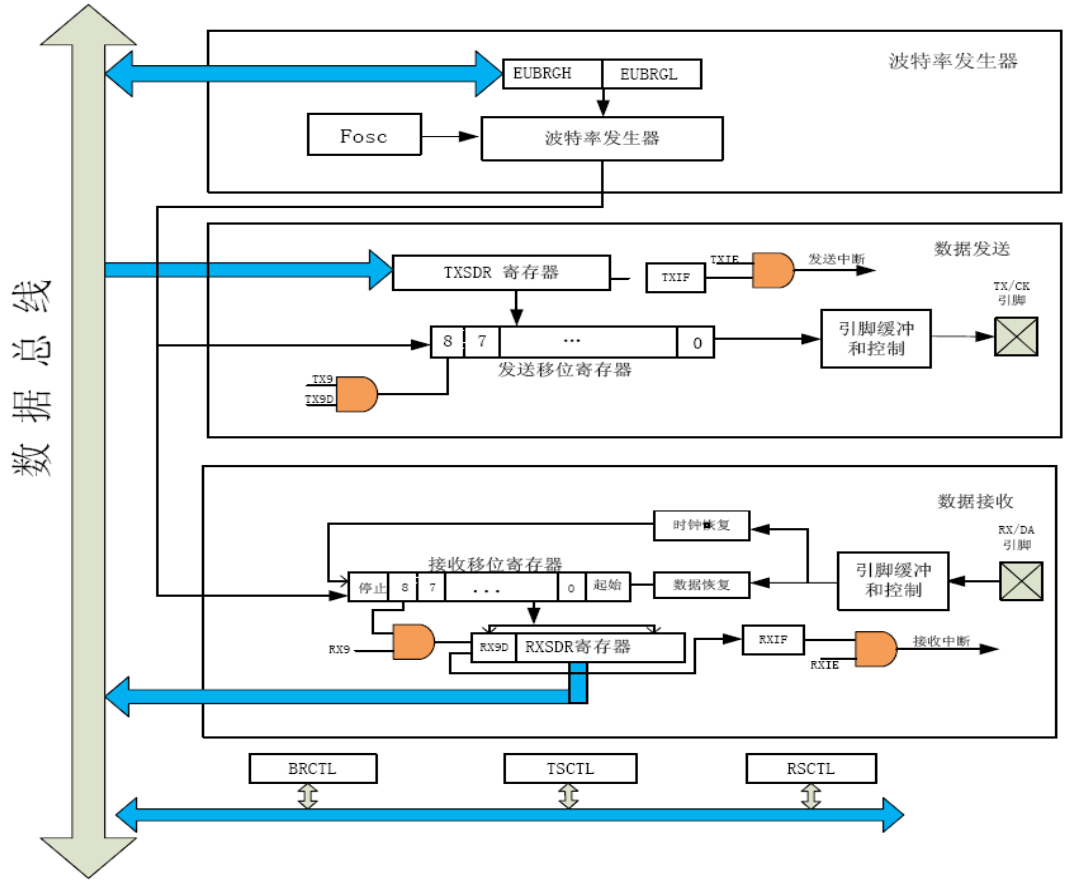


图16.3

由图16.3可知，全双工/半双工收发器主要包括波特率发生器（BRG）、数据发送和数据接收这三部分，每个部分都有相应的寄存器设置，主要包括波特率控制寄存器BRCTL，发送状态和控制寄存器TSCTL和接收状态和控制寄存器RSCTL。

波特率发生器：

$$\text{目标波特率} = \frac{Fosc}{m \times ([EUBRGH:EUBRGL]+1)}$$

由公式可知，在系统频率已知的情况下，波特率主要有三个参数决定，分别是m、EUBRGH、EUBRGL。而这三个参数又是由状态位SYNC、BRG16和HBRG决定，SYNC、BRG16和HBRG三个状态位的含义如表16.1所示。m的取值参考表16.2，在芯片手册中，各种全双工异步模式的典型波特率和误差值已经计算出来，我们只需根据SYNC、BRG16和HBRG状态位选择对应的EUBRGH、EUBRGL值即可。例如：

当频率为8M，SYNC=0,HBRG=0,BRG16=0时，如果所需波特率为9600bit/s，即查表可知只需 EUBRGL = 0x0C即可，

当频率为8M，SYNC=0,HBRG=1,BRG16=1时，如果所需波特率为9600bit/s，即查表可知只需EUBRGH = 0x00、EUBRGL = 0xCF即可。

对BRCTL寄存器的配置如下：

1、RCIDLF=1 接收器空闲

2、BRG16=1 使用16位波特率发生器

BRCTL其它位取值为0

对于SCKPS ，如果SCKPS = 1；则传送反相数据到P27/TX/CK引脚。如图16.5（SCKPS = 0）和图16.6（SCKPS = 1）所示。图16.4和图16.5都是单片机从PC机串口助手接收0xA5、0x11、0x33三个字节（绿色波形）和单片机向PC机返回0x11、0x33两个字节（黄色波形）。对比两图可发现绿色波形一样，黄色波形高低电平刚好相反（单片机TX引脚）。

	1	0
SYNC	半双工同步模式	全双工异步模式
BRG16	使用16位波特率发生器	使用8位波特率发生器
HBRG	高速 （全双工模式下）	低速 （全双工模式下）

表16.1 状态位SYNC、BRG16和HBRG的含义

配置位			BRG/USART 模式	倍频器 m
SYNC	BRG16	HBRG		
0	0	0	8 位/异步	64
0	0	1	8 位/异步	16
0	1	0	16 位/异步	
0	1	1	16 位/异步	4
1	0	x	8 位/同步	
1	1	x	16 位/同步	

注：x 为无关位

表16.2 倍频器m选择表

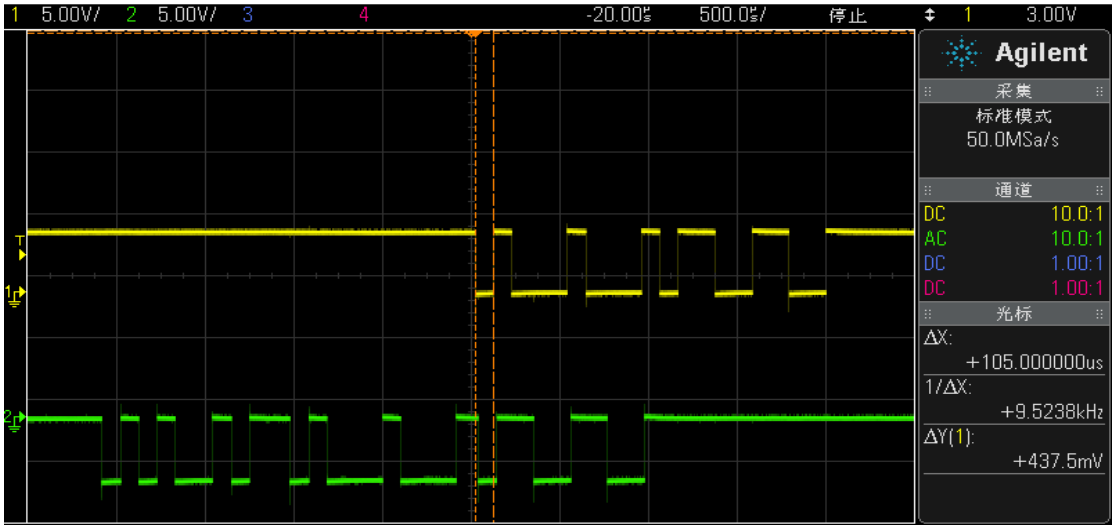


图16.4（SCKPS = 0）

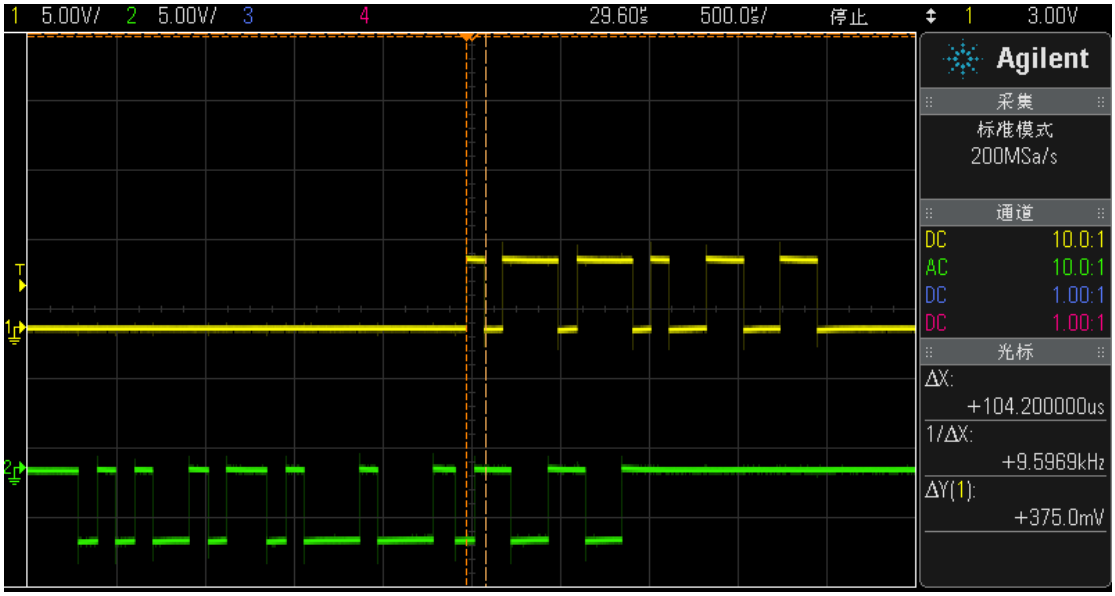


图16.5（SCKPS = 1）

USART全双工模式

在全双工异步通信中，数据是一帧一帧传送的，每一帧的数据格式如图16.6所示。在帧格式中，一个字符由4部分组成：起始位，数据位，奇偶校验位和停止位。

起始位：通常情况下是逻辑0，占用一位，用来通知接收设备一个等待接收字符的开始。

数据位：8位。

奇偶校验位：bit8，占用一位，但在字符中可以规定不用奇偶校验位，则这一位可以省去。

停止位：一定为逻辑1，用来表征字符的结束。

停止位：一定为逻辑1，用来表征字符的结束。停止位可以是1位、1.5位或2位。若停止位以后不再紧接着传送下一个字符，则使线路电平保持为高电平(逻辑1)，处于空闲状态。这也是全双工异步通信的一大特点。

USART首先发送和接收低位。USART的发送器和接收器在功能上是相互独立的，但采用相同的数据格式和波特率。硬件不支持奇偶校验，但可以用软件实现（奇偶校验位是第9 个数据位）。

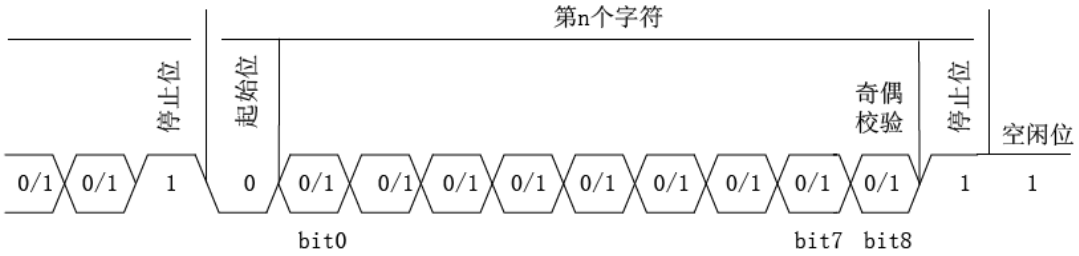


图16.6

USART全双工发送操作

由图16.3可知，发送器的核心是串行发送移位寄存器，该寄存器不能有软件直接访问，而是从TXSDR发送缓冲寄存器读取数据，所以发送数据时，可直接把所要发送的数据赋给TXSDR寄存器即可。TSCTL寄存器的TXSRS位指示发送移位寄存器的状态。TXSRS位为只读位。当发送移位寄存器为空时，TXSRS位被置1，当有字符从TXSDR传输到发送移位寄存器时，TXSRS被清0，所以可以通过查询该位的值来判断数据是否发送完成。综上所述，对TSCTL寄存器的配置如下：

1、 TX9=0 选择8位发送

2、 TXEN=1 使能发送

3、 SYNC=0 全双工异步模式

4、 SENDB=0 同步间隔字符发送完成

5、 HBRG=0 低速

6、 TXSRS =1 发送移位空

TSCTL 寄存器的其它位取值为0

图16.7为PC机串口助手向单片机发送0xA5、0x11、0x33三个字节（绿色波形），而单片机向串口助手回复0x11、0x33两个字节。从图中右下角红框可知波特率将近9600bit/s，图中中间的两条红线之间为单片机发送的0x11字节波形，波形从左往右转化为二进制为0b0100010001，其中红色的0为起始位（低电平），绿色的1代表停止位（高电平），发送数据时从字节的最低位开始进行发送。图中蓝色框内波形在时间上有重叠，说明串口在接收数据时可以同时进行发送，这就是所谓的全双工。

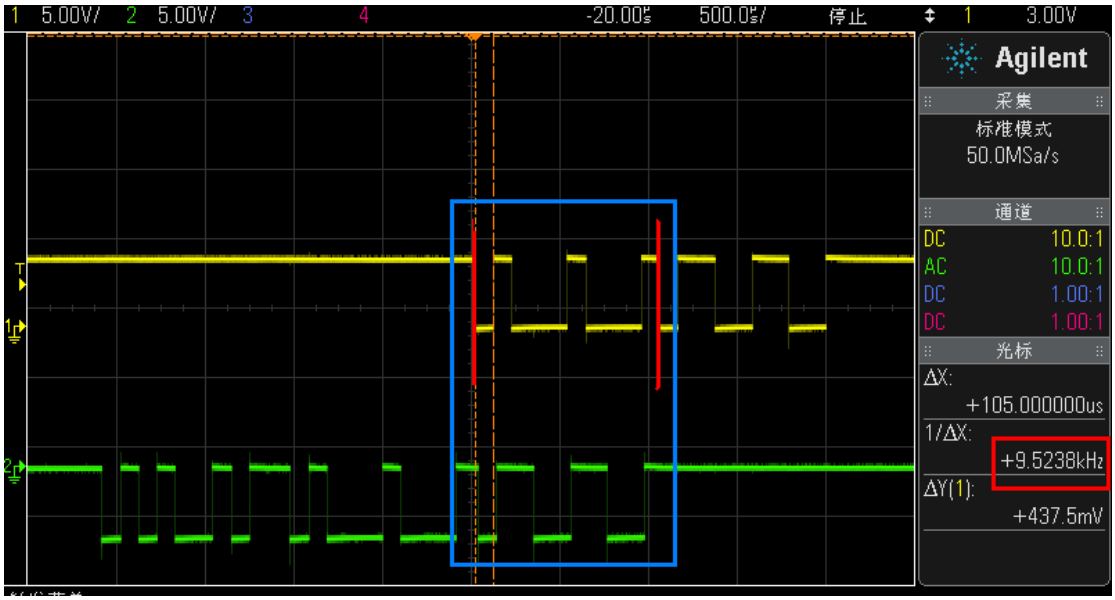


图16.7

USART全双工接收操作

数据接收模块主要由数据恢复电路、串行接收移位寄存器（RSR）和2字符的FIFO缓冲器组成。数据恢复电路实际上是一个以16倍波特率为工作频率的高速移位器，用于对原始波形数据进行采样，并将采样结果全部移入RSR，当全部8位数据移入RSR后，立即会被传输到FIFO缓冲器。RSR不能通过软件进行访问，此时可

以通过RXSDR寄存器访问接收到的数据。

关于接收中断，当接收到所有数据位和停止位后，RXIF中断标志将被置1，在接收FIFO中没有未读数据时RXIF自动清0，RXIF中断标志位为只读，不能由软件置1或清0。使能接收中断的步骤如下：

1、EIE2寄存器的RXIE中断允许位

2、INTCTL寄存器的PUIE外设中断允许位

3、INTCTL寄存器的AIE全局中断允许位

关于溢出错误，接收FIFO缓冲器可以保存2个字符。但如果在访问FIFO缓冲器之前，接收到完整的第3个字符，则会产生溢出错误。此时，RSCTL寄存器的OVFER位会置1。可以读取FIFO缓冲器内的字符，但是在错误清除之前，不能再接收其它字符。可以通过清0 RSCTL寄存器的CRXEN位或通过清0 RSCTL寄存器的SPEN位使USART复位来清除错误。

综上所述，对RSCTL寄存器的配置如下：

1、 SPEN=1 使能串行口

2、 CRXEN=1 使能接收器

RSCTL寄存器其它位清0即可

16.4 .1 C 程序代码

```
#include<KF8F312.h>
#define uchar   unsigned char
#define uint  unsigned int

#define Sent_Vaule_To_UART(key_buf)   {TXSDR=key_buf;
while(TXSRS==0);}// 调试时串口输出

unsigned char Timer_Count;
unsigned char Send_Date;
unsigned char Send_Flag;

unsigned char Rev_Temp = 0;
unsigned char Rev_Flag;

unsigned char  const Arr_num[] = {
                        0X3F,      // 0
                        0X06,
```

```
                        0X5B,
                        0X4F,
                        0X66,      // 4
                        0X6D,
                        0X7D,
                        0X07,
                        0X7F,      // 8
                        0X6F,      // 9
                        0X77,      // A
                        0X7C,      // B
                        0X58,      // C
                        0X5E,      // D
                        0X79,      // E
                        0X71,      // F
                        0X40,      // -
};                      //共阴接法,数码管显示笔形码
uchar Unit, Decade, Hundred, Thousand;   //定义个十百千位
/************************
 * 函数名     : init_fun
 * 函数功能：初始化
 * 入口参数：
 * 返回         : 无
 ************************/
void init_fun (void)
{
    OSCCTL = 0x60;          //8M

    TR0 = 0X08;
    TR1 = 0X00;
    TR2 = 0X00;

    OPTR = 0x05;  //定时器0,定时发送数据到端口
    T0 = 0x64;

    TSCTL=0x22; // 8位 发送，使能发送，全双工异步模式SYNC=0，低速模式HBRG=0，
发送移位空,低速的
    RSCTL=0x90; // 串口使能（配置引脚为串行口引脚），8位接收，接收使能
    BRCTL=0x48; // 16位波特率发送器BRG16=1
    EUBRGL=0x33;// SYNC=0,HBRG=0,BRG16=1 FOSC = 8M 9600bit/s

    EIE2 = 0X20;       // 关闭发送中断 TXIE 使能接收中断 RXIE
    EIF2 = 0;         // 清中断标志位

    INTCTL = 0XE0;     // 使能外部中断 PUIE 使能总中断 AIE  T0IE
```

```
}
/************************
 * 函数名     : Delay_200us
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         : 无
 ************************/
void Delay_200us()
{
    uchar i = 60;
    while(--i);
}
/************************
 * 函数名     : Dis_LED
 * 函数功能：LED 显示
 * 入口参数：无
 * 返回         : 无
 ************************/
void Display_LED()
{
    P2 = 0XE0;            //打开数码管的 个位
    P1 = Arr_num[Unit];
    Delay_200us();
    P1 = 0;
    P2 = 0XF0;            //消隐

    P2 = 0XB0;            //打开数码管的 百位
    P1 = Arr_num[Hundred];
    Delay_200us();
    P1 = 0;
    P2 = 0XF0;            //消隐

    Delay_200us();
}
/************************
 * 函数名     : Delay_ms
 * 函数功能：长时间延时
 * 入口参数：延时基数 uchar ms_data
 * 返回         : 无
 ************************/
void Delay_ms(uchar ms_data)
{
    uchar i;
    while(ms_data--)
```

```
    {
        i = 220;
        while(i--);
    }
}
void main()
{
    init_fun();
    Rev_Temp=0;
    Send_Date=0x00;

    Delay_ms(200);

    while (1)
    {
        if(Send_Flag)
        {
            Send_Flag=0;
            Sent_Vaule_To_UART(Send_Date);
        }
        if(Rev_Flag)
        {
            Rev_Flag=0;
            Thousand=Rev_Temp;
            Unit=Thousand&0x0F;
            Hundred=(Thousand&0xF0)>>4;
        }
        Display_LED();
    }
}
//中断函数
void int_fun() __interrupt
{
//-----------------------------------------------
    if(RXIF)
    {
        if(OVFER==1)
        {
            CRXEN=0;       // 清溢出错误
            CRXEN=1;       // 重新使能接收
        }
        else
        {
            Rev_Flag=1;
```

```
        }
        Rev_Temp=RXSDR;  // 清零 RXIF
    }
//-----------------------------------------------
    if(T0IF)
    {
        T0IF = 0;
        T0 = 0x64;
        Timer_Count++;

        if(Timer_Count>100)
        {
            Timer_Count=0;
            Send_Date++;
            Send_Flag=1;
        }
    }
//-----------------------------------------------
}
```

16.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

Rev_Temp            .EQU        0X80      ;保存接收到的数据
Send_Date           .EQU        0X81      ;保存接收到数据长度

Rev_Flag            .EQU        0X82      ;
Send_Flag           .EQU        0X83      ;

Timer_Count             .EQU        0X84      ;计数卡时间

PSW_TEMP            .EQU        0X86      ;PSW的临时保存变量
PCH_TEMP            .EQU        0X87      ;PCH的临时保存变量


UNIT                .EQU        0X88      ;数码管个位
DECADE              .EQU        0X89      ;数码管十位
HUNDRED                 .EQU        0X8A      ;数码管百位
THOUSAND            .EQU        0X8B      ;数码管千位

DELAY_NUM1              .EQU        0X8C      ;延时用的临时变量
DELAY_NUM2              .EQU        0X8D      ;延时用的临时变量
    .ORG 0X0000
```

```
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT
;*****************************************************************
*****************
;* 函 数 名: DIG_DATA
;* 函数功能: 数码管显示数据编码
;* 入口参数: 无
;* 返    回: R0，数码管显示的编码值
;* 注  意 : 查表函数需在程序开头定义，避免编译后表中元素存在Flash的不
同页造成查表出错
;*****************************************************************
*****************
DIG_DATA
    ADD PCL, R2
    RRET R0 ,#0X3F      ;0
    RRET R0 ,#0X06      ;1
    RRET R0 ,#0X5B      ;2
    RRET R0 ,#0X4F      ;3
    RRET R0 ,#0X66      ;4
    RRET R0 ,#0X6D      ;5
    RRET R0 ,#0X7D      ;6
    RRET R0 ,#0X07      ;7
    RRET R0 ,#0X7F      ;8
    RRET R0 ,#0X6F      ;9
    RRET R0 ,#0X77      ;A
    RRET R0 ,#0X7C      ;B
    RRET R0 ,#0X39      ;C
    RRET R0 ,#0X5E      ;D
    RRET R0 ,#0X79      ;E
    RRET R0 ,#0X71      ;F
    RRET R0 ,#0X40      ;-
INTERRUPT

;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保
护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
**********************************/
```

```
    JB  EIF2, 5             ;RXIF
    JMP T0_Deal

    JNB RSCTL, 1        ;OVFER
    JMP Rev_Date_OK
    CLR RSCTL, 4        ;CRXEN=0;       // 清溢出错误
    SET RSCTL, 4        ;CRXEN=1;       // 重新使能接收
    JMP Rev_Date
Rev_Date_OK
    SET Rev_Flag, 0
Rev_Date
    MOV R1, RXSDR       ;// 清零 RXIF
    MOV Rev_Temp, R1
    ;T0中断判断
T0_Deal
    JB  INTCTL, 2
    JMP INT_END

    CLR INTCTL, 2
    MOV R1,# 0x64
    MOV T0, R1
    INC Timer_Count
    MOV R1,# 0x65
    SUB R1, Timer_Count
    JB  PSW, 0
    JMP INT_END
    CLR Timer_Count
    INC Send_Date
    SET Send_Flag, 0
;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOV R1 , PCH_TEMP
    MOV PCH, R1
    SWAP R1 , PSW_TEMP
    MOV  PSW, R1
    IRET

MAIN
    CALL INIT_ALL               ;初始化寄存器
    CALL INIT_RAM               ;初始化RAM
```

```
LOOP
    ;发送间隔到时发送数据
    JB  Send_Flag,0
    JMPRev_Flag_Deal
    CLRSend_Flag
    MOVR0, Send_Date
    MOVTXSDR, R0
Wait_Send_Over
    JB TSCTL, 1
    JMPWait_Send_Over

    ;接收到数据时解析到LED
Rev_Flag_Deal
    JNBRev_Flag, 1
    JMPLED_Show

    CLRRev_Flag

    MOVR0, Rev_Temp
    MOVTHOUSAND, R0

    MOVR0,# 0x0f
    ANDR0, THOUSAND

    MOVUNIT, R0

    SWAPR R0,THOUSAND

    ANDR0,# 0x0f
    MOVHUNDRED, R0

LED_Show
    CALL LED_DISPLAY
    JMP LOOP
    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
```

```
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X08
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    CLR P2
    ;T0
    MOV R0,#0x05
    MOV OPTR,R0
    MOV R0,#0x64
    MOV T0,R0
    ;UART
    MOV R0,#0X22
    MOV TSCTL,R0     ;8位 发送，使能发送，全双工异步模式SYNC=0，低速
模式HBRG=0，发送移位空,低速的
    MOV R0,#0x90
    MOV RSCTL,R0     ;串口使能（配置引脚为串行口引脚），8位接收，接收
使能
    MOV R0,#0x48
    MOV BRCTL,R0     ;16位波特率发送器BRG16=1
    MOV R0,#0x33
    MOV EUBRGL,R0    ;SYNC=0,HBRG=0,BRG16=1 FOSC = 8M 9600bit/s

    MOV R0,#0X20     ;关闭发送中断 TXIE 使能接收中断 RXIE
    MOV EIE2,R0
    CLR EIF2         ;清中断标志位
    ;
```

```
    MOV R0,#0XE0     ;使能外部中断 PUIE 使能总中断 AIE 使能定时T0中断
    MOV INTCTL,R0

    CRET
;*************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
DELAY
    MOV R0,#0x12
    MOV DELAY_NUM1, R0
DLP0
    MOV R0,#0X12
    MOV DELAY_NUM2, R0
DLP1
    DECJZ DELAY_NUM2        ;DELAY_NUM2自减1，若为0，则跳过
    JMP DLP1
    DECJZ DELAY_NUM1        ;DELAY_NUM1自减1，若为0，则跳过
    JMP DLP0
    CRET                ;子程序返回

;*************************************************************
*****************
;* 函 数 名: LED_DISPLAY
;* 函数功能: 数码管显示函数，由4个8位LED灯显示
;* 入口参数: THOUSAND 千位 ,HUNDRED 百位 ,SMQ_S 十位, UNIT 个位  ,
每个数的范围 0~f
;* 返    回: 无
;*************************************************************
*****************
LED_DISPLAY
    MOV R0 , #0XE0
    MOV P2 , R0          ;打开数码管的 个位
    MOV R2 , UNIT
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0
```

```
    MOV R0 , #0XB0
    MOV P2 , R0          ;打开数码管的 百位
    MOV R2 , HUNDRED
    CALL DIG_DATA
    MOV P1 , R0
    CALL DELAY
    CLR P1           ;消抖
    MOV R0 ,#0XF0
    MOV P2 , R0

    CALL DELAY
    CRET
;/*****************************************************************
******************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************/
INIT_RAM
    CLR    R2         ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X20    ;操作个数
    CALL    INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80    ;起始地址0x80
    MOV    R1 ,#0X01    ;操作个数
    CALL    INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
```

```
DECJZ R1
JMP  INIT_RAM_0
CRET

.END
```

实验十七、看门狗唤醒休眠实验

17.1 程序声明

KF8F 系列单片机开发板演示程序

标　　题：看门狗唤醒休眠实验

项 目 名：17-WDT_WakeUp_LDLE

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：为降低运行功耗，使单片机在运行-睡眠-运行的模式下循环运行，开始时 LED 熄灭，单片机进入休眠，看门狗唤醒之后点亮 LED，然后又熄灭，如此循环，所以会看到 LED 在不停地闪烁。

17.2 硬件说明

本实验主要通过 LED 的点亮和熄灭验证单片机在正常运行状态和休眠状态之间切换。所以只需把单片机的 P16 口和 J13 跳上跳线帽即可，接线图如图 17.1。

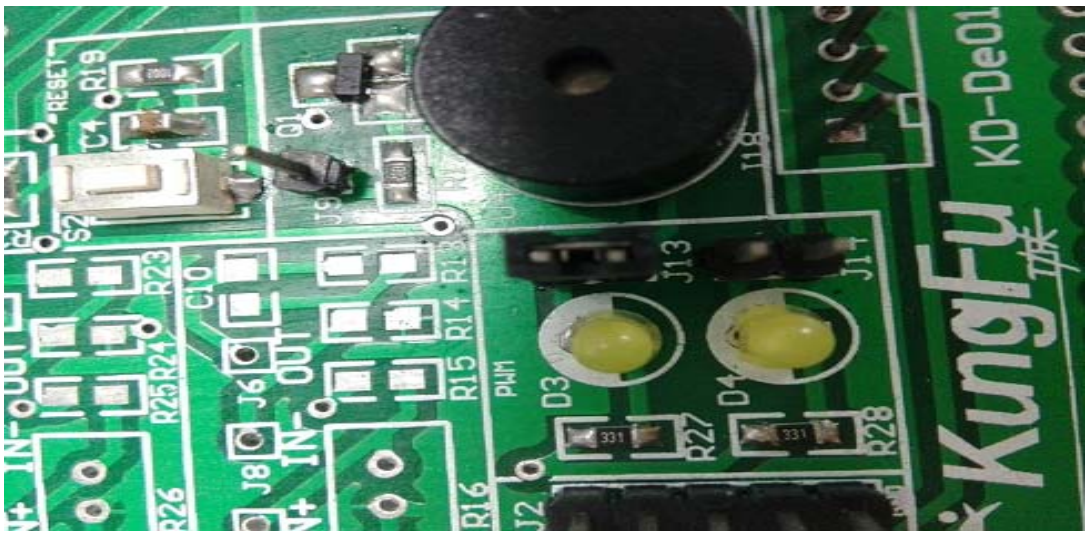


图 17.1　接线图

17.3 程序设计说明

关于休眠，当单片机空闲的时候，为使其功耗达到最低，可以将其转入休眠模式。通过执行一条IDLE指令即可进入休眠模式。

单片机进入休眠模式一段时间后由于工作的需要，要将单片机从休眠模式唤醒，在KF8F312中可通过以下方式将单片机从休眠模式唤醒:

1. RST引脚上输入的外部复位
2. 看门狗定时器唤醒(如果WDT已被使能)
3. INT0内部中断
4. P0口电平变化中断
5. 外设中断
6. T0中断

RST引脚输入的复位信号在唤醒单片机的同时也将导致单片机复位。其它唤醒时将单片机从休眠模式唤醒，并不会导致复位。可通过状态寄存器中的 TO（PSW<4>）和PD（PSW<3>）位来确定单片机唤醒的原因。上电时PD（PSW<3>）位将被置1，而当器件从休眠模式唤醒时，该位将被清0。TO（PSW<4>）位则在WDT唤醒发生时被清0。

在使用后三种方式唤醒时，必须使能相应的中断使能位，唤醒与AIE位的状态无关。如果AIE位被清0，单片机被唤醒后将继续执行IDLE指令后面的指令。如果AIE位被置1，单片机执行IDLE指令后面一条指令后进入中断子程序。如果不希望执行IDLE指令后面的那条指令直接进入中断子程序，在IDLE指令加一条NOP指令即可。

本实验使用看门狗进行唤醒。为了防止单片机在正常工作时程序跑飞，KF8F312提供一个看门狗定时器，看门狗框图如图17.2。单片机正常工作时，当看门狗定时器定时时间达到超时时间后，会使单片机产生复位。

看门狗定时器使用片内看门狗专用RC振荡器，因此它无需外接任何器件，在休眠模式仍能正常运行。在正常运行时，WDT超时事件将使单片机产生一次复位。如果单片机处于休眠模式，WDT超时事件将唤醒单片机并使其继续执行IDLE后面的

指令。在KF8F312芯片中无法使用软件打开和关闭看门狗，只能通过在IDE将配置位WDTE清0/置1，可关闭/打开WDT，如图17.3。

关于WDT周期，当WDT不使用预分频器时超时时间为18ms，通过软件将OPTR寄存器的PSA位置1，可将预分频器分配给WDT，设置PS<2:0>选择预分频器的分频比，通过设置预分频比，最长超时时间可达2.3秒。

本程序的配置需注意一下几点：

1、 为保证单片机进入休眠后功耗达到最低，应使所有I/O口状态确定，如果没用的端口悬空，应设置为输出，以确保I/O引脚没有耗散电流产生，并且其他在休眠时不用的外设都要关闭，如中断等。

2、 单片机执行IDLE指令进入休眠后，为保证能正常唤醒，最好在IDLE指令之后加上3条以上的空指令_NOP_()。

3、 程序中使用到的_NOP_()和_IDLE()语句在KF8COMMON.h文件中进行了定义。

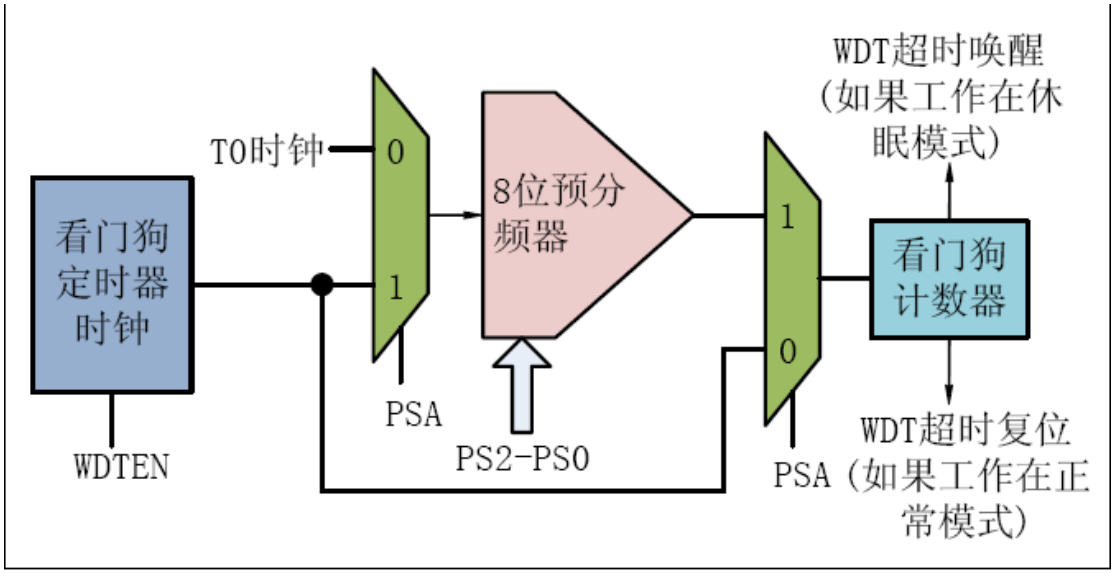


图17.2 看门狗定时器框图

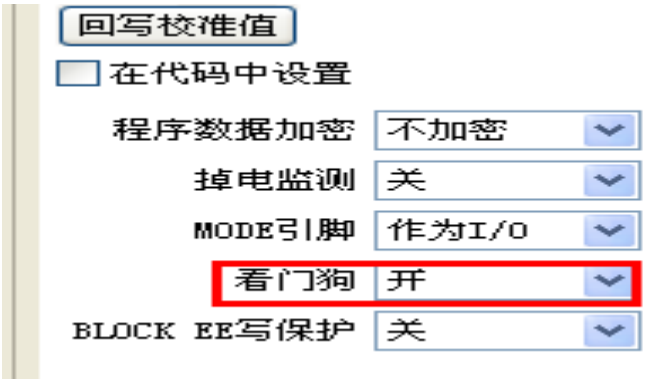


图17.3 通过配置位WDTE打开看门狗

17.4 .1 C 程序代码

```
#include<KF8F312.h>

#define LED1 P16

void delay (void)
{
    unsigned char i = 0,j = 0;

    for (i = 110;i > 0;i--)
        for (j = 110;j > 0;j--);
}

void main()
{
    OSCCTL = 0x60;                          //设置为8M
    /*********端口初始化***********/
    TR0 = 0x08;                               //设置P0端口    1100 1111
    TR1 = 0x00;                               //设置P1端口    0000 0000
    TR2 = 0x00;                               //设置P2端口    0011 1111
    P0 = 0x00;
    P1 = 0x00;
    P2 = 0xF0;

    OPTR = 0X0F;       //  预分频器分配给看门狗使用 PSA = 1   分频比为1:128，
即看门狗超时时间为2.3s
    while (1)
    {
        LED1 = !LED1;       // LED状态转换，快速转换下相当于PWM调光，常亮状态
        _IDLE();           // 进入睡眠状态
        _NOP_();
        _NOP_();
        _NOP_();
        _NOP_();
    }
}


//中断函数
void int_fun() __interrupt
{

}
```

17.4.2　汇编程序代码

```
.INCLUDE "KF8F312.INC"

DELAY_NUM0              .EQU          0x80       ;延时用的临时变量
DELAY_NUM1              .EQU          0x81       ;延时用的临时变量
PSW_TEMP            .EQU          0X85       ;PSW的临时保存变量
PCH_TEMP            .EQU          0X86       ;PCH的临时保存变量

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT

INTERRUPT
;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
     SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
**********************************/



;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
MAIN
    CALL INIT_ALL                ;初始化寄存器
    CALL INIT_RAM                ;初始化RAM

LOOP
    MOV R0,#0X40
```

```
    XOR P1,R0                   ;LED状态转换，快速转换下相当于PWM调光，
常亮状态,休眠会加长时间变为亮灭切换
    CALL DELAY_MS
    CWDT
    IDLE                        ;休眠指令
    NOP
    NOP
    NOP
    JMP LOOP
    CRET
;*****************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X08
    MOV TR0,R0
    MOV R0,#0X00
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0     ;LED COM
    MOV P2,R0
```

```
    MOV R0,#0X0F
    MOV OPTR,R0

    CRET

;/*****************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

;*****************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
```

```
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************
DELAY_MS
    MOV    R0 ,#0xC8
    MOV    DELAY_NUM0, R0
LP0
    CWDT
    MOV    R0 ,#0XC8
    MOV    DELAY_NUM1, R0
LP1
    DECJZ  DELAY_NUM1       ;DELAY_NUM2自减1，若为0，则跳过
    JMP    LP1
    DECJZ  DELAY_NUM0       ;DELAY_NUM1自减1，若为0，则跳过
    JMP    LP0
    CRET

    .END
```

实验十八、 读写BLOCK EEPROM实验

18.1 程序声明

KF8F系列单片机开发板演示程序

标 题: BLOCK EEPROM的读写操作

项 目 名: 18-W_R_BEE_TEST

开发环境：ChipON IDE

使用芯片：KF8F312

作 者：上海芯旺微电子技术有限公司

功能简述: 首先对BEE的两页（0X0F70~0X0F7F,0X0F80~0X0F8F）进行页写操作，都写入0000,1111，2222一直到EEEE,FFFF接着读取BEE地址0X0F79的值，再用块写写入0X0F99，最后将BEE地址0X0F77中的数据修改为0X55AA，实现对BEE的读、页写、块写、修改操作实验。

18.2 硬件说明

由于本实验是对单片机内部EEPROM进行操作，所以不需外设电路，只需给单片机供电即可操作。所需连接跳线如图18.1所示。

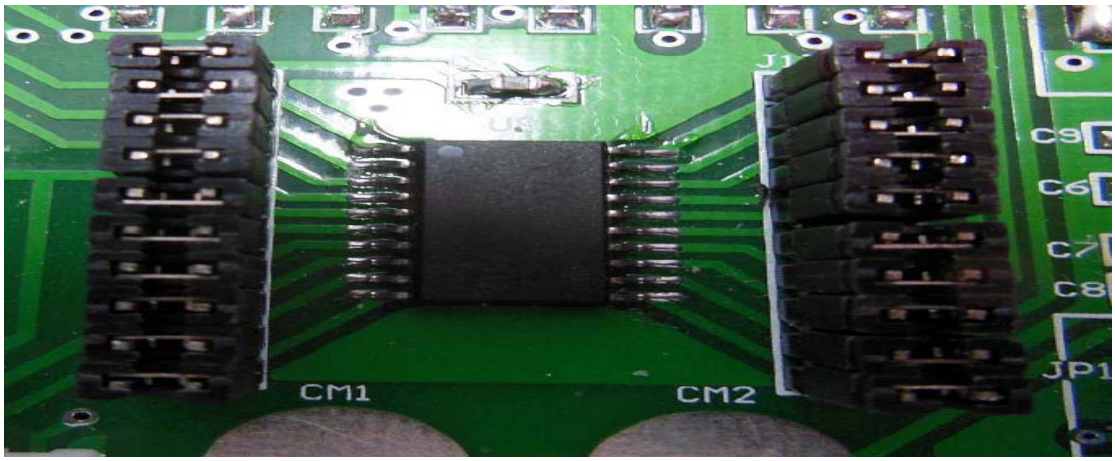


图18.1 所需连接跳线

18.3 程序设计说明

KF8F312在程序存储区后面的地址开辟了一个128X16位的BLOCK EEPROM。写BLOCK EEPROM时，BLOCK EEPROM中所有内存单元以连续的四个地址为一个数据块，四个数据块为一页。每个地址可以存放一个16位的数据，从地址0X0F70-0X0FEF,共8页。

写BLOCK EEPROM时，只能对BLOCK EEPROM成块写入数据，不能单独将一个字节(或字)的数据写入某块的一个字节(或字)中，如果实际上写入BLOCK EEPROM中的数据没有4个字或不能被4整除(例如要写入一组7个字的数据),需要将块中不需要写入数据的单元写入0或者其他值，否则可能会导致写入的数据出错。如果原来的BLOCK EEPROM保存有数据，现在需要修改原数据中的一个字或者几个字，其他单元的值不变，则需要先将其对应块中其他数据读出来保存，然后再根据实际情况将需要修改的值和之前读出的值写入即可。值得注意的是，在对每一页的第一块进行写操作的时候，是会擦除本页之后的内容的，对其他块进行写操作，不会擦除本页内容。 使用BLOCK EEPROM时，R6/R7寄存器用来存放要写入或者读出BLOCK EEPROM的数据，R6存放数据的低8位，R7存放数据的高8位。寄存器BADDRH/BADDRL。用来存放要写入BLOCK EEPROM的13位的地址信息，BADDRH存放地址的高5位，BADDRL存放地址的低8位。例如要对地址0X0F70进行写操作，那么BADDRL中存放的是0X70，BADDRH中存放的是0X0F。寄存器EECTL1/EECTL2为写BLOCK EEPROM控制寄存器，KF8F312为用户设计了对寄存器EECTL1/EECTL2固定的写命令，用户在进行读写命令时可以直接调用。所以用户在写BLOCK EEPROM时，

将R6/R7中送入要写入的数据，BADDRH/BADDRL中送入要写入的地址，然后通过向EECTL1和EECTL2送入固定的写命令，将要写入BLOCK的数据存到对应的缓冲寄存器中。

在读BLOCK EEPROM时，将要读的地址送到BADDRH/BADDRL中，然后向EECTL1写入固定的读命令，把要读的数据送到R6/R7中。 由于读写函数涉及到寄存器R6/R7的操作，而R6/R7寄存器并没有给出地址，所以用户在进行EEP读写的时候需进行一些汇编语言的操作。 例如进行页写：我们可以定义一个页写的子函数，如下所示，其中Erom_add为地址，我们可以将其定义成一个2个字节的变量，用来存放地址。Arr_erom为数组名，由于一页有四个数据块，每个数据块有四个地址，每个地址可以存放一个16位的数据，那么我们可以将其定义成一个拥有16个元素的数组，元素大小事两个字节的。R1为间接寻址的中间寄存器，存放操作数的地址，Erom_add虽然为两个字节的变量，但数据存储是将低8位放到低地址中，高8位放到高地址中的， “_Erom_add”即是提取数据存放地址的低地址，将该地址中的数据传送给BADDRL寄存器。Arr_erom数组中有16个元素，每个元素分解成低8位和高8位，依次存入R6，R7中。每存一个元素，调用一次数据写入函数，而数据写入函数即是对EECTL1和EECTL2的固定写命令。即子函数Erom_write_cmd()的格式是固定的，用户不需作修改。读操作，过程类似，在此不作敷述。

在对BLOCK EEPROM进行操作时，需注意一下几点：

1、如果用户要使用 BLOCK EEPROM，则需要把配置字里面的 BLOCK EE 写保护设置为“关”；

2、写BLOCK EEPROM 的数据个数至少要写4 个字或能被4 整除；

3、写BLOCK EEPROM 时要关掉总中断，防止中断打断写BLOCK EEPROM 的时序，写完之后再打开总中断；

4、在读写 BLOCK EEPROM 的过程中需要添加多个指令周期的.DW 0xffff指令，样例程序进行了说明。

5、写 BLOCK EEPROM 的时候，必须先对每个页的第一块进行写操作，以擦除本页的数据，如果直接写第二块、第三块或者第四块，这样会出现数据写错或者写不进去的情形。

18.4 .1 C 程序代码

```
#include <KF8F312.h>
//主函数
//====================//////////////////BEE相关声明
////////////////==============/////////
              /*参数传递说明见函数实现位置*/
#define           BEE_BUFFER_MAX    16      // 数据的大小与单次操作量相关，这里int型，但也只能是4，8，12，16，除非不用写操作。
unsigned int  BEE_BUFFER[BEE_BUFFER_MAX];

unsigned int    BEE_READ_BUF;    //当单字读函数使用非全嵌汇编模式时，使用该值获取读取结果R7 R6构成的值
unsigned int  BEE_READ_ONE  (unsigned int address);  //读一个字
void          BEE_READ_FUN  (unsigned int address,unsigned char length);//读多个数据到数据缓存区，建议最多量为1个页即16个数据，同时建议不跨页操作，与写函数对应
void          BEE_WRITE_ONE (unsigned int address,unsigned int value);
// 过程封装
void          BEE_WRITE_FUN(unsigned int address,unsigned char length);
//将缓存区数据写到对应位置，从第一个开始
///=====================================================================
======================///
void init()
{
    OSCCTL = 0x60;                          //设置为8M
    /*********端口初始化***********/
    TR0 = 0x08;                               //设置P0端口    1100 1111
    TR1 = 0x00;                               //设置P1端口    0000 0000
    TR2 = 0x00;                               //设置P2端口    0011 1111
    P0 = 0x00;
    P1 = 0x80;        //P1.6 P1.7 指示运行结束灯
    P2 = 0xF0;
}

void main()
{
    unsigned char i;
    unsigned int temp=0;
    init();
//--- 准备待写入的数据0x0000 0x1111 0x2222...0xFFFF
    for (i = 0; i < BEE_BUFFER_MAX; i++)
    {
        BEE_BUFFER[i] = temp;
```

```
        temp += 0x1111;
    }
//---setp 1       写入到0x0F70 和0x0F80的BEE空间
    BEE_WRITE_FUN(0x0F70,BEE_BUFFER_MAX);
    BEE_WRITE_FUN(0x0F80,BEE_BUFFER_MAX);
//--- setp 2      读取 0x0f79，并写入到0x0F99
    // 读取所需到缓存
    temp=BEE_READ_ONE(0x0F79);
    // 除了0xF99 ，0x0F98  0x0F9A  0x0F9B 的默认值处理,
    for (i = 0; i < 4; i++)
    {
        BEE_BUFFER[i] = 0xFFF1;
    }
    //应该擦除0x0F90页的数据 以便后续以0x0F98开始的块写，如果确定该页为0xFFFF，
可以跳过
    BEE_WRITE_FUN(0x0F90,4);
    // 修改要写入的值
    BEE_BUFFER[1]=temp;
    BEE_BUFFER[3]=0xFF33; //验证 用 0x0F9C
    // 执行0x0F98块的写入
    BEE_WRITE_FUN(0x0F98,4);
//--- setp 3  将OX0F77中的数据修改为0X55AA
#if 1  // 0 或  1
    BEE_WRITE_ONE(0x0F77,0x55AA);
#else
    BEE_READ_FUN(0x0F70,BEE_BUFFER_MAX);   //缓存必须满足操作对应大小，不满
足整页时，后续数据被删空
    BEE_BUFFER[7]=0x55AA;
    BEE_WRITE_FUN(0x0F70,BEE_BUFFER_MAX);
#endif
//---------------------------------------- setp 5  end display
    while(1)
    {
        temp++;
        if(temp>30000)
        {
            temp=0;
            P16=!P16;
            P17=!P17;
        }
    }
}
/*****************************************************************
**************
```

```
 * 函数名     ：BEE_READ_ONE
 * 函数功能：以设定地址开始，读取一个数的BEE数据并返回结果
 * 入口参数：起始地址
 * 返回         ：无

*********************************************************************
*************/
unsigned int BEE_READ_ONE(unsigned int address)
{
#if  1     //1 选用C语言表达  0 选用嵌汇编表达，嵌汇编效率更高，但提示传递参数未
使用和函数无返回，可以忽略

    // BEE_READ_BUF;  需要定义全局的该变量，获取R7 R6的值用于返回，或直接使用
    //; 参数的使用
    BADDRH=(unsigned char)(address>>8);
    BADDRL=(unsigned char)address;
    __asm
        ;//备份中断使能寄存器
            BANKSEL _INTCTL
            MOV     R1,_INTCTL
        ;//关闭中断，BEE操作不可打断
            CLR     _INTCTL,_AIE
            JNB     _INTCTL,_AIE
            JMP     $-2
        ;//时钟频率备份及降频操作BEE,建议降频到1M,此时中断已被关
            MOV     R0,#0x30
            MOV     R2,_OSCCTL
            MOV     _OSCCTL,R0
        ;//硬件使能读操作
            BANKSEL _EECTL1
            MOV     R5,#0x81
            MOV     _EECTL1,R5
            NOPZ
            NOPZ
            NOPZ
            NOPZ
        ;//时钟与中断使能的还原
            MOV     _OSCCTL,R2
            AND     R1,#0xC0   ;//中断使能仅关系高2位
            ORL     INTCTL,R1
        ;//操作结果赋值到变量，用于返回
            BANKSEL _BEE_READ_BUF
            MOV (_BEE_READ_BUF),R6
            MOV (_BEE_READ_BUF+1),R7
```

```
    __endasm;
    return BEE_READ_BUF;
#else
    // 参数的传递使用  编译器自动变量 STK00  和 R0 ，其中R0为高位，返回使用STK00
和 R0 ，其中R0为高位
    __asm
    ;//传递操作地址
        BANKSEL _BADDRH
        MOV     _BADDRH,R0
        BANKSEL STK00
        MOV     R0,STK00
        BANKSEL _BADDRL
        MOV     _BADDRL,R0
    ;//备份中断使能寄存器
        BANKSEL _INTCTL
        MOV     R1,_INTCTL
    ;//关闭中断，BEE操作不可打断
        CLR     _INTCTL,_AIE
        JNB     _INTCTL,_AIE
        JMP     $-2
    ;//时钟频率备份及降频操作BEE,建议降频到1M,此时中断已被关
        MOV     R0,#0x30
        MOV     R2,_OSCCTL
        MOV     _OSCCTL,R0
    ;//硬件使能读操作
        BANKSEL _EECTL1
        MOV     R5,#0x81
        MOV     _EECTL1,R5
        NOPZ
        NOPZ
        NOPZ
        NOPZ
    ;//时钟与中断使能的还原
        MOV     _OSCCTL,R2
        AND     R1,#0xC0   ;//中断使能仅关系高2位
        ORL     _INTCTL,R1
    ;//操作结果提供形式 整型数据返回结果使用  编译器自动变量 STK00  和 R0 ，其中
R0为高位
        BANKSEL STK00
        MOV     STK00,R6
        MOV     R0,R7
    __endasm;
#endif
```

```
}
/*********************************************************************
***************
 * 函数名      : BEE_READ_FUN
 * 函数功能：以设定地址开始，读取一定个数的BEE数据放置到数值BEE_BUFFER[x]中
 * 入口参数：起始地址，操作个数
 * 返回          : 无

*********************************************************************
*************/
void BEE_READ_FUN(unsigned int address,unsigned char length)
{
#if 1     //  1时效率比0小   1 使用传递参数   0 按编译器全嵌汇编实现，但编译提示
参数未使用，可以忽略
    //; 参数的使用
    BADDRH=(unsigned char) (address>>8);
    BADDRL=address;

    __asm
    ;//备份中断使能寄存器
        BANKSEL _INTCTL
        MOV     R1,_INTCTL
    ;//关闭中断，BEE操作不可打断
        CLR     _INTCTL,_AIE
        JNB     _INTCTL,_AIE
        JMP     $-2
    ;//时钟频率备份及降频操作BEE,建议降频到1M,此时中断已被关
        MOV     R0,#0x30
        MOV     R2,_OSCCTL
        MOV     _OSCCTL,R0
    ;//读取结果的存放起始RAM地址，即数组缓存区  BEE_BUFFER[x]
        MOV     R3,#_BEE_BUFFER
    __endasm;

    while(length--)   // 仅使用R0，不改变R1
    {
        __asm
        ;//硬件读
        BANKSEL _EECTL1
        MOV     R5,#0x81
        MOV     _EECTL1,R5
        NOPZ
        NOPZ
        NOPZ
```

```
        NOPZ
        ;//读结果处理
        BANKSEL _BEE_BUFFER
        ST      [R3],R6
        INC     R3
        ST      [R3],R7
        INC     R3
        ;//指向下一操作地址，建议每次操作内容在一个块中，此时不需要处理_BADDRL进
位的_BADDRH+1操作
        BANKSEL _BADDRL
        INC     _BADDRL
        JNB     _PSW,_Z
        INC     _BADDRH
        __endasm;
    }

    __asm
        ;//时钟与中断使能的还原
        MOV     _OSCCTL,R2
        AND     R1,#0xC0   ;//中断使能仅关系高2位
        ORL     _INTCTL,R1
    __endasm;
#else
    // 参数1的传递使用  编译器自动变量 STK00  和 R0 ，其中R0为高位
    // 参数2的传递使用  编译器自动变量 STK01
    // 整体实现的嵌汇编会提示参数未被使用，可以忽略
__asm
    ;//传递操作地址
    BANKSEL _BADDRH
    MOV     _BADDRH,R0
    BANKSEL STK00
    MOV     R0,STK00
    BANKSEL _BADDRL
    MOV     _BADDRL,R0
    ;//备份中断使能寄存器
    BANKSEL _INTCTL
    MOV     R1,_INTCTL
    ;//关闭中断，BEE操作不可打断
    CLR     _INTCTL,_AIE
    JNB     _INTCTL,_AIE
    JMP     $-2
    ;//时钟频率备份及降频操作BEE,建议降频到1M,此时中断已被关
    MOV     R0,#0x30
    MOV     R2,_OSCCTL
```

```
    MOV     _OSCCTL,R0
    ;//读取结果的存放起始RAM地址，即数组缓存区  BEE_BUFFER[x]
    MOV     R0,#_BEE_BUFFER

BEE_READ_FUN_LOOP:
    ;//硬件读
    BANKSEL _EECTL1
    MOV     R5,#0x81
    MOV     _EECTL1,R5
    NOPZ
    NOPZ
    NOPZ
    NOPZ
    ;//读结果处理
    BANKSEL _BEE_BUFFER
    ST      [R0],R6
    INC     R0
    ST      [R0],R7
    INC     R0
    ;//指向下一操作地址，建议每次操作内容在一个块中，此时不需要处理_BADDRL进位的
_BADDRH+1操作
    BANKSEL _BADDRL
    INC     _BADDRL
    JNB     _PSW,_Z
    INC     _BADDRH
    ;//读数量的判断
    BANKSEL STK01
    DECJZ   STK01
    JMP     BEE_READ_FUN_LOOP

    ;//时钟与中断使能的还原
    MOV     _OSCCTL,R2
    AND     R1,#0xC0   ;//中断使能仅关系高2位
    ORL     _INTCTL,R1
__endasm;
#endif
}
/*************************************************************************
***************
 * 函数名      : BEE_WRITE_FUN
 * 函数功能：按块或按页写入数据到BEE，个数参数只能为 4 8 12 16 ，地址必须为块的首
地址 如十六进制下末位 0  4  8  C
 *          如果地址不是页的首地址，必须确定后续块结果为0xFFFF，或前面操作过块首写，
使后续块值被0xFFFF，否则写结果异常。
```

```
 * 入口参数：待写地址，待写地址的数据
 * 返回          ：无
 * 写时间说明：除去代码，以整页BEE为例，操作完第一块需要4ms，另外3块需要2ms。即第一块执行整页的擦除后写自身块，其他块直接写。

*****************************************************************************/
void BEE_WRITE_FUN(unsigned int address,unsigned char length)
{
#if 1     //  1时效率比0小   1 使用传递参数   0 按编译器全嵌汇编实现，但编译提示参数未使用，可以忽略
    //; 参数的使用
    BADDRH=(unsigned char) (address>>8);
    BADDRL=address;

    __asm
    ;//备份中断使能寄存器
        BANKSEL _INTCTL
        MOV     R1,_INTCTL
    ;//关闭中断，BEE操作不可打断
        CLR     _INTCTL,_AIE
        JNB     _INTCTL,_AIE
        JMP     $-2
    ;//时钟频率备份及降频操作BEE,建议降频到1M,此时中断已被关
        MOV     R0,#0x30
        MOV     R2,_OSCCTL
        MOV     _OSCCTL,R0
    ;//读取结果的存放起始RAM地址，即数组缓存区  BEE_BUFFER[x]
        MOV     R3,#_BEE_BUFFER
    __endasm;

    while(length--)       // 仅使用R0，不改变R1
    {
        __asm
        ;//加载待写数据
        BANKSEL _BEE_BUFFER
        LD      R6,[R3]
        INC     R3
        LD      R7,[R3]
        INC     R3
        ;//硬件写
        MOV     R5 ,#0x84
        MOV     _EECTL1,R5
        MOV     R5,#0x69
```

```
        MOV     _EECTL2,R5
        MOV     R5,#0x96
        MOV     _EECTL2,R5
        SET     _EECTL1 , 1    ;// 写存在高压，高压还原添加空指令确保后续运行正
常
        NOPZ
        NOPZ
        NOPZ
        NOPZ
        NOPZ
        NOPZ                        ;// 至少10条
        NOPZ
        NOPZ
        NOPZ
        NOPZ
        MOV     R5,#0X80
        MOV     EECTL1,R5
        ;//指向下一操作地址，这里不考虑高位，BEE特性要求只能操作一页内的数据，不能
跨页
        BANKSEL _BADDRL
        INC     _BADDRL
        __endasm;
    }
    __asm
        ;//时钟与中断使能的还原
        MOV     _OSCCTL,R2
        AND     R1,#0xC0   ;//中断使能仅关系高2位
        ORL     _INTCTL,R1
    __endasm;
#else
    // 参数1的传递使用  编译器自动变量 STK00  和 R0 ，其中R0为高位
    // 参数2的传递使用  编译器自动变量 STK01
    // 整体实现的嵌汇编会提示参数未被使用，可以忽略
__asm
    ;//传递操作的地址(块首)
    BANKSEL _BADDRH
    MOV     _BADDRH,R0
    BANKSEL STK00
    MOV     R0,STK00
    BANKSEL _BADDRL
    MOV     _BADDRL,R0
    ;//备份中断使能寄存器
    BANKSEL    _INTCTL
    MOV     R1,_INTCTL
```

```
    ;//关闭中断，BEE操作不可打断
    CLR     _INTCTL,_AIE
    JNB     _INTCTL,_AIE
    JMP     $-2
    ;//时钟频率备份及降频操作BEE,建议降频到1M,此时中断已被关
    MOV     R0,#0x30
    MOV     R2,_OSCCTL
    MOV     _OSCCTL,R0
    ;//待写数据的存放起始RAM地址，即数组缓存区  BEE_BUFFER[x]
    MOV     R0,#_BEE_BUFFER

BEE_WRITE_PAGE_LOOP:
    ;//加载待写数据
    BANKSEL _BEE_BUFFER
    LD      R6,[R0]
    INC     R0
    LD      R7,[R0]
    INC     R0
    ;//硬件写
    MOV     R5 ,#0x84
    MOV     _EECTL1,R5
    MOV     R5,#0x69
    MOV     _EECTL2,R5
    MOV     R5,#0x96
    MOV     _EECTL2,R5
    SET     _EECTL1 , 1    ;// 写存在高压，高压还原添加空指令确保后续运行正常
    NOPZ
    NOPZ
    NOPZ
    NOPZ
    NOPZ
    NOPZ                        ;// 至少10条
    NOPZ
    NOPZ
    NOPZ
    NOPZ
    MOV     R5,#0X80
    MOV     EECTL1,R5
    ;//指向下一操作地址，这里不考虑高位，BEE特性要求只能操作一页内的数据，不能跨页
    BANKSEL _BADDRL
    INC     _BADDRL

    BANKSEL STK01
    DECJZ   STK01
```

```
    JMP     BEE_WRITE_PAGE_LOOP

    ;//时钟与中断使能的还原
    MOV     _OSCCTL,R2
    AND     R1,#0xC0   ;//中断使能仅关系高2位
    ORL     _INTCTL,R1
__endasm;
#endif
}

/*********************************************************************
***************
 * 函数名      : BEE_WRITE_ONE
 * 函数功能：向某一地址写入一个字，不建议使用，实际执行从页首读出，经对应地址的缓存
修改，在页起始回写，因此地址与页首偏移量需要数组大小满足需求。
 *          如当你BEE_BUFFER_MAX为4时，操作地址十六进制下的末尾必须只能是0 1 2 3
 * 建议：          可采用BEE_WRITE_FUN的传递个数为4 8 12 16 对应连续操作地址开始
的第一块 第二块 第三块 第四块
 *           但地址必须是某块的开始
 * 入口参数：待写地址，待写地址的数据
 * 返回          : 无

*********************************************************************
*************/
void BEE_WRITE_ONE(unsigned int address,unsigned int value)
{
    // 读出当前页到数据缓存，要求缓存大小必须满足写地址的偏移量要求
    BEE_READ_FUN(address&0xFFF0,BEE_BUFFER_MAX);
    // 修改待写数据在按照页排序中位置数据结果
    BEE_BUFFER[address&0xF] = value;
    // 整页数据回写
    BEE_WRITE_FUN(address&0xFFF0,BEE_BUFFER_MAX);
}
```

18.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

;;;;;;;;;;;;;;;;通用寄存器;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PSW_TEMP        .EQU   0X81       ;PSW的临时保存变量
PCH_TEMP        .EQU   0X82       ;PCH的临时保存变量

DELAY_NUM1          .EQU   0X83       ;延时用的临时变量
DELAY_NUM2          .EQU   0X84       ;延时用的临时变量
```

```
READ_BEE_L          .EQU   0X86       ;BEE结果低位
READ_BEE_H          .EQU   0X87       ;BEE结果高位

OPERATE_NUMBER      .EQU    0X88        ;每次读或写操作的长度设置

OPERATE_DATE_ADDR   .EQU    0X89        ;写入数据存放位置的首地址

OPERATE_DATE_BUFF   .EQU    0X90    ;联系地址占用大小与系统最大
OPERATE_NUMBER 值有关
          ;;;    0X90 ~  0xAF

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT
;;;;;;;;;;;;;;;中断部分;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
INTERRUPT
;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保
护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
    MOVR1 , PCH
    MOVPCH_TEMP, R1
    ;中断处理
    ;--------------------------------
    ;...
    ;...
    ;...
    ;--------------------------------
;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET
;;;;;;;;;;;;;;中断部分结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;入口函数,主程
```

```
序;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN
   CALL INIT_ALL
   CALL INIT_RAM

;;;;;;;;;;;;;;;;;;向RAM中写入32个字节（16个
字）;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    MOV R0 , #OPERATE_DATE_BUFF             ;写入RAM缓存区数据的首
地址
    MOV R2 , #0X00            ;待写入高位的数据
    MOV R3 , #0X00           ;待写入写低位的数据
    MOV R4 , #0X10            ;写入16个字,构成  0000  1111  2222
3333 4444  。。。 FFFF
WRITE_BUFFER
    ST [ R0 ] , R2
    INC R0
    ST [ R0 ] , R3
    INC R0

    ADD R2 , #0X11
    ADD R3 , #0X11

    DECJZ R4
    JMP WRITE_BUFFER
;;;;;;;;;;;;;;;;;将RAM中的数据写入BLOCK EE（页
写）;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;写入一页，地址
0F70~0F7F;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   MOV R7,#0x0F
   MOV R6,#0x70
   MOV R0,#0x10
   MOV OPERATE_NUMBER,R0

   MOV R2, #OPERATE_DATE_BUFF      ;数据RAM缓冲区的首地址
   CALL BEE_WRITE_FUN
;;;;;;;;;;;;;;;;再次写入一页，用以对照，地址
0F80~0F8F;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
   MOV R7,#0x0F
   MOV R6,#0x80
   MOV R0,#0x10
   MOV OPERATE_NUMBER,R0        ;计数已为零，需要重新赋值

   MOV R2, #OPERATE_DATE_BUFF      ;数据RAM缓冲区的首地址
   CALL BEE_WRITE_FUN
```

```
;;;;;;;;;;;;;;;;;;;;写BEE结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;读BEE函
数;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;从地址0X0F79读出的数据
    MOV R7,#0x0F
    MOV R6,#0x79
    CALL BEE_READ_ONE                ;读一个字
    MOV  READ_BEE_H,R7
    MOV  READ_BEE_L,R6
;;;;;;;;;;;;;;;;;;;;读BEE结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;块写操
作;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;将从地址0X0F79读出的数据写入0X0F99
;只能针对已经擦除过的地址进行块写，否则将导致写入不对，因此需要擦除
0x0F90开始的块
    CLR OPERATE_DATE_BUFF
    CLR OPERATE_DATE_BUFF+1       ; word 1    可送其他值
    CLR OPERATE_DATE_BUFF+2
    CLR OPERATE_DATE_BUFF+3       ; word 2    可送其他值
    CLR OPERATE_DATE_BUFF+4
    CLR OPERATE_DATE_BUFF+5       ; word 3    可送其他值
    CLR OPERATE_DATE_BUFF+6
    CLR OPERATE_DATE_BUFF+7       ; word 4    可送其他值

    MOV R7,#0x0F
    MOV R6,#0x90
    MOV R0,#0x04
    MOV OPERATE_NUMBER,R0            ;仅写第一块，并擦除当前页的后续
3块数据
    MOV R2, #OPERATE_DATE_BUFF       ;数据RAM缓冲区的首地址
    CALL BEE_WRITE_FUN              ;这里为无关内容项，写入的数据引用历
史
;***上面的部分是为了擦除此页，若此页已经擦除可以直接写入***

;***要写入的地址0X0F99在第三页的第三块第二个字，剩下三个字用0XFFFF补
齐***
    ;0x0FF8  0x0FFA  0x0FFB 为无关项，这里写入原始缓存数据

    MOV R0,READ_BEE_H
    MOV OPERATE_DATE_BUFF+2,R0
```

```
    MOV R0,READ_BEE_L
    MOV OPERATE_DATE_BUFF+3,R0    ; 修改所需要的数据

     MOV R7 ,#0X0F               ;地址从0x0F98开始写入
    MOV R6 ,#0X98                ;地址从0x0F98开始写入
    MOV R2, #OPERATE_DATE_BUFF       ;数据RAM缓冲区的首地址
    MOV R0 , #0X04
     MOV OPERATE_NUMBER , R0
    CALL BEE_WRITE_FUN
;;;;;;;;;;;;;;;;;;;;;块写BEE结束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;修改操作;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;这里将BEE地址OX0F77中的数据修改为0X55AA
;首先将OX0F77所在页中的数据全部读出存放在RAM中

;Read_EEPROM
     MOV R7,#0x0F
    MOV R6,#0x70
    MOV R0,#0x10
    MOV OPERATE_NUMBER,R0            ;16字
    MOV R2, #OPERATE_DATE_BUFF       ;数据RAM缓冲区的首地址
    CALL BEE_READ_FUN              ;
;OX0F77是该页第8个字，对应存在RAM缓存区的##0X9E和#0X9F
    MOV R0,#0x55
    MOV OPERATE_DATE_BUFF+14,R0
    MOV R0,#0xAA
    MOV OPERATE_DATE_BUFF+15,R0   ; 修改所需要的数据
;写部分
     MOV R7 ,#0X0F               ;地址从0x0F98开始写入
    MOV R6 ,#0X70                ;地址从0x0F98开始写入
    MOV R2, #OPERATE_DATE_BUFF       ;数据RAM缓冲区的首地址
    MOV R0 , #0X10
     MOV OPERATE_NUMBER , R0
    CALL BEE_WRITE_FUN
;;;;;;;;;;;;;;;;;;;;;修改BEE结束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MAIN_END
    NOPZ
    NOPZ
    SET P1,6
    NOPZ
    NOPZ
```

```
    CLR P1,6
    JMP $-5
    CRET
;;;;;;;;;;;;;;主程序结
束;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
    CALL #0xFFF
    MOV OSCCAL0, R0       ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0       ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0        ;设置为8M
    ;端口初始化
    ;端口初始化
    MOV R0 ,#0x08
    MOV TR0, R0            ;设置P1端口为输入
    MOV R0 ,#0x00
    MOV TR1, R0            ;设置P0端口为输入
    MOV TR2, R0            ;设置P2端口为输入

    MOV R0,#0xF0
    MOV P2,R0

    SET P1,6
    SET P1,7

    CLR INTCTL             ;关中断

    CRET
```

```
;*****************************************************************
*****************
;* 函 数 名: BEE_Read_ONE
;* 函数功能: 读取BLOKE EEPROM函数    字读
;* 入口参数: R7,R6 读取地址的高位与低位
;* 返    回:  R7,R6
;*R0-R7使用情况  R0  R1   R3  R5   R6  R7，数据的读写传递必须通过
R6,R7进行，其他不限定
;* R0 通用操作  R5 可用R0替代          R1 备份 时钟设置  R3 备份中断
设置
;*****************************************************************
*****************
BEE_READ_ONE
    ;加载到前要操作的首地址到BEE操作地址寄存器中
    MOV BADDRH ,R7 ;
    MOV BADDRL ,R6 ;
    ;降低系统的时钟频率使更加可靠的操作，建议减低到1M,保存原来
    MOV R1,OSCCTL    ;原操作值备份
    MOV R0 ,#0x30
    MOV OSCCTL ,R0
    NOP
    ;写BEE需要关闭中断使能，即过程不能被打断，系统未开启中断可以不用考
虑
    MOV R3,INTCTL    ;原操作值备份
    ;CLR INTCTL,PUIE
    CLR INTCTL,AIE
    JNB INTCTL,AIE
    JMP $-2
    ;BEE硬件写指令顺序，不需要修改
    MOV R5 ,#0X81
    MOV EECTL1, R5  ;开始读取BEE的值
    NOPZ            ;  .DW 0xFFFF
    NOPZ            ;  .DW 0xFFFF
    NOPZ            ;  .DW 0xFFFF       ;至少4个空指令
    NOPZ            ;  .DW 0xFFFF
    ;;结果保存在R7 R6寄存器中
    MOV OSCCTL,R1    ;恢复系统主频
    AND R3,#0xC0 ;中断使能仅关系高2位
    ORL INTCTL,R3    ;恢复中断使能状态
    NOP
    CRET
;*****************************************************************
*****************
```

```
;* 函 数 名: BEE_Read_Fun
;* 函数功能: 读取BLOKE EEPROM函数  连续
;* 入口参数: R7,R6 读取地址的高位与低位  OPERATE_NUMBER 操作数  R2
读入结果RAM起始地址
;* 返    回:  R2 指向RAM起始地址到长度个数的RAM结果刷新
;
;*R0-R7使用情况  R0  R1  R2  R3  R5  R6  R7，数据的读写传递必须通
过R6,R7进行，其他不限定
;* R0 通用操作  R5 可用R0替代      R2数据操作地址  R1 备份 时钟设置
R3 备份中断设置
;************************************************************
*****************
BEE_READ_FUN
    ;加载到前要操作的首地址到BEE操作地址寄存器中
    MOV BADDRH ,R7 ;
    MOV BADDRL ,R6 ;
    ;降低系统的时钟频率使更加可靠的操作，建议减低到1M,保存原来
    MOV R1,OSCCTL   ;原操作值备份
    MOV R0 ,#0x30
    MOV OSCCTL ,R0
    NOP
    ;写BEE需要关闭中断使能，即过程不能被打断，系统未开启中断可以不用考
虑
    MOV R3,INTCTL   ;原操作值备份
    ;CLR INTCTL,PUIE
    CLR INTCTL,AIE
    JNB INTCTL,AIE
    JMP $-2

RD_BEE_Continue
    ;BEE硬件写指令顺序，不需要修改
    MOV R5 ,#0X81
    MOV EECTL1, R5  ;开始读取BEE的值
    NOPZ          ;  .DW 0xFFFF
    NOPZ          ;  .DW 0xFFFF
    NOPZ          ;  .DW 0xFFFF      ;至少4个空指令
    NOPZ          ;  .DW 0xFFFF
    ;;结果转存 R2为定位的RAM地址
    ST [R2] ,R6  ;
    INC R2
    ST [R2] ,R7  ;
    INC R2

    INC BADDRL ; EEPROM地址的低8位加1，指向下一个地址
```

```
    ;;次数判断
    ; SET PSW,PR0
    ; CLR PSW,PR0
    DECJZ OPERATE_NUMBER   ;是否完成所需数量判断
    JMP RD_BEE_Continue
END_RD_BEE
    MOV OSCCTL,R1    ;恢复系统主频
    AND R3,#0xC0 ;中断使能仅关系高2位
    ORL INTCTL,R3    ;恢复中断使能状态
    NOP
    CRET
;*****************************************************************
*****************
;* 函 数 名: WRITE_BEE_LOOP
;* 函数功能:
;* 入口参数:  R7 R6 为操作使用，在之前用来传递 操作地址
OPERATE_NUMBER 操作数   R2 待写入结果RAM起始地址
;*
;*R0-R7使用情况  R0  R1   R2  R3  R5   R6  R7，数据的读写传递必须通
过R6,R7进行，其他不限定
;* R0 通用操作  R5 可用R0替代      R2数据操作地址   R1 备份 时钟设置
R3 备份中断设置
;* 返    回: 无
;*****************************************************************
*****************
BEE_WRITE_FUN
    ;加载到前要操作的首地址到BEE操作地址寄存器中
    MOV BADDRH ,R7 ;
    MOV BADDRL ,R6 ;
    ;降低系统的时钟频率使更加可靠的操作，建议减低到1M,保存原来
    MOV R1,OSCCTL    ;原操作值备份
    MOV R0 ,#0x30
    MOV OSCCTL ,R0
    NOP
    ;写BEE需要关闭中断使能，即过程不能被打断，系统未开启中断可以不用考
虑
    MOV R3,INTCTL    ;原操作值备份
    ;CLR INTCTL,PUIE
    CLR INTCTL,AIE
    JNB INTCTL,AIE
    JMP $-2
WRITE_BEE_Continue
    ;加载数据到R7,R6寄存器，形成待写入BEE的字
    LD R6 ,[R2]
```

```
    INC R2
    LD R7 ,[R2]
    INC R2
    ;BEE硬件写指令顺序，不需要修改
    MOV R5 ,#0X84
    MOV EECTL1 ,R5
    MOV R5 ,#0X69
    MOV EECTL2 ,R5
    MOV R5 ,#0X96
    MOV EECTL2 ,R5
    SET EECTL1 ,1    ;写 BLOCK EEPROM指令存在操作高压与还原
    ;;;;;;;;;;;;;;;;;为保可靠，至少10
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    NOPZ          ;同.DW 0xFFFF
    MOV R5 ,#0X80
    MOV EECTL1 ,R5 ; 关闭BLOCK EE的写操作，防止意外写
    ;指向下一个地址
    INC BADDRL              ; BLOCK EEPROM地址高8位不变，低8位加1
    ;;次数判断
    ; SET PSW,PR0
    ; CLR PSW,PR0
    DECJZ OPERATE_NUMBER   ;是否完成所需数量判断
    ;需继续
    JMP WRITE_BEE_Continue        ;继续写操作

END_WR_EEPROM
    MOV OSCCTL,R1    ;恢复系统主频
    AND R3,#0xC0 ;中断使能仅关系高2位
    ORL INTCTL,R3    ;恢复中断使能状态
    NOP

    CRET
;/*************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
```

```
;* 入口参数: 无
;* 返    回: 无
;*****************************************************************
*****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

    .end
```

实验十九、INT1中断实验

19.1 程序声明

KF8F系列单片机开发板演示程序

标　　题：INT1中断实验

项 目 名：19-INT1_Interrupt_TEST

开发环境：ChipON IDE

作　　者：上海芯旺微电子有限公司

功能简述：使能INT1外部中断功能，通过P11口改变P12口电平，当P12（INT1引

脚）有触发脉冲时产生中断，并在中断处理程序中改变LED显示状态。

19.2 硬件说明

使用杜邦线将单片机的P11口连接到单片机的P12口，改变P11口的输出电平，使得P12口的电平发生变化，从而产生中断，在中断处理程序中改变LED的亮灭状态，从而使得LED闪烁，接线图如图19.1所示。

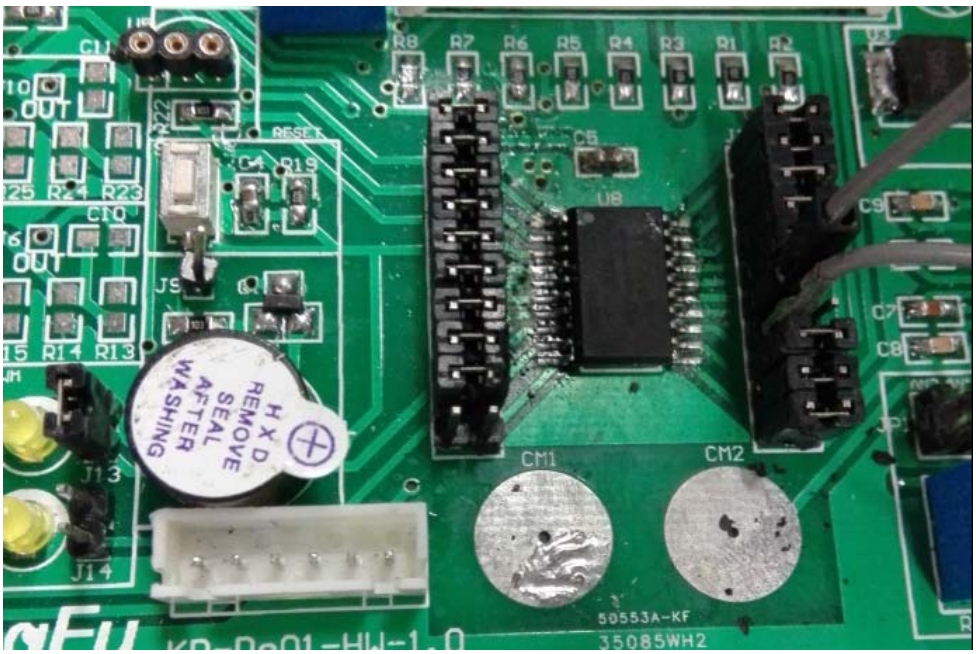


图19.1　P12和P11连接图

19.3 程序设计说明

在KF8F312芯片中，有三个外部中断源，分别为INT0、INT1和INT2。INT1中断通过寄存器EIE1中的INT1IE位置1使能INT1中断。通过PWMCTL中的INT1SE位设置触发边沿，INT1SE置1,将INT1设置为上升沿触发，清0设置为下降沿触发。EIF1中的INT1IF为INT1的中断标志位。INT1引脚有触发脉冲时，INT1IF被自动置1，如果INT1IE、PUIE和AIE位为1，则响应INT1中断。

除此之外，还需注意一下两点：

1、使用INT1外部中断功能，首先将相应引脚（P12）设置为数字输入模式。

2、使用外部中断时，需使能PUIE = 1。

19.4.1　C 程序代码

```
#include<KF8F312.h>
```

```
/*********************宏定义*****************************/
#define uchar unsigned char
#define uint  unsigned int
#define LED1 P16                                  // 对应Demo板上的D3
/*********************宏定义结束*********************** ***/

/************************
 * 函数名     : init_fun
 * 函数功能：初始化函数
 * 入口参数：无
 * 返回         : 无
 ************************/
void Init_fun()
{
    OSCCTL = 0x60;                              //设置系统时钟为8M

    /*********端口初始化***********/
    TR0 = 0x08;                                   //设置P03端口只能设置为输入
    TR1 = 0x04;                                   //P12口为外部中断引脚，需配置为
数字输入口设置P1端口为输出
    TR2 = 0x00;                                   //设置P2端口为输出

    P0 = 0x00;
    P1 = 0x00;
    P2 = 0xF0;

    ANSEL = 0;                                  //配置P12口为数字口
    EIE1 = 0X10;                                //使能INT1中断 INT1IE = 1
    PWMCTL = 0X40;                              //配置INT1为上升沿触发
    EIF1 = 0;                                   //清中断标志
    INTCTL = 0XC0;                                  //使能总中断AIE、INT1属于外部
中断，需使能外部中断PUIE = 1
}

/************************
 * 函数名     : Delay
 * 函数功能：短时间延时
 * 入口参数：无
 * 返回         : 无
 ************************/
void Delay()
{

    uchar i = 0,j = 0;
```

```
    for (i = 0;i < 200;i++)
        for (j = 0;j < 200;j++);
}

void main()
{
    Init_fun();

    while (1)
    {
        P11 = !P11;                //  改变P11口电平
        Delay();
    }

}
//中断函数
void int_fun() __interrupt
{
    if (INT1IF)
    {
        INT1IF = 0;                // 清中断标志
        LED1 = !LED1;              // 更改 D3 显示状态
    }
}
```

19.4.2 汇编程序代码

```
.INCLUDE "KF8F312.INC"

DELAY_NUM0              .EQU          0x80       ;延时用的临时变量
DELAY_NUM1              .EQU          0x81       ;延时用的临时变量
PSW_TEMP            .EQU          0X82       ;PSW的临时保存变量
PCH_TEMP            .EQU          0X83       ;PCH的临时保存变量

    .ORG 0X0000
    NOP
    JMP MAIN
    .ORG 0X0004
    JMP INTERRUPT
INTERRUPT

;;保护现场作用，根据实际情况进行保存，中断内部和外部都使用的资源需要保护，常规需要保护的内容有PSW PCH R0 R1
    SWAPR R1 ,PSW
    MOVPSW_TEMP, R1
```

```
    MOVR1 , PCH
    MOVPCH_TEMP, R1

;/*********************************中断处理程序
**********************************/
    JB EIF1,INT1IF
    JMP INT_END
    CLR EIF1,INT1IF                ;清中断标志

    MOV R1,#0X40                   ;P16输出状态取反   改变LED显示状态
    XOR P1,R1
;/*********************************中断处理程序
**********************************/

;恢复现场作用
INT_END
    MOVR1 , PCH_TEMP
    MOVPCH, R1
    SWAPR R1 , PSW_TEMP
    MOV PSW, R1
    IRET

MAIN
    CALL INIT_ALL                 ;初始化寄存器
    CALL INIT_RAM                 ;初始化RAM
LOOP
    MOV R0,#0X02                   ; p11状态取反
    XOR P1,R0

    CALL DELAY_MS
    CALL DELAY_MS
    CALL DELAY_MS
    JMP LOOP
    CRET
;*************************************************************
*****************
;* 函 数 名: INIT_ALL
;* 函数功能: 初始化函数,对各种寄存器的初始化
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
INIT_ALL
    ;调入校准信息到控制寄存器
```

```
    CALL #0xFFF
    MOV OSCCAL0, R0      ;晶振校准0
    NOP
    NOP
    CALL #0xFFE
    MOV OSCCAL1, R0      ;晶振校准1
    NOP
    NOP
    ;选择工作频率
    MOV R0 ,#0x60
    MOV OSCCTL, R0       ;设置为8M
    ;端口初始化
    MOV R0,#0X08         ;设置P03端口只能设置为输入
    MOV TR0,R0
    MOV R0,#0X04         ;P12口为外部中断引脚，需配置为数字输入口设置
P1端口为输出
    MOV TR1,R0
    MOV R0,#0X00
    MOV TR2,R0

    CLR P0
    CLR P1
    MOV R0,#0xF0
    MOV P2,R0

    CLR ANSEL            ;配置为数字口

    MOV R0,#0X10
    MOV EIE1,R0          ;使能INT1中断 INT1IE = 1
    MOV R0,#0X40
    MOV PWMCTL,R0        ;配置INT1为上升沿触发
    CLR EIF1             ;清中断标志
    MOV R0,#0XC0
    MOV INTCTL,R0        ;使能总中断AIE、INT1属于外部中断，需使能外
部中断PUIE = 1

    CRET

;/*************************************************************
*****************
;* 函 数 名: init_RAM
;* 函数功能: 初始化RAM
;* 入口参数: 无
;* 返    回: 无
```

```
;*************************************************************
****************/
INIT_RAM
    CLR    R2          ;传递默认RAM值
    ; CLR RAM SET 0  BANK0
    CLR    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X20     ;操作个数
    CALL     INIT_RAM_0
    ; CLR RAM SET 0  BANK1
    SET    PSW ,RP0
    MOV    R0 ,#0X80     ;起始地址0x80
    MOV    R1 ,#0X01     ;操作个数
    CALL     INIT_RAM_0
    ; 默认工作区域设定   BANK0
    CLR    PSW ,RP0
    ; 其他变量的数值操作



    CRET
;;;;;;;;;;;;;;;;;;执行默认数值装载到RAM区
INIT_RAM_0
    ST [R0],R2
    INC R0
    DECJZ R1
    JMP  INIT_RAM_0
    CRET

;*************************************************************
*****************
;* 函 数 名: DELAY
;* 函数功能: 延时函数,时间为1ms
;* 入口参数: 无
;* 返    回: 无
;*************************************************************
*****************
DELAY_MS
    MOV    R0 ,#0xC8
    MOV    DELAY_NUM0, R0
LP0
    CWDT
    MOV    R0 ,#0XC8
    MOV    DELAY_NUM1, R0
```

```
LP1
    DECJZ  DELAY_NUM1       ;DELAY_NUM2自减1，若为0，则跳过
    JMP    LP1
    DECJZ  DELAY_NUM0       ;DELAY_NUM1自减1，若为0，则跳过
    JMP    LP0
    CRET

    .END
```