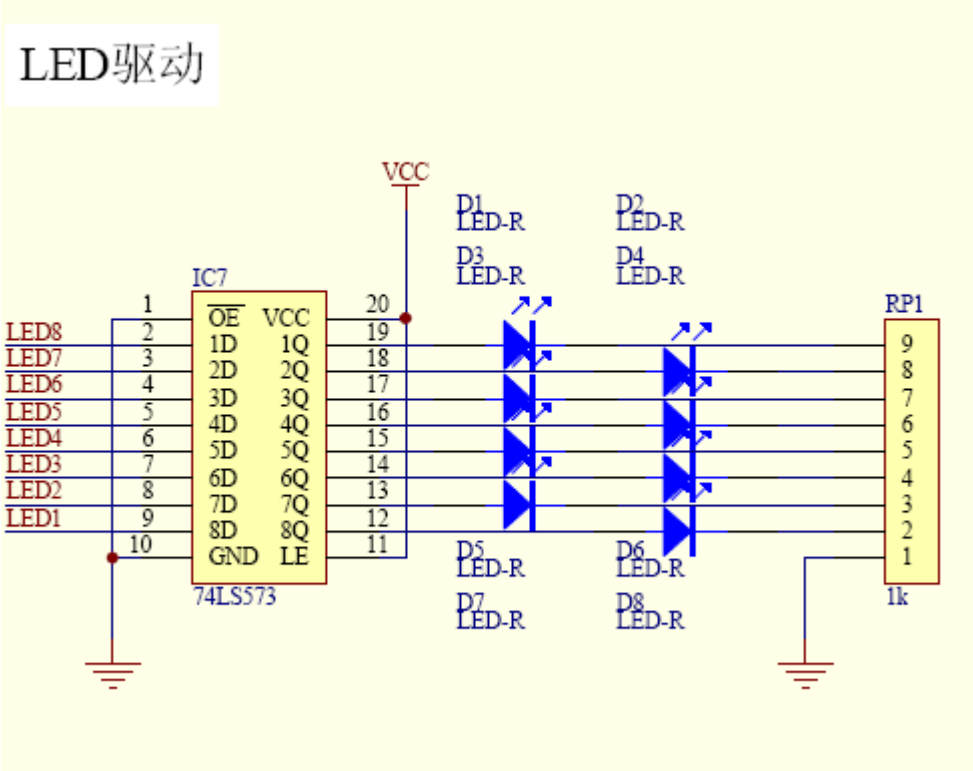
实验一：LED 灯显示实验

实验电路：



led_one：实验结果为实现 1 个灯闪烁。

led_all_timer：实验结果为控制 8 个 LED 灯同时闪烁。

led_run：实验结果为实现简单流水灯。

led_run_timer：实验结果为实现流水灯以三种流动方式和四种流动速度的不同组合而进行点亮"流动"

led_pwm：实验结果为用从 P2.3 和 P2.4 输出的 PWM 波形驱动 LED 闪烁；P2.3 口输出方波的占空比为 75%；P2.4 口输出方波的占空比为 25%。

程序实例：

led\led_one：

```
/*******************************************************************************/
#include<msp430x14x.h>
char led[8]={0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
void delay()
{
  unsigned int i;
  for(i=0;i<50000;i++);
}
void main()
{
  unsigned int k=1;
```

```
  WDTCTL = WDTPW + WDTHOLD; //关闭看门狗
  BCSCTL2 &=0xc0;           //XT2CLK+2 分频
  P2DIR = 0xff;
  for(;;)
  {
      P2OUT = led[k];
      delay();
      P2OUT = 0x00;
      delay();
  }
}
/*******************************************************************************************/
led\led_all_timer
/*******************************************************************************************/
/*************************************************
程序功能：控制8个LED闪烁，用于测试下载功能是否正常
-------------------------------------------------
测试说明：观察LED闪烁
*************************************************/
#include  <msp430x14x.h>

/***************主函数***************/
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;       //关闭看门狗
    CCTL0 = CCIE;                   //使能CCR0中断
    CCR0 = 2047;                    //设定周期0.5S
    TACTL = TASSEL_1 + ID_3 + MC_1; //定时器A的时钟源选择ACLK，增计数模式
    P2DIR = 0xff;                   //设置P2口方向为输出
    P2OUT = 0xff;

    _EINT();                        //使能全局中断
    LPM3;                           //CPU进入LPM3模式
}

/*****************************************
函数名称：Timer_A
功    能：定时器A的中断服务函数
参    数：无
返回值  ：无
*****************************************/
#pragma vector = TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
```

```
    P2OUT ^= 0xff;                    //P2口输出取反
}
/*******************************************************************************/
led\led_run
/*******************************************************************************/
#include<msp430x14x.h>
char led[8]={0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
void delay()
{
  unsigned int i;
  for(i=0;i<50000;i++);
}
void main()
{
  WDTCTL = WDTPW + WDTHOLD; //关闭看门狗
  BCSCTL2 &=0xc0;           //XT2CLK+2分频
  P2DIR = 0xff;
  P6DIR = 0xfc;                    //设置P6口方向为输出
  P6OUT = 0xff;
  for(;;)
  {
    for(int k=0;k<8;k++)
    {
      P2OUT = led[k];
      delay();
    }
  }
}
/*******************************************************************************/
led\led_run_timer
/*******************************************************************************/
/***********************************************
程序功能：实现流水灯以三种流动方式和四种流动速度
          的不同组合而进行点亮"流动"
-----------------------------------------------
测试说明：观察流水灯流动顺序和速度的变化
***********************************************/
#include  <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int  uint;
uint i = 0,j = 0,dir = 0;
uint flag = 0,speed = 0;  //flag--灯光流动方式，speed--灯光流动速度

/***************主函数***************/
```

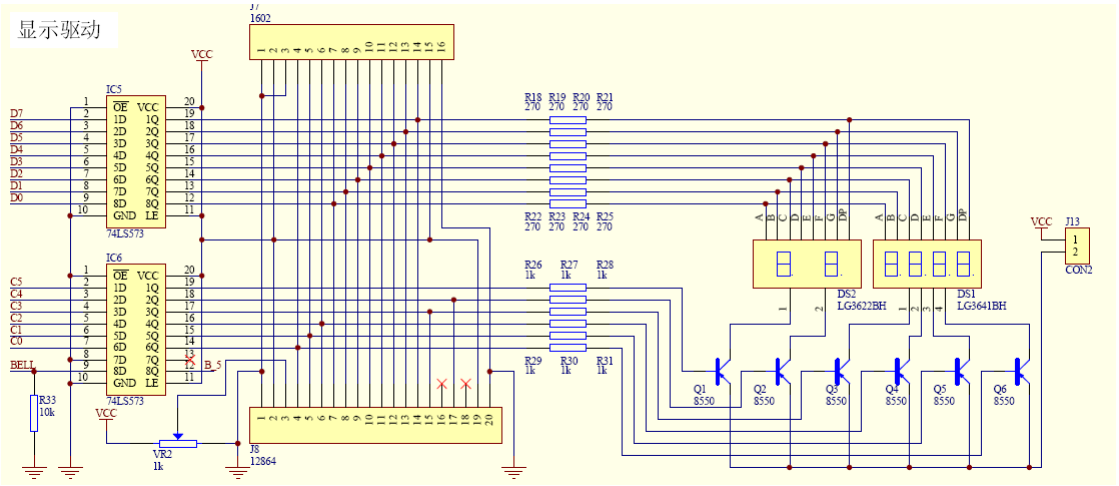
```
void main(void)
{

    WDTCTL = WDTPW + WDTHOLD;       //关闭看门狗

    CCTL0 = CCIE;                   //使能CCR0中断
    CCR0 = 50000;
    TACTL = TASSEL_2 + ID_3 + MC_1; //定时器A的时钟源选择SMCLK，增计数模式
    P2DIR = 0xff;                   //设置P2口方向为输出
    P2OUT = 0xff;
    P6DIR = 0xfc;                   //设置P6口方向为输出
    P6OUT = 0xff;
    _EINT();                        //使能全局中断
    LPM0;                           //CPU进入LPM0模式
}

/*****************************************
函数名称：Timer_A
功    能：定时器A的中断服务函数，在这里通过标志
          控制流水灯的流动方向和流动速度
参    数：无
返回值  ：无
*****************************************/
#pragma vector = TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
    if(flag == 0)
    {
        P2OUT = ~(0x80>>(i++));    //灯的点亮顺序D8 -> D1
    }
    else if(flag == 1)
    {
        P2OUT = ~(0x01<<(i++));    //灯的点亮顺序D1 -> D8
    }
    else
    {
        if(dir)                   //灯的点亮顺序  D8 -> D1,D1 -> D8,循环绕圈
        {
            P2OUT = ~(0x80>>(i++));
        }
        else
        {
            P2OUT = ~(0x01<<(i++));
        }
```

```
    }

    if(i == 8)
    {
        i = 0;
        dir = ~dir;
    }

    j++;
    if(j == 40)
    {
        i = 0;
        j = 0;
        flag++;
        if(flag == 4) flag = 0;
        switch(speed)
        {
        case 0:
            TACTL &=~ (ID0 + ID1);
            TACTL |= ID_3;
            break;
        case 1:
            TACTL &=~ (ID0 + ID1);
            TACTL |= ID_2;
            break;
        case 2:
            TACTL &=~ (ID0 + ID1);
            TACTL |= ID_1;
            break;
        case 3:
            TACTL &=~ (ID0 + ID1);
            TACTL |= ID_0;
            break;
        default:
            break;
        }
        if(flag != 3)    speed++;
        if(speed == 4) speed = 0;
    }
}

/*******************************************************************************/
led\led_pwm
/*******************************************************************************/
```

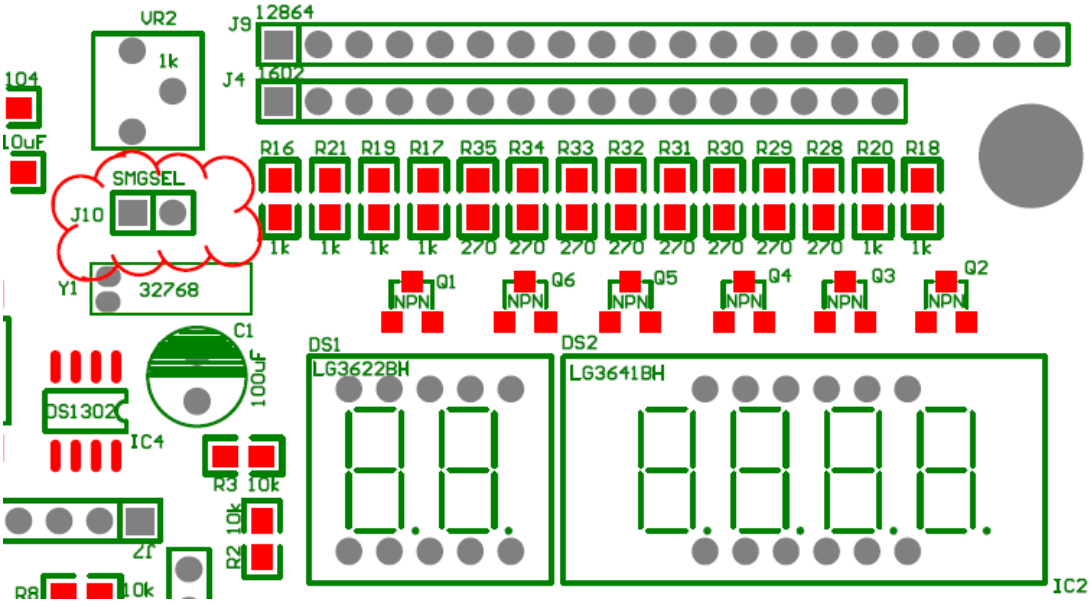
```
/*************************************************
程序功能：用从P2.3和P2.4输出的PWM波形驱动LED闪烁
          P2.3口输出方波的占空比为75%
          P2.4口输出方波的占空比为25%
-------------------------------------------------
测试说明：观察LED的亮灭的时间长短
*************************************************/
#include  <msp430x14x.h>
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // 关狗
  P2DIR = 0xff;                             // P2端口设置为输出
  P2OUT = 0x00;                             // 关闭其他LED
  P2SEL |= BIT3 + BIT4;                     // P2.3和P2.4连接内部模块
  CCR0 = 4096-1;                            // PWM周期为1S
  CCTL1 = OUTMOD_7;                         // CCR1 reset/set
  CCR1 = 3072;                              // CCR1 PWM duty cycle
  CCTL2 = OUTMOD_7;                         // CCR2 reset/set
  CCR2 = 1024;                              // CCR2 PWM duty cycle
  TACTL = TASSEL_1 + ID_3 + MC_1;           // ACLK/8, up mode

  _BIS_SR(LPM3_bits);                       // Enter LPM3
}
/*************************************************************************************/
```

实验二：数码管显示实验

实验电路：



在做数码管实验时，需要插上 J10 跳帽



smg_static：实验结果为在数码管上显示“2”

smg_dynamic：实验结果为在数码管上显示“HELLO”

smg_dynamic_timer：实验结果为在数码管上显示“543210”

smg\smg_static

```
/*****************************************************************************/
#include<msp430x14x.h>
unsigned char table[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
void delay()
{
  unsigned int i;
  for(i=0;i<50000;i++);
```

```
}
void main()
{
  WDTCTL = WDTPW + WDTHOLD; //关闭看门狗
  BCSCTL2 &=0xc0;           //XT2CLK+2分频
  P4DIR = 0xff;
  P6DIR = 0xFC;
  for(;;)
  {
          P4OUT=~table[2];
          P6OUT=~BIT2;//BIT7---BIT2位选
  }
}
/*******************************************************************************************/
smg\smg_dynamic
/*******************************************************************************************/
//*************************************************************
//*                  基本端口输出实验-显示                    *
//*************************************************************

#include "msp430x14x.h"

//************************数组阵*********************************
//共阴unsigned char seg[10]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};//段选信号
unsigned char seg[16]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,
0x88,0x83,0xc6,0xa1,0x86,0x8e};//0-f段选信号，共阳
unsigned char hello[5]={0x89,0x86,0xc7,0xc7,0xc0};//HELLO共阳段吗
unsigned char dig[6]={0xf8,0xf4,0xec,0xdc,0xbc,0x7c};//位码，由第一位到第六位单独显示//位选信
号
unsigned char led[9]={0x00,0x80,0xc0,0xe0,0xf0,0xf8,0xfc,0xfe,0xff};//LED灯显示序列

//*************************************************************
//*                  普通延时函数保证不闪就可以               *
//*************************************************************
void delay()
{
  unsigned int i;
  for (i=0;i<100;i++);
}
void delay1()
{
  unsigned int j;
  for(j=0;j<10000;j++);
}
```

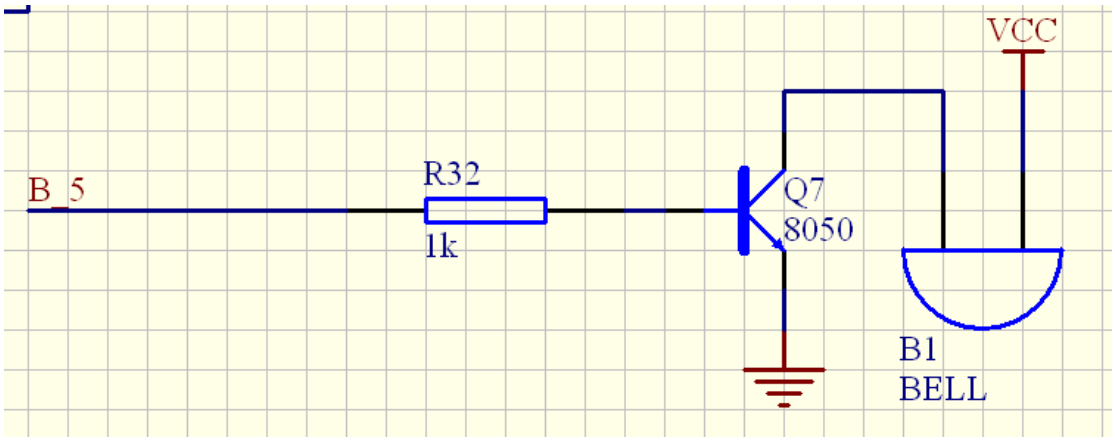
```
//*************************************************************
//*                 IO口初始化                                *
//*************************************************************
void Ioportinit()
{
  //P2DIR =0xff;  //p2输出
  P6DIR=0xfc;  //p6输出
  P4DIR=0xff;  //p4输出
  //P2DIR=0xff;
}
//*************************************************************
//*                 系统初始化                                *
//*************************************************************
void sys_init()
{
  WDTCTL = WDTPW + WDTHOLD; //关闭看门狗
  BCSCTL2 &=0xc0;           //XT2CLK+2分频
}
void main()
{
  sys_init();
  Ioportinit();
  while (1)
  {
          P4OUT=hello[4];
          P6OUT=~BIT2;
          delay();
          P4OUT=hello[3];
          P6OUT=~BIT3;
          delay();
          P4OUT=hello[2];
          P6OUT=~BIT4;
          delay();
          P4OUT=hello[1];
          P6OUT=~BIT5;
          delay();
          P4OUT=hello[0];
          P6OUT=~BIT6;
          delay();

  }
}
/*************************************************************************/
smg\smg_dynamic_timer
```

```
/*****************************************************************************/
#include <msp430x14x.h>
#define uint unsigned int
#define uchar unsigned char
  uint jj;
 uchar nn;
 unsigned char table[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
void initsys(void)
{
  uchar k;
  BCSCTL1 &= ~XT2OFF;    //开启XT2晶振
  do
  {
    IFG1 &= ~OFIFG;            // 清除振荡器失效标志
    for(k=0xee;k>0;k--)       //延时
      {
       ;
      }
    }
   while((IFG1 & OFIFG)!=0);         //判断XT2是否起振
   BCSCTL2 = SELM_2 + SELS;      // 选择MCLK SMCLK 为XT2
}                                  //初始化

void main(void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  initsys();
  _EINT();
  TACTL = TASSEL1 + TACLR;
  CCTL0 =CCIE;
  CCR0 = 8000;
  TACTL |= MC1;   //连续计数模式;启动
  P4DIR |= 0xff;
  P6DIR |= 0xfc;
  while(1);
}
 #pragma vector= TIMERA0_VECTOR
 __interrupt void Timer_A(void)
 {
   CCR0 += 8000;
   switch(nn)

    {
```

```
    case 0x00:P4OUT=~table[nn];P6OUT = ~BIT2;  break;
    case 0x01:P4OUT=~table[nn];P6OUT = ~BIT3;  break;
    case 0x02:P4OUT=~table[nn];P6OUT = ~BIT4;  break;
    case 0x03:P4OUT=~table[nn];P6OUT = ~BIT5;  break;
    case 0x04:P4OUT=~table[nn];P6OUT = ~BIT6;  break;
    case 0x05:P4OUT=~table[nn];P6OUT = ~BIT7;  break;
    }
    nn++;
   nn=nn%6;
 }
/*******************************************************************************/
```

实验三：蜂鸣器实验

实验电路：



bell：实验结果为蜂鸣器发出报警声音。
bell_music：实验结果为 430 控制蜂鸣器演奏歌曲《祝你平安》
bell\bell

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

#include "MSP430X14X.h"
#define uchar unsigned char
#define uint  unsigned int

unsigned char Count;

uint get[20]={0};
/*-------------------------------------*/

/*-------------关闭看门狗--------------*/
void close_WDT()
{WDTCTL = WDTPW + WDTHOLD;
}
/*-------------------------------------*/

/*------------430软件复位--------------*/
void reset_mcu()
{WDTCTL = 0x0000;
}
/*--------------------------------*/


/*----------设置看门狗定时器(定时模式)-----------*/
```

```
void set_wdt_timer()
{WDTCTL=WDT_MDLY_32;//定时时间的设置参见  MSP430X14X.h
 IE1 |= WDTIE;     //允许  wdt_timer  中断
 _EINT();
}
    /*看门狗中断处理函数*/
#pragma vector = WDT_VECTOR
__interrupt void wdt_timer()
{_NOP();
}
/*--------------------------------*/
/*-------设置系统时钟-------------*/
void set_sysclk()
{ uchar i;
  BCSCTL1 &= ~XT2OFF;
  do
  {
   IFG1 &= ~OFIFG;
   for(i=0xff;i>0;i--);
  }
  while((IFG1&OFIFG)!=0);
  BCSCTL2 |= SELM_2+DIVM_3;  //MCLK ---1MHz
  BCSCTL2 |= SELS+DIVS_3; //SMCLK --- 1MHz
  for(i=0Xff;i>0;i--);
  //P5DIR|=0X10;
  //P5SEL|=0X10;
}
/*--------------------------------*/

/*------------------------------------------------
功能:1MS延时子程序
------------------------------------------------*/
void Delay_xMs(unsigned int x)
     {
         unsigned int i,j;
         for( i =0;i < x;i++ )
               for( j =0;j<30;j++ );

     }

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;
  set_sysclk();
```

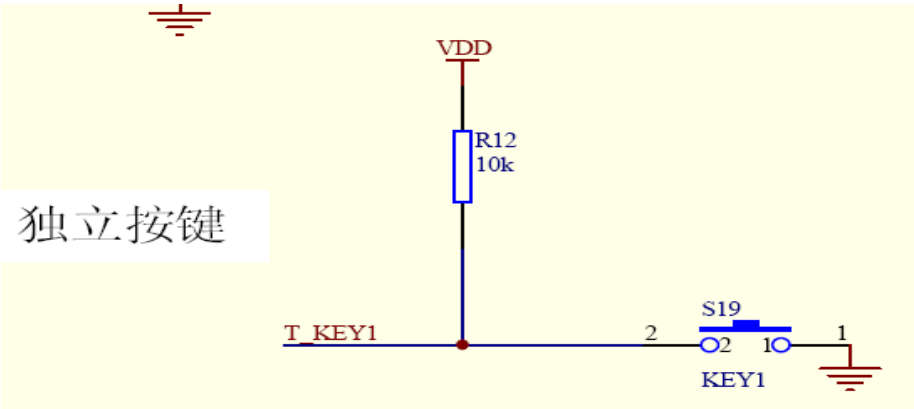
```
  P6DIR = 0xff;
  while(1)
     {
       P6OUT ^=BIT1;
           Delay_xMs(1000);
     }
}
/*******************************************************************************/
bell\bell_music
/*******************************************************************************/
/*********************************************
程序功能：MCU控制蜂鸣器演奏歌曲《祝你平安》
---------------------------------------------
测试说明：聆听蜂鸣器“唱出”的乐曲
*********************************************/
#include  <msp430x14x.h>
#include  "music.h"
typedef unsigned char uchar;
typedef unsigned int  uint;
#define   Buzzer        BIT1
#define   Buzzer_Port   P6OUT
#define   Buzzer_DIR    P6DIR

uchar counter;
void Play_Song(void);

/***************主函数***************/
void main(void)
{
    uchar i;

    WDTCTL = WDTPW + WDTHOLD;                 //关闭看门狗
    /*------选择系统主时钟为8MHz-------*/
    BCSCTL1 &= ~XT2OFF;                       // 打开XT2高频晶体振荡器
    do
    {
        IFG1 &= ~OFIFG;                       //清除晶振失败标志
        for (i = 0xFF; i > 0; i--);           // 等待8MHz晶体起振
    }
    while ((IFG1 & OFIFG));                   // 晶振失效标志仍然存在?
    BCSCTL2 |= SELM_2 + SELS;                 //主时钟和从时钟都选择高频晶振

    //设置定时器A每10ms中断一次
    CCTL0 = CCIE;
```

```
    CCR0 = 10000;
    TACTL |= TASSEL_2 + ID_3;
    //设置控制蜂鸣器的IO方向为输出
    Buzzer_DIR |= Buzzer;
    //打开全局中断
    _EINT();
    //循环演奏歌曲
    while(1)
    {
        Play_Song();
    }
}

/*****************************************
函数名称：TimerA_ISR
功    能：定时器A的中断服务函数
参    数：无
返回值  ：无
*****************************************/
#pragma vector = TIMERA0_VECTOR
__interrupt void TimerA_ISR(void)
{
    counter++;
}
/*****************************************
函数名称：Delay_Nms
功    能：延时N个ms的函数
参    数：n--延时长度
返回值  ：无
*****************************************/
void Delay_Nms(uchar n)
{
    uchar i,j;

    for( i = 0;i < n; i++ )
    {
        for( j = 0;j < 3;j++ )
          _NOP();
    }
}
/*****************************************
函数名称：Play_Song
功    能：播放《祝你平安》的乐曲
参    数：无
```

```
返回值  ：无
*****************************************/
void Play_Song(void)
{
    uchar Temp1,Temp2;
    uchar addr = 0;

    counter = 0; //中断计数器清0
    while(1)
    {
        Temp1 = SONG[addr++];
        if ( Temp1 == 0xFF )        //休止符
        {
            TACTL &=~MC_1;          //停止计数
            Delay_Nms(100);
        }
        else if ( Temp1 == 0x00 )   //歌曲结束符
        {
            return;
        }
        else
        {
            Temp2 = SONG[addr++];
            TACTL |=MC_1;           //开始计数
            while(1)
            {
                Buzzer_Port ^= Buzzer;
                Delay_Nms(Temp1);
                if ( Temp2 == counter )
                {
                    counter = 0;
                    break;
                }
            }
        }
    }
}
/*****************************************************************************/
```

实验四：独立按键实验

实验电路：



key_one_intr：中断方式按键，实验结果为按下按键在数码管上显示数值累加。
key1：实验结果为按下按键在数码管上显示数值累加。.

key\key1

```
/*****************************************************************************/
//-------------------------------------------------
//测试说明：按动K1~k2二个按键，观察数码管显示
//*************************************************/
#include <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int  uint;
//#define keyin1    (P1IN & 0x01)
#define key_in    (P1IN & 0x80)

//数码管7位段码：0--f
uchar scandata[16] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
                      0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

void delay(void);

/*******************主函数*******************/
void main( void )
{
    int key_count=0;
    WDTCTL = WDTPW + WDTHOLD;    //关闭看门狗

    P1DIR = 0x00;                //设置P1.0~P.3为输入状态，P.7为输出
    P4DIR = 0xff;
```

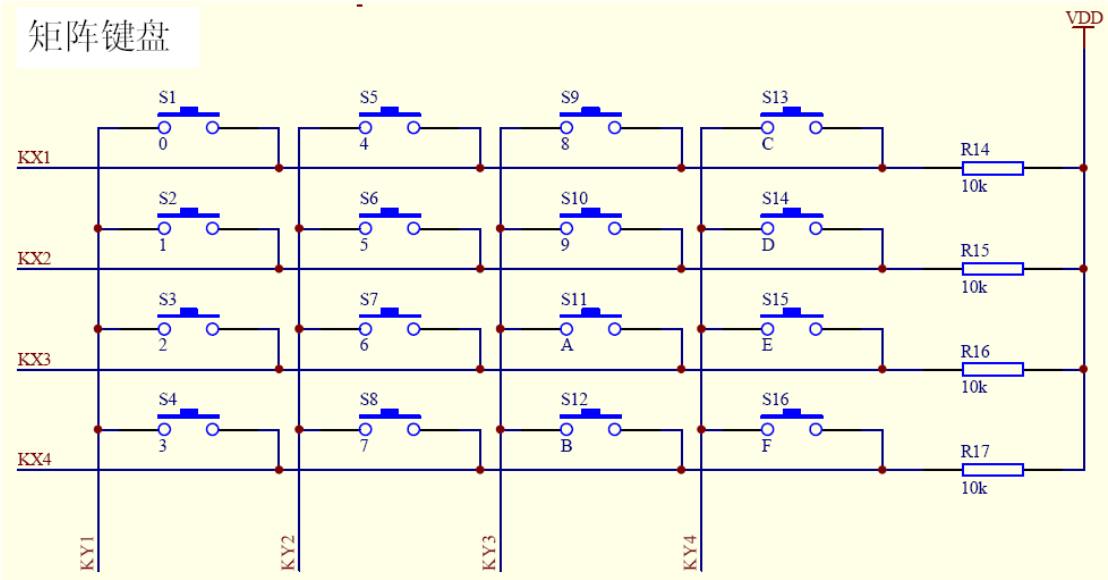
```
    P6DIR = 0xfc;
    P4OUT = 0xff;
    P6OUT = 0x55;
    while(1)
    {
        if(key_in != 0x80)      //如果有键被按下
        {
            delay();            //延时消抖
            if(key_in != 0x80)  //再次检测按键状态
            {
                key_count++;   //用一位数码管显示
                delay();delay();
                if(key_count==16)
                {
                    key_count=0;
                }
            }
        }

        P4OUT = ~scandata[key_count];

    }
}
/*******************************************
函数名称：delay
功    能：用于消抖的延时
参    数：无
返回值  ：无
*******************************************/
void delay(void)
{
    uint tmp;

    for(tmp = 12000;tmp > 0;tmp--);
}
/*****************************************************************************/
key\key_one_intr
/*****************************************************************************/
//-----------------------------------------------
//测试说明：按动K1~k2二个按键，观察数码管显示
//***********************************************/
#include <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int  uint;
```

```
//#define keyin1    (P1IN & 0x01)
#define key_in    (P1IN & 0x80)

//数码管7位段码：0--f
uchar scandata[16] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
                      0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

void delay(void);

/*******************主函数*******************/
void main( void )
{
    int key_count=0;
    WDTCTL = WDTPW + WDTHOLD;   //关闭看门狗

    P1DIR = 0x00;               //设置P1.0~P.3为输入状态，P.7为输出
    P4DIR = 0xff;
    P6DIR = 0xfc;
    P4OUT = 0xff;
    P6OUT = 0x55;
    while(1)
    {
        if(key_in != 0x80)      //如果有键被按下
        {
            delay();            //延时消抖
            if(key_in != 0x80)  //再次检测按键状态
            {
                key_count++;   //用一位数码管显示
                delay();delay();
                if(key_count==16)
                {
                    key_count=0;
                }
            }
        }

        P4OUT = ~scandata[key_count];

    }
}
/*****************************************
函数名称：delay
功    能：用于消抖的延时
参    数：无
```

```
返回值  ：无
*******************************************/
void delay(void)
{
    uint tmp;

    for(tmp = 12000;tmp > 0;tmp--);
}
/*****************************************************************************/
```

实验五：矩阵键盘 4 转独立键盘实验

实验电路：



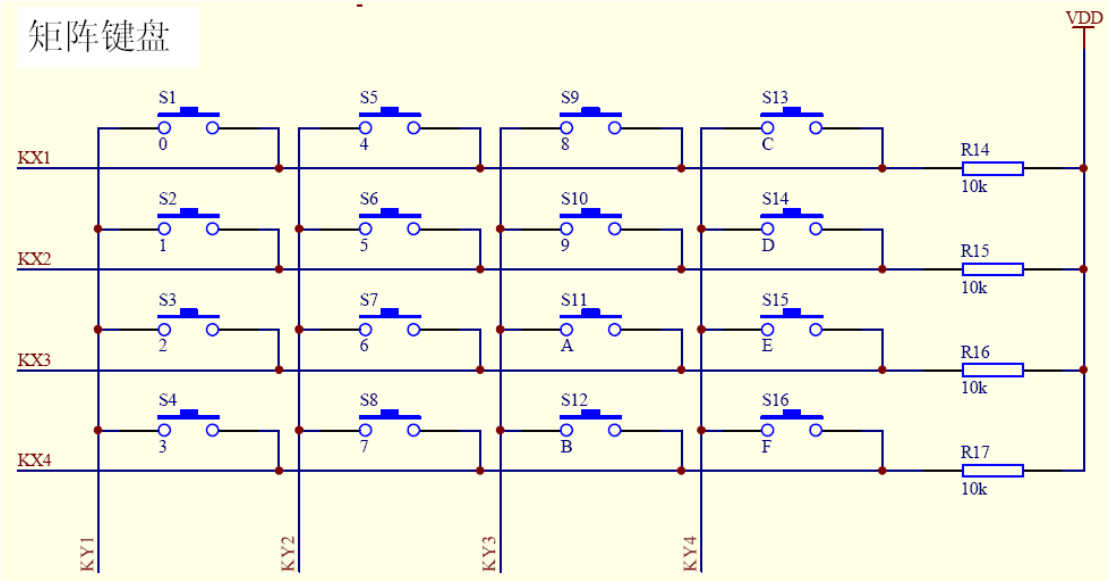
key441：实验结果为按下矩阵键盘第一行，键值显示在数码管上。

```
/*********************************************************************************************/
/*************************************************
测试说明：按动K2四个按键，观察数码管显示
*************************************************/
#include <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int  uint;
#define keyin    (P1IN & 0x0f)

//数码管7位段码：0--f
uchar scandata[16] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
                      0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};

void delay(void);

/*******************主函数*******************/
void main( void )
{
   unsigned char temp,i=0;
    WDTCTL = WDTPW + WDTHOLD;    //关闭看门狗
    BCSCTL2 &=0xc0;             //XT2CLK+2分频
  P4DIR=0xff;
  P1DIR=0x00;
  P6DIR=0xfc;
```

```
  P4OUT=0xff;
    P6OUT=0x04;
    while(1)
    {
        if(keyin != 0x0f)       //如果有键被按下
        {
            delay();            //延时消抖
            if(keyin != 0x0f)   //再次检测按键状态
            {
                temp=keyin;
                   switch(temp)    //转换键值
                {
                case 0x0E:
                            i = 1;break;
                case 0x0D:
                            i = 2;break;
                case 0x0B:
                            i = 3;break;
                case 0x07:
                            i = 4;break;
                default:
                            i = 0;break;
                }

              P4OUT =~scandata[i]; //用一位数码管显示
                delay();delay();
            }
        }
    }
}
/*****************************************
函数名称：delay
功    能：用于消抖的延时
参    数：无
返回值  ：无
*****************************************/
void delay(void)
{
    uint tmp;
        for(tmp = 12000;tmp > 0;tmp--);}
/*************************************************************************/
```

实验六：矩阵键盘实验

实验电路：



key44：实验结果为按下矩阵键盘，键值显示在数码管上。

key44uart：实验结果为 MCU 通过串口向 PC 机发送 4X4 键盘的键值。

key44\key44

```
/*******************************************************************************/
#include "msp430x14x.h"

void sys_init()
{
  WDTCTL = WDTPW + WDTHOLD; //关闭看门狗
  BCSCTL2 &=0xc0;           //XT2CLK+2分频
}

void io_init()
{
  P4DIR=0xff;
  P5DIR=0xf0;
  P6DIR=0xfc;
  P4OUT=0xff;
}

int key(int c)
{
  if (!(P1IN&BIT0))
    c+=1;
  else if (!(P1IN&BIT1))
    c+=2;
```

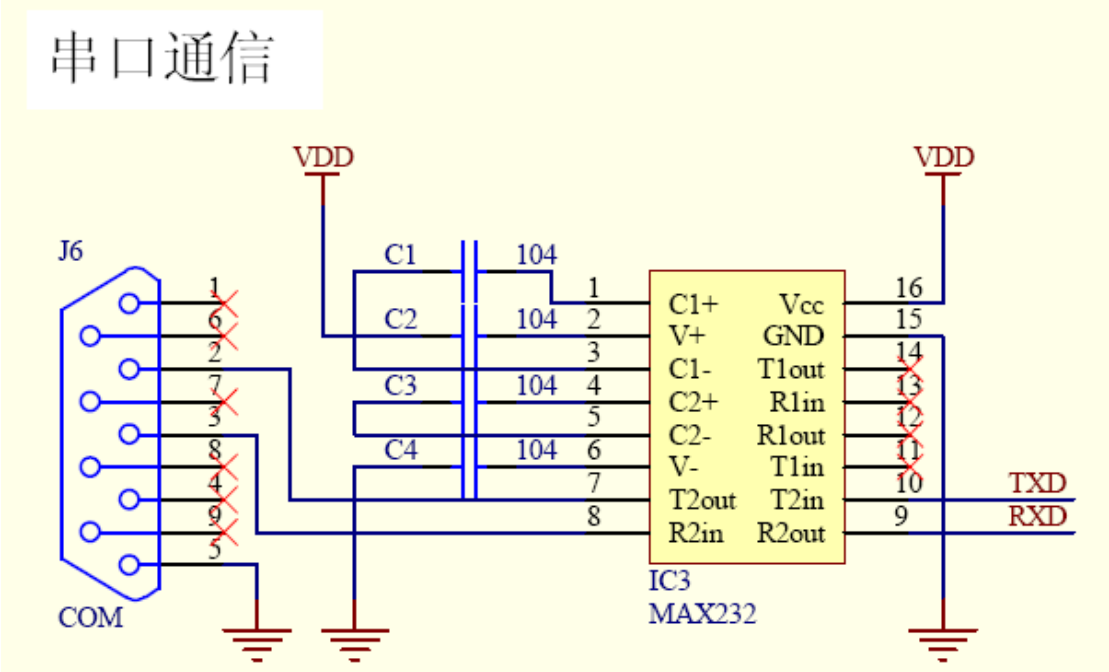
```
  else if (!(P1IN&BIT2))
    c+=3;
  else if (!(P1IN&BIT3))
    c+=4;
  else c=0;
  return c;
}
 void main()
{

  int a;
  unsigned char b[17]={0xff,0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,
0x88,0x83,0xc6,0xa1,0x86,0x8e};//0-f段选信号，共阳

  sys_init();
  io_init();

  while (1)
  {
    a=0;
    P5OUT=~BIT4;
    a+=key(0);

    P5OUT=~BIT5;//拉低P2.1
    a+=key(4);  //读取当时键值
    P5OUT=~BIT6;//拉低P2.2
    a+=key(8);  //读取当时键值
    P5OUT=~BIT7;//拉低P2.2
    a+=key(12);  //读取当时键值
    P6OUT=0xf8;
    P4OUT=b[a];
  }
}

/*****************************************************************************/
key44\key44uart
/*****************************************************************************/
/**************************************************
程序功能：MCU通过串口向PC机发送4X4键盘的键值
--------------------------------------------------
通信格式：N.8.1,  9600
--------------------------------------------------
测试说明：打开串口调试助手，正确设置通信格式，按动4X4
          键盘观察屏幕显示的按键键值。
```

```
************************************************/
#include  <msp430x14x.h>
#include  "keypad.c"
//引用外部变量的声明
extern unsigned char key_Pressed;
extern unsigned char key_val;
extern unsigned char key_Flag;

void PutString(uchar *ptr);
void PutChar(uchar zifu);
/******************主函数******************/
void main(void)
{
  uchar *tishi = "This Key's ID is:";

    WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
    P3SEL |= 0x30;                            // P3.4,5 = USART0 TXD/RXD
    ME1 |= UTXE0;                             // Enable USART0 TXD/RXD
    UCTL0 |= CHAR;                            // 8-bit character
    UTCTL0 |= SSEL0;                          // UCLK = ACLK
    UBR00 = 0x03;                             // 32k/9600 - 3.41
    UBR10 = 0x00;                             //
    UMCTL0 = 0x4A;                            // Modulation
    UCTL0 &= ~SWRST;                          // Initialize USART state machine

    Init_Keypad();                            //初始化键盘端口
    while(1)
    {
        Key_Event();

        if(key_Flag == 1)
        {
            key_Flag = 0;
            PutString(tishi);
            PutChar(key_val);
        }
    }
}
/*****************************************
函数名称：PutSting
功    能：向PC机发送字符串
参    数：ptr--指向发送字符串的指针
返回值  ：无
*****************************************/
```

```
void PutString(uchar *ptr)
{
      while(*ptr != '\0')
      {
            while (!(IFG1 & UTXIFG0));                // TX缓存空闲?
            TXBUF0 = *ptr++;                          // 发送数据
      }
      while (!(IFG1 & UTXIFG0));
      TXBUF0 = '\n';
}
/*****************************************
函数名称：PutChar
功    能：向PC机发送一个字符对应的ASCII码
参    数：zifu--发送的字符
返回值  ：无
*****************************************/
void PutChar(uchar zifu)
{
      while (!(IFG1 & UTXIFG0));
      if(zifu > 9)            //发送键值1~16对应的ASCII码
      {
          TXBUF0 = 0x30 + zifu/10;
          while (!(IFG1 & UTXIFG0));
          TXBUF0 = 0x30 + zifu%10;
      }
      else
      {
          TXBUF0 = 0x30 + zifu;
      }
      while (!(IFG1 & UTXIFG0));
      TXBUF0 = '\n';          //发送回车字符
}
/*******************************************************************************/
```

实验七：串口实验

实验电路：



uart0：实验结果为 MCU 不停向 PC 机发送数据,在屏幕上显示 0~127 对应的 ASCII 字符

uart1：实验结果为正确设置通信格式(8 个数据位，1 个停止位，波特率 9600)，在 PC 屏幕上显示 www.22sky.com。

uart2：实验结果为程序功能：接收来自 PC 机的字符，然后重新发送给 PC 机。打开串口调试助手，正确设置通信格式(8 个数据位，1 个停止位，波特率 9600)，向学习板发送一个字符，观察是否收到回发的字符，以及是否正确。

uart\uart0

```
/*******************************************************************************/
/*****************************************************
程序功能：MCU不停向PC机发送数据,在屏幕上显示0~127对应
          的ASCII字符
------------------------------------------------------
通信格式：N.8.1, 9600
------------------------------------------------------
测试说明：打开串口调试助手，正确设置通信格式，观察屏幕
*****************************************************/

#include  <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int  uint;
void Delays(void);
void PutString(uchar *ptr);
```

```
//////////////////////////////////////////
void initsys(void)
{
  BCSCTL1 &= ~XT2OFF;
  uchar i;
  do
  {
    IFG1 &= ~OFIFG;
    for(i=0xee;i>0;i--)
    {
      ;
    }
  }
  while((IFG1&OFIFG)!=0);
  BCSCTL2 |= SELM_2 + SELS;
}
void set()
{
  U0CTL |= SWRST;       //开启设置
  U0TCTL = SSEL1;   //选择 SMCLK=6M         波特率=9600
  /* 6M / 9600 = 625        625 = 0x0271     0.000 * 8= 0*/
  /* 32768/9600=3.4133    0.4133*8=4*/
  U0BR0 = 0x41;
  U0BR1 = 0x03;
  U0MCTL =0x00;     //00000000
  U0CTL |= CHAR;        //长度8位
  U0CTL &= ~SWRST;

}
/*******************主函数*********************/
void main(void)
{
    uchar *tishi = " MCU sends 0~127 to PC and the\
                   \n screen will display their corresponding\
                   \n ASCII code as follows:";
    uchar value = 0;

  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  initsys();
  P3DIR |= 0x10;
  P4DIR |= 0xff;
  P4OUT = 0x00;
  P3SEL |= 0x30;
```

```
  set();
  ME1 |= UTXE0 + URXE0;
  IE1 |= URXIE0;

  _EINT();
  ////////////////////////////////////////////
    PutString(tishi);
    while(1)
    {
        while (!(IFG1 & UTXIFG0));
        TXBUF0 = value++;
        value &= 0x7f;                          // 保证value的数值小于128
        while (!(IFG1 & UTXIFG0));
        TXBUF0 = '\n';
        Delays();
        Delays();
        Delays();
        Delays();
        Delays();
        Delays();
        Delays();
    }
}
/*****************************************
函数名称：PutSting
功    能：向PC机发送字符串
参    数：无
返回值  ：无
*****************************************/
void PutString(uchar *ptr)
{
      while(*ptr != '\0')
      {
            while (!(IFG1 & UTXIFG0));                // TX缓存空闲?
            TXBUF0 = *ptr++;                        // 发送数据
      }
      while (!(IFG1 & UTXIFG0));
      TXBUF0 = '\n';
}
/*****************************************
函数名称：Delays
功    能：延时一会
参    数：无
返回值  ：无
```

```
*****************************************/
void Delays(void)
{
    uchar i=20;
    uint j;

    while(i--)
    {
            j=2000;
            while(j--);
    }
}
/*****************************************************************************************/
```
uart\uart1
```
/*****************************************************************************************/
/*****************************************************
程序功能：在屏幕上显示www.22sky.com
-----------------------------------------------------
通信格式：N.8.1, 9600
-----------------------------------------------------
测试说明：打开串口调试助手，正确设置通信格式，观察屏幕
*****************************************************/
#include<msp430x14x.h>
#define uchar unsigned char
#define uint unsigned int
//uchar data[10] = {0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09};
uchar data[] = {'w','w','w','.','2','2','s','k','y','.','c','o','m','\n'};
uchar j=0;
uchar k=0;
delay ()
{
  uchar n;
  for(n=100;n>0;n--)
  {
    ;
  }
}
void initsys(void)
{
  BCSCTL1 &= ~XT2OFF;
  uchar i;
  do
  {
    IFG1 &= ~OFIFG;
```

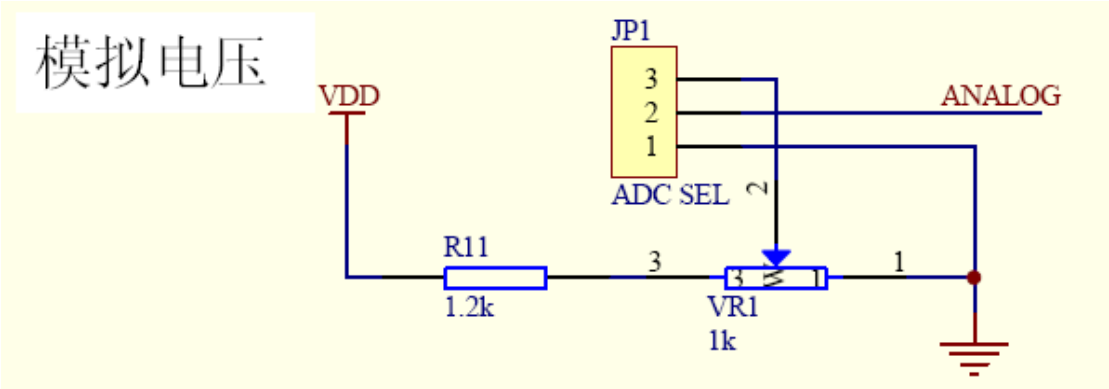
```
    for(i=0xee;i>0;i--)
    {
      ;
    }
  }
  while((IFG1&OFIFG)!=0);
  BCSCTL2 |= SELM_2 + SELS;
}
void set()
{
  U0CTL |= SWRST;       //开启设置
  U0TCTL = SSEL1;   //选择 SMCLK=6M         波特率=9600
  /* 6M / 9600 = 625        625 = 0x0271     0.000 * 8= 0*/
  /* 32768/9600=3.4133    0.4133*8=4*/
  U0BR0 = 0x41;
  U0BR1 = 0x03;
  U0MCTL =0x00;     //00000000
  U0CTL |= CHAR;        //长度8位
  U0CTL &= ~SWRST;

}
void send(uchar num)
{

  TXBUF0 = num;
  while((IFG1&UTXIFG0)==0);
}
void receive()
{
  //P2DIR |= 0xff;
  IFG1 &= 0xbf;
  j = RXBUF0;
  P2OUT = j;
}
void main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  initsys();
  P3DIR |= 0x10;
  P2DIR |= 0xff;
  P2OUT = 0x00;
  P3SEL |= 0x30;
  set();
```

```
  ME1 |= UTXE0 + URXE0;
  IE1 |= URXIE0;

  _EINT();
 /*
 for(k=0;k<255;k++)
  {
    send(k);
    //delay();
  }*/
  for(k=0;k<15;k++)
  {
    send(data[k]);
  }
  while(1);
}
#pragma vector=UART0TX_VECTOR
__interrupt void usart_tx(void)
{
  while((IFG1&URXIFG0)==1);
  delay();
}
#pragma vector=UART0RX_VECTOR
__interrupt void usart0_rx (void)
{
 // while((IFG1&URXIFG0)==0);    //UTXIFG0默认1    R默认0
  if((U0RCTL & RXERR)==0)
   {
     receive();
     delay();
   }
  else
  {
    U0RCTL &= ~(FE + OE + BRK);
  }
  //while((IFG1&URXIFG0)==0);
}
/*****************************************************************************/
uart\uart2
/*****************************************************************************/
/*****************************************************
程序功能：接收来自PC机的字符，然后重新发送给PC机
-----------------------------------------------------
通信格式：N.8.1,  9600
```

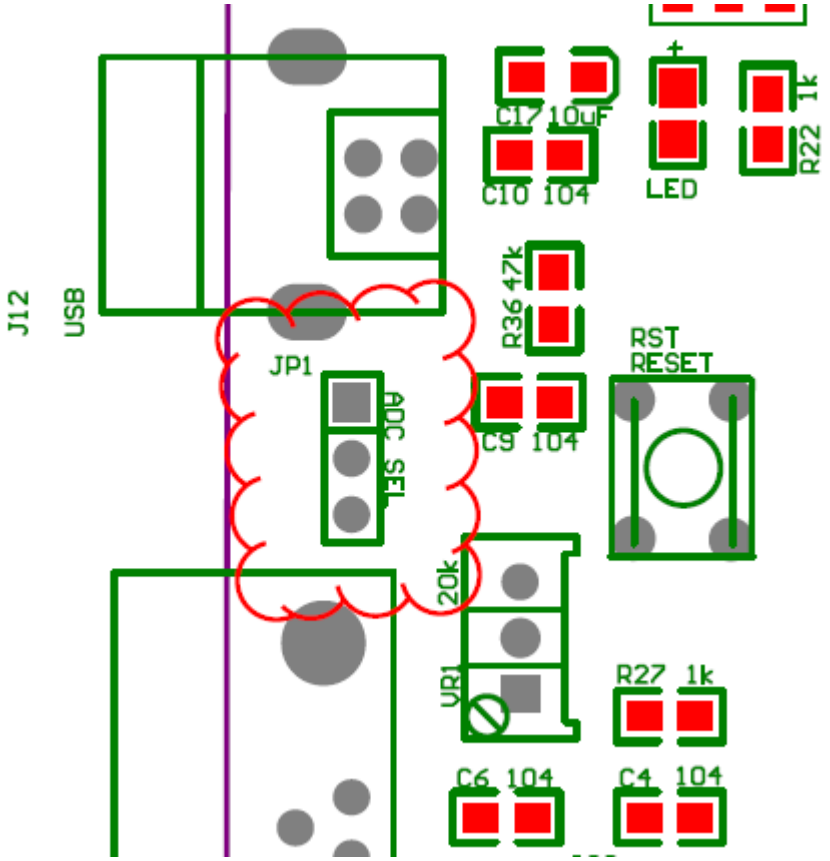
```
            无校验，8个数据位，1个停止位，波特率9600
-------------------------------------------------
测试说明：打开串口调试助手，正确设置通信格式，向学习板
            发送一个字符，观察是否收到回发的字符，以及是否
            正确
*************************************************/
#include  <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int  uint;

/*******************主函数*******************/
void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;                 // 关闭看门狗
    P3SEL |= 0x30;                            // 选择P3.4和P3.5做UART通信端口
    ME1 |= UTXE0 + URXE0;                     // 使能USART0的发送和接受
    UCTL0 |= CHAR;                            // 选择8位字符
    UTCTL0 |= SSEL0;                          // UCLK = ACLK
    UBR00 = 0x03;                             // 波特率9600
    UBR10 = 0x00;                             //
    UMCTL0 = 0x4A;                            // Modulation
    UCTL0 &= ~SWRST;                          // 初始化UART状态机
    IE1 |= URXIE0;                            // 使能USART0的接收中断
    while(1)
    {
        _EINT();                               //打开全局中断
        LPM1;                                  //进入LPM1模式
        while (!(IFG1 & UTXIFG0));             //等待以前的字符发送完毕
        TXBUF0 = RXBUF0;                       //将收到的字符发送出去
    }
}
/*****************************************
函数名称：UART0_RXISR
功    能：UART0的接收中断服务函数，在这里唤醒
          CPU，使它退出低功耗模式
参    数：无
返回值  ：无
*****************************************/
#pragma vector = UART0RX_VECTOR
__interrupt void UART0_RXISR(void)
{
  LPM1_EXIT;                  //退出低功耗模式
}
/*************************************************************************/
```

实验八：AD 实验

实验电路：



在做 AD 实验时，请将跳帽插到 AD_SEL 上端



ad_smg：实验结果为将 ADC 对 P6.0 端口电压的转换结果按转换数据和对应的模拟电压的形式通过数码管显示

ad_uart：实验结果为将 ADC 对 P6.0 端口电压的转换结果按转换数据和对应的模拟电压的形式通过串口发送到 PC 机屏幕上显示

ad\ad_smg

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

```
#include<msp430x14x.h>
#define uchar unsigned char
#define uint unsigned int
 unsigned char table[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
 //unsigned char table1[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
uint result;
uint i;
 uchar nn;
void delay()
{
  uint time;
  uint time1;
  for(time=0x0fff;time>0;time--)
  {
    for(time1=0x00ff;time1>0;time1--)
    {
      ;
    }
  }
}
void show(uint number )
{
  uint data;
  uint data1;
  uchar num1;
  uchar num2;
  uint data2;
  uchar num3;
  uint buffer;
  data=number*3/4096;
  buffer=number*3%4096;
  data1=buffer*10/4096;
  buffer=buffer*10%4096;
  data2=buffer*10/4096;
  num3=(uchar)data2;
  num1=(uchar)data;
  num2=(uchar)data1;
  switch(num1)
  {
  case 0x00:num1=table[0]+0x80;break;
  case 0x01:num1=table[1]+0x80;break;
  case 0x02:num1=table[2]+0x80;break;
  case 0x03:num1=table[3]+0x80;break;
  case 0x04:num1=table[4]+0x80;break;
```

```
 case 0x05:num1=table[5]+0x80;break;
 case 0x06:num1=table[6]+0x80;break;
 case 0x07:num1=table[7]+0x80;break;
 case 0x08:num1=table[8]+0x80;break;
 case 0x09:num1=table[9]+0x80;break;
 }
//num1 &= 0xfe;
 switch(num2)
 {
 case 0x00:num2=table[0];break;
 case 0x01:num2=table[1];break;
 case 0x02:num2=table[2];break;
 case 0x03:num2=table[3];break;
 case 0x04:num2=table[4];break;
 case 0x05:num2=table[5];break;
 case 0x06:num2=table[6];break;
 case 0x07:num2=table[7];break;
 case 0x08:num2=table[8];break;
 case 0x09:num2=table[9];break;
 }
  switch(num3)
 {
 case 0x00:num3=table[0];break;
 case 0x01:num3=table[1];break;
 case 0x02:num3=table[2];break;
 case 0x03:num3=table[3];break;
 case 0x04:num3=table[4];break;
 case 0x05:num3=table[5];break;
 case 0x06:num3=table[6];break;
 case 0x07:num3=table[7];break;
 case 0x08:num3=table[8];break;
 case 0x09:num3=table[9];break;
 }
  switch(nn)

    {
    case 0x00:P4OUT=~num1;P6OUT = ~BIT5;  break;
    case 0x01:P4OUT=~num2;P6OUT = ~BIT4;  break;
    case 0x02:P4OUT=~num3;P6OUT = ~BIT3;  break;
    case 0x03:P4OUT=0xc1;P6OUT = ~BIT2;  break;
    }
    nn++;
    if(nn>3)
    {
```

```
      nn=0;
    }

}
void initsys(void)
{
  BCSCTL1 &= ~XT2OFF; //开启晶振
  uchar j;
  do
  {
    IFG1 &= ~OFIFG;    //消除标志位
    for(j=0xee;j>0;j--)
    {
      ;
    }
  }
  while((IFG1&OFIFG)!=0);
  BCSCTL2 |= SELM_2 + SELS;  //选择时钟
}

void danduo(void)           // 单通道多次
{
  P6SEL |= 0x01;                    //通道
  ADC12CTL0 |= SHT0_8 + MSC + ADC12ON;       // 多次
  ADC12CTL1 |= SHP + CONSEQ_2;     // 单通道多次
  ADC12MCTL0 |= INCH_0;           // A0
  ADC12IE |= 0x01;                  // 中断A0
  ADC12CTL0 |= ENC;               //设定完毕
  _EINT();                        //开启总中断

  ADC12CTL0 |= ADC12SC;
}

void main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  uint dd;
  initsys();
  danduo();
  P4DIR |= 0xff;
  P6DIR |= 0xfc;

  while(1)
```

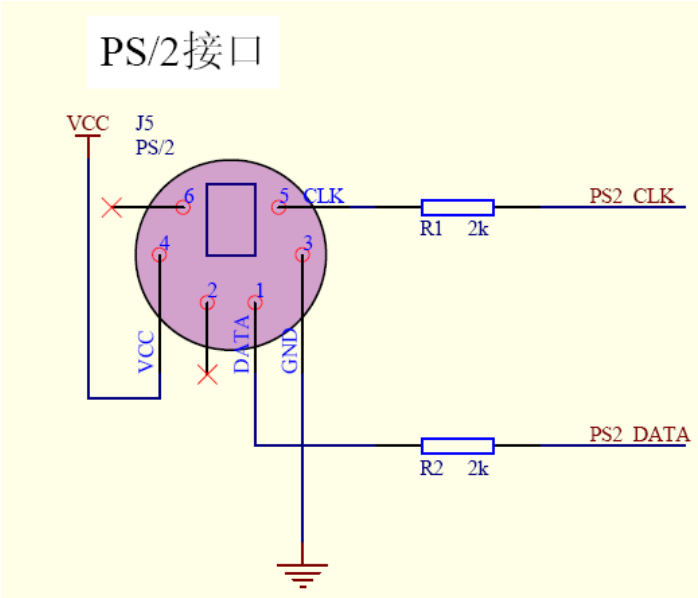
```
  {
    dd=8000;
    while(dd--);
    show(result);
  }



}
#pragma vector=ADC_VECTOR
__interrupt void ADC12(void)
{
  //单通道单次
  //result[0] = ADC12MEM0;
  //result = ADC12MEM0;

  /*//序列通道单次
  result[0]=ADC12MEM0;
  result[1]=ADC12MEM1;
  result[2]=ADC12MEM2;
  result[3]=ADC12MEM3;*/
  //单通道多次

    result = ADC12MEM0;
   //show(result[0]);

  //序列通道多次

  while((ADC12IFG & BIT0)!=0);

}
/*****************************************************************************/
ad\ad_uart
/*****************************************************************************/
/*********************************************************
程序功能：将ADC对P6.0端口电压的转换结果按转换数据和对应的
          模拟电压的形式通过串口发送到PC机屏幕上显示
---------------------------------------------------------
通信格式：N.8.1, 9600
---------------------------------------------------------
测试说明：打开串口调试精灵，正确设置通信格式，观察接收数据
*********************************************************/
#include <msp430.h>
#include "adc_fun.c"
```

```
#include "uart_fun.c"

#define   Num_of_Results   32
uint results[Num_of_Results];    //保存ADC转换结果的数组
uint average;
uchar tcnt = 0;

/**********************主函数**********************/
void main( void )
{
    uchar i;
    uchar buffer[5];

    WDTCTL = WDTPW + WDTHOLD;           //关狗

    InitUART();
    Init_ADC();
    _EINT();

    buffer[4] = '\0';
    while(1)
    {
        LPM1;
        Hex2Dec(average,buffer);
        for(i = 0; i < 4; i++)
            buffer[i] += 0x30;
        PutString0("The digital value is: ");
        PutString(buffer);


        Trans_val(average,buffer);
        buffer[3] = buffer[2];
        buffer[2] = buffer[1];
        buffer[1] = 0x2e - 0x30;
        for(i = 0; i < 4; i++)
            buffer[i] += 0x30;
        PutString0("The analog value is: ");
        PutString(buffer);
    }

}

/*****************************************
函数名称：ADC12ISR
```

```
功    能：ADC中断服务函数，在这里用多次平均的
          计算P6.0口的模拟电压数值
参    数：无
返回值  ：无
*****************************************/
#pragma vector=ADC_VECTOR
__interrupt void ADC12ISR (void)
{
    static uchar index = 0;

    results[index++] = ADC12MEM0;               // Move results
    if(index == Num_of_Results)
    {
        uchar i;

        average = 0;
        for(i = 0; i < Num_of_Results; i++)
        {
            average += results[i];
        }
        average >>= 5;                            //除以32

        index = 0;
        tcnt++;
        if(tcnt == 250)      //主要是降低串口发送速度
        {
            LPM1_EXIT;
            tcnt = 0;
        }
    }
}
/*******************************************************************************/
```

实验九：PS2 键盘实验

实验电路：



ps2_led：实验结果为 ps2 键盘键值按下，将键值显示在 LED 灯上

ps2_smg：实验结果为 ps2 键盘键值（小键盘 0-9）按下，将键值显示在数码管上

ps2\ps2_led

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

#include <MSP430X14X.h>
/*-------------------------------------*/
  #define   uchar   unsigned char
  #define   uint    unsigned int
  #define   ulong   unsigned long
  #define   ps2_clk   BIT4
  #define   ps2_dat   BIT5
 /*P1.4---ps2_clk*/
 /*P1.5---ps2_dat*/
 /*-------------------------------------*/
  uchar key_num;
  void set_PS2_int();
  uchar receive_ps2() ;
  void search_key();
  void set_sysclk();
  /*-------------------------------------*/
void main( void )
{
  set_sysclk();
```

```
  WDTCTL = WDTPW + WDTHOLD;
  set_PS2_int();
  P2DIR=0xff;
  P2OUT=0x00;
judge: while(P1IN&ps2_clk);   //判断clk线是否给拉低
   search_key();
   goto judge;
}
/*-------设置系统时钟-------------*/
void set_sysclk()
{ uchar i;
  //DCOCTL=0XE0;
  BCSCTL1=0X17;   //asck 2分频为wdt提供时钟
  for(i=10;i>0;i--);
}
/*-------------ps2中断设置-----------------------*/
void set_PS2_int()
{P1SEL &=~(BIT4+BIT5);//选择I/O功能
 P1DIR |=(BIT4+BIT5);
 P1OUT |=(BIT4+BIT5);//置1
 P1DIR &=~(BIT4+BIT5);//设为输入
}
 /*-----------ps2读数据子程序----------------------*/
uchar receive_ps2()
 {uchar i;uchar key8=0;
      while(P1IN&ps2_clk);//等待clk变低      |越过
      while(!(P1IN&ps2_clk));//等待clk变高   |起始位
       for(i=0;i<8;i++)
          {key8>>=1;
           while(P1IN&ps2_clk);//等待clk变低
           if(P1IN&ps2_dat)key8=key8|0x80;
           while(!(P1IN&ps2_clk));//等待clk变高
          }
       for(i=0;i<2;i++)
          { while(P1IN&ps2_clk);//等待clk变低     |
            while(!(P1IN&ps2_clk));//等待clk变高 |
          }
      return(key8);
 }
 /*----------ps2中断处理子程序-------------------*/
 void search_key()
 {
  if(P1IN&ps2_dat)goto back;   //判断DAT数据线是否给拉低(起始位)
  key_num=receive_ps2();
```

```
  P2OUT=key_num;
 back:;
 }
/*********************************************************************/
/* switch ( temp2 )
     {
      case 0x45: k="0";  break; //0
      case 0x16: k="1";  break; //1
      case 0x1e: k="2";  break; //2
      case 0x26: k="3";  break; //3
      case 0x25: k="4";  break; //4
      case 0x2e: k="5";  break; //5
      case 0x36: k="6";  break; //6
      case 0x3d: k="7";  break; //7
      case 0x3e: k="8";  break; //8
      case 0x46: k="9";  break; //9
      case 0x1c: k="10"; break; //a
      case 0x32: k="11"; break; //b
      case 0x21: k="12"; break; //c
      case 0x23: k="13"; break; //d
      case 0x24: k="14"; break; //e
      case 0x2b: k="15"; break; //f
      case 0x34: k="16"; break; //g
      case 0x33: k="17"; break; //h
      case 0x43: k="18"; break; //i
      case 0x3b: k="19"; break; //j
      case 0x42: k="20"; break; //k
      case 0x4b: k="21"; break; //l
      case 0x3a: k="22"; break; //m
      case 0x31: k="23"; break; //n
      case 0x44: k="24"; break; //o
      case 0x4d: k="25"; break; //p
      case 0x15: k="26"; break; //q
      case 0x2d: k="27"; break; //r
      case 0x1b: k="28"; break; //s
      case 0x2c: k="29"; break; //t
      case 0x3c: k="30"; break; //u
      case 0x2a: k="31"; break; //v
      case 0x1d: k="32"; break; //w
      case 0x22: k="33"; break; //x
      case 0x35: k="34"; break; //y
      case 0x1a: k="35"; break; //z
      case 0x0e: k="36"; break; //~
      case 0x4e: k="37"; break; //-
```

```
      case 0x55: k="38"; break; //=
      case 0x29: k="46"; break; //SPACE
      case 0x66: k="47";break; //del
      case 0x5b: k="39";break; //]
      case 0x4c: k="40";break;//;
      case 0x52: k="41";break;// '
      case 0x41: k="42"; break; //,
      case 0x49: k="43";break;//.
      case 0x4a: k="44";break; // /
            case 0x54: k="45"; break; //[
   case 0x58: k="49"; break; //cap
   case 0x5a: k="48"; break; //回车键
   case 0x76 : k="0" ; break; //小键盘部分
   case 0x69 : k="1" ; break;
   case 0x72 : k="2" ; break;
   case 0x7a : k="3" ; break;
   case 0x6b : k="4" ; break;
   case 0x73 : k="5" ; break;
   case 0x74 : k="6" ; break;
   case 0x6c : k="7" ; break;
   case 0x75 : k="8" ; break;
   case 0x7d : k="9" ; break;
   case 0x71 : k="43" ; break;;

      default : k="255";
    }
*/
/*******************************************************/
/*****************************************************************************************/
ps2\ps2_smg
/*****************************************************************************************/
/*****************************************************
程序功能：接收并解码来自标准键盘的基本按键的扫描码
然后将0~9、a~f和A~F字符在数码管上显示

注：所谓基本按键就是当此键被按下时对应产生三个字节
的扫描码的按键，详见光盘中的《第二套扫描码》
-----------------------------------------------------
跳线设置：将跳线座J4的1脚和2脚（右侧的两个）短接
-----------------------------------------------------
测试说明：敲定标准键盘上的按键，观察数码管显示
*****************************************************/
#include  <msp430x14x.h>
#include "Keyboard.c"
```

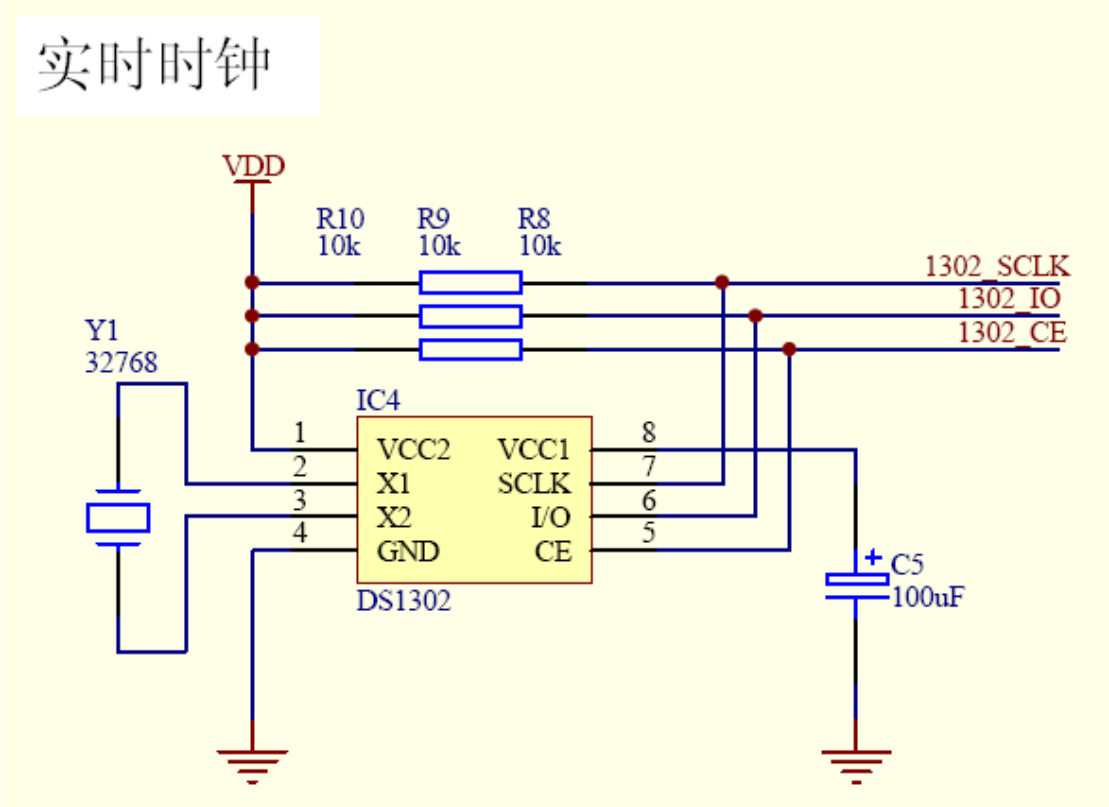
```
#define SIDval  P1IN & BIT5

//数码管7位段码：0--f,和不显示字符(0x00)
uchar scandata[17] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                      0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e,0xff};
//数码管的显示缓存
uchar DispBuf = 16;

/***************主函数***************/
void main(void)
{
    uchar disptmp,i;

    WDTCTL = WDTPW + WDTHOLD;          //关闭看门狗

    /*------选择系统主时钟为8MHz-------*/
    BCSCTL1 &= ~XT2OFF;                // 打开XT2高频晶体振荡器
    do
    {
        IFG1 &= ~OFIFG;                // 清除晶振失败标志
        for (i = 0xFF; i > 0; i--);    // 等待8MHz晶体起振
    }
    while ((IFG1 & OFIFG));            // 晶振失效标志仍然存在?
    BCSCTL2 |= SELM_2;                 // 主时钟选择高频晶振

    P4DIR = 0xff;                  //设置P4，P5的IO方向为输出
    P6DIR = 0xfc;

    P4OUT = 0x00;                  //设置P4，P5的输出初值
    P6OUT = 0x00;


    Init_KB();                     //初始化键盘端口
    _EINT();                       //打开全局中断

    while(1)
    {
        LPM1;                      //进入低功耗模式
        disptmp = GetChar();       //读取键值对应的ASCII码

        if((disptmp > 0x2f)&&(disptmp < 0x3a)) //如果接收到的字符是0~9
        {
            DispBuf = disptmp - 0x30; //得到实际的数字
        }
```

```
        else if((disptmp > 0x40)&&(disptmp < 0x47))//如果接收到的字符是A~F
        {
            DispBuf = disptmp - 0x37; //得到实际的字母
        }
        else if((disptmp > 0x60)&&(disptmp < 0x67))//如果接收到的字符是a~f
        {
            DispBuf = disptmp - 0x57; //得到实际的字母
        }
        else
        {
            DispBuf = 16;             //控制数码管不显示
        }
        P4OUT = scandata[DispBuf];
    }
}
/*****************************************
函数名称：PORT1_ISR
功    能：P1端口的中断服务函数，在这里接收来
          自键盘的字符
参    数：无
返回值  ：无
*****************************************/
#pragma vector=PORT1_VECTOR
__interrupt void  PORT1_ISR(void)
{
    if(P1IFG & BIT4)            //如果是clock的中断
    {
        P1IFG &=~ BIT4;         //清除中断标志

        if(bitcount == 11)        //接收第1位
        {
            if(SIDval)          //如果不是起始位
                return;
            else
                bitcount--;
        }
        else if(bitcount == 2)    //接收奇偶校验位
        {
            if(SIDval)          //如果校验位等于1
                pebit = 1;
            else
                pebit = 0;
            bitcount--;
        }
```

```
        else if(bitcount == 1)    //接收停止位
        {
            if(SIDval)          //若停止位正确
            {
                bitcount = 11;    //复位位计数变量
                if( Decode(recdata) )    //解码获得此键值的ASCII值并保存
                    LPM3_EXIT;           //退出低功耗模式
                recdata = 0;          //清除接收数据
            }
            else                //如果出错
            {
                bitcount = 11;
                recdata = 0;
            }
        }
        else                    //接收8个数据位
        {
            recdata >>= 1;
            if(SIDval)  recdata |= 0x80;
            bitcount--;
        }
    }
}
/*******************************************************************************/
```

实验十：DS1302（SPI 协议）实验

实验电路：



DS1302+SMG：实验结果为将实时时钟显示在数码管上。

DS1302+LCD：实验结果为将实时时钟显示在液晶上。

DS1302+SMG：

```
/*****************************************************************************/
#include <msp430x14x.h>
#include "ds1302.h"

#define dpydat P4OUT
#define dpy0 BIT2;
#define dpy1 BIT3;
#define dpy2 BIT4;
#define dpy3 BIT5;
#define dpy4 BIT6;
#define dpy5 BIT7;

unsigned char table[]={0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f};
int min10,min1,hour10,hour1,sec10,sec1,secflash,dpybit=0x00;

void gettime()
```

```
{
  unsigned char hour,min,sec;
  hour=rtc_gethour();
  min=rtc_getmin();
  sec=rtc_getsec();
  hour1=hour%10;
  hour10=hour/10;
  min1=min%10;
  min10=min/10;
  sec1=sec%10;
  sec10=sec/10;
  if (sec%2)
    secflash=1;
  else
    secflash=0;
}

void rtcinit ()
{
  rtc_wp(0);
  rtc_stop(0);
  rtc_charger(1,1);
}

void sysinit ()
{
  P4OUT = 0xff;
  P4DIR = 0xff;
  P6OUT = 0xfc;
  P6DIR = 0xfc;
  IE1 |= WDTIE;                         //开WDT中断
}

void main ()
{
  WDTCTL = WDT_ADLY_1_9;                //WDT工作于定时模式,定时时间为(2^6)/(2^15)s
  sysinit ();
  rtcinit ();
  _EINT();
  while (1)
  {
 gettime();
  }
```

```
}
#pragma vector=WDT_VECTOR
__interrupt void wdtisr ()
{
  switch (dpybit)
  {
  case 0: P6OUT |= 0xfc;
          dpydat = ~table[hour10];
          P6OUT &= ~dpy5;
          break;
  case 1: P6OUT |= 0xfc;
            dpydat = ~(table[hour1]|0x80);
          P6OUT &= ~dpy4;
          break;
  case 2: P6OUT |= 0xfc;
          dpydat = ~table[min10];
          P6OUT &= ~dpy3
          break;
  case 3: P6OUT |= 0xfc;
   //       if (secflash)
            dpydat = ~(table[min1] | 0x80);
   //       else
  //          dpydat = ~table[min1];
          P6OUT &= ~dpy2;
          break;
  case 4: P6OUT |= 0xfc;
          dpydat = ~table[sec10];
          P6OUT &= ~dpy1;
          break;
  case 5: P6OUT |= 0xfc;
          dpydat = ~table[sec1];
          P6OUT &= ~dpy0;
          break;
  }
  dpybit++;
    dpybit=dpybit%6;
}
/*****************************************************************************/
```

DS1302+LCD：

```
/*****************************************************************************/
/**********************main.c*****************************
```

程序功能：从DS1302中读出时间数据在1602液晶模块上显示

测试说明：用户可以更改"gdata.h"中wdata，bwdata，rwdata

```
            三个数组中的数据，但是请注意数据格式。
            根据程序中提示，设置断点观察数据。
*************************************************************/
#include <msp430x14x.h>

#include "ds1302.h"
#include "cry1602.h"
#include "gdata.h"

typedef unsigned char uchar;
typedef unsigned int  uint;

void main(void)
{
    uchar disptemp[8];

    WDTCTL = WDTPW + WDTHOLD;   //关闭看门狗

    disptemp[2]=0x3a;   // ":"对应的编码
    disptemp[5]=0x3a;
    Reset_DS1302();



    /*****测试更改和读出时间*****/
    Set_DS1302(wdata);
    Get_DS1302(rdata);

    _NOP();          //在此处设置断点，观察rdata是否与wdata一致

    /*****测试连续读写时间寄存器*****/
    BurstWrite1302(bwdata);
    BurstRead1302(brdata);

    _NOP();          //在此处设置断点，观察brdata是否与bwdata一致

    /*****测试连续读写RAM*****/
    BurstWriteRAM(rwdata);
    BurstReadRAM(rrdata);

    _NOP();          //在此处设置断点，观察rrdata是否与rwdata一致

    /***********************************************
    注释：用户在利用27~43行的程序完成测试以后，请设置好
```

```
    正确的当前时间然后将这端程序屏蔽掉，重新make以后写入
    CPU中，这样才能保证每次上电时CPU都从DS1302中读出正确
    的当前时间送到液晶中去显示，而不会发生每次都重新改写
    DS1302内部时间的问题。
    ************************************************/
    //初始化液晶
    LcdReset();
    DispNchar(1,0,14,tishi);
    //读取时间转换数值显示
    while(1)
    {
        BurstRead1302(rdata);
        disptemp[6] = shuzi[(rdata[0]&0xf0)>>4];
        disptemp[7] = shuzi[rdata[0]&0x0f];
        disptemp[3] = shuzi[(rdata[1]&0xf0)>>4];
        disptemp[4] = shuzi[rdata[1]&0x0f];
        disptemp[0] = shuzi[(rdata[2]&0xf0)>>4];
        disptemp[1] = shuzi[rdata[2]&0x0f];
        DispNchar(4,1,8,disptemp);
        delay(50000);
    }
}
/**************************cry1602.c****************************/
#include <msp430x14x.h>
#include "cry1602.h"
typedef unsigned char uchar;
typedef unsigned int  uint;

/**************宏定义***************/
#define DataDir     P4DIR
#define DataPort    P4OUT
#define Busy        0x80
#define CtrlDir     P6DIR
#define CLR_RS P6OUT &= ~BIT2;    //RS = P6.2
#define SET_RS P6OUT |= BIT2;
#define CLR_RW P6OUT &= ~BIT3;    //RW = P6.3
#define SET_RW P6OUT |= BIT3;
#define CLR_EN P6OUT &= ~BIT4;    //EN = P6.4
#define SET_EN P6OUT |= BIT4;

void delay1();                    //申明延时函数
/*****************************************
函数名称：DispNchar
功    能：让液晶从某个位置起连续显示N个字符
```

```
参    数：x--位置的列坐标
          y--位置的行坐标
          n--字符个数
          ptr--指向字符存放位置的指针
返回值  ：无
*******************************************/
void DispNchar(uchar x,uchar y, uchar n,uchar *ptr)
{
    uchar i;

    for (i = 0;i < n;i++)
    {
        Disp1Char(x++,y,ptr[i]);
        if (x == 0x0f)
        {
           x = 0;
           y ^= 1;
        }
    }
}
/*******************************************
函数名称：LocateXY
功    能：向液晶输入显示字符位置的坐标信息
参    数：x--位置的列坐标
          y--位置的行坐标
返回值  ：无
*******************************************/
void LocateXY(uchar x,uchar y)
{
    uchar temp;

    temp = x&0x0f;
    y &= 0x01;
    if(y)   temp |= 0x40;  //如果在第2行
    temp |= 0x80;

    LcdWriteCommand(temp,1);
}
/*******************************************
函数名称：Disp1Char
功    能：在某个位置显示一个字符
参    数：x--位置的列坐标
          y--位置的行坐标
          data--显示的字符数据
```

```
返回值  ：无
*******************************************/
void Disp1Char(uchar x,uchar y,uchar data)
{
    LocateXY(x, y);
    LcdWriteData( data );
}
/*******************************************
函数名称：LcdReset
功    能：对1602液晶模块进行复位操作
参    数：无
返回值  ：无
*******************************************/
void LcdReset(void)
{
    CtrlDir |= 0x1C;                  //控制线端口设为输出状态 0001,1100
    DataDir  = 0xFF;                  //数据端口设为输出状态

    LcdWriteCommand(0x38, 0);        //规定的复位操作
    Delay5ms();
    LcdWriteCommand(0x38, 0);
    Delay5ms();
    LcdWriteCommand(0x38, 0);
    Delay5ms();

    LcdWriteCommand(0x38, 1);        //显示模式设置
    LcdWriteCommand(0x08, 1);        //显示关闭
    LcdWriteCommand(0x01, 1);        //显示清屏
    LcdWriteCommand(0x06, 1);        //写字符时整体不移动
    LcdWriteCommand(0x0c, 1);        //显示开，不开游标，不闪烁
}
/*******************************************
函数名称：LcdWriteCommand
功    能：向液晶模块写入命令
参    数：cmd--命令，
          chk--是否判忙的标志，1：判忙，0：不判
返回值  ：无
*******************************************/
void LcdWriteCommand(uchar cmd,uchar chk)
{

    //if (chk) WaitForEnable();   // 检测忙信号?

    CLR_RS;
```

```
    CLR_RW;
    _NOP();

    DataPort = cmd;             //将命令字写入数据端口
    _NOP();

    SET_EN;                     //产生使能脉冲信号
            delay1();                   //保持使能信号为低一段时间
    _NOP();
    _NOP();
    CLR_EN;
}

/*******************************************
函数名称：LcdWriteData
功    能：向液晶显示的当前地址写入显示数据
参    数：data--显示字符数据
返回值  ：无
*******************************************/
void LcdWriteData( uchar data )
{
    //WaitForEnable();        //等待液晶不忙

    SET_RS;
    CLR_RW;
    _NOP();

    DataPort = data;        //将显示数据写入数据端口
    _NOP();

    SET_EN;                 //产生使能脉冲信号
            delay1();                   //保持使能信号为低一段时间
    _NOP();
    _NOP();
    CLR_EN;
}
/*******************************************
函数名称：WaitForEnable
功    能：等待1602液晶完成内部操作
参    数：无
返回值  ：无
*******************************************/
void WaitForEnable(void)
{
```

```
    P4DIR &= 0x00;  //将P4口切换为输入状态

    CLR_RS;
    SET_RW;
    _NOP();
    SET_EN;
    _NOP();
    _NOP();

    while((P4IN & Busy) != 0);  //检测忙标志

    CLR_EN;

    P4DIR |= 0xFF;  //将P4口切换为输出状态
}

/*********************************************
函数名称：Delay5ms
功    能：延时约5ms
参    数：无
返回值  ：无
*********************************************/
void Delay5ms(void)
{
    uint i = 40000;
    while (i != 0)
    {
        i--;
    }
}
/*********************************************
函数名称：Delay400ms
功    能：延时约400ms
参    数：无
返回值  ：无
*********************************************/
void Delay400ms(void)
{
    uchar i = 50;
    uint j;

    while(i--)
    {
        j = 7269;
```

```
        while(j--);
    }
}
//-----------------------------------
//延时函数
void delay1()
 {
   unsigned int i;
   for(i=0;i<300;i++);
 }
/*************************ds1302.c*****************************/
#include <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int  uint;

/***************宏定义***************/
#define DS_RST  BIT2        //DS_RST = P3.2
#define DS_SCL  BIT6        //DS_SCL = P3.6
#define DS_SDA  BIT7        //DS_SDA = P3.7
#define DS_RST_IN P3DIR &= ~DS_RST
#define DS_RST_OUT P3DIR |= DS_RST
#define DS_RST0 P3OUT &= ~DS_RST
#define DS_RST1 P3OUT |= DS_RST
#define DS_SCL_IN P3DIR &= ~DS_SCL
#define DS_SCL_OUT P3DIR |= DS_SCL
#define DS_SCL0 P3OUT &= ~DS_SCL
#define DS_SCL1 P3OUT |= DS_SCL
#define DS_SDA_IN P3DIR &= ~DS_SDA
#define DS_SDA_OUT P3DIR |= DS_SDA
#define DS_SDA0 P3OUT &= ~DS_SDA
#define DS_SDA1 P3OUT |= DS_SDA
#define DS_SDA_BIT P3IN & DS_SDA

/*******************************************
函数名称：delay
功    能：延时一段时间
参    数：time--延时长度
返回值  ：无
*******************************************/
void delay(uint time)
{
    uint i;
    for(i = 0;i < time;i++)     _NOP();
}
```

```
/*********************************************
函数名称：Reset_DS1302
功    能：对DS1302进行复位操作
参    数：无
返回值  ：无
*********************************************/
void Reset_DS1302(void)
{
    DS_RST_OUT;  //RST对应的IO设置为输出状态
    DS_SCL_OUT;  //SCLK对应的IO设置为输出状态
    DS_SCL0;     //SCLK=0
    DS_RST0;     //RST=0
    delay(10);
    DS_SCL1;    //SCLK=1
}
/*********************************************
函数名称：Write1Byte
功    能：对DS1302写入1个字节的数据
参    数：wdata--写入的数据
返回值  ：无
*********************************************/
void Write1Byte(uchar wdata)
{
    uchar i;

    DS_SDA_OUT;     //SDA对应的IO设置为输出状态
    DS_RST1;        //REST=1;

    for(i = 8; i > 0; i--)
    {
        if(wdata&0x01)  DS_SDA1;
        else            DS_SDA0;
        DS_SCL0;
        delay(10);
        DS_SCL1;
        delay(10);
        wdata >>= 1;
    }
}
/*********************************************
函数名称：Read1Byte
功    能：从DS1302读出1个字节的数据
参    数：无
返回值  ：读出的一个字节数据
```

```
*********************************************/
uchar Read1Byte(void)
{
    uchar i;
    uchar rdata = 0X00;

    DS_SDA_IN;  //SDA对应的IO设置为输入状态
    DS_RST1;    //REST=1;

    for(i = 8; i > 0; i--)
    {
        DS_SCL1;
        delay(10);
        DS_SCL0;
        delay(10);
        rdata >>= 1;
        if(DS_SDA_BIT)  rdata |= 0x80;
    }

    return(rdata);
}
/*********************************************
函数名称：W_Data
功    能：向某个寄存器写入一个字节数据
参    数：addr--寄存器地址
          wdata--写入的数据
返回值  ：无
*********************************************/
void W_Data(uchar addr, uchar wdata)
{
    DS_RST0;
    DS_SCL0;
    _NOP();
    DS_RST1;
    Write1Byte(addr);   //写入地址
    Write1Byte(wdata);  //写入数据
    DS_SCL1;
    DS_RST0;
}
/*********************************************
函数名称：R_Data
功    能：从某个寄存器读出一个字节数据
参    数：addr--寄存器地址
返回值  ：读出的数据
```

```
*****************************************/
uchar R_Data(uchar addr)
{
    uchar rdata;

    DS_RST0;
    DS_SCL0;
    _NOP();
    DS_RST1;
    Write1Byte(addr);    //写入地址
    rdata = Read1Byte();  //读出数据
    DS_SCL1;
    DS_RST0;

    return(rdata);
}
/*****************************************
函数名称：BurstWrite1302
功    能：以burst方式向DS1302写入批量时间数据
参    数：ptr--指向时间数据存放地址的指针
返回值  ：读出的数据
说    明：时间数据的存放格式是：
          秒，分，时，日，月，星期，年，控制
            【7个数据（BCD格式）+1个控制】
*****************************************/
void BurstWrite1302(uchar *ptr)
{
    uchar i;

    W_Data(0x8e,0x00);      //允许写入
    DS_RST0;
    DS_SCL0;
    _NOP();
    DS_RST1;
    Write1Byte(0xbe);       // 0xbe:时钟多字节写入命令
    for (i = 8; i > 0; i--)
    {
        Write1Byte(*ptr++);
    }
    DS_SCL1;
    DS_RST0;
    W_Data(0x8e,0x80);      // 禁止写入
}
/*****************************************
```

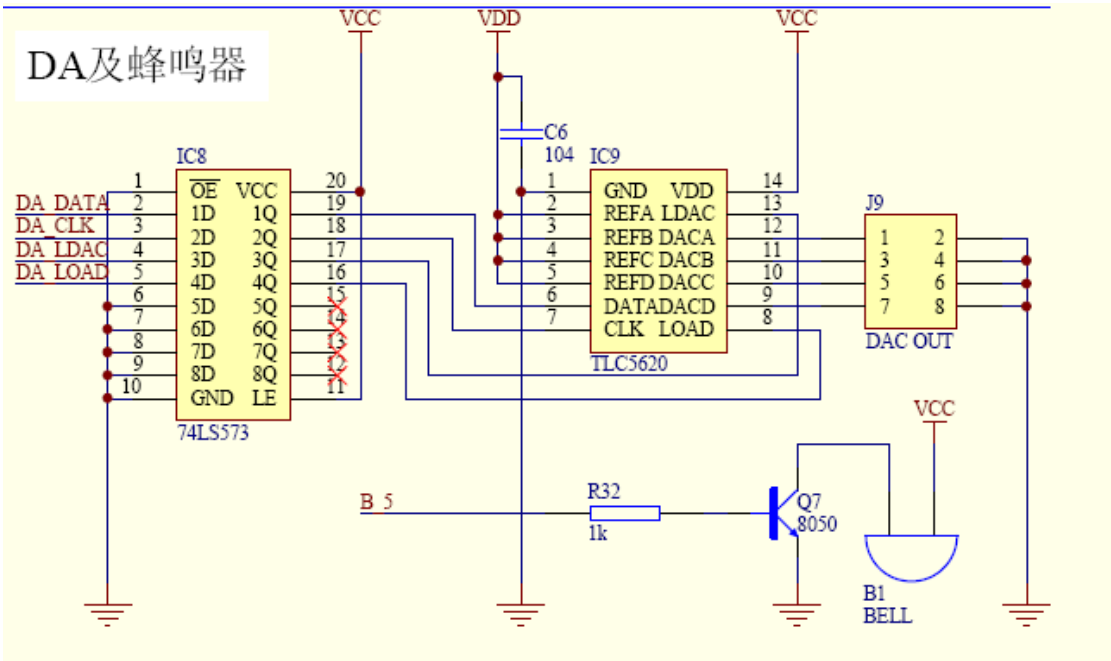
```
函数名称：BurstRead1302
功    能：以burst方式从DS1302读出批量时间数据
参    数：ptr--指向存放时间数据地址的指针
返回值  ：无
说    明：时间数据的存放格式是：
          秒，分，时，日，月，星期，年，控制
            【7个数据（BCD格式）+1个控制】
*********************************************/
void BurstRead1302(uchar *ptr)
{
    uchar i;

    DS_RST0;
    DS_SCL0;
    _NOP();
    DS_RST1;
    Write1Byte(0xbf);              //0xbf:时钟多字节读命令
    for (i = 8; i > 0; i--)
    {
       *ptr++ = Read1Byte();
    }
    DS_SCL1;
    DS_RST0;
}
/*********************************************
函数名称：BurstWriteRAM
功    能：以burst方式向DS1302的RAM中写入批量数据
参    数：ptr--指向存放数据地址的指针
返回值  ：无
说明    ：共写入31个字节的数据
*********************************************/
void BurstWriteRAM(uchar *ptr)
{
    uchar i;

    W_Data(0x8e,0x00);          //允许写入
    DS_RST0;
    DS_SCL0;
    _NOP();
    DS_RST1;
    Write1Byte(0xfe);           //0xfe:RAM多字节写命令
    for (i = 31; i>0; i--)      //RAM共有31个字节
    {
        Write1Byte(*ptr++);
```

```
    }
    DS_SCL1;
    DS_RST0;
    W_Data(0x8e,0x80);            //禁止写入
}
/*********************************************
函数名称：BurstReadRAM
功    能：以burst方式从DS1302的RAM中读出批量数据
参    数：ptr--指向数据存放地址的指针
返回值  ：无
说明    ：共读出31个字节的数据
*********************************************/
void BurstReadRAM(uchar *ptr)
{
    uchar i;

    DS_RST0;
    DS_SCL0;
    _NOP();
    DS_RST1;
    Write1Byte(0xff);              //0xff:RAM的多字节读命令
    for (i = 31; i > 0; i--)
    {
       *ptr++ = Read1Byte();
    }
    DS_SCL1;
    DS_RST0;
}
/*********************************************
函数名称：Set_DS1302
功    能：设置DS1302内部的时间
参    数：ptr--指向存放数据地址的指针
返回值  ：无
说明    ：写入数据的格式：
          秒 分 时 日 月 星期 年  【共7个字节】
*********************************************/
void Set_DS1302(uchar *ptr)
{
    uchar i;
    uchar addr = 0x80;

    W_Data(0x8e,0x00);    //允许写入

    for(i = 7;i > 0;i--)
```

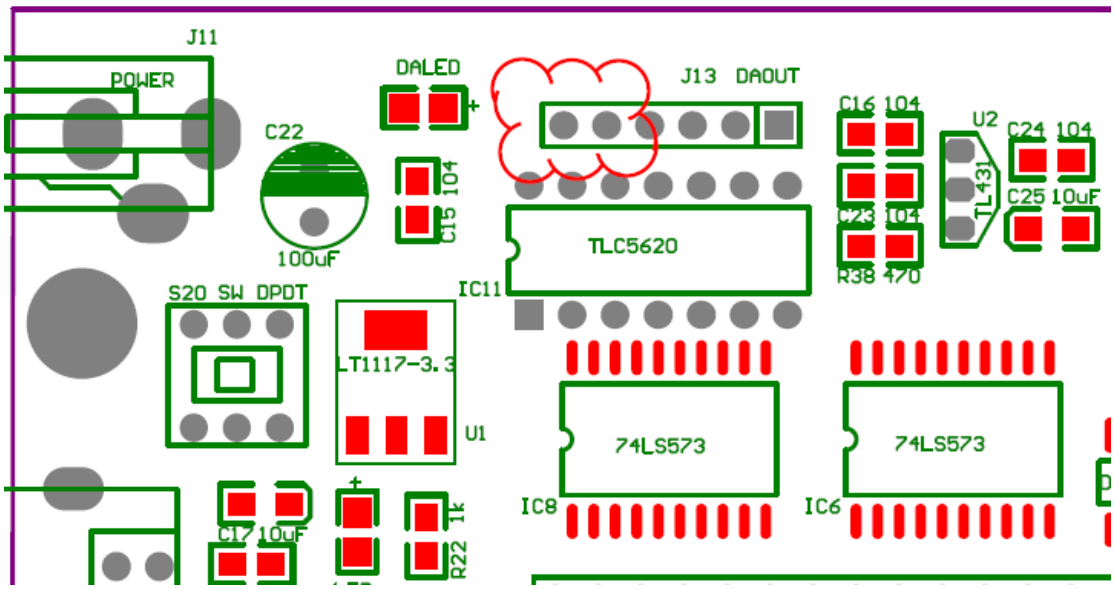
```
    {
        W_Data(addr,*ptr++);
        addr += 2;
    }
    W_Data(0x8e,0x80);      //禁止
}
/*************************************************************************
*
* 名称: Get_DS1302
* 说明:
* 功能: 读取DS1302当前时间
* 调用: R_Data(uchar addr)
* 输入: ucCurtime: 保存当前时间地址。当前时间格式为: 秒 分 时 日 月 星期 年
* 7Byte (BCD码) 1B 1B 1B 1B 1B 1B 1B
* 返回值: 无
*************************************************************************/
/*****************************************
函数名称：Get_DS1302
功    能：读取DS1302内部的时间
参    数：ptr--指向数据存放地址的指针
返回值  ：无
说明    ：读出数据的格式：
          秒 分 时 日 月 星期 年  【共7个字节】
*****************************************/
void Get_DS1302(uchar *ptr)
{
    uchar i;
    uchar addr = 0x81;

    for(i = 0;i < 7;i++)
    {
        ptr[i] = R_Data(addr);
        addr += 2;
    }
}
/*************************************************************************/
```

实验十一：DA（TLC5620）实验

实验电路：



DA 输出从 DAC_OUT 输出，左侧为接地。



dac5620（0 到 255）：

实验结果为 430 以 spi 协议方式控制 TLC5620，使其在 DAC_OUT 处产生锯齿波，通过 LED 灯显示出来。

dac5620 三角波：

实验结果为 430 以 spi 协议方式控制 TLC5620，使其在 DAC_OUT 处产生由暗到亮的变化，通过 LED 灯显示出来。

dac5620（0 到 255）：

```
/*****************************************************************************/
/*************************************************************
*                  SPI控制D A(TLC5620)同步通信                *
*                       SPI三线主模式                         *
*                         硬件连接图                          *
*                  P5.0----STE1----DA LOAD                    *
*                  P5.1----SIMO1----DA DATA                   *
*                  P5.2----SOMI1----DA LDAC                   *
*                  P5.3----UCLK1----DA CLK                    *
*                        采用的是:                            *
Figure 3. Load-Controlled Update Using 8-Bit Serial Word (LDAC = Low)
                    A1=0 A0=0 RNG=0
*************************************************************/

#include <msp430x14x.h>

#define   uint8 unsigned char
#define   uint32 unsigned int
#define   uint16 unsigned short

#define  DAC_RNG    0                                  // 幅值选择

/* TLC5620由P0口控制，控制I/O定义如下 */
#define  CLK1           (1<<3)
#define  DAT1           (1<<1)
#define  LOAD           (1<<0)
#define  LDAC           (1<<2)
//#define  DAC5620CON     (CLK1 | DAT1 | LOAD | LDAC)

/* I/O输出操作函数 */
#define  CCLK1()    P5OUT &= ~CLK1
#define  SCLK1()    P5OUT |= CLK1

#define  CDAT1()    P5OUT &=~DAT1
#define  SDAT1()    P5OUT |= DAT1

#define  CLOAD()    P5OUT &= ~LOAD
#define  SLOAD()    P5OUT |= LOAD

#define  CLDAC()    P5OUT &=~LDAC
#define  SLDAC()    P5OUT |= LDAC

void  DAC5620_Data(uint8 no, uint8 dat);
void  PIN_Init(void);
```

```
void  Delay(void);

int  main(void)
{   uint8  i;

    PIN_Init();                                          // 初始化CPU的IO口

    while(1)                                             // 发生锯齿波
    {
        DAC5620_Data(1, i);
        DAC5620_Data(2, 0);
        DAC5620_Data(3, i);
        DAC5620_Data(4, i);
        i++;
    }

    return(0);
}

/*****************************************************************************
* 名    称：IRQ_Eint3()
* 功    能：外部中断EINT3服务函数，取反B1控制口。
* 入口参数：无
* 出口参数：无
*****************************************************************************/
void  PIN_Init(void)                                     // CPU的IO口初始化函数
{
  WDTCTL = WDTPW + WDTHOLD;                  //关闭关门狗
    //BCSCTL2 &=0xc0;           //XT2CLK+2分频
  P5DIR |=0x0f;

    CCLK1();                                             // CLK1 = 0
    CDAT1();                                             // DAT1 = 0
    SLOAD();                                             // LOAD = 1
    SLDAC();                                             // LDAC = 1
}

/*****************************************************************************
* 名    称：DAC5620_Data()
* 功    能：DAC芯片TLC5620的控制函数
* 入口参数：uint8 no  通道选择  uint8 dat 输出数值
* 出口参数：无
*****************************************************************************/
void  DAC5620_Data(uint8 no, uint8 dat)
```

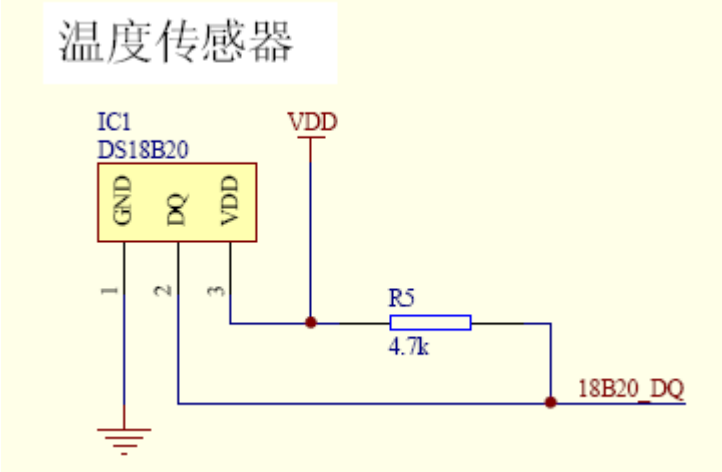
```
{   unsigned int bak;
    uint8  m;

#if DAC_RNG==0
    bak = (dat<<5) | ((no&0x03)<<14);
#else
    bak = (dat<<5) | ((no&0x03)<<14) | (1<<13);
#endif

    for(m=0; m<11; m++)
    {
        SCLK1();        // CLK1 = 1
        Delay();

        if((bak&0x8000) == 0)
        {
            CDAT1();    // DAT1 = 0
        }
        else
        {
            SDAT1();    // DAT1 = 1
        }
        Delay();

        CCLK1();        // CLK1 = 0
        Delay();

        bak <<= 1;
    }
    CLOAD();            // LOAD = 0
    Delay();
    SLOAD();            // LOAD = 1
    Delay();
    CLDAC();            // LDAC = 0
    Delay();
    SLDAC();            // LDAC = 1
    Delay();
}

void  Delay(void)
{   uint32  i;

    for(i=0; i<50; i++);
}
```

```
/*****************************************************************************/
```
dac5620 三角波：
```
/*****************************************************************************/
/*********************************************************
*                  SPI控制D A(TLC5620)同步通信           *
*                       SPI三线主模式                    *
*                         硬件连接图                     *
*                  P5.0----STE1----DA LOAD               *
*                  P5.1----SIMO1----DA DATA              *
*                  P5.2----SOMI1----DA LDAC              *
*                  P5.3----UCLK1----DA CLK               *
*                        采用的是:                       *
Figure 3. Load-Controlled Update Using 8-Bit Serial Word (LDAC = Low)
                     A1=0 A0=0 RNG=0
*********************************************************/

#include <msp430x14x.h>

unsigned char a;
delay()
{ int i;
  for(i=0;i<30000;i++);
}

void main()
{
  WDTCTL = WDTPW + WDTHOLD;                 //关闭关门狗
  P5SEL = 0x0a;                  //P5.1  5.3 用于spi模式
  P5DIR |=0x0f;                            //P5.0~P5.3为输出
  U1CTL = CHAR  + SYNC+MM + SWRST;          // CHAR:字符长度为1
                                            //SYNC:SPI模式
                                            //MM:主机模式
                                            //SWRST:控制位
  U1TCTL = CKPH+ SSEL0 + STC;                     //CKPL+//CKPL:时钟相位控制位
                                            //SSEL1:时钟源选择位
                                            //STC:从机发送控制位     spi三线模式

  U1BR0 = 0x02;                             //波特率选择寄存器0
  U1BR1 = 0x00;                             //波特率选择寄存器1
  U1MCTL = 0x00;
  ME2 |= USPIE1;                            //模块使能
  U1CTL&=~SWRST;                            //开启SPI
  P5OUT |=BIT0+BIT2;                        //P5.0 P5.2 拉高
  P5OUT ^=0x04;                             //P5.2 拉低
```

```
  //给DA发数据     图形:锯齿波
  while(1)
  {
    //以下注释部分为产生锯齿波的程序
    for(a =0x00;a<0x7f;)
    {
      U1TXBUF = a;
      while((IFG2&UTXIFG1)==0);             //USART1发送数据是否准备好
      a=a+1;
      P5OUT ^=0x01;                         //P5.0 拉低
      P5OUT |=BIT0;                         //P5.0 拉高
    }

    //以下程序为发光二极管程序
/*      for(a =0x30;a<0x50;)
    {
      U1TXBUF = a;
      while((IFG2&UTXIFG1)==0);             //USART1发送数据是否准备好
      a=a+1;
      P5OUT ^=0x01;                         //P5.0 拉低
      P5OUT |=BIT0;                         //P5.0 拉高
      delay();

    }
*/
  }
}
/*****************************************************************************/
```

实验十二：DS18b20 温度传感器实验

实验电路：



实验结果为用 DS18B20 测量室温并用 6 位数码管显示。

```
/*********************************************************************************************/
/************************************************
程序功能：用DS18B20测量室温并在数码管上显示。
-------------------------------------------------
测试说明：观察显示温度数值。
************************************************/
#include <msp430x14x.h>
#include "DS18B20.c"

//要显示的6位温度数字
uchar dN[6];
//数码管七段码：0--f
//uchar scandata[16] = {0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,
//                      0x7f,0x6f,0x77,0x7c,0x39,0x5e,0x79,0x71};
//共阳笔段码
uchar scandata[16] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                      0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};

//数码管位选变量
uchar cnt = 2;

void Disp_Numb(uint temper);

/***************主函数***************/
void main(void)
{
```

```
    uchar i;

    WDTCTL=WDTPW+WDTHOLD;

    /*------选择系统主时钟为8MHz-------*/
    BCSCTL1 &= ~XT2OFF;                  //打开XT2高频晶体振荡器
    do
    {
        IFG1 &= ~OFIFG;                  //清除晶振失败标志
        for (i = 0xFF; i > 0; i--);      //等待8MHz晶体起振
    }
    while ((IFG1 & OFIFG));              //晶振失效标志仍然存在?
    BCSCTL2 |= SELM_2 + SELS;            //MCLK和SMCLK选择高频晶振


    //设置看门狗定时器，初始化控制数码管的IO
    WDTCTL = WDT_ADLY_1_9;
    IE1 |= WDTIE;
    P4DIR = 0xff;
    P6DIR = 0xfc;
    P4OUT = 0x00;
    P6OUT = 0xfc;

    //设置DS18B20的IO状态
    P3DIR |= BIT0;
    P3OUT |= BIT0;
    //计数时钟选择SMLK=8MHz，1/8分频后为1MHz
    TACTL |= TASSEL_2 + ID_3;
    //打开全局中断
    _EINT();
    //循环读数显示
    while(1)
    {
       Disp_Numb(Do1Convert());
    }
}
/*****************************************
函数名称：watchdog_timer
功    能：看门狗定时器中断服务函数，进行数码
          管动态扫描
参    数：无
返回值  ：无
*****************************************/
#pragma vector = WDT_VECTOR
```

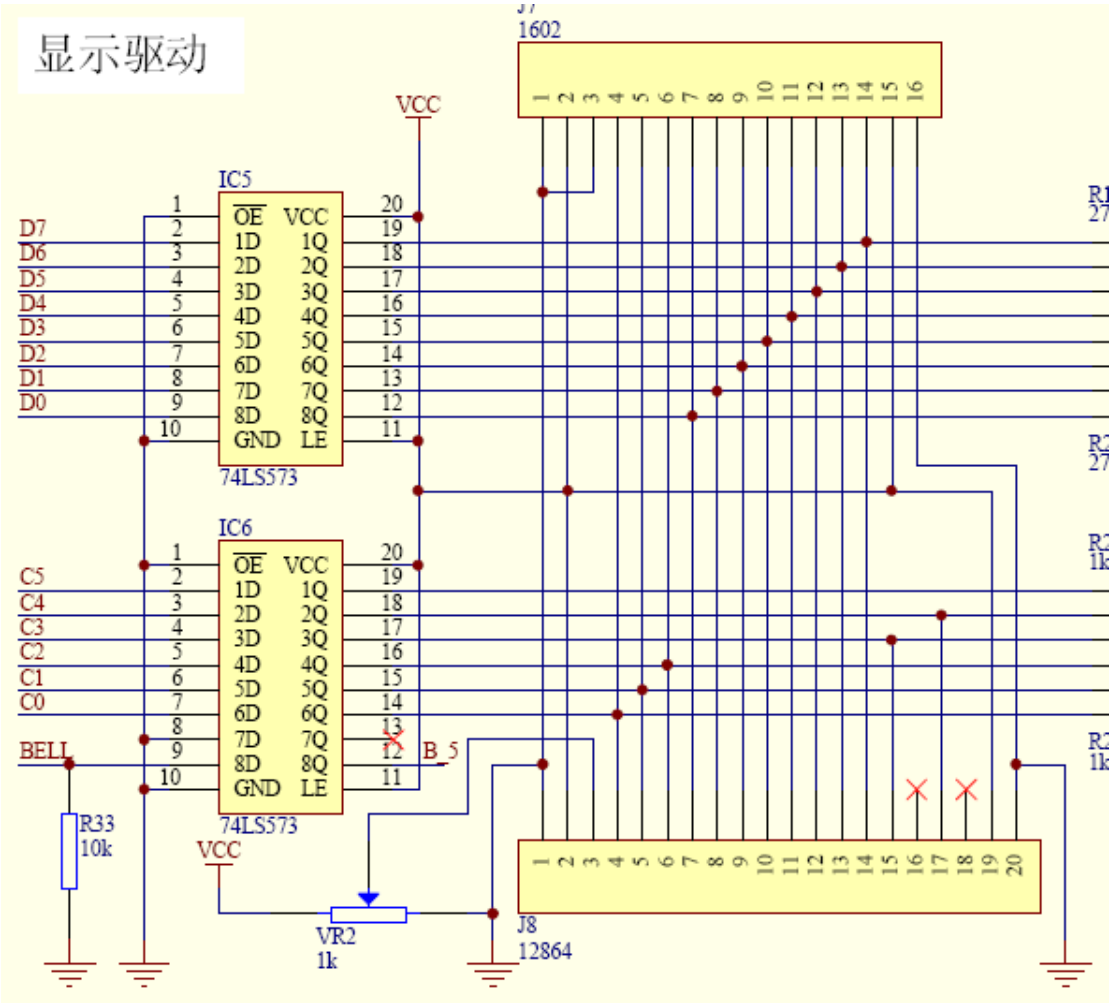
```
__interrupt void watchdog_timer(void)
{
    P6OUT = 0xfc;
    P4OUT = scandata[dN[cnt-2]];
    if(cnt==6) P4OUT &= 0x7f;   //在第二位显示小数点
    P6OUT &= ~(1<<cnt);

    cnt++;
    if(cnt == 8) cnt = 2;
}
/*****************************************
函数名称：Disp_Numb
功    能：将从DS18B20读取的11bit温度数据转换
          成数码管显示的温度数字
参    数：temper--11bit温度数据
返回值  ：无
*****************************************/
void Disp_Numb(uint temper)
{
    uchar i;

    for(i = 0;i < 6;i++) dN[i] = 0; //初始化显示变量

    //数值转换
    if(temper & BIT0)
    {
        dN[0] = 5;
        dN[1] = 2;
        dN[2] = 6;
    }
    if(temper&BIT1)
    {
        dN[1] += 5;
        dN[2] += 2;
        dN[3] += 1;
    }
    if(temper & BIT2)
    {
        dN[2] += 5;
        dN[3] += 2;
        if(dN[2] >= 10)
        {
            dN[2] -= 10;
            dN[3] += 1;
```

```
    }
}
if(temper&BIT3)
{
    dN[3] += 5;
}
if(temper & BIT4)
{
    dN[4] += 1;
}
if(temper & BIT5)
{
    dN[4] += 2;
}
if(temper & BIT6)
{
    dN[4] += 4;
}
if(temper & BIT7)
{
    dN[4] += 8;
    if(dN[4] >= 10)
    {
        dN[4] -= 10;
        dN[5] += 1;
    }
}
if(temper & BIT8)
{
    dN[4] += 6;
    dN[5] += 1;
    if(dN[4] >= 10)
    {
        dN[4] -= 10;
        dN[5] += 1;
    }
}
if(temper & BIT9)
{
    dN[4] += 2;
    dN[5] += 3;
    if(dN[4] >= 10)
    {
        dN[4] -= 10;
```

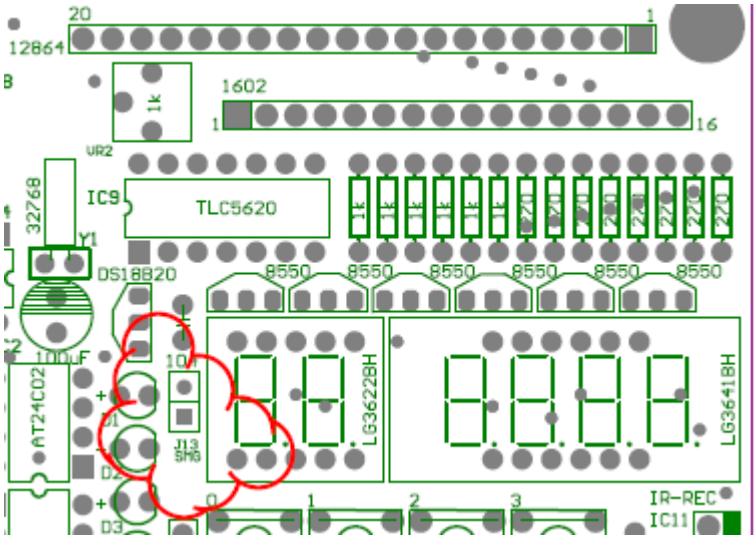
```
            dN[5] += 1;
        }
    }
    if(temper & BITA)
    {
        dN[4] += 4;
        dN[5] += 6;
        if(dN[4] >= 10)
        {
            dN[4] -= 10;
            dN[5] += 1;
        }
        if(dN[5] >= 10)
        {
            dN[5] -= 10;
        }
    }
}
/*****************************************************************************/
```

实验十三：LCD1602 液晶实验

实验电路：



在做 1602 或者 12864 液晶实验时，如果液晶显示亮度很低是由于 usb 供电不足所造成，可以选择将数码管控制 J13 跳帽拔下，以减少电流损耗!



实验结果为熟悉 LCD 的使用方法，将 LCD 本站网址和电话显示在液晶上！

```
/*******************************************************************************/
//实验目的：熟悉LCD的使用方法
//LCD循环显示本站网址和电话
//硬件要求：LCD直接与单片机的A口和D口相连接
//所有拨码开关置OFF
//调节电位器，调节LCD亮度。
#include<msp430x14x.h>

#define set(x) P6OUT |=x
#define clr(x) P6OUT &=~x
#define rs BIT2
#define rw BIT3
#define e  BIT4

const char web[ ]={' ','w','w','w','.','2','2','s','k','y','.','c','o','m',' ',' '};
//'C','1','6','.','C','0','M',' ',' '};
//显示公司web地址
const char tel[ ]={'T','E','L',':','1','3','0','7','4','5','6','7','7','0','4',' '};
//显示公司电话号码

void init();                    //申明I/O口初始化函数
void lcd_init();                //申明LCD初始化函数
void write_web();               //申明显示公司web地址函数
void write_tel();               //申明显示公司tel函数
void write(char x);             //申明显示1字节数据函数
void lcd_enable();              //申明LCD显示设置函数
void delay();                   //申明延时函数

//--------------------------------------
//主函数
```

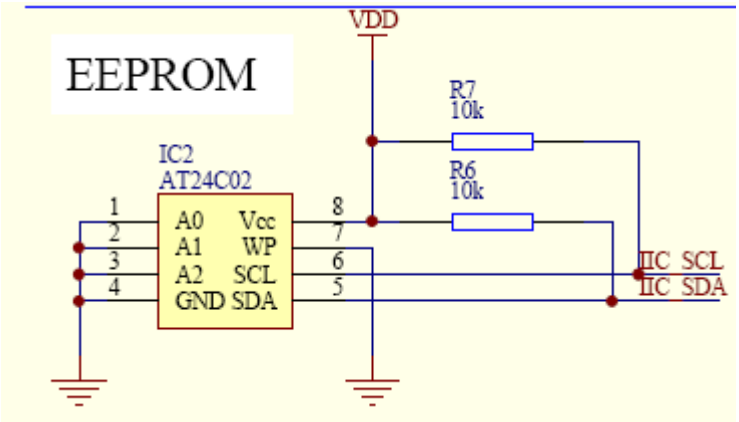
```
void main()
 {

    init();                //调用I/O口初始化函数
            lcd_init();            //调用LCD初始化函数
       write_web();          //调用显示公司web地址函数
       P4OUT=0Xc0;//PORTD=0XC0;            //设置第2行显示地址
       lcd_enable();         //调用LCD显示设置函数
       write_tel();          //调用显示公司tel函数
    while(1);
 }

//---------------------------------------
//I/O口初始化函数
void init()
 {
   //ADCON1=0X07;              //设置A口为普通I/O口
   //TRISA=0X00;               //设置A口为输出
   //TRISD=0X00;               //设置D口为输出
    P4DIR=0xff;
       P6DIR=0x1c;//设置A口为输出
    WDTCTL = WDTPW + WDTHOLD;  //杀狗


 }

//---------------------------------------
//LCD初始化函数
void lcd_init()
 {
   P4OUT=0x01;//PORTD=0X1;                //清除显示
   lcd_enable();
   P4OUT=0x38;//PORTD=0X38;                //8位2行5*7点阵
   lcd_enable();
   P4OUT=0x0e;//PORTD=0X0e;                //显示开，光标开，闪烁
   lcd_enable();
   P4OUT=0x06;//PORTD=0X06;                //文字不动，光标右移
   lcd_enable();
   P4OUT=0x80;//PORTD=0X80;                //公司web显示地址
   lcd_enable();
 }

//---------------------------------------
//显示公司web地址
```

```
void write_web()
 {
    int i;
    for(i=0;i<0x10;i++)        //一共16字节数据
       {
         write(web[i]);        //查表获取数据并调用写一个字节数据函数送LCD显示
       }
 }

//---------------------------------------
//显示公司tel函数
 void write_tel()
 {
    int i;
    for(i=0;i<0x10;i++)       //一共显示16字节数据
       {
         write(tel[i]);       //查表获取数据并调用写一个字节数据函数送LCD显示
       }
 }

//---------------------------------------
//写一个字节数据函数
void write(char x)
 {
  P4OUT=x;//PORTD=x;                    //待显示数据送PORTD口
  set(rs);//rs=1;                       //该字节数据为数据，而不是命令
  clr(rw);//rw=0;                       //此次操作为写，而不是读
  clr(e);//e=0;                        //拉低使能信号
  delay();                    //保持使能信号为低一段时间
  set(e);//e=1;                         //拉高使能信号，建立LCD操作所需要的上升沿
 }

//---------------------------------------
//LCD显示设置函数
void lcd_enable()
 {
   clr(rs);                     //该字节数据为命令，而不是数据
   clr(rw);                     //此次操作为写，而不是读
   clr(e);                      //拉低使能信号
   delay();                    //保持使能信号为低一段时间
   set(e);                      //拉高使能信号，建立LCD操作所需要的上升沿
 }

//---------------------------------------
```

```
//延时函数
void delay()
 {
   unsigned int i;
   for(i=0;i<10000;i++);
 }
/*****************************************************************************/
```

实验十四：24C02（I2C 协议）实验

实验电路：



i2c：实验结果为 MCU 先向 I2C 写十个数，然后再从 I2C 中读出显示在数码管上。
i2c_uart：实验结果为 PC 机通过串口可以控制从 EEPROM 的指定地址中读出一个字节的数据，或者向指定地址写入一个字节的数据
i2c\i2c

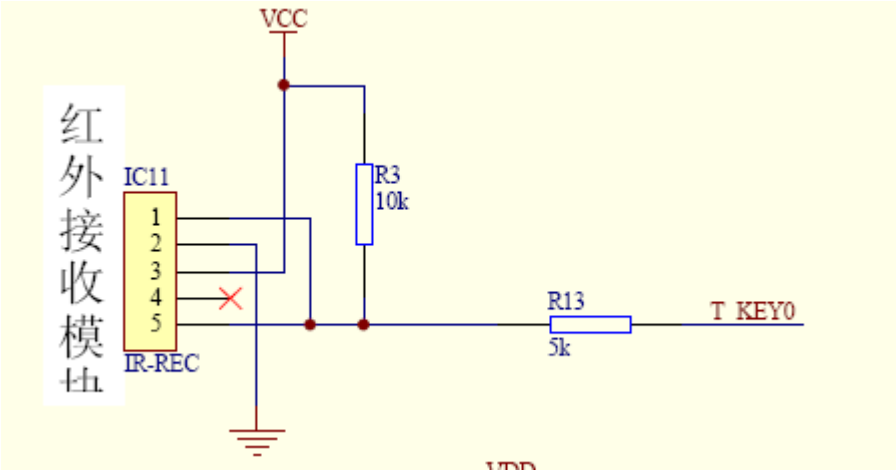
```
/*******************************************************************************/
/*************************************************
程序功能：PC机通过串口可以控制从EEPROM的指定地址中读出一个
          字节的数据，或者向指定地址写入一个字节的数据
特别说明：在PC机发送EEPROM内的存储地址或者是写入数据时，必须
          使用十六进制发送，而在发送读写控制命令时要使用默认
          的发送ASCII码的设置，这点要注意，否则操作无法完成
-------------------------------------------------
通信格式：N.8.1,  9600
-------------------------------------------------
测试说明：打开串口调试精灵，正确设置通信格式，根据屏幕显示的
          提示信息发送控制数据进行测试
*************************************************/
#include <msp430x14x.h>
#include "EEPROM.c"
uchar table[16] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                   0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};
delay1(uint ts)
{while(ts--);
}
void main( void )
{
  uchar i;
  uchar rd[10]={0};
```

```
    WDTCTL = WDTPW + WDTHOLD;            //关狗
    P4DIR = 0xff;
    P6DIR = 0xfc;
    P4OUT = 0x00;
    P6OUT = 0xfc;
    P3DIR |= BIT3 + BIT1;                //设置相应端口为输出状态
    P3OUT |= BIT3 + BIT1;
      for(i=0;i<10;i++)
      {
       delay1(2000);              //24c02写入后需要擦除操作，一定要加延时
       Write_1Byte(i,i);            //功     能：向EEPROM中写入1个字节的数据
       }
    delay1(2000);              //24c02写入后需要擦除操作，一定要加延时

  for(i=1;i<10;i++)
  {
  rd[i]=Read_1Byte_Randomaddress(i);
      P4OUT=table[rd[i]];
          P6OUT = ~BIT2;
      delay1(20000);              //24c02写入后需要擦除操作，一定要加延时
  }

    while(1);

}
/*****************************************************************************/
i2c\i2c_uart
/*****************************************************************************/
/*************************************************
程序功能：PC机通过串口可以控制从EEPROM的指定地址中读出一个
          字节的数据，或者向指定地址写入一个字节的数据
特别说明：在PC机发送EEPROM内的存储地址或者是写入数据时，必须
          使用十六进制发送，而在发送读写控制命令时要使用默认
          的发送ASCII码的设置，这点要注意，否则操作无法完成
-------------------------------------------------------
通信格式：N.8.1,  9600
-------------------------------------------------------
测试说明：打开串口调试精灵，正确设置通信格式，根据屏幕显示的
          提示信息发送控制数据进行测试
*************************************************/
#include <msp430x14x.h>
#include "EEPROM.c"
```

```
#include "uart.c"
uchar step = 0xff;

void main( void )
{
    uchar i;

    WDTCTL = WDTPW + WDTHOLD;           //关狗

    P3DIR |= BIT3 + BIT1;               //设置相应端口为输出状态
    P3OUT |= BIT3 + BIT1;
    InitUART();                         //初始化UART端口

    DisplayConfigMenu();
    while(1)
    {
        Send1Char('>');
        Send1Char('\n');
        i=Get1Char();
        HandleConfigMenu(i);
    }

}
/*****************************************************************************/
```

实验十五：HS0038 红外接收解码实验

实验电路：



HS0038：实验结果为将遥控器按下键值，在数码管上显示解码结果。

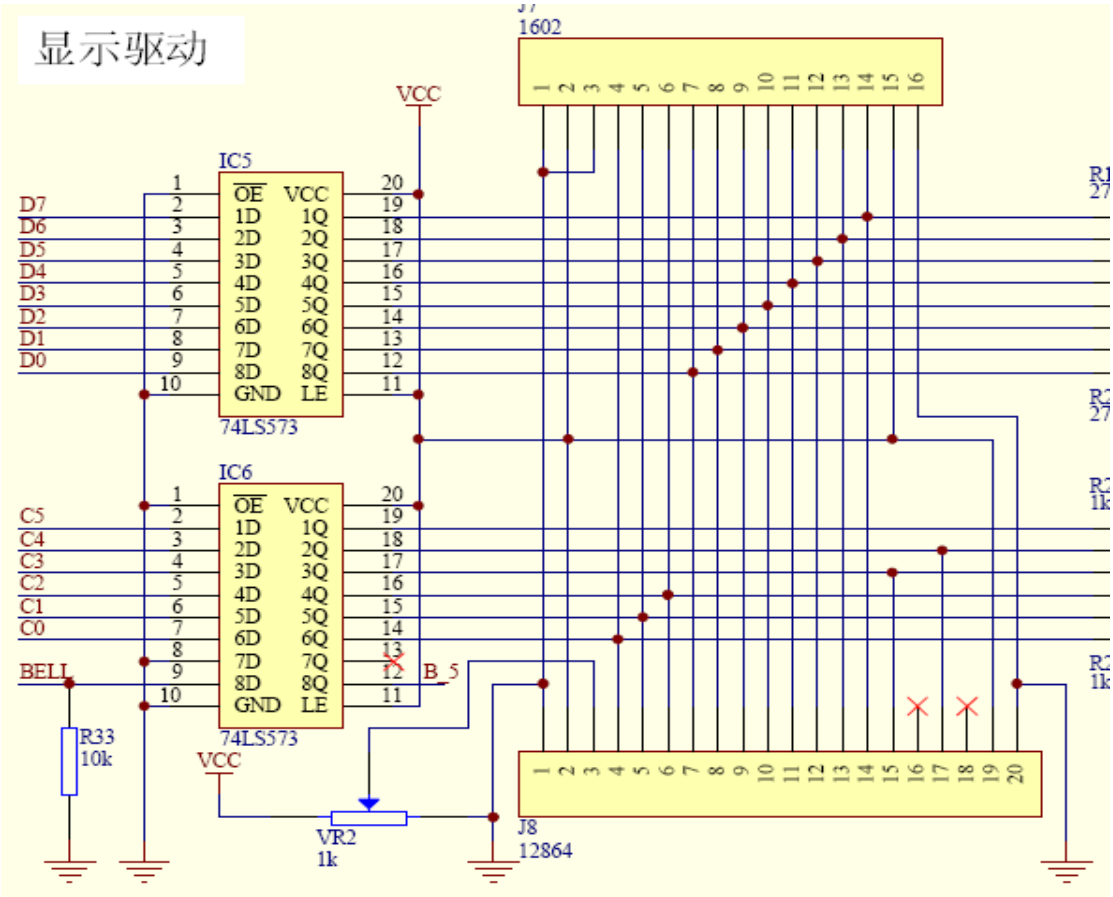
```
/*******************************************************************************/
#include  <MSP430X14X.h>
#define   uchar   unsigned char
#define   uint    unsigned int
#define t_3ms5  2300
#define t_1ms   666
uchar get_code[4] = {0};
uchar dat_code=0;
uint  timer;

#include "msp430.c"
#include "disp_4led.c"

void delay(uint time)    //10ms--10000
{
 while(time--);
}

void dat_high()
{P1DIR |= BIT6;
 P1OUT |= BIT6;
}
void dat_low()
{P1DIR |= BIT6;
 P1OUT &= ~BIT6;
}
```

```
uchar rd_dat()
{ uchar stat;
  P1DIR |= BIT6;
  P1OUT |= BIT6;
  P1DIR &= ~BIT6;
  stat = P1IN;
  return (stat);
}

uchar get_num()
{ uchar i,j,rd,dat=0;
  _DINT();              //关闭中断
  for(j=0;j<4;j++)
  { for(i=8;i>0;i--)
     {dat>>=1;
      do
       rd=rd_dat();
      while(!(rd & BIT6));  //   wait high
      set_timer_b0();          //检测高电平时间
      do
      rd=rd_dat();
      while(rd & BIT6);//   wait low
      timer=TBR;
      stop_timer_b0();
       if(timer>t_1ms)dat=dat|0x80;
      else dat=dat&0x7f;
      }
   get_code[j]=dat;
   }
 _EINT();
return(get_code[2]);
}

void main( void )
{ uchar rd;
  WDTCTL = WDTPW + WDTHOLD;
  dat_high();
  set_timer_a0();
judge:
  do
  rd=rd_dat();
  while(rd & BIT6);    //wait low
  delay(100);
  rd=rd_dat();
```

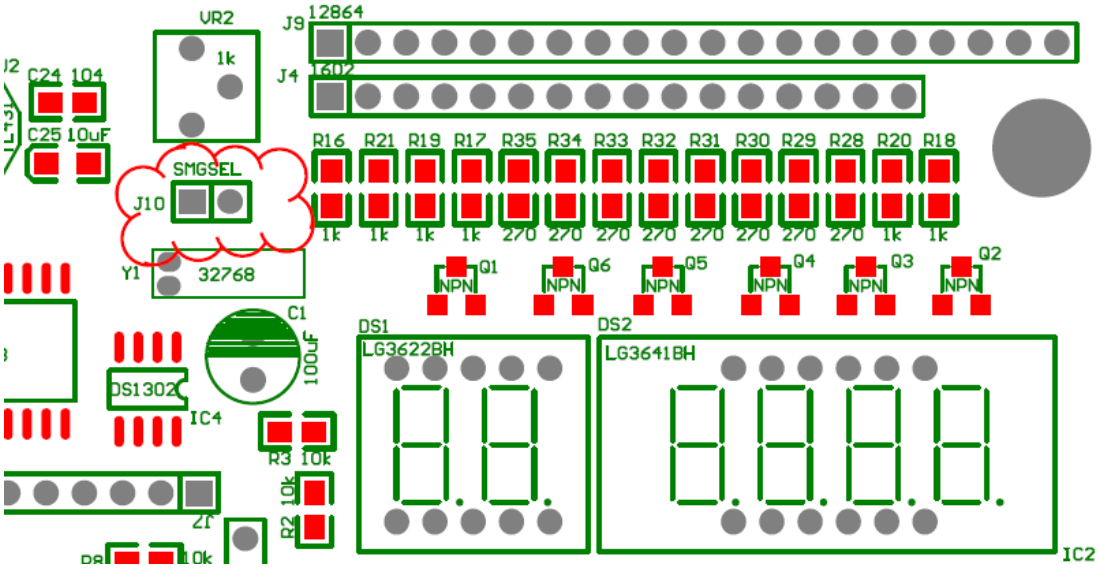
```
  if(rd & BIT6)goto judge; //xiao dou
  do
  rd=rd_dat();
  while(!(rd & BIT6));  //wait high
  set_timer_b0();
  do
   rd=rd_dat();
  while(rd & BIT6);     //wait low
  timer=TBR;
  stop_timer_b0();
  if(timer<t_3ms5){delay(1000);goto judge;}//连续按下，重复标志
  dat_code=get_num();
  goto judge;
}
/*****************************************************************************/
```

实验十六：LCD12864 液晶实验

实验电路：



在做 1602 或者 12864 液晶实验时，如果液晶显示亮度很低是由于 usb 供电不足所造成，可以选择将数码管控制 J10 跳帽拔下，以减少电流损耗!



lcd12864b：实验结果为在屏幕上显示“祥瑞不在凤凰麒麟，太平须得边将忠臣，但得百僚师长肝胆，不用三军罗绮金银”。
lcd12864c：实验结果为在液晶上显示“**天空电子**”
lcd12864\lcd12864b

```
/*****************************************************************************/
/*************************************************************
程序功能：在12864液晶上显示一屏汉字
测试说明：观察液晶显示
*************************************************************/
#include  "msp430.h"
#include  "cry12864.c"

const uchar hang1[] = {"祥瑞不在凤凰麒麟"};
const uchar hang2[] = {"太平须得边将忠臣"};
const uchar hang3[] = {"但得百僚师长肝胆"};
const uchar hang4[] = {"不用三军罗绮金银"};

/*************************主函数************************/
void main( void )
{

    WDTCTL = WDTPW + WDTHOLD;     //关狗

    Ini_Lcd();                   //初始化液晶

    Disp_HZ(0x80,hang1,8);
    Disp_HZ(0x90,hang2,8);
    Disp_HZ(0x88,hang3,8);
    Disp_HZ(0x98,hang4,8);

    LPM4;
}
/*****************************************************************************/
```

lcd12864\lcd12864c

```
/*****************************************************************************/
#include <msp430x14x.h>
#define uint unsigned int
#define uchar unsigned char
//p62=cs p63=sid p64=clk p65=psb p66=rst

uchar set0[]={0x30,0x0f,0x01,0x06};//写字
uchar set1[]={0x30,0x0c,0x06,0x32,0x36};//画图
void delay(uint time)
{
```

```
  while(time--);
}
void initsys(void)
{
  uchar k;
  BCSCTL1 &= ~XT2OFF;    //开启XT2晶振
  do
  {
    IFG1 &= ~OFIFG;           // 清除振荡器失效标志
    for(k=0xee;k>0;k--)      //延时
      {
       ;
      }
    }
   while((IFG1 & OFIFG)!=0);        //判断XT2是否起振
   BCSCTL2 = SELM_2 + SELS;     // 选择MCLK SMCLK 为XT2
}    //初始化
void wri_ord(uchar num)
{
 uchar num1,num2;
 uchar wri_data;
 uchar i,j;
 num1 = num & 0xf0;
 num <<= 4;
 num2 = num & 0xf0;
 uchar data[]={0xf8,num1,num2};
 for(i=0;i<3;i++)
 {
   wri_data = data[i];
   P6OUT &= ~(BIT4);
   delay(5);
   for(j=0;j<8;j++)
   {
      if(wri_data & BIT7)
        P6OUT |= BIT3;
      else
        P6OUT &= ~(BIT3);
      delay(5);
      P6OUT |= BIT4;
      wri_data <<= 1;
      delay(10);
      P6OUT &= ~(BIT4);
   }
 }
```

```
}
void wri_data(uchar num)
{
 uchar num1,num2;
 uchar wri_data;
 uchar i,j;
 num1 = num & 0xf0;
 num <<= 4;
 num2 = num & 0xf0;
 uchar data[]={0xfa,num1,num2};
 for(i=0;i<3;i++)
 {
   wri_data = data[i];
   P6OUT &= ~(BIT4);
   for(j=0;j<8;j++)
   {
      if(wri_data & BIT7)
        P6OUT |= BIT3;
      else
        P6OUT &= ~(BIT3);
      _NOP();
      P6OUT |= BIT4;
      _NOP();
      wri_data <<= 1;
      P6OUT &= ~(BIT4);
   }
 }

}
void show()
{
  uchar data[] = {"  **天空电子**"};
  uchar i;
  wri_ord(0x80);
  for(i=0;i<14;i++)
  {
    wri_data(data[i]);
  }
}
void initial(void)
{
  uchar add;
  P6OUT &= ~(BIT5);
```

```
  _NOP();
  P6OUT &= ~(BIT2);
  _NOP();
  P6OUT |= BIT2;
  _NOP();
  P6OUT &= ~(BIT5);
  for(add=0;add<4;add++)
  {
     wri_ord(set0[add]);
  }
  delay(1000);
}
void main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;
  //initsys();
  P6DIR |= 0xff;
  P6OUT |= BIT6;
  initial();
  //wri_ord(0x88);
  show();
  while(1);
}
/*****************************************************************************/
```