MSP430学习笔记

MSP430学习笔记1

```
/*************************************************
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
孙浩，2010.7.20
*************************************************/
#include   <msp430x14x.h>
/****************主函数****************/
void main(void)
{
    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;

    WDTCTL = WDTPW + WDTHOLD;          //与 AVR 不同，默认看门狗是打开的，所以要先关闭看门狗

    P6DIR |= BIT2;P6OUT |= BIT2;      //关闭电平转换，这个是对应开发板上的电平转换芯片，与实际功能无关
    CCTL0 = CCIE;                      //使能 CCR0 中断
    CCR0 = 2047;                       //设定周期 0.5S  计算:32768/8/2=2048；使用的是手表晶振
    TACTL = TASSEL_1 + ID_3 + MC_1; //定时器 A 的时钟源选择 ACLK，增计数模式
    //在 msp430x14x.h 已经对相应的寄存器和标志位都做了定义，只需要进行选择即可具体查看头文件。
    //TASSEL_1 表示时钟源选择 ACLK，ID_3 表示八分频，MC_1 表示增计数模式
    P2DIR = 0xff;                      //设置 P2 口方向为输出
    P2OUT = 0xff;

    _EINT();                           //使能全局中断
    LPM3;                              //设置工作模式，使 CPU 进入 LPM3 模式，此模式下 CPU MCLK SMCLK 被禁止  ACLK 活动
}
```

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

MSP430学习笔记2-跑马灯程序，熟悉定时器寄存器的配置

```
/*************************************************
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"
-------------------------------------------------
拨码开关设置：将 LED 位拨至 ON，其余拨至 OFF
测试说明：观察流水灯流动顺序和速度的变化
*************************************************/
#include   <msp430x14x.h>
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)
{

    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;

    WDTCTL = WDTPW + WDTHOLD;           //关闭看门狗
    P6DIR |= BIT2;P6OUT |= BIT2;       //关闭电平转换
    P5OUT &= ~BIT7;

    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，8 分频增计数
模式
    P2DIR = 0xff;                       //设置 P2 口方向为输出
    P2OUT = 0xff;

    _EINT();                            //使能全局中断
    LPM0;                               //CPU 进入 LPM0 模式各时钟全部工作
}
/*****************************************
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 == 10)
        {
            i = 0;
            j = 0;
            flag++;
            if(flag == 4) flag = 0;
            switch(speed)//根据 speed 的值来选择相应的速度，速度的选择主要通过改变分频来
实现
            {
            case 0:
                TACTL &=~ (ID0 + ID1);//这个地方其实没必要，意思是清零 TACTL 的 78 两
位
                //ID1 的值是 0x0080     ID2 的值是 0x0040 通过计算就可以得出
                TACTL |= ID_3;//改为八分频
                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;
        }
}
```

MSP430学习笔记3-PWM 的产生

这个程序主要是利用定时器的比较输出功能来产生 PWM 波控制 LED，定时器 A 的比较输出对应 P2.3 P2.4，因此在程序的一开始需要设置比较匹配的工作模式，需要说明的是头文件中已经对各种模式给了详细的定义，不需要在去配置寄存器，程序较为简单，稍微扩展一下去控制舵机也很容易，大家自己分析。

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

程序功能：用从 P2.3和 P2.4输出的 PWM 波形驱动 LED 闪烁

```
-------------------------------------------------------
```

拨码开关设置：将 LED 位拨至 ON，其余拨至 OFF

```
测试说明：观察 LED 的亮灭的时间长短
**********************************************************/
#include   <msp430x14x.h>

void main(void)
{
  /*下面六行程序关闭所有的 IO 口*/
     P1DIR = 0XFF;P1OUT = 0XFF;
     P2DIR = 0XFF;P2OUT = 0XFF;
     P3DIR = 0XFF;P3OUT = 0XFF;
     P4DIR = 0XFF;P4OUT = 0XFF;
     P5DIR = 0XFF;P5OUT = 0XFF;
     P6DIR = 0XFF;P6OUT = 0XFF;

  WDTCTL = WDTPW + WDTHOLD;                    // 关狗
  P6DIR |= BIT2;P6OUT |= BIT2;                //关闭电平转换
  P2DIR = 0xff;                               // P2端口设置为输出
  P2OUT = 0xff;                               // 关闭其他 LED
  P2SEL |= BIT3 + BIT4;                       // P2.3和 P2.4连接内部模块,使用的是第二
功能作为 TimerA 的比较输出。
  CCR0 = 4096-1;                              // PWM 周期为1S
  CCTL1 = OUTMOD_7;                           // 捕捉比较寄存器的配置，工作在方
式7，计数值等于比较值时输出置位
  CCR1 = 4000;                                // 输出1的比较值
  CCTL2 = OUTMOD_7;                           // CCR2 reset/set
  CCR2 = 500;                                 // 输出2的比较值改变此值即可以改变输
出的 pwm 波的占空比
  TACTL = TASSEL_1 + ID_3 + MC_1;             // ACLK/8, up mode

  //_BIS_SR(LPM3_bits);                       // Enter LPM3
  LPM3;

}
```

MSP430学习笔记4-两个定时器产生步进单频音

这个程序是开发板中用来产生不同频率声音的程序，整体程序较为简单，主要是两个定时器的使用，代码及我的注释如下。

```
/*********************************************************
程序功能：用固定频率的方波驱动蜂鸣器，共16种音调；在蜂鸣器
          发出不同音调的同时，LED 发光以二进制数字形式指示
          当前音调的编号（1~16）
---------------------------------------------------------
拨码开关设置：将 BUZZER 位拨至 ON，其余位拨至 OFF
测试说明：聆听蜂鸣器发声的音调变化。同时 led 也有对应的指示
```

```
**********************************************************/
#include <msp430.h>

typedef unsigned char uchar;

uchar step = 0xff;

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

    WDTCTL = WDTPW + WDTHOLD;                   //关狗

    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;
    P6DIR |= BIT2;P6OUT |= BIT2;              //关闭电平转换

    /*------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
        //IFG1是中断寄存器  OFIFG 是晶振启动失败中断标志位
        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？
    //上面这一步主要是等待晶振正常工作
    BCSCTL2 |= SELM_2 + SELS;                  //MCLK 和 SMCLK 选择高频晶振

    TACCTL0 |= CCIE;                            //使能比较中断
    TACTL |= TASSEL_2 + ID_3 ;                //计数时钟选择 SMLK=8MHz，1/8分频后为
1MHz

    TBCCR0 = 4096*2 - 1;                        //周期两秒
    //时间计算：32768/8*2+1注意使用的是手表晶振
    TBCCTL0 |= CCIE;
    TBCTL |= TBSSEL_1 + ID_3 + MC_1;       //时钟源 ACLK/8，up mode
```

```
    P6DIR |= BIT7;                          //蜂鸣器对应 IO 为6.7设置为输出
    P2DIR = 0xff;                           //指示对应的状态
    P2OUT = 0xff;

    _EINT();

    LPM1;
}
/*****************************************
函数名称：Timer_A
功    能：定时器 A 的中断服务函数，在这里驱动
          蜂鸣器发声
参    数：无
返回值  ：无
*****************************************/
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  P6OUT ^= BIT7;                                // Toggle P6.7
}
/*****************************************
函数名称：Timer_B
功    能：定时器 B 的中断服务函数，在这里更改
          蜂鸣器发声频率
参    数：无
返回值  ：无
*****************************************/
#pragma vector=TIMERB0_VECTOR
__interrupt void Timer_B (void)
{
    if(step == 0xff)                            //step 的初值就是0xff，
      TACTL |= MC_1;//需要将 TimerA 设置为增计数模式，可以在初始化的时候设置，不明白为什么放在这个地方。
    step++;
    switch(step)
    {
    case    0:  TACCR0 = 5000;  P2OUT = ~1;   break;      // 100Hz
    //P2OUT 使用 led 显示对应的数值，只是为了便于演示，没实际意义
    case    1:  TACCR0 = 2500;  P2OUT = ~2;   break;      // 200Hz
    case    2:  TACCR0 = 1250;  P2OUT = ~3;   break;      // 400Hz
    case    3:  TACCR0 = 625;   P2OUT = ~4;   break;      // 800Hz
    case    4:  TACCR0 = 500;   P2OUT = ~5;   break;      // 1KHz
    case    5:  TACCR0 = 250;   P2OUT = ~6;   break;      // 2KHz
    case    6:  TACCR0 = 167;   P2OUT = ~7;   break;      // 3KHz
```

```
        case    7:  TACCR0 = 125;   P2OUT = ~8;   break;        // 4KHz
        case    8:  TACCR0 = 100;   P2OUT = ~9;   break;        // 5KHz
        case    9:  TACCR0 = 83;    P2OUT = ~10;   break;        // 6KHz
        case    10:  TACCR0 = 71;   P2OUT = ~11;   break;        // 7KHz
        case    11:  TACCR0 = 63;   P2OUT = ~12;   break;        // 8KHz
        case    12:  TACCR0 = 56;   P2OUT = ~13;   break;        // 9KHz
        case    13:  TACCR0 = 50;   P2OUT = ~14;   break;        // 10KHz
        case    14:  TACCR0 = 33;   P2OUT = ~15;   break;        // 15KHz
        case    15:  TACCR0 = 25;   P2OUT = ~16;   break;        // 20KHz
        case    16:  step = 0xff;                   // 接着往上加，和清零的效果一样，循环播
放
    }
}
```

MSP430学习笔记5-利用蜂鸣器演奏音乐

这个小程序仍然是定时器的运用，比较简单，具体的地方都在注释中注明了，参考注释。

```
/*********************************************
程序功能：MCU 控制蜂鸣器演奏歌曲《祝你平安》
---------------------------------------------
拨码开关设置：将 BUZZER 位拨至 ON，其余位拨至 OFF
测试说明：聆听蜂鸣器“唱出”的乐曲

既然是演奏乐曲对于一个音符应该包括两个部分
一是声调 二是持续时间，在这个程序中声调是用简单的
延时-电平翻转来实现的，改变了延时的时间就改变了
声调，而时间是通过计数比较来实现的，当计数值相等时
就跳出循环演奏下一个音符。
*********************************************/
#include   <msp430x14x.h>

typedef unsigned char uchar;

#include   "music.h"

#define    Buzzer          BIT7
#define    Buzzer_Port     P6OUT
#define    Buzzer_DIR      P6DIR

uchar counter;
void Play_Song(void);

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

```
    uchar i;

    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;
    P6DIR |= BIT2;P6OUT |= BIT2;              //关闭电平转换

    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;//改变这个值就改变了演奏的速度
    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                    ps：不知道这个地方怎么算出来的是延时毫秒
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;//电平取反
                Delay_Nms(Temp1);//Temp1的值决定了延时的长短，也决定了声音的频率
                if ( Temp2 == counter )//决定了音调持续的时间，计数时间到时就跳出循环
演奏下一个。
                {
                    counter = 0;
                    break;
                }
            }
        }
    }
}
```

MSP430学习笔记6-动态数码管的显示

这个程序主要部分是我改写的，程序把看门狗当做普通定时器使用，在看门狗中断中进行数码管扫描，另外我加入了显示的缓冲区，想在数码管上显示什么数字只需要对缓冲数组中的值进行操作即可。最后加入了小数点的操作，可以很方便的加入小数点，程序如下：

```
/*********************************************
程序功能：在八位数码管上显示任意数字
---------------------------------------------
拨码开关设置：将 SMG 位拨至 ON，其余拨至 OFF
测试说明：观察数码管显示
*********************************************/
#include   <msp430x14x.h>
typedef unsigned char uchar;

//数码管7位段码：0--f
uchar scandata[16] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                      0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};//段码
uchar dispbitcode[]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};//将端口的片选信号编码
uchar disbuf[8] = {8,8,1,3,1,9,4,2};//存放要显示的数据
//记录显示位数的全局变量
uchar cnt = 0;//用来扫描计数
uchar dotn = 2;//用来选择第几位后面显示小数点

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

```
void main(void)
{
    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;

    //这里面看门狗用作普通的定时器，要来对数码管进行动态扫描，如果看门狗有其他作
用
    //可以用定时器代替
    WDTCTL = WDT_ADLY_1_9;            // 设置内部看门狗工作在定时器模式，1.9ms 中
断一次
    //可以去看头文件中具体的配置，这里使用的手表晶振64分频计算可得是1.9ms
    IE1 |= WDTIE;                     // 使能看门狗中断

    P6DIR |= BIT2;P6OUT |= BIT2;   //关闭电平转换

    P5DIR = 0xff;                     //设置 P4，P5的 IO 方向为输出
    P4DIR = 0xff;

    P5OUT = 0x00;                     //设置 P4，P5的输出初值
    P4OUT = 0xff;

    _BIS_SR(LPM3_bits + GIE);         //CPU 进入 LPM3低功耗模式，同时打开全局中断
    //这个地方要注意 LPM3低功耗模式下其他时钟源是关闭的，只能使用 ACLK 时钟源
}

/*****************************************
函数名称：watchdog_timer
功    能：看门狗中断服务函数，在这里输出数码管的
          段选和位选信号
参    数：无
返回值  ：无
*****************************************/
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
    P4OUT = 0xff;
    uchar temp;
     if(cnt == (dotn-1))//判断要不要显示小数点
    {
```

```
        temp = scandata[disbuf[cnt]]&0x7F;//如果要显示的就将要显示的段码先进行一步运算
        P5OUT = temp;
    }
    else
    P5OUT = scandata[disbuf[cnt]];          //输出段选信号
    P4OUT ^= dispbitcode[7-cnt];              //输出位选信号由于用了三极管驱动，实际是低电平选通因此翻一下。
    //发现显示的顺序是倒着的，减一下把顺序正过来
    cnt++;                                  //位计数变量在0~8之间循环
    if(cnt == 8) cnt = 0;


}
```

MSP430学习笔记7-4*4键盘的对应数码管显示

这个例程也是开发板上面的一个经典例程，我对程序的框架进行了修改，更适合以后的调用。具体的4*4键盘扫描原理较为基础这里不再赘述，这里我认为比较重要的是要养成一个良好的写程序的习惯，想4*4键盘扫描这样的函数可以写成一个固定的 C 或者 H 文件，便于以后调用。先看看主程序：

```
/************************************************
程序功能：扫描4X4键盘并将键值在数码管上显示

------------------------------------------------
测试说明：按动 K1~K16按键，观察数码管显示
************************************************/
#include <msp430x14x.h>
#include "Keypad.C"

//数码管7位段码：0--f
unsigned char scandata[16] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,
                              0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e};
//记录显示位数的全局变量
unsigned char cnt = 0;
//显示缓存
unsigned char Dispbuf[2];

//引用外部变量的声明
extern unsigned char key_Pressed;
extern unsigned char key_val;
extern unsigned char key_Flag;


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

```
{
    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;

    P6DIR |= BIT2;P6OUT |= BIT2;   //关闭电平转换
    WDTCTL = WDT_ADLY_1_9;          //设置内部看门狗工作在定时器模式，1.9ms 中
断一次
    IE1 |= WDTIE;                   //使能看门狗中断

    _EINT();                        //打开全局中断
    Init_Keypad();
    while(1)
    {
        Key_Event();

        if(key_Flag == 1)
        {
            key_Flag = 0;
            Dispbuf[1] = key_val / 10;
            Dispbuf[0] = key_val % 10;

        }
    }
}
/*****************************************
函数名称：watchdog_timer
功    能：看门狗中断服务函数，在这里输出数码管的
          段选和位选信号
参    数：无
返回值  ：无
*****************************************/
#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
    P4OUT = 0xff;
    P5OUT = scandata[Dispbuf[cnt]];          //输出段选信号
    P4OUT &= ~(1 << (cnt+2));                //输出位选信号

    cnt++;                                   //位计数变量在0~1之间循环
```

```
    if(cnt == 2) cnt = 0;
}
```

主程序较为简单，具体的原理参见上一个笔记，在主程序中与键盘扫描相关的函数被击中在 Keypad.C 中，需要说明的是在主函数中需要对 Keypad.C 的部分变量做声明，这样才能使用 Keypad.C 中的变量。Keypad.C 代码如下：

```
/************************************************************
4*4键盘扫描函数
作者：孙浩
修改时间：2010.8.2
程序说明：
此程序以开发板的例程做部分修改，更方便程序调用。
在调用之前需要注意需要在主函数中对程序中使用的变量做外部变量声明
需要在主函数中声明的变量如下：
//引用外部变量的声明
extern unsigned char key_Pressed; //按键是否被按下:1--是，0--否
extern unsigned char key_val;         //存放键值
extern unsigned char key_Flag;       //按键是否已放开：1--是，0--否
另外对应的键值可以在本文件的全局变量中修改，只需要对应修改
数组 uchar key_Map[]的对应的值即可。
调用示例如下：
Init_Keypad();              //先进行初始化
    while(1)
    {
        Key_Event();    //在死循环中进行键值扫描

        if(key_Flag == 1)//用来判断有无按键按下，从而读取键值进行操作
        {
            key_Flag = 0;//key_Flag 需要手动清零
        }
    }
************************************************************/
#include   <msp430x14x.h>
//相应的函数声明
void Init_Keypad(void);//键盘初始化
void Check_Key(void);
void delay();
void Key_Event(void);//读取键盘函数，在使用时通过此函数进行键盘扫描

//选择对应的端口
#define KEYOUT P1OUT
#define KEYIN P1IN
#define KEYDIR P1DIR
```

```
typedef unsigned char uchar;
typedef unsigned int    uint;

/***************全局变量***************/
uchar key_Pressed;          //按键是否被按下:1--是，0--否
uchar key_val;              //存放键值
uchar key_Flag;             //按键是否已放开：1--是，0--否
//设置键盘逻辑键值与程序计算键值的映射
uchar key_Map[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16};

/*****************************************
函数名称：Init_Keypad
功      能：初始化扫描键盘的 IO 端口
参      数：无
返回值   ：无
*****************************************/
void Init_Keypad(void)
{
    KEYDIR = 0xf0;          //KEY.0~KEY.3设置为输入状态, KEY.4~KEY.7设置为输出状态
    KEYOUT |= 0xf0;         // KEY.4~KEY.7输出高电平
    key_Flag = 0;
    key_Pressed = 0;
    key_val = 0;
}
/*****************************************
* Check_Key(),检查按键，确认键值
*****************************************/
/*****************************************
函数名称：Check_Key
功      能：扫描键盘的 IO 端口，获得键值
参      数：无
返回值   ：无
*****************************************/
void Check_Key(void)
{
    uchar row ,col,tmp1,tmp2;

    tmp1 = 0x80;
    for(row = 0;row < 4;row++)                  //行扫描
    {
        KEYOUT = 0xf0;                          //KEY.4~KEY.7输出全1
        KEYOUT -= tmp1;                         //KEY.4~p1.7输出四位中有一个为0
        tmp1 >>=1;
        if ((KEYIN & 0x0f) < 0x0f)              //是否 KEYIN 的 KEY.0~KEY.3中有一位为0
```

```
        {
            tmp2 = 0x01;                                    // tmp2用于检测出那一位为0
            for(col = 0;col < 4;col++)                      //  列检测
            {
                if((KEYIN & tmp2) == 0x00)                  //  是否是该列,等于0为是
                {
                    key_val = key_Map[row * 4 + col];   //  获取键值
                    return;                                 //  退出循环
                }
                tmp2 <<= 1;                                 // tmp2右移1位
            }
        }
    }
}
/*****************************************
函数名称：delay
功      能：延时约15ms，完成消抖功能
参      数：无
返回值   ：无
*****************************************/
void delay()
{
    uint tmp;

    for(tmp = 12000;tmp > 0;tmp--);
}
/*****************************************
函数名称：Key_Event
功      能：检测按键，并获取键值
参      数：无
返回值   ：无
*****************************************/
void Key_Event(void)
{
    uchar tmp;

    KEYOUT &= 0x00;                     //  设置 KEYOUT 全为0，等待按键输入
    tmp = KEYIN;                        //  获取  p1IN
    if ((key_Pressed == 0x00)&&((tmp & 0x0f) < 0x0f)) //如果有键按下
    {
        key_Pressed = 1;            //  如果有按键按下，设置 key_Pressed 标识
        delay();                    //消除抖动
        Check_Key();                //  调用 check_Key(),获取键值
    }
```

```
    else if ((key_Pressed == 1)&&((tmp & 0x0f) == 0x0f)) //如果按键已经释放
    {
        key_Pressed = 0;            //  清除 key_Pressed 标识
        key_Flag    = 1;            //  设置 key_Flag 标识
    }
    else
    {
        _NOP();
    }
}
```

具体的注意提防都已经在注释中做了说明，在原来的程序中这个文件是分为 Keypad.C 和 Keypad.h 两个文件，键盘扫描函数的声明单独为一个头文件，我觉得并不是特别需要。源程序中全局变量是单独列为一个头文件 gdata.h 的。这样在全局变量比较多的情况下可以使程序更有条理可以参考。

MSP430学习笔记8-ST7920 12864液晶显示并行接口

ST7920较为常见，由于自带字库使用较为方便。下面的这个程序是根据开发板上的例程做适当修改而来，总的来说较为简单，对着 ST7920的 DATASheet 看一下时序和指令集就能明白。驱动程序如下

```
#include <msp430x14x.h>
typedef unsigned char uchar;
typedef unsigned int    uint;

extern const unsigned char shuzi_table[];

#define LCD_DataIn       P4DIR=0x00       //数据口方向设置为输入
#define LCD_DataOut      P4DIR=0xff       //数据口方向设置为输出
#define LCD2MCU_Data     P4IN
#define MCU2LCD_Data     P4OUT
#define LCD_CMDOut       P3DIR|=0x07        //P3口的低三位设置为输出
#define LCD_RS_H         P3OUT|=BIT0          //P3.0
#define LCD_RS_L         P3OUT&=~BIT0         //P3.0
#define LCD_RW_H         P3OUT|=BIT1          //P3.1
#define LCD_RW_L         P3OUT&=~BIT1         //P3.1
#define LCD_EN_H         P3OUT|=BIT2          //P3.2
#define LCD_EN_L         P3OUT&=~BIT2         //P3.2

/*****************************************
函数名称：Delay_1ms
功      能：延时约1ms 的时间
参      数：无
返回值   ：无
*****************************************/
void Delay_1ms(void)
{
    uchar i;
```

```
    for(i = 150;i > 0;i--)    _NOP();
}
/*****************************************
函数名称：Delay_Nms
功      能：延时 N 个1ms 的时间
参      数：n--延时长度
返回值   ：无
*****************************************/
void Delay_Nms(uint n)
{
    uint i;

    for(i = n;i > 0;i--)       Delay_1ms();
}
/*****************************************
函数名称：Write_Cmd
功      能：向液晶中写控制命令
参      数：cmd--控制命令
返回值   ：无
*****************************************/
void Write_Cmd(uchar cmd)
{
    uchar lcdtemp = 0;

    LCD_RS_L;
    LCD_RW_H;
    LCD_DataIn;
    do                                  //判忙
    {
        LCD_EN_H;
        _NOP();
        lcdtemp = LCD2MCU_Data;
        LCD_EN_L;

    }
    while(lcdtemp & 0x80);

    LCD_DataOut;
    LCD_RW_L;
    MCU2LCD_Data = cmd;
    LCD_EN_H;
    _NOP();
    LCD_EN_L;
```

```
}
/*****************************************
函数名称：Write_Data
功　　能：向液晶中写显示数据
参　　数：dat--显示数据
返回值　：无
*****************************************/
void   Write_Data(uchar dat)
{
    uchar lcdtemp = 0;

    LCD_RS_L;
    LCD_RW_H;
    LCD_DataIn;
    do                                  //判忙
    {
        LCD_EN_H;
        _NOP();
        lcdtemp = LCD2MCU_Data;
        LCD_EN_L;
    }
    while(lcdtemp & 0x80);

    LCD_DataOut;
    LCD_RS_H;
    LCD_RW_L;

    MCU2LCD_Data = dat;
    LCD_EN_H;
    _NOP();
    LCD_EN_L;
}
/*****************************************
函数名称：Ini_Lcd
功　　能：初始化液晶模块
参　　数：无
返回值　：无
*****************************************/
void Ini_Lcd(void)
{
    LCD_CMDOut;      //液晶控制端口设置为输出

    Delay_Nms(500);
    Write_Cmd(0x30);    //基本指令集
```

```
    Delay_1ms();
    Write_Cmd(0x02);     //  地址归位
    Delay_1ms();
    Write_Cmd(0x0c);     //整体显示打开,游标关闭
    Delay_1ms();
    Write_Cmd(0x01);     //清除显示
    Delay_1ms();
    Write_Cmd(0x06);     //游标右移
    Delay_1ms();
    Write_Cmd(0x80);     //设定显示的起始地址
}
/*****************************************
函数名称：Clear_LCD
功      能：清屏
参      数：无
返回值   ：无
*****************************************/
void Clear_LCD(void)
{
    Write_Cmd(0x01);
}
/*****************************************
函数名称：Set_XY
功      能：设置显示的位置
参      数：x--行
          y--列
返回值   ：无
*****************************************/
void Set_XY( uchar x, uchar y )
{
  uchar address;
  switch (x)
    {
        case 0 :   address=0x80+y;     break;
        case 1 :   address=0x90+y;     break;
        case 2 :   address=0x88+y;     break;
        case 3 :   address=0x98+y;     break;
        default:   address=0x80+y;     break;
    }
    Write_Cmd(address);
}
/*****************************************
函数名称：Disp_HZ
功      能：控制液晶显示汉字
```

```
参      数：addr--显示位置的首地址
              pt--指向显示数据的指针
              num--显示字符个数
返回值   ：无
*****************************************/
void Disp_HZ(uchar addr,const uchar * pt,uchar num)
{
    uchar i;

    Write_Cmd(addr);
    for(i = 0;i < (num*2);i++)
        Write_Data(*(pt++));
}
/*****************************************
函数名称：Disp_XY
功      能：控制液晶显示汉字
参      数：addr--显示位置的首地址
              pt--指向显示数据的指针
              num--显示字符个数
返回值   ：无
*****************************************/
void Disp_XY(uchar x,uchar y,const uchar *pt)
{
    Set_XY(x,y);
    while (*pt)
    {
      Write_Data(*pt);
      pt++;
    }
}
/*****************************************
函数名称:Clear_GDRAM
功      能:清除液晶 GDRAM 中的随机数据
参      数:无
返回值   :无
*****************************************/
void Clear_GDRAM(void)
{
    uchar i,j,k;

    Write_Cmd(0x34);            //打开扩展指令集
    i = 0x80;
    for(j = 0;j < 32;j++)
    {
```

```
            Write_Cmd(i++);
            Write_Cmd(0x80);
            for(k = 0;k < 16;k++)
            {
                Write_Data(0x00);
            }
        }
        i = 0x80;
        for(j = 0;j < 32;j++)
        {
            Write_Cmd(i++);
            Write_Cmd(0x88);
            for(k = 0;k < 16;k++)
            {
                Write_Data(0x00);
            }
        }
        Write_Cmd(0x30);              //回到基本指令集
}
/*****************************************
函数名称:Draw_PM
功      能:在整个液晶屏幕上画图
参      数:无
返回值   :无
*****************************************/
void Draw_PM(const uchar *ptr)
{
        uchar i,j,k;

        Write_Cmd(0x34);              //打开扩展指令集
        i = 0x80;
        for(j = 0;j < 32;j++)
        {
            Write_Cmd(i++);
            Write_Cmd(0x80);
            for(k = 0;k < 16;k++)
            {
                Write_Data(*ptr++);
            }
        }
        i = 0x80;
        for(j = 0;j < 32;j++)
        {
            Write_Cmd(i++);
```

```
            Write_Cmd(0x88);
            for(k = 0;k < 16;k++)
            {
                Write_Data(*ptr++);
            }
        }
        Write_Cmd(0x36);            //打开绘图显示
        Write_Cmd(0x30);            //回到基本指令集
}
/*****************************************
函数名称：Draw_TX
功      能：显示一个16*16大小的图形
参      数：Yaddr--Y 地址
            Xaddr--X 地址
            dp--指向图形数据存放地址的指针
绘图的坐标 X 以字节为单位，Y 以位为单位
初始坐标  0x80,0x80
绘图时需要将 YX 的坐标连续写入 RAM，写入期间绘图必须关闭
返回值   ：无
*****************************************/
void Draw_TX(uchar Yaddr,uchar Xaddr,const uchar * dp)
{
    uchar j;
    uchar k=0;

    Write_Cmd(0x01);   //清屏,只能清除 DDRAM
    Write_Cmd(0x34);   //使用扩展指令集，关闭绘图显示
    for(j=0;j<16;j++)
    {
        Write_Cmd(Yaddr++);          //Y 地址
    Write_Cmd(Xaddr);       //X 地址
    Write_Data(dp[k++]);
    Write_Data(dp[k++]);
    }
    Write_Cmd(0x36);   //打开绘图显示
    Write_Cmd(0x30);   //回到基本指令集模式
}
```

程序中只包含了基本的显示和画图功能，使用起来较为简单，其他的划线反白等功能也可以通过画图来实现，以后再继续研究，对应的主程序例程如下：

```
#include <msp430x14x.h>
#include "LCD12864P.h"
#include "hohai.h"
unsigned char title[]="河海大学";
unsigned char flag;
```

```
int main( void )
{
    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;

    WDTCTL = WDTPW + WDTHOLD;                   //关闭看门狗
    P6DIR |= BIT2;P6OUT |= BIT2;              //关闭电平转换

    TACTL = TASSEL_1 + MC_1;                    //计数时钟 ACLK, 增计数模式
    CCR0 = 32768 - 1;
    CCTL0 |= CCIE;                              //使能 CCR0比较中断
    _EINT();

    Ini_Lcd();                                  //显示文字
    Draw_PM(hohai);

    Draw_TX(0x81,0x81,speaker);

    flag = 0;
    while(flag < 5);                           //等待5秒钟
    Clear_LCD();                                //清屏
    flag = 0;
    Disp_XY(3,0,title);
    while(flag < 5);                           //等待5秒钟
    Clear_LCD();                                //清屏
    while(1);
}
/*****************************************
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)
{
    flag++;
}
```

MSP430学习笔记9-PS2键盘解码

PS2键盘解码的基本原理是通过外部中断读取键盘输出的串行信号，在根据扫描码进行查表解码。键盘发送往主机的信号总是在时钟的下降沿因此此中断是在下降沿触发，且时钟信号是由键盘给出，因此使用 P1口中断（已经在初始化端口时设置）。发送的数据位11位，第一位是起始位，总为0，紧接是8个数据位，然后是奇校验位，最后是停止位总为1.

本程序只能对基本按键（即键被按下时产生三个字节的扫描码的按键）做出解码，包括所有的可显示字符键和 Table，Back Space 和 Ente 三个特殊功能键。基本按键的扫描码由三个字节组成，第1个字节为接通码，第2、3字节为断开码；其中第1字节和第3字节相同，中间字节为断开标志0xf0。例如：通码和断码是以什么样的序列发送到你的计算机使得字符 G 出现在你的字处理软件里呢？因为这是一个大写字母需要发生这样的事件次序按下 Shift 键按下 G 键释放 G 键释放 Shift 键，与这些时间相关的扫描码如下：Shift 键的通码12h G 键的通码34h G 键的断码 F0h 34h Shift 键的断码 F0h 12h 因此发送到你的计算机的数据应该是12h 34h F0h 34h F0h 12h 如果按键按着不放会连续发送通码命令，可以连续显示字符（没有验证，实验验证是可以的）。

具体的说明都已经在程序中做了注释，主程序，中断服务函数中读取键盘发送的值：

```
/*****************************************************
程序功能：接收并解码来自标准键盘的基本按键的扫描码
然后在1602液晶上显示。按 Back Space 键可以前向删除显
示字符，按 Space 键可以后向删除显示字符。

-----------------------------------------------------
将拨码开关的 SN74LVC2454和 LCD 位拨至 ON
读取键盘的信号需要电平转换，注意设置 SN74LVC2454的转换方向
跳线设置：将跳线座 J13的 B8脚和 P1.7脚短接
-----------------------------------------------------
测试说明：敲定标准键盘上的按键，观察液晶显示
*****************************************************/
#include   <msp430.h>
#include "cry1602.h"
#include "cry1602.C"
#include "PS2Keyboard.h"
#include "PS2Keyboard.C"

#define SIDval   P5IN & BIT6
#define BufferSize   32          //显示缓存大小
unsigned char bitcount=11;               //位计数变量
unsigned char kb_buffer[BufferSize];      //显示缓存
unsigned char input=0;           //数据压入缓存位置指针
```

```
unsigned char output=0;          //数据弹出缓存位置指针
unsigned char pebit=0xff;       //奇偶校验标志位
unsigned char recdata=0;        //接收到的数据
unsigned char tishi[]={"this is a demo!"};

/****************主函数****************/
void main(void)
{
    uchar disptmp,i;
    uchar x = 0,y = 0;
    uchar first = 1;

    WDTCTL = WDTPW + WDTHOLD;           //关闭看门狗
    P6DIR |= BIT2;P6OUT &= ~BIT2;     //打开电平转换
    P2DIR |= BIT3;P2OUT |= BIT3;      //方向5V-->3.3V
    /*------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

    LcdReset();                          //复位液晶
    DispNchar(0,0,15,tishi);       //液晶显示提示信息
    Init_KB();                           //初始化键盘端口
    _EINT();                             //打开全局中断

    while(1)
    {
        LPM3;                            //进入低功耗模式

        if(first)
        {
```

```
    first = 0;
    LcdWriteCommand(0x01, 1);    //显示清屏
    LcdWriteCommand(0x0f, 1);    //打开游标
}

disptmp = GetChar();      //读取键值对应的 ASCII 码
if(disptmp != 0xff)       //取出了一个有效字符
{
    if(disptmp == 8) //如果是退格键
    {
        if((x == 0) && (y == 0))//如果游标在第1行第1位
        {
            x = 15;
            y = 1;
            Disp1Char(x,y,0x20); //0x20是空格的 ASCII 码
            LocateXY(x,y);
        }
        else if((x == 0) && (y == 1))//如果游标在第2行第1位
        {
            x = 15;
            y = 0;
            Disp1Char(x,y,0x20);
            LocateXY(x,y);
        }
        else
        {
            Disp1Char(--x,y,0x20);
            LocateXY(x,y);
        }
    }
    else if((disptmp == 9) || (disptmp == 13)) //如果是 Table 键或 Enter 键
    {
        _NOP();
    }
    else      //其余字符显示
    {
```

```
                Disp1Char(x++,y,disptmp);
                if(x == 16)             //如果一行显示完毕
                {
                    x = 0;
                    y ^= 1;
                    LocateXY(x,y);  //重新定位游标位置
                }
            }
        }
    }
}

/*******************************************
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

说明：键盘发送往主机的信号总是在时钟的下降沿
因此此中断是在下降沿触发，且时钟信号是由键盘
给出，因此使用 P1口中断（已经在初始化端口时
设置）。发送的数据位11位，第一位是起始位，总
为0，紧接是8个数据位，然后是奇校验位，最后是
停止位总为1.
参      数：无
返回值   ：无
*******************************************/
#pragma vector=PORT1_VECTOR
__interrupt void   PORT1_ISR(void)
{
    if(P1IFG & BIT7)                //如果是 clock 的中断
    {
        P1IFG &=~ BIT7;             //清除中断标志

        if(bitcount == 11)          //接收第1位
        {
            if(SIDval)              //起始位总为0如果是1就不是起始位
```

```
                return;             //返回
            else
                bitcount--;       //是起始位就接着接收下一位，进行计数
        }
        else if(bitcount == 2)     //接收奇偶校验位
        {
            if(SIDval)             //如果校验位等于1
                pebit = 1;         //这个程序中只是对校验位进行读取，正确与否并为做判断
            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--;
        }
    }
}
```

解码程序 PS2Keyboard.C：

```
#include <msp430x14x.h>
#include "PS2Keyboardcode.h"

#define BufferSize   32
extern uchar kb_buffer[BufferSize];
extern uchar input;
extern uchar output;
extern uchar flag;
/*****************************************
函数名称：PushBuff
功      能：将一个字符压入显示缓存，如果缓存以
          满则覆盖前面的数据
参      数：c--要显示的字符
返回值   ：无
*****************************************/
void PutChar(uchar c)
{
    kb_buffer[input] = c;
    if (input < (BufferSize-1))
        input++;
    else
        input = 0;
}
/*****************************************
函数名称：PopChar
功      能：从显示缓存中取出一个字符
参      数：无
返回值   ：取出的字符
*****************************************/
uchar GetChar(void)
{
    uchar temp;

    if(output == input)
```

```
            return 0xff;
        else
        {
            temp = kb_buffer[output];
            if(output < (BufferSize-1))
            {
                output++;
            }
            else
            {
                output = 0;
            }
            return temp;
        }
}
/*******************************************
函数名称：Init_KB
功      能：初始化与键盘相关的 IO
参      数：无
返回值   ：无
*******************************************/
void Init_KB(void)
{
    P1DIR &=~ BIT7;        //Clock 接 P1.7，设置为输入
    P5DIR &=~ BIT6;        //SID 接 P5.6，设置为输入
    P1IES |= BIT7;         //下降沿中断
    P1IFG = 0x00;          //中断标志清零
    P1IE  |= BIT7;         //使能时钟端口中断
    P1SEL = 0x00;          //P1口作为 IO 使用
}
/*******************************************
函数名称：Decode
功      能：对来自键盘的信息进行解码，转换成对
            应的 ASCII 编码并压入缓存
参      数：sc--键盘发送过来的信息
返回值   ：是否收到有效数据：0--否，1--是
```

说明　　：本程序只能对基本按键（即键被按下时产
　　生三个字节的扫描码的按键）做出解码，
　　包括所有的可显示字符键和 Table，
　　Back Space 和 Enter 三个特殊功能键。
基本按键的扫描码由三个字节组成，第1个字节为接通
码，第2、3字节为断开码；其中第1字节和第3字节相
同，中间字节为断开标志0xf0。
例如：通码和断码是以什么样的序列发送到你的计算机使得字符 G 出现在你的字处理软件里呢？因为这是一个大写字母需要发生这样的事件次序按下 Shift 键按下 G 键释放 G 键释放 Shift 键，与这些时间相关的扫描码如下：Shift 键的通码12h G 键的通码34h G 键的断码 F0h 34h Shift 键的断码 F0h 12h 因此发送到你的计算机的数据应该是12h 34h F0h 34h F0h 12h
如果按键按着不放会连续发送通码命令，可以连续显示字符（没有验证，实验验证是可以的）
**********/

```
uchar Decode(uchar sc)
{
    static uchar shift = 0; //Shift 键是否按下标志：1--按下，0--未按
    static uchar up = 0;    //键已放开标志：        1--放开，0--按下
    uchar i,flag = 0;

    if(sc == 0xf0)      //如果收到的是扫描码的第2个字节---0xf0：按键断开标志
    {
        up = 1;
        return 0;
    }
    else if(up == 1)   //如果收到的是扫描码的第3个字节
    {
    up = 0;           //收到第3字节表示按键已经放开
        if((sc == 0x12) || ( sc==0x59))     shift = 0;//shift 按下之后放开，不进行操作
        return 0;
    }

    //如果收到的是扫描码的第1个字节，第一个字节为通码
    if((sc == 0x12) || (sc == 0x59)) //如果是左右 shift 键
    {
    shift = 1;                  //设置 Shift 按下标志
```

```
            flag = 0;
        }
        else
        {
        if(shift) //对按下 Shift 的键进行解码
            {
              for(i = 0;(shifted[i][0] != sc) && shifted[i][0];i++);//查表找到对应的码值
                  if (shifted[i][0] == sc)
                  {
                        PutChar(shifted[i][1]);       //存入显示缓存
                        flag = 1;                           //解码成功标志
                  }
         }
          else   //直接对按键进行解码
          {
                for(i = 0;(unshifted[i][0] != sc) && unshifted[i][0];i++);
                  if(unshifted[i][0] == sc)
                  {
                      PutChar(unshifted[i][1]);
                      flag = 1;
                  }
          }
        }
        if(flag)    return 1;                                 //根据解码是否成功返回相应的值
        else          return 0;
}
```

需要查的表 PS2Keyboardcode.h

```
 //不按 Shift 的字符对应的编码
const unsigned char unshifted[][2] =
{
      0x0d,9,         //Table
      0x0e,'`',
      0x15,'q',
      0x16,'1',
      0x1a,'z',
      0x1b,'s',
```

0x1c,'a',

0x1d,'w',

0x1e,'2',

0x21,'c',

0x22,'x',

0x23,'d',

0x24,'e',

0x25,'4',

0x26,'3',

0x29,' ',

0x2a,'v',

0x2b,'f',

0x2c,'t',

0x2d,'r',

0x2e,'5',

0x31,'n',

0x32,'b',

0x33,'h',

0x34,'g',

0x35,'y',

0x36,'6',

0x39,',',

0x3a,'m',

0x3b,'j',

0x3c,'u',

0x3d,'7',

0x3e,'8',

0x41,',',

0x42,'k',

0x43,'i',

0x44,'o',

0x45,'0',

0x46,'9',

0x49,'.',

0x4a,'/',

0x4b,'l',

```
        0x4c,';',
        0x4d,'p',
        0x4e,'-',
        0x52,0x27,
        0x54,'[',
        0x55,'=',
        0x5a,13,        //Enter
        0x5b,']',
        0x5d,0x5c,
        0x61,'<',
        0x66,8,    //Back Space
        0x69,'1',
        0x6b,'4',
        0x6c,'7',
        0x70,'0',
        0x71,',',
        0x72,'2',
        0x73,'5',
        0x74,'6',
        0x75,'8',
        0x79,'+',
        0x7a,'3',
        0x7b,'-',
        0x7c,'*',
        0x7d,'9',
        0,0
};

//按住 Shift 后字符对应的编码
const unsigned char shifted[][2] =
{
        0x0d,9,         //Table
        0x0e,'~',
        0x15,'Q',
        0x16,'!',
        0x1a,'Z',
```

0x1b,'S',

0x1c,'A',

0x1d,'W',

0x1e,'@',

0x21,'C',

0x22,'X',

0x23,'D',

0x24,'E',

0x25,'$',

0x26,'#',

0x29,' ',

0x2a,'V',

0x2b,'F',

0x2c,'T',

0x2d,'R',

0x2e,'%',

0x31,'N',

0x32,'B',

0x33,'H',

0x34,'G',

0x35,'Y',

0x36,'^',

0x39,'L',

0x3a,'M',

0x3b,'J',

0x3c,'U',

0x3d,'&',

0x3e,'*',

0x41,'<',

0x42,'K',

0x43,'I',

0x44,'O',

0x45,')',

0x46,'(',

0x49,'>',

0x4a,'?',

```
    0x4b,'L',
    0x4c,':',
    0x4d,'P',
    0x4e,'_',
    0x52,"",
    0x54,'{',
    0x55,'+',
    0x5a,13,        //Enter
    0x5b,'}',
    0x5d,'|',
    0x61,'>',
    0x66,8,         //Back Space
    0x69,'1',
    0x6b,'4',
    0x6c,'7',
    0x70,'0',
    0x71,',',
    0x72,'2',
    0x73,'5',
    0x74,'6',
    0x75,'8',
    0x79,'+',
    0x7a,'3',
    0x7b,'-',
    0x7c,'*',
    0x7d,'9',
    0,0
};
```

MSP430学习笔记10-ADC 采集1602显示

同样是开发板中的例程，对关键的地方做了说明，程序如下：

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

程序注意点：

首先可以选择是否开启内部参考电压还是使用外部参考电压

每个通道可以独立选择参考电压

如果连接了外部参考电压应该注意关闭内部参考电压防止损坏

单片机

```
程序功能：MCU 的片内 ADC 对 P6.0端口的电压进行转换
          将模拟电压值显示在1602液晶上。
-----------------------------------------------------
拨码开关设置：将 LCD 位拨至 ON，其余位拨至 OFF
测试说明：调节电位器 W1的旋钮观察液晶显示数字变化。
*****************************************************/
#include   <msp430x14x.h>
#include   "cry1602.h"
#include   "cry1602.c"

//typedef unsigned char uchar;
//typedef unsigned int    uint;

#define     Num_of_Results     32

uchar shuzi[] = {"0123456789."};
uchar tishi[] = {"The volt is:"};

static uint results[Num_of_Results];       //保存 ADC 转换结果的数组
void Trans_val(uint Hex_Val);
/************************主函数*****************************/
void main(void)
{
    WDTCTL = WDTPW+WDTHOLD;                          //关闭看门狗

    /*下面六行程序关闭所有的 IO 口*/
    P1DIR = 0XFF;P1OUT = 0XFF;
    P2DIR = 0XFF;P2OUT = 0XFF;
    P3DIR = 0XFF;P3OUT = 0XFF;
    P4DIR = 0XFF;P4OUT = 0XFF;
    P5DIR = 0XFF;P5OUT = 0XFF;
    P6DIR = 0XFF;P6OUT = 0XFF;

    P6DIR |= BIT2;P6OUT |= BIT2;                     //关闭电平转换
    LcdReset();                                      //复位1602液晶
    DispNChar(2,0,12,tishi);                        //显示提示信息
    Disp1Char(11,1,'V');                            //显示电压单位
    P6SEL |= BIT0;                                   // 使能 ADC 通道
    ADC12CTL0 = ADC12ON+SHT0_8+MSC;                  // 打开 ADC，设置采样时间
    // 上面的配置中并没有打开内部的参考电压
    // ADC12MCTLx 用来选择通道和参考电压，这里面没有对此寄存器进行配置为默认值
    // 默认值是参考电压选择 AVCC（3.3V），通道是 A0，所以测量范围是0-3.3V
    ADC12CTL1 = SHP+CONSEQ_2;                        // 使用采样定时器
```

```
    //上面的寄存器配置采样保持触发源选择时 ADC12SC，采集信号使用采样时序电路产生的信号
    // 转换模式为单路重复转换      上面的设置必须在 ENC=0的情况下设置
    ADC12IE = BIT0;                                 // 使能 ADC 中断
    ADC12CTL0 |= ENC;                               // 使能转换
    ADC12CTL0 |= ADC12SC;                           // 开始转换
    _EINT();                                       //开启全局中断
    LPM0;
}

/*****************************************
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 uint index = 0;

    results[index++] = ADC12MEM0;                   // 将转换的结果存入数组
    if(index == Num_of_Results)                    //如果数组存满
    {
        uchar i;
        unsigned long sum = 0;

        index = 0;                                  //在从头开始存，会覆盖原有的数据
    for(i = 0; i < Num_of_Results; i++)            //计算和
    {
        sum += results[i];
    }
    sum >>= 5;                                      //除以32
    Trans_val(sum);
    }
}

/*****************************************
函数名称：Trans_val
功    能：将16进制 ADC 转换数据变换成三位10进制
          真实的模拟电压数据，并在液晶上显示
参    数：Hex_Val--16进制数据
          n--变换时的分母等于2的 n 次方
```

```
返回值  ：无
*******************************************/
void Trans_val(uint Hex_Val)
{
    unsigned long caltmp;
    uint Curr_Volt;
    uchar t1,i;
    uchar ptr[4];

    caltmp = Hex_Val;
    caltmp = (caltmp << 5) + Hex_Val;              //caltmp = Hex_Val * 33
    caltmp = (caltmp << 3) + (caltmp << 1);        //caltmp = caltmp * 10
    Curr_Volt = caltmp >> 12;                      //Curr_Volt = caltmp / 2^n
    // 参考电压为3.3V，所以计算公式应该为 Hex_val*3.3/2^n
    // 乘除计算通过移位来进行可以有效的提高程序运行效率
    ptr[0] = Curr_Volt / 100;                      //Hex->Dec 变换
    t1 = Curr_Volt - (ptr[0] * 100);
    ptr[2] = t1 / 10;
    ptr[3] = t1 - (ptr[2] * 10);
    ptr[1] = 10;                                   //shuzi 表中第10位对应符号"."
    //在液晶上显示变换后的结果
    for(i = 0;i < 4;i++)
        Disp1Char((6 + i),1,shuzi[ptr[i]]);
}
```